

# Numerical Computing with MATLAB



Cleve B. Moler

siam



# Numerical Computing with MATLAB

Cleve B. Moler  
The MathWorks, Inc.



Society for Industrial and Applied Mathematics  
Philadelphia

Copyright © 2004 by Cleve B. Moler

Published by the Society for Industrial and Applied Mathematics.

10 9 8 7 6 5 4 3 2

All rights reserved. Printed in the United States of America. No part of this book may be reproduced, stored, or transmitted in any manner without the written permission of the publisher. For information, write to the Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, PA 19104-2688.

Google and PageRank are trademarks of Google, Inc.

MATLAB is a registered trademark of The MathWorks, Inc. For MATLAB product information, please contact The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098 USA, 508-647-7000, Fax: 508-647-7101, *info@mathworks.com*, *www.mathworks.com/*

Web edition published by The MathWorks, Inc.  
*http://www.mathworks.com/moler*

### **Library of Congress Cataloging-in-Publication Data**

Moler, Cleve B.

Numerical computing with MATLAB / Cleve B. Moler.

p. cm.

Includes bibliographical references and index.

ISBN 0-89871-560-1 (pbk.)

1. Numerical analysis—Data processing. 2. MATLAB. I. Title.

QA297.M625 2004

518'.0285—dc22

2004045360

**siam** is a registered trademark.

# Preface

*Numerical Computing with MATLAB* is a textbook for an introductory course in numerical methods, MATLAB, and technical computing. The emphasis is on informed use of mathematical software. We want you learn enough about the mathematical functions in MATLAB that you will be able to use them correctly, appreciate their limitations, and modify them when necessary to suit your own needs. The topics include

- introduction to MATLAB,
- linear equations,
- interpolation,
- zero finding,
- least squares,
- quadrature,
- ordinary differential equations,
- random numbers,
- Fourier analysis,
- eigenvalues and singular values,
- partial differential equations.

George Forsythe initiated a software-based numerical methods course at Stanford University in the late 1960s. The textbooks by Forsythe, Malcolm, and Moler [20] and Kahaner, Moler, and Nash [33] that evolved from the Stanford course were based upon libraries of Fortran subroutines.

This textbook is based upon MATLAB. NCM, a collection of over 70 M-files, forms an essential part of the book. Many of the over 200 exercises involve modifying and extending the programs in NCM. The book also makes extensive use of computer graphics, including interactive graphical expositions of numerical algorithms.

The prerequisites for the course, and the book, include

- calculus,
- some familiarity with ordinary differential equations,



- some familiarity with matrices,
- some computer programming experience.

If you've never used MATLAB before, the first chapter will help you get started. If you're already familiar with MATLAB, you can glance over most of the first chapter quickly. Everyone should read the section in the first chapter about floating-point arithmetic.

There is probably too much material here for a one-quarter or one-semester course. Plan to cover the first several chapters and then choose the portions of the last four chapters that interest you.

Make sure that the NCM collection is installed on your network or your personal computer as you read the book. The software is available from a Web site devoted to the book [3]:

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

There are three types of NCM files:

- `gui` files: interactive graphical demonstrations;
- `tx` files: textbook implementations of built-in MATLAB functions;
- others: miscellaneous files, primarily associated with exercises.



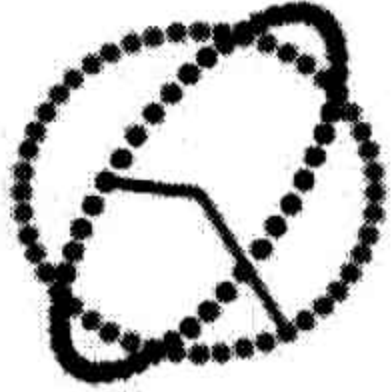
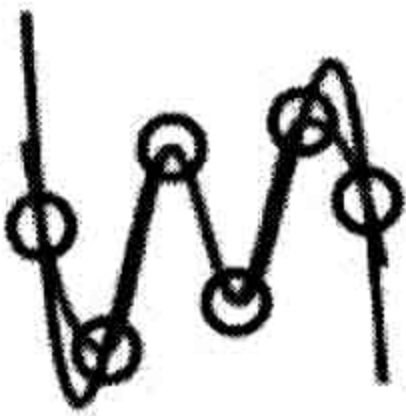
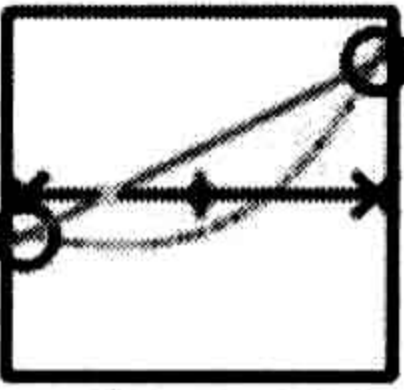



