INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
RICE UNIVERSITY

Introduction to Computational Science: A High School Curriculum

by

Sarah Benkendorf

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE
Master of Arts

APPROVED, THESIS COMMITTEE:

[Signatures and names]

Dr. Richard Tapia, Chairman
Noah Harding Professor of Computational and Applied Mathematics

Dr. Robert Cartwright
Professor of Computer Science

Dr. Keith Cooper
Professor of Computer Science

Cynthia Lanius
Mathematics Teacher
Milby High School
Houston Independent School District
Houston, Texas

Houston, Texas
May, 1996
Introduction to Computational Science: 
A High School Curriculum

Sarah Benkendorf

Abstract
This thesis is an introductory unit to the computational sciences for the high school level. This unit will attempt to give the reader an idea of what the computational sciences consist. It will attempt to explain, through examples of early algorithms and later advancements, how and why algorithmic improvements are developed. Because of its focus toward math teachers and students, the algorithms included will be primarily mathematical in nature, but will include applications in other areas of science and industry. The algorithms included involve solving systems of linear equations, finding solutions to nonlinear equations, linear regression and linear programming. We also attempted to give some background on computer numbers and error. The goal is for the reader to come away with the knowledge of what is computational science, why is it necessary, and why is it difficult. In the end, it is hoped that as teachers become better informed in the area of computational science, more students will be encouraged to enter the field of research in mathematics. It is also hoped that teachers will be better prepared to answer questions from the students as to applications and “When am I ever going to use this.”
# Contents

Abstract  
Acknowledgments  
Preface  

1 What Is Computational Science? Why is It Important?  
1.1 Introduction .............................................. 1  
1.2 Grand Challenge Problems ........................................... 4  
1.3 Problem Solving Applications ........................................... 5  
1.4 Conclusion: The Importance of Further Study  
For Further Research ........................................... 12  

2 The Solution of Linear Equations  
2.1 Direct Computational Methods ........................................... 18  
2.1.1 Gaussian elimination ........................................... 18  
2.1.2 Gauss-Jordan elimination ........................................... 22  
2.1.3 Solving a System of Linear Equations using the Inverse ........................................... 24  
2.2 Operation Counts and a Comparison of Methods ........................................... 26  
2.2.1 Cost of Gaussian Elimination ........................................... 26  
2.2.2 Cost of Gauss-Jordan Elimination ........................................... 29  
2.2.3 Cost of Computing the Inverse of a Matrix ........................................... 30  
2.2.4 Cost of Using Cramer’s Rule ........................................... 31  
2.3 Linear Iterative Solvers ........................................... 32  
Chapter 2 Problems ........................................... 40  

3 Solutions of Nonlinear Equations  
3.1 The Method of Bisection ........................................... 45  
3.2 Newton’s Method ........................................... 48  
3.3 The Secant Method ........................................... 51  
3.4 The Method of False Position (Regula Falsi) ........................................... 54
Acknowledgments

I would like to thank Dr. Richard Tapia, the advisor, for ensuring that this document would be well written and would serve a need. I would also like to thank the members of my committee, Dr. Robert Cartwright, Dr. Keith Cooper, and Cynthia Lanius for their help and interest during the course of this work. I would also like to thank Dr. Richard Tapia, the committee chair, for ensuring that this document would be well written and would serve a need.

I would like to thank Dr. Virginia Torczon for her talk on parallel computing. Her presentation was insightful and intuitive, lending itself well to explanation at the level of this unit.

I would like to thank the faculty and staff at Rice University for freely sharing their knowledge and contacts for this unit. I would especially like to thank Dr. Robert Bixby, Dr. Steve Cox, Dr. Dan Sorenson and Dr. William Symes for their technical assistance, Dr. Anne Papakostantinou for her input on the current direction of math curriculum, Maged Hamed for his help on groundwater remediation, and Tamara Anthony for going first and giving me something to aim for.

I would like to thank Dr. Kris Stewart for her help on finding numerous web sites and contacts to gain information in this area.

Finally, I would like to thank my husband, Rod Benkendorf, for supporting my goal to return to school as well as providing hands on insight into the world of K-12 mathematics.
Preface

This book is an introduction to the computational sciences. It is intended for use by both high school teachers and students alike. For the high school students, it would be a fitting unit to add to or suplement a pre-calculus course. It also would be a fitting unit for use in an elective course. Students with experience up to and including Algebra II and Pre-calculus should be able to comprehend the material following. This material does assume some prior knowledge of matrices, although some review is provided. We begin with a chapter explaining what computational science is and why we have a need for this subject. It gives a general idea of the broad area and the vital importance of computational science. Next, we discuss various algorithms and the improvements to them that have come about as a result of computational science. We allow the students to become familiar with methods for solving systems of linear equations such as Gaussian elimination and the conjugate gradient method. We give the students an understanding of the importance of finding solutions to nonlinear equations with some explanation as to why one method is better than another. We discuss the purpose and background of linear regression, with least squares regression receiving the most attention. We also introduce the student to a more in-depth look at the area of linear programming. Finally, we introduce the student to some computational or computer aspects of computational science - that is, the areas of computer numbers and errors and the field of parallel computing. While some of these topics may be touched on minimally in some high school courses, rarely are they given the attention they deserve, nor does the student often come away with the feeling that there are vast areas of mathematics that still need to be explored. This unit was developed with the goal in mind of introducing students and teachers alike to the idea that the real world is not made up of text-book style problems that can easily and quickly be solved with “nice answers.” It is our goal to convince the reader that computational science is a viable, interesting area of research waiting to be explored.

Until recently, the area of computational science did not exist. The courses at the college and university level that could fit into this area were included in computer
science departments, applied math departments, engineering departments or statistics departments. Recently, however, universities have begun to recognize computational science as a field of its own and therefore have begun to develop departments for its study. As part of the problems at the end of chapter 1, the students will be given the opportunity to explore the possibilities for further study in the area of computational science. While much of the work is being done at the graduate school level, some colleges and universities recognize the need for introducing the field at the undergraduate level and are developing curriculum to fit this need. It is the belief of others that even this is not going far enough - that the students need to be made aware of this field as they are making their choices for a career, a process that starts in high school.

In *The Psychology of Learning Mathematics*, Richard Skemp states “...the learning of mathematics, especially in its early stages and for the average student, [is] very dependent on good teaching” (1987, p. 21). However, most teachers are not familiar with the area of computational science, so it would be difficult for them to explain it to the student. Therefore, we have attempted to write this unit with both the teacher and the student in mind. The writing is done at slightly higher than the high school level with the purpose of attracting the interest of the more precocious student and teacher. The explanations are detailed in order to keep the interest of the average student. It is our hope that this unit is detailed enough in order to provide the teacher with enough background to feel comfortable helping the student. Solutions to most problems have been provided at the end of the text.

Since the field of computational science is beginning to have an important role in the world, it is important that the foundation is taught correctly and well. It is very much the goal of this unit not only to teach for the understanding of basic procedures, but also how this understanding can lead to further developments and understanding. It is hoped that this unit will not only help students understand the topics presented in the unit, but also gain an understanding and possibly even an interest in the broad field of computational science.
Chapter 1

What Is Computational Science? Why is It Important?

1.1 Introduction

Many cultures have contributed to the development of mathematics. The ancient Egyptians, Babylonians, Romans, Mayans and Chinese all had their own numbering system to use for enumeration. Although the Mayan civilization was destroyed, the remains suggest a remarkable system of numeration in base 20 with a symbol to represent 0. (The other ancient numbering systems did not include this.) They were also believed to have a great knowledge of astronomy - a science that has led to many mathematical hypotheses and theories. The abacus was used as a device for calculation in various forms by the Chinese and Japanese and other countries in Europe, the Middle East and Asia as early as about 400 A.D. The ancient Greeks, with their learned mathematicians such as Euclid, Archimedes, Aristotle, and others, developed a mathematics that was far superior to its predecessors. However, mathematics as we know it still had a long way to go. Although the mathematical theory of the ancient Greeks was limited, for nearly 2000 years, until the 16th century, the views of Aristotle and the ancient Greeks formed the basis for all the branches of science.

Since the 16th century, the study of mathematics has made great strides, both for the study of mathematical theory and in the application of mathematics to everyday physical phenomenon. Many milestones have been set over the years that have been building blocks for study even today, both in mathematics as it applies to the physical world and mathematics in theory. Galileo, who was known as the father of dynamics, the branch of science that deals with the action of force in producing motion, laid the foundation for scientific method. Although he was not a mathematician and he is known to have made incorrect conjectures, his genius and creativity must be appreciated despite his lack of mathematical rigor. Archimedes' work in the science of balance may have enabled Newton to solve the problem of the motion of the planets and develop the law of universal gravitation. In order to explain the phenomenon of
the motion of a particle, Newton and Leibniz each individually worked on what we now consider the beginnings of differential calculus. Isaac Newton, Joseph Raphson, and Thomas Simpson, among others, worked on developing methods for finding the solutions to nonlinear equations in the late 1600's. Mathematics had become an effective tool for pursuing mechanics, physics, and other sciences. Early mathematicians such as Gauss made little distinction between theory and applications - they did both. In fact, the application often drove the development of the theory. The physical phenomena were studied and then, through the use of formulas and mathematical relations, rules and laws were developed to explain these phenomenon. Mathematics had become the manner in which science expressed, formulated, continued and communicated itself. It is amazing that the abstract concepts of mathematics, themselves so removed from reality, can be used to model so effectively and explain the world around us.

What, you may ask, does this have to do with computational science? As you have seen, mathematics has been a progressive science. It was not a finished, complete bank of information at the time of the ancient Egyptians that was simply passed on from one generation to the next. Instead, mathematics was developed over the centuries. Previous theories and study were used to develop better theories and describe more problems or phenomena. These laws have been used to model the world so well that we often fail to distinguish between the mathematical model and the real world situation that it describes. Just as Newton used Archimedes' work to develop his theory of the motion of the planets, mathematicians today are constantly improving mathematical methods, formulas, and theories, but now we have the added advantage of computer technology. The computer has made a fundamental change in the way we use mathematics to solve problems.

Technology is changing the way we view and what we do in mathematics. We are still formulating models for real life and real world phenomenon and problems, just as Newton and many others did many years ago, but now, with the aid of the computer, our models can encompass a broader field. Just as the field of science can be divided into biology, chemistry, astronomy, genetics, physics and many others, so also is the field of math involved in a variety of areas. Mathematicians and scientists do so much more together today in the areas of astronomy, chemistry, and physics, but have now been able to work on modeling problems in the areas of biology and virology, electro-magnetics, electronics, engineering, fluid mechanics, geometric modeling, materials
science, signal and imaging processing, structures technology and weather and ocean modeling as well.

Mathematics is a mixture of methods used to solve problems. As technology improves and mathematics advance, these methods are replaced with methods that can deal with larger problems or that may be applied to a more general problem rather than a very specific one - for instance, developing methods that can be applied to the study of a class of proteins rather than a single protein. Therefore, there will never be an end to the work that a mathematician can do. We will be constantly improving and advancing what we already know. It is in this spirit that the area of computational science has developed.

The field of Computational Science had its beginnings in the early 1940's. In the midst of World War II, scientists were working on developing the computer. With the development of the computer, new techniques were developed to experiment with various problems and methods that would utilize the advancing computer technology. Leaders in this new class of problems included Dantzig in the field of linear programming and Lanczos, Stiefel, and Hestenes in the field of alternative methods to direct methods for linear systems of equations. Ever since that time, we have been working to improve and speed up the methods that can be used on the computer as well as expanding the field of problems that the computer can be used with. The size of the problems that can now be solved is much greater than ever before.

In this unit we will attempt to tell you about the size and scope of the field of computational science. It is important for you to know what type of problems could be included in the area of computational science and why. As you have been told and will see, the field of computational science covers many subject areas. We will also attempt to show you some considerations that constantly concern computational scientists, such as the capabilities of their computers. Also, we will attempt to show you how the field of computational science is continuing to further the study of mathematics and science, just as the early mathematicians did. We can do this by looking at the development of new algorithms for previous problems that have become much larger. You cannot possibly hope to learn everything about the area of computational science in this small unit, the field is simply too large. The goal is to get you as the student to have a feel for how large and important the field and the problems in this field actually are. If you know this, along with the fact that because of their size, some problems may be too large to work with or find a solution for, then you will have gained considerable understanding of the field of computational science.
1.2 Grand Challenge Problems

In 1991, a class of projects were developed that utilize the support from high performance computing centers, or HPCC's. This class of problems was designated Grand Challenge Problems by Ken Wilson, a Nobel prize winning physicist from Cornell. These problems were so named by Wilson in his campaign to get federal support for high performance computing in science. Problems were included in this class because they required extensive computational techniques as well as being critical to the national interest. Some of the current projects being worked on deal with climate (or weather) modeling, convective turbulence and mixing in astrophysics, computational biology, geophysical databases, condensed matter physics, and binary black holes. They also deal with flow modeling, quantum chromodynamics, ground water remediation, and contaminant containment and many others. You may or may not know what each of these problems deals with, or what most of the terms mean, but you can see that the grand challenge problems cover a variety of areas of science and are as important for their practical results as well as their theoretical results.

Research and development in grand challenge areas serves to increase knowledge while serving the nation's economic, security, and educational needs as well as the global environment. All of these problems involve computer simulation - detailed modeling of the situation or problem both graphically and mathematically with the aid of the computer. Due to the advances in computational science, researchers can now recreate numerically, or simulate, natural phenomenon on high performance computers. This is a huge breakthrough because, previously, many natural phenomena could not be investigated in the laboratory due to safety constraints, cost, speed, or time. These problems also may involve extensive data processing. The amount of data required for these problems, may be so massive that the larger memory space of a high performance computer may be required. There are several issues involved in increasing the capabilities of computers. Different algorithms can be more effective as far as memory allocation and speed. Researchers are always working to improve current algorithms and to write newer, better algorithms. Also, the innovation of parallel computation, or breaking a task into smaller independent tasks and sending them to different processors, has come to the forefront in computer research and with it, several other questions are being raised. Consider the following problems:

The United States currently faces a very large ground water contamination problem. Although the total number of contaminated sites is unknown, estimates of
the total number of waste sites where ground water and soil may be contaminated range from approximately 300,000 to 400,000. Recent estimates of the total costs of cleaning up these sites over the next 30 years range as high as one trillion dollars. Currently, 95% of the drinking water in rural areas and 75% of all water used comes from ground water sources. This makes groundwater remediation a necessary and important problem.

The crew scheduling problem has been studied continually for the past 40 years. Almost all of the major U.S. airline companies, as well as many non-U.S. companies, have used a common mathematical modeling technique for assigning crews to flights. The goal is to minimize crew costs while conforming to the many constraints imposed by the governmental and labor work rules. This problem is extremely important since personnel costs of the major U.S. carriers now often exceed $1.3 billion per year. They are the second largest part, next to fuel costs, of the total operating costs of major U.S. carriers. For this problem, the computational scientist begins with the airline flight schedule - departure and arrival locations and times - and the equipment type for each flight segment, or nonstop flight between cities. All of the possible crew rotations that conform to FAA regulations, union requirements, and company restrictions are then formulated. A cost is assigned to each rotation based on all of the costs associated with the rotation - length of time that a crew is away from home, per diem, lodging expenses, etc. Given this set of rotations, one attempts to find the “best” collection of rotations so that each flight is covered by exactly one crew rotation.

These are examples of the types of problems that can be classified as Grand Challenge Problems or computational science problems. Informally, computational science has been defined as the description of those activities in science and engineering disciplines that depend on computing as their main tool. As the above examples show, the area of computational science is not just used to further the area of mathematics, but to improve the quality of life for society as well.

1.3 Problem Solving Applications

More than ever before, the area of computational science relies heavily on the use of a logical and complete approach to the problem. In the development of computational science, scientists have not simply been given a real world situation and automatically sat down at the computer and solved it. Most real life situations are much more com-
plex than they appear initially. Problems are not solved with a haphazard approach, but rather with a logical sequence of steps in order to completely solve the correct problem. These steps may be familiar, as they are useful steps for solving any type, size, or complexity of problem. The steps are as follows:

1. Identify an appropriate real-world phenomenon
2. Construct a mathematical model (mathematics problem)
3. Design a numerical algorithm
4. Build a computer code that implements the numerical algorithm
5. Perform numerical experimentation (simulation) with the intention of
   (a) evaluating (validating) and revising (if indicated) the model.
   (b) generating mathematical conjecture and subsequent new theory concerning numerical and theoretical properties of both the model and the algorithm.

Now let’s take a look at what each of these steps involves:

1.) **Identify an appropriate real-world phenomenon:** write down what’s known and unknown. Identify what it is that you are trying to solve. If you can’t clearly identify the problem, you cannot know whether you are finding the correct information. It is also very important to know what you need the answer to be. If you aren’t sure what you are looking for, setting up a mathematical model will be extremely difficult.

2.) **Construct a mathematical model (mathematics problem):** while the computational sciences are not only math and computer science, both subjects are deeply embedded in the problems considered by computational scientists. Problems in computational science involve some form of a mathematical model - usually a rather large system of differential or algebraic equations and/or inequalities with a large number of variables. It is extremely important to translate the problem into the correct mathematical terms. If we seek to simplify our assumptions too much, it is possible to overlook seemingly minor details. These details could lead to an analysis of a completely different problem. However, we do wish to leave out those details that do not affect the problem. It is a process that can be extremely difficult at times, yet that, with practice, can be less intimidating.
3.) **Design a numerical algorithm:** Once you have the mathematical model designed, you must develop a series of steps that you will use to solve this model or extract the necessary pieces of information. This could involve solving a system of linear algebraic equations or any of a large number of more complicated processes.

4.) **Build a computer code that implements the numerical algorithm:** Up until this point, most of the problems that you have worked are straightforward and small, involving 3 or 4 variables. Many real world problems are much larger and more complex, involving possibly millions of variables. Problems of this size cannot be solved by hand calculation. Therefore we must enlist the aid of the computer. At this step, we must translate the algorithm that we developed to solve this problem into a computer code that will perform those steps.

5.) **Perform numerical experimentation (simulation):** At this point in the process, extensive testing of our computer code and our mathematical algorithm must be performed to ensure that our algorithm will correctly give us the information that we need. Any simplifying assumptions that were made must now be reanalyzed and understood. Have you answered the question that you set out to answer? Are there any changes that would invalidate any of your simplifying assumptions? Did you use all of the correct information? Was your method correct? Normally, the solution that is obtained is only an approximation due to inaccuracies in the model and inexact calculations. Even if current theory says that an answer is available, a specific answer should be computed. In these cases, the researcher will determine if an approximation is acceptable or if other methods need to be tried. In many cases, approximations provide useful insight into the errors of certain methods or the nature of certain types of problems. For many problems, we can show that computing the precise answer is either impossible or prohibitively expensive.

Each phase in this process is very important. Small mistakes at any stage in the process could lead to the wrong answer. Each step must be carefully executed. Each problem is unique and each step in the problem solving process is important to getting an answer to the problem. If you have done all of the work involved in finding a solution to a problem correctly and the answer to the problem is still not usable, you have still gained the information that this method will not satisfy the
situation at hand and that you must try another method. This in and of itself is important for further study - much knowledge that we have today has been found by not only finding methods that do work, but by testing ones that do not. However, if you approach a problem in a haphazard manner, you can not be sure that it is your method or solution that is faulty rather than some of the work along the way.

Let's look again at the problem of groundwater remediation in order to see how the problem solving steps are applied. As you will see, there are many factors involved in understanding the problem. There are several considerations for modeling the problem. Not only are we able to create a mathematical model for the problem, but through advances in the computer, we can also create a physical model or simulation of the problem. In the translation of the conclusions back into the original context of the problem, the scientist will look at how the results can be applied not only to the exact problem being studied, but what they can tell us about the general class of problems related to the groundwater remediation problems.

In approaching this problem, there are several factors important in understanding the problem. In the recognition of the problem, the problem of groundwater contamination has several features. The scientist must understand the physics, chemistry, geology and hydrology of the system. They are concerned with how the contaminant reacts with the surface. Each of the above properties of the contaminant and the area contaminated yields a mathematical equation or set of equations that are important in the problem. Thus, the problem becomes very complex in nature. There are complex interactions between the chemical and the porous media (the ground, sand, water, or other substance that the contaminant has seeped into.) The porous media itself can be and usually is very complex in nature. Another factor that adds complexity to the model is the difficulty of converting from a microscopic (or a laboratory sized) model up to a macroscopic or realistic sized model. There is generally not an easy conversion from one scale to another, and some of the behaviors may have altered from one model to the next. These issues all yield a problem that cannot easily be solved with simple, algebraic methods. The problem cannot be solved without adding restrictive assumptions that make the problem less general, and therefore not as applicable to as many situations.

We have now discussed some of the complexities of the problem. We have worked to gain an understanding of the problem as we know it. Now we must build the model in order to work with the complex nature. This will lead to differential equations which can be approximated by a large system of algebraic equations. In a very
simple, two-dimensional case, suppose that a contaminated area is broken down into a sequence of one hundred cells. This gives a system of 100 equations. Covering the problem over a period of 300 days with a time step (interval of time between measurements) of 3 days yields 100 time intervals. This means then that we have a system of 100 equations that must be solved 100 times. We would find that our equations involved some terms that are derivatives, which you will begin to discuss in calculus, as well as terms that are exponential, or not linear. Therefore, we cannot simply solve this system with normal methods that we already know for solving systems of linear equations. Scientists must use iterative (repetitive) methods to solve this system. (We will discuss one of these iterative methods later.) Moving up to a more realistic, three-dimensional model, even a small model can yield a system of thousands of equations.

After the mathematical model has been built, the scientist will use this model to draw some conclusions. One of the goals for the groundwater remediation problems is to find the number of well locations as well as the injection rates used to clean up the site. However, they are not simply looking for an answer but rather for the best or optimal answer. The scientists may wish to optimize concerning a number of factors. They may wish to remediate the largest area of contamination at the least cost, causing the smallest problem for the people in the area. There are several advantages to the computational approach to this problem over a more trial and error approach. The computational approach allows for a rigorous formulation of the problem. It yields a good understanding of the decision variables, the constraints - or functions that put limits on our answer - in the problem, and the objective function - or the function that we want to find the best solution for. It also gives a rigorous way of testing what some scientists call the "goodness" of the result. They can get a good idea of how one solution compares to another in terms of performance, operation costs, operation time, etc. Because of the complex nature of the problem, there is not an absolute answer to the problem. As the scientists work to optimize the problem, they may "play" with a combination of factors that affect the problem in order to see how the solution is changed and what yields the "best" solution.

As important as the mathematical optimization of the problem is the physical simulation of the problem. At the outset of the problem, some important questions arise. How did contaminants get into the subsurface? Where would the contaminant go if left alone or flushed out? How can it be cleaned up? One method of testing the subsurface is to dig a large number of holes in the contaminated areas and perform
tests on the samples pulled from these holes. This has the disadvantage of being very labor and cost intensive. Another method involves building a physical model in the laboratory and testing for a number of results. This again leads to the problem of scaling the model up to the actual size. A third method involves placing a grid over the site map and with the help of a few soil or surface samples, running a variety of computer simulations - or testing the "what ifs". This can give a variety of solutions for a variety of situations. Scientists can find the worst case scenario, the best case, and possibly somewhere in between, the scenario that fits the problem at hand. All of this can be found with a enormous savings in time and money.

As you can see with this example, the work involved in solving this problem is by no means simple or trivial. This is not a problem that can be formulated or solved in a short period or time or effort. It is a problem whose solution and formulation can give hints and suggestions for solving other similar problems. As often as possible, computational scientists will work to keep the formulation of a problem as general as possible, with a minimum number of restrictive assumptions. This is what allows them to move or apply some or all of their knowledge to other problems.

1.4 Conclusion: The Importance of Further Study

Because of the amount of data involved in real world problems and the complexity of the problems, the difficulty in working with these problems is not only in formulating and solving them, but also in improving the time taken to solve them. For example, suppose we are given a problem which, when formulated, yields a system of linear equations that has 100 variables. Suppose also that the this problem requires that a solution be computed 100 times. If solving this system once requires one and a half hours to solve, it would of course take 150 hours to complete this problem. Then suppose that the equation instead must be solved 20 million times. Now completing this problem will take 30 million hours. This is time that we simply don't have. Unless we wish progress to come to a screeching halt, then we must constantly be working to find ways to decrease the amount of time that we are taking to solve these problems.

There are several important areas that, as improvements are made, decrease the amount of time taken to work a problem. Right now, advances are being made in the technology of parallel computations which allow the programmer to distribute the work being done over a number of processors so that a number of tasks can be
done simultaneously. Scientists are also working to better understand the algorithms that we currently use. Sometimes, through some minor changes in the algorithm, a computer can be made to solve a problem more accurately or more quickly than with the original algorithm. Other advances lead to entirely new algorithms that may work better for larger problems or may be more suitable for the problem at hand. The following chapters will attempt to give you an idea of why and how algorithmic improvements come about.
For Further Research

This chapter can by no means tell you about all of the areas of Computational Science. To learn more about it, or to delve further into some of the topics, you may wish to investigate some of the following internet resources. We have listed some of the highlights - to get a more extensive list, try searching for Computational Science on your own.

1.) http://www.cs.uoregon.edu/~conery/resources.html This document is a collection of starting points for searching the web for resources related to Computational Sciences and Engineering.

2.) http://www.computer.org/pubs/cs&e/cs&e.html This document is an electronic publication of the magazine Computational Science and Engineering. This magazine, published quarterly, has many interesting articles about computational science in general as well as listing projects and problems in specific scientific disciplines as they relate to computational science.

3.) http://k12colostate.edu/textbook/toc.text.html This document is a high school computational science text book, focusing more on the research and modeling end of computational science. It delves more in depth into some of the computing issues as well.

4.) http://cesp2.phy.ornl.gov/seep.old/textalk.html This document includes a survey of computational science tools as well as some more explanation on some computational methods. Although it is aimed at the undergraduate, the first section especially may be interesting reading for the high school student and teacher.

5.) http://compsci.cas.vanderbilt.edu/cesp/TEXTOC.html This document is another undergraduate computational science text book. In this text, the mathematics sections are classified according to what mathematical background is necessary for various sections.

6.) http://www.mbhs.edu/mvhspmj/resources.html This document describes the project of one virtual high school. They provide links to various other web sites on more information as well as projects and research done by high school students in the areas of computational science.

7.) http://www.cray.com/PUBLIC/HPC/ind/univ/Comp_Sci_Paper.html and http://www.cz3.nus.sg:8100/guidebook/contents/html. These two documents list and describe some of the various academic programs and degrees offered in the com-
putational sciences at the college level. If the field of computational science is of interest to you, this may help you get a start on finding the college of choice.

**Teachers**

The following URL's are locations of some programs aimed at introducing computational science at the high school level. Some are summer programs and some describe curriculum already in place at the high school level.

10.) [http://goober.mbhs.edu/mvhsproj/bground.html](http://goober.mbhs.edu/mvhsproj/bground.html)
Chapter 2

The Solution of Linear Equations

The fundamental computational task performed by a computer is the solution of square, nonsingular systems of linear equations. By square, we mean a system of linear equations where the number of equations is equal to the number of variables or unknowns, and by nonsingular, we mean that this system has a unique solution. With the exception of data manipulation and comparisons, computers can only do arithmetic operations - addition, subtraction, multiplication, and division. Therefore, the only type of system of equations that can be solved in a straightforward manner are square, nonsingular systems of linear equations. Here is where the computational scientist enters the picture. The task of the computational scientist, when given a particular mathematical problem, is to construct an algorithm that will approximate the solution of the given problem by solving a sequence of square systems of linear equations. It is extremely important that the method be both efficient and accurate.

In general, a square system of linear equations is represented as

\[ a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \]
\[ \vdots \]
\[ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \]

where \( a_{ij} \) represents the coefficient for the variable \( x_j \) in the \( i^{th} \) equation. With any single equation, you learned early in algebra that multiplying every element in that equation by a nonzero constant did not change the solution set to the equation - that is the set of points satisfying the equations. You also may have learned with systems of equations that you can add any two equations together without altering the solutions to that system. In fact, in practice, you often multiplied one equation by a non-zero constant and added it to another equation, all in the process of finding the solutions to that system of equations. You also learned that you would arrive at the same solutions no matter which equation was written first. None of these operations ever changed the solutions of the original problem.
Matrix notation can be used to write the system of linear equations more compactly. A matrix is a rectangular array of numbers consisting of rows and columns of numbers. Some matrices may have one row and many columns or one column and many rows. We will call these special cases row vectors and column vectors, respectively. Another special case matrix is the matrix that has the same number of rows as columns. This is called a square matrix. There are three operations that we are especially interested in when solving systems of linear equations. They are called elementary row operations and consist of

- multiplying any row by a nonzero constant,
- adding any two rows together, or
- interchanging any two rows.

As you may have noticed, these steps have a direct counterpart for a system of linear equation. The general square system of linear equations (2.1) can be written in matrix notation as

\[ \mathbf{A} \mathbf{x} = \mathbf{b} \]

where \( \mathbf{A} \) is the \( n \times n \) matrix whose elements are the coefficients, \( a_{ij} \), of the system of linear equations, \( \mathbf{x} \) is the \( n \)-vector \( [x_1, x_2, \ldots, x_n]^T \), and \( \mathbf{b} \) is the \( n \)-vector \( [b_1, b_2, \ldots, b_n]^T \). [Note: \( n \) is regularly used to represent the number of variables in a system of equations.] Just as in the system of equations, the elementary row operations produce different matrices but leave the solution to the system of linear equations unchanged. Using the elementary row operations, we will transform our matrix, \( \mathbf{A} \), into a matrix with nonzero elements in the upper triangle of the matrix, or in some cases, only on the diagonal of our square matrix. Matrices of this type are much easier to use to solve for the values of our unknowns.

An even more compact, common way of representing a system of linear equations is with the use of augmented matrices. The augmented matrix combines the coefficient matrix \( \mathbf{A} \) and the vector \( \mathbf{b} \) containing the right hand side into one larger matrix, \([\mathbf{A}\mathbf{b}]\). Our system of equations (2.1) can be represented by the augmented matrix

\[
\begin{bmatrix}
a_{11} & \cdots & a_{1n} & b_1 \\
\vdots & \ddots & \vdots & \vdots \\
a_{n1} & \cdots & a_{nn} & b_n
\end{bmatrix}
\]
In this matrix, you see the vertical line separating the coefficient matrix, A, and the right hand side. This is only used to keep our representation clear; it obviously would not be included when storing this matrix in a computer. The application of any of our elementary row operations to the augmented matrix would not change the solution to the system of linear equations that it represented because it would amount to doing the same operation to the left and right hand side of each equation. Therefore, we arrive at the same solution with or without augmented matrices, and the augmented matrix is very convenient.

