An Introduction to Matlab and Mathcad

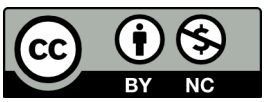
Spring 2014 Edition

Troy Siemers, Ph.D.

Department of Mathematics and Computer Science

Virginia Military Institute

www.dbooks.org



Copyright © 2014 Troy Siemers
Licensed to the public under Creative Commons Attribution-Noncommercial 3.0 United States License

Thanks

Many thanks to Greg Hartman for his huge amount of help with the LaTeX layout.

Thanks to students for finding errors in the drafts leading up to this text, including Chris Fraser, Stephen States, Heather Chichura and Joey Bishop.

The LaTeX community has provided a large amount of online (and free!) assistance, including MiKTeXand TeXnicCenter. These are flexible and powerful tools that allowed me to produce a text with just the look I wanted. Many thanks.

www.dbooks.org

Preface

A Note to Students, Teachers, and other Readers

This text is used in a mathematical software course at VMI that provides an introduction to Matlab and Mathcad. However, it is also intended to be a *course* book instead of an all inclusive resource. I encourage my students to take full advantage of the built-in help capabilities of these software packages, additional texts (I keep a few in a small library that is always available to students) and to use Google - I certainly did when learning the material and still do today.

This text is much shorter than a traditional text. I have tried to provide the right amount of discussion and examples. The exercise sets have a small number of problems, but I assign all of them when I teach the course. I have tried to give problems that come from real-world situations and contemporary data and aren't simply contrived just to fill space. Some of the problems reference the electronic course management system at VMI (Angel), but these can be ignored as the data sets are also included in the Appendix.

This text is also an "open" text. If you wish to change the text for your needs, please do so. I would also share source files if you are interested. The Creative Commons copyright must be honored and that any changes be acknowledged and that the resulting work be used only in non-commercial areas.

Note that throughout the text, I make reference to colors (regarding text or pictures). This book is printed in black and white on purpose, but you can contact me for the full color pdf version (www.vmi.edu/math > faculty).

I welcome any comments or corrections.

Sincerely,

Troy Siemers
troy@vmi.edu

www.dbooks.org

Contents

Thanks **iii**

Preface **v**

Table of Contents **vii**

1 Matlab: Introduction **1**
1.1 Matlab: Introduction 1
1.2 Matlab: Layout 1
1.3 Matlab: Command Window Examples 3
1.4 Matlab: Editor 4
1.5 Matlab: Headers 5
1.6 Matlab: Editor Tips 6
Exercises 8

2 Matlab: Matrices **9**
2.1 Matlab: Matrices 9
2.2 Matlab: Using Matrices 12
2.3 Matlab: Matrix Operations 14
2.4 Matlab: Common Matrix Functions 15
2.5 Matlab: Systems of Equations 19
Exercises 21

3 Matlab: Functions **23**
3.1 Matlab: Built-In Functions 23
Exercises 29

4 Matlab: Graphics **30**
4.1 Matlab: Graphics 30
Exercises 35

5 Matlab: User Defined Functions **37**
5.1 Matlab: User-Defined Functions 37
Exercises 41

www.dbooks.org

6 Matlab: Input/Output **43**
6.1 Matlab: Input Commands 43
Exercises 50

7 Matlab: Programming Structures **51**
7.1 Matlab: Relational Operators 51
7.2 Matlab: Logical Operators 51
7.3 Matlab: `if` and `switch` commands 53
7.4 Matlab: `for` and `while` Loops 54
Exercises 57

8 Matlab: Applications **58**
8.1 Matlab: Numerical Methods 58
8.2 Matlab Traveling Salesman 60

9 Matlab: Curve Fitting **63**
9.1 Matlab: Curve Fitting 63
Exercises 68

10 Mathcad: Introduction **70**
10.1 Mathcad: Introduction 70

11 Mathcad: Entering Equations **72**
11.1 Mathcad: Equations 72
11.2 Mathcad: Editing Equations 77
11.3 Mathcad: Units 79
Exercises 81

12 Mathcad: Given/Find and Solve **82**
12.1 Mathcad: Given/Find Blocks 82
12.2 Mathcad: Solve Blocks 84
Exercises 86

13 Mathcad: Functions **87**
13.1 Mathcad: Built-in Functions 87
13.2 Mathcad: User-Defined Functions 88
Exercises 90

14 Mathcad: Matrices **91**
14.1 Mathcad: Matrix Definition 91
14.2 Mathcad: Editing Matrices 93
14.3 Mathcad: Referencing Parts of Matrices 95
14.4 Mathcad: Solving Systems of Linear Equations 97
Exercises 101

15 Mathcad: Graphing **103**
15.1 Mathcad: Graphing 103
Exercises 110

16 Mathcad: Curve Fitting **111**
16.1 Mathcad: Curve Fitting 111
Exercises 116

17 Mathcad: Calculus and Symbolics **118**
17.1 Mathcad: Calculus 118
17.2 Mathcad: Symbolics 119
Exercises 121

Appendix **123**

Index **124**

www.dbooks.org

Chapter 1

Matlab: Introduction

In this section, we discuss the basics of Matlab.

1.1 Matlab: Introduction

There are many different software packages available. It's important to understand the different capabilities of each as one software package may be the best tool depending on the task. Here, we introduce Matlab, which stands for MATrix LABoratory. Matlab is cost-effective, easy to learn and blends powerful number crunching capabilities with graphical display. It is an excellent tool for working with large data sets.

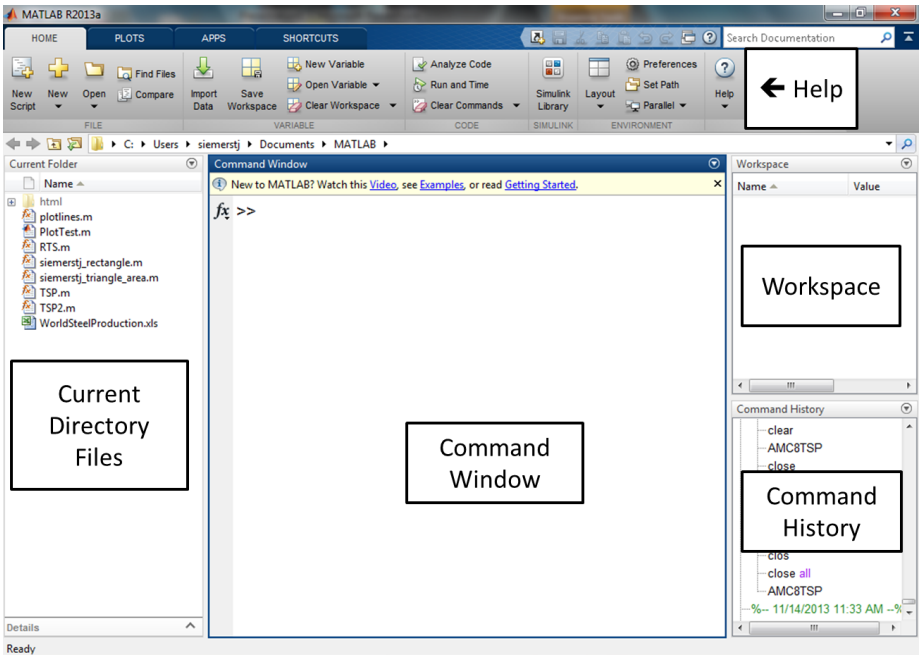
Other packages that you may wish to consider, include Mathematica, Maple, Fortran, C, C++, or Java. In some cases, even a basic TI calculator or Excel is the appropriate tool.

Matlab has become the industry standard for use in many fields including Mathematics, Bioinformatics, Finance, and Engineering. The basic Matlab package includes the core package and Simulink, a platform for modeling dynamic systems. Matlab can also be enhanced through the addition of "toolboxes" (available from Mathworks) including such topics as Control Systems, Image Processing, Splines and SimBiology.

1.2 Matlab: Layout

We now take a look at the (default) layout of Matlab:

www.dbooks.org



You can customize how the layout appears, but there are a few main components:

Command Window

The command window is where you can perform basic calculations, enter commands, run programs, and view the numeric output.

Command History

The command history keeps a record of past commands that were entered in the command window. To run a command again from the command history, you can simply double click it. Or, if you want to alter the command before running it, you can click and drag it to the command window, change it and then run it (by hitting Enter).

Directory and Workspace (in tabs)

The directory shows the files in the current directory (which itself is listed at the top of the screen). These files can be run by double clicking, or by clicking and dragging them into the command window. The workspace keeps track of the variables that are created. Double clicking a variable in the workspace opens a spreadsheet where the variable can be altered.

Current Directory

When you run any program, it is important that your Current Directory is set to the location of your program. You can also run programs in a different directory by setting the "path" to that directory. To keep it simple, we will not explain how to do this here, but refer you to the help files.

Help

The help capabilities in Matlab are well documented. It does take some time to understand Matlab syntax, but if a user is familar with another programming language, the commands are easy to pick up. Functions in Matlab are also well named, so you can often guess the name of a function that you may need.

1.3 Matlab: Command Window Examples

Let's try some basic examples in the command window. Enter 5+5 in the command window (and press Enter)

```
>> 5+5
ans =
     10
```

The value 10 has been assigned to the variable `ans`. Next, try the following

```
>>a=5+5
a=
     10

>>b=5+5; % the semicolon suppresses output.
```

A few things to note. First, assignment of values to the variables is right to left (whatever is on the right of the equals sign is assigned to the variable on the left). Here, both both `a` and `b` are equal to 10, but only `a`'s value is displayed because the semicolon is used to suppress the output to the screen. This is a very useful tool in programs where, for example, you want to hide the intermediate calculations.

While there are thousands of commands in Matlab, here are a few that you will use often:

`clc` clears the command window
`clear all` clears all the variables
`clear variablename` clears the variable named `variablename`
`close all` closes all open plot windows
`ctrl + c` stops a running process (important later!)

www.dbooks.org

There are several rules about the naming of variables (look those up in Help or Google), but in particular, all variable names must start with a letter and all variables are case sensitive (upper case and lower case letters are considered different)!! So, for example, the variables `Math` and `math` are treated as different variables.

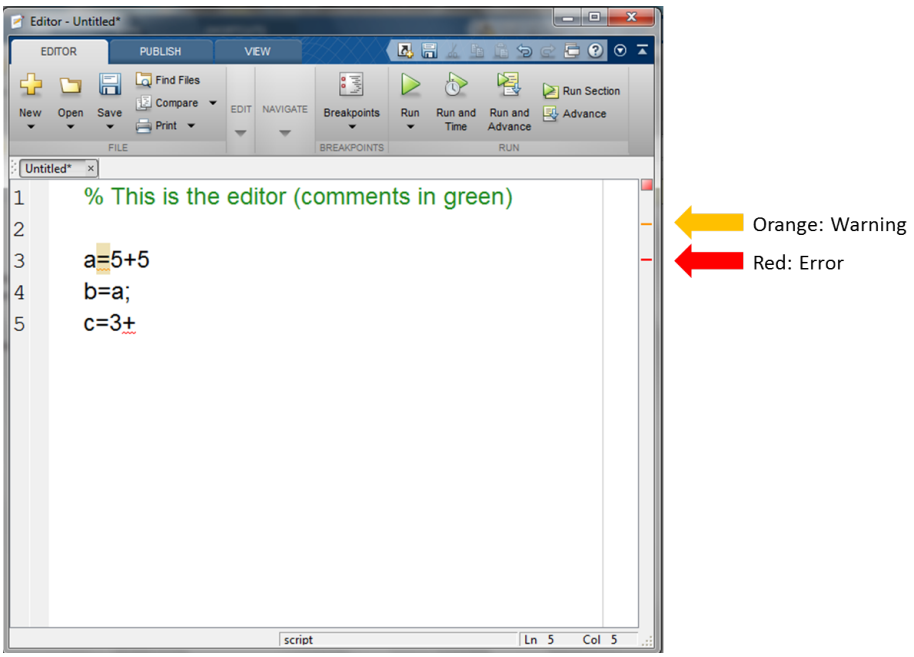
One final "peculiar" aspect of Matlab, is that you can only edit the line you are on!! You can however recall previous commands with the up and down arrow keys, or type text and use the up and down arrow keys to scroll through the commands starting with that text.

1.4 Matlab: Editor

It is tedious to type everything into the command window, not to mention trying to fix any errors. We now look at the **Editor**, where you will write your Matlab **script files** (basic programs) and **function files** (user-defined functions).

As with many aspects of Matlab, there are usually many ways to perform a task. To start the Editor window, you can either type `edit` in the command window or you can click on the "new script" button .

The Editor is shown below. In the blue bar at the top of this image, you can see that the file is currently Untitled. The asterisk indicates that the file has not been saved since the last change to the file. Try entering the text exactly as shown.



There are few more key parts to this file. Any text that follows a percent sign (`%`) is a **comment** and is colored green. These are ignored when the program is actually run, but are an important part of the documentation of the file. It is crucial that you document your files properly both to remind yourself what parts of the program do and to inform any other users that may work with the program in the future.

As you type lines into the program (hitting Enter at the end of the lines), either **orange or red lines** may appear on the right. You can hover your cursor over these lines to see related messages. In general, orange lines are suggestions, for example, that you should add semicolons on the end of lines to suppress output, or perhaps that another command would be better. Red lines indicate errors. You should pay close attention to these. The messages related to red lines may indicate missing parentheses, a missing "end" to close a loop, or something worse, like improper syntax. In this particular file, we are missing a semicolon in the definition of the variable `a` (the orange line) and haven't completed the line for the definition of `c` (the red line).

When you save this program, the file name will have a ".m" extension. We refer you to the help files for proper naming conventions of files (and variables in general).

1.5 Matlab: Headers

For each assignment for this course, you will be submitting either a script or function M-file. At the top of each file, you must include a descriptive header (as comments). They must look like the following, adjusted to fit your name, the date, etc.

```
% Troy Siemers
% Program Name:  Assignment1.m
% Date:  10 January 2010
% Course:  MA110
% Description:  In this assignment, we focus on how
% matrices are entered and referenced in Matlab.  We also
% use component-wise multiplication and
% matrix exponentiation.
```

We can not stress enough the importance of proper documentation. You may be working with other people on coding or may return to a file that you wrote several days ago (or weeks, years, etc.). Without the comments to explain your work (to others or as reminder to yourself), it is often difficult to understand the code (and fix it when it doesn't function properly).

www.dbooks.org

1.6 Matlab: Editor Tips

Here we include some tips and keyboard shortcuts that come up often and can save time. Note that many of these are accessible by using the right mouse button while in the Editor.

Block Commenting

Instead of deleting code, it is often adventageous to simply "comment it out" for later editing. To do this, simply highlight the code that you wish to comment out and use `Ctrl + R`. Note that if you only want to comment out a single line, just make sure your cursor is on the line and use `Ctrl + R` (you don't have to highlight the whole line). You can uncomment any of these later by highlighting any comment lines and using `Ctrl + T`.

Running Code

There are several ways to execute your files, but there are a few short-cuts when running script files. To run the entire file, you can either use the `F5` key or click on the green "run" button . To see how smaller blocks of code may run, you can highlight sections of the code and use the `F9` button.

Indenting

Inside of several structures, like loops, it is important that you indent the code properly. This not only for correct syntax, but also makes for easier reading. To make sure code is indented in the right way, simply highlight the code and type `Ctrl + I`.

Example 1

For this example, you could run these commands one at a time at the >> in the Command Window, but we will create a script M-file in the Editor, which we open by typing `edit` and hitting return or you can click on the "new script" button . Enter the following commands on separate lines:

```
SideLength = sqrt(5)
Value = cos(pi)
DegValue = cosd(180)
radius = 5;
Area = pi * radius ^ 2
```

Now save the file as `test.m`. To run the file you have the options listed in

this chapter, but another easy way is to go to the Command Window and type `test` at the >> line. The output for this file should look like:

```
SideLength =
     2.2361
Value =
     -1
DegValue =
     -1
Area =
     78.5398
```

Note that there was no output of the value for `radius` to the Command Window (the semicolon suppresses the output), but the value was stored in the variable named `radius` (and can be found in the Workspace window - look!).

www.dbooks.org

Chapter 1 Exercises

1. Which of the following are valid and useable variable names? Explain your answers

   a. Homework 1

   b. 1Homework

   c. Homework#1

   d. Homework_1

   e. HoMeWoRkNuMbEr1

2. Compute the following using the correct order of operations

   a. $2 - 4 * (5^{3-2} + 2 * (5 + 6))$

   b. $2 - \dfrac{4 * 5^{3-2}}{2 * (5 + 6)}$

   c. $2 - \dfrac{4 * 5^{3-2}}{\frac{2}{5} + 6}$

3. The volume of a truncated pyramid with a square base is given by

$$V = \frac{1}{3}(a^2 + ab + b^2)h$$

   where h is the height, a is the length of one of the sides of the base and b is the length of the sides of one of the top (also a square). Find the volume if $a = 5, b = 3, h = 10$ by first defining a, b and h as separate variables and then defining V in terms of them.

4. The escape velocity from a planet is given by $v = \sqrt{\dfrac{2GM}{R}}$ where G is the gravitational constant, M is the mass and R is the radius of the planet. Compute the escape velocity of both earth and the moon using an online source to find the constants. In each case, first define G, M and R and then define v in terms of them. Make sure to give the citation of your source.

5. Stirling's formula for computing the factorial is given by $n! \approx \sqrt{2\pi n}\left(\dfrac{n}{e}\right)^n$. For each of $n = 1, 2, 3, \cdots, 20$, compute both sides of this approximation. Hint: Use the `factorial` command in Matlab.

6. In considering Stirling's formula, since the numbers grow so quickly in using the factorial, it can be advantageous to use the natural logarithm function. A related approximation is given by $\ln(n!) \approx n \ln n - n$. Compute the difference $\ln(n!) - (n \ln n - n)$ for $n = 10, 20, 30, 40, 50$. What is the largest value of n that Matlab will allow in this difference calculation?

Chapter 2

Matlab: Matrices

In this section, we discuss how matrices are created, referenced and used in calculations.

2.1 Matlab: Matrices

Each variable in Matlab is stored as a **matrix**, which is an array of numbers arranged in a rectangle of m rows and n columns. One says that such a matrix is an m by n matrix, written as m $\times$ n. A **vector** is any matrix that has either only one row (a "row vector") or one column (a "column vector"). A scalar, or number, is stored as a matrix that has exactly one row and one column (i.e, a 1×1 matrix).

Let's look at some examples of how matrices are entered in Matlab. Each matrix is enclosed in the symbols `[` and `]`, each comma (or space) separates entries on the same row and each semicolon indicates a new row.

Example 2

```
>> A = [1,2,3,4;5,6,7,8] (or A = [1 2 3 4;5 6 7 8])
```

gives the 2×4 matrix

```
A =
     1 2 3 4
     5 6 7 8
```

and

```
>> B = [1; 6; 0; 9]
```

gives the 4×1 matrix

www.dbooks.org

```
B =
     1
     6
     0
     9
```

and

```
>> C=[3]
```

gives the 1×1 matrix (i.e. a scalar)

```
C =
     3
```

You may have noticed that the semicolon is used in a new way in the last example. It is perhaps unfortunate, but there are symbols that are reused throughout Matlab code (including the semicolon, colon and comma) and the meaning of a particular symbol will depend on context. Semicolons are used *inside* matrix definitions to indicate a new row while semicolons are used at the end of a line to suppress output. Let's look at a few more examples.

Example 3

Suppose we wanted a matrix with the values 1 through 19 in a single (row) vector. We could enter this as

```
>> M = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]
```

although this takes a little while to type (what if we wanted 1 to 1000?!). Instead we use the **colon operator**. The syntax to create this same matrix of 1 to 19 is shortened to

```
>> M = [1:19].
```

In the last example, we can think of the colon as a "range operator." Note that Matlab also accepts the slightly shorter `M = 1:19` as well, which will be useful when we come to loops. In fact, the colon is more flexible (pun intended?) as we see in the next example.

