Experiments with MATLAB®

Cleve Moler

April 6, 2008

Chapter 1

Iteration

An investigation of fixed point iterations introduces the assignment statement, `for` *and* `while` *loops, the* `plot` *function, and the Golden Ratio.*

Start by picking a number, any number. Enter it into MATLAB by typing

```
 x = your number
```

This is a MATLAB *assignment statement*. The number you chose is stored in the *variable* `x` for later use. For example, if you start with

```
x = 3
```

MATLAB responds with

```
x =
     3
```

Next, enter this statement

```
 x = sqrt(1 + x)
```

The abbreviation `sqrt` is the MATLAB name for the square root function. The quantity on the right, $\sqrt{1+x}$, is computed and the result stored back in the variable `x`, overriding the previous value of `x`. Now, repeatedly execute the statement by using the up-arrow key, followed by the enter or return key. Here is what you get when you start with `x = 3`.

```
x =
```

Chapter 3

Calendars and Clocks

Computations involving time, dates, biorhythms and Easter.

Calendars are interesting mathematical objects. The Gregorian calendar was first proposed in 1582. It has been gradually adopted by various countries and churches over the four centuries since then. The British Empire, including the colonies in North America, adopted it in 1752. Turkey did not adopt it until 1923. The Gregorian calendar is now the most widely used calendar in the world, but by no means the only one.

In the Gregorian calendar, a year y is a *leap year* if and only if y is divisible by 4 and not divisible by 100, or is divisible by 400. In MATLAB the following expression must be `true`.

```
mod(y,4) == 0 && mod(y,100) ~= 0 || mod(y,400) == 0
```

For example, 2000 was a leap year, but 2100 will not be a leap year. This rule implies that the Gregorian calendar repeats itself every 400 years. In that 400-year period, there are 97 leap years, 4800 months, 20871 weeks, and 146097 days. The average number of days in a Gregorian calendar year is $365 + \frac{97}{400} = 365.2425$.

The MATLAB function `clock` returns a six-element vector `c` with elements

```
c(1) = year
c(2) = month
c(3) = day
c(4) = hour
c(5) = minute
c(6) = seconds
```

Chapter 4

T Puzzle

A classic puzzle demonstrates complex arithmetic.

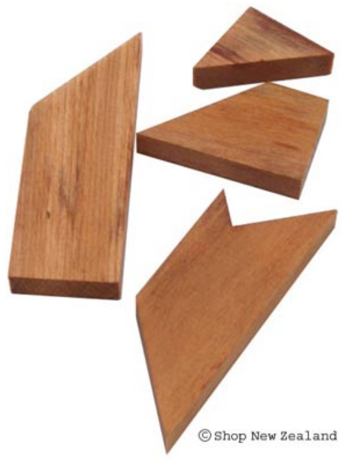

Figure 4.1. *The wooden T puzzle. Photo courtesy of Shop New Zeland,* *http://www.shopnewzealand.co.nz.*

I first saw the wooden T puzzle shown in figure 4.1 at Puzzling World in

Chapter 5

Matrices

MATLAB *began as a matrix calculator.*

The Cartesian coordinate system was developed in the 17th century by the French mathematician and philosopher René Descartes. A pair of numbers corresponds to a point in the plane. We will display the coordinates in a *vector* of length two. In order to work properly with matrix multiplication, we want to think of the vector as a *column* vector, So

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

denotes the point x whose first coordinate is x_1 and second coordinate is x_2. When it is inconvenient to write a vector in this vertical form, we can anticipate MATLAB notation and use a semicolon to separate the two components,

$$x = (\ x_1;\ x_2\ )$$

For example, the point labeled `x` in figure 5.1 has Cartesian coordinates

$$x = (\ 2;\ 4\ )$$

Arithmetic operations on the vectors are defined in natural ways. Addition is defined by

$$x + y = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \end{pmatrix}$$

Multiplication by a single number, or *scalar*, is defined by

$$sx = \begin{pmatrix} sx_1 \\ sx_2 \end{pmatrix}$$

Chapter 6

Fractal Fern

The fractal fern involves 2-by-2 matrices.

The programs `fern` and `finitefern` in the `exm` toolbox produce the *Fractal Fern* described by Michael Barnsley in *Fractals Everywhere* [?]. They generate and plot a potentially infinite sequence of random, but carefully choreographed, points in the plane. The command

```
fern
```

runs forever, producing an increasingly dense plot. The command

```
finitefern(n)
```

generates `n` points and a plot like Figure 6.1. The command

```
finitefern(n,'s')
```

shows the generation of the points one at a time. The command

```
F = finitefern(n);
```

generates, but does not plot, `n` points and returns an array of zeros and ones for use with sparse matrix and image-processing functions.

