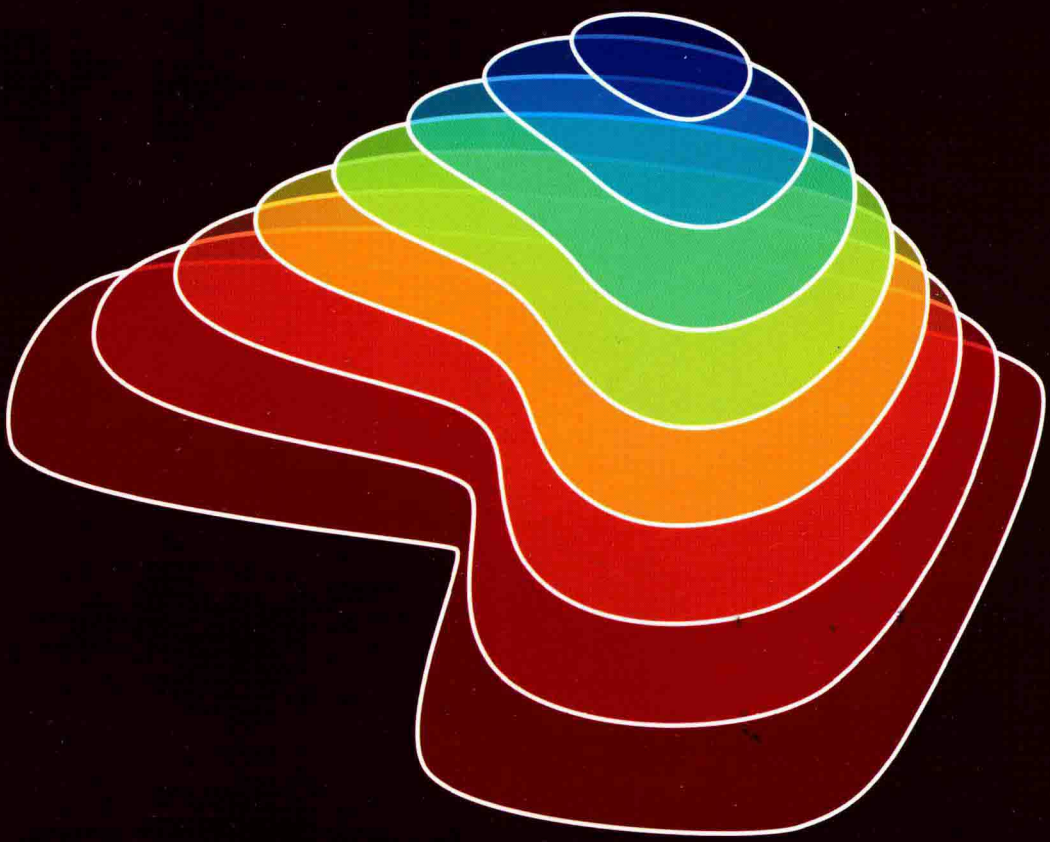
Experiments with MATLAB

MATLAB之父：编程实践

（英文版）



Cleve Moler
MathWorks

MATLAB® examples

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

Experiments with MATLAB

MATLAB 之父：编程实践
（英文版）

Cleve Moler

北京航空航天大学出版社

版权贸易合同登记号　图字:01－2013－5836

图书在版编目(CIP)数据

MATLAB之父：编程实践 = Experiments with MATLAB：英文 /（美）莫勒（Moler, C. B.）著. --北京：北京航空航天大学出版社，2013.12
ISBN 978－7－5124－1229－3

Ⅰ. ①M… Ⅱ. ①莫… Ⅲ. ①Matlab软件－程序设计－教材－英文 Ⅳ. ①TP317

中国版本图书馆CIP数据核字(2013)第187632号

Experiments with MATLAB
MATLAB之父：编程实践
(英文版)
Cleve Moler
责任编辑　陈守平

*

北京航空航天大学出版社出版发行
北京市海淀区学院路37号(邮编100191)　http://www.buaapress.com.cn
发行部电话：(010)82317024　传真：(010)82328026
读者信箱：goodtextbook@126.com　邮购电话：(010)82316936
涿州市新华印刷有限公司印装　　各地书店经销

*

开本：710×1 000　1/16　印张：18.5　字数：484千字
2013年12月第1版　2013年12月第1次印刷　印数：2 000册
ISBN 978－7－5124－1229－3　定价：68.00元

Preface

Welcome to *Experiments with MATLAB*. This is not a conventional book. It is currently available only via the Internet, at no charge, from

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

There may eventually be a hardcopy edition, but not right away*.