In your previous courses, you should have already learned one method of solving a system of linear equations - that is, the substitution method. The substitution method works well for learning the properties of systems of equations and is easy to use for small systems of linear equations, but it is not a method that is easily used for larger systems of linear equations. It is by no means the only method or the best method to use for solving systems of linear equations. The main goal of this section is to show you the significant differences among various methods for solving systems of linear equations in terms of expense - computer work. This is important when the systems are large and may even make the difference between success and failure.

Another method of solving a system of linear equations is to use the so-called Cramer’s Rule. Cramer’s rule makes use of the determinant (|A|) to find the solution to a system of linear equations. The determinant of the matrix, A, is defined as the sum of all possible products formed by taking one element from each row in order, starting with the first row, and one element from each column where the sign of the product depends on a permutation of the column. Let us look at a 3 × 3 matrix to see what this means.

\[
A = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  a_{31} & a_{32} & a_{33}
\end{bmatrix}
\]

The definition says that we want a product formed by taking one element from each row and one element from each column. In this case, let us start with \(a_{11}\). We now have an element from the first row and need one from the other two rows. However, we already have an element from the first column as well, so we can only use \(a_{22}\) and \(a_{23}\) as our choices from the second row. If we chose \(a_{22}\) this leaves as our only choice from the third row \(a_{33}\). So our product is \(a_{11} \cdot a_{22} \cdot a_{33}\). Another possible product is \(a_{12} \cdot a_{23} \cdot a_{31}\). There are 3! or 3 · 2 · 1 of these products. With a 4 × 4 matrix, there are 4! products, and with an \(n \times n\) matrix, there are \(n!\) possible products. As we said, the
sign of the product is determined by the permutation or inversion of the columns. An inversion refers to column indices that are out of order. Consider our first product. The order of the columns (the second index on $a_{ij}$) was $(1,2,3)$. In this case there were no inversions. However, on our second product, the order of our columns was $(2,3,1)$. In this case there are two inversions - 2 coming before 1 and 3 coming before 1. If our columns were in the order $(2,1,3)$ or $(1,3,2)$, we would only have one inversion in each of these instances. If the number of inversions of the column is odd, the sign attached to the product is negative, and if the number of inversions is even, the sign attached to the product is positive. This determines whether the product will be added or subtracted. It is truly beautiful and amazing that this very abstract notion of determinant has several practical, useful interpretations or characterizations.

Calculating determinants for $2 \times 2$ or even $3 \times 3$ matrices is fairly easy, but as you can see even with a small, $4 \times 4$ matrix, this method would be time consuming. Finding all of the $4!$ or 24 different products would take a bit of time at best. There are other methods of computing the determinant, but we will leave those to a linear algebra course. As you will see in the next example, computing the determinant of a $2 \times 2$ matrix is quite simple.

As a quick overview of Cramer’s rule, consider the following small system of equations.

\begin{align*}
ax_1 + bx_2 &= c \\
dx_1 + ex_2 &= f
\end{align*}

(2.2)

For this particular system of equations, Cramer’s rule leads to:

\[
x_1 = \frac{\begin{vmatrix} c & b \\ f & e \\ a & b \\ d & e \end{vmatrix}}{\begin{vmatrix} a & c \\ d & f \\ a & b \\ d & e \end{vmatrix}} \quad \text{or} \quad x_1 = \frac{ce - fb}{ae - db} \quad \text{and} \quad x_2 = \frac{\begin{vmatrix} a & c \\ d & f \\ a & b \\ d & e \end{vmatrix}}{\begin{vmatrix} a & c \\ d & f \\ a & b \\ d & e \end{vmatrix}}\quad \text{or} \quad x_2 = \frac{af - dc}{ae - db}
\]

As you may have noticed, in order to find the solution for $x_1$, the matrix in the numerator has replaced the coefficients for $x_1$ with the right hand side. To find the value for $x_2$, the matrix in the numerator of this fraction has replaced the coefficients for $x_2$ with the right hand side. In both fractions, the matrix in the denominator is simply the coefficient matrix. In general, to use Cramer’s rule to compute the solution of a system of equations in $n$ variables would require finding the determinant
of \((n + 1) \times n\) matrices - one for the denominator and one for each of the \(n\) different variables or unknowns, making \((n + 1)!\) multiplications. This can be an extremely large number for even moderate \(n\). If we apply the definition of the determinant to these \(2 \times 2\) matrices, we find that we only have two distinct products (in the first determinant, \(ce\) and \(fb\)), making Cramer's rule quick and efficient for this problem.

As the systems of linear equations get larger, the determinants used by Cramer's rule get larger and and the process more complicated. As we will discuss later in the chapter, computer work is an important issue in using a particular algorithm. As finding the determinants becomes more complicated, the amount of work increases. Because of this, Cramer's rule quickly becomes too expensive and therefore is never used as a tool for finding the solution of a system of linear equations. Cramer's rule does work well for small problems as does the substitution method, and like the substitution method, Cramer's rule is very useful in that it can be used to produce some very nice theoretical results for systems of linear equations. So what does work? What are computational scientists looking for?

2.1 Direct Computational Methods

2.1.1 Gaussian elimination

Gaussian elimination is considered the workhorse of the computational sciences for the direct solution of the system of linear equations (2.1). Therefore we will now discuss it in some detail. Karl Friedrich Gauss, the great 19th century mathematician, suggested the elimination method as a constructive mathematical proof that square, nonsingular, systems of linear equations have a unique solution. His constructive proof has also had great numerical consequences.

Gaussian elimination is a systematic application of elementary row operations to the system of linear equations in order to get the system of linear equations in upper triangular form. Upper triangular form for a matrix means that all of the elements below the diagonal elements (the first element in the first column, the second element in the second column, and so on) are zeros. While Gauss originally developed his proof using the notation of (2.1), we will use the more compact notation of the augmented matrices. We will illustrate this method with a system of four linear equations in four variables.
Consider the following system of equations. If we rewrite the system in matrix notation, we get the matrix equation on the right.

\[
\begin{align*}
-4x_2 + 2x_3 + 2x_4 &= -6 \\
6x_1 - 2x_2 + 2x_3 + 4x_4 &= -6 \\
3x_1 - 13x_2 + 9x_3 + 3x_4 &= -19 \\
-6x_1 + 4x_2 + x_3 - 18x_4 &= -34
\end{align*}
\]

\[
\begin{bmatrix}
0 & -4 & 2 & 2 \\
6 & -2 & 2 & 4 \\
3 & -13 & 9 & 3 \\
-6 & 4 & 1 & -18
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{bmatrix}
= 
\begin{bmatrix}
-6 \\
16 \\
-19 \\
-34
\end{bmatrix}
\]

As we said earlier, with Gaussian elimination, we wish to use our elementary row operations to reduce this matrix to upper triangular form, getting zeros below the diagonal. As you may have noticed, our first equation has a zero as the diagonal element. However, if we switch row 1 and row 2, we now have a -4 as the diagonal element for that row and we have our first zero below the diagonal.

\[
\begin{align*}
&\text{the augmented matrix for this system} & & \text{switching the first and second rows} \\
&\begin{bmatrix}
0 & -4 & 2 & 2 & -6 \\
6 & -2 & 2 & 4 & 16 \\
3 & -13 & 9 & 3 & -19 \\
-6 & 4 & 1 & -18 & -34
\end{bmatrix} & & \begin{bmatrix}
6 & -2 & 2 & 4 & 16 \\
0 & -4 & 2 & 2 & -6 \\
3 & -13 & 9 & 3 & -19 \\
-6 & 4 & 1 & -18 & -34
\end{bmatrix}
\end{align*}
\]

In our next step we will subtract multiples of the first row from the third and fourth rows in order to introduce 0’s in the first column below the diagonal. We will be adding \( \frac{3}{2} \) (the multiplier is the quotient \( 3 \div 6 \)) times the first row to the third row and simply adding the first row to the fourth row. The result is

\[
\begin{align*}
&\begin{bmatrix}
6 & -2 & 2 & 4 & 16 \\
0 & -4 & 2 & 2 & -6 \\
0 & -12 & 8 & 1 & -27 \\
0 & 2 & 3 & -14 & -18
\end{bmatrix} & & r_3 + \frac{3}{2} \cdot r_1 \\
&\begin{bmatrix}
6 & -2 & 2 & 4 & 16 \\
0 & -4 & 2 & 2 & -6 \\
0 & -12 & 8 & 1 & -27 \\
0 & 2 & 3 & -14 & -18
\end{bmatrix} & & r_4 + r_1
\end{align*}
\]

Notice that the first row remained unchanged during this process although it was used to create the zero entries. In our next step, we will be “ignoring” the first equation and first column of 0’s since we have completely eliminated all of the terms below the diagonal. Now we wish to add multiples of the second row to the third and fourth row in order to introduce the 0’s in the second column. This time we will add -3 times the second row to the third row (again, this comes from the quotient \( 12 \div 4 \))
and we will add $\frac{1}{2}$ times the second row to the fourth, giving us the following matrix.

\[
\begin{bmatrix}
6 & -2 & 2 & 4 & 16 \\
0 & -4 & 2 & 2 & -6 \\
0 & 0 & 2 & -5 & -9 \\
0 & 0 & 4 & -13 & -21
\end{bmatrix}
\]

\(r_3 - 3 \cdot r_2\)

\(r_4 + \frac{1}{2} \cdot r_2\)

You will notice that this time, the first and second row remained unchanged. Our final task is to introduce a 0 into the third element of the fourth row. To do this, we simply add -2 times the third row to the fourth row and we have our matrix in upper triangular form. Notice the triangle of zeros in the lower left-hand corner.

\[
\begin{bmatrix}
6 & -2 & 2 & 4 & 16 \\
0 & -4 & 2 & 2 & -6 \\
0 & 0 & 2 & -5 & -9 \\
0 & 0 & 0 & -3 & -3
\end{bmatrix}
\]

\(r_4 + -2r \cdot 3\)

Some computational scientists, when working with Gaussian elimination, advocate making all of the diagonal elements one, as this is helpful in the next phase of the process. In our case, it is simply a matter of multiplying the first row times $\frac{1}{6}$, the second row times $\frac{1}{4}$, the third row times $\frac{1}{2}$ and the fourth row by $\frac{1}{3}$.

\[
\begin{bmatrix}
1 & \frac{-1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{8}{3} \\
0 & 1 & \frac{-1}{2} & \frac{1}{2} & \frac{3}{2} \\
0 & 0 & 1 & \frac{-5}{6} & \frac{5}{2} \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

\(r_1 \cdot \frac{1}{6}\)

\(r_2 \cdot \frac{1}{4}\)

\(r_3 \cdot \frac{1}{2}\)

\(r_4 \cdot \frac{1}{3}\)

We have now completed the first phase of solving a system of linear equations using Gaussian elimination. Now all that is left is to back solve for the unknown values of the variables. Recall that our augmented matrix simply was another way of representing the system of equations. Therefore, if we start at the bottom of our matrix, we can solve for the unknowns. From the fourth equation, (using equation notation again for clarity), we find

\[x_4 = 1.\]

Substituting the value that we have found for $x_4$ back into the third equation, we find

\[x_3 - \frac{5}{2} = \frac{-9}{2}, \text{ or } x_3 = -2.\]
If we substitute in the values that we have found for $x_3$ and $x_4$ into the second equation, we find

$$x_2 + 1 - \frac{1}{2} = \frac{3}{2}, \text{ or } x_2 = 1.$$ 

And finally, substituting the values that we have found for $x_2, x_3$ and $x_4$ into the first equation, we find

$$x_1 - \frac{1}{3} + \frac{-2}{3} + \frac{2}{3} = \frac{8}{3}, \text{ or } x_1 = 3.$$ 

Our solution set for this system of equations is

$$x_1 = 3, x_2 = 1, x_3 = -2 \text{ and } x_4 = 1.$$ 

The general procedure for Gaussian elimination is as follows:

1. Write the augmented matrix of the system of linear equations

2. Use elementary row operations to reduce the augmented matrix to upper triangular form

3. Use back substitution to find the solution to the problem.

In some cases, as you are working to reduce a matrix to upper triangular form, it is possible that you will get a zero on the diagonal of the matrix. If this occurs while you are reducing the matrix to upper triangular form, simply switching the rows as we did in the first step of the above example may take care of the problem. However, if you still have a zero on the diagonal once your matrix is completely reduced to upper triangular form, then the coefficient matrix is singular and either the system has an infinite number of solutions, or the system has no solution. Consider the following example

The original augmented matrix

$$\begin{bmatrix} 3 & 5 & 7 & 14 \\ 2 & 2 & -1 & 11 \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{4} & \frac{11}{4} \end{bmatrix}$$

The reduced matrix

$$\begin{bmatrix} 1 & \frac{5}{3} & \frac{7}{3} & \frac{14}{3} \\ 0 & 1 & \frac{11}{4} & \frac{3}{4} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Notice that the third diagonal element is a 0. If we were to attempt to back solve this system to find the solution, at the first step we would find that

$$0x_3 = 0$$
Since this is a true statement regardless of what value we assign to $x_3$, this means that we have an infinite number of possible values for $x_3$, and as long as $x_1$ and $x_2$ satisfy the relationships

\[
x_1 + \frac{5}{3}x_2 + \frac{7}{3}x_3 = \frac{14}{3}
\]

\[
x_2 + \frac{11}{4}x_3 = \frac{-5}{4},
\]

there are an infinite number of solutions to this system of equations. However, if the row containing the zero on the diagonal leads to a false equation, then our system has no solution. Consider the following example.

The original system
\[
\begin{bmatrix}
4 & 7 & -3 & 7 \\
3 & -2 & 4 & 6 \\
\frac{1}{4} & \frac{-1}{6} & \frac{1}{3} & 3
\end{bmatrix}
\]

The reduced matrix
\[
\begin{bmatrix}
1 & \frac{7}{4} & \frac{-3}{4} & \frac{7}{4} \\
0 & 1 & \frac{-29}{28} & \frac{-3}{28} \\
0 & 0 & 0 & -30
\end{bmatrix}
\]

Attempting to back solve for this problem, we get the equation

\[0x_3 = -30\]

which is a false statement. There is no value of $x_3$ that could make this a true statement, and therefore there is no solution to this system of linear equations.

2.1.2 Gauss-Jordan elimination

Gauss-Jordan Elimination is a variant of Gaussian elimination. In Gauss-Jordan elimination, the goal is to reduce the matrix to a diagonal matrix. That is, we wish to get zeros not only below the elements on the diagonal, but also above the elements on the diagonal. This then allows us to see the solution to our system without back solving. In general, the zeros are introduced into the matrix one column at a time, i.e. we work to eliminate the elements both above and below the diagonal element of a given column in one pass through the matrix. Let us go back to the system of equations on page 15. Again, since the first element in the first row of our augmented matrix is a 0, we will switch the first and second rows to get the following matrix.

\[
\begin{bmatrix}
6 & -2 & 2 & 4 & 16 \\
0 & -4 & 2 & 2 & -6 \\
3 & -13 & 9 & 3 & -19 \\
-6 & 4 & 1 & -18 & -34
\end{bmatrix}
\] switch r1 and r2
The next step is the same as in Gaussian elimination - we wish to eliminate the elements in the first column from the third and fourth rows. This time, however, we will introduce the ones on the diagonal as we go rather than as the final step, as this is how it is more commonly done.

\[
\begin{bmatrix}
1 & -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{8}{3} \\
0 & -4 & 2 & 2 & -6 \\
0 & -12 & 8 & 1 & -27 \\
0 & 2 & 3 & -14 & -18
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{8}{3} \\
0 & -4 & 2 & 2 & -6 \\
0 & -12 & 8 & 1 & -27 \\
0 & 2 & 3 & -14 & -18
\end{bmatrix}
\rightarrow
r1 + \frac{1}{6} \\
r2 + \frac{1}{3} \cdot \text{current } r1 \\
r3 + -3 \cdot \text{current } r1 \\
r4 + r1
\]

In the next step, however, we wish to eliminate the second element not only from the third and fourth row, but also from the first. First we will introduce the one to the diagonal element of the second row and then we will add appropriate multiples of the second row to the first, third and fourth rows.

\[
\begin{bmatrix}
1 & 0 & \frac{1}{6} & \frac{1}{2} & \frac{19}{6} \\
0 & 1 & -\frac{1}{2} & -\frac{1}{2} & \frac{3}{2} \\
0 & 0 & 2 & -5 & -9 \\
0 & 0 & 4 & 13 & -21
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 0 & \frac{1}{6} & \frac{1}{2} & \frac{19}{6} \\
0 & 1 & -\frac{1}{2} & -\frac{1}{2} & \frac{3}{2} \\
0 & 0 & 2 & -5 & -9 \\
0 & 0 & 4 & 13 & -21
\end{bmatrix}
\rightarrow
r1 + \frac{1}{3} \cdot \text{current } r2 \\
r2 + \frac{1}{4} \\
r3 + 12 \cdot \text{current } r2 \\
r4 + -2 \cdot \text{current } r2
\]

Now that we have introduced a leading one (the one on the diagonal) and the zeros in the second column, we are ready to add a multiple of the third row to the first, second and fourth rows in order to introduce zeros into the third column of these rows.

\[
\begin{bmatrix}
1 & 0 & 0 & \frac{11}{12} & \frac{47}{12} \\
0 & 1 & 0 & \frac{7}{4} & \frac{-3}{4} \\
0 & 0 & 1 & \frac{5}{2} & \frac{-9}{2} \\
0 & 0 & 0 & -3 & -3
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 0 & 0 & \frac{11}{12} & \frac{47}{12} \\
0 & 1 & 0 & \frac{7}{4} & \frac{-3}{4} \\
0 & 0 & 1 & \frac{5}{2} & \frac{-9}{2} \\
0 & 0 & 0 & -3 & -3
\end{bmatrix}
\rightarrow
r1 + \frac{1}{6} \cdot \text{current } r3 \\
r2 + \frac{1}{2} \cdot \text{current } r3 \\
r3 + \frac{1}{2} \\
r4 + -4 \cdot \text{current } r3
\]

We are now ready to introduce zeros into the fourth column of the first, second, and third rows.

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 3 \\
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & -2 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 0 & 0 & 0 & 3 \\
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & -2 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\rightarrow
r1 + \frac{11}{12} \cdot \text{current } r4 \\
r2 + \frac{7}{4} \cdot \text{current } r4 \\
r3 + \frac{5}{2} \cdot \text{current } r4 \\
r4 \times \frac{1}{3}
\]

At this point, you will notice that we have a diagonal matrix with ones on the diagonal. If we were to back solve this system, we would find exactly what the right hand side is showing us - that

\[x_1 = 3, x_2 = 1, x_3 = -2 \text{ and } x_4 = 1.\]
When doing the calculations by hand, many individuals prefer Gauss-Jordan over Gauss because it avoids the back solve; however, we will show later on that Gauss-Jordan involves slightly more work than does Gaussian elimination and is therefore not the method of choice for solving equations with a computer.

2.1.3 Solving a System of Linear Equations using the Inverse

Another method for solving a system of equations involves using the inverse of a matrix. However, in order to explain the inverse matrix, we must first define the identity matrix, $I$. The identity matrix is the square, $n \times n$ matrix with 1's on the diagonal and zeros at every other entry.

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

In regular algebraic operations or real numbers, you know that $a \times a^{-1} = 1$. We call $a^{-1}$ the multiplicative inverse of $a$, and 1 is known as the multiplicative identity. Similarly, for matrices, if there exists a matrix $A^{-1}$ such that $AA^{-1} = A^{-1}A = I$, then $A^{-1}$ is said to be the inverse matrix of $A$.

Let's look at the development of this method. When you wanted to solve $ax = b$, you could say that you multiplied both sides of the equation by $a^{-1}$ to find $x$ (i.e. $a^{-1}ax = a^{-1}b$). The same thing holds true for matrices. Consider again the equation

$$Ax = b.$$ 

If we multiply both sides of the equation by $A^{-1}$ we get the following equations:

$$A^{-1}Ax = A^{-1}b$$

But $A^{-1}A = I$, so

$$Ix = A^{-1}b \text{ or } x = A^{-1}b$$

Unfortunately, the inverse of a matrix, $A^{-1}$, is not as easy to find as the inverse of a real number. In order to find the inverse, we can solve the matrix equation $AX = I$ (where $X = A^{-1}$) the same as if we were solving a system of linear equations with multiple right-hand sides. In this case, our multiple right-hand sides would be the
matrix I, while the coefficient matrix is still the matrix A. This can be best shown by a simple example. Suppose we wish to find the inverse of the following matrix A.

\[
\begin{bmatrix}
3 & 2 & 1 \\
1 & 2 & 3 \\
4 & 5 & 7
\end{bmatrix}
\]

We wish to solve \( AX = I \) in a manner similar to solving a system of linear equations with multiple right hand sides where the columns of I, in this case are the multiple right hand sides. We have the following augmented matrix.

\[
\begin{bmatrix}
3 & 2 & 1 & 1 & 0 & 0 \\
1 & 2 & 3 & 0 & 1 & 0 \\
4 & 5 & 7 & 0 & 0 & 1
\end{bmatrix}
\]

Now we will use our elementary row operations on A, with the goal of transforming the coefficient matrix into a diagonal matrix with ones on the diagonal. At this point, we could use Gaussian elimination or Gauss-Jordan. We will use Gauss Jordan because it is somewhat cleaner. However, we remind the reader that Gaussian elimination would be somewhat less work. Hence, on a computer, Gaussian elimination would be the method of choice. First we introduce a 1 into the first element of the first row and eliminate the elements below the diagonal element in the first column.

\[
\begin{bmatrix}
1 & \frac{2}{3} & \frac{2}{3} & | & \frac{1}{3} & 0 & 0 \\
0 & \frac{4}{3} & \frac{8}{3} & | & \frac{-1}{3} & 1 & 0 \\
0 & \frac{7}{3} & \frac{12}{3} & | & \frac{-4}{3} & 0 & 1
\end{bmatrix}
\]

\[ r1 \times \frac{1}{3} \]

\[ r2 - r1 \]

\[ r3 + -4 \times r1 \]

Next, we introduce a 1 on the diagonal element of the second row and eliminate the elements above and below the second diagonal element.

\[
\begin{bmatrix}
1 & 0 & -1 & | & \frac{1}{2} & \frac{-1}{2} & 0 \\
0 & 1 & 2 & | & \frac{-1}{4} & \frac{3}{4} & 0 \\
0 & 0 & 1 & | & \frac{-3}{4} & \frac{-7}{4} & 1
\end{bmatrix}
\]

\[ r1 + \frac{-2}{3} \times r2 \]

\[ r2 \times \frac{3}{4} \]

\[ r3 + \frac{-7}{3} \times r2 \]

You will notice that we already have a 1 on the third diagonal element, so all that is left is to eliminate the elements above the diagonal in the third column.

\[
\begin{bmatrix}
1 & 0 & 0 & | & \frac{-1}{4} & \frac{-9}{4} & -1 \\
0 & 1 & 0 & | & \frac{5}{4} & \frac{12}{4} & -2 \\
0 & 0 & 1 & | & \frac{-3}{4} & \frac{-7}{4} & 1
\end{bmatrix}
\]

\[ r1 + r3 \]

\[ r2 + -2 \times r3 \]
In this manner we have solved for X, which in this case is our inverse. Therefore, the right-hand part of the final matrix should be the inverse of A, $A^{-1}$.

This method, in fact, is similar to the method that you would use on the TI-82 to solve a system of linear equations. You will notice, if you go into the matrix math menu on your TI-82, that there is no menu choice “Solve”. However, you do have the option of computing the inverse. Let’s try this with the linear system we solved earlier. Enter the coefficient matrix into your calculator as matrix A (that is, only the first three columns of the matrix.) Now for the matrix B, have the calculator compute the inverse of A - that is, enter $B \rightarrow A^{-1}$ into your calculator. For the matrix C, enter the right-hand side from our system of equations. Now, to compute the solution to the system of equations, simply multiply matrix B times the matrix C. This should give the same solution that we obtained before in the previous section. As we will show, however, this is not a good way to solve a system of linear equations because it requires significantly more work than does Gaussian elimination.

### 2.2 Operation Counts and a Comparison of Methods

So far we have shown you several methods for solving systems of equations. The question arises, which method is the method of choice? As we have already pointed out Cramer’s rule and the substitution methods are really only acceptable for small systems of linear equations or for theoretical usage. To illustrate this, we would like to show you a comparison of the work done by the computer for the above methods. The expense, or computer work, is measured by counting the number of arithmetic operations that are done for a procedure. In general, the amount of work done by the computer depends on the size of the problem. Multiplication and division operations are generally more expensive than addition and subtraction operations as they are more time consuming.

#### 2.2.1 Cost of Gaussian Elimination

As we have stated before, computational scientists are interested in the amount of work, or the expense, for a given algorithm. Let’s take a look at what the Gaussian elimination algorithm costs to solve a system of linear equations. We look at the

---

1It is quite simple to verify that X is the inverse of A. Simply multiply the matrix A times the matrix X, and the result should be the identity, $I = AA^{-1} = A^{-1}A = I$. 

forward elimination and the back substitution separately. Consider the following matrix.

\[
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\
  a_{21} & a_{22} & a_{23} & a_{24} & b_2 \\
  a_{31} & a_{32} & a_{33} & a_{34} & b_3 \\
  a_{41} & a_{42} & a_{43} & a_{44} & b_4
\end{bmatrix}
\]

The dimensions of this matrix are \(4 \times 5\), or in the general case, \(n \times (n + 1)\). The first step of Gaussian elimination is to introduce zeros in all of the elements in the first column below the diagonal element. To do this, we would multiply the last four elements in the second row by a constant and then add the first row to them. You may be asking at this point, why are we not counting the multiplication and addition on the first element of the second column. Since the goal is to introduce a zero in this position, it is assumed that the multiplication and addition will correctly replace this element with a zero, and so no work is required. Do note, however, that the remaining elements of this row have changed, which will be denoted by \(a'_{ij}\). We do the same for the third and fourth rows. In general, the first step of Gaussian elimination takes \(n\) multiplications and \(n\) additions on \(n - 1\) rows, or \(n(n - 1)\) additions and \(n(n - 1)\) multiplications.

\[
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\
  0 & a'_{22} & a'_{23} & a'_{24} & b'_2 \\
  0 & a'_{32} & a'_{33} & a'_{34} & b'_3 \\
  0 & a'_{42} & a'_{43} & a'_{44} & b'_4
\end{bmatrix}
\]

Now that we have the zeros in the first column, we wish to introduce the zeros in the second column below the diagonal. Since the zeros in the first column will not affect our rows, this time we only have to perform 3 multiplications on the third and fourth rows of our matrix, and consequently, we only need to perform 3 additions on each row. The general operation count at the second step is \(n - 1\) multiplications and \(n - 1\) additions on \(n - 2\) rows, or \((n - 1)(n - 2)\) multiplications and \((n - 1)(n - 2)\) additions.

\[
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\
  0 & a''_{22} & a''_{23} & a''_{24} & b''_2 \\
  0 & 0 & a'''_{33} & a'''_{34} & b'''_3 \\
  0 & 0 & a''''_{43} & a''''_{44} & b''''_4
\end{bmatrix}
\]

The last step for this matrix is to introduce the zeros into the third column of our matrix. Again, we will “ignore” the zeros, so we only need to perform 2 multiplications
on the fourth row and add the last two elements of the fourth row to the last two elements of the third row. As you may have already guessed, the general operation count at the third step of Gaussian elimination is \( n - 2 \) multiplications and \( n - 2 \) additions for \( n - 3 \) rows.

\[
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\
    0 & a_{22} & a_{23} & a_{24} & b_2' \\
    0 & 0 & a_{33}'' & a_{34}'' & b_3'' \\
    0 & 0 & 0 & a_{44}''' & b_4'''
\end{bmatrix}
\]

For larger matrices, this would continue until the matrix is in upper triangular form. The operations count so far would be the sum total of the operations for all of the steps of the elimination, or

\[n(n - 1) + (n - 1)(n - 2) + \ldots + 2(1)\] multiplications or additions.

Recall that some prefer to make the diagonal elements one, as this was helpful in back solving. This simply requires \( n \), or 4 multiplications on the first row, \( n - 1 \) or 3 multiplications on the second row, and so on.

In the general case we have

\[
\begin{bmatrix}
    1 & a_{12}' & a_{13}' & a_{14}' & b_1' \\
    0 & 1 & a_{23}'' & a_{24}'' & b_2'' \\
    0 & 0 & 1 & a_{34}''' & b_3''' \\
    0 & 0 & 0 & 1 & b_4'''
\end{bmatrix}
\]

\[n \text{ multiplications}
\]

\[n - 1 \text{ multiplications}
\]

\[n - 2 \text{ multiplications}
\]

\[n - 3 \text{ multiplications}
\]

Now we are ready to back solve to find the solution to our system of equations. Starting from bottom of our array, we already know the value of the \( n^{th} \) variable. Substituting this value into the next equation above and solving requires 1 multiplication and 1 addition. Substituting both values into the the next equation and solving will require 2 multiplications and 2 additions. This continues through the system until we have solved for all of the values of the variables. Substituting the values in to the top equation and solving requires \( n - 1 \) multiplications and \( n - 1 \) additions.

Altogether for the process of solving a system of linear equations using Gaussian elimination, the number of addition operations and multiplication operations are very nearly the same. The only difference lies in introducing the 1's on the diagonal. In total, the number of operations that we performed on the system was

\[
\sum_{i=1}^{n} i^2 = \frac{n(n + 1)(2n + 1)}{6}
\]

multiplications and
\[
\sum_{i=1}^{n} i^2 - \sum_{i=1}^{n} i = \frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2}
\]
additions to reduce

\[
\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}
\]
multiplications to back solve

\[
\sum_{i=1}^{n-1} n - 1i = \frac{(n-1)n}{2}
\]
additions to back solve

If we add all of this together, we find that Gaussian elimination in general requires
\[
\frac{n^3}{3} + n^2 - \frac{5n}{2}
\]
multiplications and
\[
\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}
\]
additions to solve a system of equations.

### 2.2.2 Cost of Gauss-Jordan Elimination

The cost of computing a solution using Gauss-Jordan eliminations is similar to the cost of Gaussian elimination. A small amount of extra work is introduced by introducing the 0's above the diagonal element. Using the same matrix as was used for the Gaussian elimination count, the operations count is developed as follows.

**Step 1:** Introduce the zeros in the first column

\[
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\
  0 & a'_{22} & a'_{23} & a'_{24} & b'_2 \\
  0 & a'_{32} & a'_{33} & a'_{34} & b'_3 \\
  0 & a'_{42} & a'_{43} & a'_{44} & b'_4
\end{bmatrix}
\]

In the general case we have

- \((n-1)\) rows to be operated on
- \(n\) multiplications and
- \(n\) additions.