Example 4

The matrix defined as

```
>> N = [1:2:19]
```

is the same as if we typed

```
>> N = [1 3 5 7 9 11 13 15 17 19].
```

What is going on in this example? Well, the general form in using two colons this way is

```
start value : step size : end value
```

In `N`, the 2 means to simply count "by twos" starting at 1 and ending at 19. There's a slight limitation in that if we tried `P = [1:2:20]` we would also get `P` to equal `[1 3 5 7 9 11 13 15 17 19]`. Where's the 20? Well, the syntax of `start:step:end` actually means to begin at `start`, add the `step` one at a time and then stop at the point where adding one more step size would take us past the `end` value. So, in the matrix `P`, we have to stop at 19 since adding two more would take us to 21 (past the `end` value of 20). But, suppose we actually wanted to end exactly at your last value? That's where we use the command `linspace` instead.

Example 5

```
>> P = linspace(1,6,4)
```

Here we get

```
P =
     1.0000 2.6667 4.3333 6.0000
```

You can see that we started at 1 and ended at 6 and have 4 total values. That's the exact syntax:

```
linspace(start,end,number of values)
```

There are two special matrices that come up a lot too:

Example 6

```
>> G = ones(3,4)
```

gives the 3×4 matrix of all ones:

```
G =
     1 1 1 1
     1 1 1 1
     1 1 1 1
```


www.dbooks.org

```
>> H = zeros(2,5)
```

gives the 2×5 matrix of all zeros:

```
H =
     0  0  0  0  0
     0  0  0  0  0
```

2.2 Matlab: Using Matrices

Let's look at how we can reference parts of a matrix.

Example 7

Consider the matrix `A = [1 2 3 4;5 6 7 8]`. There are actually two ways to view this matrix, either as a rectangular array of 2 rows and 4 columns, or as a list of 8 elements. Suppose we wanted to isolate the 7 in the matrix `A` and store it as the variable `temp`. First, we can think of the 7 as being located in the second row and third column. In this case, we can type:

```
>> A = [1 2 3 4;5 6 7 8];
>> temp = A(2,3)
```

with the result being:

```
temp =
     7
```

Second, we can think of 7 as being one of the eight elements total. But, it is crucial to realize that we count elements in this way using a "column precedence." This means that we count, one at a time, down the columns. This means that we can think of 7 as being located in the 6th entry, or:

```
>> temp = A(6)
```

also gives the result:

```
temp =
     7
```

For completeness, in the last example, if we think of `A` as a matrix:

```
A(1,1) = 1   A(1,2) = 2   A(1,3) = 3   A(1,4) = 4
A(2,1) = 5   A(2,2) = 6   A(2,3) = 7   A(2,4) = 8
```

or, thinking of `A` as a vector:

```
A(1) = 1   A(3) = 2   A(5) = 3   A(7) = 4
A(2) = 5   A(4) = 6   A(6) = 7   A(8) = 8
```

If we wanted to store the entire first row of `A` in the variable `firstrow`, we would say that we want "all four columns of the first row." This suggests that we can use the colon operator to shorten our work. Namely,

```
>> firstrow = A(1,1:4)
```

which gives

```
firstrow =
     1 2 3 4
```

But, there's an even shorter way to do this! If the colon doesn't have a start and end value, it simply lists all possible values! Namely,

```
>> firstrow = A(1,:)
```

also gives

```
firstrow =
     1 2 3 4
```

Ok, now what if we wanted the first row, but not the element in the first column? There are two ways to do this. First, we can use the colon as:

```
>> mostoffirstrow = A(1,2:4)
```

which gives

```
mostoffirstrow =
     2 3 4
```

But, what if the matrix changes and we don't know how big `A` has changed to? Those sneaky programmers at Mathworks have a work around:

```
>> mostoffirstrow = A(1,2:end)
```

also gives

```
mostoffirstrow =
     2 3 4
```


www.dbooks.org

2.3 Matlab: Matrix Operations

Here we will explore the algebra of matrices. It would be wise for the reader to have a basic knowledge of matrix algebra (or even linear algebra), but we will try and give plenty of explanatory examples.

Just as for scalars, many of the common algebraic operations apply to matrices. The symbols + and − carry over quite nicely in "element-by-element" operations as one would expect (or at least hope for). Similarly, if you want element-by-element multiplication, division, or even exponentiation, these are given by .∗, ./ and .∧ (yes, those each have a preceeding period and are pronounced **"dot times"**, **"dot divide"** and **"dot exponent"**). Operations like "regular" multiplication, division, exponents, etc. have a very different meaning than one who hasn't been exposed to linear algebra might expect. We also have operations like the matrix transpose.

Let's look at some examples using the matrices `A = [1 2 3 4;5 6 7 8]` and `B = [1; 6; 0; 9]`. Again,

```
A =
     1 2 3 4
     5 6 7 8
and
B =
     1
     6
     0
     9
```

Example 8 Scalar Multiplication

Here we show how to multiply every entry in a matrix by a scalar:

```
>> A = [1 2 3 4;5 6 7 8];3*A
```

gives

```
ans =
      3  6  9 12
     15 18 21 24
```

One special operation for matrices is the matrix **transpose**, given by either `transpose(A)` or the shorter `A'`. The transpose of a matrix is another matrix with the rows and columns interchanged.

Example 9 Transpose

```
>> A = [1 2 3 4;5 6 7 8];A'
```

```
ans =
     1   5
     2   6
     3   7
     4   8
```

and

```
>> B = [1; 6; 0; 9];B'

ans =
     1 6 0 9
```

The example A∧2 gives an error saying A must be square (here it is good to know some linear algebra), but A.∧2 gives element-by-element squaring:

Example 10 Element by element squaring

```
>> A = [1 2 3 4;5 6 7 8];A.∧2

ans =
      1  4  9 16
     25 36 49 64
```

Finally, we can create "block" matrices from smaller matrices by treating them as elements themselves and using the comma (or space) and the semicolon to create rows and columns. We just have to make sure the dimensions of the matrices line up properly. For example, we can stack two matrices.

Example 11 Stacking Matrices

```
>> A = [1 2 3 4;5 6 7 8];B = [1; 6; 0; 9];AoverBprime = [A ; B']

AoverBprime =
     1 2 3 4
     5 6 7 8
     1 6 0 9
```

2.4 Matlab: Common Matrix Functions

`size` and `length`

Many functions involving matrices will only work if the dimensions of the matrices satisfy certain conditions. The `size(M)` command returns the num-

www.dbooks.org

ber of rows and columns in `M`.

Example 12 size of a matrix

```
>> A = [1 2 3 4;5 6 7 8];size(A)

ans =
     2 4
```

and

```
>> B = [1; 6; 0; 9];size(B)

ans =
     4 1
```

The `size` command is also what is known as an "overloaded" function in that it can be used in a couple of ways. Suppose we only wanted to know only the number of rows in a matrix `M`. We could find the `size(M)`, store this as a variable and then select the first entry. Instead, the `size` command takes a second entry that will allow us to get what we want.

Example 13 Number of Rows or Columns Using `size`

```
>> A = [1 2 3 4;5 6 7 8];size(A,1)

ans =
     2
```

(the number of rows)

and

```
>> A = [1 2 3 4;5 6 7 8];size(A,2)

ans =
     4
```

(the number of columns).

The length of a vector (either a row or column) is simply the number of elements in the vector:

Example 14 Length of a vector

```
>> c=1:5;
```

```
>> length(c)

ans =
     5
```

However, the "length of a matrix" is defined in Matlab as the larger of the number of rows and the number of columns. That is, `length(A)` is equivalent to `max(size(A))`.

`max` and `min`

These two functions are (almost) self explanatory. For the matrix that we are using, `A = [1 2 3 4;5 6 7 8]`, if we try `max(A)`, we get `5 6 7 8`. What's going on? Remember that Matlab works on **column precedence** so that what `max` is doing is not finding the maximum value of the entire matrix, but instead finding the maximums of each column. The only exception occurs when the starting matrix is either a row or column vector. For example, for `B = [1; 6; 0; 9]`, `max(B)` does give us `9` (the largest value in `B`). So, to get the largest element in `A`, we would have to "nest" the functions as `max(max(A))`, which would give us 8.

www.dbooks.org

sum and prod

Once again, these seem reasonably named functions (see previous bullet). And once again, they return not the sum\product of every entry in matrix, but the column sums\products with the exception being for vectors, in which case you do get the sum\product of every entry in the vector. But, what if you want to get the sum along the rows? Well, once again `sum` is an overloaded operator and we can use:

Example 15 Column and Row sums

```
>> A = [1 2 3 4;5 6 7 8]
>> sum(A)

ans =
     6 8 10 12
```

(the column sums)

and

```
>> sum(A,2)

ans =
     10
     26
```

(the row sums)

For those that know more linear algebra, we list some familar commands.

cross and dot

These are the functions to find the cross product or dot product of two vectors using `dot(v1,v2)` and `cross(v1,v2)`. A few things to note. First, the vectors both have to be the same length, but it doesn't matter if they are both row vectors, both column vectors, or even one of each. Second, for the cross product, recall that you need them both to be of length 3 (i.e. each of dimension 1×3 or 3×1).

det and inv

For a square matrix `S`, you can find the determinant and inverse using the commands `det(S)` and `inv(S)`. You can also use the command `S^(-1)` although a common mistake is to forget the parentheses around the -1.

eye

The folks at Mathworks do have an interesting sense of humor. To create the 5×5 identity matrix, for example, you could type all 25 entries of ones and zeros, or you can use the command `eye(5)`. Get it? "eye"? Get it? Nevermind.

2.5 Matlab: Systems of Equations

We can use matrices to solve systems of linear equations. Here it is a good idea to read up a bit on some matrix algebra.

Suppose we have the following system of equations:

$$\left\{\begin{array}{rcl} 3x+2y-z & = & 10 \\ -x+3y+2z & = & 5 \\ x-y-z & = & -1 \end{array}\right\}$$

We will solve this system, i.e. find the values of the variables that satisfy all of the equations simultaneously, in three ways: using reduced row echelon form, using matrix inverses, and using "left division."

Method 1: Reduced Row Echelon Form

Here we create the "augmented matrix" of the coefficients of the variables with the constants to the right of the equals signs.

```
>> AugmentedMatrix = [3 2 -1 10;-1 3 2 5;1 -1 -1 -1];
>> rref(AugmentedMatrix)

ans =
     1  0  0 -2
     0  1  0  5
     0  0  1 -6
```

This tells us that there is only one way to solve this system, i.e. only one solution, namely `x = -2, y = 5, z = -6`. You can check that is correct by substituting these values back into the system of equations:

$$\left\{\begin{array}{rcl} 3(-2)+2(5)-(-6) & = & 10 \\ -(-2)+3(5)+2(-6) & = & 5 \\ (-2)-(5)-(-6) & = & -1 \end{array}\right\}$$

and verifying that they are all correct.

www.dbooks.org

Method 2: Using the matrix inverse

Here we create two matrices, one for the coefficients of the variables and one for the constants to the right of the equals signs. Note that we can define these on the same line to save space:

```
>> Coeffs = [3 2 -1;-1 3 2;1 -1 -1]; Constants=[10; 5; -1];
```

Since the determinant of `Coeffs` is non-zero (check!) we can solve the system with the inverse:

```
>> inv(Coeffs)*Constants

ans =
     -2
      5
     -6
```

This also tells us that the only solution is `x = -2, y = 5, z = -6`.

Method 3: Using left division

The motivation for this method is complicated. We suggest that you read the Matlab documentation on left (and right) division of matrices. Again we create the two matrices, `Coeffs` and `Constants`

```
>> Coeffs = [3 2 -1;-1 3 2;1 -1 -1]; Constants=[10; 5; -1];
```

and use the backslash (be careful to use the correct slash):

```
>> Coeffs\Constants

ans =
     -2
      5
     -6
```

This also tells us that the only solution is `x = -2, y = 5, z = -6`.

Chapter 2 Exercises

1. Basic calculations

   a. Using colons, create the matrix that contains the numbers from 1 to 491 by steps of 10.

   b. Using `linspace`, create the matrix that contains the numbers from 1 to 491 by steps of 10.

   c. Using colons, create a 1×50 matrix containing evenly spaced numbers from 1 to 32 (inclusive).

   d. Using `linspace`, create a 1×50 matrix containing evenly spaced numbers from 1 to 32 (inclusive).

2. Consider the 5×5 "Hilbert matrix"

$$H_5 = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{bmatrix}$$

   a. Find the determinant of H_5.

   b. Find the transpose and inverse of H_5.

   c. Using the commands in the text, find the dimensions of H_5, the column sums, and the row sums of H_5.

   d. Use the `max` function to determine the value of the maximum entry of H_5.

   e. Find the eigenvalues and eigenvectors of H_5.

   f. Find the matrices $H_5^2, H_5.\wedge 2$ and $H_5./H_5$ and explain your answers.

   g. By referencing H_5, create a matrix that consists of the 2nd and 3rd rows of H_5 (do not simply retype the entries).

3. Counterclockwise rotation in two dimensions (about the origin) can be done through matrix multiplication (multiplying on the left) by the matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

   Rotate the point $(x, y) = (-2, 3)$ counterclockwise about the origin by each of $\theta = 30^\circ, 90^\circ$ and 200° and give the final positions in each case.

4. Counterclockwise rotation in two dimensions about an arbitrary point, P, can be done by a translation (moving P to the origin), rotation about the origin (through matrix multiplication as in the last problem), and then retranslation (of the origin back to P) .

   Rotate the point $(x, y) = (-2, 3)$ counterclockwise about the point $(1, 3)$ by each of $\theta = 30^\circ, 90^\circ$ and 200° and give the final positions in each case.

www.dbooks.org

5. Rotation in 3-dimensions is a more difficult process. For example, if you rotate the point $(1, 0, 0)$ counterclockwise around the z axis, you will arrive at the point $(0, 1, 0)$. Look up (online) for the matrix that corresponds how to rotate by an angle θ around the z axis in 3D and make sure you cite the online source. Implement this matrix to indicate the rotation of $(1, 0, 0)$ about the z axis by θ radians for $\theta = \pi/4, \pi/2$ and $\pi/8$.

6. In computing an approximating function for a set of data, a spline is often used. The theory of splines is covered in numerical analysis, but for a specific data set, the following system of equations must be solved.

$$\left\{\begin{array}{rcrcrcrcrcr} 0.28S_1 & + & 0.1S_2 & & & & & & & = & -64.65 \\ 0.1S_1 & + & 0.34S_2 & + & 0.07S_3 & & & & & = & -54.81 \\ & & 0.07S_2 & + & 2.16S_3 & + & 1.01S_4 & & & = & -8.43 \\ & & & & 1.01S_3 & + & 2.58S_4 & + & 0.28S_5 & = & -7.92 \\ & & & & & & 0.28S_4 & + & 1.42S_5 & = & -2.78 \end{array}\right\}$$

   Find the values for S_1 through S_5 by solving this system in three ways:

   a. Using the command `rref`.

   b. Using left division.

   c. Using the matrix inverse.

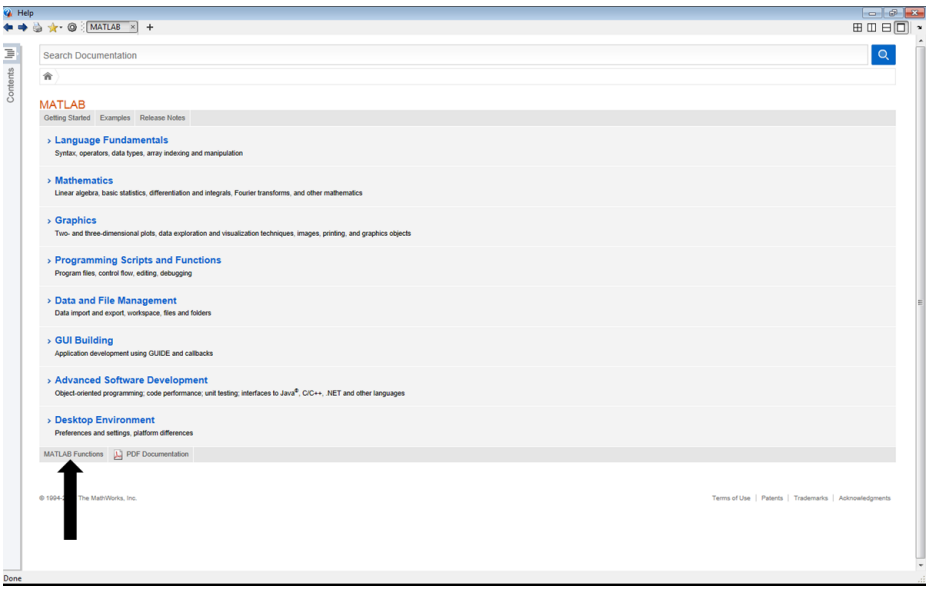
Chapter 3

Matlab: Functions

In this section, we look at some functions that are built-in to Matlab. In a later section, we discuss how a user may write their own.

3.1 Matlab: Built-In Functions

As with most parts of Matlab, the Help window is useful in describing the functions that are in Matlab, Simulink, and any toolbox add-ons that you may have. To access the entire list of functions, grouped in various ways, click on the Help button, the one shaped like a question mark at the top of the main Matlab window, or simply use the F1 button. Once on the Help window, you can click on "MATLAB Functions" at the bottom.



www.dbooks.org

In discovering what a function does, we suggest to simply try it. Let's start with the matrix `A = [1 2 49 4;25 36 3 81]` and look at the output from several functions. First type in

```
>> A = [1 2 49 4;25 36 3 81]
```

- `sqrt(A)`

The output:

```
>> sqrt(A)
ans =
     1.0000 1.4142 7.0000 2.0000
     5.0000 6.0000 1.7321 9.0000
```

is (hopefully) what you might expect. It finds the square root of each of the entries. If you know a little more linear algebra and was expecting the "principal" square root, or a matrix `B` so that `B * B = A`, this is created using `B = sqrtm(A)`.

- `sin(A), sind(A)`

The function `sin(A)` finds the sine of every entry in `A`, assuming the entries of `A` are in radians:

```
>> sin(A)

ans =
      0.8415 0.9093 -0.9538 -0.7568
     -0.1324 -0.9918 0.1411 -0.6299
```

and the function `sind(A)` does the same, but assumes the entries of `A` are in degrees:

```
>> sind(A)

ans =
     0.0175 0.0349 0.7547 0.0698
     0.4226 0.5878 0.0523 0.9877
```

The other trigonometric functions are similarly named.

- `exp(A), log(A), log10(A)`

These three functions are base e exponentiation, the natural log and the logarithm base 10. They act entry-wise:

```
>> exp(A)

ans =
1.0e+035 *
     0.0000 0.0000 0.0000 0.0000
     0.0000 0.0000 0.0000 1.5061

>> log(A)

ans =
           0 0.6931 3.8918 1.3863
     3.2189 3.5835 1.0986 4.3944

>> log10(A)

ans =
           0 0.3010 1.6902 0.6021
     1.3979 1.5563 0.4771 1.9085
```

We do see something curious here for the function `exp(A)`. It looks like all of the entries are zero until a closer look shows the leading term "`1.0e+035 *`". This means that every entry in the answer is multiplied by this factor 10^{35} (a rather large number). The fact that the other entries look like zero is that there aren't enough decimal places to store each answer.

- `mean(A), median(A), std(A)`

These are some of the basic statistical functions. The output for these are the mean, meadian, and standard deviation of the columns of `A`.

```
>> mean(A)

ans =
     13.0000 19.0000 26.0000 42.5000

>> std(A)

ans =
     16.9706 24.0416 32.5269 54.4472

>> median(A)

ans =
     13.0000 19.0000 26.0000 42.5000
```


www.dbooks.org

- `sort(A), sortrows(A)`

The command `sort` rearranges the data in the columns of `A` in increasing order. The command `sortrows` sorts the rows of `A` in increasing order (determined by the first column).

```
>> sort(A)

ans =
      1  2  3  4
     25 36 49 81

>> sortrows(A)

ans =
      1  2 49  4
     25 36  3 81
```

However, both of these functions can take other arguments. For example, the `sort` function can also be used to sort the columns, or can sort using either 'descend' or 'ascend'. The `sortrows` function can also be used to sort according to other columns. See the Help files for more details.