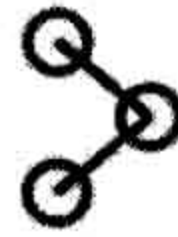
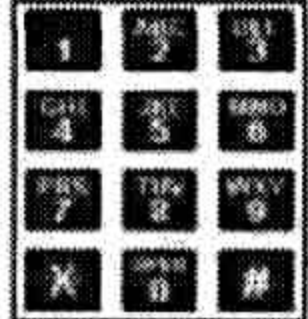
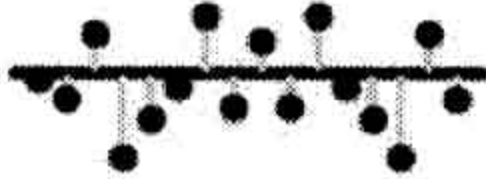

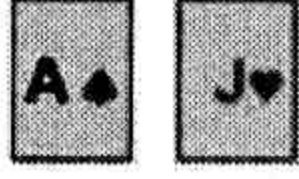
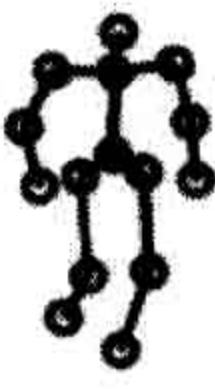


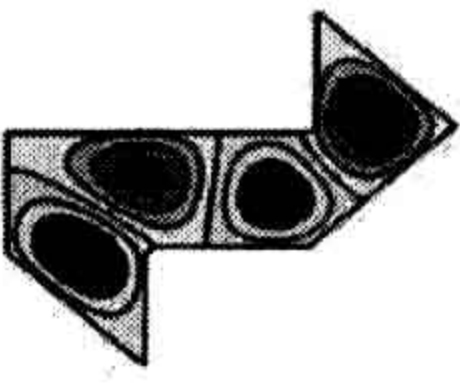

When you have NCM available,

`ncmgui`

produces the figure shown on the next page. Each thumbnail plot is actually a push button that launches the corresponding `gui`.

This book would not have been possible without the people at The MathWorks and at SIAM. Both groups are professional, creative, and delightful to work with. They have been especially supportive of this book project. Out of the many friends and colleagues who have made specific contributions, I want to mention five in particular. Kathryn Ann Moler has used early drafts of the book several times in courses at Stanford and has been my best critic. Tim Davis and Charlie Van Loan wrote especially helpful reviews. Lisl Urban did an immaculate editing job. My wife Patsy has lived with my work habits and my laptop and loves me anyway. Thanks, everyone.

Cleve Moler  
March 28, 2004

 fern	 floatgui	<table><tr><td>2</td><td>6</td><td>0</td><td>3</td><td>1</td></tr><tr><td>5</td><td>6</td><td>1</td><td>5</td><td>9</td></tr><tr><td>8</td><td>0</td><td>4</td><td>2</td><td>2</td></tr><tr><td>1</td><td>8</td><td>6</td><td>8</td><td>4</td></tr><tr><td>9</td><td>3</td><td>8</td><td>6</td><td>4</td></tr></table> lugui	2	6	0	3	1	5	6	1	5	9	8	0	4	2	2	1	8	6	8	4	9	3	8	6	4	 eigshow	 interpgui
2	6	0	3	1																									
5	6	1	5	9																									
8	0	4	2	2																									
1	8	6	8	4																									
9	3	8	6	4																									
 fzerogui	 censusgui	 quadgui	 lorenzgui	 swinger																									
 touchtone	 fftgui	 randgui	 blackjack	 walker																									
 eigsvdgui	 waves	 pdegui	 pennymelt	<div>ncm books</div> <div>helpwin ncmgui</div> <div>helpwin ncm</div> <div>close</div>																									

ncmgui.