**Step 2:** Introduce the zeros in the second column

\[
\begin{bmatrix}
  a_{11} & 0 & a'_{13} & a'_{14} & b'_1 \\
  0 & a'_{22} & a'_{23} & a'_{24} & b'_2 \\
  0 & 0 & a''_{33} & a''_{34} & b''_3 \\
  0 & 0 & a''_{43} & a''_{44} & b''_4
\end{bmatrix}
\]

In the general case we have

- \((n-1)\) rows to be operated on
- \((n-1)\) multiplications and
- \((n-1)\) additions.

**Step 3:** Introduce the zeros in the third column

\[
\begin{bmatrix}
  a_{11} & 0 & 0 & a'''_{14} & b'''_1 \\
  0 & a''_{22} & 0 & a''_{24} & b''_2 \\
  0 & 0 & a'''_{33} & a'''_{34} & b'''_3 \\
  0 & 0 & 0 & a'''_{44} & b'''_4
\end{bmatrix}
\]

In the general case we have

- \((n-1)\) rows to be operated on
- \((n-2)\) multiplications and
- \((n-2)\) additions.
Step 4: Introduce the zeros in the fourth column

\[
\begin{bmatrix}
  a_{11} & 0 & 0 & 0 \\
  0 & a_{22} & 0 & 0 \\
  0 & 0 & a_{33} & 0 \\
  0 & 0 & 0 & a_{44}
\end{bmatrix}
\begin{bmatrix}
  b_1 \\
  b_2 \\
  b_3 \\
  b_4
\end{bmatrix}
\]

in the general case we have \((n-1)\) rows to be operated on \((n-3)\) multiplications and \((n-3)\) additions.

Once we have reduced the matrix to a diagonal matrix, we simply have to divide each row by its diagonal element - \(n\) multiplications - and then our right hand side will be the solution to our system of equations. If we add together all of the operations that were done for Gauss-Jordan elimination for a general system of linear equations, we find that we have

\[
\begin{align*}
n(n-1) + (n-1)(n-1) + \ldots + (n-1) + n & \text{ multiplications} \\
(n-1) + (n-1)(n-1) + \ldots + (n-1) & \text{ additions}
\end{align*}
\]

This simplifies to \(\frac{n^3}{3} - n^2 + \frac{3}{2}n\) multiplications and \(\frac{n^3}{2} - n^2 + \frac{3}{2}\) additions, somewhat more work than Gaussian elimination.

2.2.3 Cost of Computing the Inverse of a Matrix

The method of computing the inverse and using it to solve the system of linear equations may seem fairly straightforward and nice to use. However, this method too has its drawbacks. In fact, this method is rarely used. The reason for this is that it is excessively costly to compute the inverse of a matrix. Although the method of computing the inverse that you were shown earlier may seem very similar to the method of Gauss-Jordan elimination, computing the inverse takes more operations. Recall that we stated that computing the inverse was similar to solving a system of linear equations with multiple right-hand sides. These multiple right-hand sides introduce the greater expense involved in computing the inverse. In general, computing the inverse of a matrix costs \(n^3\) multiplications - nearly three times the amount of multiplications done by Gaussian elimination. However, recall that our general operations count from Gaussian elimination included the expense of back solving to find the solution. In this case, we have simply computed the inverse. Finding the solution to the system of equations by using the formula \(x = A^{-1}b\) in general requires another \(n^3 + n^2\) operations. The number of additions is similar, requiring \(n^3 - 2n^2 + n\) additions to find the inverse, and \(n^3 - n^2\) additions to find the solution. Altogether, to compute the inverse and solve the system of linear equations requires \(2n^3 + n^2\)
multiplications and \(2n^3 - 3n^2 + n\) additions using this method, considerably more than using Gaussian or Gauss-Jordan elimination.

### 2.2.4 Cost of Using Cramer’s Rule

Earlier in the chapter, we discussed Cramer’s rule. For Cramer’s rule, we found that we were computing \((n + 1)\) determinants, and that by definition, the determinant was the sum of all \(n!\) possible products. Therefore, to find the solution for a \(n \times n\) system of equations would require \(n!\) multiplications and \(n!\) additions for all \((n + 1)\) determinants, leading to an operations count of \((n + 1)!\) multiplications and \((n + 1)!\) additions to find the solution to this system of equations. The following table compares the multiplication count for solving a system of linear equations using Cramer’s rule and Gaussian elimination for various \(n\). To illustrate our point, we give actual time in terms of a Cray YMP, one of today’s fastest computers.

<table>
<thead>
<tr>
<th>(n)</th>
<th>Gaussian Elimination</th>
<th>Cramer’s Rule</th>
<th>Actual time - C.R.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
<td>(1.8462 \times 10^{-8}) sec</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>24</td>
<td>(7.3846 \times 10^{-8}) sec</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
<td>120</td>
<td>(3.6923 \times 10^{-7}) sec</td>
</tr>
<tr>
<td>5</td>
<td>65</td>
<td>720</td>
<td>(2.2154 \times 10^{-6}) sec</td>
</tr>
<tr>
<td>6</td>
<td>106</td>
<td>5040</td>
<td>(1.5508 \times 10^{-5}) sec</td>
</tr>
<tr>
<td>10</td>
<td>430</td>
<td>39,916,800</td>
<td>(0.1228) sec</td>
</tr>
<tr>
<td>100</td>
<td>343,300</td>
<td>(9.42594 \times 10^{159})</td>
<td>(9.197 \times 10^{141}) centuries</td>
</tr>
<tr>
<td>1000</td>
<td>334,333,000</td>
<td>(100!)</td>
<td></td>
</tr>
<tr>
<td>(n)</td>
<td>(\frac{n^3}{3} + n^2 - \frac{n}{3})</td>
<td>((n + 1)!)</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2.1** work comparison for Gaussian elimination and Cramer’s rule

As you can see, as the size of the matrix grows, the number of operations increases quite quickly. In fact, for Cramer’s rule with a relatively small system of only 1000 variables, the number of operations becomes too large to compute. Real life applications often involve systems of equations with hundreds of thousands of variables. Cramer’s rule would take many, many lifetimes to compute the solution to a system of this size. In comparison, Gaussian elimination of a system with 1000 unknowns on the Cray YMP would take approximately \(1.0287\) sec.

For an upper triangular matrix, the reader can readily show from the definition that the determinant is simply found by finding the product of the diagonal elements
n multiplications. Suppose we choose to find the determinant by first using Gaussian elimination to reduce to upper triangular form, rather than using the definition of the determinant. To solve a linear n × n system using Cramer's rule, we still must compute (n + 1) determinants. Counting just the multiplications required to compute each determinant, it will cost us as much as the forward elimination step of Gaussian elimination (\( \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6} \) multiplications) to reduce the matrix to upper triangular form, with an additional n multiplications. This is a total of \( \frac{n^3}{3} + \frac{n^2}{2} + \frac{7n}{6} \) multiplications required to compute each determinant. Considering that there are (n + 1) determinants to be computed, solving using Cramer's rule in this manner would require \((n + 1)(\frac{n^3}{3} + \frac{n^2}{2} + \frac{7n}{6})\) or \(\frac{n^4}{3} + \frac{5n^3}{6} + \frac{5n^2}{3} + \frac{7n}{6}\) multiplications. While this is considerably less work for large n than using the definition of the determinant, as you can see, it is still much more costly than Gaussian or Gauss-Jordan elimination. It is worth pointing out that using the inverse method for large n is about 6 times as much work as using Gaussian elimination; this is a luxury that we cannot afford.

In conclusion, we leave you with a table comparing the amount of work required for each of our different methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Additions</th>
<th>Multiplications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian elimination</td>
<td>(\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6})</td>
<td>(\frac{n^3}{3} + n^2 - \frac{n}{3})</td>
</tr>
<tr>
<td>Gauss-Jordan elimination</td>
<td>(\frac{n^3}{3} - \frac{n^2}{2})</td>
<td>(\frac{n^3}{3} + \frac{n^2}{2})</td>
</tr>
<tr>
<td>Solving using the inverse</td>
<td>(2n^3 - 3n^2 + n)</td>
<td>(2n^3 + n^2)</td>
</tr>
<tr>
<td>Cramer's Rule(best form)</td>
<td>(\frac{n^4}{3} + \frac{5n^3}{6} + \frac{5n^2}{3} + \frac{7n}{6})</td>
<td>(\frac{n^4}{3} + \frac{5n^3}{6} - \frac{n^2}{3} - \frac{5n}{6})</td>
</tr>
<tr>
<td>Cramer's Rule(definition)</td>
<td>((n + 1))</td>
<td>((n + 1))</td>
</tr>
</tbody>
</table>

Table 2.2 Approximate Operation Counts for Different Methods of Solving Ax = b

2.3 Linear Iterative Solvers

In many applications, the expense of Gaussian elimination is simply too much. Think back to the groundwater remediation problem of Chapter 1. In this problem, the mathematical model involved a large system of differential equations. The scientist may wish to know what the situation would look like each day for the next 50 years. Our model could require the approximate solutions of millions of systems of linear equations at each time interval. Applications such as these motivate a class of solution
techniques called linear iterative solvers. The best known example of these solvers is the conjugate gradient method.

Gaussian elimination, Cramer's rule and the substitution method for solving systems of linear equations are called direct methods because after one pass through the algorithm, we have the exact solution when we are using exact precision arithmetic. In contrast, the conjugate gradient method is called an indirect or iterative method. An iterative method consists of a relatively simple numerical activity that takes an approximation to the solution and attempts to improve upon it. This improvement is then fed back into the algorithm to possibly find a better approximation. The iterative procedure may or may not eventually give the exact answer in a finite number of iterations.

There are three basic premises behind using a linear iterative solver to solve the linear system of equations described in (2.1)

1. First, as in many cases, an exact solution is not needed and a good approximation will suffice.

2. Second, a good approximation can be generated in a relatively few iterations.

3. Third, the simple numerical activity used to find the approximation is quite inexpensive in terms of computer work relative to using Gaussian elimination to solve the original system of linear equations.

One particularly successful iterative procedure is the conjugate gradient method. The class of problems for which the conjugate gradient method is effective is linear systems of the form (2.1) where the coefficient matrix is symmetric and positive definite and has relatively few distinct eigenvalues.\(^2\) It can be shown that for a symmetric, positive-definite linear system, the conjugate gradient method will converge to the solution in at most \(m\) iterations, where \(m \leq n\) is the number of distinct eigenvalues for the coefficient matrix \(A\). Furthermore, if \(A\) is positive-definite and has only a few large eigenvalues, while the remaining eigenvalues are all very small, the conjugate gradient method will generate a good approximate solution in a few iterations. These

\(^2\)Recall that a matrix \(A\) is symmetric if it is equal to its transpose - that is, row \(i\) is equal to column \(i\) \((i = 1 \ldots n)\) for \(A\). Moreover, a matrix is positive-definite if and only if \(x^T A x > 0\) for vectors \(x \neq 0\). A real number \(a\) is positive if and only if there exists a nonzero real number \(b\) such that \(a = b^2\). Similarly, a matrix \(A\) is positive-definite if and only if there exists a nonsingular matrix \(B\) such that \(A = B^2\). Eigenvalues (\(\lambda\)) are defined by the relationship \(A \lambda = \lambda x\) where \(x \neq 0\). For a symmetric, positive-definite matrix, all of the eigenvalues are positive.
situations are not uncommon when linear systems arise from approximations to systems of partial differential equations. The iterative steps for the conjugate gradient method are as follows:

**Initial step:** Select a point \( x_1 \) and compute

\[
 r_1 = b - Ax_1, \quad p_1 = r_1 
\]  

(2.3)

**Iterative steps:** Successively, for \( k = 1, 2, \ldots \), proceed as follows:

having found \( x_k, r_k, p_k \), compute

\[
 A p_k, \quad d_k = p_k^T A p_k, \quad c_k = p_k^T r_k \quad a_k = c_k / d_k 
\]  

(2.4)

\[
 x_{k+1} = x_k + a_k p_k, \quad r_{k+1} = r_k - a_k A p_k 
\]  

(2.5)

\[
 e_k = -p_k^T A r_{k+1}, \quad b_k = e_k / d_k 
\]  

(2.6)

\[
 p_{k+1} = r_{k+1} + b_k p_k 
\]  

(2.7)

**Termination.** Terminate at the \( m^{th} \) step if \( r_{m+1} = 0 \). Then \( m < n \) and \( x_0 = x_{m+1} \) is the solution of \( Ax = b \). The scalars \( c_k \) and \( b_k \) are also given by the formulas

\[
 c_k = r_k^T r_k, \quad b_k = c_{k+1} / c_k 
\]  

(2.8)

The above algorithm will find a good estimate to the the solution to a linear system of equations if it exists. In comparison to the work required for Gaussian elimination (on the order of \( \frac{n^3}{3} \) operations), each step of the conjugate gradient method takes \( n^2 \) operations to complete. Because we know in theory that the conjugate gradient will find the exact solution in at most \( n \) steps, the worst case scenario is that the conjugate gradient method requires \( n \) steps of order \( n^2 \) operations, or order \( n^3 \) operations. However, the conjugate gradient method is most useful for symmetric, positive-definite systems with few distinct eigenvalues. In these cases, it will more often require \( m \) steps of \( n^2 \) operations, or \( mn^2 \) operations with \( m \) small, thus making it significantly less expensive than Gaussian elimination.
Consider the following system of linear equations $Ax = h$ where $A$ is given by:

$$
\begin{bmatrix}
1.765890 & -0.007595 & -0.244492 & -0.093132 & -0.102423 & 0.076507 & 0.012395 & -0.231535 & -0.200103 \\
-0.007595 & 1.090807 & -0.171620 & 0.095189 & -0.108132 & -0.094154 & 0.110673 & 0.085191 & 0.132168 \\
-0.244492 & -0.171620 & 1.612695 & -0.095848 & -0.088303 & 0.088226 & -0.079160 & 0.001396 & -0.341813 \\
-0.093132 & 0.095189 & -0.095848 & 1.409259 & -0.047575 & -0.179338 & -0.404463 & -0.124017 & 0.116068 \\
-0.102423 & -0.108132 & -0.088303 & -0.047575 & 1.748798 & 0.345282 & -0.037281 & -0.133367 & 0.144702 \\
0.076507 & -0.094154 & 0.088226 & -0.179338 & 0.345282 & 1.248795 & 0.056238 & -0.070513 & -0.096636 \\
0.012395 & 0.110673 & -0.079160 & -0.404463 & -0.037281 & 0.056238 & 1.506471 & 0.264543 & 0.045697 \\
-0.231535 & 0.085191 & 0.001396 & -0.124017 & -0.133367 & -0.070513 & 0.264543 & 1.214792 & 0.113458 \\
-0.200103 & 0.132168 & -0.341813 & 0.116068 & 0.144702 & -0.096636 & 0.045697 & 0.113458 & 1.357845
\end{bmatrix}
$$

and $h = \begin{bmatrix} 1 \\ 5 \\ 3 \\ 7 \\ 3.2 \\ 7.5 \\ 5.6 \\ 3 \end{bmatrix}$

This system of equations has the property that the coefficient matrix above is symmetric and positive-definite. It also has the property that while it has 9 distinct eigenvalues, these eigenvalues all are close to 1 or 2. Therefore, we expect that the conjugate gradient method will find a good approximation to this system of equations in around 2 steps. Let us work through the conjugate gradient method with this system of linear equations.
**Initial Step** The initial step requires that we choose a point, $x_1$ and compute $r_1$ and $p_1$ as follows.

$$
x_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad r_1 = h - Ax_1 = \begin{bmatrix} -0.768590 \\ 5.007595 \\ 3.244492 \\ 7.093132 \\ 3.302423 \\ 7.423493 \\ 5.587605 \\ 3.231535 \\ 1.200103 \end{bmatrix} \quad p_1 = r_1
$$

The termination conditions are that we stop when $r_{m+1} = 0$. It is easier to determine how quickly this is happening if we look at the product $r_k^T r_k$ - a scalar - rather than the entire vector $r$. For this reason, from here on we will look at the product $r_k^T r_k$ rather than $r$. Currently, $r_k^T r_k = 195.6205097630452$ - a rather large number.

**Iterations** If we perform the calculations as prescribed by the iterative steps we get the following:

Iteration 1:

$d_1 = 227.08396; c_1 = 195.62051; a_1 = 0.861446; e_1 = 21.509349; b_1 = 0.096036; and \ r_2^T r_2 = 19.8193$.

The stopping criteria - that $r_{m+1}$ be sufficiently close to zero - has not been met, so we must repeat this process. This time we find the following information.

Iteration 2:

$d_2 = 32.726778; c_2 = 18.787070; a_2 = 0.574058; e_2 = 0.006598; b_2 = 0.0216075; and \ r_3^T r_3 = 0.02654$

As you can see, the values of $r_k^T r_k$ are steadily decreasing. Table 2.3 is a table of the data obtained from running the conjugate gradient steps on this matrix for $n = 9$ iterations.

If we look carefully at the column $r_{i+1}^T r_{i+1}$, recalling that our original value was 195.6205, we notice that the largest improvements in this value occur after the first two iterations. After iteration two, we are relatively close to zero - close enough for the vector $x$ at this iteration to be considered a good estimate. Remember that the conjugate gradient method is as useful or more useful for the good estimates it
<table>
<thead>
<tr>
<th>$i$</th>
<th>$r_i^T r_{i+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>195.6205</td>
</tr>
<tr>
<td>1</td>
<td>19.8193</td>
</tr>
<tr>
<td>2</td>
<td>0.02654</td>
</tr>
<tr>
<td>3</td>
<td>0.00339</td>
</tr>
<tr>
<td>4</td>
<td>6.91575e-06</td>
</tr>
<tr>
<td>5</td>
<td>8.39394e-07</td>
</tr>
<tr>
<td>6</td>
<td>1.85728e-09</td>
</tr>
<tr>
<td>7</td>
<td>9.80238e-11</td>
</tr>
<tr>
<td>8</td>
<td>8.79953e-13</td>
</tr>
<tr>
<td>9</td>
<td>9.691734e-15</td>
</tr>
<tr>
<td>10</td>
<td>6.56988e-16</td>
</tr>
</tbody>
</table>

Table 2.3 Nine iterations of the conjugate gradient method.

provides with few iterations. In this case, a good estimate for $x$ is

$$x_3 = \begin{bmatrix} 1.73383 \\ 4.96348 \\ 2.714932 \\ 7.74344 \\ 1.52573 \\ 6.735869 \\ 4.85313 \\ 2.69639 \\ 0.44697 \end{bmatrix} \quad eig(A) = \begin{bmatrix} 0.95636 \\ 1.00773 \\ 1.00773 \\ 1.99651 \\ 1.99903 \\ 1.99903 \\ 0.99889 \\ 1.00012 \\ 2.00002 \end{bmatrix}$$

As you may recall from the earlier reading, this matrix had the property that its eigenvalues are all distinct yet clustered around 1 and 2. As you can see, we have listed the eigenvalues for our matrix, $A$, and this is the case with 5 of the eigenvalues very near 1 and 4 of the eigenvalues close to 2.

Next we would like to see what happens if we work on a matrix $B$ with exactly two eigenvalues - 1 and 2. As we stated before, the conjugate gradient method should certainly converge to a good estimate for $x$ in two iterations. However, in order to more fully recognize this, we performed 15 iterations of the conjugate gradient method on our symmetric, positive-definite matrix, $B$. However, in the first two steps, the conjugate gradient quickly converged to a good approximation. The remaining steps less significant improvements - improvements that were so small that it was not worth
the computer expense to compute them. In the initial step for this problem, the value of \( r^T r \) was 187.1044.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( r_i^T r_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>187.1044</td>
</tr>
<tr>
<td>1</td>
<td>12.24565</td>
</tr>
<tr>
<td>2</td>
<td>2.11267e-29</td>
</tr>
<tr>
<td>3</td>
<td>1.10720e-30</td>
</tr>
</tbody>
</table>

**Table 2.4** Three iterations of the conjugate gradient method

Sparse matrices often appear in practice in the area of partial differential equations. A sparse matrix is a matrix that has a large number of entries that are zeros. The matrix in Figure 2.1 is an example of a large, sparse matrix. This matrix is symmetric and positive definite and has 7 distinct eigenvalues. A very good approximation of the solution for a system of equations with this coefficient matrix can be found in seven iterations. Sparse matrices often have the property of having few clustered eigenvalues, thus making them good candidates for the conjugate gradient method.
Figure 2.1 a large sparse matrix.
Chapter 2 Problems

It is possible to manually perform Gaussian elimination on the graphing calculator. Using the elementary row operations, use your graphing calculator to help complete the following problems.

1.) [Morris 2.3.1, 45] Calculate the solution of the following system of linear equations using Cramer’s Rule.

\[
\begin{align*}
3x_1 + 2x_2 - 3x_3 &= -4 \\
x_1 + x_2 - x_3 &= 0 \\
2x_1 - x_2 - x_3 &= -7
\end{align*}
\]

2.) [Morris 2.3.2, 45] Calculate the solution of the following system of linear equations using Cramer’s rule.

\[
\begin{align*}
3x_1 + 4x_2 - 2x_3 + x_4 &= 6 \\
2x_1 - 2x_2 + 3x_3 + x_4 &= 4 \\
-x_1 + x_2 - x_3 + x_4 &= 0 \\
2x_1 - 2x_2 + 3x_3 + 3x_4 &= 6
\end{align*}
\]

3.) [Morris, 2.3.4] Determine \( x_1 \) and \( x_2 \) using Gaussian elimination and three decimal arithmetic. How well do the solutions satisfy the equations?

a.) \[
\begin{bmatrix}
1 & 6 \\
2 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} = 
\begin{bmatrix}
3 \\
1
\end{bmatrix}
\]

b.) \[
\begin{bmatrix}
0.43 & 1.1 \\
0.48 & 2.3
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} = 
\begin{bmatrix}
5.128 \\
3.601
\end{bmatrix}
\]

c.) \[
\begin{bmatrix}
-2.1 & 1.01 \\
3.675 & 2.3
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} = 
\begin{bmatrix}
-2.1 \\
3.1
\end{bmatrix}
\]

4.) Solve the following system of equations, retaining only four significant figures in each step of the calculation, and compare your answer with the solution obtained when eight significant figures are retained.

\[
\begin{align*}
0.1036x + 0.2122y &= 0.7381 \\
0.2081x + 0.4247y &= 0.9327
\end{align*}
\]
Be consistent by either always rounding to the number of significant figures that are being carried or always chopping.

5.) [Froberg 10.3, 166] Solve the following system of linear equations using Gaussian elimination.

\[
\begin{align*}
x + 2y + 3z &= 13 \\
2x + 4y + 2z &= 14 \\
3x + 5y - z &= 11
\end{align*}
\]

6.) [Nakamura, ex 6.1, 187] Solve the following system of linear equations using Gaussian elimination.

\[
\begin{align*}
2x_1 + x_2 - 3x_3 &= -1 \\
x_1 + 3x_2 + 2x_3 &= 12 \\
3x_1 + x_2 - 3x_3 &= 0
\end{align*}
\]

7.) [Nakamura, 6.3, 190] Solve the following system of linear equations using Gaussian elimination.

\[
\begin{bmatrix}
-0.04 & 0.04 & 0.12 & 3 \\
0.56 & -1.56 & 0.32 & 1 \\
-0.24 & 1.24 & -0.28 & 0
\end{bmatrix}
\]

8.) [Froberg, 150] Use Gaussian elimination to solve the following system of linear equations.

\[
\begin{align*}
x - 2y + 3z - 4u &= 0 \\
3x - 2y + 3z - 7u &= 5 \\
5x - 18y + 29z - 23u &= 1 \\
4x - 4y - 29u &= -25
\end{align*}
\]

9.) [Froberg 10.1, 166] Solve the following system of linear equations using Gaussian elimination.

\[
A = \begin{bmatrix}
3 & 7 & -6 & 2 \\
9 & 23 & -13 & 10 \\
15 & 31 & -36 & 10 \\
-12 & -24 & 38 & 9
\end{bmatrix}
\quad \text{and} \quad b = \begin{bmatrix}
4 \\
9 \\
30 \\
-17
\end{bmatrix}
\]
10.) [Buchanan 6.1.1, 228] Solve the following system of equations using Gaussian elimination.

\[
x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 11.2345 \\
x_1 - 2x_2 + 3x_3 - 4x_4 + 5x_5 - 6x_6 = 8.4970 \\
2x_1 + 2x_2 - 2x_4 + x_6 = 21.9405 \\
-4x_1 + 10x_3 + 5x_5 - 11x_6 = -37.9855 \\
x_2 - x_3 + 2x_4 - x_5 = 0.8560 \\
x_1 - 9x_2 - 20x_6 = 0.9900
\]

11.) [Nakamura, ex 6.2] Solve the same system of linear equations in problem 6 by Gauss-Jordan elimination.

12.) [Nakamura, 6.2] Solve the following sets of equations by Gauss-Jordan elimination.

a.) \(4x + y - z = 9\)  
\(3x + 2y - 6z = -2\)  
\(x - 5y + 3z = 1\)

b.) \(x - y = 0\)  
\(-x + 2y - z = 1\)  
\(-y + 1.1z = 0\)

13.) The following sets of linear equations have common coefficients but different right-hand sides.

a.) \(x + y + z = 1\)  
\(2x - y + 3z = 4\)  
\(3x + 2y - 2z = -2\)

b.) \(x + y + z = -2\)  
\(2x - y + 3z = 5\)  
\(3x + 2y - 2z = 1\)
c.) \[ x + y + z = 2 \]
\[ 2x - y + 3z = -1 \]
\[ 3x + 2y - 2z = 4 \]

The coefficients and the 3 sets of right-hand sides may be combined in an array

\[
\begin{bmatrix}
1 & 1 & 1 \\
2 & -1 & 3 \\
3 & 2 & -2
\end{bmatrix}
\begin{bmatrix}
1 & -2 & 2 \\
4 & 5 & -1 \\
-2 & 1 & 4
\end{bmatrix}
\]

If we apply Gauss-Jordan elimination to this array and reduce the first 3 columns to the unit matrix form, the solutions for the 3 problems are automatically obtained in the fourth, fifth and sixth columns when the elimination is completed. Calculate the solution in this way.

14.) [Nakamura 6.12] Calculate the inverse of

\[ A = \begin{bmatrix}
7 & 1 \\
4 & 5
\end{bmatrix} \]

15.) [Nakamura 6.14] Calculate the inverse of the following:

\[ A = \begin{bmatrix}
3 & 1 & 0 \\
1 & 2 & 1 \\
0 & 1 & 1
\end{bmatrix} \]

16.) [Nakamura 6.13] Calculate the inverse of the following matrices.

\[ A = \begin{bmatrix}
1 & -1 & 0 & 0 \\
-1 & 2 & -1 & 0 \\
0 & -1 & 2 & -1 \\
0 & 0 & -1 & 2
\end{bmatrix} \]
\[ B = \begin{bmatrix}
1 & 4 & 5 \\
2 & 1 & 2 \\
8 & 1 & 1
\end{bmatrix} \]
Chapter 3

Solutions of Nonlinear Equations

Many problems in business, science and engineering require that we approximate the solution of a nonlinear equation (or the zero of a nonlinear function). In fact, in many applications we are interested in finding the solution to a system of nonlinear equations. While the more difficult problem is extremely important, we will restrict our attention in this text to the problem of finding the solution to a single nonlinear equation. But first we must define a nonlinear equation.

Consider the function \( f(x) \). We say that the real number \( x^* \) is a zero or solution of \( f(x) \) if \( f(x^*) = 0 \). So a zero or root of \( f \) is a solution of the equation \( f(x) = 0 \). Contemporary usage favors the use of zero over root. Traditionally, root was used in the context of solutions of polynomial equations, that is when \( f \) was described by a polynomial. Recall that \( f \) is linear if it is of the form \( f(x) = ax + b \) where \( a \) and \( b \) are real numbers. The term nonlinear means that \( f \) is not necessarily linear - that is, the linear function is a special case of a nonlinear function. In this chapter we discuss several methods that allow us to approximate a zero of \( f \). We will always assume that \( f \) is at least a continuous function.

A very common, important example of finding a zero of a nonlinear equation is the problem of calculating the square root of a positive number \( \phi \). The problem can be stated as solving the nonlinear equation \( x^2 - \phi = 0 \). Recall that in the previous chapter, we argued that calculators and computers can only perform simple comparisons, data manipulations, and simple arithmetic. Therefore they can only solve nonsingular linear systems. In the simply case of only one variable, a nonsingular linear system is of the form \( ax + b = 0 \) where \( a \neq 0 \).

We will now derive four well-known algorithms for approximating solutions of the nonlinear equation \( f(x) = 0 \) where \( x \) represents an unknown real number. As discussed above, our tools are either comparisons or approximations by nonsingular linear equations. The attention to comparisons alone will lead us to the method of bisection, discussed first in the chapter. On the other hand, attention to approximating by linear equations alone will lead us to Newton’s method and the secant method.
Finally, our fourth method, the method of false positions, results from an attempt to derive a method which combines the strength of the bisection method with the strength of the secant method. As such, it will employ both comparisons and linear approximations.

First, we will make some preliminary background statements with the goal of better motivating and understanding Newton's method and the secant method. Let us return to the nonlinear equation \( f(x) = 0 \). We are interested in approximating a solution \( x^* \). Suppose that we have \( x_0 \) as an approximation of \( x^* \) and we wish to find a better approximation, say \( x_1 \). Our approach is to utilize a linear approximation of \( f \). A reasonable requirement is to ask the linear approximation to pass through the point \((x_0, f(x_0))\). Appealing to the slope-point form of a line, we obtain as our approximation a linear function \( l \) of the form

\[
l(x) = a(x - x_0) + f(x_0). \tag{3.1}\]

Our task is now to make an intelligent choice for the slope \( a \). This will be the topic of discussion in the sections on Newton's method and the secant method. For the moment, suppose that we have made a choice for the slope \( a \). Then it seems reasonable to use the solution of the linear equation \( l(x) = 0 \) as our improved approximation \( x_1 \). We then obtain \( x_1 \) from the expression

\[
a(x_1 - x_0) + f(x_0) = 0 \tag{3.2}\]

which gives us the formula

\[
x_1 = x_0 - \frac{f(x_0)}{a}. \tag{3.3}\]

We now see why we must require the linear system to be nonsingular, so \( a \neq 0 \). This entire approach is extended in an iterative fashion by replacing \( x_0 \) with \( x_1 \) and hopefully obtaining an even better approximation to \( x^* \).

### 3.1 The Method of Bisection

If we choose to look at simple comparisons rather than linear approximations, we are led to a rather simple minded, yet highly reliable algorithm known as the Method of Bisection. With the goal of motivating this method, let us return to the problem of solving the nonlinear equation \( f(x) = 0 \). A typical graphical representation is shown in Figure 3.1.
Figure 3.1  a typical graphical representation of $f(x)$

In the bisection method, we are using the somewhat intuitive notion that if a continuous function $f$ has the property that $f(a)$ and $f(b)$ are of opposite signs, then there exists a zero of $f$ somewhere in between $a$ and $b$. This intuitive fact is depicted in Figure 3.1.