- `flipud(A), fliplr(A)`

These two functions are abbreviations of "flip up-and-down" and "flip left-to-right". That's exactly what they do:

```
>> flipud(A)

ans =
     25 36  3 81
      1  2 49  4

>> fliplr(A)

ans =
      4 49  2  1
     81  3 36 25
```

- `find`

The function `find` is used to located the position of values in a matrix.

```
>> find(A==1)
```

```
ans =
     1
```

A couple of comments about this example. First, there is a double equals sign in `A==1`. There will be more about this in a future chapter (on relational operators), but you can read this as a question "does `A` equal 1?." The entire line `find(A==1)` indicates the location where the (element of) `A` does in fact equal 1 (namely the first entry). Another example:

```
>> find(a>5)

ans =
     2
     4
     5
     8
```

Here we must again remember to read down the columns of `A` to get to the 2nd, 4th, 5th and 8th entries. We can tweak this last example to get the exact rows and columns containing entries greater than 5:

```
>> [r,c]=find(A>5);[r,c]

ans =
     2 1
     2 2
     1 3
     2 4
```

Here we have given the `find` function two outputs to write to, the variables `r` and `c`, which are displayed together for easier reading using `[r,c]`. The output indicates that the entries of `A` that are greater than 5 can be found in the 2nd row & 1st column, 2nd row & 2nd column, 1st row & 3rd column, and 2nd row & 4th column.

In the next example, we investigate the function `meshgrid`. The `meshgrid` function is used to transform vectors `x` and `y` into arrays `X` and `Y` of sizes appropriate for computation and plotting.

Example 16 Using `meshgrid`

Suppose we wanted to compute the area of a triangle for all possible combinations of the base, ranging from 7 to 10 units, and the height, ranging from 2 to 6 units. Here's the set up (output suppressed):

```
>> Base = 7:10;
```


www.dbooks.org

```
>> Height = 2:6;
```

Now, we can't just multiply these two matrices together, nor can we "dot multiply" them either, since the dimensions of `Base`, 1×4, and `Height`, 1×5, do not line up properly in either case. Instead we resize using `meshgrid`:

```
>> [NewBase,NewHeight] = meshgrid(Base,Height)
```

This creates two matrices, `NewBase` and `NewHeight`, which are both 5×4. We did not suppress the output, so one can see that these are:

```
NewBase =
     7  8  9 10
     7  8  9 10
     7  8  9 10
     7  8  9 10
     7  8  9 10
```

and

```
NewHeight =
     2  2  2  2
     3  3  3  3
     4  4  4  4
     5  5  5  5
     6  6  6  6
```

We can now "dot multiply" them together:

```
>> Area= (NewBase.*NewHeight)/2

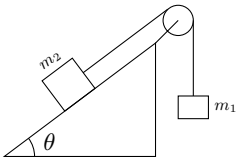
Area =

      7.0000  8.0000  9.0000 10.0000
     10.5000 12.0000 13.5000 15.0000
     14.0000 16.0000 18.0000 20.0000
     17.5000 20.0000 22.5000 25.0000
     21.0000 24.0000 27.0000 30.0000
```

Chapter 3 Exercises

1. Consider a triangle ABC. Suppose the length of the sides AB and AC are 3 and 4, resp. Let θ denote the angle at the vertex A. Use the law of cosines to create a table with two columns: the first column containing θ in increments of 10 degrees from 0 to 180 degrees and the second column containing the length of the side BC. First define the variables AB, AC and θ and then define BC in terms of them. What is the average length of BC for θ between 20 and 150 degrees?

2. Two blocks of masses m_1 and m_2 are connected by a string over a smooth pulley as shown. Assume that $m_2 > m_1$. If the coefficient of friction is μ and $r = m_1/m_2$, then the masses will move at a constant speed (m_2 slides down the slope) if the angle θ is given by $\cos(\theta) = \dfrac{-\mu r + \sqrt{1 - r^2 + \mu^2}}{1 + \mu^2}$.



   a. Compute the values of the angle θ if m_2 is twice the value of m_1 and μ ranges from 0 to 1 in increments of 0.1.

   b. Compute the values of the angle θ if m_2 is ten times the value of m_1 and μ ranges from 0 to 1 in increments of 0.1.

   c. What can you say if $m_1 = m_2$?

3. Write a file that will convert a given number of seconds into years, days, hours, minutes and seconds. Use this file to give the conversion for $1{,}000$ seconds, $1{,}000{,}000$ seconds and $1{,}000{,}000{,}000$ seconds. Some of the following commands may help `fix`, `floor`, `mod`, and `rem`.

4. For a continuous money stream, we have the equation $F = Pe^{rt}$ where F is final value, r is the interest rate, P is the principal, and t is time. Use the `meshgrid` and the `exp` commands to find possible values of F for P ranging from $10,000$ to $50,000$ (in increments of $10,000$) and t ranging from 0 to 10 (in increments of 1). You can select the value of r, but use a reasonable, real-world interest rate (explain how you chose your value of r too).

5. A uniform beam is freely hinged at its ends $x = 0$ and $x = L$, so that the ends are at the same level. It carries a uniformly distributed load of W per unit length and there is a tension T along the x-axis. The deflection y of the beam a distance x from on end is given by

$$y = \frac{W \cdot EI}{T^2}\left[\frac{\cosh\left[a(L/2 - x)\right]}{\cosh(a\,L/2)} - 1\right] + \frac{Wx(L - x)}{2T}$$

   where $a^2 = T/EI$, E is Young's Modulus of the beam and I is the moment of inertia of a cross-section of a beam. If the beam is $10m$ long, the tension is $1000N$, the load $100N/m$ and EI is 10^4, make a table of x versus y where x ranges from 0 to 10 in increments of $1m$. Make sure to define the variables a, W, EI, T and x and then define y in terms of them.

www.dbooks.org

Chapter 4

Matlab: Graphics

In this section, we discuss the graphics capabilities of Matlab.

4.1 Matlab: Graphics

The main commands that we will use in this chapter are

`plot`, `subplot`,`figure` and `hold`

and the customizations

`xlim`, `ylim`, `xlabel`, `ylabel`, and `title`.

Plots in Matlab are created by simply "connecting the dots." This means that you have to provide the actual coordinates of the points − both the x and y coordinates. It is not enough to simply say plot $y = x^2$, you must give exactly which x values you want squared.

Example 17 Basic Plot

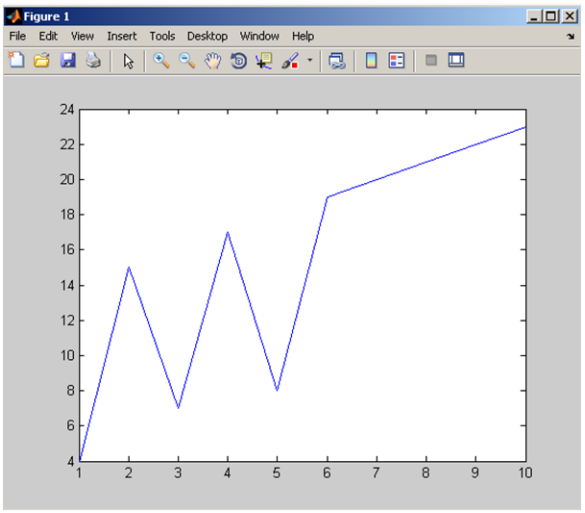
Define:

```
>> x = [1:10]; y = [4 15 7 17 8 19:23];
```

Then to plot the corresponding coordinates, connected by line segments, use

```
plot(x,y)
```

That's all you do. The resulting plot is:

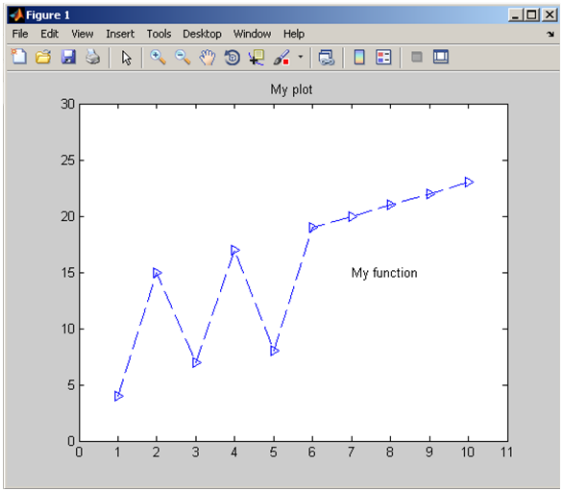


Let's take the last example and add some markers and labels.

Example 18 Plot with labels

```
>> x = [1:10]; y = [4 15 7 17 8 19:23];
>> plot(x,y,'-->b')
>> title('My plot')
>> xlim([0,11])
>> ylim([0,30])
>> text(7,15,'My function')
```

The plot is now:



www.dbooks.org

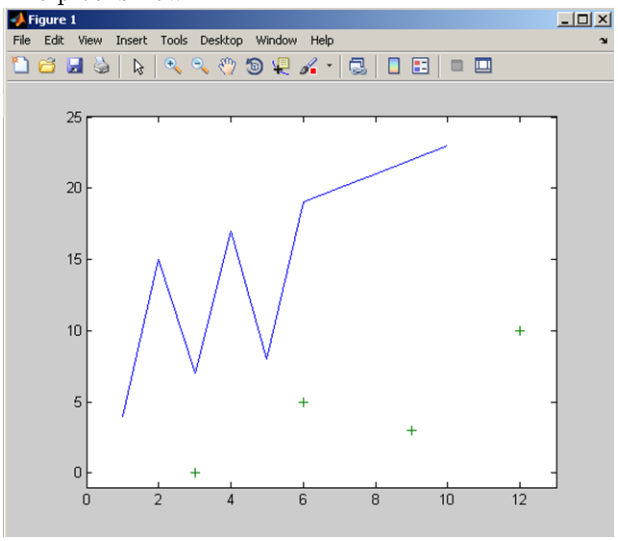
In this last example, you can see that there is a title, the limits on the graph are now 0 to 11 (for x) and 0 to 30 (for y), and there is text on the screen, starting at the coordinate (7,15). The symbols `'-->b '` indicate that the lines should be dashed, the points marked with triangles and the lines colored blue (the default color if none is provided).

If you want to plot more than one set of data in the same figure window, there are two ways to do this. You can put both in the same `plot` command or by using the `hold` command with two `plot` commands (see next example).

Example 19 Two Functions - one `plot` command

```
>> x1 = [1:10];y1 = [4 15 7 17 8 19:23];
>> x2 = [3:3:12];y2 = [0 5 3 10];
>> plot(x1,y1,x2,y2,'+')
>> xlim([0,13]),ylim([-1,25])
```

The plot is now:



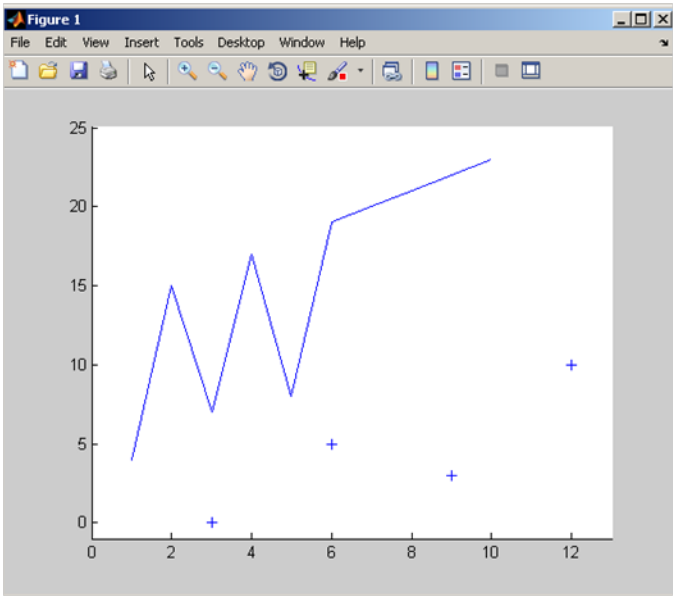
So, from this example we can see that if you want to use the same `plot` command, you simply list the pairings, (with any details following each pair) `x1,y1,x2,y2,x3,y3,`$\cdots$. The default colors start with blue followed by green. The symbol `'+'` means that the second plot is points (not lines) marked with plus signs.

Example 20 Two Functions - two `plot` commands

We use data from the last example, but use the `hold` command.

```
>> x1 = [1:10];y1 = [4 15 7 17 8 19:23];
>> x2 = [3:3:12];y2 = [0 5 3 10];
>> hold on
>> plot(x1,y1)
>> plot(x2,y2,'+')
>> xlim([0,13]),ylim([-1,25])
```

The plot is now:



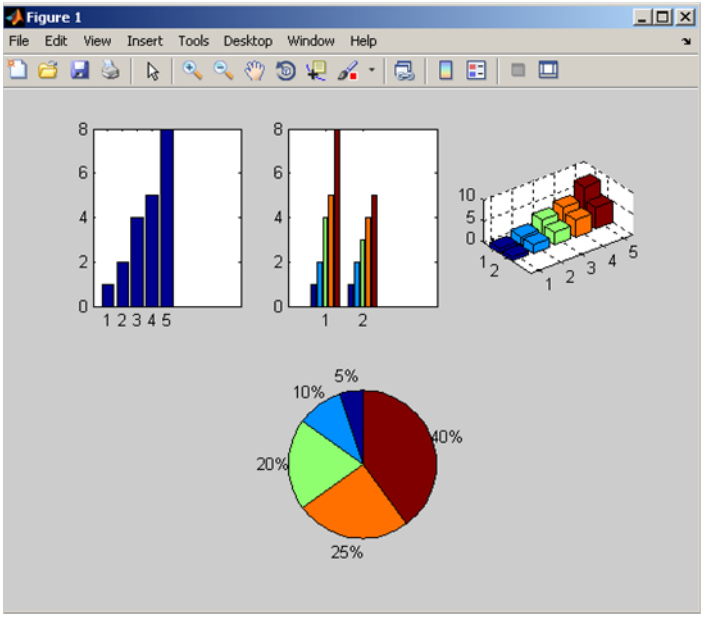
In the last example, if you didn't include the `hold` command, the second plot would simply overwrite the first. Also, since we didn't specify new colors, both plots are blue (the default color of a single plot).

Here we look at the `subplot` command as well as bar graphs and pie charts to create multiple graphs on the same figure.

Example 21 The `subplot` command

```
>> x = [1,2,4,5,8]; y = [x;1:5];
>> subplot(2,3,1), bar(x)
>> subplot(2,3,2), bar(y)
>> subplot(2,3,3), bar3(y)
>> subplot(2,1,2), pie(x)
```


www.dbooks.org



In this example, the `subplot` on each line does two things. First, it (invisibly) subdivides the figure into rows and columns and second, it indicates where the plot will be shown (counting across the rows). That is:

```
subplot(# rows,# columns, location)
```

You can see in the example that even though the plot was "broken up" into 2 rows and 3 columns in the first three subplot lines, the fourth line re-subdivides the plot into 2 rows and 1 column so that the final plot can take up the entire bottom of the plot. Using the subplots does not require the `hold` command.

The previous examples give just a beginning of the graphics capabilities of Matlab (including 3D as in the last example). We will see more graphics in the remainder of the text, but we suggest reading through the help files for (many) more details.

Chapter 4 Exercises

For each problem, include appropriate titles and labels. Make sure to use `xlim` and `ylim` so that the plots fit well in their figure windows.

1. The double logistic curve is defined by

$$y = \text{sign}(x-5) \cdot \left(1 - \exp\left[-\left(\frac{x-5}{50}\right)^2\right]\right)$$

   Use the command `linspace` to get x values from -100 to 100 and plot y for these x values. Add appropriate labels (something simple) to the axes and a title.

2. For the functions $y = e^{x/5}$, $y = \sin(x)$, $y = \sqrt{x}$ for x between 0 and 10:

   a. Plot all three on the same plot, but with different colors (use `linspace` with enough points so that the curves look smooth). Put in a descriptive title and add text inside the window to label the individual graphs.

   b. Plot all three in the same figure window, but with each in their own subplot and each with its own descriptive title.

3. Plot the following curves, either on separate plot windows or as subplots in the same figure window, by first defining the parameter `t=linspace(0,6π)` and then defining x and y in terms of t. Use the standard `plot` command to plot them and add appropriate titles. For each, adjust t for a smoother curve as necessary.

   a. Cardioid: $x = 2\cos t - \cos 2t, y = 2\sin t - \sin 2t$.

   b. Astroid: $x = \cos^3 t, y = \sin^3 t$.

   c. Hypotrochoid: $x = 2\cos t + 5\cos\left(\frac{2}{3}t\right), y = 2\sin t - 5\sin\left(\frac{2}{3}t\right)$.

   d. Epicycloid: $x = 6.5\cos(t) - \cos(6.5t), y = 6.5\sin(t) - \sin(6.5t)$

4. In the same figure window, plot $y = \sin(t + \phi)$ for each of $\phi = 0, \pi/4$ and $\pi/2$, on the range for t from 0 to 2π. Using the `text` command, label the plots inside the figure window. Also, use `xlim`, `ylim` to make the plots fit nicely in the plot window.

Problems continue on the next page...

www.dbooks.org

5. The following data are coordinates of an analemma. Look up the definition of an analemma and give a short description here. Make sure to cite your source.

y	**x**
89.814	−4.67
83.222	−6.824
71.282	−7.7697
60.581	−7.33
50.572	−6.22
41.748	−4.759
38.36	−4.105
32.748	−3.045
26.664	−1.957
21.141	−1.146
17.445	−0.946
14.548	−1.193
13.38	−1.713
13.1497	−1.946
13.569	−2.388
14.942	−2.854
18.269	−3.087
22.189	−2.84
28.455	−1.992
34.742	−0.834
38.876	−0.117
43.349	0.607
52.124	1.824
64.591	2.842
74.997	2.775
85.45	1.196
90.649	−1.265
91.403	−2.559

As you can see, the data was unfortunately entered backwards so that the second column gives the x coordinate and the first column gives the y coordinate. First enter the data into a 2-column matrix (in a variable of your choice). Use `fliplr` to redefine the columns and define the variable x by referencing the first column and y by referencing the second column. Plot the data with an appropriate title. Note that you may have to adjust the data so that the analemma is a closed figure.

Extra credit Use the data to find the length of the analemma curve.

Chapter 5

Matlab: User Defined Functions

In this section, we discuss how to create new functions in Matlab.

5.1 Matlab: User-Defined Functions

Up to now, the M-files that you have created are called "script" files. Now we use the editor to create **function** M-files. Why would you want to do this? Well, Matlab has many built-in functions, but you may need to create your own to solve a specialized problem.

Lets look at how a function is put together. In the command window, if you were to type

```
>> help sin
```

you would get

```
SIN Sine of argument in radians.
SIN(X) is the sine of the elements of X.

   See also asin, sind

   Reference page in Help browser
      doc sin
```

In reading this, you can see the purpose of the function, the correct usage of the function and references to additional related functions. When you create your own functions, you will also need to create similar information.

www.dbooks.org

Here's how to create a function in Matlab. In the Editor window, follow these steps:

1) On the first line(s) put comments that describe the function and, most importantly, a usage statement or HOW THE USER ENTERS THE FUNCTION!!!
2) On the next line, put the word `function` followed by:
a) In `[ ]`, put the variable(s) (separated by commas) for the function output(s).
b) The = symbol
c) Your function name
d) In `( )`, put the variable(s) (separated by commas) for the function input(s).
3) On the following lines, put the calculations that define the output variables based on the input variables.

Easy right??

A couple of comments before we get to some examples.

Key Idea 1

Function Tips
1) If you have exactly one output the brackets aren't needed (see step 2a)
2) Functions must follow the same naming conventions as for variables (do a Google search)

Let's look at an example of a function that computes the area of a triangle.

Example 22 Area of a triangle function

Here is the syntax that you would type into an editor window:

```
% Usage:  Area=TriangleArea(base,height)
% Inputs:  base and height are scalars
% Output:  Area - area of a triangle
function Area=TriangleArea(base,height)
Area = 0.5*base.*height;
```

Now save this file as `TriangleArea.m`. Your file name must exactly match the name of the function (except for the .m part) or else the function will not run.