# Contents

<b>Preface</b>	<b>ix</b>
<b>1 Introduction to MATLAB</b>	<b>1</b>
1.1 The Golden Ratio . . . . .	1
1.2 Fibonacci Numbers . . . . .	7
1.3 Fractal Fern . . . . .	13
1.4 Magic Squares . . . . .	18
1.5 Cryptography . . . . .	26
1.6 The $3n + 1$ Sequence . . . . .	31
1.7 Floating-Point Arithmetic . . . . .	33
1.8 Further Reading . . . . .	41
Exercises . . . . .	41
<b>2 Linear Equations</b>	<b>53</b>
2.1 Solving Linear Systems . . . . .	53
2.2 The MATLAB Backslash Operator . . . . .	54
2.3 A 3-by-3 Example . . . . .	54
2.4 Permutation and Triangular Matrices . . . . .	56
2.5 LU Factorization . . . . .	57
2.6 Why Is Pivoting Necessary? . . . . .	58
2.7 <code>lutsx</code> , <code>bslashtx</code> , <code>lugu</code> . . . . .	60
2.8 Effect of Roundoff Errors . . . . .	63
2.9 Norms and Condition Numbers . . . . .	66
2.10 Sparse Matrices and Band Matrices . . . . .	72
2.11 PageRank and Markov Chains . . . . .	74
2.12 Further Reading . . . . .	81
Exercises . . . . .	82
<b>3 Interpolation</b>	<b>93</b>
3.1 The Interpolating Polynomial . . . . .	93
3.2 Piecewise Linear Interpolation . . . . .	98
3.3 Piecewise Cubic Hermite Interpolation . . . . .	99
3.4 Shape-Preserving Piecewise Cubic . . . . .	100
3.5 Cubic Spline . . . . .	102

---

3.6	pchiptx, splinetx . . . . .	105
3.7	interpgui . . . . .	108
	Exercises . . . . .	110
<b>4</b>	<b>Zeros and Roots</b>	<b>117</b>
4.1	Bisection . . . . .	117
4.2	Newton's Method . . . . .	119
4.3	A Perverse Example . . . . .	121
4.4	Secant Method . . . . .	122
4.5	Inverse Quadratic Interpolation . . . . .	123
4.6	Zeroin . . . . .	124
4.7	fzerotx, feval . . . . .	124
4.8	fzerogui . . . . .	129
4.9	Value Finding and Reverse Interpolation . . . . .	132
4.10	Optimization and fmintx . . . . .	132
	Exercises . . . . .	135
<b>5</b>	<b>Least Squares</b>	<b>141</b>
5.1	Models and Curve Fitting . . . . .	141
5.2	Norms . . . . .	143
5.3	censusgui . . . . .	144
5.4	Householder Reflections . . . . .	145
5.5	The QR Factorization . . . . .	147
5.6	Pseudoinverse . . . . .	152
5.7	Rank Deficiency . . . . .	154
5.8	Separable Least Squares . . . . .	156
5.9	Further Reading . . . . .	159
	Exercises . . . . .	159
<b>6</b>	<b>Quadrature</b>	<b>167</b>
6.1	Adaptive Quadrature . . . . .	167
6.2	Basic Quadrature Rules . . . . .	168
6.3	quadtx, quadgui . . . . .	170
6.4	Specifying Integrands . . . . .	173
6.5	Performance . . . . .	175
6.6	Integrating Discrete Data . . . . .	177
6.7	Further Reading . . . . .	179
	Exercises . . . . .	179
<b>7</b>	<b>Ordinary Differential Equations</b>	<b>187</b>
7.1	Integrating Differential Equations . . . . .	187
7.2	Systems of Equations . . . . .	188
7.3	Linearized Differential Equations . . . . .	189
7.4	Single-Step Methods . . . . .	191
7.5	The BS23 Algorithm . . . . .	194
7.6	ode23tx . . . . .	196
7.7	Examples . . . . .	199