In order to start the bisection method, we need $a$ and $b$ such that $f(a)f(b) < 0$. Once we have the starting points, the next step is to find the midpoint of our interval $[a, b],
\[ c = \frac{(a + b)}{2}. \]

If we are extremely fortunate, we will find that $f(c) = 0$. However, this is not the usual case and we will have $f(c) \neq 0$. If $f(a)f(c) < 0$, we know that there is a zero of $f$ in the interval $[a, c]$. Otherwise, there is a zero in the interval $[c, b]$.

Once again, we have a situation as we did at the beginning of the problem, only this time the interval is only half as large. So we begin again - dividing the interval in half and checking the signs. We continue this procedure until we have either found the solution of the equation or our interval is suitably small, implying that our error is small.

Consider the problem of finding the $\sqrt{13}$. Let us look at the graph of the function $f(x) = x^2 - 13$, recalling that we wish to know when $f(x) = 0$. On your graphing calculator, plot the function $f(x) = x^2 - 13$. Now use the trace button on your calculator to find an approximate zero of this function. Remember that you are looking for what value of $x$ makes the function equal to 0, so you wish to find where this function is crossing the x axis. You will notice that it crosses the x axis in two places, so this function has at least two zeros. The first graph on Figure 3.2 shows
the graph of \( f(x) = x^2 - 13 \) with its zeros circled. As you trace along the graph of
the function, you may find that you cannot find the point where \( y \) is exactly zero.
In fact, the closest we were able to come using the graphing calculator is to find the
point where \( y \approx 0.083 \), so we will need some way of fine-tuning our tracing in order
to focus in on that point. For this, we will use the bisection method.

![Graph of the original function and the first interval [a, b]](image)

**Figure 3.2** Finding \( \sqrt{13} \) using the Bisection Method

We must begin with an interval, \([a, b]\) where \( f(a) \) and \( f(b) \) have different signs, so
we will use \( a = 4 \) and \( b = 2 \). Then \( f(a) = 3 \) and \( f(b) = -9 \), so \( f(a)f(b) < 0 \) and we
have satisfied our starting criteria. If we proceed as above, we find that the next step
is to find the midpoint of the interval. Our midpoint is found at

\[
c = \frac{4 + 2}{2}
\]

or 3. In this case, \( f(c) = f(3) = -4 \), so we have not found the zero and must continue
on. Since \( f(c) < 0 \), we will replace 2 with 3 (\( b \) with \( c \)) and \( f(a)f(c) \) is still less than
0. Again we find the midpoint of the interval and continue to do so until we have
reached \( f(c) \approx 0 \). The following chart shows the progression of the bisection method.
We will stop when \( |f(c)| \leq 0.0001 \).

As you will see, eventually we came to the same conclusion as we would if we were
finding the square root on the calculator - that \( \sqrt{13} \approx 3.6055 \). However, as you may
<table>
<thead>
<tr>
<th>n</th>
<th>a</th>
<th>b</th>
<th>f(a)</th>
<th>f(b)</th>
<th>c</th>
<th>f(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>-9</td>
<td>3</td>
<td>-4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>-4</td>
<td>3.5</td>
<td>-.75</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3.5</td>
<td>3</td>
<td>-.75</td>
<td>3.75</td>
<td>1.0625</td>
</tr>
<tr>
<td>4</td>
<td>3.75</td>
<td>3.5</td>
<td>1.6025</td>
<td>-.75</td>
<td>3.625</td>
<td>.140625</td>
</tr>
<tr>
<td>5</td>
<td>3.625</td>
<td>3.5</td>
<td>0.140625</td>
<td>-.75</td>
<td>3.5625</td>
<td>-.30859375</td>
</tr>
<tr>
<td>6</td>
<td>3.625</td>
<td>3.5625</td>
<td>0.140625</td>
<td>-.30859</td>
<td>3.59375</td>
<td>-.084960938</td>
</tr>
<tr>
<td>7</td>
<td>3.625</td>
<td>3.59375</td>
<td>0.140625</td>
<td>-.08496</td>
<td>3.609375</td>
<td>0.027587891</td>
</tr>
<tr>
<td>8</td>
<td>3.609375</td>
<td>3.59375</td>
<td>0.02759</td>
<td>-.08496</td>
<td>3.6015625</td>
<td>-.028749359</td>
</tr>
<tr>
<td>9</td>
<td>3.609375</td>
<td>3.6015625</td>
<td>0.02759</td>
<td>-.02875</td>
<td>3.6054688</td>
<td>-.000594155</td>
</tr>
<tr>
<td>10</td>
<td>3.609375</td>
<td>3.6054688</td>
<td>0.02759</td>
<td>-.000595</td>
<td>3.6074219</td>
<td>0.013492765</td>
</tr>
<tr>
<td>11</td>
<td>3.6074219</td>
<td>3.6054688</td>
<td>0.013493</td>
<td>-.000595</td>
<td>3.6069844</td>
<td>0.01033639</td>
</tr>
<tr>
<td>12</td>
<td>3.6069844</td>
<td>3.6054688</td>
<td>0.010336</td>
<td>-.000595</td>
<td>3.6062266</td>
<td>0.0048702905</td>
</tr>
<tr>
<td>13</td>
<td>3.6062266</td>
<td>3.6054688</td>
<td>0.00487</td>
<td>-.000595</td>
<td>3.6058477</td>
<td>0.0021376356</td>
</tr>
<tr>
<td>14</td>
<td>3.6058477</td>
<td>3.6054688</td>
<td>0.00214</td>
<td>-.000595</td>
<td>3.6056483</td>
<td>0.0007714157</td>
</tr>
<tr>
<td>15</td>
<td>3.6056483</td>
<td>3.6054688</td>
<td>0.000771</td>
<td>-.000595</td>
<td>3.6055586</td>
<td>0.0000526</td>
</tr>
</tbody>
</table>

Table 3.1 Bisection Method Data Table

also notice, it took us quite a few iterations to approximate the zero of the function. The bisection method is very reliable - if you began with an interval where f at the endpoints is of different signs, then there must be at least one zero of f in the interval and the bisection method will systematically narrow down the interval until the zero is found to a specified tolerance. Unlike Newton's method or the secant method, which we discuss later in the chapter, it is not possible to make a "poor choice" of starting values as long as they meet the criteria. However, the bisection method is the slowest of the methods that we describe. Hence the price we pay for reliability is a loss of speed.

3.2 Newton's Method

Newton's method, or as it is also called, the Newton-Raphson method, is a very common method used to find solutions of nonlinear equations. Finding square roots is by far one of the simplest applications for which it is useful. Newton's method can be applied to a large class of problems.

Throughout his scientific writings, Newton shows interest in methods for finding the solutions of nonlinear equations. His writings show work on the secant method,
which we will discuss in the next section. What we now call Newton’s method involves the use of derivatives, but Newton’s notes show work on this method that predates his “development” of the derivative and calculus. An early published work of Newton concerned solving Kepler’s equation

\[ x - \epsilon \sin(x) = M \]

for specified real numbers \( \epsilon \) and \( M \). Kepler’s equation was used to determine the position of a planet moving in an elliptical orbit around the sun. Although earlier mathematicians used equivalent techniques for finding solutions using only algebra to find the derivative term, Newton’s method using the derivative from calculus is still commonly used today. In spite of the fact that this method uses the knowledge of the derivative from calculus, we will attempt to give you an understanding of this method.

Consider the situation depicted in Figure 3.3. One fairly obvious linear approximation to the nonlinear function \( f \) at the point \( x_0 \) is the line \( l(x) \) passing through the point \((x_0, f(x_0))\) and tangent to the function \( f \) at \( x_0 \). Let \( f'(x_0) \) denote the slope of the tangent line. Then using the slope-point form of this line, as described earlier in this chapter, we obtain the line

\[ l(x) = f'(x_0)(x - x_0) + f(x_0). \]

As our improved approximation to \( x^* \), a zero of \( f \), we use \( x_1 \), the \( x \)-intercept of this tangent line. So we solve \( l(x) = 0 \) for \( x_1 \). This leads to

\[ x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \]

where \( f'(x_0) \) represents the slope of our tangent line.\(^1\) Continuing this activity in an iterative fashion, we arrive at the formula for Newton’s method

\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \]

Let’s look again at the function \( f(x) = x^2 - 13 \). For this function, we know from calculus that \( f'(x) = 2x \).\(^2\) Using this information, let us use the formula for

\(^1\)If \( f'(x) \) denotes the slope of the line tangent to \( f \) at \( x \), we know from calculus that \( f'(x) \) is the derivative of \( f \) at \( x \) and is defined by the formula \( f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x)-f(x)}{\Delta x} \).

\(^2\)For this simple function, we see that \( \frac{f(x+\Delta x)-f(x)}{\Delta x} = 2x + \delta x \); This quantity approaches 2x as \( \delta x \) approaches zero. So \( f'(x) = 2x \).
Newton's method to find $\sqrt{13}$. We first need an initial value for $x_0$, so we will choose the value $x_0 = 3.5$ (as we can tell from the graph of the function in Figure 3.2, we know that the zero lies somewhere between 3 and 4). We have $f(x_0) = -0.75$ and $f'(x_0) = 2x_0 = 7$. Using the formula for Newton's method, our new approximation is $x_1 = 3.607$. We have $f(x_1) = -1.4882296$ - not close enough to 0 - so we iterate again. The Newton's method formula gives us $x_2$ as 3.6122153. Now we check again to see how close we have come to $f(x) = 0$. This time, $f(x_2) = 0.0479884$, and we have moved closer to 0. We use as our stopping criterion $|f(x_n)| \leq 0.0001$ as we did in the bisection method example. Using Newton's formula again, we find $x_3 = 3.6055574$ and $f(x_3) = 0.0000629136$, which is below our stopping tolerance. The following table illustrates the data used to find this solution. As you can see,

<table>
<thead>
<tr>
<th>i</th>
<th>$x_i$</th>
<th>$f(x_i)$</th>
<th>$f'(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.5</td>
<td>-0.75</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>3.39286</td>
<td>-1.488230</td>
<td>6.7858</td>
</tr>
<tr>
<td>2</td>
<td>3.61222</td>
<td>0.0479884</td>
<td>7.2244</td>
</tr>
<tr>
<td>3</td>
<td>3.60556</td>
<td>0.0000629136</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2 Newton's Method Data

Newton's method converged (reached the tolerance for $f(x)$) much quicker than did the bisection method. In fact, it can be shown that if our starting point, $x_0$, is well chosen, (close to a solution) then Newton's method will converge quadratically.
Informally, we can say that if at the $i^{th}$ step our current value for $x^*$ is accurate to the $k^{th}$ decimal place, then in the $(n + 1)^{st}$ step, it will be accurate to the $2k^{th}$ decimal place. In other words, if we choose a good starting point, then for each step of Newton's method, we will receive twice as many digits of accuracy than we had at the previous step. Newton's method is extremely fast and quite effective at finding the zeros of a nonlinear function. So you may be asking at this point, why would we wish to look further? Recall from Chapter 2 that a computational scientist measures work in terms of the number of operations done by a specific method. In Newton's method, we have to evaluate two functions at $x_n$: $f(x_n)$ and $f'(x_n)$. Depending on the complexity of our nonlinear function $f$, this could require a significant amount of work at each step, possibly making it more expensive than some of the other methods. Moreover, if $x_0$ is not sufficiently close to a zero, $x^*$, then Newton's method may get in trouble and not converge. This can be seen by looking at the nonlinear function depicted in Figure 3.5

### 3.3 The Secant Method

We have just seen that the fundamental ingredient in Newton's method is the approximation of the nonlinear function $f$ at $x_0$ by the line passing through the point $(x_0, f(x_0))$ and tangent to $f$ at $x_0$. This activity required the use of the derivative $f'(x_0)$. An alternative approach which would not use the derivative would be to instead approximate $f$ at $x_0$ by the secant line $l(x)$ passing through $(x_0, f(x_0))$ and another point, say $(x_{-1}, f(x_{-1}))$. This situation is depicted in Figure 3.4. As an improved approximation to $x^*$, a zero of $f$, we use $x_1$, the $x$-intercept of this two point line. The slope of $l(x)$, the line passing through $(x_0, f(x_0))$ and $(x_{-1}, f(x_{-1}))$ is

$$\frac{f(x_0) - f(x_{-1})}{x_0 - x_{-1}}.$$  \hfill (3.5)

As described earlier in this chapter, our line is given by (3.1) with the slope as given by (3.5) and can be written as

$$l(x) = \frac{f(x_0) - f(x_{-1})}{x_0 - x_{-1}}(x_1 - x_0) + f(x_0)$$ \hfill (3.6)

Solving $l(x) = 0$ for $x_1$ gives

$$x_1 = x_0 - \frac{x_0 - x_{-1}}{f(x_0) - f(x_{-1})}f(x_0)$$ \hfill (3.7)
If we continue this activity in an iterative fashion, we arrive at the formula for the secant method

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n). \quad (3.8)$$

It is important to notice that at each iteration, except the first, we use only one new point, \((x_n, f(x_n))\), and one old point, \((x_{n-1}, f(x_{n-1}))\), to define the improved approximate \(x_{n+1}\). Therefore, except for the first iteration, the secant method requires only one new function evaluation per iteration. Webster defines a secant as "any straight line intersecting a curve at two or more points," so it should be clear how the secant method got its name.

For \(x_0\) close to \(x_{-1}\), the secant line passing through the points \((x_0, f(x_0))\) and \((x_{-1}, f(x_{-1}))\) is close to the line passing through \((x_0, f(x_0))\) and tangent to \(f\) at \(x_0\). It follows that the slope of the secant line, \(\frac{f(x_0) - f(x_{-1})}{x_0 - x_{-1}}\) will be close to \(f'(x_0)\), the slope of the tangent line. So, not only can we view the secant method as Newton's method with a fairly obvious approximation for the derivative; but the two methods will differ less and less as our iterates approach \(x^*\), a zero of \(f\).

Consider again the problem of finding the zeros of \(f(x) = x^2 - 13\). To begin the secant method, we need two starting points, \(x_0\) and \(x_{-1}\). Again, the choosing of these starting points is rather arbitrary, yet choosing starting points closer to \(x^*\) will lead to finding the solution in fewer iterations. The points that we have chosen from tracing the graph of the function on the calculator are \(x_{-1} = 3.19\) and \(x_0 = 4.04\).
Substituting these values into the function, \( f \), we find that the two points used for our secant line are \((x_{-1}, f(x_{-1})) = (3.19, -2.81)\) and \((x_0, f(x_0)) = (4.04, 3.34)\). In Figure 3.5, you can see the graph of the function \( f \) with the zeros circled. As you may have noticed, it appears that our secant line is on top of our function. However, if we zoom in on this function, as we did in the second graph of Figure 3.5, you will see that this is not the case.

![chosen x values and connecting line](image1)

![zooming in on the function](image2)

**Figure 3.5** the secant method

For the secant method, just as in Newton’s method and the bisection method, we will stop iterating when \(|f(x_n)|\) is below some tolerance level. Again, in this case we will use \(|f(x_n)| \leq 0.0001\) as our tolerance. Neither \(f(x_{-1})\) nor \(f(x_0)\) is less than our tolerance level, so we need to fine tune our \(x_0\). Substituting our two points into equation (3.8), we find the improvement, \(x_1\), to be 3.578. Evaluating \(f(x_1)\), we get a value of -0.198, still not below our tolerance level, so we must try again. This time, substituting the points \((x_0, f(x_0)) = (4.04, 3.34)\) and \((x_1, f(x_1)) = (3.578, -0.198)\) into equation (3.8), we find a better approximation to \(x^*\) at \(x_2 = 3.604\) which gives us \(f(x_2) = -0.0123\). While we are approaching our tolerance, we still are not close enough, so we must try again. In this case, substituting \((x_1, f(x_1)) = (3.578, -0.198)\) and \((x_2, f(x_2)) = (3.604, -0.0123)\) into equation (3.8), we find our new \(x\)-value to
be \( x_3 = 3.608 \), with \( f(x_3) = -0.0196 \). Again, we must find a better \( x^* \). This time, substituting \((x_2, f(x_2)) = (3.604, -0.0123)\) and \((x_3, f(x_3)) = (3.608, -0.0196)\) into equation (3.8) gives us a new \( x \)-value of \( x_4 = 3.6055 \). If we evaluate \( f \) at this \( x_4 \), we find that \( f(x_4) = -0.0003 \), a value below our stopping criteria. Table 3.3 more clearly illustrates the data used to find this solution.

<table>
<thead>
<tr>
<th>i</th>
<th>( x_i )</th>
<th>( f(x_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>3.19</td>
<td>-2.81</td>
</tr>
<tr>
<td>0</td>
<td>4.04</td>
<td>3.34</td>
</tr>
<tr>
<td>1</td>
<td>3.57</td>
<td>-0.198</td>
</tr>
<tr>
<td>2</td>
<td>3.604</td>
<td>-0.0123</td>
</tr>
<tr>
<td>3</td>
<td>3.608</td>
<td>-0.0196</td>
</tr>
<tr>
<td>4</td>
<td>3.6055</td>
<td>-0.0003</td>
</tr>
</tbody>
</table>

**Table 3.3** Secant Method Data Table

### 3.4 The Method of False Position (Regula Falsi)

The method of false position (also referred to as regula falsi) results from an attempt to combine the excellent global reliability of the method of bisection with the excellent local speed of the secant method. The modification that we make to the method of bisection is actually quite straightforward and rather obvious. Instead of taking the midpoint of the interval \([a, b]\) as our next approximation to the zero of the nonlinear function, \( f \), we take the \( x \)-intercept of the line passing through the points \((a, f(a))\) and \((b, f(b))\). This means that we replace the formula

\[
c = \frac{a + b}{2}
\]

with the formula

\[
c = b - \frac{(b - a)f(b)}{f(b) - f(a)}.
\] (3.9)

It should be clear that the method of false position will be as reliable as the method of bisection. While it has the potential of taking steps that are essentially the same as steps that would be taken by the secant method, it will in general not be significantly faster than the method of bisection.
We will now apply the method of false position to the same problem of finding $\sqrt{13}$ that we used to illustrate the other three methods, using once again the stopping criteria of $|f(x_n)| \leq 0.0001$. We will choose our beginning interval $[a, b]$ to be the same as in the example for the bisection method, $[a, b] = [2, 4]$. Now that we have the starting interval, we substitute these values into equation (3.9) to find our improved estimate of $x^*$. This gives us $c = 3.5$ and evaluating $f(c)$, we find that $f(c) = -0.75$. If we replace $a$ with $c$, so that $f(a)f(b) < 0$, our new interval becomes $[a, b] = [3.5, 4]$. Using equation (3.9) on this interval, we find $c = 3.6$ and $f(c) = -0.04$. Once again, we replace $a$ with $c$ so that $f(a)f(b) < 0$. Again, we use equation (3.9) to find an improvement on our zero, this time finding that $c = 3.6053$, so $f(c) = -0.0018$. We are quickly approaching our tolerance level. We again replace $a$ with $c$ and $f(a)$ with $f(c)$. This time we find that substituting $a, b, f(a)$ and $f(b)$ into equation (3.9) gives us $c = 3.60554$, so $f(c) = -0.00008$ - below our tolerance level. For this particular case, you will notice that the method of false positioning converged quite quickly, as is shown in table 3.4.

<table>
<thead>
<tr>
<th>i</th>
<th>a</th>
<th>b</th>
<th>$f(a)$</th>
<th>$f(b)$</th>
<th>c</th>
<th>$f(c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>-9</td>
<td>3</td>
<td>3.5</td>
<td>-0.75</td>
</tr>
<tr>
<td>2</td>
<td>3.5</td>
<td>4</td>
<td>-0.75</td>
<td>3</td>
<td>3.6</td>
<td>-0.04</td>
</tr>
<tr>
<td>3</td>
<td>3.6</td>
<td>4</td>
<td>-0.04</td>
<td>3</td>
<td>3.6053</td>
<td>-0.0018</td>
</tr>
<tr>
<td>4</td>
<td>3.6053</td>
<td>4</td>
<td>-0.0018</td>
<td>3</td>
<td>3.60554</td>
<td>0.000081</td>
</tr>
</tbody>
</table>

Table 3.4 False Positioning Method Data Table

### 3.5 A Comparison of Methods

#### 3.5.1 Difficulties

In all of the above methods, we have demonstrated how each method finds a single zero of some function $f(x)$. However, in many cases, as in our function, $f(x) = x^2 - 13$, the function has more than one zero. Unfortunately, each method that we described above will only find one zero at a time. In fact, attempting to find multiple zeros could greatly increase the complexity of the method.

For the bisection method, we know that if $f(a)f(b) < 0$, then there must be at least one zero of the function in the interval $[a, b]$. In reality, it is possible that there
are more than one zero in this interval. It is also possible that if $f(a)f(b) > 0$, then there are an even number of zeros in the interval $[a, b]$ (or none). The bisection method will always find a zero in an interval $[a, b]$ with the property $f(a)f(b) < 0$. However, if there are several roots, several bisections may be necessary in order to find an interval with this behavior. The complexity of the bisection method to avoid finding the same root more than once is very great.

Another difficulty lies in functions that are tangent to the $x$-axis. Consider the function $f(x) = \sin x + 1$, as shown in Figure 3.6. While this function has many zeros, it never actually crosses the $x$-axis. For any interval $[a, b]$ that we chose, we would never find $f(a)f(b) < 0$, yet a zero does exist. The bisection method will neither locate these zeros nor indicate their presence.

![Graph of $f(x) = \sin x + 1$](image)

Figure 3.6 a function with multiple tangent zeros

Newton’s method also leads to difficulty in finding multiple roots. Consider the function shown in Figure 3.7. In this oscillatory function, the first guess, $x_0$, is relatively close to the zero found at A. However, the tangent line strikes the $x$-axis at $x_1$, which leads to a new tangent line coming closer to the zero at B. In this case, although we began “close” to the zero at A, Newton’s method will converge to the different, more distant root at B. Newton’s method will also encounter difficulty when attempting to find zeros where the function is tangent to the $x$-axis as in Figure 3.6, although it will find these zeros, unlike the bisection method. For zeros where the function is tangent to the $x$-axis, it can be shown that as $f(x)$ approaches zero, so
Figure 3.7 Newton's method will find B although starting close to A

does $f'(x)$. The rate of convergence in such cases is very slow, and in practice can make the convergence of multiple roots expensive and slow.

Both the secant method and Newton's method have difficulties locating roots, single or multiple, when a poor starting location is chosen. Consider Figure 3.8. In this case, it is apparent that the function has a zero. Yet in this case, by choosing our starting point $f(x_0)$ as we did, we find that both the secant method and Newton's method will oscillate back and forth on this interval where there is no zero, causing both methods to converge very slowly, or in some cases, not at all.

One drawback of the method of false position is that stagnation can occur at an endpoint - that is, one end of successive intervals does not move from the original endpoint. In this case, approximation to the zero converges from the one side only, slowing down the convergence. This can occur particularly when the interval is large or when the function $f$ deviates significantly from a straight line. A modification of the method of false positioning halves the function value of the stagnant endpoint if that endpoint has been repeated two times or more.

In most cases, the best method of finding the initial $x_0$ may often be prior knowledge of the function and the approximate location of the zeros. The approximate behavior of the function can be determined from a graph of the function, as we did using the graphing calculator. It can also be determined by a computer tabulation of the function at reasonably fine intervals. Even simple hand calculations or rough plotting can help you identify troublesome intervals with multiple or tangent zeros.
Once we move out of the class of single variable, nonlinear functions into a more complicated class with even as few as two variables, we run into problems using several of these methods. Because the method of bisection and the method of false positioning require the use of comparisons, it is not at all obvious how to adapt them for use in the two variable situation. The obvious secant method extension was found to be unstable when moving to a two variable situation. Newton's method extends quite nicely to the multivariable cases.

3.5.2 Further Comparisons - Rates of Convergence

So far, we have discussed four methods of finding zeros of nonlinear functions. We have also discussed certain functions that may cause difficulty for each of these methods. You may be asking at this point "What are some of the advantages for any of these methods - Which method should I use?"

As we stated early in the chapter, and as you may have noticed by now, the bisection method is by far the slowest of the four methods yet the most reliable because it relies on simple comparisons to find the zeros of a function $f$. It is possible to compute the number of steps required for the bisection method using a given tolerance, $\epsilon$. The number of steps can be found using the formula $n > \frac{\log(b-a) - \log 2\epsilon}{\log 2}$. At each iteration of the bisection method, we have one simple calculation of finding the midpoint of the interval and one function evaluation. The amount of work or
computer expense for the bisection method depends largely on the complexity of the function, \( f \), for which we are trying to find the zeros.

For our example problem, Newton’s method converged quite quickly. In fact, it can be shown that if our starting point, \( x_0 \), is well chosen, (close to a solution) then Newton’s method will converge quadratically. Informally, we can say that if at the \( i^{th} \) step our current value for \( x^* \) is accurate to the \( k^{th} \) decimal place, then in the \((n + 1)^{st}\) step, it will be accurate to the \( 2k^{th} \) decimal place. In other words, if we choose a good starting point, then for each step of Newton’s method, we will receive twice as many digits of accuracy than we had at the previous step. Newton’s method is extremely fast and quite effective at finding the zeros of many nonlinear functions. Here again, the amount of computer work varies with the complexity of the function. In Newton’s method, we have two function evaluations at \( x_n - f(x_n) \) and \( f'(x_n) \). Depending on the complexity of our nonlinear function \( f \), this could require a significant amount of work at each step, possibly making it more expensive than some of the other methods.

In considering the convergence of Newton’s method, we stated that it was said to converge quadratically. While Newton’s method would give us approximately twice the number of accurate places for each new iteration, the secant method can be shown to give \( \frac{1 + \sqrt{5}}{2} \) or \( \approx 1.62 \) times as many accurate digits. While this is not quite as good as Newton’s method, the secant method has an advantage over Newton’s method in that in each step, the secant method requires only one function evaluation\(^3\). If we consider the work spent in one step of Newton’s method to be approximately equal to the work of two steps of the secant method, we can see that the secant method will give us more accuracy after two steps than Newton’s method will give us in one step after a similar amount of work. (Two steps of the secant method will give us an answer 3.236 times more accurate than \( x_{n-1} \), whereas one step of Newton’s method will give us an answer that is only 2 times as accurate.)

\(^3\)While the formula requires both \( f(x_n) \) and \( f(x_{n-1}) \) to find the new value for \( x \) at each iteration, recall that \( f(x_{n-1}) \) has already been computed for us in the previous iteration
Chapter 3 Problems

As was stated earlier, the best method for finding initial values $x_0$ may be through prior knowledge of the function and the approximate location of the zeros. Use your graphing calculator to determine the intervals $[a, b]$ or the necessary initial values.

1.) [Nakamura 3.7, 103] Find all positive roots of the following equations by the bisection method with a tolerance of 0.0001.
   a.) $\tan x - x + 1 = 0, 0 < x < 3\pi$
   b.) $\sin x - 0.3e^x = 0, x > 0$
   c.) $x^3 + x + 1 = 0$
   d.) $16x^5 - 20x^3 + x^2 + 5x - 0.5 = 0$

2.) [Nakamura, 3.2, 102] Find the positive root for $x\sin x - 0.1 = 0, 0 < x < 1.0$. by the bisection method with a tolerance of 0.0001.

3.) [Nakamura, 3.1, 102] Find the positive root for $x^2 - 0.9x - 1.52 = 0$ in the interval $[1, 2]$ by the bisection method with a tolerance of 0.0001.

4.) Find all of the positive roots of the function $f(x) = \cos x - \cos 3x$, given that the function has zeros tangent to the x-axis at $x = 0$ and $x = 2\pi$, using the bisection method.

5.) Find where the graphs of $y = 3x$ and $y = e^x$ intersect by finding roots of $f(x) = e^x - 3x = 0$ correct to four decimals using the bisection method.

6.) Two of the four zeros of $x^4 + 2x^3 - 7x^2 + 3$ are positive. Find them by Newton’s method, stopping when $|f(x_n)| \leq 0.001$. [For this function, $f'(x) = 4x^3 + 6x^2 - 14x$.]

7.) [Nakamura, 3.16] Find the roots of the equations in 1. by Newton’s method with a tolerance of 0.0001. The derivatives for each of the functions are given below.
   a.) $f'(x) = \sec^2 x - 1$
   b.) $f'(x) = \cos x - 0.3e^x$
   c.) $f'(x) = 3x^2 + 1$
   d.) $f'(x) = 80x^4 - 60x^2 + 2x + 5$

8.) [Buchanan, 2.3.1, 69] Perform the first 5 iterations of Newton’s method to find the solution of $x = \cos x$ from an initial point $x_0$. [$f'(x) = -\sin x - 1$]

9.) [Nakamura, 3.19] Find the root of $f(x) = \sin x - x + 1$ that is know in $1 < x < 3$ by Newton’s method. Stop calculations after 4 iterations. [$f'(x) = \cos x - 1$]
10.) [Nakamura 3.13, 104] Find the roots of the following equations by Newton’s method. Stop calculation after 4 iterations.

a.) \( f(x) = 0.5e^{\frac{x}{5}} - \sin x, x > 0 \)
\[ f'(x) = \frac{1}{5}e^{\frac{x}{5}} + \cos x \]

b.) \( f(x) = e^{x} - 5x^{2} \)
\[ f'(x) = e^{x} - 10x \]

c.) \( f(x) = x^{3} + 2x - 1 = 0 \)
\[ f'(x) = 3x^{2} + 2 \]

d.) \( f(x) = \sqrt{x + 2} \)
\[ f'(x) = \frac{1}{\sqrt{x + 2}} \]

11.) Find a root of \( f(x) = \cos x - x \) in the range \( 0 \leq x \leq \pi/2 \) by using the secant method. Stop when \( |f(x_n)| \leq 0.0001 \).

12.) Find a real root of \( f(x) = x^{3} - 3.23x^{2} - 5.54x + 9.84 \) by using the secant method with \( x_{-1} = 0.9 \) and \( x_{0} = 1.0 \). Stop when \( |f(x_n)| \leq 0.0001 \).