In order to execute any function, you cannot use the Matlab run button. You must run the function from the command window or another M-file since you need to provide the input(s). First, check that you have provided enough comments about how to use the function:

```
>> help TriangleArea
```

which will give your description of how to use the function. Now, run the function with:

```
>> A = TriangleArea(3,4)

A =
     6
```

NOTE: In the program `TriangleArea.m` the variables `Area, base` and `height` are called local variables. That means they are only used in this specific program. They don't appear outside the function (or in the Workspace).

Example 23 Rectangle function

Here is the syntax that you would type into an editor window:

```
% Usage:  [Area,Perimeter]=Rectangle(base,height)
% Inputs:  base and height are vectors of the same length
% Outputs:
% Area - area of a rectangle
% Perimeter  perimeter of a rectangle
function [Area,Perimeter]=Rectangle(base,height)
Area = base.*height;
Perimeter = 2*(base+height);
```

Now save this file as `Rectangle.m` Again, make sure the usage comments are correct:

```
>> help Rectangle
```

to see your helpful comments. Now run the function:

```
>> [A,P] = Rectangle(3,4)

A =
     12
P =
     14
```

Here again we are reminded of the fact that `Area` and `Perimeter` are local to the function, so when we run the function, we can call the outputs by different names. When we run the function we use the shorter `A` and `P` to store

www.dbooks.org

the areas and perimeters.

What happens if you try to run the function without output variables? Try

```
>> Rectangle(3,4)

ans =
     12
```

You only get the value that would have been assigned to the first output variable (here it's the area). Since the calculations are just as they would be for matrices, we can try:

```
>> base = 1:4; height = [3 6 10 12];
```

and run the function:

```
>> [A,P] = Rectangle(base,height)

A =
     3 12 30 48
P =
     8 16 26 32
```

which gives us the areas and perimeters of the rectangles of the corresponding bases and heights.

Here are a few commands of interest:

```
nargin('TriangleArea')          (number of inputs)
nargout('TriangleArea')         (number of outputs)
nargin('mesh')                  (variable number of inputs)
type('TriangleArea')            (returns code of M file)
```

One more example:

```
type('sin')
```

gives an output of `'sin' is a built-in function`, which indicates that this last one is a built-in function whose code is not accessible.

Chapter 5 Exercises

When you submit this assignment, there should be seven separate files. Problems 1-6 should be in their own function m-files (properly commented of course). Problem 7 should be a separate script m-file that uses the code from the first 6 problems. Make sure each of your files starts with your VMI id (mine would be `siemerstjFunction`).

1. The height of a projectile is given by

$$f(t) = -\frac{9.8}{2}t^2 + 147t + 500, \quad t > 0$$

   Create the function `ProjMotion` that has one input, `time`, and two outputs `height` and `velocity`.

2. Create an appropriately named conversion function that has one input (`dollars`) and one output (`ConversionTable`). The conversion table should contain four columns, with `dollars` in the first column and the conversions into Yen, Euros, and British Pounds in the next three columns (look up the current conversion rates online and cite your source).

3. Create a function called `plotlines` with two inputs and no outputs with the following details. The first input should be a 1×2 matrix (representing the base point, B, in the plane). The second input should be a 2×2 matrix (representing two more points in the plane, P and Q). When the function is run, there should be a plot of two line segments, BP and BQ, with the two distances displayed halfway along each line.

4. Create a function called `RemRow` with two inputs `M` and `n`. The output should be a matrix `NewM` that comes from removing the nth row of `M`.

5. Create a function called `RemCol` with two inputs `M` and `n`. The output should be a matrix `NewM` that comes from removing the nth column of `M`.

6. Create a function that will convert a given number of seconds (the input of the function) into years, days, hours, minutes and seconds (listed in one 1×5 matrix output). Some of the following commands may help: `fix`, `floor`, `mod`, and `rem`.

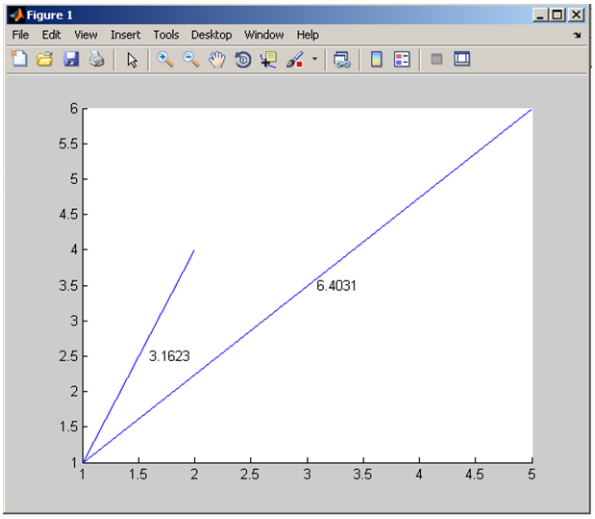
7. In this problem, you will use your functions from the previous problems, running them with specific numeric input(s).

   (a) Run your `ProjMotion` function for 0 to 30 seconds in increments of 0.5 seconds and use the output to plot `time` versus `height`. Use the output of the function (and appropriate Matlab functions) to figure out at what time the object starts to fall back to the ground.

   (b) Run your currency conversion function to give the output from an input `dollars`) ranging from 0 to 100 in increments of 5.

   (c) Run your `plotlines` function with the inputs `[1, 1]` and `[2, 4;5, 6]`. Your output should look like

www.dbooks.org



(d) Run your `RemRow` function with inputs `[1 2 3; 4 5 6; 7 8 9]` and `2`. The output should be the matrix with the 2nd row removed.

(e) Run your `RemCol` function with inputs `[1 2 3; 4 5 6; 7 8 9]` and `2`. The output should be the matrix with the 2nd columns removed.

(f) Run your time conversion function to give the conversion for $1{,}000$ seconds, $1{,}000{,}000$ seconds and $1{,}000{,}000{,}000$ seconds.

Chapter 6

Matlab: Input/Output

In this section, we discuss the input/output (I/O) capabilities in Matlab.

6.1 Matlab: Input Commands

In order to run either script M-files or function M-files, the user needs to provide data, either numerical or text ("strings"). The first command we consider is simply called `input`.

Example 24 Using the `input` command

Here is a simple section of code to collect a person's first name and their age.

```
>> FirstName = input('Enter your first name :  ','s');
>> Age = input('Enter your age :  ');
```

Once executed, the user enters values for each, say

```
Enter your first name :  Bob
Enter your age :  25
```

In this example, the `'s'` in the first `input` indicates that the program expects text input and the second `input` (without the `'s'`) indicates that the program expects numeric input. When executed, the lines collect input one at a time. Note the text `Waiting for input` on the bottom left of the Matlab window while the user enters the name, `Bob`, and age, `25`. Due to the semicolons, there is no output to the screen, but the values have been stored in the variables `FirstName` and `Age` (see the Workspace window to confirm).

So how do we display this information? There are several ways to output this information. Here we consider the two commands `disp` (for "display") and `fprintf` ("formated print to file"). The help files are useful so look them up!

www.dbooks.org

We can present this information as follows.

Example 25 Using the `disp` command

Using the data from the last example:

```
>> disp('Your first name is ')
>> disp(FirstName)
>> disp('Your age is ')
>> disp(Age)
```

which has the output:

```
Your first name is
Bob
Your age is
     25
```

This example is ok, but it would be nicer if the data was in one line. That's where the command `fprintf` comes in. Even though it is a formatted print "to file" you can have the output sent to the Command Window instead.

Example 26 Using the command `fprintf`

```
>> fprintf('%s is %.0f years old.\n ',FirstName,Age)
```

produces

```
Bob is 25 years old.
```

Certainly, there is more involved in the function `fprintf` in order to produce the output we want. Let's look at the individual pieces of this example.

• The `%` signs are NOT for comments this time (note: they aren't green), they are now used as place holders for the data.

• The `%s` indicates that a string is expected as input.

• The `%.0f` indicates that we expect a numeric input, want zero decimal points and the value should be in Fixed-point format.

For more information on specific formats and "conversion characters" (which is what `f` and `s` are here), look at the `fprintf` command in the help files.

• The `\n` is an "escape character" that creates a New Line.

• The data `FirstName` and `Age` are listed after this, separated by commas.

• When the command is run, Matlab places the first data value, `FirstName` (i.e. `Bob`), in the `%s` position and the second data value, `Age` (i.e. `25`), in the `%.0f` position. Got it?

Just for completeness, we can also force the `disp` command to act like `fprintf` as follows:

Example 27 `disp` using concatenated strings

```
>> disp([FirstName,' is ', num2str(Age),' years old.'])
```

produces

```
Bob is 25 years old.
```

Basically, we are displaying a string array made up of several strings through concatenation. We also see the command `num2str` which does exactly what it says, it converts a number into a string. This is necessary since numeric data and strings don't play well together in Matlab. In order to create a string array, all of the parts must be strings. If we tried to run this without the `num2str` command, we would get

```
>> disp([FirstName,' is ', Age,' years old.'])
```

with output

```
Bob is □ years old.
```

where the square indicates that the ASCII code for the value 25 is unprintable (look it up or don't worry about it - just remember the `num2str`).

Let's look at how to read and write larger data files next.

Example 28

Create the following table of years and monthly world steel production (in thousand metric tons) in an Excel file and save the file as WorldSteelProduction.xls (or .xlsx). Make sure to save this file in the directory listed at the top of the Matlab window, the Currect Directory.

www.dbooks.org

2005	2006	2007	2008	2009	2010
91377	95037	107789	112870	86476	113375
84980	91183	101465	107465	86610	107119
93198	102183	112988	119934	92144	122196
93381	101604	110241	117023	89644	120497
95290	105501	112933	121062	96177	124567
92095	104636	112159	118851	100661	118346
90205	104350	110377	116770	104701	114365
91536	102630	109024	112726	108351	113141
93144	103676	111956	107723	110773	112340
98534	106913	114703	99202	114765	117377
94340	104817	109936	86513	108596	114637
95368	105306	111506	81704	107792	116157

We will now read in this data using `xlsread`, plot it, and access data from it using the mouse and the graphical input command `ginput` ("gee-input").

```
>> Steel = xlsread('WorldSteelProduction.xls');
```

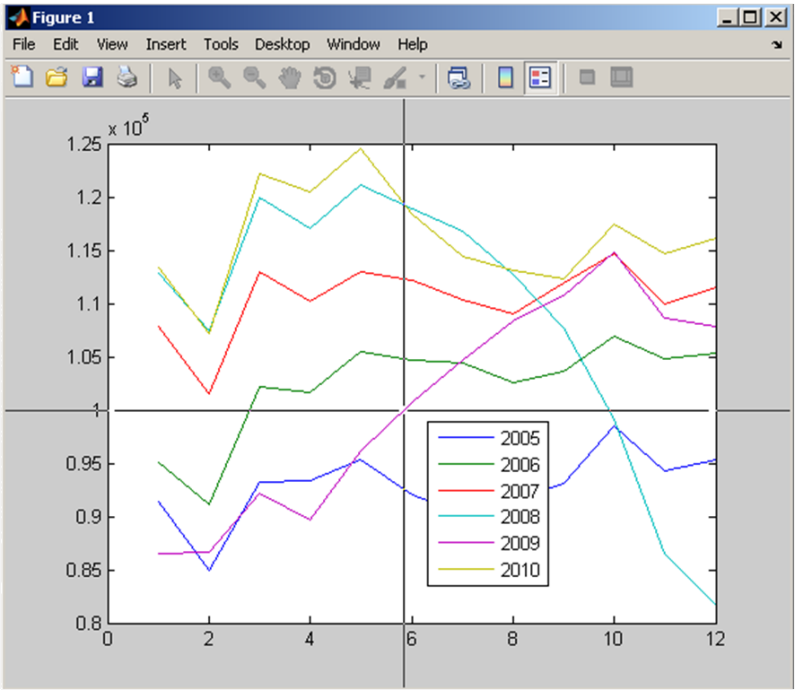
Next, we plot the data

```
>> plot(1:12,Steel(2:end,:))
>> legend('2005','2006','2007','2008','2009','2010', ...
   'Location','best')
```

The initial problem is to find the first time in 2009 that steel production is at 100,000 thousand metric tons. We use the `ginput` command for the user to collect this data.

```
>> disp('Select the 2009 time when production is 100,000.');
>> [t,l] = ginput(1) % mouse click
```

If the user hovers the cursor over the resulting plot, we see cross-hairs appear:



A left-click creates the output to appear in the Command Window:

```
t =
      5.8756
l =
      9.9934e+004
```

Note that the level is listed in engineering format. If we want to finish this example with a formatted output, we could use the following:

```
>> fprintf('Level %6.1f occurs %3.1f months into 2009.\n',l,t)

Level 99934.2 occurs 5.9 months into 2009.
```

which is in mid-June 2009. Note that the accuracy of the level and time are dependent on the user's mouse click so the output may not look exactly the same.
(Data from `http://www.worldsteel.org/`)

www.dbooks.org

Key Idea 2

`fprintf` with matrices
If there are not enough place holders, `fprintf` will cycle through the values in the given data matrix, USING COLUMN PRECEDENCE, until all of the values are exhausted.

Example 29

Here we use `fprintf` and Key Idea 2 to display a matrix of values. Find the sine, cosine and tangent of theta from 0 to 2π by steps of $\pi/10$.

```
>> theta=[0:pi/10:2*pi];
>> values=[theta;sin(theta);cos(theta);tan(theta)];
```

To display these, we can use a simple `fprintf`.

```
>> disp('Theta Sine Cosine Tangent ');
>> fprintf('%5.2f %6.2f %7.2f %9.2f\ n ',values)
```

gives output:

```
Theta   Sine   Cosine   Tangent
 0.00   0.00   1.00       0.00
 0.31   0.31   0.95       0.32
 0.63   0.59   0.81       0.73
 0.94   0.81   0.59       1.38
 1.26   0.95   0.31       3.08
 1.57   1.00   0.00 16331239353195370.00
 1.88   0.95  -0.31      -3.08
 2.20   0.81  -0.59      -1.38
 2.51   0.59  -0.81      -0.73
 2.83   0.31  -0.95      -0.32
 3.14   0.00  -1.00      -0.00
 3.46  -0.31  -0.95       0.32
 3.77  -0.59  -0.81       0.73
 4.08  -0.81  -0.59       1.38
 4.40  -0.95  -0.31       3.08
 4.71  -1.00  -0.00 5443746451065123.00
 5.03  -0.95   0.31      -3.08
 5.34  -0.81   0.59      -1.38
 5.65  -0.59   0.81      -0.73
 5.97  -0.31   0.95      -0.32
 6.28  -0.00   1.00      -0.00
```

In this example, the reason for the large values in the last column are because the tangent function tends to infinity as the angle tends to $\pi/2$ (and 1.5708 is close to $\pi/2$). Actually, the value $\tan(\pi/2)$ returns `1.6331e+016`. Why isn't "infinity" the returned value? Well, it has to do with how values are stored in Matlab (using double-precision). The value `1.6331e+016` is actually the reciprocal of the value `eps = 10^(-52)`. If you want to know more, head to the help files (or Google, of course).

www.dbooks.org

Chapter 6 Exercises

When you submit this assignment, there should be three separate files. Problems 1 and 2 should be in their own function m-files (properly commented of course). Problems 3-6 should be a separate script m-file that uses the code from the first 2 problems and Excel spreadsheets on Angel. As usual, make sure each of your file names start with your VMI id.

1. For this problem, you will modify your projectile motion function from the last homework. Now, as part of your new function, `ProjMotion2`, plot the `height` versus `time`, use `ginput` for the user to identify the maximum height, and use `fprintf` to display a sentence describing the maximum height and corresponding time.

2. For this problem, modify your conversion function from the previous homework and rename it `Conversion2`. Now, use `fprintf` in order to create a nice-looking table showing conversions between Dollar, Euro, Yen, and British Pounds. Include a header for each column.

3. Run your `ProjMotion2` function for `time` 0 to 30 seconds (in increments of 0.5 seconds) and your function `Conversion2` for `dollars` from 0 to 100 in increments of 5.

4. Data on the Federal GDP and Deficit (as a percentage of GDP) is in a file called `GDPAndDeficitByYear.xlsx` (in the Appendix in Table 1 and saved on Angel). First save this file to a directory on your computer and then use the command `xlsread` to load this file. Assign the data in the file to a variable of your choice and create a plot of year (1st column) versus GDP (2nd column, in billions of dollars) including appropriate titles and labels on the axes. Use `ginput` for the user to click on the first time the GDP exceeds $10 trillion. Use `fprintf` to display the information in a useful way.

5. Consider the data from the previous problem (loaded with `xlsread` from the file called `GDPAndDeficitByYear.xlsx`). Now create a plot of year versus Deficit (3rd column), which is listed as a percentage of the GDP for the corresponding year. Including appropriate titles and labels on the axes. Use `ginput` for the user to click on **all** of the times the Deficit percentage has hit 10 percent and then use `fprintf` to display the information in a useful way.

6. Data on unemployment percentages (per month) during 2000-2010 is in a file called `UnemploymentByYear.xlsx` (in the Appendix in Table 2 and saved on Angel). First save this file to a directory on your computer and then use the command `xlsread` to load this file. Note that the first row of the file is text so you will only want to load the information on rows 2 through 12. Assign the data in the file to a variable of your choice. Since the data for December 2010 has not been entered, correct it (in Matlab) with the correct value (look it up online). Next, create a plot of month versus unemployment (including appropriate titles and labels on the axes). Your plot should have 11 curves, one for each year and in different colors. Include a legend in your plot for reference (labelled correctly).

Chapter 7

Matlab: Programming Structures

In this section, we discuss relational operators, logical operators and loop structures in Matlab.

7.1 Matlab: Relational Operators

If we want to compare two values, we have the following relation operations:

`< , <= , >, >=, ==, ~=`

It is helpful to read any statement involving these symbols as a question as the answer is either `0` (false) or `1` (true).

Example 30 Basic Relations

```
>> 3 == 4   ("is 3 equal to 4?") gives 0 (false)
>> 3 <= 4   ("is 3 less than or equal to 4?") gives 1 (true)
```

7.2 Matlab: Logical Operators

To create compound statments of the relational operators, we can combine these using the logical operators. The truth or falsity of these follows basic rules of logic, so it helps to have some knowledge of truth tables. Again, read them as

www.dbooks.org

questions!

`&`	("and" returns true if both parts are true)
$\sim$	("not" returns true if the initial value is false)
`|`	('or" returns true if either part is true)
`xor`	("exclusive or" returns true if either part is true, but NOT both true)

Example 31 Basic Logic

```
>> (3 >= 2) & (3+4 == 6)

ans=
     0
```

Here, the question "is 3 greater than or equal to 2 AND 3 plus 4 equal to 6?" is answered as false (or a zero) since even though 3 is greater than 2, it is not true that 3 plus 4 is 6.

Note that there are also the operators && and || which also mean "and" and "or" but are called "short-circuited" operators (look it up). As you work with these in the Editor, orange lines on the right side of the screen may appear suggesting you use && in place of & (or vice-versa) and || in place of | (or vice versa). Just take those suggestions and you'll be fine.

These operations extend to matrices with an entry-by-entry comparison of matrices of the same size.

Example 32

```
>> a=[0,1;1,2], b=[0,0;1,1]

a =
     0 1
     1 2
b =
     0 0
     1 1
```

with

```
>> a&b, a|b, xor(a,b)
```

returns

```
ans =
     0 0
     1 1
```

```
ans =
     0 1
     1 1

ans =
     0 1
     0 0
```

One minor point to note is that while Matlab treats 0 as false, any other positive whole number is considered as true. So, in this example, even though the $(2, 2)$ entry of a is a 2, it is considered as "true" for logical comparison.

7.3 Matlab: `if` and `switch` commands

We discuss the constructs if\elseif\else and switch\case\otherwise in this section.