7.8	Lorenz Attractor . . . . .	202
7.9	Stiffness . . . . .	204
7.10	Events . . . . .	208
7.11	Multistep Methods . . . . .	212
7.12	The MATLAB ODE Solvers . . . . .	212
7.13	Errors . . . . .	213
7.14	Performance . . . . .	217
7.15	Further Reading . . . . .	219
	Exercises . . . . .	219
<b>8</b>	<b>Fourier Analysis</b>	<b>237</b>
8.1	Touch-Tone Dialing . . . . .	237
8.2	Finite Fourier Transform . . . . .	241
8.3	fftgui . . . . .	242
8.4	Sunspots . . . . .	244
8.5	Periodic Time Series . . . . .	248
8.6	Fast Finite Fourier Transform . . . . .	249
8.7	ffttx . . . . .	250
8.8	fftmatrix . . . . .	251
8.9	Other Fourier Transforms and Series . . . . .	252
8.10	Further Reading . . . . .	254
	Exercises . . . . .	254
<b>9</b>	<b>Random Numbers</b>	<b>257</b>
9.1	Pseudorandom Numbers . . . . .	257
9.2	Uniform Distribution . . . . .	257
9.3	Normal Distribution . . . . .	260
9.4	randtx, randntx . . . . .	263
	Exercises . . . . .	265
<b>10</b>	<b>Eigenvalues and Singular Values</b>	<b>269</b>
10.1	Eigenvalue and Singular Value Decompositions . . . . .	269
10.2	A Small Example . . . . .	272
10.3	eigshow . . . . .	273
10.4	Characteristic Polynomial . . . . .	275
10.5	Symmetric and Hermitian Matrices . . . . .	276
10.6	Eigenvalue Sensitivity and Accuracy . . . . .	277
10.7	Singular Value Sensitivity and Accuracy . . . . .	283
10.8	Jordan and Schur Forms . . . . .	284
10.9	The QR Algorithm . . . . .	285
10.10	eigsvdgui . . . . .	287
10.11	Principal Components . . . . .	289
10.12	Circle Generator . . . . .	293
10.13	Further Reading . . . . .	298
	Exercises . . . . .	298



---

<b>11</b>	<b>Partial Differential Equations</b>	<b>307</b>
11.1	Model Problems . . . . .	307
11.2	Finite Difference Methods . . . . .	308
11.3	Matrix Representation . . . . .	310
11.4	Numerical Stability . . . . .	312
11.5	The L-Shaped Membrane . . . . .	314
	Exercises . . . . .	319
	<b>Bibliography</b>	<b>327</b>
	<b>Index</b>	<b>332</b>



## Chapter 1

# Introduction to MATLAB

This book is an introduction to two subjects: MATLAB and numerical computing. This first chapter introduces MATLAB by presenting several programs that investigate elementary, but interesting, mathematical problems. If you already have some experience programming in another language, we hope that you can see how MATLAB works by simply studying these programs.

If you want a more comprehensive introduction, an on-line manual from The MathWorks is available. Select **Help** in the toolbar atop the MATLAB command window, then select **MATLAB Help** and **Getting Started**. A PDF version is available under **Printable versions**. The document is also available from The MathWorks Web site [42]. Many other manuals produced by The MathWorks are available on line and from the Web site.

A list of over 600 MATLAB-based books by other authors and publishers, in several languages, is available at [43]. Three introductions to MATLAB are of particular interest here: a relatively short primer by Sigmon and Davis [52], a medium-sized, mathematically oriented text by Higham and Higham [30], and a large, comprehensive manual by Hanselman and Littlefield [28].