The `exm` toolbox also includes `fern.jpg`, a 768-by-1024 color image with half a million points that you can view with a browser or a paint program. You can also view the file with

```
F = imread('fern.png');
image(F)
```

Chapter 7

Magic Squares

With origins in centuries old recreational mathematics, magic squares demonstrate MATLAB *array operations.*

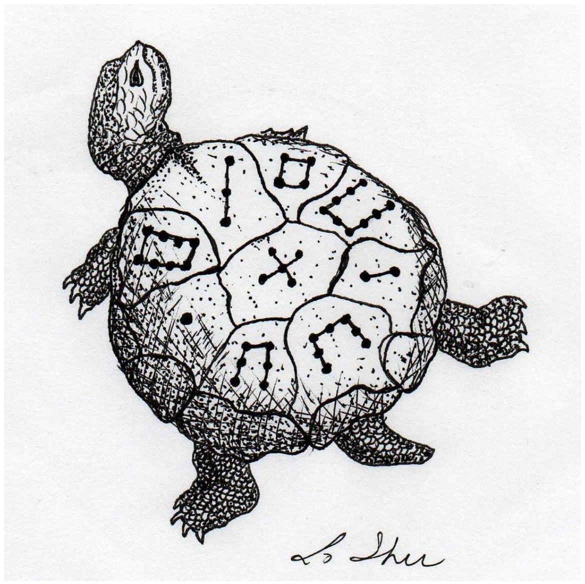

Figure 7.1. *Lo Shu. (Thanks to Byerly Wiser Cline.)*

Chapter 8

TicTacToe Magic

Three simple games are related in a surprising way.

The first of the three games is Pick15. Harold Stark, who was then at the University of Michigan, told me about the game in the late 1960s. I suspect that this is the first time you have heard of it.

The game involves two players. You start by listing the single digit numbers from 1 to 9. You then take turns selecting numbers from the list, attempting to acquire three numbers that add up to 15. Each number can be chosen only once. You may eventually acquire more than three numbers, but you must use exactly three of them to total 15. If neither player can achieve the desired total, the game is a draw.

For example, suppose that Green and Blue are playing. They start with the list.

List	:	1 2 3 4 5 6 7 8 9
Green	:	
Blue	:	

Suppose Green has the first move and chooses 8. Then Blue chooses 4 and Green chooses 2. Now Blue should respond by choosing 5 to prevent Green from getting $2 + 5 + 8 = 15$. Here is the situation after the first two rounds.

List	:	$1\ \cancel{2}\ 3\ \cancel{4}\ \cancel{5}\ 6\ 7\ \cancel{8}\ 9$
Green	:	2 8
Blue	:	4 5

Chapter 9

Game of Life

Conway's Game of Life makes use of sparse matrices.

The "Game of Life" was invented by John Horton Conway, a British-born mathematician who is now a professor at Princeton. The game made its public debut in the October 1970 issue of *Scientific American*, in the "*Mathematical Games*" column written by Martin Gardner. At the time, Gardner wrote

> This month we consider Conway's latest brainchild, a fantastic solitaire pastime he calls "life". Because of its analogies with the rise, fall and alternations of a society of living organisms, it belongs to a growing class of what are called "simulation games" – games that resemble real-life processes. To play life you must have a fairly large checkerboard and a plentiful supply of flat counters of two colors.

Of course, today we can run the simulations on our computers.

The *universe* is an infinite, two-dimensional rectangular grid. The *population* is a collection of grid cells that are marked as *alive.* The population evolves at discrete time steps known as *generations.* At each step, the fate of each cell is determined by the vitality of its eight nearest neighbors and this rule:

- A live cell with two live neighbors, or any cell with three live neigbhors, is alive at the next step.

The fascination of Conway's Game of Life is that this deceptively simple rule leads to an incredible variety of patterns, puzzles, and unsolved mathematical problems – just like real life.

If the initial population consists of only one or two live cells, it expires in one step. If the initial population consists of three live cells then, because of rotational

Chapter 10

Mandelbrot Set

Fractals, topology, complex arithmetic and fascinating computer graphics.