Suppose we want to have part of our program run only under certain conditions. For this, we use an if\else structure. The basic format of this structure is:

```
if (put condition(s) here)
     (put calculations here to be run if the conditions are met)
elseif (other condition(s))
     (calculations that will run under the new conditions)
...  (more elseif statements, if desired)
else
     (calculations run if none of the previous conditions are met)
end
```

Example 33 Using `if`

Let's write a script M-file that lets a user input a number and then displays if that number is less than 5, between 5 and 10 (inclusive), or greater than 10.

```
number = input('Input a number ');
if number < 5
     disp('Your number is less than 5.')
elseif number >=5 && number <= 10
     disp('Your number is between 5 and 10.')
else
     disp('Your number is greater than 10.')
end
```

Try running this program using various numbers for input.

www.dbooks.org

The switch\case structure is similar to the if structure, but has a few advantages. First of all it is easier to read and second, it is better if you are comparing strings (of possibly different lengths). The basic format is:

```
switch (expression to test)
     case (case condition)
          (output in that case)
     case (case condition)
          (output in that case)
     ...(more cases)
     otherwise
          (do this if no cases are met)
end
```

Example 34 Using `switch`

Let's check to see if a cadet is in his\her first two years at VMI.

```
Year = 'second class';

switch Year
     case {'fourth class','third class'}
          disp('You are in the first two years.')
     case {'second class','first class'}
          disp('You are in the last two years.')
     otherwise
          disp('You must be a 5th year.')
end
```

The cases are grouped by curly brackets so that a case will be satisfied if the value of Year is any of the values in a specific case. Once this code is executed, the switch command will look at the value of Year and the output should be

```
You are in the last two years.
```

7.4 Matlab: `for` and `while` Loops

We discuss the for and while loops in this section.

Suppose we want to have part of our program re-run a preset number of times. For this, we use a for loop. The basic format of this structure is:

```
for (put counter conditions here)
     (put calculations here)
end
```

Example 35 Comparing for and sum

Let's compare two methods for adding up the first five integers. Using the sum command we can use

```
>> sum(1:5)
ans =
     15
```

Now, using a for loop to create a cumulative sum:

```
totalsum=0; % initialize
for i=1:5
     totalsum=totalsum+i;
end
disp(totalsum)
```

The variable totalsum will have value 15.

In example 35, the for loop was the long way of doing the problem (and therefore stresses the power of the sum function), but the following example shows a more in-depth for loop.

Example 36 Using for

Let's find the first 10 Fibonacci numbers using the recursive definition.

```
%initialize the matrix
A=zeros(1,10);
A(1)=0;
A(2)=1;

for i=3:10
     A(i)=A(i-1)+A(i-2);
end

disp(A)
```

The output for this would be

```
A =
     0 1 1 2 3 5 8 13 21 34
```

Now, suppose want to have part of our program run until a certain condition

www.dbooks.org

is met, even though we may not know how many times the loop will need to run until that happens. For this, we use a `while` loop.

The basic format of this structure is:

```
while (put conditions for the loop to keep running)
     (put calculations here)
end
```

Let's rewrite the last example with a slight twist. Let's find the Fibonacci numbers until they exceed 1000.

Example 37 Using `while`

```
%initialize the matrix (we dont know how big it will be so
%   we will grow it in the while loop).
A(1)=0;
A(2)=1;
j=2; %initialize a counter
while A(j) < 1000
     j=j+1; %move the counter along
     A(j)=A(j-1)+A(j-2);
end

disp(A)
```

The output for this would be

```
A =
     0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

The loop concludes when we pass 1000.

Chapter 7 Exercises

When you submit this assignment, there should be five separate files. Problems 1-3 should be in one script m-file. Problems 4a and 4b should be in their own function m-files (properly commented of course). Make sure each of your files starts with your VMI id.

1. In some computer programs, a loop is necessary to add a list of numbers. Consider the cubes of the first 100 integers, $1^3, 2^3, 3^3, \cdots, 100^3$.
   a. Create a matrix of the numbers 1 to 100 and use the `sum` command to find the sum of the cubes of the numbers.
   b. Use a `for` loop to find the same sum by adding the cubes of the numbers one at a time (onto a cumulative sum).
   c. A formula that you may have seen is that the sum of the cubes of the numbers from 1 to n is given by $\frac{n^2(n+1)^2}{4}$. Confirm your sum in the previous parts using this formula.
2. Use the data in chapter 4 exercises and a loop structure to find the length of the Analemma.
3. In this exercise, you will be comparing the capabilities of the if\else and switch\case structures by creating two programs. In each program, prompt the user to enter the last name of one of the last 10 presidents of the United States. Your program should display an informative sentence in response (including the presidency number and when he served), or display an error message if the entered name is invalid. In the first program, use an if\else structure and in the second program, use a switch\case structure. Note: you will need to know how to compare strings (look in the help files).
4. Modify your `RemoveRow` Function from the previous homework in two ways (you should have two functions, perhaps called `RemoveRowIf` and `RemoveRowWhile`)
   a. In the first function, use an `if` statement to have an error message displayed if the user enters an invalid row. Then prompt them to enter a new row number. In your prompt, you must indicate the allowable range of rows. Here they only have one chance to re-enter the row.
   b. In the second function, use a `while` loop so that if the user enters an invalid row on their first try, they will be able to continue to enter rows until they enter a correct value. Again, in your prompt, you must indicate the allowable range of rows.

www.dbooks.org

Chapter 8

Matlab: Applications

In this chapter, we put several aspects of programming together. You will learn about the numerical methods of Riemann sums, the Trapezoidal Rule and Simpson's rule. We'll also investigate aspects of the Traveling Salesman problem.

8.1 Matlab: Numerical Methods

In Calculus, you learn how to use Riemann sums, the Trapezoidal Rule and Simpson's rule to approximate definite integrals, which represent area under a curve for positive functions $f(x)$.

$$\text{The area between the function and the } x\text{-axis} = \int_a^b f(x)\,dx$$

Each involves finding the sum of areas of approximating shapes. Given a function $f(x)$ defined on an interval $[a, b]$, and a value n, the process is as follows.

1. First subdivide the interval $[a, b]$ into n equal subintervals.

2. Assign $x_0 = a, x_1$ as the right endpoint of the first subinterval, x_2 as the right endpoint of the second subinterval, $\cdots$, x_n as the right endpoint of the nth interval (that is $x_n = b$).

   For example, if $a = 0, b = 1, n = 4$, then $x_0 = 0, x_1 = .25, x_2 = .5, x_3 = .75$ and $x_4 = 1$.

   In this example, note that even though $n = 4$, there are five x values.

3. Next, calculate the values $f(x_0), f(x_1), f(x_2), \cdots, f(x_n)$ by substituting the x_i values into $f(x)$.

The formulas for each of the methods are:

Riemann Sum:

$$\int_a^b f(x)\,dx \approx \left(\frac{b-a}{n}\right)[f(x_0)+f(x_1)+f(x_2)+\cdots+f(x_{n-2})+f(x_{n-1})]$$

Trapezoidal Rule:

$$\int_a^b f(x)\,dx \approx \left(\frac{b-a}{2n}\right)[f(x_0)+2f(x_1)+2f(x_2)+\cdots+2f(x_{n-2})+2f(x_{n-1})+f(x_n))]$$

Simpson's Rule:

$$\int_a^b f(x)\,dx \approx \left(\frac{b-a}{3n}\right)[f(x_0)+4f(x_1)+2f(x_2)+\cdots+2f(x_{n-2})+4f(x_{n-1})+f(x_n))]$$

NOTE: The pattern of coefficients in the Trapezoidal rule is $1, 2, 2, 2, \cdots, 2, 2, 1$ (each of the middle terms is a 2). In Simpson's rule, the pattern is $1, 4, 2, 4, 2, 4, \cdots, 2, 4, 2, 4, 1$ where the middle terms start with a 4, alternate between 2 and 4, and end on a 4 (important!!). This last fact forces n to be an even number for Simpson's rule to work.

Homework

Write a Matlab **function** with the following details.

- Name the function appropriately (mine could be siemerstj_num_methods)
- Comment the function appropriately including a description of the usage.
- The function should have three inputs **a**, **b**, **n**
- The function should have three outputs **Rsum, Tsum, Ssum**
- Once the function is run, a **menu** should appear with three choices of functions to choose from:

$$x^2, \quad \sin x, \quad \sqrt{\frac{1}{2\pi}}e^{-x^2/2}$$

- Once the user clicks on a function from the menu, the function should compute the three approximations and output the values in a nice format (Hint: use `fprintf`).
- If n is not even, then only the Rsum and Tsum should be displayed with a message saying the Ssum could not be calculated.

Here is a sample run. Suppose I typed this :

```
>> [Rsum,Tsum,Ssum]=siemerstj_num_methods(0,1,6)
```

and then clicked on the x^2 button in the menu. Then the output should be:

```
Rsum=.25463, Tsum=.3380, Ssum=0.33333
```


www.dbooks.org

8.2 Matlab Traveling Salesman

A famous problem in mathematics is the Traveling Salesman Problem where the minimum distance of traveling from a home city, through n cities and back to the home city. In this program, you will simulate part of this problem.

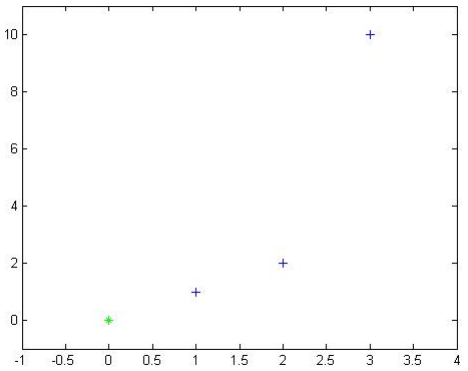
Homework

You will write a Matlab `function` with the following details:

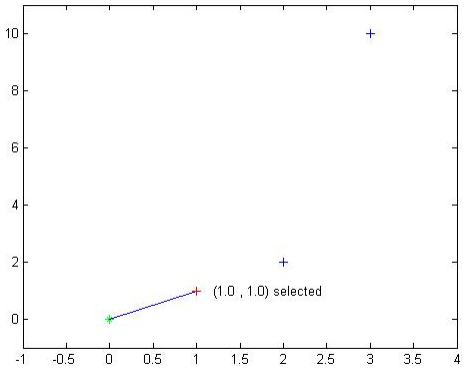
- Name the function appropriately (like siemerstjTravelingSalesman)
- Comment the function appropriately including a description of the usage.
- The function has **one input** (a matrix of locations of the cities) and **no outputs**.
- Once the function is run, a figure will appear with the cities indicated with one type of symbol and the home city (at the **origin**) with another.
- The user will select cities one at a time with the mouse.
- Each time a city is selected, a line is drawn from the previous city with a message of the city selected.
- Repeat until all cities are selected.
- Once the last remaining city is selected, a line is drawn back to the home city and the total distance of the trip is displayed.

Code execution:

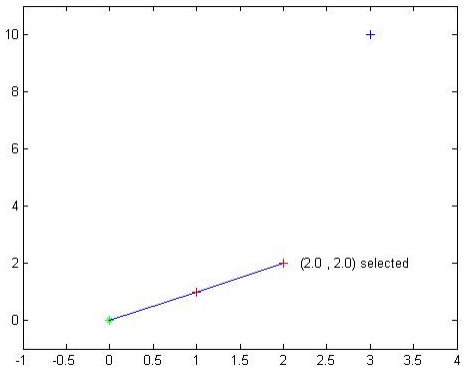
Suppose I enter `siemerstjTravelingSalesman([1 1;2 2;3 10])`. Then the following picture chould appear. The "home base" plotted as a green asterisk at the origin and the three cities are plotted at the coordinates $(1,1),(2,2),(3,10)$. Note that the screen is a little larger so that the cities do not fall on the edge of the plot (see either `xlim\ylim` or `axes`).



When I click on the closest city (at $(1, 1)$), I now have

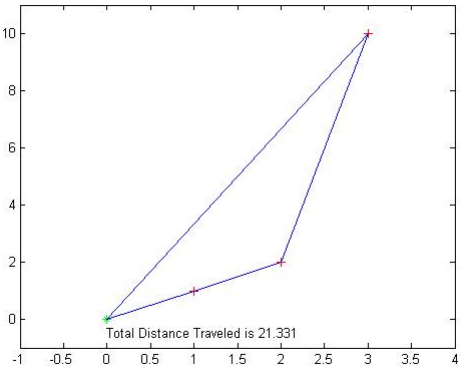


Click the next city...



Finish with the last city...

www.dbooks.org



A few issues arise as you work through the code for this project:

1) Does the user have to click exactly on top of a city, or can you write the code so that they can click close to the city (and how "close" is close enough?).
2) How do you keep the user from selecting the same city twice?
3) How do you make the text for one city "disappear" when the next city is clicked? (see `set` command for one possibility).

Chapter 9

Matlab: Curve Fitting

In this section, we discuss how to fit various types of curves to data.

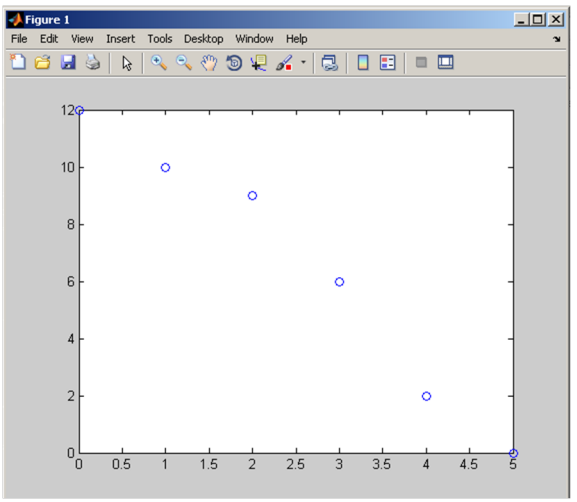
9.1 Matlab: Curve Fitting

The main commands for this section are `polyfit`, `polyval` and `interp1`. Let's begin with linear regression.

Example 38 **Linear Regression**

Plot a set of data:

```
>> x=0:5;y=[12,10,9,6,2,0];
>> plot(x,y,'o')
```



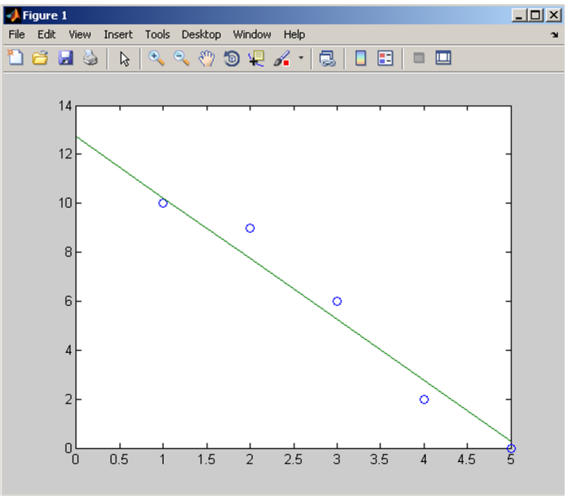
www.dbooks.org

The data certainly seems to have a (downward) linear trend. We can find the line of best fit as follows:

```
>>coeffs=polyfit(x,y,1);
>>besty=coeffs(1)*x+coeffs(2);
```

The command `polyfit` returns the matrix `[-2.4857 12.7143]` of the slope and y-intercept of the line of best fit. The y values of this line corresponding to x (from 0 to 5) are stored in the variable `besty`. We plot them together using:

```
>>plot(x,y,'o',x,besty)
```



We often use regression lines to guess y values for x values that are not included in the data set. For this, we use interpolation.

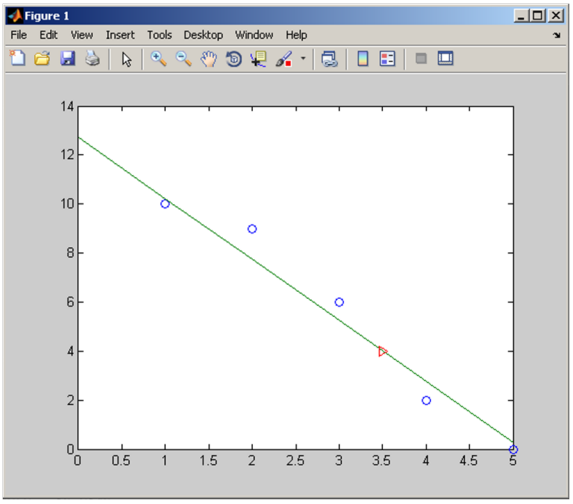
Example 39 Interpolation

Using the data from example 38, we would like to know the y value corresponding to the x value 3.5. For this, we use the command `interp1` (that's the number one on the end, not a lowercase for the letter L).

```
>> interp1(x,y,3.5,'linear')
ans =
     4
```

We plot these together with the interpolated point indicated with a red triangle.

```
plot(x,y,'o',x,besty,3.5,4,'r>')
```



Note that if you try to use the `interp1` function for data that is outside the data set, you will get an error since you are trying to "extrapolate" instead of interpolate.

```
>>interp1(x,y,10.5,'linear')
```

gives

```
ans =
     NaN
```

This indicates that the answer is `NaN` or "Not a Number" (look it up in the Help files).

Higher Order Polynomial Fitting

Let's try fitting a fifth degree polynomial to the data in example 38.

Example 40

```
>> x=0:5;y=[12,10,9,6,2,0];
>> coeffs5=polyfit(x,y,5)
```

returns

```
coeffs5 =
     -0.0167  0.3333 -2.0833  4.6667 -4.9000 12.0000
```


www.dbooks.org

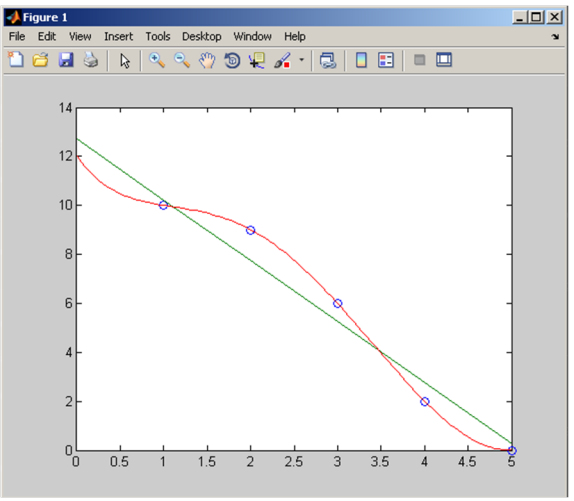
which are the coefficients for the approximating 5th order polynomial, namely $y = -0.0167x^5 + 0.3333x^4 - 2.0833x^3 + 4.6667x^2 - 4.9x + 12$.

We could type out the full polynomial, but there is a shortcut. We can use the function `polyval` along with `linspace` to give a smooth approximating curve.