You should have a copy of MATLAB close at hand so you can run our sample programs as you read about them. All of the programs used in this book have been collected in a directory (or folder) named

NCM

(The directory name is the initials of the book title.) You can either start MATLAB in this directory or use

`pathtool`

to add the directory to the MATLAB path.

## 1.1 The Golden Ratio

What is the world's most interesting number? Perhaps you like  $\pi$ , or  $e$ , or 17. Some people might vote for  $\phi$ , the *golden ratio*, computed here by our first MATLAB statement.



```
phi = (1 + sqrt(5))/2
```

This produces

```
phi =  
1.6180
```

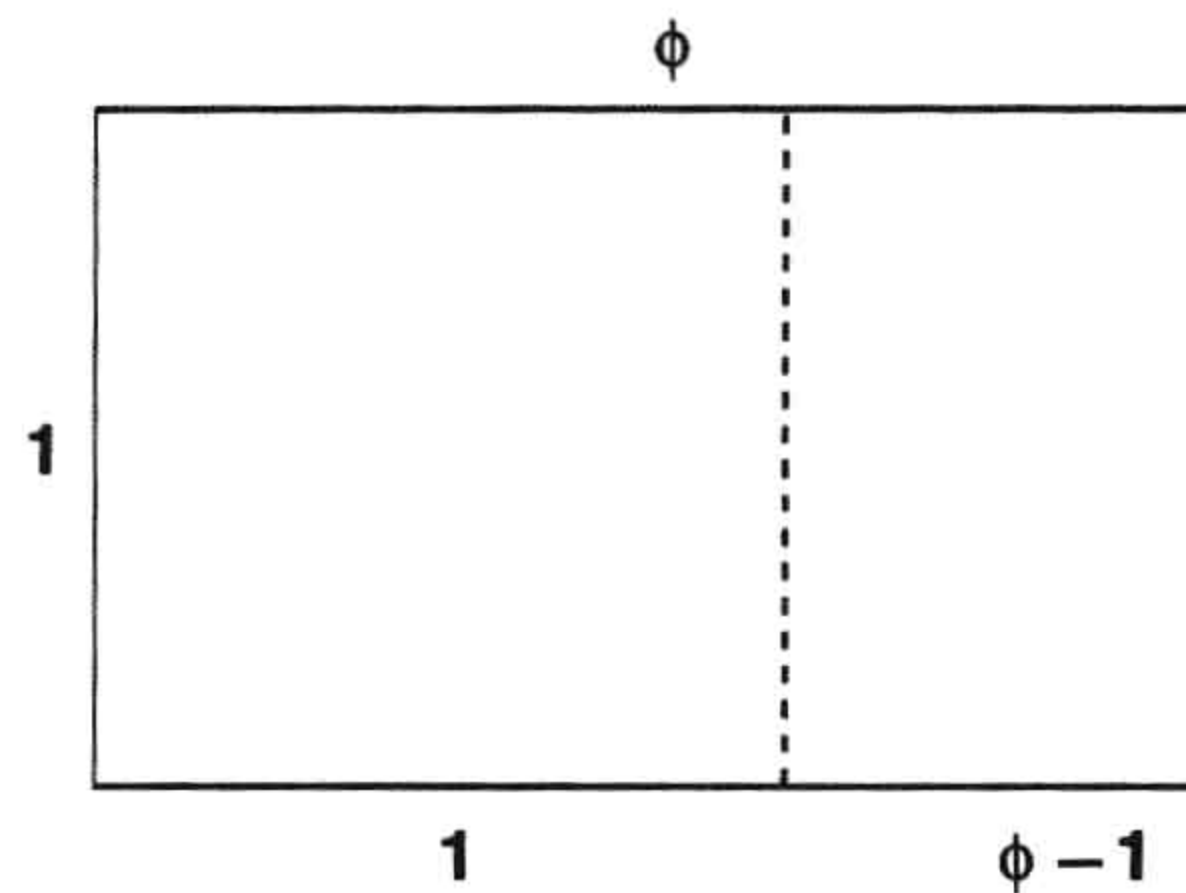
Let's see more digits.

```
format long  
phi
```

```
phi =  
1.61803398874989
```

This didn't recompute  $\phi$ , it just displayed 15 significant digits instead of 5.