Although MATLAB is now a full-fledged Technical Computing Environment, it started in the late 1970s as a simple "Matrix Laboratory". We want to build on this laboratory tradition by describing a series of experiments involving applied mathematics, technical computing, and MATLAB programming.

We expect that you already know something about high school level material in geometry, algebra, and trigonometry. We will introduce ideas from calculus, matrix theory, and ordinary differential equations, but we do not assume that you have already taken courses in the subjects. In fact, these experiments are useful supplements to such courses.

We also expect that you have some experience with computers, perhaps with word processors or spread sheets. If you know something about programming in languages like C or Java, that will be helpful, but not required. We will introduce MATLAB by way of examples. Many of the experiments involve understanding and modifying MATLAB scripts and functions that we have already written.

You should have access to MATLAB and to our `exm` toolbox, the collection of programs and data that are described in *Experiments with MATLAB*. We hope you will not only use these programs, but will read them, understand them, modify them, and improve them. The `exm` toolbox is the apparatus in our "Laboratory".

You will want to have MATLAB handy. For information about the Student Version, see

`http://www.mathworks.com/academia/student_version`

For an introduction to the mechanics of using MATLAB, see the videos at

*This preface was from the original internet version. The current hardcopy version is now brought to the readers, authorized by Cleve Moler.

```
http://www.mathworks.com/academia/student_version/start.html
```

For documentation, including "Getting Started", see

```
http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html
```

For user contributed programs, programming contests, and links into the world-wide MATLAB community, check out

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

To get started, download the `exm` toolbox, use `pathtool` to add `exm` to the MATLAB path, and run

```
exmgui
```

This should generate Figure 1. You can click the icons to preview some of the experiments.

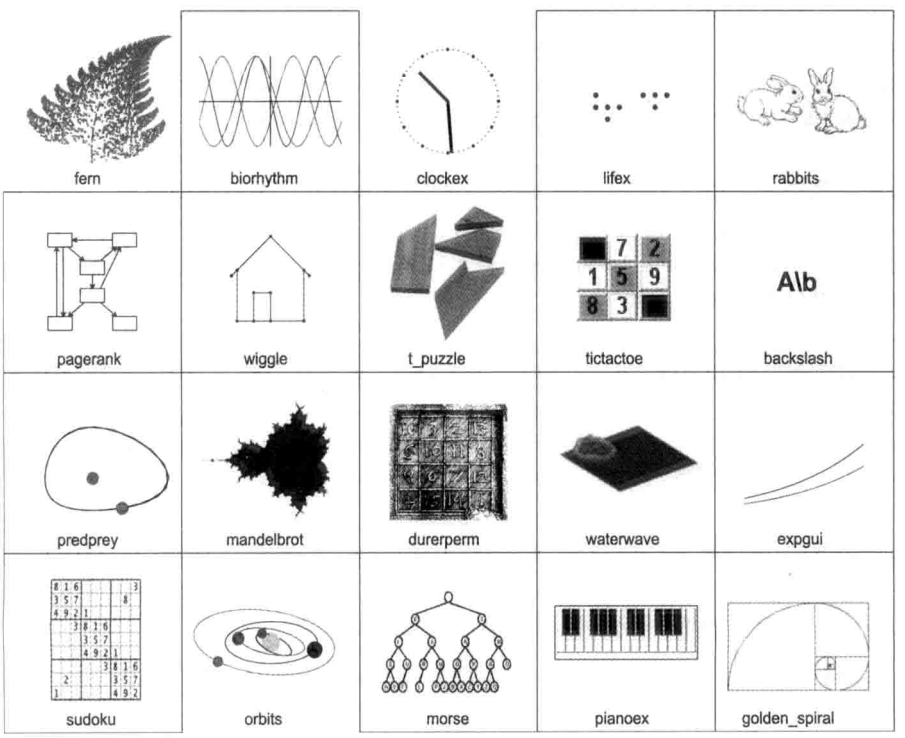


Figure 1. *exmgui provides a starting point for some of the experiments.*

You will want to make frequent use of the MATLAB help and documentation facilities. To quickly learn how to use the command or function named `xxx`, enter

```
help xxx
```