```
>> x5=linspace(0,5);
>> y5=polyval(coeffs5,x5);
```

We plot the curves

```
>> plot(x,y,'o',x,besty,x5,y5)
```



Interactive Fitting Tools

If you aren't sure about the fit you want, you can use the interactive fitting tools. We start again by closing any figure windows, clearing out the variables and clearing the Command Window. A short cut to do this could be:

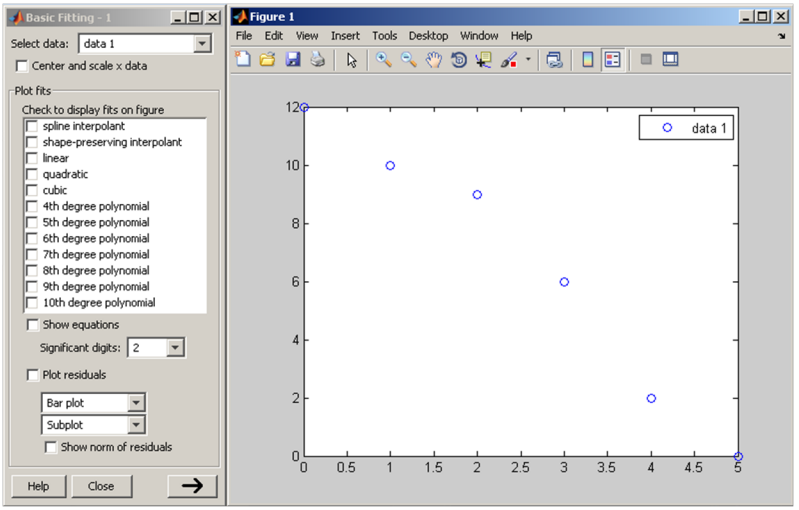
```
>> close all,clear,clc
```

Re-enter the data.

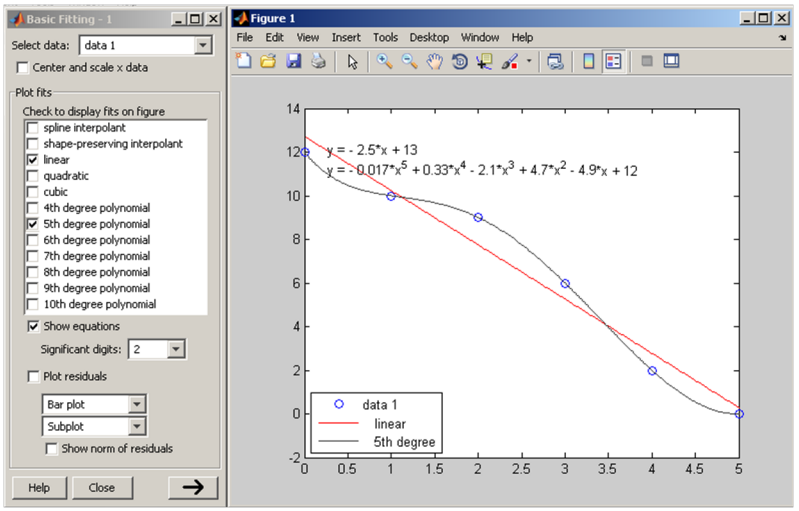
```
>> x=0:5;
>> y=[12,10,9,6,2,0];
>> plot(x,y,'o')
```

On the figure screen, go to the menu item Tools > Basic Fitting. This will

launch a new window:



Now you simply have to select the polynomial(s) that you want to fit (try a few!). You can even see the corresponding equations (and more) using the big arrow button . If we select the "linear" and "5th degree polynomial" check boxes, and the "Show equations" check box, we will see the equations on the graph itself.



www.dbooks.org

Chapter 9 Exercises

In all of the sections that require plots, make sure that your graphs are smooth, data is represented as points (with markers) and both titles and labels are added appropriately.

1. The volume (in m^3) and pressure (in kPa) of a gas at a constant temperature is recorded in the table as

Volume	**Pressure**
1	2494
2	1247
3	831
4	623
5	499
6	416

   a. Use the `polyfit` and `polyval` commands to find the first, second, third, and fourth-order approximating polynomials.

   b. Plot the data from the table along with the curves that you found in part **a**. Make sure the curves are smooth, the plot is well labeled and there is a legend describing the different curves.

2. Resistance (in ohms) and current (in amps) are related through the equation I=V/R. Data was collected from a circuit with unknown constant voltage and is shown in the table.

Resistance(R)	**Current(I)**
10	11.11
15	8.04
25	6.03
40	2.77
65	1.97
100	1.51

   a. Plot R (x-axis) versus I (y-axis) as data points. What kind of relationship do you see?

   b. Plot 1/R (x-axis) versus I (y-axis) as data points. Now, what kind of relationship do you see?

   c. Use `polyfit` to calculate the slope and intercept of the line in part **b**. What does the slope of the line represent?

   d. Use `interp1` to approximate the current when the resistance is 80 ohms.

   e. Create a new plot with the data points from part b, a line with slope and intercept from part c, and the interpolated point (with a different symbol) from part d. Label and title appropriately.

Problems continue on the next page...

3. Suppose that a population, P, grows over time, t, according to exponential growth with the following data.

Time(years)	**Population(thousands)**
1	1.61
2	2.51
3	2.17
4	6.14
5	6.81
6	17.53
7	16.36
8	25.04
9	39.61
10	55.50

We have that the population is therefore growing according to the equation $P = P_0 e^{kt}$. Taking a logarithm, we can convert this into $\ln(P) = \ln(P_0) + kt$ (i.e. $\ln(P)$ is linear in t).

   a. Plot the data with t on the horizontal axis and $\ln(P)$ on the vertical axis.

   b. Use the `polyfit` and `polyval` functions to find the slope and intercept and add the regression line to the plot in part **a**.

   c. Find the values of P_0 and k and, in a second figure, plot the original data along with the curve $P = P_0 e^{kt}$.

www.dbooks.org

Chapter 10

Mathcad: Introduction

In this chapter, we discuss the basics of Mathcad.

10.1 Mathcad: Introduction

Let's dive right in and take a look at Mathcad. The first thing you notice that the screen looks like a blank sheet. In fact, the vertical lines define the edges of how the document would be printed.

To enter information in a certain place, you can (left) click there and start typing (try it!).

Mathcad provides for a nice blend of text and mathematics, including equations, data and graphs. There is also a good balance between using toolbar buttons and the flexibility of entering commands, for those who like "command line" type languages.

In general, Mathcad is inexpensive, easy to learn & use and provides for readable documents. Mathcad handles units and unit conversions very well with a large, built-in list of units. There are many reference tables available including: Basic Science constants, Calculus Formula, Geometric tables, Mechanics, Electromagnetics, and Properties of Liquids, Solids, Gases & Metals. Mathcad has both the ability to combine numerical and symbolic capabilities.

As for Mathcad's weaknesses, programming is awkward. Also, while it can combine both the ability to do numerics and symbolics, there are packages that do these better individually, like Matlab for numerical calculations and Mathematica for symbolics.

HOMEWORK ASSIGNMENT RULE:

When you turn in your assignments, you must provide a header file. To create one, simply click near the top left corner of the page and start typing; you will be put into text mode as you type the first word and a space.

An example for the first homework may look like:

Name: Troy Siemers
Assignment: Mathcad Chapter 10 and 11
Course: MA110
Date: <fill in>
Description: In this file we investigate the overall layout of Mathcad and how the toolbars are used.

We refer you to the help files for additional information. There are helpful "Quick sheets" that give tutorials on many subjects.

www.dbooks.org

Chapter 11

Mathcad: Entering Equations

In this section, we learn about different uses of the equal sign, entering and editing equations, using the blue guidelines, formatting output, highlighting, alignment of equations, and working with units.

11.1 Mathcad: Equations

There are often many ways to access tools in Mathcad: through a menu at the top of the screen, a toolbar, a right click with the mouse, or a shortcut keystroke.

Entering Text

To enter simple text anywhere on the screen, click at the desired spot (a red plus sign will appear) and do one of the following:

- Type " (the region will become a text region). Then type your text.
- Use the menu: Insert > Text Region
- Start typing text (the region will become a text region after the first word).

Different uses of the equals sign

To enter equations in Mathcad, it is a bit trickier. There are in fact FOUR different ways to use an equals sign (these will be explained soon):

- The evaluation equals sign (=)
- The assignment equals sign (:=)
- The symbolic equals sign (**=**)
- The global equals definition (≡)

Evaluation Equals Sign (=)

Mathcad can be used as a simple calulator. To compute $1+1, 2^{10}$, or $5^{0.5}$, we type these, as one might expect, in the normal way and then type =.

$$1 + 1 = 2$$

$$2^{10} = 1.024 \times 10^{3}$$

$$5^{0.5} = 2.236$$

For a few more advanced computations, we can try $\cos(\pi), \sin(90°), \sqrt{5}$, and $\ln\left(\frac{1}{2}\right)$.

$$\cos(\pi) = -1$$

$$\sin(90\text{deg}) = 1$$

$$\sqrt{5} = 2.236$$

$$\ln\left(\frac{1}{2}\right) = -0.693$$

Creating the symbols in these calculations can be done in many ways! The trigonometric and natural logarithm are simply typed in as sin, cos and ln. The symbol for pi can be created from the Greek toolbar, or with the shortcut key ctrl+shift+p (or type p followed by ctrl+g). Note that the default is radian mode for the trig functions, but if you want degrees, type the word deg inside the trig function (just as it appears). To get a fraction, type 1, backslash and 2 (the format of the fraction and size of the parentheses are automatically done by Mathcad). To get the square root symbol, use the calculator toolbar (or shortcut key \).

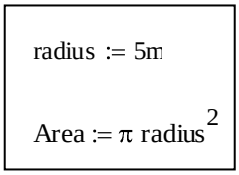
We especially encourage the reader to explore the toolbars, menus, keyboard shortcuts, mouse clicks, help files, etc. That's how we learned a lot of Mathcad's functionality. Also, if there is something that you can't figure out, try your favorite internet search engine. We do.

www.dbooks.org

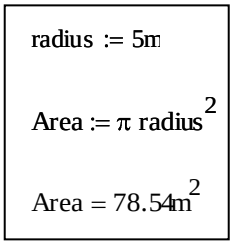
Assignment Equals Sign (:=)

It is often useful to assign values to variables that can be used later. This is done with the assignment equals sign, created not by using the = key, but by typing a colon (:).

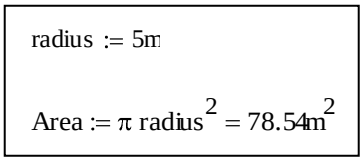
For example, suppose we wanted to find the area of a circle of radius 5 meters. We could of course do the calculation "pi r squared" but we will instead store the value $5m$ in the variable **radius** and then compute and store the area in the variable **Area**. Make sure to type a colon instead of = in order to get the symbol :=.



Note that with the assignment equals, you don't see the actual value of **Area**. To see the actual value, you need to either put in an additional line

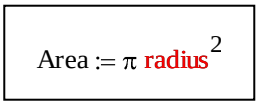


(where the second equation with the variable **Area** does use the = sign) or type an additional equals sign (=) at the end of the line defining **Area**

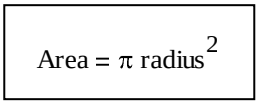


Symbolic Equals Sign (=)

The symbolic equals sign (an equals sign appearing in bold font) is used in setting up an equation without actually providing any values for the variables. For example, in computing the area of a circle, we may not necessarily know the value of the radius, but want to use the equation "A equals pi r squared." If we try to enter this with the assignment equals (without providing the radius value), we get an error (the variable **radius** is in red).



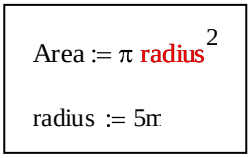
To fix this, we use the symbolic equals sign (a bold equals sign) by typing Ctrl + =.



One reason for using the symbolic equals is that you might want to later solve (symbolically) for the radius in terms of the area. We will show how to do this and give more reasons why you might want to use the symbolic equals sign in later sections.

Global Equals Definition (≡)

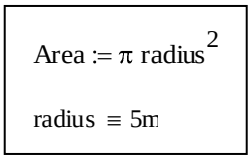
Consider the following example:



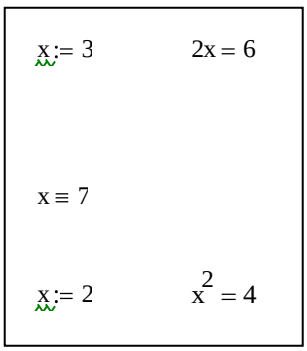
Why is the word **radius** in red? All of the parts of the computation are present, but for Mathcad, this is not enough. The order of the computations is important (as we will discuss further). Mathcad computations have the order "left to right, top to bottom" meaning that any variable used in a calculation must be defined previously (either higher up on the page, or to the left on the page). There is one exception to this rule - the global equals definition. The global equals definition is created using the tilde (∼) symbol. A variable that

www.dbooks.org

is defined with the global equals definition can be used in any other equation, regardless of its location on the page.



Also, if a variable is defined globally, that does not mean that it will override another definition of the same variable, for example consider



Here we see that even though x is globally defined to be 7, the top line computation of $2x$ is using x equal to 3 and the bottom line computation of x^2 is using x equal to 2.

We strongly suggest against using the global equals definition. Finding errors in equations can be difficult enough without having to worry about possible miscalculations due to a globally defined variable.

Key Idea 3

Variable Names

In deciding on variable names, consider the following:

- Variable names cannot begin with a number.
- Use descriptive names such as "Area" instead of simply A
- Try not to use variable names for predefined Mathcad units, like m (meter), c (speed of light), K (degrees Kelvin), etc. If you try to do so, the variable will be marked with a green squiggly line as a reminder.

11.2 Mathcad: Editing Equations

Once an equation is entered, editing it can be tricky. We recommend first reading the help file under

Contents > Getting Started > Entering and Evaluating Expressions > Editing an Expression.

Here we give a few pointers, but also suggest plenty practice and trial-and-error.

Blue Guidelines

As you type, you should notice a pair of blue lines appear and change size as you enter the expression, one as an underline and one as a vertical line. This pair of guidelines indicates the insertion point if you were to type something new. In order to move the blue guidelines to the desired spot you can try the following

- Using the **spacebar**

Each time the spacebar is depressed, the blue guidelines increase in length and enclose more of the equation for editing. Try it.

- Using the **arrow keys**

You can scroll through an equation using the left and right arrow keys. With the up and down arrow keys, you can move between exponents and subscripts.

www.dbooks.org

Give it a shot.

- Using the **insert key**

Using the insert key, you can move the vertical blue guideline to the opposite side of the horizontal blue guideline. Once, again, you'll figure it out by trying it.

Using the mouse

You can of course use the mouse to click within an equation or to move selected equation blocks.

To edit within an equation, click at the desired location in the equation. If you want to select a variable name, try double clicking on the variable name, or highlighting it with the usual click-and-drag.

To select multiple equations into a block that can be moved or aligned, click and hold at a location outside the equation and drag the dotted box that appears to include the desired equations. Once the mouse button is released, the selected equations can be moved by putting the mouse cursor over one of the selected boxes (a small hand should appear), then click and move the entire block to the desired location.