15.) The formula given to find \( c \) in the method of false position was

\[ c = b - \frac{(b - a)f(b)}{f(b) - f(a)}. \]

Prove that this is the same as using the formula

\[ c = a - \frac{(a - b)f(a)}{f(a) - f(b)}. \]

16.) Find \( \sqrt{3} \) by finding a solution of \( x^{2} - 3 = 0 \) using any method with an initial guess of \( x_{0} = 2 \). Stop when \( |f(x_n)| \leq 0.000001 \). [If you choose to use Newton’s method, \( f'(x) = 2x \)]

17.) Find \( \sqrt[3]{75} \) by finding a solution to \( x^{3} - 75 = 0 \) using any method with an initial guess of \( x_{0} = 4 \). Stop when \( |f(x_n)| \leq 0.000001 \). [If you choose to use Newton’s method, \( f'(x) = 3x^{2} \)]

18.) Find \( \sqrt[7]{7} \) by solving \( x^{2} - 7 = 0 \) using the four root solving methods in this chapter. Compare the number of steps for each method. Retain 6 digits of accuracy at each step and stop when \( |f(x_n)| \leq 0.001 \).

20.) [Nakamura, 3.22] An equimolar mixture of carbon monoxide and oxygen attains equilibrium at 300 K and 5 atm pressure. The theoretical reaction is

\[ \text{CO} + \frac{1}{2}\text{O}_2 \rightleftharpoons \text{CO}_2. \]

The actual chemical reaction is written as

\[ \text{CO} + \text{O}_2 \rightarrow x\text{CO} + \frac{1}{2}(1 + x)\text{O}_2 + (1 - x)\text{CO}_2. \]

The chemical equilibrium equation to determine the fraction of the remaining CO, namely \( x \), is written as

\[ K_p = \frac{(1 - x)(3 + x)^{1/2}}{x(x + 1)^{1/2}P^{1/2}}, \quad 0 < x < 1 \]

where \( K_p = 3.06 \) is the equilibrium constant for \( \text{CO} + \frac{1}{2}\text{O}_2 + \text{CO}_2 \) at 3000° K and \( P = 5 \) is the pressure. [Wark pg 608]. Determine the value of \( x \) using the secant method.

21.) [Buchanan, 2.1.2, 53] Write a program to implement the bisection algorithm. Check the program to check the result quoted below to machine accuracy on your computer.

\[ f(x) = x - \tan x = 0; \quad x = 4.4934094579 \]
Chapter 4

Linear Regression

4.1 The Line of Best Fit

In the physical, social and behavioral sciences, an experiment often produces a set of data points in the plane,

\[(x_1, y_1), \ldots, (x_n, y_n)\]

(4.1)

Graphing this set of data could result in the following scatterplot. Suppose that it is known that the relationship underlying the generation of this data is linear, that is, that the relationship can be described by a linear expression of the form

\[y = c_1 x + c_0\]

(4.2)

for some currently unknown coefficients \(c_0\) and \(c_1\). Any data point failing to fall on the line determined by the relationship (4.2) would be due to experimental error. The goal is to recover a best estimate of \(c_0\) and \(c_1\) using the data at hand.
If there was no experimental error in our data, then we know that \( c_0 \) and \( c_1 \) would solve the system of linear equations

\[
\begin{align*}
  c_0 + c_1 x_1 - y_1 &= 0 \\
  c_0 + c_1 x_2 - y_2 &= 0 \\
  \vdots \ \\
  c_0 + c_1 x_n - y_n &= 0
\end{align*}
\]  

(4.3)

In fact, since we are looking for only 2 unknowns, \( (c_0 \text{ and } c_1) \), we could find \( c_0 \) and \( c_1 \) by solving any two of the equations (4.3) for \( c_0 \) and \( c_1 \). However, it is very improbable that we would have no error in our data, so we will not be able to find \( c_0 \) and \( c_1 \) satisfying each equation in (4.3). We say that this system is overdetermined. We could still find \( c_0 \) and \( c_1 \) by choosing any two of the equations, but this would be equivalent to finding the line that passes through two of the points and ignoring the remaining points. The dilemma becomes finding which two points best represent the relationship of the data at hand and we have no way of knowing the answer to this question. So we must use a method that considers all of the data points in determining the relationship (4.2)

The system of equations (4.3) represents the situation where no experimental error exists. Realizing that this is a rather naive assumption, we need a way to represent the situation when there is error in an experiment. For a given \( c_0 \) and \( c_1 \), the error at the \( i^{th} \) data point is \( |c_0 + c_1 x_i - y_i| \). If we let \( \delta_i \) represent the error (also called the residual) at the point \( (x_i, y_i) \), then we can write

\[
\begin{align*}
  |c_0 + c_1 x_1 - y_1| &= \delta_1 \\
  |c_0 + c_1 x_2 - y_2| &= \delta_2 \\
  \vdots \\
  |c_0 + c_1 x_n - y_n| &= \delta_n
\end{align*}
\]  

(4.4)

Since we know that expecting \( \delta_i \) to equal zero is unrealistic, we will choose \( c_0 \) and \( c_1 \) so that the \( \delta_i \)'s are small in some sense. In this case, the question becomes “What is small?”

For \( 1 \leq p < +\infty \) consider the following function:

\[
G(c_0, c_1)_p = \delta_1^p + \delta_2^p + \ldots + \delta_n^p
\]
For \( p = \infty \) we write
\[
G(c_0, c_1)_{\infty} = \max_{1 \leq i \leq n} |\delta_i|.
\]
For a fixed \( c_0 \) and \( c_1 \), it can be shown that \( G(c_0, c_1)_p \) approaches \( G(c_0, c_1)_{\infty} \) as \( p \) approaches \( \infty \); therefore our notation is consistent. Notice that the function \( G(c_0, c_1)_p \) takes all of the data points into consideration. We now ask for \( c_0 \) and \( c_1 \) that minimizes \( G(c_0, c_1)_p \) for a given, fixed \( p \). The resulting line is then called the line of best fit or the line of regression in the \( l_p \) sense, or simply the best \( l_p \) fit. Different values of \( p \) lead to differences in the line of best fit and significantly affect our solution technique. This can be illustrated by the following example.

Suppose that our data set (4.1) actually led to the following graph where all the

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{scatter_plot.png}
\caption{illustrative scatter plot}
\end{figure}

data points except one fall on the line \( y = c \) for some \( c \). Consider the problem of doing a best \( l_p \) fit to a constant line of the form \( y = c \). We can see that
\[
G(c)_p = |y_1 - c|^p + \cdots + |y_n - c|^p \text{ for } 1 \leq p < \infty
\]
and
\[
G(c)_{\infty} = \max_{1 \leq i \leq n} |y_i - c|.
\]
From this, we can argue that the best \( l_p \) fit in this application can be described by the situation in Figure (4.3), where the best \( l_p \) fit for \( p \) not equal to 1, 2, or \( \infty \) would fall between the lines for \( p = 1, 2, \) or \( \infty \). There are certain important things to observe from this picture. The best \( l_1 \) fit actually ignores one point in favor of a line passing through the remaining points. For this reason, computational scientists often say that
Figure 4.3 representation of various lines of best $l_p$ fit

this best fit ignores outliers, or points that seem to be relatively far away from the remaining points. All of the error is then concentrated at that one point or outlier. At the other extreme, the best $l_\infty$ fit placed the best fit line so that all of the points have the same error. The best $l_p$ fit for $2 \leq p < \infty$ placed the line in between the two extremes. In actual practice, computational scientists often restrict their attention to the choices $p = 1$, $p = 2$, and $p = \infty$. In application, if you knew that only a few points had errors, the $l_1$ fit would be used. If you knew that all of your points had similar or equal errors, then the $l_\infty$ fit would be used. However, due to practical, computational considerations, scientists usually use $p = 2$.

The best $l_2$ fit is called the least squares fit because for $p = 2$ we are minimizing the sum of the squares of the errors. Only for $p = 2$ can we solve the regression problem by considering a system of linear equations. For all of the other $p$, the optimization problem leads to a much more complicated problem. However, as you will learn in Chapter 5, for $p = 1$ and $p = \infty$, the problem of finding the line of best fit can be stated equivalently as a linear programming problem. While it is possible to show that $p = 2$ is optimal for error distributions of a particular form, (errors that are normally distributed,) computational scientists use $p = 2$ because it leads to solving a system of linear equations - a most effective solution technique. Indeed, this is clearly the motivation that led Gauss to suggest the technique of least squares when he was a very young man.
4.2 Least Squares Regression

From this point on in this chapter we will be concerned with finding the line of best fit using the method of least squares, or the line of best \( l_2 \) fit. Consider the data in Table 4.1. In this case we want to find a linear equation, \( y = c_1 x + c_0 \) that best represents the relationship for this set of data in the least squares sense. We begin by substituting the data points in for \( x \) and \( y \), to obtain the following overdetermined system of linear equations.

\[
\begin{align*}
    c_0 + 30c_1 &= 73 \\
    c_0 + 20c_1 &= 50 \\
    c_0 + 60c_1 &= 128 \\
    c_0 + 80c_1 &= 170 \\
    c_0 + 40c_1 &= 87 \\
    c_0 + 50c_1 &= 108 \\
    c_0 + 60c_1 &= 135
\end{align*}
\]  

(4.5)

The overdetermined linear system (4.5) can be written in matrix form as

\[
Ac = y
\]  

(4.6)

where

\[
A = \begin{bmatrix}
1 & 30 \\
1 & 20 \\
1 & 60 \\
1 & 80 \\
1 & 40 \\
1 & 50 \\
1 & 60
\end{bmatrix}
\]

\[
c = \begin{bmatrix}
c_0 \\
c_1
\end{bmatrix}
\]

and

\[
y = \begin{bmatrix}
73 \\
50 \\
128 \\
170 \\
87 \\
108 \\
135
\end{bmatrix}
\]
Now, it is well-known that a least squares solution of the overdetermined linear system (4.6) must satisfy the so-called normal equations

\[ A^T A c = A^T y \]  

(4.7)

and conversely, any solution of the normal equations (4.7) is a least squares solution of the overdetermined linear system (4.6). For a proof in the case that the least squares solution is unique, (the usual case), see Appendix B. For the problem at hand, the normal equations are

\[
\begin{bmatrix}
1 & 30 \\
1 & 20 \\
1 & 60 \\
30 & 20 & 60 & 80 & 40 & 50 & 60
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1
\end{bmatrix} =
\begin{bmatrix}
73 \\
50 \\
128 \\
170 \\
87 \\
108 \\
135
\end{bmatrix}.
\]

or

\[
\begin{bmatrix}
7 & 340 \\
340 & 19000
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1
\end{bmatrix} =
\begin{bmatrix}
751 \\
41450
\end{bmatrix}.
\]

What you can now see is that this leads us to a set of equations that we can solve. When we solve this system of equations, we find the coefficients for our line of best fit to be

\[ c = \begin{bmatrix} 10.1149 \\ 2.0006 \end{bmatrix} \]

The equation for the line is

\[ y = 10.1149 + 2.0006x \]

Suppose that your data would fit better to a parabola than a line. Consider the following data from a different experiment. In this case, a parabola would appear to be a better fit than the straight line. Least squares approximation can allow for this as well. The general equation for a parabola is \( y = c_0 + c_1 x + c_2 x^2 \). As before, we begin with the following overdetermined system of equations

\[
c_0 - 1c_1 + 1c_2 = 4
\]

\[
c_0 + 0c_1 + 0c_2 = 2
\]

<table>
<thead>
<tr>
<th>x</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>
Figure 4.4  best fit parabola

\[
\begin{align*}
c_0 + c_1 + c_2 &= 0 \\
c_0 + 2c_1 + 4c_2 &= 2 \\
c_0 + 3c_1 + 9c_2 &= 4
\end{align*}
\]

In matrix form we have \(Ax = b\) where

\[
A = \begin{bmatrix}
1 & -1 & 1 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
1 & 2 & 4 \\
1 & 3 & 9
\end{bmatrix},
\quad
b = \begin{bmatrix}
4 \\
2 \\
0 \\
2 \\
4
\end{bmatrix},
\quad
x = \begin{bmatrix}
c_0 \\
c_1 \\
c_2
\end{bmatrix}.
\]

The normal equations for this problem are \(A^TAx = A^Tb\) or

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 2 & 3 \\
1 & 0 & 1 & 4 & 9 \\
1 & 3 & 9 & 2 & 2
\end{bmatrix} \begin{bmatrix}
c_0 \\
c_1 \\
c_2
\end{bmatrix} = \begin{bmatrix}
4 \\
2 \\
0 \\
2 \\
4
\end{bmatrix}.
\]

which simplify to

\[
\begin{bmatrix}
5 & 5 & 15 \\
5 & 15 & 35 \\
15 & 35 & 99
\end{bmatrix} \begin{bmatrix}
c_0 \\
c_1 \\
c_2
\end{bmatrix} = \begin{bmatrix}
12 \\
12 \\
48
\end{bmatrix}.
Solving the normal equations leads us to
\[
\begin{bmatrix}
  c_0 \\
  c_1 \\
  c_2 
\end{bmatrix} =
\begin{bmatrix}
  1.5429 \\
  -1.7143 \\
  0.8571 
\end{bmatrix}
\]
and therefore our least squares parabola is \( y = 1.5429 - 1.7143x + 0.8571x^2 \).

You have seen how we can use least squares regression techniques to find a line of best fit and a parabola of best fit. These two examples motivate the following very important summary comments.

Least squares regression as we have used it (often called linear least squares) can be used whenever the parametric form of the underlying functional relationship is linear in the unknown parameters (coefficients). This means that the underlying function has the form
\[
f(x) = c_1 f_1(x) + c_2 f_2(x) + \ldots + c_n f_n(x) \quad (4.8)
\]
where the \( c_i \)'s are the unknown coefficients that are to be determined and the \( f_i \)'s are known functions. The \( f_i \)'s can have any form and certainly do not have to be linear. Moreover, \( x \) does not have to be a single variable, it can be a vector variable and have more than one component.

Now, given the data set \((x_1, y_1), \ldots, (x_n, y_n)\) and a functional form \((4.8)\), (i.e. specific functions \( f_i \)) we consider the system of linear equations
\[
\begin{align*}
  c_1 (x_1) + \ldots + c_m (x_n) &= y_1 \\
  \cdots \\
  c_1 (x_n) + \ldots + c_m (x_n) &= y_n
\end{align*} \quad (4.9)
\]
In matrix form, \((4.9)\) can be written
\[
Ac = y \quad (4.10)
\]
where
\[
A = \begin{bmatrix}
  f_1 (x_1) & \ldots & f_m (x_1) \\
  \vdots & \ddots & \vdots \\
  f_1 (x_n) & \ldots & f_m (x_n)
\end{bmatrix}, \quad
\begin{bmatrix}
  c_1 \\
  \vdots \\
  c_m
\end{bmatrix}, \quad \text{and} \quad
\begin{bmatrix}
  y_1 \\
  \vdots \\
  y_n
\end{bmatrix}
\]
In general, \((4.10)\) will be very much overdetermined (i.e. \( n \) will be much larger than \( m \)) and will not have solutions. So we are willing to accept, instead of a solution, the
so-called least squares solution which minimizes the sum of the squares of the residuals 
\((f(x_i) - y_i)\). Only in rare cases will a least squares solution of (4.10) actually be a
solution. So don’t fall into the trap that even many good college students fall into,
and think that a least squares solution is necessarily a solution. The reason that
least squares is so extremely useful and effective follows directly from the beautiful
and elegant mathematical fact that the least squares solutions of the overdetermined
system \(Ac = y\) and the solutions of the so-called normal equations \(A^TAc = A^Ty\) are
exactly the same.

Let us consider two examples. One researcher we know was studying the effects
of temperature and charge rate on the life of a battery. The equation that he used to
study this phenomena was

\[ y = c_1 + c_2x_1 + c_3x_2 + c_4x_1^2 + c_5x_2^2 + c_6x_1x_2 \]  

(4.11)

where \(x_1\) represents the charge rate and \(x_2\) represents the temperature. If we put
(4.11) in the context of (4.8), we see that \(x = (x_1, x_2)\), a vector variable, and \(f_1(x_1, x_2) = 1, f_2(x_1, x_2) = x_1, f_3(x_1, x_2) = x_2, f_4(x_1, x_2) = x_1^2, f_5(x_1, x_2) = x_2^2, \) and \(f_6(x_1, x_2) = x_1x_2\). In order to plot the data in this case we would need a three dimensional plot.
This is a good example where a computer graphics package would be useful.

Another example is one in which a researcher was concerned with the relation
of amount of body fat to skinfold thickness \((x_1)\), thigh circumference \((x_2)\), and mid-
darm circumference \((x_3)\). One model that was considered was based on the following
equation

\[ y = c_1 + c_2x_1 + c_3x_2 + c_4x_3. \]

In this application, we see that in (4.8) we have \(x = (x_1, x_2, x_3)\), a vector in three vari-
ables, and \(f_1(x_1, x_2, x_3) = 1, f_2(x_1, x_2, x_3) = x_1, f_3(x_1, x_2, x_3) = x_2, f_4(x_1, x_2, x_3) = x_3.\)

We end this chapter with the comment that the technique of least squares solution
is one of the more useful linear algebra tools in today’s scientific world.
Chapter 4 Problems

1.) What line best represents these data in the least squares sense?

\[
\begin{array}{c|c|c|c}
 x & 0 & 1 & 2 \\
 y & 5 & -6 & 7 \\
\end{array}
\]

2.) [Nakamura 8.1, 286] Determine a linear function fitted to the following data points by the least squares method.

\[
\begin{array}{c|c|c|c|c}
 x & 1.0 & 1.5 & 2.0 & 2.5 & 3.0 \\
 y & 2.0 & 3.2 & 4.1 & 4.9 & 5.9 \\
\end{array}
\]

3.) [Morris, 5.1, 155] Calculate the line that is the least squares approximation to the data in the following table. Graph your approximation.

\[
\begin{array}{c|c|c|c|c|c|c|c|c}
 x & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 \\
 y & 0 & 2.1220 & 3.0244 & 3.2568 & 3.1399 & 2.8579 & 2.5140 & 2.1639 & 1.8358 \\
\end{array}
\]
4.) Find the equation of a parabola of the form \( y = ax^2 + bx + c \) that best represents the following data. Use the method of least squares.

\[
\begin{array}{c|c|c|c}
  x & -1 & 0 & 1 \\
  y & 3.1 & 0.9 & 2.9 \\
\end{array}
\]

5.) Suppose that the following table is expected to conform to a function of the form \( y = x^2 - x - c \). What value of \( c \) is obtained by least squares theory?

\[
\begin{array}{c|c|c|c}
  x & 1 & 2 & 3 \\
  y & \frac{1}{4} & \frac{1}{3} & \frac{5}{12} \\
\end{array}
\]

6.) Find the quadratic polynomial that best fits the following data in the least squares sense.

\[
\begin{array}{c|c|c|c|c|c|c}
  x & -2 & -1 & 0 & 1 & 2 \\
  y & 2 & 1 & 1 & 1 & 2 \\
\end{array}
\]
7.) [Nakamura 8.3] Fit a quadratic polynomial to the following data set.

\[
\begin{array}{c|c|c|c|c}
 x  & 0  & 1  & 2  & 3 \\
 y  & 1  & 0  & 0  & 2 \\
\end{array}
\]

8.) [Nakamura 8.4] Fit a quadratic polynomial to the following data set.

\[
\begin{array}{c|c|c|c|c|c|c|c}
 x  & 0  & 1  & 2  & 3  & 4  & 5  & 6 \\
 y  & 0  & 2.3 & 4.2 & 5.7 & 6.5 & 6.9 & 6.8 \\
\end{array}
\]

9.) [Morris, 5.2] Calculate the quadratic least squares approximation to the data in problem 3. Graph your approximation.

10.) [Nakamura 8.9] Fit \( g(x) = a_0 + a_1 x + a_2 \sin(\pi x) + a_3 \sin(2\pi x) \) to the following table.

\[
\begin{array}{c|c|c|c|c|c|c}
 x  & 0.0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 \\
 y  & 0.11 & 0.251 & 0.352 & 0.453 & 0.54 & 0.66 \\
\end{array}
\]
11.) Find the best function (in the least squares sense) of form \( f(x) = a \sin \pi x + b \cos \pi x \) that fits these data points.

\[
\begin{array}{c|c|c|c|c}
 x & -1 & -\frac{1}{2} & 0 & \frac{1}{2} & 1 \\
y & -1 & 0 & 1 & 2 & 1 \\
\end{array}
\]

12.) [Nakamura ex 8.3, 281] Determine the coefficients of the function

\[
g(x) = a_1 + a_2 x + a_3 \sin x + a_4 e^x
\]

fitted in the least squares sense to the data in the following table.

\[
\begin{array}{c|c|c|c|c|c|}
x & 0.1 & 0.4 & 0.5 & 0.7 & 0.9 \\
y & 0.61 & 0.92 & 0.99 & 1.53 & 1.47
\end{array}
\]

\[
\begin{array}{c}
2.03
\end{array}
\]
Chapter 5

A Few Methods for Linear Programming.

5.1 A brief history

Another area where new algorithms are being developed is the area of linear programming. Unlike calculus and geometry and some of the other branches of math, the area of linear programming has not developed over the centuries, but rather over the decades. Quite possibly because it is so young, changes in the methods and algorithms are taking place continuously. You may be familiar with small examples of linear programs as they are covered in many high school algebra and pre-calculus texts. However, this chapter will attempt to show you that what is taught at the high school level barely skims the surface of the field of linear programs. The use of linear programming theory arises in a wide variety of situations and is researched both in the academic and industrial world alike. The problems vary in size, definition and complexity. Just like with linear systems, computational scientists are continuously working to find faster algorithms that will solve larger problems. They are also working to apply parallel processing to this class of problems and algorithms.

As was said earlier, most of the work in the area of linear programming is only a few decades old. Linear programming had its beginnings around the same time as the computer and World War II. In 1939, L.V. Kantorovich published a paper entitled “Mathematical Methods in the Organization and Planning of Production.” He realized that a large class of production problems led to the same mathematical model. In 1941, Frank Hitchcock formulated the transportation problem, while in 1945, George Stigler formulated the diet problem. Many other problems were developed at this time, many of them being related to the war effort. In 1947, George Dantzig developed the simplex method, which is the method that you will learn to solve linear programming problems. It was one of the very first methods used, but, just as with other methods, many advancements have been made in the area of linear programming and new methods have been developed to solve linear programs. These, however, are beyond the scope of this unit.
Since the 1940's, much work has been done in the area of linear programming. In 1975, the Royal Sweden Academy of Sciences awarded the Nobel Prize in economics science to L. V. Kantorovich and T. C. Koopmans “for their contributions to the theory of optimum allocation of resources.” In 1979, L.G. Khachian published an algorithm for solving linear programming problems that was theoretically, although not numerically, satisfactory. Much of the work has been on specific problem types, such as the traveling salesperson problems, or in developing computational techniques for given types of machines. One of the more recent methods developed has been the interior point method used for solving linear programs. Also, the area of linear programming branches out into another area of more complex problems called mixed integer programs and pure integer programs, where the problem has the restriction that some or all of the variables must be integers, rather than any real number. These will be explained in more detail at the end of the chapter.

With the area of linear programming having so many applications in the areas of industry and economics, it has been important not only to find methods to solve these problems, but to solve them in a timely fashion. The development of computational techniques - especially those for parallel machines - has aided this greatly. Obviously the area of linear programming has come a long way in a short period of time, but improvements and advancements are still being made. Currently, the area of integer programming is receiving much study to improve the existing methods. There are also many large problems that as yet cannot be solved because they are simply too large.

5.2 A relatively familiar example

In your previous algebra or pre-calculus courses, you may have been introduced to the topic of Linear Programming. The linear programming models that exist in most algebra and pre-calculus texts consist of a story problem that has as its model a system of linear inequalities.

You are taking a test in which items of type A are worth 10 points and items of type B are worth 15 points. It takes 3 minutes to answer each item of type A and 6 minutes to answer each item of type B. The total time allowed is 60 minutes, and you may not answer more than 16 questions. Assuming all of your answers are correct, how many items of each type should you answer to get the highest score?
The above problem is an example of a problem that can be modeled as a linear program. If we were to take each of the criteria given and formulate a sentence, we will develop a system of inequalities.

- Let \( x \) = number of items of type A
- Let \( y \) = number of items of type B
- total number of questions allowed, not more than 16: \( x + y \leq 16 \)
- time, not more than 60 minutes: \( 3x + 6y \leq 60 \)
- number of items of type A - nonnegative: \( x \geq 0 \)
- number of items of type B - nonnegative: \( y \geq 0 \)

After you have determined the system of inequalities for this problem, then you may have learned to look at the problem graphically. This is not possible for anything except the most trivial linear programs - with more than three variables to determine, the problem is not easily represented by a graph. For this problem, however, if we graph each of the inequalities, we find a shaded polygon enclosed by our four inequalities, as shown in Figure 5.1. This polygon has the following four vertices:

\[
\begin{align*}
(0,0) \\
(16,0) \\
(12,4) \\
(0,10)
\end{align*}
\]

So far, we have formulated a mathematical model and used the process of graphing in order to attempt to solve the problem. However, if we look back to the problem, we find that we haven’t yet answered the problem. The goal of this problem was to determine how many questions of each type we should answer in order to get the highest score. In other words, we want to maximize the number of points that we get by answering a combination of type A and type B questions. A number sentence that would represent this goal would be as follows:

\[
\text{Maximize } T = 10x + 15y
\]

\( T \) represents our total score.) If we were to take the set of vertices that we found for our polygon and substitute them into this sentence, we find that the vertex \((0,0)\) gives a value of 0, (a minimum for this system) and that the vertex \((12,4)\) gives a
value of 180 (a maximum for this system). Thus we have found that if we answer 12 questions of type A and 4 questions of type B, we will get the highest score possible on this test.

You may wonder why the maximum value for this problem was found at a vertex. Consider the graph of the polygon for this problem as shown in Figure 5.2. We have added contour lines to our graph. Contour lines are the set of all $x$ such that $T(x) = 1$ (the first contour line), $T(x) = 2$ (the second contour line), $T(x) = 3$ (the third contour line) and so on. If you take a close look, the first point on your polygon that any of the contour lines touches is the vertex where you found the maximum, and the last point on your polygon that one of your contour lines touches is the vertex where the minimum would be found. (In this case, the minimum is found at the vertex $(0,0)$) By definition, a linear program consists of maximizing (or minimizing) a linear function subject to a finite number of linear constraints. In the above example, the linear function that we were maximizing was $T$, while the finite number of linear constraints was the system of linear inequalities that we developed in order to draw our polygonal region. The function to be maximized or minimized is also called the objective function. The last two constraints that were used which state that neither of the variables can have a negative value are called non-negativity constraints.

5.3 Linear Programs in Standard Form

As was pointed out in the first chapter of the text, an important phase in problem solving is identifying the problem and formulating a mathematical model. The area
of linear programming has many prime examples with which to practice this skill. As you will find out, most problems are not so simple as to have only 2 or 3 variables with as many constraints.

One of the early linear programming problems as mentioned earlier in the history is the diet problem. Suppose that you are wondering how much money you must spend on food in order to get all the nutrients necessary for a healthy diet. To simplify the problem, we will assume a regime of 2000 kcal, 55 g of protein, and 800 mg of calcium. You have found a list of foods that seem to be a cheap source of necessary nutrients.

<table>
<thead>
<tr>
<th>food</th>
<th>serving</th>
<th>Energy(kc)</th>
<th>Protein</th>
<th>Calcium</th>
<th>Price/serving</th>
</tr>
</thead>
<tbody>
<tr>
<td>oatmeal</td>
<td>28 g</td>
<td>110</td>
<td>4 g</td>
<td>2 mg</td>
<td>3 cents</td>
</tr>
<tr>
<td>chicken</td>
<td>100 g</td>
<td>205</td>
<td>32 g</td>
<td>12 mg</td>
<td>24 cents</td>
</tr>
<tr>
<td>eggs</td>
<td>2 large</td>
<td>160</td>
<td>13 g</td>
<td>54 mg</td>
<td>13 cents</td>
</tr>
<tr>
<td>whole milk</td>
<td>237cc</td>
<td>160</td>
<td>8 g</td>
<td>285 mg</td>
<td>9 cents</td>
</tr>
<tr>
<td>cherry pie</td>
<td>170 g</td>
<td>420</td>
<td>4 g</td>
<td>22 mg</td>
<td>20 cents</td>
</tr>
<tr>
<td>pork &amp; beans</td>
<td>260 g</td>
<td>260</td>
<td>14 g</td>
<td>80 mg</td>
<td>19 cents</td>
</tr>
</tbody>
</table>

Table 5.1 diet problem data
This may appear to be a rather odd choice of foods, but these are the choices for the time. However, you realize that although 10 servings of pork and beans would satisfy all of your nutritional needs at a low price, it does not sound very appetizing to you. So you restrict the number of servings that you will allow yourself per day.

<table>
<thead>
<tr>
<th>Oatmeal</th>
<th>at most 4 servings per day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicken</td>
<td>at most 3 servings per day</td>
</tr>
<tr>
<td>Eggs</td>
<td>at most 2 servings per day</td>
</tr>
<tr>
<td>Milk</td>
<td>at most 8 servings per day</td>
</tr>
<tr>
<td>Cherry pie</td>
<td>at most 2 servings per day</td>
</tr>
<tr>
<td>Pork and beans</td>
<td>at most 2 servings per day</td>
</tr>
</tbody>
</table>

**Table 5.2** serving size

Now if we check some of our combinations with the restrictions in table 5.2, we find that 8 servings of milk with 2 servings of cherry pie will meet our nutritional need at a cost of only $1.12. There are many other combinations that could be tried, but we will try a more systematic approach to solving this problem. We will let the variables represent the various servings of food - i.e. $x_1$ servings of oatmeal, $x_2$ servings of chicken, $x_3$ servings of eggs, $x_4$ servings of milk, $x_5$ servings of cherry pie, and $x_6$ servings of pork and beans. We have just begun the process of formulating the mathematical model by identifying the variables for this problem.