For more extensive information about `xxx`, use

```
doc xxx
```

We hope you will find the experiments interesting, and that you will learn how to use MATLAB along the way. Each chapter concludes with a "Recap" section that is actually an executable MATLAB program. For example, you can review the Magic Squares chapter by entering

```
magic_recap
```

Better yet, enter

```
edit magic_recap
```

and run the program cell-by-cell by simultaneously pressing the **Ctrl-Shift-Enter** keys.

A fairly new MATLAB facility is the `publish` command. You can get a nicely formatted web page about `magic_recap` with

```
publish magic_recap
```

If you want to concentrate on learning MATLAB, make sure you read, run, and understand the recaps.

Cleve Moler
Natick, MA and Santa Fe, NM
September 4, 2013

Contents

Chapter 1
Iteration

Iteration is a key element in much of technical computation. Examples involving the Golden Ratio introduce the MATLAB *assignment statement,* `for` *and* `while` *loops, and the* `plot` *function.*

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`.

Somewhere on your computer keyboard, probably in the lower right corner, you should be able to find four arrow keys. These are the *command line editing* keys. The up-arrow key allows you to recall earlier commands, including commands from

previous sessions, and the other arrow keys allow you to revise these commands. Use the up-arrow key, followed by the **Enter** or **Return** key, to iterate, or repeatedly execute, this statement:

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

Here is what you get when you start with `x = 3`.

```
x =
     3
x =
     2
x =
    1.7321
x =
    1.6529
x =
    1.6288
x =
    1.6213
x =
    1.6191
x =
    1.6184
x =
    1.6181
x =
    1.6181
x =
    1.6180
x =
    1.6180
```

These values are 3, $\sqrt{1+3}$, $\sqrt{1+\sqrt{1+3}}$, $\sqrt{1+\sqrt{1+\sqrt{1+3}}}$, and so on. After 10 steps, the value printed remains constant at `1.6180`. Try several other starting values. Try it on a calculator if you have one. You should find that no matter where you start, you will always reach `1.6180` in about 10 steps. (Maybe a few more will be required if you have a very large starting value.)

MATLAB is doing these computations to accuracy of about 16 decimal digits, but is displaying only 5. You can see more digits by first entering

```
format long
```

and repeating the experiment. Here are the beginning and end of 30 steps starting at $x = 3$.

```
x =
     3
```

```
x =
     2
x =
   1.732050807568877
x =
   1.652891650281070
    ....
x =
   1.618033988749897
x =
   1.618033988749895
x =
   1.618033988749895
```

After about 30 or so steps, the value that is printed doesn't change any more. You have computed one of the most famous numbers in mathematics, ϕ, the *Golden Ratio.*

In MATLAB, and most other programming languages, the equals sign is the assignment operator. It says compute the value on the right and store it in the variable on the left. So, the statement

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

takes the current value of `x`, computes `sqrt(1+x)`, and stores the result back in `x`.

In mathematics, the equals sign has a different meaning.

$$x = \sqrt{1+x}$$

is an *equation.* A solution to such an equation is known as a *fixed point.* (Be careful not to confuse the mathematical usage of *fixed point* with the computer arithmetic usage of *fixed point.*)

The function $f(x) = \sqrt{1+x}$ has exactly one fixed point. The best way to find the value of the fixed point is to avoid computers all together and solve the equation using the quadratic formula. Take a look at the hand calculation shown in Figure 1.1. The positive root of the quadratic equation is the Golden Ratio:

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

You can have MATLAB compute ϕ directly using the statement

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

With `format long`, this produces the same value we obtained with the fixed point iteration,

```
phi =
   1.618033988749895
```

$$X = \sqrt{1+X}$$
$$X^2 = 1 + X$$
$$X^2 - X - 1 = 0$$
$$X = \frac{1 \pm \sqrt{1+4}}{2}$$
$$\varphi = \frac{1+\sqrt{5}}{2}$$

Figure 1.1. *Compute the fixed point by hand.*

Figure 1.2 is our first example of MATLAB graphics. It shows the intersection of the graphs of $y = x$ and $y = \sqrt{1+x}$.