To align a selected block of equations, do one of the following:

- Use the menu: Format > Align Regions > (either Across or Down)
- Click on the align Across or align Down button (as pictured).



Other Customization of Equation Blocks

One of Mathcad's strengths is the ability of the user to make the work simply look nice. Combining text and equations along with graphs is easy to do. In addition, you can customize your equations in a number of ways

- Highlighting the equation

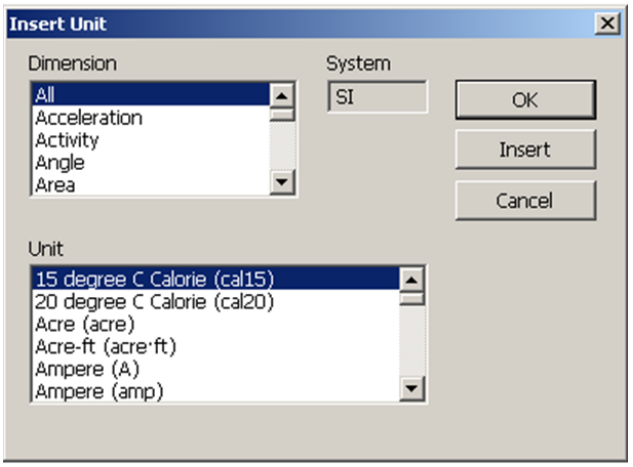
This is useful to make parts of the sheet stand out. If you right click on an equation, you can select Properties. From there, you can select "Highlight Region" (and select a color) or "Show Border" (around the equation).

- Formatting the result

Under the menu Format > Result, you can set the number of decimal places, change the "Number format" (Decimal, Scientific, Engineering, etc.), "Display Options" (useful for tables and matrices) and customize how units are shown in "Unit Display." You can also double click any evaluated equation to get this dialog box.

11.3 Mathcad: Units

A very powerful part of Mathcad comes from its ability to handle units and unit conversions. When entering an equation, you can access the complete set of units under the menu at Insert > Units



or if you know the name of the unit, you can just type it in:

$$\text{Volume} := 34\text{gal}$$

$$\text{Volume} = 128.704\text{L}$$

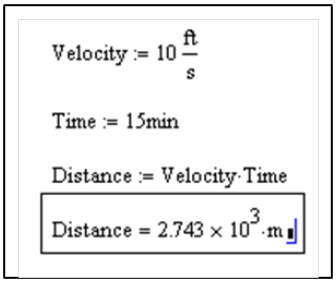
$$\text{Distance} := 10\text{ft}$$

$$\text{Distance} = 3.048\text{m}$$

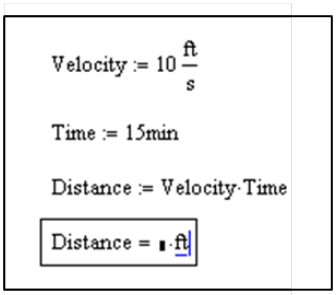
www.dbooks.org

In this example, we entered the first and third lines exactly as they look, using the colon to make the assignment equals sign := and typing the letters "gal" and "ft". In the second and fourth equations, we used the evaluation equals sign = to calculate. Note that Mathcad automatically calculates using metric units as a default.

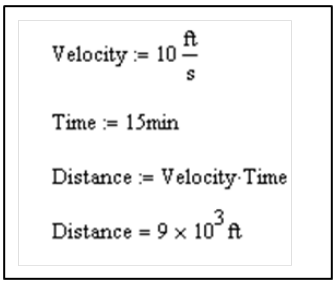
To keep an expression with a specific unit, consider the following example where we want the final answer in feet. We want to calculate the distance travelled by an object moving at 10 feet per second for 15 minutes. We enter the corresponding variables using the assignment equals in defining **Velocity**, **Time** and **Distance**. But, in the fourth line, when we use the evaluation equals, Mathcad automatically converts **Distance** to meters. To adjust this, first, click at the end of the line where **Distance** is computed. A small black box appears that we select...



... type ft in the box:



... and hit return:



Chapter 11 Exercises

1. For each part, define appropriate variables and complete the conversions.
   a. Gravity on earth from $9.80665\,m/s^2$ to ft/s^2.
   b. Speed of light from $299,792,458\,m/s$ to mph and also to miles per second.
   c. Length of the marathon from 26 miles, 385 yards into kilometers.
   d. KFC Original Recipe Double Down from 610 calories to Joules.
2. The volume of a truncated pyramid with a square base is given by $V = \frac{1}{3}(a^2 + ab + b^2)h$ where h is the height, a is the length of one of the sides of the base and b is the length of the sides of one of the top (also a square).
   a. Find the volume if $a = 5ft, b = 3ft, h = 10ft$ by first defining a, b and h as separate variables and then defining V in terms of them.
   b. In part a, the solution is probably in liters. Convert the answer to ft^3.
3. In this problem, you will use the quadratic equation.
   a. Using the bold equals sign (a symbolic equals), write out the general form of a quadratic equation $ax^2 + bx + c = 0$.
   b. Using the bold equals sign in each case, write out both possible solutions to the quadratic equation. Recall: $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.
   c. Define $a = 1, b = 6$ and $c = 5$ and actually compute both x values that are solutions to $ax^2 + bx + c = 0$.
   d. What are the solutions to the quadratic equation when $a = 1, b = 2$ and $c = 2$. Explain what's going on.
4. In this problem, you will see part of Cardano's formula for solving a cubic equation.
   a. Using the bold equals sign (a symbolic equals), write out the general form of a cubic equation $ax^3 + bx^2 + cx + d = 0$.
   b. The "discriminant" of the cubic equation in part a is defined as
   $$\Delta = 18abcd - 4b^3d + b^2c^2 - 4ac^3 - 27a^2d^2$$
   Using symbolic equals & Greek toolbar write this equation in your worksheet.
   c. Existence of solutions to the cubic equation will depend on the value of Δ in part b. In one particular case, it is important that Δ be negative. In the case that a, b, c and d all have value 1, show that Δ is negative. First define a through d and then write Δ in terms of them.
   d. When Δ is negative (like in part c), one solution is (all one line):
   $$x = -\frac{b}{3a} - \frac{1}{3a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d + \sqrt{(2b^3 - 9abc + 27a^2d)^2 - 4(b^2 - 3ac)^3}\right]} - \frac{1}{3a}\sqrt[3]{\frac{1}{2}\left[2b^3 - 9abc + 27a^2d - \sqrt{(2b^3 - 9abc + 27a^2d)^2 - 4(b^2 - 3ac)^3}\right]}$$
   In the case where a, b, c and d are all 1, compute this root. (Hint: Try using ctrl + Enter in the middle)

   E.C. What are the other two roots? (Hint: Look at the Mathcad "factor" function).

www.dbooks.org

Chapter 12

Mathcad: Given/Find and Solve

Here we investigate using "given/find" and "solve" blocks for solving equations or systems of equations.

12.1 Mathcad: Given/Find Blocks

given/find block

We illustrate this through the following example.

Example 41 **Solve** $2 = x^4 + 3x^2 - 1$**.**

In the worksheet, the syntax is

given
$2 = x^4 + 3x^2 - 1$ (Use ctrl + = for bold symbolic equals sign)
guess
x := 1
x := find(x)
x =

The last line will show x = 0.89 once the equals sign is entered. The line x := find(x) solves for x and then stores it back as the variable x. For full details, the guess of x := 1 provides the "seed" for the Newton-Raphson algorithm (Google it). Try changing the seed value to see what happens. For example, if we start with x := −1 we end up with a different answer, namely x = −0.89.

The given/find process can be extended to solving systems of equations as well as in the next example:

Example 42 **Simultaneously solve $a+b=3, a-b=4$ for a, b.**

In the worksheet, using ctrl + = for bold symbolic equals sign, the syntax is

given
a+b = 3
a-b = 4
guess
a := 1
b := 1
find(a,b)=

Once, you hit return on the last line, it will become

$$\text{find(a,b)=} \begin{pmatrix} 3.5 \\ -0.5 \end{pmatrix}$$

So, the solution is $a = 3.5, b = -0.5$.

The given/find capabilities are not limited to linear equations as can be seen in the next example.

Example 43 **Simultaneously solve $t^4 + r^3 = 1$ and $r - t^2 = -2$**

In the worksheet, using ctrl + = for bold symbolic equals sign, the syntax is

given
$t^4 + r^3 = 1$
$r - t^2 = -2$
guess
r := 1
t := 1
find(r,t)=

Once, you hit return on the last line, it will become

$$\text{find(r,t)=} \begin{pmatrix} -0.783 \\ 1.103 \end{pmatrix}$$

So, one solution is $r = -0.783, t = 1.103$. Note that this is not the only solution. Can you change the seed (or "guess") values to get the rest of the solutions?

www.dbooks.org

12.2 Mathcad: Solve Blocks

solve block

To use the solve block, first open the symbolic toolbar. We show how to solve the example 41 with this new technique.

Example 44 **Solve** $2 = x^4 + 3x^2 - 1$**.**

In the worksheet, using ctrl + = for the bold symbolic equals sign and clicking on the word "solve" in the symbolic toolbar, the syntax is

$2 \mathbf{=} x^4 + 3x^2 - 1 \text{ solve, } x \rightarrow$

Once you hit return, it will look like:

$$2 \mathbf{=} x^4 + 3x^2 - 1 \text{ solve, } x \rightarrow \begin{pmatrix} -\sqrt{-\frac{1}{2}\sqrt{21} - \frac{3}{2}} \\ \sqrt{-\frac{\sqrt{21}}{2} - \frac{3}{2}} \\ -\sqrt{-\frac{1}{2}\sqrt{21} - \frac{3}{2}} \\ \sqrt{\frac{\sqrt{21}}{2} - \frac{3}{2}} \end{pmatrix}$$

If you type an equals sign on the end of the last line, it will become:

$$2 \mathbf{=} x^4 + 3x^2 - 1 \text{ solve, } x \rightarrow \begin{pmatrix} -\sqrt{-\frac{1}{2}\sqrt{21} - \frac{3}{2}} \\ \sqrt{-\frac{\sqrt{21}}{2} - \frac{3}{2}} \\ -\sqrt{-\frac{1}{2}\sqrt{21} - \frac{3}{2}} \\ \sqrt{\frac{\sqrt{21}}{2} - \frac{3}{2}} \end{pmatrix} = \begin{pmatrix} -1.947i \\ 1.947i \\ -0.89 \\ 0.89 \end{pmatrix}$$

Note that two of these solutions are not real valued. Also, it is proper syntax to leave off the ", x" after the word solve as Mathcad will solve for the only variable present by default. That is,

$2 \mathbf{=} x^4 + 3x^2 - 1 \text{ solve} \rightarrow$

will still give the same solution.

The solve block can be extended to equations with multiple variables as seen in the next example.

Example 45 **Solve** $1 = \dfrac{x^2 y}{x^2 + y^2}$ **for each variable**

Using the bold equals and the word "solve" from the symbolic toolbar, the syntax is:

$$1 = \frac{x^2 y}{x^2 + y^2} \text{ solve, } x \rightarrow$$

Once you hit return, it will look like:

$$1 = \frac{x^2 y}{x^2 + y^2} \text{ solve, } x \rightarrow \begin{pmatrix} -\dfrac{y}{\sqrt{y-1}} \\ \dfrac{y}{\sqrt{y-1}} \end{pmatrix}$$

and

$$1 = \frac{x^2 y}{x^2 + y^2} \text{ solve, } y \rightarrow$$

Once you hit return, it will look like:

$$1 = \frac{x^2 y}{x^2 + y^2} \text{ solve, } y \rightarrow \begin{pmatrix} \dfrac{x(x + \sqrt{x^2 - 4})}{2} \\ \dfrac{x^2}{2} - \dfrac{x\sqrt{x^2 - 4}}{2} \end{pmatrix}$$

www.dbooks.org

Chapter 12 Exercises

1. In this exercise, we apply both given/find and solve techniques to a quadratic equation.
   a. Use the solve command to find the solutions to $x^2 + 5x + 6 = 0$.
   b. Create a given/find setup for the equation $x^2 + 5x + 6 = 0$ similar to the example at the beginning of this section. Find the value of x using the different the seed values $10, -5$ and -2.5. What is special about the seed value -2.5?
2. The volume of a truncated pyramid with a square base is given by $V = \frac{1}{3}(a^2 + ab + b^2)h$ where h is the height, a is the length of one of the sides of the base and b is the length of the sides of one of the top (also a square).
   a. Use the solve command to isolate the variable b in terms of V, a and h.
   b. If $b = 1ft, h = 10ft$ and $V = 100ft^3$, use a given/find block (and a guess for the value of a) to find the value of a. Make sure your answer is in feet.
3. In the exercises for chapter 2 there is an exercise dealing with the theory of splines in which you are to find values S_1, S_2, S_3, S_4 and S_5. Using the equations from that exercise, set up a given/find block to solve for the S values if the initial guesses are $S_1 = S_2 = -100$ and $S_3 = S_4 = S_5 = 0$.
4. Using a given/find block to find the points of intersection between the circle of radius 2 centered at the origin and the line of slope 1 and y-intercept 1. You will have to adjust your initial guesses to get all of the solutions.

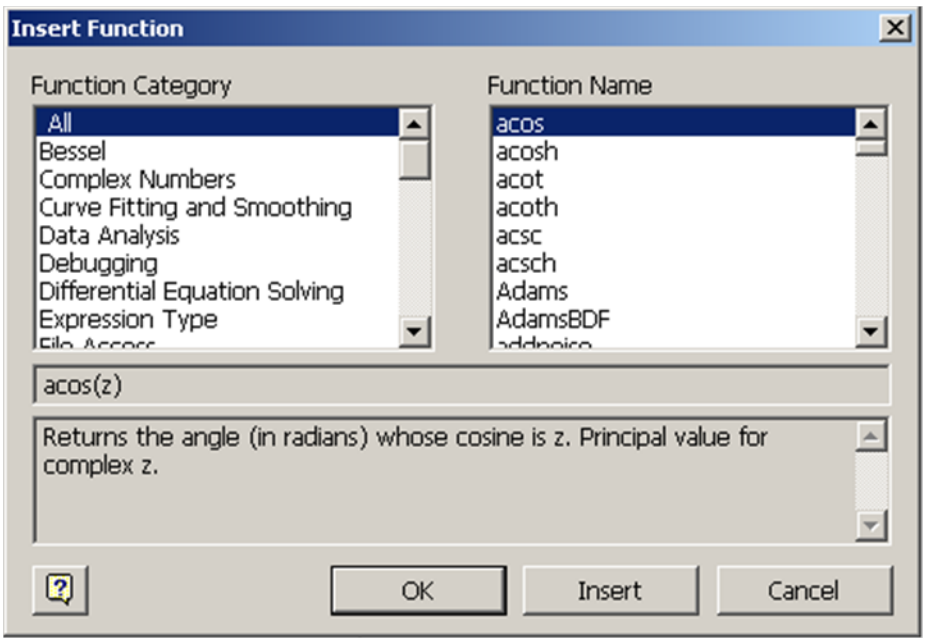
Chapter 13

Mathcad: Functions

In this section, we learn about functions. Mathcad has a large library of built-in functions (logarithms, trigonometric functions, statistical functions, etc), but we can create new functions as well.

13.1 Mathcad: Built-in Functions

To access the complete list of built-in functions in Mathcad, use the menu with Insert > Function or type Ctrl + e.



www.dbooks.org

Selecting a "Function Category" (left side of the dialog box) will restrict the list under "Function Name" (right side of the dialog box) for easier searching. A description of the highlighted function's output will appear in the lower part of the dialog box.

A few special functions that one might use often:

- **Square root**

The square root function can be created either using the Calculator toolbar or by simply typing \. To create an nth root, also in the Calculator toolbar, the shortcut key is Ctrl + \.

- **Absolute value**

Both bars in the total absolute value symbol are created at the same time when selected from the Calculator toolbar (the |x| button), or by typing the vertical bar | (located on your keyboard above the Enter key - it looks like two stacked bars, but will create one solid vertical bar).

Key Idea 4

Absolute Value versus Matrix Determinant
If you already know about determinant of matrices:
The determinant and the absolute value look exactly the same on the page (both are |x|) and both are created using the same keyboard shortcut key (the vertical bar |)!
If you have an error using the vertical bars, it could be that you are either trying to take the absolute value of a matrix (not allowed), or the determinant of a number (not allowed either). You have to be very careful as to the context of your problem in using these two functions!

13.2 Mathcad: User-Defined Functions

User-Defined Functions

Sometimes you will need to create your own function in Mathcad, perhaps for repeated use throughout a worksheet. The syntax looks very similar to the way a function might appear in a math textbook. The function name is followed immediately by the input variables (in parentheses and separated by commas), a definition equals sign (colon), and then the expression in terms of the input