Benoit Mandelbrot is a Polish/French/American mathematician who has spent most of his career at the IBM Watson Research Center in Yorktown Heights, N.Y. He coined the term *fractal* and published a very influential book, *The Fractal Geometry of Nature*, in 1982. An image of the now famous Mandelbrot set appeared on the cover of *Scientific American* in 1985. This was about the time that computer graphical displays were first becoming widely available. Since then, the Mandelbrot set has stimulated deep research topics in mathematics and has also been the basis for an uncountable number of graphics projects, hardware demos, and Web pages.

To get in the mood for the Mandelbrot set, consider the region in the complex plane consisting of the values z_0 for which the trajectories defined by

$$z_{k+1} = z_k^2, \quad k = 0, 1, ...$$

remain bounded at $k \to \infty$. It is easy to see that this set is simply the unit disc, $|z_0| <= 1$, shown in figure 10.1. If $|z_0| <= 1$, the repeated squares remain bounded. If $|z_0| > 1$, the repeated squares are unbounded. The boundary of the unit disc is the unit circle, $|z_0| = 1$. There is nothing very difficult or exciting here.

The definition is the Mandelbrot set is only slightly more complicated. It involves repeatedly adding in the initial point. The Mandelbrot set is the region win the complex plane consisting of the values z_0 for which the trajectories defined by

$$z_{k+1} = z_k^2 + z_0, \quad k = 0, 1, ...$$

remain bounded at $k \to \infty$. That's it. That's the entire definition. It's amazing that such a simple definition can produce such fascinating complexity.

Chapter 11

Linear Equations

The most important task in technical computing.

I am thinking of two numbers. Their average is 3. What are the numbers? Please remember the first thing that pops into your head. I will get back to this problem in a few pages.

Solving systems of simultaneous linear equations is the most important task in technical computing. It is not only important in its own right, it is also a fundamental, and often hidden, component of other more complicated computational tasks.

The very simplest linear equations involve only one unknown. Solve

$$7x = 21$$

The answer, of course, is

$$x = \frac{21}{7} = 3$$

Now solve

$$ax = b$$

The answer, of course, is

$$x = \frac{b}{a}$$

But what if $a = 0$? Then we have to look at b. If $b \neq 0$ then there is no value of x that satisfies

$$0x = b$$

Chapter 12

Google PageRank

The world's largest matrix computation.

One of the reasons why Google™ is such an effective search engine is the PageRank™ algorithm developed by Google's founders, Larry Page and Sergey Brin, when they were graduate students at Stanford University. PageRank is determined entirely by the link structure of the World Wide Web. It is recomputed about once a month and does not involve the actual content of any Web pages or individual queries. Then, for any particular query, Google finds the pages on the Web that match that query and lists those pages in the order of their PageRank.

Imagine surfing the Web, going from page to page by randomly choosing an outgoing link from one page to get to the next. This can lead to dead ends at pages with no outgoing links, or cycles around cliques of interconnected pages. So, a certain fraction of the time, simply choose a random page from the Web. This theoretical random walk is known as a *Markov chain* or *Markov process.* The limiting probability that an infinitely dedicated random surfer visits any particular page is its PageRank. A page has high rank if other pages with high rank link to it.

Let W be the set of Web pages that can be reached by following a chain of hyperlinks starting at some root page, and let n be the number of pages in W. For Google, the set W actually varies with time, but by June 2004, n was over 4 billion. Let G be the n-by-n *connectivity matrix* of a portion of the Web, that is, $g_{ij} = 1$ if there is a hyperlink to page i from page j and $g_{ij} = 0$ otherwise. The matrix G can be huge, but it is very sparse. Its jth column shows the links on the jth page. The number of nonzeros in G is the total number of hyperlinks in W.

Chapter 13

Ordinary Differential Equations

Mathematical models in many different fields.

Systems of differential equations form the basis of mathematical models in a wide range of fields – from engineering and physical sciences to finance and biological sciences. Differential equations are relations between unknown functions and their derivatives. Computing numerical solutions to differential equations is one of the most important tasks in technical computing, and one of the strengths of MATLAB.

If you have studied calculus, you have learned a kind of mechanical process for differentiating functions represented by formulas involving powers, trig functions, and the like. You know that the derivative of x^3 is $3x^2$ and you may remember that the derivative of $\tan x$ is $1 + \tan^2 x$. That kind of differentiation is important and useful, but not our primary focus here. We are interested in situations where the functions are not known and cannot be represented by simple formulas. We will compute numerical approximations to the values of a function at enough points to print a table or plot a graph.