The golden ratio shows up in many places in mathematics; we'll see several in this book. The golden ratio gets its name from the golden rectangle, shown in Figure 1.1. The golden rectangle has the property that removing a square leaves a smaller rectangle with the same shape.



**Figure 1.1.** *The golden rectangle.*

Equating the aspect ratios of the rectangles gives a defining equation for  $\phi$ :

$$\frac{1}{\phi} = \frac{\phi - 1}{1}.$$

This equation says that you can compute the reciprocal of  $\phi$  by simply subtracting one. How many numbers have that property?

Multiplying the aspect ratio equation by  $\phi$  produces the polynomial equation

$$\phi^2 - \phi - 1 = 0.$$

The roots of this equation are given by the quadratic formula:

$$\phi = \frac{1 \pm \sqrt{5}}{2}.$$

The positive root is the golden ratio.



If you have forgotten the quadratic formula, you can ask MATLAB to find the roots of the polynomial. MATLAB represents a polynomial by the vector of its coefficients, in descending order. So the vector

```
p = [1 -1 -1]
```

represents the polynomial

$$p(x) = x^2 - x - 1.$$

The roots are computed by the `roots` function.

```
r = roots(p)
```

produces

```
r =  
-0.61803398874989  
1.61803398874989
```

These two numbers are the only numbers whose reciprocal can be computed by subtracting one.

You can use the Symbolic Toolbox, which connects MATLAB to Maple, to solve the aspect ratio equation without converting it to a polynomial. The equation is represented by a character string. The `solve` function finds two solutions.

```
r = solve('1/x = x-1')
```

produces

```
r =  
[ 1/2*5^(1/2)+1/2]  
[ 1/2-1/2*5^(1/2)]
```

The `pretty` function displays the results in a way that resembles typeset mathematics.

```
pretty(r)
```

produces

```

      1/2
[      ]
[1/2  5  + 1/2]
[      ]
[      1/2]
[1/2 - 1/2 5  ]

```

The variable `r` is a vector with two components, the symbolic forms of the two solutions. You can pick off the first component with

```
phi = r(1)
```

which produces

```
phi =  
1/2*5^(1/2)+1/2
```



This expression can be converted to a numerical value in two different ways. It can be evaluated to any number of digits using variable-precision arithmetic with the `vpa` function.

```
vpa(phi, 50)
```

produces 50 digits.

```
1.6180339887498948482045868343656381177203091798058
```

It can also be converted to double-precision floating point, which is the principal way that MATLAB represents numbers, with the `double` function.

```
phi = double(phi)
```

produces

```
phi =  
1.61803398874989
```

The aspect ratio equation is simple enough to have closed-form symbolic solutions. More complicated equations have to be solved approximately. The `inline` function is a quick way to convert character strings to objects that can be arguments to the MATLAB functions that operate on other functions.

```
f = inline('1/x - (x-1)');
```

defines  $f(x) = 1/x - (x - 1)$  and produces

```
f =  
    Inline function:  
    f(x) = 1/x - (x-1)
```

The graph of  $f(x)$  over the interval  $0 \leq x \leq 4$  shown in Figure 1.2 is obtained with

```
ezplot(f, 0, 4)
```

The name `ezplot` stands for “easy plot,” although some of the English-speaking world would pronounce it “e-zed plot.” Even though  $f(x)$  becomes infinite as  $x \rightarrow 0$ , `ezplot` automatically picks a reasonable vertical scale.