For a linear programming problem, there are two parts to determining the needed equations, as you saw in the earlier example. First we need to develop the system of constraints. In order to satisfy our above serving limits, our menu must satisfy the following inequalities:

\[
0 \leq x_1 \leq 4
\]
\[
0 \leq x_2 \leq 2
\]
\[
0 \leq x_3 \leq 2
\]
\[
0 \leq x_4 \leq 8
\]
\[
0 \leq x_5 \leq 2
\]
\[
0 \leq x_6 \leq 2
\]
Then our requirements for the appropriate amount of nutrients yield the following inequalities:

\[ 110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 > 2000 \]
\[ 4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 > 55 \]
\[ 2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 > 800 \]

After we have set up our set of constraints, we have to then develop the objective function. Remember that the goal of a linear function is to minimize or maximize a linear function subject to a set of linear constraints. In this particular problem, when we reread the problem, we notice that the original idea was to design a diet that would fulfill the nutritional requirements at the lowest cost. We have already set up inequalities to satisfy the nutritional requirements, so all that is left is to minimize the cost. The function that expresses this is

\[ 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6 \]

We now have the mathematical model formulated for this problem. A computational scientist would put this in standard form as follows:

\[
\begin{align*}
\text{minimize} & \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6 \\
\text{subject to:} & \quad 0 \leq x_1 \leq 4 \\
& \quad 0 \leq x_2 \leq 3 \\
& \quad 0 \leq x_3 \leq 2 \\
& \quad 0 \leq x_4 \leq 8 \\
& \quad 0 \leq x_5 \leq 2 \\
& \quad 0 \leq x_6 \leq 2 \\
& \quad 110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \geq 2000 \\
& \quad 4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \geq 55 \\
& \quad 2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \geq 800
\end{align*}
\]

Some other examples of linear programs written in standard form are

\[
\begin{align*}
\text{minimize} & \quad 3x_1 + 2x_2 + 4x_3
\end{align*}
\]
subject to \[ 30x_1 + 100x_2 + 85x_3 = 2500 \]
\[ 6x_1 + 2x_2 + 3x_3 = 90 \]
\[ x_1, x_2, x_3 \geq 0 \]

or

maximize \[ 3x_1 - 2x_2 - x_3 + x_4 - 87 \]
subject to \[ 4x_1 - x_2 + x_4 \leq 6 \]
\[ -7x_1 + 8x_2 + x_3 \geq 7 \]
\[ x_1 + x_2 + 4x_4 = 12 \]
\[ x_1, x_2, x_3, x_4 \geq 0 \]

In general, the objective function can be written as the following linear function with real variables:

\[ z(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots c_nx_n = \sum_{j=1}^{n} c_jx_j \]

where \( x_k \) represents real numbers and \( c_k \) is the coefficient correspond with \( x_k \). The constraints can be written more generally as

\( f(x_1, x_2, \ldots, x_n) = b \) if they are linear equations

or

\( f(x_1, x_2, \ldots, x_n) \geq b \) if they are linear inequalities.

If we were to write a linear program in general standard form, we would have

maximize \[ \sum_{j=1}^{n} c_jx_j \]
subject to \[ \sum_{j=1}^{n} a_{ij}x_j \leq b_i \quad (i = 1, 2, \ldots, m) \]
\[ x_j \geq 0 \quad (j = 1, 2, \ldots, n) \]

We can also call the above form canonical form or symmetric form.

### 5.4 The Simplex Method

In the problems that were previously solved, the first problems were solved by the graphing method. This is a simple way to work with linear programs that have
at most two variables and a relatively small number of constraints. However, most problems are not two-dimensional (containing two variables). The problem set of the last section had several problems with three and four variables or more, and while you may be able to graph in three dimensions, there is no simple way to draw a figure in four or more dimensions. Most problems in the area of linear programming cannot be solved by simply graphing the constraints and testing the vertices in order to find the maximal or minimal solution. Because of their higher dimensions, graphing the problem and checking each vertex would be more difficult than trying to find another way to get the solution. Consider the following: in order to find the maximum or minimum using the graphical method, you would have to know and test each one of the vertices formed where two constraints intersected. A formula that would allow you to find the number of vertices is as follows.

\[
\frac{(\text{number of variables})! \times (\text{number of variables} - \text{number of constraints})!}{(\text{number of constraints})!}
\]

Now consider this with the following actual problem. A linear program was solved (that is, the maximum or minimum was found) using some method other than the graphical approach. This particular linear program had 6,000,000 variables and 837 constraints. If we substitute these numbers into our formula, we get the following equation:

\[
\text{the number of vertices} = \frac{6000000!(6000000 - 837)!}{837!}
\]

In order to give you a concept of how large these numbers really are consider the following two facts:

1. the largest factorial that a basic, scientific calculator can find is 69!

2. 100! may be larger than the number of atoms in the universe

If that is the case, you simply cannot find all of the vertices and check them against your objective function.

Since we have determined that we cannot graph and test the vertices for any problems larger than the two-dimensional ones, we will need another method to use to work with our linear programs. Previously, we mentioned that in 1947, George Dantzig developed the simplex method for solving linear programming problems. The simplex method is an algorithmic technique for solving linear programming problems. In order to perform the simplex method, the problem must be in standard form. But
standard form for the simplex method means that all of our constraints must be inequalities. So how do we change the equalities to inequalities?

Consider a linear program with inequality constraints. For example, suppose a linear programming problem reduced to

\[
\begin{align*}
\text{maximize:} & \quad 3x_1 + 2x_2 + 4x_3 \\
\text{subject to:} & \quad 30x_1 + 100x_2 + 85x_3 \leq 2500 \\
& \quad 6x_1 + 2x_2 + 3x_3 \geq 90 \\
& \quad x_1, x_2, x_3 \geq 0
\end{align*}
\]

In order to convert the above inequalities into equalities, we will add two extra variables, called slack variables. The problem then becomes

\[
\begin{align*}
\text{Maximize} & \quad 3x_1 + 2x_2 + 4x_3 \\
\text{subject to} & \quad 30x_1 + 100x_2 + 85x_3 + x_4 = 2500 \\
& \quad 6x_1 + 2x_2 + 3x_3 + x_5 = 90 \\
& \quad x_1, x_2, x_3, x_4, x_5 \geq 0
\end{align*}
\]

Notice that these two statements are the same as the earlier constraints. If we manipulate our equalities, we can show this.

\[
\begin{align*}
30x_1 + 100x_2 + 85x_3 & = 2500 - x_4 \quad (\text{our first equality constraint rewritten}) \\
2500 - x_4 & \leq 2500 \quad (\text{because } x_4 \text{ is restricted to be positive}) \\
6x_1 + 2x_2 + 3x_3 & = 90 - x_5 \quad (\text{our second equality constraint rewritten}) \\
90 - x_5 & \leq 90 \quad (\text{because } x_5 \text{ is also positive})
\end{align*}
\]

Therefore, the addition of slack variables does not change the problem. Also, with a little more algebraic manipulation, we can state the relationship of our slack variables to our original variables in a very useful form. We find that

\[
x_4 = 2500 - 30x_1 - 100x_2 - 85x_3 \quad \text{and} \\
x_5 = 90 - 6x_1 - 2x_2 - 3x_3
\]

(5.1)

So, given any problems with a system of constraints involving inequalities, by adding additional nonnegative variables, an equivalent problem can be formulated with a
constraint system consisting of equalities and nonnegativity constraints on the variables. The number of slack variables that we add is determined by the number of constraints that were in the problem originally. But what do these slack variables do for us?

The strategy of the simplex method is to make successive improvements on our current value of the objective function. We will proceed from vertex to vertex of the region bounded by our constraints. We wish to try and proceed from values for \( x_1, \ldots, x_n \) to \( \bar{x}_1, \ldots, \bar{x}_n \) which cause the value of the objective function to be increased.

**Improvement Number One**

In order to begin, we need a feasible value for the objective function. A feasible value is one which does not violate any of our constraints. This is found by setting the decision variables, or the original variables, to 0. This then gives us the following values for each of the variables.

\[
x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 2500, x_5 = 90
\]

Our initial values then yield the feasible value for the objective function \( z \) of \( z = 0 \). Once we have an initial value, then we wish to look for a set of feasible values for our variables that give a smaller value for \( z \). This is not difficult to find. Let's start by seeing what an increase in \( x_1 \) will do to our objective function.

If we allow \( x_2 \) and \( x_3 \) to remain 0, we find that the objective function relies on \( x_1 \) - or \( z = 3x_1 \). If we increase the value of \( x_1 \) to 1, we find that \( z \) increases to 3, (while \( x_4 = 2470 \) and \( x_5 = 84 \), so they are still greater than 0 or feasible). If we set \( x_1 = 2 \), we get that \( z = 6 \) and \( x_4 = 2440, x_5 = 78 \). However, if we set \( x_1 = 16 \), while this still gives us a higher value for our objective function (\( z = 48 \)), we will find that \( x_5 = -6 \) is no longer feasible. So the question becomes, *exactly how much can we increase \( x_1 \) (while keeping \( x_2 = x_3 = 0 \) at the same time) and still keep the values of any other variables \( \geq 0 \)?* Going back to our equations for the slack variables should give us our answer. In the equation \( x_4 = 2500 - 30x_1 - 100x_2 - 85x_3 \), if we substitute in our values for \( x_2 = x_3 = 0 \), we get the equation \( x_4 = 2500 - 30x_1 \). This tells us that in order for \( x_4 \) to be \( \geq 0 \), \( 2500 - 30x_1 \geq 0 \), so \( x_1 \leq 250/3 \). In the same way, from the equation \( x_5 = 90 - 6x_1 - 2x_2 - 3x_3 \), we find that for \( x_5 \) to remain feasible, \( x_1 \) must remain \( \leq 15 \). These two equations allow us to put a bound on \( x_1 \), and since the second is the most stringent, we will use it. Increasing \( x_1 \) to that bound, we obtain the next set of values for our variables:

\[
x_1 = 15, x_2 = 0, x_3 = 0, x_4 = 2050, x_5 = 0
\]
This gives us the feasible value of $z = 45$, which is better than our original value of $z = 0$.

**Improvement Number Two**

Now that we have found a better value for the objective function, we wish to look for a better solution yet. The first time that we looked for a better value, we found that we had the system of equations (5.1) to help us decide how large we could make $x_1$. In order to find our bound this time, we will manufacture a new system of equations to help us out. You will notice that our system of equations (5.1) expressed the variables that did not have values of zero ($x_4, x_5$) in terms of variables that did have values of zero ($x_1, x_2, x_3$). Therefore, this time we wish our system of equations to express $x_1$ and $x_4$ in terms of $x_2, x_3$ and $x_5$ (because they have values of zero from our last manipulation.) We also need to express $z$ in terms of these three variables. So we will be moving $x_1$ from the right-hand side of our system to the left, and $x_5$ will move over to the right.

In order to begin this process, we must first solve the second equation for $x_1$. This will give us an equation for $x_1$ in terms of $x_2, x_3$ and $x_5$ just as we want it. Then, if we substitute our equation for $x_1$ into the equation for $x_4$ as well as the equation for $z$ and simplify, we have our new system of equations as follows:

\[
\begin{align*}
    x_1 &= 15 - 1/3x_2 - 1/2x_3 - 1/6x_5 \\
    x_4 &= 2050 - 90x_2 - 70x_3 + 5x_5 \\
    z &= 45 + x_2 + 5/2x_3 - 1/2x_5
\end{align*}
\] (5.2)

We also currently have the following values for the variables:

\[x_1 = 15, x_2 = x_3 = x_5 = 0, x_4 = 2050\]

**Improvement Number Three**

Now we are ready to perform our next improvement. We need to make a decision as to which variable to improve. If you take a look at the objective function $z$, you may notice that if $x_5$ were to be increased, the value of the objective function will not increase, but rather, because of the negative coefficient, the function will decrease. So it would seem that $x_5$ would not be the variable to increase. Instead, we will choose to increase $x_2$. From (5.2) we find that $x_2 \le 45$ or $x_2 \le 205/9$. Using the stricter bound, we find that our objective function increases from 45 to 610/9 or $\approx 67.78$. 
This tells us that we should solve our second equation for $x_2$ in terms of $x_3$, $x_4$ and $x_5$. Then, as before, we will substitute this equation into the equations for $x_1$ and $z$ in order that they are also expressed in terms of $x_3$, $x_4$ and $x_5$. In doing so, we get the following system of equations:

\[
\begin{align*}
    x_2 &= 205/9 - 7/9x_3 - 1/90x_4 + 1/16x_5 \\
    x_1 &= 200/27 - 13/54x_3 + 1/270x_4 - 9/48x_5 \\
    z &= 200/9 + 31/18x_3 - 1/90x_4 - 7/16x_5
\end{align*}
\] (5.3)

The current values for our variables are then:

\[x_1 = 200/27, x_2 = 205/9, x_3 = x_4 = x_5 = 0\]

**Improvement Number Four**

Now we are ready to make the next improvement to our objective function, which for this problem turns out to be the final improvement. If we take a look at the current equation for our objective function, we find that increasing $x_4$ or $x_5$ will only decrease the value of the function, so this leaves $x_3$ as our only choice to increase. We find that $x_3 \leq 205/7$ is the strictest bound of the two. When we use this information and solve and substitute as before, we find the following set of equations.

\[
\begin{align*}
    x_3 &= 205/7 - 9/7x_2 - 1/70x_4 - 9/112x_5 \\
    x_1 &= 135/378 + 117/378x_2 + 27/3780x_4 - 1017/6048x_5 \\
    z &= 9155/126 - 279/126x_2 - 31/1260x_4 - 279/2016x_5
\end{align*}
\] (5.4)

If you look closely at the coefficients on the variables left in the objective function, you will find that they are all negative. This indicates that any variable that we choose to increase will cause our objective function to decrease. Therefore, we are finished with this problem - we have found the maximum value for the objective function. The solution to this problem is as follows:

\[x_1 = 135/378, x_2 = 0, x_3 = 205/7, x_4 = 0, x_5 = 0, z = 9155/126\]

### 5.5 Other Methods for Solving Linear Programs

So far, we have looked at two methods for solving linear programs. But as was pointed out, the first method was simply not practical for solving linear programs in any more
that two dimensions. The Simplex method provides a logical structured method of moving along the constraints toward the vertices and the solution of larger linear programs. However, since the amount of time used to solve a problem is important, computational scientists studied the number of iterations that the simplex method would take to completely solve the problem. It was determined, after a series of simulations, that in general, the average number of iterations is about $2m$ (where $m$ is the number of constraints in the problem). Unfortunately, this is not the case for every problem. Mathematicians V. Klee and G.J. Minty (1972) have shown that in the worst case, a certain linear program can go through $2^{n-1}$ iterations. At the rate of 100 iterations per second, a problem with 50 variables would take more than 300,000 years to solve. While these problems are not the norm, that still would suggest the need for improvement.

Currently, computational scientists in the area of linear programming are also working with branch and bound methods. The branch and bound method attempts to move through the region bounded by the constraints toward the vertex solution rather than moving along the vertices. At each iteration, the method "branches" off the value from the previous iteration in an attempt to tighten the current "bound" and almost focus in on the solution to the problem. Computational scientists are also working in the area of integer programming. Integer programming is a close relative to linear programming in that the basic structure is the same, yet integer programs have the added restriction that some or all of the variables must be integers. Consider the problem that we worked above. If we restricted any or all of our variables to the set of integers, our value of our objective function would decrease slightly.

Linear Programming is a very current, applicable area of research. As was stated earlier, linear programming has many applications in the area of business and industry. The crew scheduling problems are problems that are constantly under development. The traveling salesperson problems, while not generally used to find a salesperson's route, are extremely important in testing new linear programming methods. By no means is the study of linear programs a complete field.
Chapter 5 Problems

For the following problems, formulate the mathematical models and then solve the problems using the graphical method as above.

1a.) A poultry producer must feed his stock daily at least 124 units of nutritional element A and 60 units of nutritional element B. He has available two feeds. One pound of Feed 1 costs 16 cents and contains 10 units of A and 3 units of B. One pound of Feed 2 costs 14 cents and contains 4 units of A and 5 units of B. Determine the least expensive adequate feeding diet.

b.) As above, determine the least expensive adequate feeding diet, but suppose that the cost of Feed 2 has doubled to 28 cents/lb.

2.) Premium loam is 60% soil, 40% domestic manure, and costs $5 per 50 lb. Generic loam is 20% soil, 10% domestic manure (and 70% sand, stone, etc.) and costs $1 per 50 lb. We need loam for our backyard that is at least 36% soil and at least 20% domestic manure. What combination of the two loams should we use to minimize costs?

Practice your modeling skills. Formulate the mathematical model for the following linear programs. Put the final answer in standard form. Do not attempt to solve the problems.

3.) [This] A paint manufacturer must produce a base for its line of indoor domestic paints. Four chemicals, A, B, C, D are critical in its manufacture. The final composition of the base by weight must be at least 5% of Chemical A, 3% of Chemical B, 26% of Chemical C, and no more than 15% of Chemical D. The manufacturer can produce this base by combining three crude minerals. The compositions by weight and the costs of these minerals are given in the following table.

<table>
<thead>
<tr>
<th>Mineral</th>
<th>% Chem A</th>
<th>% Chem B</th>
<th>% Chem C</th>
<th>% Chem D</th>
<th>Cost (dollars/lb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mineral 1</td>
<td>0%</td>
<td>5%</td>
<td>30%</td>
<td>20%</td>
<td>$4.00</td>
</tr>
<tr>
<td>Mineral 2</td>
<td>6%</td>
<td>8%</td>
<td>30%</td>
<td>10%</td>
<td>$7.50</td>
</tr>
<tr>
<td>Mineral 3</td>
<td>7%</td>
<td>0%</td>
<td>25%</td>
<td>16%</td>
<td>$3.00</td>
</tr>
</tbody>
</table>

The manufacturer could used just Mineral 2. However, he wants to know if some combination of the three minerals will provide a base with the desired characteristics at a lower cost.
4.) [Chvatal 1.7] An oil refinery produces four types of raw gasoline: alkylate, catalytic-cracked, straight-run, and isopentane. Two important characteristics of each gasoline are its performance number PN (indicating antiknock properties) and vapor pressure RVP (indicating volatility). These two characteristics, together with production levels in barrels per day, are as follows:

<table>
<thead>
<tr>
<th>Gasoline</th>
<th>PN</th>
<th>RVP</th>
<th>Barrels Produced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alkylate</td>
<td>107</td>
<td>5</td>
<td>3814</td>
</tr>
<tr>
<td>Catalytic-cracked</td>
<td>93</td>
<td>8</td>
<td>2666</td>
</tr>
<tr>
<td>Straight-run</td>
<td>87</td>
<td>4</td>
<td>4016</td>
</tr>
<tr>
<td>Isopentane</td>
<td>108</td>
<td>21</td>
<td>1300</td>
</tr>
</tbody>
</table>

These gasolines can be sold either raw, at $4.83 per barrel, or blended into aviation gasolines (Avgas A and/or Avgas B). Quality standards impose certain requirements on the aviation gasolines; these requirements, together with the selling prices, are as follows:

<table>
<thead>
<tr>
<th>Type</th>
<th>PN</th>
<th>RVP</th>
<th>Price per barrel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avgas A</td>
<td>at least 100</td>
<td>at most 7</td>
<td>$6.45</td>
</tr>
<tr>
<td>Avgas B</td>
<td>at least 91</td>
<td>at most 7</td>
<td>$5.91</td>
</tr>
</tbody>
</table>

The PN and RVP of each mixture are simply weighted averages of the PNs and RVPs of its constituents. The refinery aims for the plan that yields the largest possible profit. Formulate as an LP problem in the standard form.

5.) [Thie, 2.6.1] A soap manufacturer uses 1200 gal of Mineral Oil A and 2000 gal of Mineral Oil B weekly. These oils can be obtained from 3 products. The yield and cost of a drum of each is as follows:

<table>
<thead>
<tr>
<th>Product</th>
<th>A (gal)</th>
<th>B (gal)</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>15</td>
<td>$13</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>16</td>
<td>$7</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>12</td>
<td>$8</td>
</tr>
</tbody>
</table>

Supplies of these products are unlimited. However, Products 2 and 3 require special processing to separate out the desired mineral oils, with each drum of Product 2
requiring 1 hour of processing and each drum of Product 3 requiring 2 hour. Sixty hours of processing time is available weekly at $4 per hour. Determine what combination of these products should be used to meet the weekly demands and minimize purchase plus processing time costs.
Chapter 6

Some Topics in Numerical Analysis

6.1 Numerical Errors

So far we have looked at different algorithmic improvements that computational scientists are working with to increase the speed at which problems are worked and the size of the problems that the computer can handle. Another area that concerns computational scientists is how the computer represents the data that they need to use.

It is possible for a number to be too large or too small for a computer to store or manipulate. You have seen this yourself with your calculators. Play with your calculators for a moment. What is the smallest number that you can enter into your calculator? the largest? On the small scientific calculators, you will find that you can only enter as many as 10 digits, with or without a decimal place. In order to work with numbers larger than ten billion or smaller than one ten billionth, you must convert your number into the correct power of ten, or scientific notation. Even then, the calculator will not allow you to enter any power larger than 99 or smaller than -99. But we know that there are numbers much larger than these, which means that there are many numbers that we simply cannot represent on our calculator. Obviously, large computers can represent more numbers than a scientific calculator, but they too have their limits. The size of numbers that are too large or too small is changing as the memory space of computers increases, but it still is an important issue. If some of the data that the computational scientist is working with is too large or too small for computer representation, the computer may perform a type of rounding on the number in order to be able to work with it. But then that piece of data is no longer as precise as the original measurement, and errors may occur when the calculations are performed because they will not be as precise as when all of the numbers can be represented. These errors are important because they can change the entire nature of the problem. If the scientist is aware of the data that cannot be represented, he may be able to adjust the calculations to use a close approximation without affecting
the problem or the general conclusion very drastically. Let us look and see how such errors might take place

6.1.1 Number representation

Normally, with pencil and paper, we represent numbers with three parts - an integer part, a decimal point, and a fractional part. Another way that you may be familiar with is called normalized scientific notation. This form is obtained by shifting the decimal point to the left and multiplying it by the appropriate power of 10. Thus, numbers change to normalized scientific notation as follows:

\[
\begin{align*}
4356.792 &= 4.356792 \times 10^3 \\
532.23 &= 5.3223 \times 10^2 \\
0.0045 &= 4.5 \times 10^{-3}
\end{align*}
\]

In normalized scientific notation, the number is expressed as a rational part (i.e. \(4.356792\)) multiplied by 10 to some power (i.e. \(10^3\)). Notice that the power of 10 can be a negative power, as in the last example. The leading digit, or the digit immediately preceding the decimal cannot be zero. If that happens, you have moved the decimal too far, or, as in the last example, not far enough.

Computers use a normalized floating point representation. This is similar to normalized scientific notation. The difference in the two terms identifies whether a number is a computer number or otherwise. A normalized floating point number consists of three parts: a sign that is either positive or negative, a fractional part that lies within the interval \(0.1 \leq r < 1\), (rather than \(1 \leq r \leq 10\) as in normalized scientific notation) and an integral power of 10. (Because of the integral power of ten, the decimal point is moved so that all numbers expressed in normalized scientific or normalized floating point notation have a digit other than zero in the first place after the decimal. The exponent then takes into account the movement of the decimal place.) The fractional part is called the normalized mantissa, and the integral power of ten is called the exponent. In general, however, computers do not use the decimal system, or base ten, to represent numbers. Instead, another base, such as base 2, base 8, base 16, or possibly others, is used. \(^1\) In the binary system (or base 2), the

\(^1\)To review or discover how numbers are translated into other bases, see appendix A. The following material will be assuming some knowledge of the binary system or base 2.
floating point representation is similar to that of the decimal system. The difference is that the fractional part lies within the interval \(1/2 \leq r < 1\), and the exponent is an integral power of 2, rather than 10.

You have seen that normalized floating point notation and normalized scientific notation are very similar. However, in a computer, there is a limited amount of storage space. Every computer only has a finite word length - or a given number of bits - used to store each numerical value and a finite total capacity. Irrational numbers, such as \(\sqrt{2}\), \(\pi\), and \(e\) cannot be represented exactly - as you recall, one of the characteristics of an irrational number is that it is a non-terminating decimal, of which these are examples. In single precision mode, numbers are allowed only one word - or block - of storage. Even numbers that we can represent exactly that do not fit within the space of that one word of storage cannot be represented as precisely as may be necessary. Those numbers that are representable within the structure of a computer are called machine numbers.

The list of numbers that can be represented on a given computer is not a continuous list, but rather it is an oddly discrete set. For example, consider the system where the numbers must be in the form \(x = \pm(0.b_1b_2)_2 \times 2^m\) The numbers \(b_1, b_2,\) and \(m\) are allowed only to have the values of 1 or 0. Using all of the different combinations yields only 11 different numbers for this system.

\[
\begin{align*}
0.00 \times 2^0 &= 0 & 0.00 \times 2^1 &= 0 & -0.00 \times 2^0 &= 0 & -0.00 \times 2^1 &= 0 \\
0.01 \times 2^0 &= 1/4 & 0.01 \times 2^1 &= 1/2 & -0.01 \times 2^0 &= -1/4 & -0.01 \times 2^1 &= -1/2 \\
0.10 \times 2^0 &= 1/2 & 0.10 \times 2^1 &= 1 & -0.10 \times 2^0 &= -1/2 & -0.10 \times 2^1 &= -1 \\
0.11 \times 2^0 &= 3/4 & 0.11 \times 2^1 &= 3/2 & -0.11 \times 2^0 &= -3/4 & -0.11 \times 2^1 &= -3/2 
\end{align*}
\]

Obviously, most computer systems will allow larger numbers than the above, but this should give you a good idea of some of the limitation of machine numbers, and why that limitation exists.

In the representation of numbers in the computer, we have said that numbers get one word of storage. These words are made up of bits. In storing a number in the computer, one bit is taken by the representation of the sign of the number. A zero bit corresponds to a plus sign and a 1 bit corresponds to a negative sign. Another bit of our word of storage holds the sign of the exponent. In a computer with 64 bit word length, the next 14 bits are allocated to represent the exponent of the number and the final 48 bits are used to represent the normalized mantissa. Depending on the size of the words of a given computer, these numbers may vary, but this would
give you an idea of how the numbers must be stored in a computer. If only 14 bits are allowed to represent the exponent, there still remains a large set of numbers that are unrepresentable. We can say that these numbers exceed machine precision.

Because number representation is limited by the size of the word of storage, the computer cannot always represent numbers as precisely as we would like. If the number is extremely large - perhaps on the order of $10^{16}$ - or a number with a lot of digits after the decimal place, the computer will have to round the number in order to fit it into the space allocated to it. This can be done by replacing the number by its nearest machine number. This creates an error in calculations, called round-off error. The closest machine number can be found either by rounding the number using the proper rounding procedures, or it can be found by simply chopping off, or truncating, the digits that don’t fit into the space allocated. This tends to give a larger error than if the proper rounding rules are followed.

It is possible to figure the error, and to know whether a machine number is a good representation of the actual number. This is done by finding either the relative or the absolute error. Suppose that $a$ and $b$ are two numbers, one which is an approximation of the other. The absolute error is computed by the formula $a - b$. The relative error is computed with the formula

$$\frac{a - b}{a} \quad \text{or} \quad \frac{a - b}{b}$$

The relative error assumes that one value is “correct.” If errors occur just in representing numbers and storing them in memory, then it should follow that performing an operation (addition, multiplication ...) on these numbers that have become slightly inexact should lead to even greater error.

### 6.1.2 Finite Precision Arithmetic

As we stated earlier, round-off error is one of the biggest deterrents to effective computation. Not only does this cause problems in the representation of numbers, but it also creates problems in those simple mathematical operations that the computer can do. For the sake of simplicity in the following examples, we will pretend that our computer can only store 4-digit mantissas with at most 2-digit exponents. We will also continue working in the decimal system rather than the binary system usual to computer arithmetic.

Consider the problem of adding $30 + 0.00004$. In exact arithmetic ("pencil and paper"), the solution to this problem is $30.00004$. When the computer is asked
to perform this problem, it first stores the numbers in normalized floating point notation as $0.3000 \times 10^2$ and $0.4000 \times 10^{-4}$. Then it performs the operation, yielding $0.300004 \times 10^2$ Remember, however, that our computer will only store four digits of the mantissa, so our number is rounded to $0.3000 \times 10^2$. While we are only off by $\frac{4}{1000000}$, and this really may seem like an insignificant error, you will notice that the answer to our problem $30 + 0.00004$ has become 30. Does this make much sense? Suppose that we wanted to repeatedly add 0.00004 to 30 ten thousand times rather than just once. The exact answer to this is 30.4, yet if we did the repeated additions, our answer would still come out to be simply 30 again because of the rounding error. If we chose to multiply 0.00004 times ten thousand before adding it to 30, this would give us a much better solution. Some round-off errors can be minimized by paying attention to the size of the numbers that you are working with and ordering your computations accordingly.

Subtraction problems can also cause round-off difficulties. Suppose you wish to subtract two numbers that are very close to each other. This will yield a small difference and introduce another type of error. Consider the following problem. Suppose that we want to perform $0.9999 - 0.9998$ on our imaginary computer. The accurate solution to this problem is 0.0001. As we stated before, our computer stores a four digit mantissa and since this is no more than four digits from the decimal place, we should not have a problem, right? Unfortunately not. Remember that the first thing that a computer does is store the numbers as computer numbers. So our problem becomes $0.9999 \times 10^0 - 0.9998 \times 10^0$ with the solution being $0.1 \times 10^{-3}$. The computer holds a four digit mantissa. There is no accurate information past the first place of these four digits, so that when we use this number in future calculations, we will have lost some of the precision that we might have had to begin with. Because of this error, we often try to avoid subtracting numbers that are too close to each other.

If we are running into problems with accuracy for single operations, imagine the problems that are caused when working with a system of equations where there are numerous operations. Consider the following small system of equations. In precise arithmetic, the actual solutions are $x_1 = 13.6658$ and $x_2 = -6.2$. Let's see how close we can come performing the operations on our imaginary computer.

$$3.000x_1 + 4.127x_2 = 15.41$$
$$1.000x_1 + 1.374x_2 = 5.147$$

or in matrix form,

$$\begin{bmatrix}
3.000 & 4.127 \\
1.000 & 1.374
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} = \begin{bmatrix}
15.41 \\
5.147
\end{bmatrix}$$

Reducing this to upper triangular form, remembering that we only have space for four digits, we get the following:

\[
\begin{bmatrix}
1.000 & 1.376 & 5.137 \\
1.000 & 1.374 & 5.147
\end{bmatrix}
\quad r_1 \div 3
\quad \begin{bmatrix}
1.000 & 1.376 & 5.137 \\
0.000 & -0.0020 & 0.0100
\end{bmatrix}
\quad r_2 - r_1
\]

If we back solve to find the solution at this stage, we get the following solution:

\[x_1 = 12.02 \text{ and } x_2 = -5.000.\]

Obviously, this is not the correct answer. Recall from chapter 2 that there are methods of computing solutions to systems of equations that can minimize, although not remove, the effect of rounding error.

In this case, dividing the first row by three caused extensive rounding. This then led us to subtracting numbers that were quite close to each other, something that we would like to avoid. Together, this rounding introduced significant error into our system. If this much error is introduced into a system requiring so few steps, consider how much error would be introduced into much larger systems.

You may now be asking why we are covering computer numbers and errors to this extent. You probably thought that you were learning about computational science, and you still are. As it was earlier stated, computational scientists are constantly working to improve the conclusions reached through certain methods. In reality, the numbers that occur in a problem are generally not nice numbers such as 150 or 10,000,000. More than likely, there will be numbers that are extremely large, or, at the other end of the spectrum, very small. Consider what you already know about atoms. Just representing the size of one atom requires the use of very small numbers, and most situations aren’t concerned with just one atom, but with very large numbers of them. These alone are cases where numbers can be too large or too small for the computer that you may be somewhat familiar with. Since errors of this type are prevalent, computational scientists are then concerned with the extent to which these errors affect their solutions, as well as finding algorithms to avoid the worst case errors. These methods, however, are topics for another course.