Imagine you are travelling on a mountain road. Your altitude varies as you travel. The altitude can be regarded as a function of time, or as a function of longitude and latitude, or as a function of the distance you have traveled. Let's consider the latter. Let x denote the distance traveled and $y = y(x)$ denote the altitude. If you happen to be carrying an altimeter with you, or you have a deluxe GPS system, you can collect enough values to plot a graph of altitude versus distance, like the first plot in figure 13.1.

Suppose you see a sign saying that you are on a 6% uphill grade. For some

Chapter 14

Exponential Function

The function e^x.

The exponential function is denoted mathematically by e^t and in MATLAB by `exp(t)`. This function is the solution to the world's simplest, and perhaps most important, differential equation

$$\dot{y} = ky$$

This differential equation is the basis for any mathematical model describing the time evolution of a quantity with a rate of production that is proportional to the quantity itself. Such models include populations, investments, feedback, and radioactive decay. We are using t for the independent variable, y for the dependent variable, k for the proportionality constant, and

$$\dot{y} = \frac{dy}{dt}$$

for the derivative with respect to t. We are looking for a function that is proportional to its own derivative.

Let's start by examining the function

$$y = 2^t$$

We know what 2^t means if t is an integer, 2^t is the t-th power of 2.

$$2^{-1} = 1/2, \quad 2^0 = 1, \quad 2^1 = 1, \quad 2^2 = 4, ...$$

We also know what 2^t means if $t = p/q$ is a rational number, the ratio of two integers, $2^{p/q}$ is the q-th root of the p-th power of 2.

$$2^{1/2} \;=\; \sqrt{2} = 1.4142...,$$

Chapter 15

Predators and Prey

Models of population growth.

The simplest model for the growth, or decay, of a population says that the growth rate, or the decay rate, is proportional to the size of the population itself. Increasing or decreasing the size of a the population results a proportional increase or decrease in the number of births and deaths. Mathematically, this is described by the differential equation

$$\dot{y} = ky$$

The proportionality constant k relates the size of the population, $y(t)$, to its rate of growth, $\dot{y}(t)$. If k is positive, the population increases; if k is negative, the population decreases.

As we know, the solution to this equation is a function $y(t)$ that is proportional to the exponential function

$$y(t) = \eta e^{kt}$$

where $\eta = y(0)$.

This simple model is appropriate in the initial stages of growth when there are no restrictions or constraints on the population. A small sample of bacteria in a large Petri dish, for example. But in more realistic situations there are limits to growth, such as finite space or food supply. A more realistic model says that the population competes with itself. As the population increases, its growth rate decreases linearly. The differential equation is sometimes called the *logistic* equation.

$$\dot{y} = k(1 - \frac{y}{\mu})y$$

Chapter 16

Shallow Water Equations

The shallow water equations model tsunamis and waves in bathtubs.

This chapter is more advanced mathematically than earlier chapters, but you might still find it interesting even if you do not master the mathematical details.

The shallow water equations model the propogation of disturbances in water and other incompressible fluids. The underlying assumption is that the depth of the fluid is small compared to the wave length of the disturbance. For example, we do not ordinary think of the Indian Ocean as being shallow. The depth is two or three kilometers. But the devastating tsunami in the Indian Ocean on December 26, 2004 involved waves that were dozens or hundred of kilometers long. So the shallow water approximation provides a reasonable model in this situation.

The equations are derived from the principles of conservation of mass and conservation of momemtum. The independent variables are time, t, and two space coordinates, x and y. The dependent variables are the fluid height or depth, h, and the two-dimensional fluid velocity field, u and v. With the proper choice of units, the conserved quantities are mass, which is proportional to h, and momentum, which is proportional to uh and vh. The force acting on the fluid is gravity, represented by the gravitational constant, g. The partial differential equations are:

$$\frac{\partial h}{\partial t} + \frac{\partial (uh)}{\partial x} + \frac{\partial (vh)}{\partial y} = 0$$

$$\frac{\partial (uh)}{\partial t} + \frac{\partial (u^2 h + \frac{1}{2} g h^2)}{\partial x} + \frac{\partial (uvh)}{\partial y} = 0$$

$$\frac{\partial (vh)}{\partial t} + \frac{\partial (uvh)}{\partial x} + \frac{\partial (v^2 h + \frac{1}{2} g h^2)}{\partial x} = 0$$