The statement

```
phi = fzero(f, 1)
```

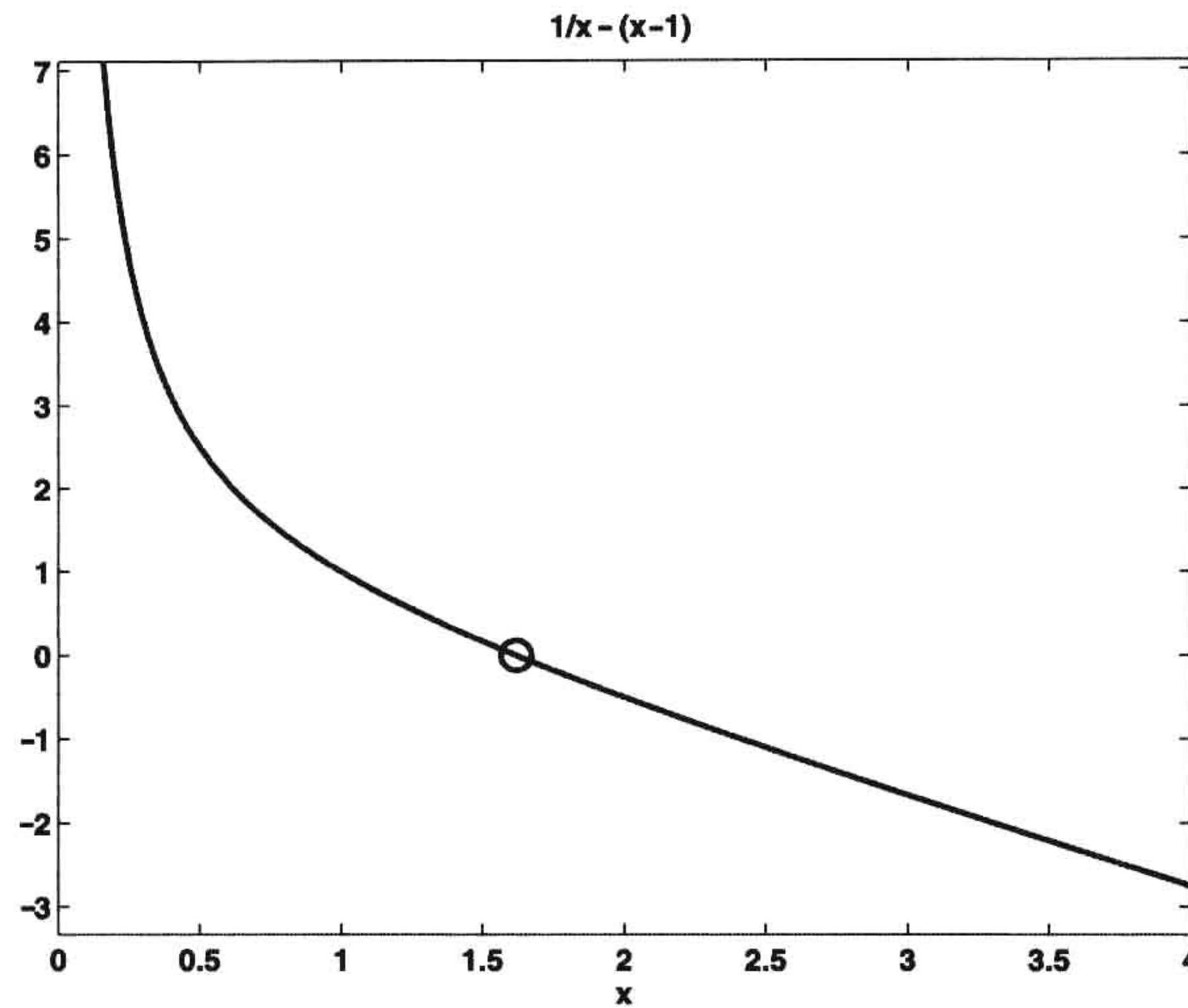
looks for a zero of  $f(x)$  near  $x = 1$ . It produces an approximation to  $\phi$  that is accurate to almost full precision. The result can be inserted in Figure 1.2 with

```
hold on  
plot(phi, 0, 'o')
```

The following MATLAB program produces the picture of the golden rectangle shown in Figure 1.1. The program is contained in an M-file named `goldrect.m`, so issuing the command

```
goldrect
```



Figure 1.2.  $f(\phi) = 0$ .