### 6.2 Parallel Programming

As we have repeatedly stated, the performance of computers is measured in floating point operations, sometimes referred to as flops. Any time that a floating point
number is operated on, (added to, subtracted from, multiplied by, divided into, etc.) the computer counts a flop. The more flops performed in a code, the more time it will take your code to run. Also, time is an important consideration for computational science. It doesn’t make a lot of sense to spend more time working with a problem on the computer with our “new” methods than it would if we were to simply use our adequate method from before. We do know in theory both how to model and how to solve many problems, but not necessarily in a practical and timely fashion. Even on the most powerful computers, it could take days, weeks, months, years, or maybe even centuries to work through some problems. So the question becomes “Is there a more effective, efficient way to work with these large problems? “ In some cases, the answer to this question can be parallel computation. If we can spread the number of flops out on a number of processors so that the work can be done simultaneously, then perhaps the whole process will take less time.

There are two methods of completing a task. The work can be done sequentially, or in parallel or as a combination of the two methods. In sequential operations, a task is broken down into a sequence of smaller tasks, and then each of these tasks is completed, one after the other. An example of sequential tasking, as given by Virginia Torczon in her presentation on parallel computation, is doing your laundry at home. Since you only have one washing machine and dryer, and all of your clothing will not fit into the washer at one time, you will find it helpful to divide your load of laundry into smaller loads. You may want to divide it by colors, or by a particular water setting. Once you have your smaller loads, you then put your first load into the washer. After that load is finished, you proceed to wash the next load and you continue until you have completed your laundry. Depending on the amount of laundry that you have to wash and the time that it takes to wash a load of clothing, completing your laundry could take you most of a day. On the other hand, in parallel operations, a task is broken down into smaller, individual tasks, and then each task is completed by a number of different processors simultaneously. Consider your laundry again. It still will not fit into one machine, so you divide the pile into a number of different loads, just as before. However, this time you have access to a laundromat rather than a single machine. You are now able to wash each load of clothing in a different washer at the same time, thus cutting down on the entire time that it takes you to complete your wash. Now what would have taken you most of the day takes a couple of hours, thus speeding up the entire process. You may do a number of tasks in parallel without even realizing it. In computing, the process is similar.
One simple schematic of a computer is as follows: The commands are sent to

![Computer Schematic](image)

**Figure 6.1** a simple computer schematic

the processor, which then accesses the other parts and then sends the result back as output. The Cray YMP, a supercomputer or high speed computer, can perform 325,000,000 flops (325 megaflops). This is a large number of floating point operations. A typical personal computer has a processor that operates at about 20 mhz. On a parallel computer, instead of working with just the one processor, the machine is set up with a number of processors. A number of different schematics then follow with the different arrangement of processors, memory, and the manner in which the different processors communicate. The end result, however, allows the computer to work with a number of different commands or operations simultaneously. The Intel Delta is parallel machine. A single processor on the Intel Delta is capable of 60,000,000 flops. (60 megaflops) You may notice that this is fewer flops than the Cray YMP. However, the difference lies in the fact that the Delta has 512 processors, each operating at 60,000,000 flops, so the peak speed that the Delta can operate at is 31,000,000,000 flops (31 gigaflops). In perspective, if you did one flop per second on a hand-held calculator, it would take about 1000 years to complete the computations that, in theory, can be performed on the Delta in 1 second.

Consider computing the area under a curve. A useful method can be breaking down the area under the curve into a number of smaller, rectangular areas, as seen in figure 6.2.
Figure 6.2 a function broken down into smaller rectangles.

We then can find the different areas of the rectangles and add them together to find the total area under the curve. Solving this problem sequentially involves computing each area, one after the other, and then adding them all together. Depending on the number of different segments, this can add up to quite a few operations on that one processor. However, a method for solving the problem in parallel might break the large set of rectangles down into several smaller sets (the number of sets depending on the number of processors available) and then send each of the smaller sets to a different processor. After each processor has completed the work on its set of rectangles, the results are added together to find the total area under the curve.

Knowing that parallel computing can speed up the process, the issue becomes data management, or input/output. To that end, we can either redesign the hardware and the software support, or we can rearrange the computations.

In all of this rearranging, is parallel computation always the best answer? A popular example of a large scale, linear programming problem is an airline crew scheduling problem. This is a large class of problems with a variety of sizes, but the basic premise is as follows: Given a number of airline crews (pilots, flight attendants etc), and a set number of flight legs, what is the most profitable way to send those crews so that each trip is covered by a flight crew. One of these problems requires approximately 127,500,000 pieces of information just to define the problem. We
cannot fit all of the information required to define this problem, as well as all of
the information necessary to solve the problem at once into the memory of a Cray
YMP. However, we can store all of this information across the 512 processors of the
Delta. In this case, parallelism seems to be the best solution. Another example that
shows how parallelism can improve the time is the work currently being done in a
problem called the traveling salesperson problem. The jist of the problem is that we
have a salesman traveling to a number of cities. We wish for the salesman to
to travel to all of the cities on a given list, yet traveling the shortest distance over all.
Currently work is being done to increase the number of cities in the list as well as
decrease the amount of time that it takes to come up with a solution. Below is a
listing of three different traveling salesperson problems. Each problem was spread out
over a number of parallel processors. If the code that works on this problem were to
be run sequentially, we would find that the time that it would take to complete the
code would be approximately as listed below.

3038 cities, 1.5 SPARCstation 2 years
4461 cities, 1.9 SPARCstation 10 years
7397 cities, 4.0 SPARCstation 10 years

However, the code used to work with these three problems used an extensive
amount of parallelism - the code was broken down into a sequence of tasks and sent
to a large number of processors. The actual wall-clock time that was spent on each of
the problems was about 1/2 a month per problem. If added together all of the time
that each processor spent on its segment of code, we should find that the total time
used was approximately the times listed above.

Again, for the traveling salesperson problem, parallelism seems to be the a good
solution. However, in splitting a large problem into smaller problems, we must coor-
dinate the computation to ensure that when we are done, we have solved the original
problem. This involves exchanging the information across processors. If we exchange
too much information, or if we exchange information too often, then we may actually
slow down the solution time.

So where does this leave us? Parallelism can be well worth the effort. We can
now solve problems that could not be solved effectively. We can consider problems
that were deemed "impossible", and this effort leads to new and unexperienced dis-
coveries. However, this is not true for all problems. The effective organization of
the computation - the mathematics of the computation - is critical to success. The
following considerations are very important.
• How are the computations to be divided into independent pieces?
• How are the results of the independent calculations to be pieced together?
• How is the data to be mapped into memory?
• How is communication between processors to be minimized?
• Is it worth the effort?

The last consideration may be one of the most important. In some cases, preparing the problem to be computed on a parallel machine can take more time and effort than simply performing the calculations sequentially, or the gain in time taken to complete the task in parallel may be not enough to warrant the work to break up the problem in the first place. It may be that sequential programming is all that can be done. In considering parallel computation, you must first consider if it is the best choice. We then leave you with the following question: Think back to the algorithms discussed in chapter 2. We stated that using Cramer's rule and the inverse matrix to solve systems of linear equations were not the method of choice due to the fact that they required more computer work than Gaussian elimination. However, Gaussian elimination does not adapt well to parallel computing. This being the case, could there be some value in using Cramer's rule and the method of using the inverse matrix to solve a system of linear equations?
Chapter 6 Problems

1.) If 6 bits are used to represent positive and negative integers in base 2, how many different numbers can be represented? (Assume the numbers have the form \( \pm b_1 b_2 b_3 b_4 \times 2^m \) where \( b_1, \ldots, b_4 \), and \( m \) are only allowed to have values of 1 or zero). For this problem, be sure to discount the repetitions.

2.) Repeat problem 1 using base 8. For this problem, ignore the repetitions - simply count all of the different combinations.

3.) Generally, in adding a list of floating point numbers, less round-off error will occur if the numbers are added in order of increasing magnitude. Give some examples to illustrate this principle.

4.) Using \( a = 3.526437 \), compute the relative error for the following list of approximations, \( b \), to \( a \) using the formula from section 6.1.1.

<table>
<thead>
<tr>
<th>( b )</th>
<th>Relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.526430</td>
<td>( 2.0 \times 10^{-6} )</td>
</tr>
<tr>
<td>3.526400</td>
<td></td>
</tr>
<tr>
<td>3.526000</td>
<td></td>
</tr>
<tr>
<td>3.520000</td>
<td></td>
</tr>
<tr>
<td>3.500000</td>
<td></td>
</tr>
<tr>
<td>3.000000</td>
<td></td>
</tr>
</tbody>
</table>

Perform the following computations. Give the answer in three decimal arithmetic, first using chopping and then using rounding. In each case, compute the relative error to determine which number is the more accurate approximation.

5.) \( 0.137 \times 10^1 + 0.269 \times 10^{-1} \)
6.) \( 0.485 \times 10^4 - 0.482 \times 10^4 \).
7.) \( 0.378 \times 10^4 + 0.727 \times 10^4 \).
8.) \( 0.403 \times 10^6 \times 0.197 \times 10^1 \).
9.) \( 0.356 \times 10^{-2} \div 0.156 \times 10^4 \).
Appendix A

Computer representation of numbers

In order to understand how improvements can be made in the calculations on the computer, it is first important to see how a computer sees or stores the numbers. Then we can get an idea of how they are stored and where errors can be made that can affect our calculations. Just like everything else, computers too have their limits. Although these limits are constantly being improved, they still exist. Once we know the limits of our computers and how errors can occur, we can design our algorithms to work around them and thus have improved the algorithm and, hopefully, the solution.

A.1 Representing Numbers in Different Bases.

In doing calculations, most computers do not use the same number representation that you use. The number system that you use is in base 10 - using the digits 0 through 9 and then starting over with combinations of these digits at 10. Most computers, however, work in base 2. In this case, the only digits that are used are the digits 0 and 1. Some computers also work in base 8 and others in base 16. Base 8 uses the digits 0 through 7 before it recombines. Base 16 is different than the other bases. If we were to use as the base or foundation the numbers 1 through sixteen, we we would have difficulty distinguishing the different places in the number - for instance, is 1612 a number with two digits, 16 and 12, or does each number represent a different digit. To avoid this confusion, base 16 has as its base the numbers 1 through 9 just like any of the other bases, and the capital letters A - E to represent the remaining 6 digits in this system. Table A.1 is a brief list of the four bases so that you can see how they compare.

Another name for base two is the binary system. The system of numbers in base 8 can also be called the octal system, while the numbers written in base 16 can also be called the hexadecimal system.
<table>
<thead>
<tr>
<th>base 10</th>
<th>base 2</th>
<th>base 8</th>
<th>base 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>101</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>111</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>1110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>1111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>11110</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>11111</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>111110</td>
<td>12</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>111111</td>
<td>13</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>1111110</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>1111111</td>
<td>15</td>
<td>D</td>
</tr>
<tr>
<td>14</td>
<td>11111110</td>
<td>16</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>11111111</td>
<td>17</td>
<td>F</td>
</tr>
<tr>
<td>16</td>
<td>111111110</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>

Table A.1  A comparison of four different bases

A.1.1 Converting Numbers from One Base to Another

Knowing how a computer represents numbers is important. However, if we were not able to convert back and forth between the different systems, it would be of little use to us.

Converting from any base to base 10 is relatively simple. When we look at a large number written normally in base ten, each place has a special meaning. Moving left from the decimal place, we have the ones place, the tens place, the hundreds place, the thousands place, and so on. Another way of saying this could be that we have the $10^0$ place, the $10^1$ place, the $10^2$ place, the $10^3$ place, and so on. Likewise, we could represent numbers in the following manner:

\[
12,345_{10} = 5 \times 10^0 + 4 \times 10^1 + 3 \times 10^2 + 2 \times 10^3 + 1 \times 10^4 \\
1474_{10} = 1 \times 10^3 + 4 \times 10^2 + 7 \times 10^1 + 4 \times 10^0 \\
249854_{10} = 2 \times 10^5 + 4 \times 10^4 + 9 \times 10^3 + 8 \times 10^2 + 5 \times 10^1 + 4 \times 10^0
\]
You will notice that in the second example, the order of the exponents is the opposite of those in the first example, but you can quickly remember that the commutative property of addition allows for this.

This is a rather long and complicated way of writing any number; however, it is the process for converting any number into base 10. In the number 101101₁₂, the places to the left of the decimal place represent the 2⁰ place, the 2¹ place, the 2² place, and so on, just as in the decimal system. In order to convert a number into base two, we will exploit the above relationship for the binary system.

\[
101101₂ = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
= 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \\
= 45₁₀
\]

\[
111₁₂ = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
= 1 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 \\
= 15₁₀
\]

The above process works for converting any number into base 10.

\[
2476₃₈ = 2 \times 8^4 + 4 \times 8^3 + 7 \times 8^2 + 6 \times 8^1 + 3 \times 8^0 \\
= 2 \times 4096 + 4 \times 512 + 7 \times 64 + 6 \times 8 + 3 \times 1 \\
= 10739₁₀
\]

\[
12ADF₁₆ = 1 \times 16^4 + 2 \times 16^3 + A \times 16^2 + D \times 16^1 + F \times 16^0 \\
= 1 \times 65536 + 2 \times 4096 + A \times 256 + D \times 16 + F \times 1 \\
= 76511₁₀
\]

In the case of numbers less than 1 or numbers that have decimal portions, the rules do not change much. Remember that in the decimal system, the numbers to the right of the decimal point represent the tenths place, the hundredths place, the thousandths place, etc., or the 10⁻¹ place, the 10⁻² place, the 10⁻³ place, . . . . Likewise, in the binary system, the places to the right of the decimal point represent the 2⁻¹ place, the 2⁻² place, and so on. In order to change a number from another base into base 10, the process remains the same.

\[
1010.11₁₂ = 1 \times 2^3 + 1 \times 2¹ + 1 \times 2⁻¹ + 1 \times 2⁻² + 1 \times 2⁻³
\]
\[
= 1 \times 8 + 1 \times 2 + 1 \times 1/2 + 1 \times 1/4 + 1 \times 1/8 \\
= 8 + 2 + 0.5 + 0.25 + 0.125 \\
= 10.875_{10}
\]

\[
0.6573_8 = 6 \times 8^{-1} + 5 \times 8^{-2} + 7 \times 8^{-3} + 3 \times 8^{-4} \\
= 6 \times 0.125 + 5 \times 0.015625 + 7 \times 0.001953125 + 3 \times 0.000244140625 \\
= 0.1675292996875_{10}
\]

\[
0.0B3F_{16} = 0 \times 16^{-1} + 11 \times 16^{-2} + 3 \times 16^{-3} + 15 \times 16^{-4} \\
= 0 + 11 \times 0.00390625 + 3 \times 0.000244140625 + 15 \times 0.0000152587890625 \\
= 0.043993005371094_{10}
\]

Now that we can convert a number into base 10, it would be good to know how the computer converts into another base in the first place. In order to convert a number into base 10 from binary or another base, we could use the same method that we used to convert into base 10. Notice that in the second step of each of these conversions, as the multiplication was carried out, the intermediate answers were written in base 10 or the base that we were converting to. If we were to change a number from base 10 into base 2, we could follow the same steps of expanding out the multiplication, then writing the intermediate answers this time in base 2. This can get to be very complicated very fast. As you saw from our earlier table, the numbers quickly grew very large. So, we suggest a different method. We will undo the multiplication of the number. We could write the first step of the above procedure in nested form, giving the following:

\[
N_b = c_0 + (b(c_1 + b(c_2 + \ldots b(c_m)))) = (c_mc_{m-1}\ldots c_2c_1c_0)
\]

If we were to divide the quantity in parenthesis by b again, we would get

\[
N_b = c_0 + c_1 + (b(c_2 + b(c_3 + \ldots b(c_m)))
\]
Each time, the remainder that drops out is next digit as we move left from the decimal point. More precisely, the method works as follows:

\[
\begin{array}{cccccc}
2 & 3781 & r & 11011010101 & 1 & 2 \\
2 & 1890 & 1 & 11011010101 & 0 & 2 \\
2 & 945 & 0 & 11011010101 & 1 & 2 \\
2 & 472 & 1 & 11011010101 & 0 & 2 \\
2 & 236 & 0 & 11011010101 & 1 & 2 \\
2 & 118 & 1 & 11011010101 & 0 & 2 \\
2 & 59 & 0 & 11011010101 & 1 & 2 \\
2 & 29 & 1 & 11011010101 & 0 & 2 \\
2 & 14 & 1 & 11011010101 & 1 & 2 \\
2 & 7 & 0 & 11011010101 & 0 & 2 \\
2 & 3 & 1 & 11011010101 & 1 & 2 \\
2 & 1 & 1 & 11011010101 & 0 & 2 \\
\end{array}
\]

In order to convert the decimal part of a number from base 10 into another base, instead of dividing by the old base, we multiply and take the integral part of the multiplication. That is, \(0.624_{10} \longrightarrow \_\_\_8\)

\[
\begin{align*}
0.624 & \quad \text{At each step - we are only multiplying the decimal part of a number by 8} - \\
\times 8 & \\
4.992 & \quad \text{at this step we are ignoring the 4 - it is the first digit} - \\
\times 8 & \quad \text{after the decimal of our new number.} \\
7.936 & \quad \text{at this step we are ignoring the 7 - it is the second digit} - \\
\times 8 & \quad \text{after the decimal point - and so on,} \\
7.488 & \quad \text{until we have reached an acceptable level of significance.} - \\
\times 8 & \\
3.904 & \quad \text{Our new decimal number is 0.4773}
\end{align*}
\]

The above examples were the methods for changing from base 10. There is a simpler way to convert between base 2, base 8, and base 16. There is a basic relationship between 2 and 8 (i.e. \(8 = 2^3\)), and a similar relationship between 2 and 16 (i.e. \(16 = 2^4\)). The base numbers in the octal system can be expressed as the combinations
of three digit numbers in base 2.

<table>
<thead>
<tr>
<th>base 8</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>base 2</td>
<td>001</td>
<td>010</td>
<td>011</td>
<td>100</td>
<td>101</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

If we take a number written in base 2 and group the numbers in groups of threes, we can do a quick conversion into base 8.

\[
\begin{array}{cccc}
101 & 101 & 001 & . & 110 & 010 & 1_2 \\
5 & 5 & 1 & . & 6 & 2 & 4_8
\end{array}
\]

Notice that for the integer part we work left from the decimal, while for the decimal part, we work right from the decimal. Notice also that any threesomes that are not complete are filled in with 0's. The method for converting from base 2 into base 16 is the same - only instead of groups of threes, we work with groups of four of the number in base 2.

You have now seen several methods in which numbers are converted from one base to another. Technically, there are only two basic methods. The first procedure is to express the number in its exponential notation. Simplify the sentence by carrying out the multiplication, and add the numbers together to get the number in the new base. The second procedure was our remainder-quotient-split method. Remember that this was the method for converting the integer part of a number. The integer fraction split method was simply the second half of this method. The first procedure is better to use when the base you are converting from is smaller than the base that you are converting to. The second method is preferable otherwise. Whenever possible, 10 ↔ 8 ↔ 2 or 10 ↔ 8 ↔ 2 ↔ 16 should be used, for they are the easiest ways to convert numbers between the four systems.
Appendix B

Proof of the Normal Equations

This proof follows closely the proof of the normal equations as found in An Introduction to Linear Algebra: A Curricular Unit for Pre-Calculus Students by T. A. Carter. For this proof, we will assume that $A^T A$ is invertible. This is a reasonable assumption since we have more equations than unknowns when we are attempting to find the best $l_2$ fit line. The length of the vector $x$ is defined to be $||x|| = \sqrt{x \cdot x} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$, so the length of the vector squared is $||x||^2 = \langle x, x \rangle = x^T x = x_1^2 + x_2^2 + \cdots + x_n^2$. Since each element of $Ax - b$ is the error $\delta$ at some point, the square of the length of the vector $Ax - b$ is the sum of the squares of the errors. We want to minimize this quantity, and we claim that the solution to the normal equations, which we will denote as $x^* = (A^T A)^{-1} A^T b$, minimizes the sum of the squares of the errors. That is what we claim, and will prove that

$$||Ax - b||^2 = ||Ax^* - b||^2 + ||A(x - x^*)||^2.$$  \hspace{1cm} (B.1)

Since we are minimizing over $x$, $||Ax - b||^2$ will be minimized when $x = x^*$ so that $||A(x - x^*)||^2$ is zero. This is true because $||A(x - x^*)||^2$ can never be negative.

To prove this claim, we will work with the right side of the equation and prove that it is equal to the left side.

$$= ||Ax^* - b||^2 + ||A(x - x^*)||^2$$

$$= (Ax^* - b)^T (Ax - b) + (A(x - x^*))^T (A(x - x^*))$$

$$= (A(A^T A) - 1A^T b - b)^T (A(A^T A)^{-1} A^T b - b) +$$

$$+ (A(x - (A^T A)^{-1} A^T b))^T A(x - (A^T A)^{-1} A^T b)$$

$$= (A(A^T A) - 1A^T b - b)^T (A(A^T A)^{-1} A^T b - b) +$$

$$+ (Ax - A(A^T A)^{-1} A^T b)^T (Ax - A(A^T A)^{-1} A^T b)$$

$$= (b^T A(A^T A)^{-1} A^T - b^T) (A(A^T A)^{-1} A^T b - b) +$$

$$+ (x^T A - b^T A(A^T A)^{-1} A^T) (Ax - A(A^T A)^{-1} A^T b)$$

$$= b^T A(A^T A)^{-1} A^T A(A^T A)^{-1} A^T b - 2b^T A(A^T A)^{-1} A^T b + b^T b +$$
\[(x^T A^T A x - 2b^T A(A^T A)^{-1} A^T A x + b^T A(A^T A)^{-1} A^T A(A^T A)^{-1} A^T A b) \] (B.6)

\[= (b^T A(A^T A)^{-1} A^T b - 2b^T A(A^T A)^{-1} A^T b + A^T b + b^T b) +\]

\[(x^T A^T A x - 2b^T A x + b^T A(A^T A)^{-1} A^T b) \] (B.7)

\[= b^T b + x^T A^T A x - 2b^T A x \] (B.8)

If we look at the left side of the equation, we see that it equals the right because

\[||Ax - b||^2 = (Ax - b)^T (Ax - b) = (x^T A^T - b^T) (Ax - b) = x^T A^T A x - 2b^T A x + b^T b.\]

This proves that \(x = x^*\) uniquely minimizes the sum of the squares of the error for the equation \(Ax - b\). The following are comments on the steps of the proof:

(B.1) This is the right hand side of the equation from the claim.

(B.2) \(||y||^2 = y^T y\)

(B.3) Replace \(x^*\) with \((A^T A)^{-1} A^T b\).

(B.4) Distribute \(A\) in the second term.

(B.5) \((M + N)^T = M^T + N^T, (RS)^T = S^T R^T, and (A^{-1})^T = (A^T)^{-1}\).

(B.6) Because each term is a vector and \(x^T y = y^T x\), which can also be written as \((x, y) = (x, y)\) we can multiply in a manner similar to polynomial multiplication. However, we must remember that matrix multiplication is not commutative, so the order of multiplication of the matrices matters.

(B.7) \((A^T A)^{-1}(A^T A) = I\)

(B.8) Combine like terms.
Solutions to Chapter Problems

Chapter 2 Problems

1.) For this problem, to find the values for \( x_1 \), \( x_2 \) and \( x_3 \), we must simplify the following determinants:

\[
\begin{align*}
  x_1 &= \begin{vmatrix}
    -4 & 2 & -3 \\
    0 & 1 & -1 \\
    -7 & -1 & -1
  \end{vmatrix} \\
  x_2 &= \begin{vmatrix}
    3 & -4 & -3 \\
    1 & 0 & -1 \\
    2 & -7 & -1
  \end{vmatrix} \\
  x_3 &= \begin{vmatrix}
    3 & 2 & -4 \\
    1 & 1 & 0 \\
    2 & -1 & -7
  \end{vmatrix}
\end{align*}
\]

Since the determinant is defined as all of the possible products of an element from each row and column, we find that for a \( 3 \times 3 \) matrix there are 6 possible products, 
\( a_{11}a_{22}a_{33} - a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{31}a_{22}a_{13} + a_{32}a_{23}a_{11} + a_{33}a_{21}a_{12} \). Using this formula, we find that

\[
  x_1 = \frac{1}{1} = 1; \quad x_2 = \frac{4}{1} = 4; \quad \text{and} \quad x_3 = \frac{5}{1} = 5
\]

2.) For this problem, again we need to find the values of the following determinants:

\[
\begin{align*}
  x_1 &= \begin{vmatrix}
    6 & 4 & -2 & 1 \\
    4 & -2 & 3 & 1 \\
    0 & 1 & -1 & 1 \\
    6 & -2 & 3 & 3
  \end{vmatrix} \\
  x_2 &= \begin{vmatrix}
    3 & 6 & -2 & 1 \\
    2 & 4 & 3 & 1 \\
    -1 & 0 & -1 & 1 \\
    2 & 6 & 3 & 3
  \end{vmatrix}
\end{align*}
\]
\[
x_3 = \begin{vmatrix}
3 & 4 & 6 & 1 \\
2 & -2 & 4 & 1 \\
-1 & 1 & 0 & 1 \\
2 & -2 & 6 & 3 \\
\end{vmatrix}
\quad x_4 = \begin{vmatrix}
3 & 4 & -2 & 6 \\
2 & -2 & 3 & 4 \\
-1 & 1 & -1 & 0 \\
2 & -2 & 3 & 6 \\
\end{vmatrix}
\]

From this point, the determinants can be found using the sum of the products, using expansion by minors, (a method that will be explained in most linear algebra texts), or as in our case, by using the determinant function on the graphing calculator. We found the solution to be:

\[
x_1 = \frac{-14}{-14} = 1; \quad x_2 = \frac{-14}{-14} = 1; \quad x_3 = \frac{-14}{-14} = 1; \quad \text{and} \quad x_4 = \frac{-14}{-14} = 1.
\]

3a.) Using Gaussian elimination, this matrix reduces to

\[
\begin{bmatrix}
1 & 6 & 3 \\
0 & -11 & -5 \\
\end{bmatrix}
\quad \text{or} \quad
\begin{bmatrix}
1 & 0 & 6 \\
0 & 1 & .455 \\
\end{bmatrix}
\]

Back solving, we find that \(x_1 = .273\) and \(x_2 = .455\).

3b.) Using Gaussian elimination, this matrix reduces to

\[
\begin{bmatrix}
1 & 2.56 & 11.9256 \\
0 & 1.0721 & -2.1233 \\
\end{bmatrix}
\quad \text{or} \quad
\begin{bmatrix}
1.0 & 2.56 & 11.9256 \\
0 & 1 & -2.0645 \\
\end{bmatrix}
\]

Using the reduced matrix to back solve, we find that \(x_1 = 17.207\) and \(x_2 = -2.0645\).

3c.) For this problem, we find that the matrix reduces to

\[
\begin{bmatrix}
1.0000 & -0.4810 & 1.0000 \\
0 & 4.0675 & -0.1414 \\
\end{bmatrix}
\quad \text{or} \quad
\begin{bmatrix}
1.0000 & -0.4810 & 1.0000 \\
0 & 1.0000 & -0.1414 \\
\end{bmatrix}
\]

Using the reduced matrix to back solve, we find that \(x_1 = .9320\) and \(x_2 = -.1414\).

4.) For this problem, retaining four significant figures gave the following reduced matrix.

\[
\begin{bmatrix}
-.2081 & -.4262 & -1.4826 \\
0 & 1 & 366.6 \\
\end{bmatrix}
\]
The solution to this system of equations was found to be \[
\begin{bmatrix}
743.7 \\
366.6
\end{bmatrix}
\]
In checking the problem, we multiplied the coefficient matrix, A, by the solution matrix to get the following:
\[
\begin{bmatrix}
.1036 & .2122 \\
.2018 & .4247
\end{bmatrix}
\times
\begin{bmatrix}
743.7 \\
366.6
\end{bmatrix}
= 
\begin{bmatrix}
.7242 \\
.9292
\end{bmatrix}
\]
As you can see, the rounding to four significant digits does cause some error in this system of equations, as we will discuss in more detail in chapter 6.

5.) For this problem, the original matrix
\[
\begin{bmatrix}
1 & 2 & 3 & 13 \\
2 & 4 & 2 & 14 \\
3 & 5 & -1 & 11
\end{bmatrix}
\]
reduces to
\[
\begin{bmatrix}
1 & 2 & 3 & 13 \\
0 & 1 & 10 & 28 \\
0 & 0 & 1 & 3
\end{bmatrix}
\]
Using this reduced matrix to back solve, we find that \(x = 8, y = -2\) and \(z = 3\).

6.) Using Gaussian elimination, we find that this matrix reduces to
\[
\begin{bmatrix}
1.0000 & 0.5000 & -1.5000 & -0.5000 \\
0 & 1.0000 & 1.4000 & 5.0000 \\
0 & 0 & 1.0000 & 1.8182
\end{bmatrix}
\]
Using this reduced matrix to back solve, we find that \(x_1 = 1, x_2 = \frac{27}{11}or2.4545\) and \(x_3 = \frac{40}{11}or1.8182\).

7.) Using Gaussian elimination, we find that the original matrix reduces to
\[
\begin{bmatrix}
1 & -1 & -3 & -75 \\
0 & 1 & -2 & -43 \\
0 & 0 & 1 & 25
\end{bmatrix}
\]
Using this matrix to back solve, we find that the solution to this system of equations is \(x_1 = 3, x_2 = 1\) and \(x_3 = 0\).

8.) For this problem, the matrix reduces to
\[
\begin{bmatrix}
1.0000 & -2.0000 & 3.0000 & -4.0000 & 0 \\
0 & 1.0000 & -1.5000 & 1.2500 & 1.2500 \\
0 & 0 & 1.0000 & 3.5000 & 5.5000 \\
0 & 0 & 0 & 1.0000 & 1.0000
\end{bmatrix}
\]
Using this reduced matrix to back solve, we find that \(x = 4, y = 3, z = 2\) and \(u = 1\).
9.) Using Gaussian elimination, we find that the original augmented matrix reduces to

\[
\begin{bmatrix}
1 & 2\frac{1}{3} & -2 & \frac{2}{3} & 1\frac{1}{3} \\
0 & 1 & 2\frac{1}{2} & 2 & -1\frac{1}{2} \\
0 & 0 & 1 & 2 & 1 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

Using this reduced matrix to back solve, we find that \(x_1 = 1, \ x_2 = -1, \ x_3 = -1\) and \(x_4 = 1\).