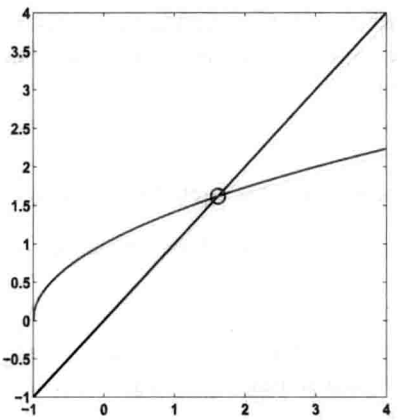


Figure 1.2. *A fixed point at $\phi = 1.6180$.*

The statement

```
x = -1:.02:4;
```

generates a vector x containing the numbers from -1 to 4 in steps of .02. The statements

```
y1 = x;
y2 = sqrt(1+x);
plot(x,y1,'-',x,y2,'-',phi,phi,'o')
```

produce a figure that has three components. The first two components are graphs of x and $\sqrt{1+x}$. The `'-'` argument tells the `plot` function to draw solid lines. The last component in the plot is a single point with both coordinates equal to ϕ. The `'o'` tells the `plot` function to draw a circle.

The MATLAB `plot` function has many variations, including specifying other colors and line types. You can see some of the possibilities with

```
help plot
```

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 1.3.

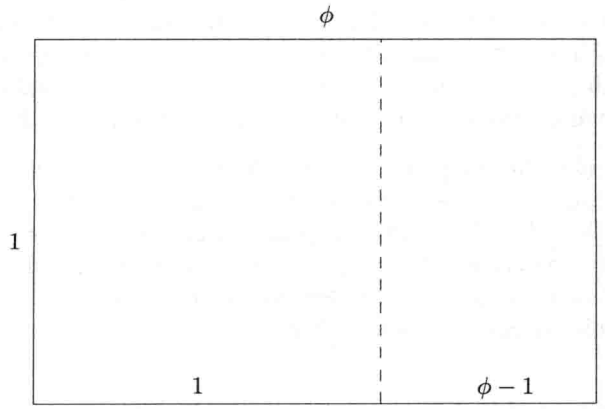


Figure 1.3. *The golden rectangle.*

The golden rectangle has the property that removing a square leaves a smaller rectangle with the same shape. Equating the aspect ratios of the rectangles gives a defining equation for ϕ:

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

Multiplying both sides of this equation by ϕ produces the same quadratic polynomial equation that we obtained from our fixed point iteration:

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

The up-arrow key is a convenient way to repeatedly execute a single statement, or several statements (separated by commas or semicolons) on a single line. Two more powerful constructs are the `for` loop and the `while` loop. A `for` loop executes a block of code a prescribed number of times.

```
x = 3
for k = 1:31
   x = sqrt(1 + x)
end
```

produces 32 lines of output, 1 from the initial statement and 1 more each time through the loop.

A `while` loop executes a block of code an unknown number of times. Termination is controlled by a logical expression, which evaluates to `true` or `false`. Here is the

simplest `while` loop for our fixed point iteration.

```
x = 3
while x ~= sqrt(1+x)
   x = sqrt(1+x)
end
```

This produces the same 32 lines of output as the `for` loop. However, this code is open to criticism for two reasons. The first possible criticism involves the termination condition. The expression `x ~= sqrt(1+x)` is the MATLAB way of writing $x \neq \sqrt{1+x}$. With exact arithmetic, `x` would never be exactly equal to `sqrt(1+x)`, the condition would always be true, and the loop would run forever.

Like most technical computing environments, MATLAB does not do arithmetic exactly. To economize on both computer time and computer memory, MATLAB uses *floating point* arithmetic. Eventually our program produces a value of `x` for which the floating point numbers `x` and `sqrt(1+x)` are exactly equal and the loop terminates. Expecting exact equality of two floating point numbers is a delicate matter. It works OK in this particular situation, but may not work with more complicated computations.