runs the script and creates the picture.

```
% GOLDRECT Plot the golden rectangle

phi = (1+sqrt(5))/2;
x = [0 phi phi 0 0];
y = [0 0 1 1 0];
u = [1 1];
v = [0 1];
plot(x,y,'b',u,v,'b--')
text(phi/2,1.05,'\phi')
text((1+phi)/2,-.05,'\phi - 1')
text(-.05,.5,'1')
text(.5,-.05,'1')
axis equal
axis off
set(gcf,'color','white')
```

The vectors  $x$  and  $y$  each contain five elements. Connecting consecutive  $(x_k, y_k)$  pairs with straight lines produces the outside rectangle. The vectors  $u$  and  $v$  each contain two elements. The line connecting  $(u_1, v_1)$  with  $(u_2, v_2)$  separates the rectangle into the square and the smaller rectangle. The `plot` command draws these lines—the  $x - y$  lines in solid blue and the  $u - v$  line in dashed blue. The next four statements place text at various points; the string `'\phi'` denotes the Greek letter. The two `axis` statements cause the scaling in the  $x$  and  $y$  directions to be equal and then turn off the display of the axes. The last statement sets the background color of `gcf`, which stands for *get current figure*, to white.



A *continued fraction* is an infinite expression of the form

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \dots}}}$$

If all the  $a_k$ 's are equal to 1, the continued fraction is another representation of the golden ratio:

$$\phi = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \dots}}}$$

The following MATLAB function generates and evaluates truncated continued fraction approximations to  $\phi$ . The code is stored in an M-file named `goldfract.m`.

```
function goldfract(n)
%GOLDFRACT    Golden ratio continued fraction.
%  GOLDFRACT(n) displays n terms.

p = '1';
for k = 1:n
    p = ['1+1/(' p ')'];
end
p

p = 1;
q = 1;
for k = 1:n
    s = p;
    p = p + q;
    q = s;
end
p = sprintf('%d/%d',p,q)

format long
p = eval(p)

format short
err = (1+sqrt(5))/2 - p
```

The statement

```
goldfract(6)
```

produces

```
p =
1+1/(1+1/(1+1/(1+1/(1+1/(1+1/(1)))))))

p =
21/13
```



```
p =
    1.61538461538462
```

```
err =
    0.0026
```

The three  $p$ 's are all different representations of the same approximation to  $\phi$ .

The first  $p$  is the continued fraction truncated to six terms. There are six right parentheses. This  $p$  is a string generated by starting with a single '1' (that's `goldfract(0)`) and repeatedly inserting the string '1+1/' in front and the string ')' in back. No matter how long this string becomes, it is a valid MATLAB expression.

The second  $p$  is an "ordinary" fraction with a single integer numerator and denominator obtained by collapsing the first  $p$ . The basis for the reformulation is

$$1 + \frac{1}{\frac{p}{q}} = \frac{p+q}{p}.$$

So the iteration starts with

$$\frac{1}{1}$$

and repeatedly replaces the fraction

$$\frac{p}{q}$$

with

$$\frac{p+q}{p}.$$

The statement

```
p = sprintf('%d/%d',p,q)
```

prints the final fraction by formatting  $p$  and  $q$  as decimal integers and placing a '/' between them.

The third  $p$  is the same number as the first two  $p$ 's, but is represented as a conventional decimal expansion, obtained by having the MATLAB `eval` function actually do the division expressed in the second  $p$ .

The final quantity `err` is the difference between  $p$  and  $\phi$ . With only 6 terms, the approximation is accurate to less than 3 digits. How many terms does it take to get 10 digits of accuracy?

As the number of terms  $n$  increases, the truncated continued fraction generated by `goldfract(n)` theoretically approaches  $\phi$ . But limitations on the size of the integers in the numerator and denominator, as well as roundoff error in the actual floating-point division, eventually intervene. Exercise 1.3 asks you to investigate the limiting accuracy of `goldfract(n)`.

## 1.2 Fibonacci Numbers

Leonardo Pisano Fibonacci was born around 1170 and died around 1250 in Pisa in what is now Italy. He traveled extensively in Europe and Northern Africa. He wrote several