10.) Using Gaussian elimination, this matrix reduces to

\[
\begin{bmatrix}
1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 & 11.2345 \\
0 & 1.0000 & -0.6667 & 1.6667 & -1.3333 & 2.3333 & 0.9125 \\
0 & 0 & 1.0000 & 2.0000 & 1.0000 & 0.5000 & 0.2643 \\
0 & 0 & 0 & 1.0000 & 0.0648 & 0.6852 & 0.0306 \\
0 & 0 & 0 & 0 & 1.0000 & -4.7385 & 0.0016 \\
0 & 0 & 0 & 0 & 0 & 1.0000 & 0.0005
\end{bmatrix}
\]

Using this matrix to back solve, we find that \(x_1 = 10, \ x_2 = 1, \ x_3 = .2, \ x_4 = .03, \ x_5 = 0.004\) and \(x_6 = 0.0005\).

11.) The answer to this should be the same as problem 6.

12a.) By Gauss-Jordan elimination, we reduce this matrix to

\[
\begin{bmatrix}
1 & 0 & 0 & \frac{112}{47} \\
0 & 1 & 0 & \frac{70}{47} \\
0 & 0 & 1 & \frac{95}{47}
\end{bmatrix}
\]

From this we know that \(x = 2.3830, \ y = 1.4894\) and \(z = 2.0213\).

12b.) By Gauss-Jordan elimination, we reduce this matrix to

\[
\begin{bmatrix}
1 & 0 & 0 & 11 \\
0 & 1 & 0 & 11 \\
0 & 0 & 1 & 10
\end{bmatrix}
\]

From this, we know that \(x = 11, \ y = 11\) and \(z = 10\).

13.) Reducing the large, \(3 \times 6\) matrix using Gauss-Jordan elimination, we get the following:

\[
\begin{bmatrix}
1 & 0 & 0 & \frac{1}{4} & 2 & \frac{1}{4} \\
0 & 1 & 0 & \frac{-13}{4} & 27 & \frac{16}{4} \\
0 & 0 & 1 & \frac{17}{4} & -3 & \frac{1}{4}
\end{bmatrix}
\]
Therefore, the three solutions are as follows:

a.) $x = \frac{1}{4}$, $y = -\frac{3}{16}$ and $z = \frac{17}{16}$;
b.) $x = 2, y = -\frac{13}{14}$, and $z = -\frac{3}{4}$;
c.) $x = \frac{1}{4}$, $y = \frac{27}{16}$ and $z = \frac{1}{16}$;

14.) The inverse of

$$A = \begin{bmatrix} 7 & 1 \\ 4 & 5 \end{bmatrix} \quad \text{is} \quad A^{-1} = \begin{bmatrix} \frac{5}{31} & -\frac{1}{31} \\ -\frac{4}{31} & \frac{7}{31} \end{bmatrix}$$

15.) The inverse of

$$A = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad \text{is} \quad A^{-1} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ -1 & 2 & -3 \\ -1 & 0 & 1 \end{bmatrix}$$

16.) The inverse of

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \quad \text{is} \quad A^{-1} = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

The inverse of

$$B = \begin{bmatrix} 1 & 4 & 5 \\ 2 & 1 & 2 \\ 8 & 1 & 1 \end{bmatrix} \quad \text{is} \quad B^{-1} = \begin{bmatrix} -\frac{1}{25} & \frac{1}{25} & \frac{3}{25} \\ -\frac{14}{25} & -\frac{39}{25} & \frac{8}{25} \\ -\frac{6}{25} & \frac{31}{25} & -\frac{7}{25} \end{bmatrix}$$

Chapter 3 Problems

Please note that the answers that you receive may be slightly different, due to rounding or chopping error.

1a.) Graphing this function on the graphing calculator, we found that the function had zeros between 4.3 and 4.7 and between 7.7 and 7.8 within the given boundaries, $0 < x < 3\pi$. Stopping when $|f(x_n)| \leq 0.0001$ we found zeros at $x = 4.42858887$ and $x = 7.705951$.

1b.) Graphing this function on the graphing calculator, we found that the function had a zero between 1.06 and 1.17 when $x > 0$. Stopping when $|f(x_n)| \leq 0.0001$, we found the zero at $x = 1.076328125$. 
1c.) Graphing this function on the graphing calculator, we found that the function had a zero between -0.63 and -0.74. Stopping when \(|f(x_n)| \leq 0.0001\), we found the zero at \(x = -0.68231445\).

1d.) Graphing this function on the graphing calculator, we found that the function had zeros between -0.65 and -0.606, between 0.08 and 0.13, between 0.51 and 0.56, between 0.88 and 0.93, and between -0.92 and -0.97. Stopping when \(|f(x_n)| \leq 0.0001\), we found the zero at \(x = -0.60862109375\), \(x = 0.102119140625\), \(x = 0.556337890625\), \(x = 0.925947265625\), and \(x = -0.97583008\).

2.) Graphing this function on the graphing calculator, we found that the function had zeros between 0 and 0.4. Stopping when \(|f(x_n)| \leq 0.0001\), we found the zero at \(x = 0.318848\).

3.) Given the interval [1,2], we found the zero for this function to be at \(x = 1.762451171875\).

4.) Graphing this function on the graphing calculator, we found that the function had zeros between 1.5 and 1.65 and between 4.6 and 4.7. Stopping when \(|f(x_n)| \leq 0.0001\), we found the zeros at \(x = 1.5708963\) and \(x = 4.71246\).

5.) Graphing this function on the graphing calculator, we found that the function had zeros between 0.60 and 0.65 and between 1.51 and 1.56. Stopping when \(|f(x_n)| \leq 0.0001\), we found the zeros at \(x = 0.619140625\) and \(x = 1.5121484375\).

6.) Graphing this function on the graphing calculator, we found that the function had zeros near 1 and near 1.5. Stopping when \(|f(x_n)| \leq 0.001\), we found the zeros at \(x = 0.79095\) and \(x = 1.61804\).

7a.) From problem 1, we know that the zero for this function is near 4.4 and 7.7. Stopping when \(|f(x_n)| \leq 0.0001\), we found the zero at \(x = 4.428597\) and 7.7059515.

7b.) From problem 1, we know that the zero for this function is near 1.06. Stopping when \(|f(x_n)| \leq 0.0001\), we found the zero at \(x = 1.07646489\).

7c.) From problem 1, we know that the zero for this function is near -0.67. Stopping when \(|f(x_n)| \leq 0.0001\), we found the zero at \(x = 0.682328\).

7d.) From problem 1, we know that the zeros are near -0.606, 0.1, 0.55, -0.92, and 0.92, so we choose to use these as our \(x_0\). From this information, we find the zeros at \(x = -0.6086036926\), \(x = 0.1021386655\), and \(x = 0.55636973151249\). For the two remaining zeros, we found that rather than getting closer to 0, the function values began increasing. This would lead us to believe that Newton’s method is not the method of choice for this problem.
8.) Graphing this function on the graphing calculator, we found that the function has a zero near 0.8. After four iterations, \(|f(x_n)|\) was reasonably small, so \(x = 0.7390851\).

9.) For this function, since we are asked to find the zero that lies in the interval \(1 < x < 3\), we chose \(x_0 = 1\). Stopping after four iterations, we find that \(x = 1.93456896\).

10a.) Graphing this function on the graphing calculator, we find that it has zeros near 1.91 and 0.63. Using either of these points as starting points, or any values relatively near to these points, we find that the value of the function at each new \(x\) increases rather than converging to zero, leading us to believe that Newton's method is not the method of choice for this problem. See problem 14.

10b.) Graphing this function on the graphing calculator, we found that it had zeros near 0.53 and 4.68. Using these values as our starting point and stopping after four iterations, we found the zeros at \(x = 0.605267\) and \(x = 4.7107938\).

10c.) Graphing this function on the graphing calculator, we found that it had a zero near 0.5. Using this information, we found that \(x = 0.45339833667909\).

10d.) Graphing this function on the graphing calculator, we found that it had zeros near -1.91. Using this as our starting point, we quickly found that the zero is at \(x = 2\).

11.) Given that we wish to find a root in the interval \(0 < x < \pi/2\), we use these as our starting points \(x_{-1}\) and \(x_0\). Stopping when \(|f(x_n)| < 0.0001\), we find the zero for this function at \(0.73908\).

12.) Using the given \(x\) values as our initial values, we find that this function has a zero at \(x = 1.22999\).

13.) From problem 9, we know that we wish to find the zero in the interval \(1 < x < 3\). Using these two values as our interval, \([a, b]\), we find the zero for this function at \(x = 1.9345464\).

14a.) Graphing this function on the graphing calculator, we find that this function has zeros between 0.63 and 0.68, and between 1.87 and 1.91. Stopping after less than 4 iterations, we found the zeros at \(x = 1.906705154\) and \(x = 0.677253\).

14b.) Graphing this function on the graphing calculator, we find that this function has zeros between 4.68 and 4.78 and between 0.53 and 0.63. Using this information and stopping after four iterations, we find the zeros for this function at \(x = 0.605244319\) and \(x = 4.7079379655\).
14c.) Graphing this function on the graphing calculator, we find that this function has a zero between 0.42 and 0.63. Using this information and stopping after 4 iterations, we find that this function has a zero at \( x = 0.45337007 \).

14d.) Graphing this function on a graphing calculator, we find that this function has a zero between 0.29 and 0.34. However, again for this function we find that the value of \( f(x_n) \) increases rather than converging to zero.

16.) Since it tends to converge the fastest, we chose to use Newton's method to find the zero for this function. Graphing this function on the graphing calculator, we found that this function has a zero near 2. Using this information and stopping when \( |f(x_n)| \leq 0.000001 \), we found the zero for this function at \( x = 1.73205 \).

17.) We also chose to use Newton's method for this problem. Graphing this function on a graphing calculator, we found that this function has a zero near 4. Using this information, and stopping when \( |f(x_n)| \leq 0.000001 \), we found the zero for this function at \( x = 4.21716 \).

18.) Graphing this function on the graphing calculator, we found that this function had a zero between 2 and 3. The zeros that we found using the various methods are as follows:

- Bisection method: \( x = 2.6456 \cdot 9 \text{ iterations} \).
- Newton's method: \( x = 2.6458 \cdot 3 \text{ iterations} \).
- Secant Method: \( x = 2.6457 \cdot 3 \text{ iterations} \).
- Method of False position: \( x = 2.6456 \cdot 3 \text{ iterations} \).

19.) Using the starting points of \( x_{-1} = 1 \) and \( x_0 = 3 \) and stopping after 4 iterations, we found the zero at \( x = 1.93456216121447 \).

20.) Substituting the given values into the formula, we find that the function that we wish to find the zeros for is

\[
f(x) = \frac{(1-x)(3+x)^{\frac{1}{2}}}{x(x+1)^{\frac{1}{2}}\sqrt{5}} - 3.06
\]

Graphing this function on the graphing calculator, we find that this function has a zero between 0.1 and 0.21. Using these values as our initial values for the secant method, we find that this function has a zero at \( x = 0.1929618 \) moles.

Chapter 4 Problems

1.) For this problem we wish to find the line, \( y = ax + b \) that best fits the data given. If we substitute the data given into this formula, we get the following matrices for
the normal equations:

\[
\begin{bmatrix}
0 & 1 & 2 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
0 & 1 \\
1 & 1 \\
2 & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b
\end{bmatrix}
=
\begin{bmatrix}
0 & 1 & 2 \\
1 & 1 & 1 \\
-6 & 7
\end{bmatrix}
\]

This simplifies to

\[
\begin{bmatrix}
5 & 3 \\
3 & 3
\end{bmatrix}
\begin{bmatrix}
a \\
b
\end{bmatrix}
=
\begin{bmatrix}
8 \\
6
\end{bmatrix}
\]

From here, we use Gaussian elimination to solve this small system of equations so that \(a = 1\) and \(b = 1\). The equation for the line of best fit in the least squares sense is \(y = x + 1\).

2.) For this problem we wish to find the line, \(y = ax + b\) that best fits the data given. If we substitute the data given into this formula, we get the following matrices for the normal equations:

\[
\begin{bmatrix}
1.0 & 1.5 & 2.0 & 2.5 & 3.0 \\
1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
1.0 & 1 \\
1.5 & 1 \\
2.0 & 1 \\
2.5 & 1 \\
3.0 & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b
\end{bmatrix}
=
\begin{bmatrix}
1.0 & 1.5 & 2.0 & 2.5 & 3.0 \\
1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
2.0 \\
3.2 \\
4.1 \\
4.9 \\
5.9
\end{bmatrix}
\]

which simplifies to

\[
\begin{bmatrix}
22.5 & 10 \\
10 & 5
\end{bmatrix}
\begin{bmatrix}
a \\
b
\end{bmatrix}
=
\begin{bmatrix}
44.95 \\
20.1
\end{bmatrix}
\]

This gives us \(a = 1.9\) and \(b = .22\), so the line of best fit in the least squares sense is \(y = 1.9x + 0.22\).

3.) For this problem we wish to find the line, \(y = ax + b\) that best fits the data given. If we substitute the data given into this formula, we get the following matrices for the normal equations:

\[
\begin{bmatrix}
0.1 & 1 \\
0.2 & 1 \\
0.3 & 1 \\
0.4 & 1 \\
0.5 & 1 \\
0.6 & 1 \\
0.7 & 1 \\
0.8 & 1 \\
0.9 & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b
\end{bmatrix}
\]
which simplifies to \[
\begin{bmatrix}
2.85 & 4.5 & 11.06227 \\
4.5 & 9 & 20.9147
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= \begin{bmatrix}
0 \\
2.1220 \\
3.0244 \\
3.2568 \\
3.1399 \\
2.8579 \\
2.5140 \\
2.1639 \\
1.8358
\end{bmatrix}
\] This gives us \( a = 1.819756 \) and \( b = 1.0082 \), so the equation for our line of best fit in the least squares sense is \( y = 1.819756x + 1.0082 \).

4.) For this problem, we wish to find the parabola, \( y = ax^2 + bx + c \) that best fits the data given. If we substitute the data given into this formula, we get the following matrices in the normal equations:

\[
\begin{bmatrix}
1 & 0 & 1 \\
1 & 0 & 1 \\
-1 & 0 & 1 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 1 \\
0 & 1 & 0 \\
-1 & 0 & 1 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
3.1 \\
0.9 \\
2.9
\end{bmatrix}
\]

This simplifies to \[
\begin{bmatrix}
2 & 0 & 2 \\
0 & 2 & 0 \\
2 & 0 & 3
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= \begin{bmatrix}
6 \\
-0.2 \\
6.9
\end{bmatrix}
\] This gives us \( a = 2.1 \), \( b = -0.1 \), and \( c = 0.9 \), so the equation for our parabola that best fits the data in the least squares sense is \( y = 2.1x^2 - 0.1x + 0.9 \).

6.) For this problem, we wish to find the parabola, \( y = ax^2 + bx + c \) that best fits the data given. If we substitute the data given into this formula, we get the following matrices in the normal equations:

\[
\begin{bmatrix}
4 & 1 & 0 & 1 & 4 \\
4 & 1 & 0 & 1 & 4 \\
-2 & -1 & 0 & 1 & 2 \\
1 & 1 & 1 & 1 & 1 \\
4 & 2 & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= \begin{bmatrix}
2 \\
1 \\
1 \\
1 \\
1
\end{bmatrix}
\]
This simplifies to \[
\begin{bmatrix}
34 & 0 & 10 \\
0 & 10 & 0 \\
10 & 0 & 5
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
18 \\
0 \\
7
\end{bmatrix}.
\]
From this we find that \(a = \frac{2}{3}\), \(b = 0\), and \(c = \frac{29}{35}\) which yields \(y = \frac{2}{3}x^2 + \frac{29}{35}\) as the equation for the parabola of best fit in the least squares sense.

7.) For this problem, we wish to find the parabola, \(y = ax^2 + bx + c\) that best fits the data given. If we substitute the data given into this formula, we get the following matrices in the normal equations:

\[
\begin{bmatrix}
0 & 1 & 4 & 9 \\
1 & 1 & 1 & 1 \\
4 & 2 & 1 & 9
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
18 \\
0 \\
1
\end{bmatrix}
\]

This simplifies to \[
\begin{bmatrix}
98 & 36 & 14 \\
36 & 14 & 6 \\
14 & 6 & 4
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
18 \\
0 \\
3
\end{bmatrix}.
\]
From this we find that \(a = 0.75\), \(b = -1.95\) and \(c = 1.05\), which yields \(y = 0.75x^2 - 1.95x + 1.05\) as the equation for the parabola that best fits our data in the least squares sense.

8.) For this problem, we wish to find the parabola, \(y = ax^2 + bx + c\), that best fits the data given. If we substitute the data given into this formula, we get the following matrices in the normal equations:

\[
\begin{bmatrix}
0 & 1 & 4 & 9 & 16 & 25 & 36 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
4 & 2 & 1 & 9 & 3 & 1 & 6
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
0 & 1 & 4 & 9 & 16 & 25 & 36 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

This simplifies to \[
\begin{bmatrix}
2275 & 441 & 91 \\
441 & 91 & 21 \\
91 & 21 & 7
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
591.7 \\
129.1 \\
32.4
\end{bmatrix}.
\]
From this we find that \(a = -0.252381\), \(b = 2.6536\), and \(c = -0.051191\), so that the equation for our parabola of best fit in the least squares sense is \(y = -0.252381x^2 + 2.6536x - 0.051191\).

9.) For this problem, we wish to find the parabola, \(y = ax^2 + bx + c\) that best fits the data given. If we substitute the data given into this formula, we get the following
matrices in the normal equation:

\[
\begin{bmatrix}
0.01 & 0.04 & 0.09 & 0.16 & 0.25 & 0.36 & 0.49 & 0.64 & 0.81 \\
0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= \begin{bmatrix}
0.01 & 0.04 & 0.09 & 0.16 & 0.25 & 0.36 & 0.49 & 0.64 & 0.81 \\
0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= \begin{bmatrix}
6.795737 \\
11.06227 \\
20.9147
\end{bmatrix}
\]

This simplifies to

\[
\begin{bmatrix}
1.5333 & 2.025 & 2.85 \\
2.025 & 2.85 & 4.5 \\
2.85 & 4.5 & 9
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= \begin{bmatrix}
6.795737 \\
11.06227 \\
20.9147
\end{bmatrix}
\]

From this we find that \( a = 14.0315; \ b = 15.0397; \) and \( c = -0.7527, \) so the equation for our parabola of best fit in the least squares sense is \( y = 14.0315x^2 + 15.0397x - 0.7527. \)

10.) For this problem, we need to find the coefficients \( a_0, a_1, a_2, \) and \( a_3. \) If we substitute the data given into this formula, we get the following matrices in the normal equations:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 \\
0 & 0.309 & 0.588 & 0.809 & 0.951 & 1 \\
0 & 0.588 & 0.951 & 0.951 & 0.588 & 0
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 & 0 \\
0.1 & 0.309 & 0.588 & 0.951 \\
0.2 & 0.588 & 0.951 & 1.029 \\
0.3 & 0.809 & 0.951 & 1.029 \\
0.4 & 0.951 & 0.951 & 0.588 \\
0.5 & 0.588 & 0.951 & 0
\end{bmatrix}
\]
\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 \\
0 & 0.309 & 0.588 & 0.809 & 0.951 & 1 \\
0 & 0.588 & 0.951 & 0.951 & 0.588 & 0 \\
\end{bmatrix} \begin{bmatrix}
.11 \\
.251 \\
.352 \\
.453 \\
.54 \\
.66 \\
\end{bmatrix}
\]

This simplifies to
\[
\begin{bmatrix}
6 & 1.5 & 3.657 & 3.078 \\
1.5 & .55 & 1.2716 & 0.7695 \\
3.657 & 1.2716 & 3.000107 & 2.069427 \\
3.078 & 0.7695 & 2.069427 & 2.50029 \\
\end{bmatrix} \begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
\end{bmatrix} = \begin{bmatrix}
2.366 \\
0.7774 \\
1.824552 \\
1.230663 \\
\end{bmatrix}.
\]

From this, we find that \( a_0 = 0.11249, \ a_1 = 2.67457, \ a_2 = -0.78956 \) and \( a_4 = 0.18409 \), so the equation that best fits this data in the least squares sense is \( y = 0.11249 + 2.67457x - 0.78956\sin \pi x + 0.18409\sin 2\pi x \).

11.) For this problem, we need to find the coefficients \( a \) and \( b \). If we substitute the data given into this formula, we get the following matrices in the normal equations:

\[
\begin{bmatrix}
0 & -1 & 0 & 1 & 0 \\
-1 & 0 & 1 & 0 & -1 \\
\end{bmatrix} \begin{bmatrix}
 a \\
 b \\
\end{bmatrix} = \begin{bmatrix}
0 & -1 & 0 & 1 & 0 \\
-1 & 0 & 1 & 0 & -1 \\
\end{bmatrix} \begin{bmatrix}
-1 \\
0 \\
1 \\
2 \\
1 \\
\end{bmatrix}
\]

This simplifies to \( \begin{bmatrix}
2 & 0 \\
0 & 3 \\
\end{bmatrix} \begin{bmatrix}
a \\
b \\
\end{bmatrix} = \begin{bmatrix}
2 \\
1 \\
\end{bmatrix} \). From this we find that \( a = 1 \) and \( b = \frac{1}{3} \), so the equation that best fits this data in the least squares sense is \( y = \sin \pi x + \frac{1}{3} \cos \pi x \).

12.) For this problem, we need to find the coefficients \( a_1, \ a_2, \ a_3 \) and \( a_4 \). If we substitute the data given into this formula, we get the following matrices in the normal equations:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
0 & 0.4 & 0.5 & 0.7 & 0.7 & 0.9 \\
0.9998 & 0.389 & 0.479 & 0.644 & 0.644 & 0.783 \\
1.105 & 1.492 & 1.649 & 2.014 & 2.014 & 2.460 \\
\end{bmatrix} \begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
a_4 \\
\end{bmatrix} = \begin{bmatrix}
1 & 0.1 & 0.9998 & 1.105 \\
1 & 0.4 & 0.389 & 1.492 \\
1 & 0.5 & 0.479 & 1.649 \\
1 & 0.7 & 0.644 & 2.014 \\
1 & 0.7 & 0.644 & 2.014 \\
1 & 0.9 & 0.783 & 2.460 \\
\end{bmatrix}
\]
This simplifies to
\[
\begin{pmatrix}
6 & 3.3 & 3.0388 & 10.734 \\
3.3 & 2.21 & 2.01138 & 6.5654 \\
3.0388 & 2.0138 & 1.8333 & 6.001 \\
10.734 & 6.5654 & 6.001 & 20.3303
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3
\end{pmatrix}
= \begin{pmatrix} 7.55 \\ 4.851 \\ 4.414 \\ 14.715 \end{pmatrix}
\]
From this, we find that \(a_1 = -5.9973\), \(a_2 = -23.2868\), \(a_3 = 16.2129\), and \(a_4 = 6.6250\), so the equation that best fits this data in the least squares sense is \(-5.9973 + -23.2868 x + 16.2129 \sin x + 6.6250 e^x\).

Chapter 5 Problems

1a.) The following table organizes the information that we have been given. From

<table>
<thead>
<tr>
<th>Nutritional Elements</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Feed 1</td>
<td>10</td>
</tr>
<tr>
<td>Feed 2</td>
<td>4</td>
</tr>
</tbody>
</table>

the problem, we know that we wish to minimize the cost. If we let Feed 1 be \(x\) and Feed 2 be \(y\), the problem is formulated as

\[ \text{Minimize } C = .16x + .14y \]

\[ 124 \leq 10x + 4y \]

\[ 60 \leq 4x + 5y \]

When we graph the two constraints, we find that the vertices of this polygonal region are at \((10\frac{2}{5}, 10\frac{5}{3})\), \((0, 12)\), \((12\frac{2}{5}, 0)\). Substituting each of these ordered pairs into the cost equation, we find the minimum at \((0,12)\), so the farmer should by 12 pounds of Feed 2 in order to meet the nutritional needs and minimize costs.

1b.) The only information that has changed for this problem is that the cost of Feed 2 has increased to $0.28/lb. Therefore the only equation that changes is \(C\), the
cost equation. The vertices also remain the same. Substituting these values into the cost equation finds the minimum of $1.98/\text{lb.}$ at the vertex \((12\frac{2}{5}, 0)\), so the farmer should buy \(12\frac{2}{5}\ \text{lbs.}\) of feed number two in order to meet the nutritional needs and minimize the costs.

2.) Organizing the information in the following table, we find that The problem then becomes

\[
\begin{array}{cccc}
\text{Soil} & \text{Manure} & \text{Sand} & \text{Cost} \\
\hline
\text{Premium Loam} & 60\% & 40\% & $5/50 \ \text{lbs.} \\
\text{Generic Loam} & 20\% & 10\% & 70\% & $1/50 \ \text{lbs.}
\end{array}
\]

\[
\text{Minimize } C = 5x + y \\
.36 \leq .6x + .2y \\
.2 \leq .4x + .1y \\
.44 \geq .7y
\]

For this problem, the vertices are found at \((0, \frac{44}{7}), (0, .6), (.343, .628), (\frac{1}{6}, 0)\) and \((\frac{1}{2}, 0)\). Substituting each of these values into our cost equation, we find that the minimum is obtained at \((0, .6)\). Therefore the minimum would be found at buying 12 lbs of premium loam.

3.) This problem in general form becomes

\[
\text{Minimize } 4x + 7.5y + 13z \\
\text{s.t. } 0.06y + 0.07z \geq 0.05 \\
0.05x + 0.08y \geq 0.03 \\
0.30x + 0.30y + 0.25z \geq 0.26 \\
0.20x + 0.10y + 0.16z \leq 0.15
\]

4.) This problem in general form becomes

\[
\text{Maximize } 1.62(x_1 + x_2 + x_3 + x_4) + 1.08(x_5 + x_6 + x_7 + x_8) \\
\text{s.t. } x_1 + x_5 \leq 3814 \\
x_2 + x_6 \leq 2666 \\
x_3 + x_7 \leq 4016
\]
\[
x_4 + x_8 \leq 1300 \\
-7x_1 + 7x_2 + 13x_3 - 8x_4 \leq 0 \\
-2x_1 + x_2 - 3x_3 + 14x_4 \leq 0 \\
-16x_5 - 2x_6 + 4x_7 - 17x_8 \leq 0 \\
-2x_5 + x_6 - 3x_7 + 14x_8 \leq 0 \\
\]

\[
x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \geq 0
\]

where:

- \( x_1 \) number of barrels of Alkylate in Avgas A
- \( x_2 \) number of barrels of Catalytic-cracked in Avgas A
- \( x_3 \) number of barrels of Straight-run in Avgas A
- \( x_4 \) number of barrels of Isopentane in Avgas A
- \( x_5 \) number of barrels of Alkylate in Avgas B
- \( x_6 \) number of barrels of Catalytic-cracked in Avgas B
- \( x_7 \) number of barrels of Straight-run in Avgas B
- \( x_8 \) number of barrels of Isopentane in Avgas B

Constraint 1:

There are 3814 barrels of Alkylate produced.

Constraint 2:

There are 2666 barrels of Catalytic-cracked produced.

Constraint 3:

There are 4016 barrels of Straight-run produced.

Constraint 4:

There are 1300 barrels of Isopentane produced.

Constraint 5:

\[ 107x_1 + 93x_2 + 87x_3 + 108x_4 \geq 100(x_1 + x_2 + x_3 + x_4) \]

Avgas A must have a PN of at least 100

Constraint 6:

\[ 5x_1 + 8x_2 + 4x_3 + 21x_4 \leq 7(x_1 + x_2 + x_3 + x_4) \]

Avgas A must have an RVP of at most 7

Constraint 7:

\[ 107x_5 + 93x_6 + 87x_7 + 108x_8 \geq 91(x_5 + x_6 + x_7 + x_8) \]

Avgas B must have a PN of at least 91

Constraint 8:
5x5 + 8x6 + 4x7 + 21x8 ≤ 7(x5 + x6 + x7 + x8)
Avgas B must have an RVP of at most 7.

5.) This problem in general form becomes

Minimize 13x + 11y + 16z
s.t. 10x + 9y + 12z = 1200
     15x + 16y + 12z = 2000
     x + 2y    ≤ 60

The cost function comes from adding the cost of the drum of the product with the
cost of the extra labor per drum for products 2 and 3.

The first and second constraints are a result of the fact that the sum of the 2
products must yield 1200 and 2000 gallons weekly.

The third constraint is a result of the restriction on the number of man-hours
available.

Chapter 6 Problems

1.) There are 32 possible ways that the 1's and 0's can be arranged with the 2
exponents. Of these 32, 7 of them are the same (for example, +.1000 × 2⁰ = +.0100 × 2¹.) The same is true for the negative numbers, so altogether, 50 numbers can be
represented using this form.

2.) For this problem, there are 32,768 ways that the numbers can be placed - 8
choices for b₁, 8 choices for b₂, 8 choices for b₃, 8 choices for b₄ and 8 choices for m
to form all of the different positive numbers. Altogether, we can represent less than
65,536 different numbers.

3.) Answers may vary. Students should demonstrate an understanding of in-
creasing magnitude, and examples should show that adding in this manner makes a
difference.

4.) The completed table should look like the following.

   5.)0.137 × 10¹
   + 0.00269 × 10¹
     0.13969 × 10¹
     chopped 0.139 × 10¹
     rounded 0.140 × 10¹

   6.)0.485 × 10⁴
   − 0.482 × 10⁴
     0.300 × 10²
     chopped 0.300 × 10²
     rounded0.300 × 10²
<table>
<thead>
<tr>
<th>$b$</th>
<th>Relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.526430</td>
<td>$2.0 \times 10^{-6}$</td>
</tr>
<tr>
<td>3.526400</td>
<td>$1.0 \times 10^{-5}$</td>
</tr>
<tr>
<td>3.526000</td>
<td>$1.2 \times 10^{-4}$</td>
</tr>
<tr>
<td>3.520000</td>
<td>$1.8 \times 10^{-3}$</td>
</tr>
<tr>
<td>3.500000</td>
<td>$7.5 \times 10^{-3}$</td>
</tr>
<tr>
<td>3.000000</td>
<td>$1.5 \times 10^{-1}$</td>
</tr>
</tbody>
</table>

7.) $0.378 \times 10^4$

+ $0.727 \times 10^4$

$0.1005 \times 10^4$

chopped 0.100 $\times 10^5$

rounded 0.101 $\times 10^5$

9.) $0.356 \times 10^{-2} \div 0.156 \times 10^4$

= $0.22820513 \times 10^{-6}$

chopped: $0.228 \times 10^{-6}$

rounded: $0.228 \times 10^{-6}$

8.) $0.403 \times 10^6$

*$0.197 \times 10^1$

$0.079391 \times 10^7$

chopped 0.793 $\times 10^6$

rounded 0.794 $\times 10^6$
Bibliography