The second possible criticism of our simple `while` loop is that it is inefficient. It evaluates `sqrt(1+x)` twice each time through the loop. Here is a more complicated version of the `while` loop that avoids both criticisms.

```
x = 3
y = 0;
while abs(x-y) > eps(x)
   y = x;
   x = sqrt(1+x)
end
```

The semicolons at the ends of the assignment statements involving `y` indicate that no printed output should result. The quantity `eps(x)` is the spacing of the floating point numbers near `x`. Mathematically, the Greek letter ϵ, or *epsilon*, often represents a "small" quantity. This version of the loop requires only one square root calculation per iteration, but that is overshadowed by the added complexity of the code. Both `while` loops require about the same execution time. In this situation, I prefer the first `while` loop because it is easier to read and understand.

Help and Doc

MATLAB has extensive on-line documentation. Statements like

```
help sqrt
help for
```

provide brief descriptions of commands and functions. Statements like

```
doc sqrt
doc for
```

provide more extensive documentation in a separate window.

One obscure, but very important, `help` entry is about the various punctuation marks and special characters used by MATLAB. Take a look now at

```
help punct
doc punct
```

You will probably want to return to this information as you learn more about MATLAB.

Numbers

Numbers are formed from the digits `0` through `9`, an optional decimal point, a leading `+` or `-` sign, an optional `e` followed by an integer for a power of 10 scaling, and an optional `i` or `j` for the imaginary part of a complex number. MATLAB also knows the value of π. Here are some examples of numbers.

```
42
9.6397238
6.0221415e23
-3+4i
pi
```

Assignment Statements and Names

A simple assignment statement consists of a name, an equal sign (=), and a number. The names of variables, functions and commands are formed by a letter, followed by any number of uppercase and lowercase letters, digits and underscores. Single character names, like `x` and `N`, and anglicized Greek letters, like `pi` and `phi`, are often used to reflect underlying mathematical notation. Nonmathematical programs usually employ long variable names. Underscores and a convention known as camel casing are used to create variable names out of several words.

```
x = 42
phi = (1+sqrt(5))/2
Avogadros_constant = 6.0221415e23
camelCaseComplexNumber = -3+4i
```

Expressions

Power is denoted by ^ and has precedence over all other arithmetic operations. Multiplication and division are denoted by *, /, and \ and have precedence over addition and subtraction, Addition and subtraction are denoted by + and - and have lowest precedence. Operations with equal precedence are evaluated left to right. Parentheses delineate subexpressions that are evaluated first. Blanks help readability, but have no effect on precedence.

All of the following expressions have the same value. If you don't already recognize this value, you can use Google to check its importance in popular culture.

```
3*4 + 5*6
3 * 4+5 * 6
2*(3 + 4)*3
-2^4 + 10*29/5
3\126
52-8-2
```

Recap

```
%% Iteration Chapter Recap
% This is an executable program that illustrates the statements
% introduced in the Iteration chapter of "Experiments in MATLAB".
% You can run it by entering the command
%
%    iteration_recap
%
% Better yet, enter
%
%    edit iteration_recap
%
% and run the program cell-by-cell by simultaneously
% pressing the Ctrl-Shift-Enter keys.
%
% Enter
%
%   publish iteration_recap
%
% to see a formatted report.

%   Copyright 2012 Cleve Moler
%   Copyright 2012 The MathWorks, Inc.
%% Help and Documentation
%   help punct
```

```
%   doc punct

%% Format
    format short
    100/81
    format long
    100/81

    format short
    format compact

%% Names and assignment statements
    x = 42
    phi = (1+sqrt(5))/2
    Avogadros_constant = 6.0221415e23
    camelCaseComplexNumber = -3+4i

%% Expressions
   3*4 + 5*6
   3 * 4+5 * 6
   2*(3 + 4)*3
   -2^4 + 10*29/5
   3\126
   52-8-2

%% Iteration
% Use the up-arrow key to repeatedly execute
   x = sqrt(1+x)
   x = sqrt(1+x)
   x = sqrt(1+x)
   x = sqrt(1+x)

%% For loop
   x = 42
   for k = 1:12
      x = sqrt(1+x);
      disp(x)
   end

%% While loop
   x = 42;
   k = 1;
   while abs(x-sqrt(1+x)) > 5e-5
      x = sqrt(1+x);
      k = k+1;
   end
```