variables. When the function is executed, the input variables are replaced with constants, including units as desired, and followed by the evaluation equals sign. Consider the following example.

Example 46 Create a function for the volume of a cylinder

The function, called **Volume**, is defined in terms of the variables **radius** and **height** (don't forget to use the assignment equals, created with a colon :).

Volume(radius,height):=π radius^2 height

The symbol for π can be created on the Greek toolbar, or with the keyboard shortcut Ctrl + Shift + p. Now, when we want to compute with the function, we substitute values for radius and height (including any unit) into the function

Volume(2ft,3ft):= 1.068×10^3 L

or

Volume(500cm,2000cm):= 1.571×10^6 L

Note that Mathcad converts to the metric unit L (liters) as default.

www.dbooks.org

Chapter 13 Exercises

1. The volume of a truncated pyramid with a square base is given by

$$V = \frac{1}{3}(a^2 + ab + b^2)h$$

where h is the height, a is the length of one of the sides of the base and b is the length of the sides of one of the top (also a square).

   a. Create a function in Mathcad for the volume in terms of a, b and h.

   b. Use the functon in part a to compute the volume for the values $a = 5ft, b = 3ft, h = 10ft$. The answer should be in corresponding cubic units.

   c. Use the functon in part a to compute the volume for the values $a = 1m, b = 0.5m, h = 2m$. The answer should be in corresponding cubic units.

2. A uniform beam is freely hinged at its ends $x = 0$ and $x = L$, so that the ends are at the same level. It carries a uniformly distributed load of W per unit length and there is a tension T along the x-axis. The deflection y of the beam a distance x from on end is given by

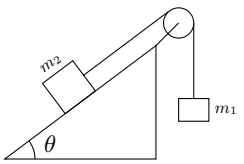
$$y = \frac{W \cdot EI}{T^2}\left[\frac{\cosh\left[a(L/2 - x)\right]}{\cosh(a\,L/2)} - 1\right] + \frac{Wx(L - x)}{2T}$$

where $a^2 = T/EI$, E is Young's Modulus of the beam and I is the moment of inertia of a cross-section of a beam.

   a. Create a function y in terms of all of these variables.

   b. Use the function from part a to compute y if the beam is $10m$ long, the tension is $1000N$, the load $100N/m$ and EI is 10^4, and x is $5m$. Note that $\cosh$ may have trouble with units, but make sure to give the units of your answer.

3. Two blocks of masses m_1 and m_2 are connected by a string over a smooth pulley as shown. Assume that $m_2 > m_1$. If the coefficient of friction is μ and $r = m_1/m_2$, then the masses will move at a constant speed (m_2 slides down the slope) if the angle θ is given by $\cos(\theta) = \dfrac{-\mu r + \sqrt{1 - r^2 + \mu^2}}{1 + \mu^2}$.



   a. Create a function for θ in terms of the other variables.

   b. Use the function in part a. to find θ if $m_1 = 5kg, m_2 = 10kg$, and $\mu = 0.6$.

   c. Use a solve block to find μ if $m_1 = 5kg, m_2 = 10kg$, and $\theta = 30^\circ$.

Chapter 14

Mathcad: Matrices

In this section, we learn about matrices. We discuss how to input matrices, alter existing matrices, and use them in computations. We use such functions as inverse, determinant, and transpose. We show how to select specific rows, columns or entries in a matrix and how to use matrices in solving systems of equations.

14.1 Mathcad: Matrix Definition

We begin with showing how to input a matrix into a Mathcad worksheet.

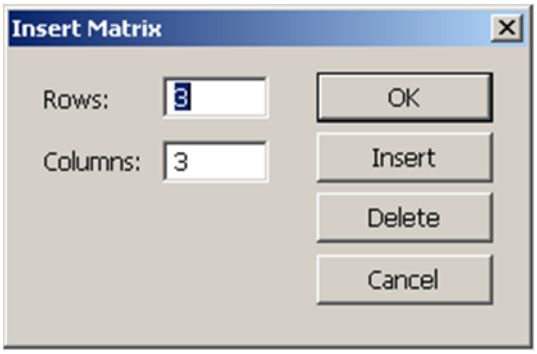
Entering matrices and basic operations

First, open the matrix toolbar with File > Toolbars > Matrix:



and select the  button in the upper left corner to launch the "Insert Matrix" dialog box:

www.dbooks.org



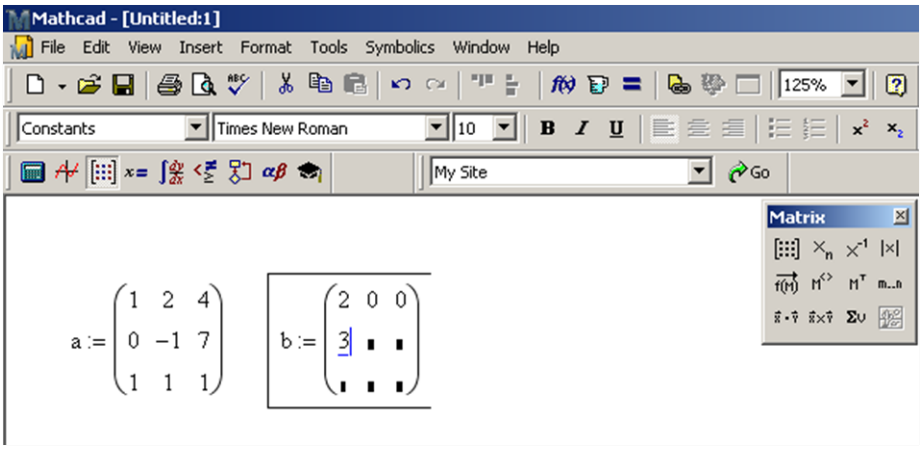
Input the desired number of rows and columns and click on "Insert". This creates a template of the correct size that you can use for your matrix. Note that as you enter numbers in the black boxes, you can move to the next black box with the Tab key, or between the entries with the arrow keys.

Let's try an example.

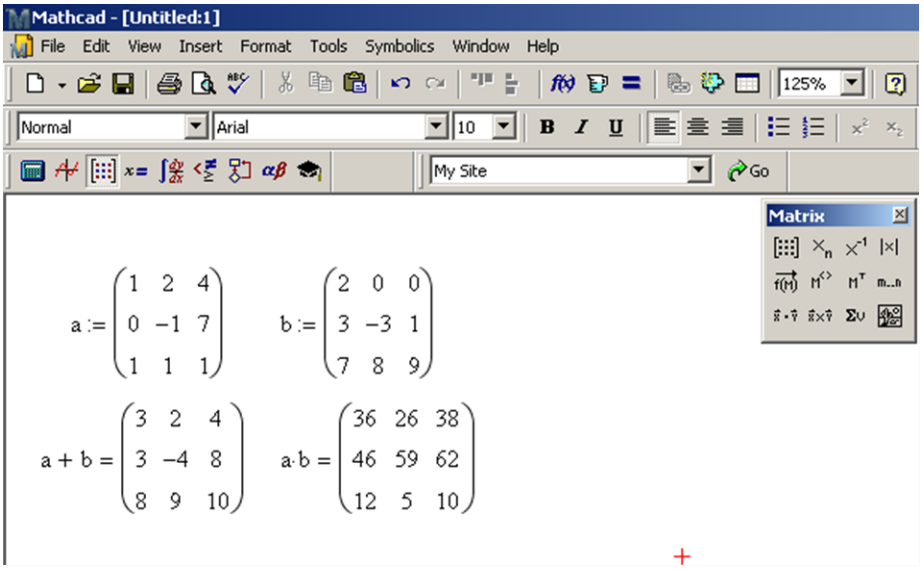
Find the sum and the product of the matrices

$$\begin{bmatrix} 1 & 2 & 4 \\ 0 & -1 & 7 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 2 & 0 & 0 \\ 3 & -3 & 1 \\ 7 & 8 & 9 \end{bmatrix}$$

We input the matrices and store them as variables **a** and **b**. Here we see them partially filled in:



The sum and product of the resulting matrices are done using the standard plus $+$ and times $*$ (which appears as a dot):

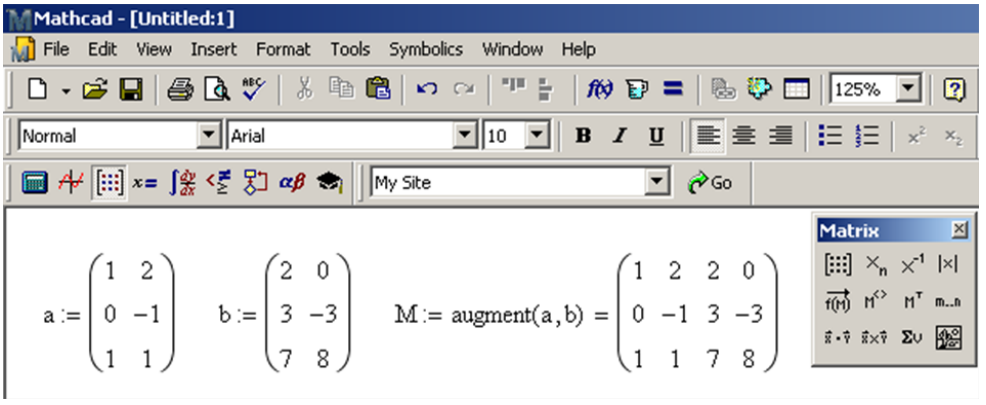


14.2 Mathcad: Editing Matrices

There are many different ways to alter an existing matrix. In these examples, note that we don't use the variable **c** since that is the built-in variable for the speed of light (although we could redefine **c**, there is no need to do so and it is generally a good idea not to overwrite the built-in variables).

- **Augmenting a matrix by another matrix** (side-by-side)

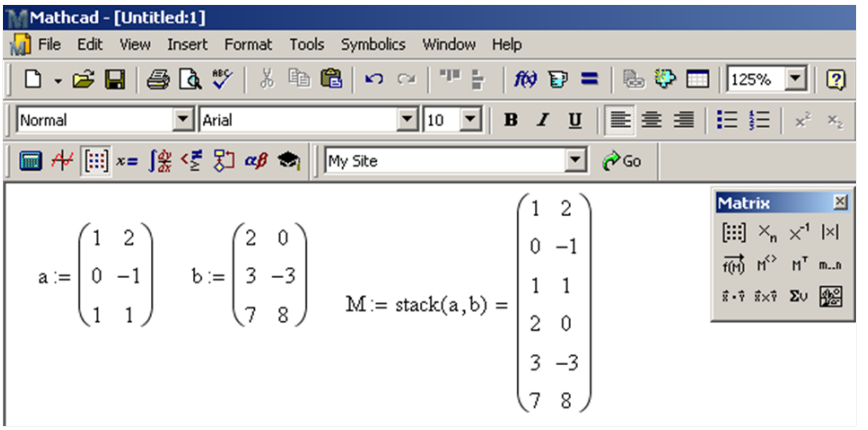
If two matrices have the same number of rows, we can create a new matrix with the command **augment**:



www.dbooks.org

• **Stacking two matrices**

If two matrices have the same number of columns, we can create a new matrix with the command **stack**:



• **Inserting a row or column**

To insert new row(s) or column(s) to an existing matrix, select an entry of the matrix. The inserted row(s) or column(s) will appear below or to the right of the selected entry. Then click the  button. Change the entries for the desired number of rows and columns and click "Insert". If you only want to insert a single row, set the number of columns to zero. If you only want to insert a single column, set the number of rows to zero.

Before inserting a column $a := \begin{pmatrix} 1 & 2 \\ 0 & -1 \\ 1 & 1 \end{pmatrix}$ and after $a := \begin{pmatrix} 1 & & 2 \\ 0 & & -1 \\ 1 & & 1 \end{pmatrix}$

New rows are inserted below a highlighted row and columns are inserted to the right of a highlighted row. If you want to insert a new first row or column, you need to select the whole matrix before using the  button and making the insertion.

• **Deleting a row or column**

The process of deleting rows or columns is similar to making insertions. To delete rows or columns from an existing matrix, select an entry of the matrix.

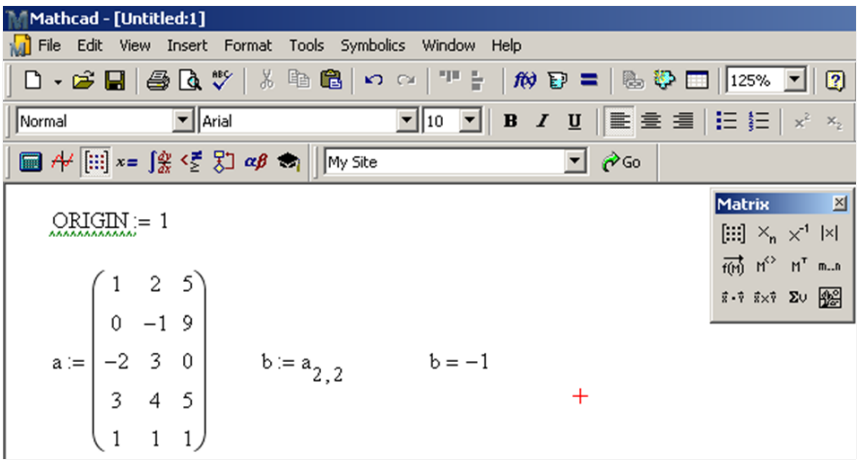
The deleted rows) or columns will contain that entry and delete rows below and columns to the right. Then click the  button. Change the entries for the desired number of rows and columns and click “Delete”.

14.3 Mathcad: Referencing Parts of Matrices

Sometimes you may need to use part of a matrix: an entry, a row, a column, or a submatrix inside another matrix. You must be very careful in referencing parts of a matrix, since the default for the starting index is zero!! This means that the “first” row of a matrix is referenced as if it was “row zero”. The start index is stored in a variable called **ORIGIN** and can be changed for a worksheet in two ways, either by using Tools > Worksheet Options (and changing **ORIGIN**) or by typing ORIGIN:=1 at the top of the worksheet.

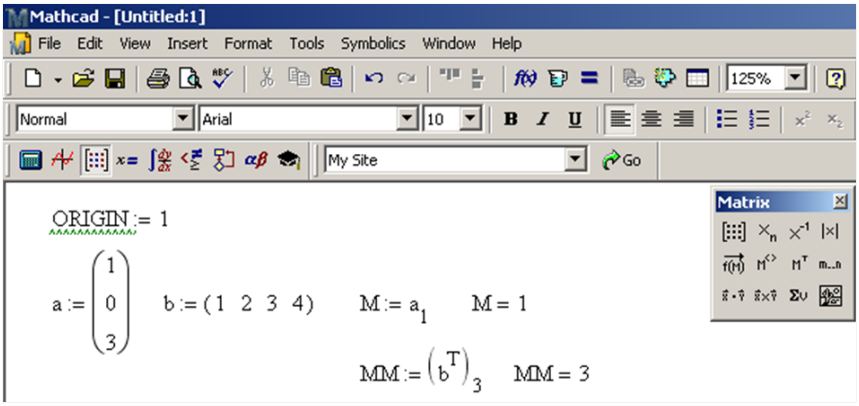
- **Referencing one entry in a matrix**

To select a single number out of an existing matrix, one can use the “subscript” operator from the matrix toolbar (the $\mathbf{x_n}$ button), or with the [key. Here we select the entry in the 2nd row and 2nd column by resetting the **ORIGIN** to 1 and defining **b** from the matrix **a**.



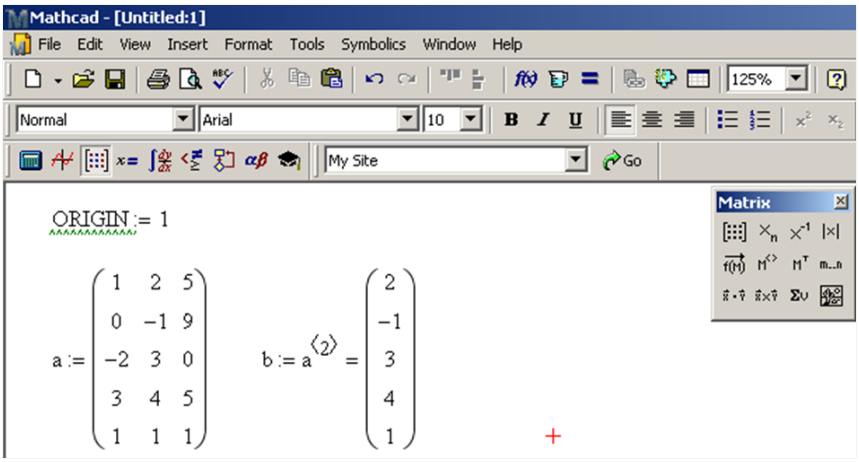
Note that if a matrix has only one column (a “column vector”) or a single row (a “row vector”) then only one subscript is necessary.

www.dbooks.org



- **To reference one column in a matrix**

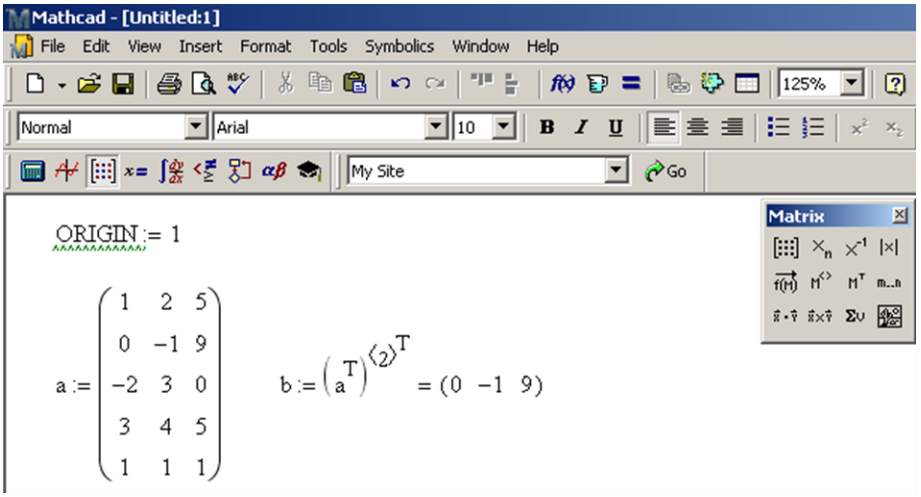
To select a single column out of an existing matrix, one can use the "Column" operator from the matrix toolbar (the $\mathbf{M}^{\langle\rangle}$ button), or with the keyboard shortcut Ctrl +6. Here we select the 2nd column by resetting the **ORIGIN** to 1 and defining **b** from the matrix **a**.



- **To reference one row in a matrix**

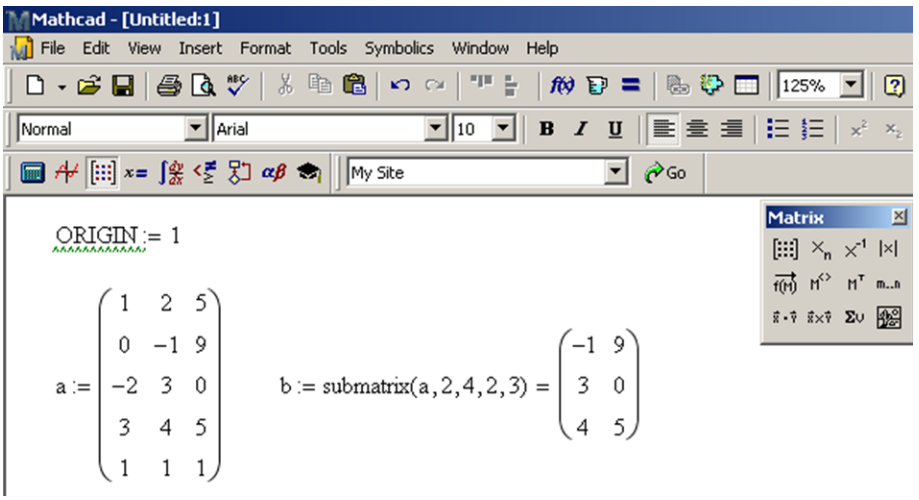
It is trickier to select a single row out of an existing matrix. One must use a combination of the "Transpose" operator (that switches rows and columns) together with the "Column" operator. The transpose operation on the matrix toolbar is the $\mathbf{M}^{\mathbf{T}}$ button and can also be created with the keyboard shortcut Ctrl +1. Here we select the 2nd row by resetting the **ORIGIN** to 1 and defining **b** from the matrix **a**. Typing this is a bit tricky. The exact sequence of keys is

$$\mathbf{b} \quad : \quad \mathbf{a} \quad (\text{Ctrl}+1) \quad (\text{Ctrl}+6) \quad 2 \quad (\text{spacebar}) \quad (\text{Ctrl}+1) \quad =$$



- **To reference a submatrix**

Selecting a submatrix from an existing matrix is done only through the command **submatrix**. Here we select the submatrix using the 2nd through 4th rows and 2nd through 3rd columns from the matrix **a**.



14.4 Mathcad: Solving Systems of Linear Equations

Suppose we have the following system of linear equations.

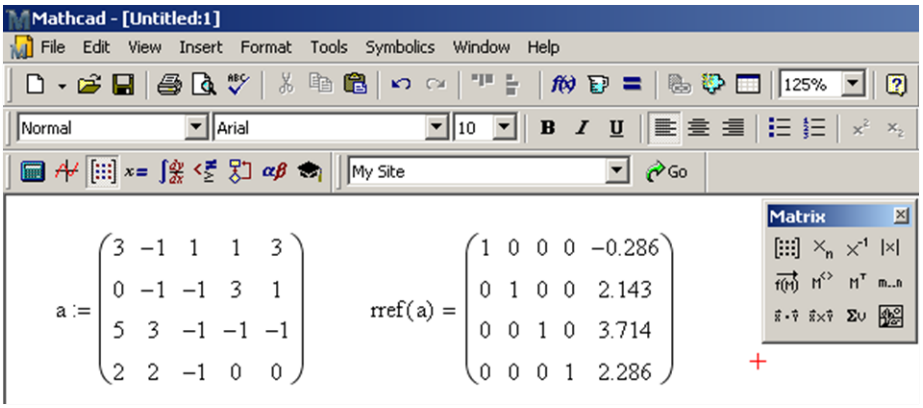
www.dbooks.org

$$\begin{aligned} 3w - x + y + z &= 3 \\ -x - y + 3z &= 1 \\ 5w + 3x - y - z &= -1 \\ 2w + 2x - y &= 0 \end{aligned}$$

The goal is to solve them simultaneously for the variables w, x, y and z. Assuming that there is exactly one answer (i.e. one quadruple for w, x, y, z) then there are many ways to do this. Here we use rref, lsolve and the matrix inverse as examples.

- **rref**

The command rref is short for "reduced row echelon form." Through a process called Gaussian elimination the matrix of coefficients (on w, x, y, z) augmented by the matrix of constants (the numbers on the right of the equals signs) is reduced to a matrix of zeros and ones (the "identity matrix") together with the solution (in the last column). In the matrix **a**, we store the cofficients for w in the first column, x in the second column, y in the third column, z in the fourth column and the constants in the fifth column. If a variable does not occur in an equation, a zero is in the corresponding entry of the matrix **a**.

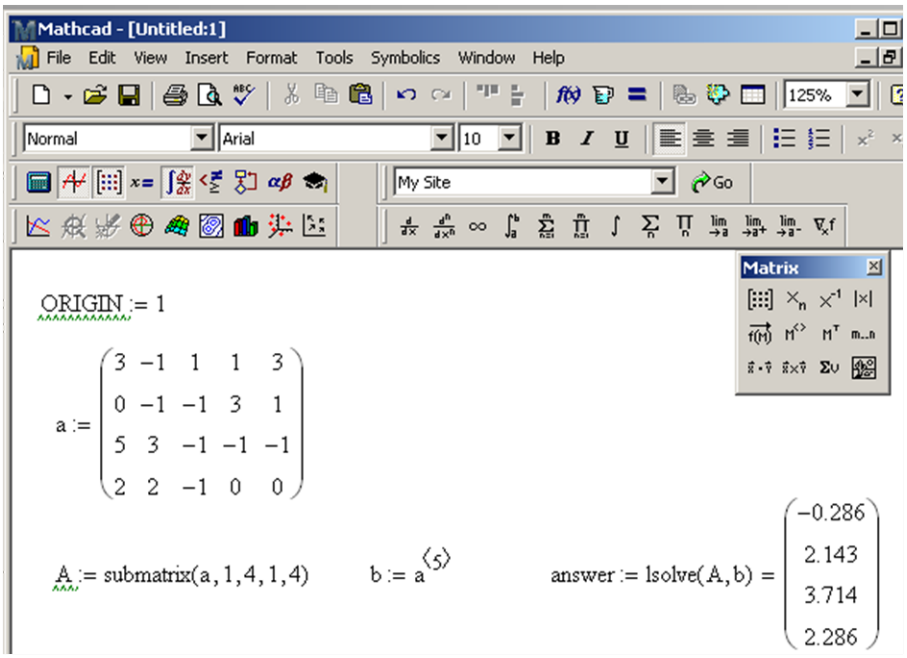


$$a := \begin{pmatrix} 3 & -1 & 1 & 1 & 3 \\ 0 & -1 & -1 & 3 & 1 \\ 5 & 3 & -1 & -1 & -1 \\ 2 & 2 & -1 & 0 & 0 \end{pmatrix} \qquad \text{rref}(a) = \begin{pmatrix} 1 & 0 & 0 & 0 & -0.286 \\ 0 & 1 & 0 & 0 & 2.143 \\ 0 & 0 & 1 & 0 & 3.714 \\ 0 & 0 & 0 & 1 & 2.286 \end{pmatrix}$$

The last column indicates that the only solution to the system of equations is $w = -0.286, x = 2.143, y = 3.714, z = 2.286$ to three place decimal accuracy. Recall, for more decimal places, select Format > Result or simply double click the equation.

• **lsolve**

To use the command **lsolve**, we need two separate matrices, one with the coefficients of w, x, y, z and one with the constants to the right side of the equals sign. We could input this from scratch or practice referencing them from the matrix **a** above. Here is the syntax for how to use the **lsolve** command.



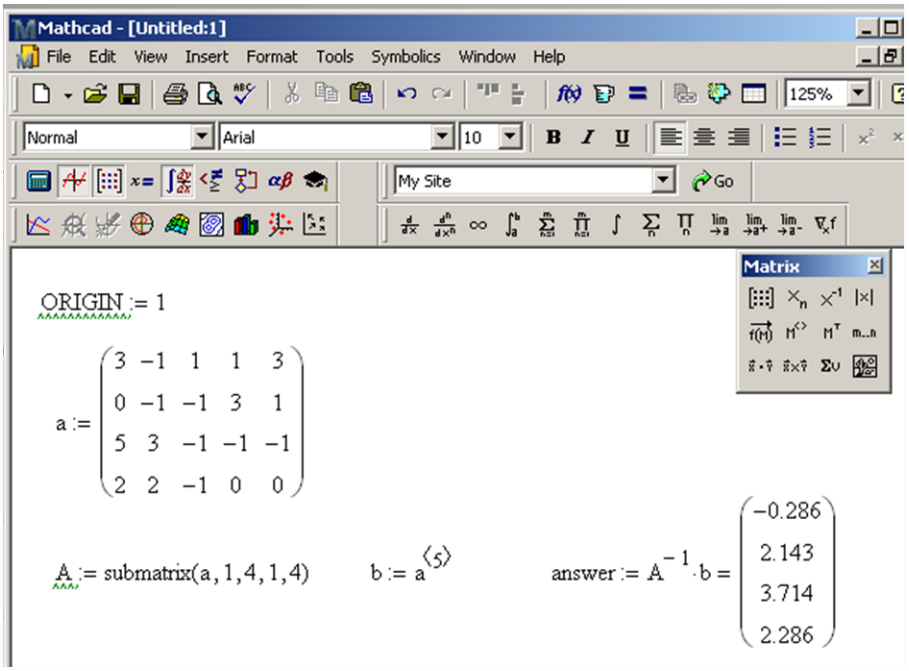
The variable answer contains the only solution to the system of equations as $w = -0.286, x = 2.143, y = 3.714, z = 2.286$.

• **Using the matrix inverse**

The setup for the using the inverse of a matrix is similar as for using **lsolve**. We need the coefficient matrix of w, x, y, z and the constant matrix. The inverse of a matrix is found by typing the name of the matrix and then selecting the $\mathbf{x^{-1}}$ button. When entering the equation for the variable **answer**, make sure that **b** is multiplying correctly. Here is the sequence of commands:

answer : **A** x^{-1} (spacebar) (spacebar) **b** =

www.dbooks.org



The variable **answer** contains the only solution to the system of equations as $w = -0.286, x = 2.143, y = 3.714, z = 2.286$.

You might be wondering "if there are several methods to find the solution, then which technique should you use?" The answer is (of course) more complex than we will go into here, but leave it to say that some techniques are faster or more accurate than others. In fact if a system is either "non-square" (has a different number of equations and variables) or the coefficient matrix is "singular" (look it up) then the **lsolve** and inverse techniques don't work and in this case the system of equations either has no solution ("inconsistent") or has infinitely many solutions ("underdetermined"). Ok, enough theory.

Chapter 14 Exercises

1. Consider the following matrices.

$$H_5 = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad A = \begin{bmatrix} 1 & -1 \\ 0 & 2 \\ 3 & 5 \\ 10 & 4 \end{bmatrix}$$

   a. Find $det(H_5), det(M_3)$.

   b. Find H_5^2

   c. Find H_5^{-1}, M_3^{-1}.

   d. Augment A with itself to form a new matrix (called B) and then find $det(B), B^2$ and B^{-1}.

   e. Find M_3^2, M_3^3, M_3^4 and M_3^5. What is the pattern you see? What is $M_3^{20112011}$?

2. A square matrix A is called **nilpotent** if some power of the matrix is equal to the zero matrix, i.e. $A^n = 0$ for some integer n. The smallest such n is called the **degree** of A. Consider the matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

   a. Show that A is nilpotent, and find its degree.

   b. The eigenvalues of any nilpotent matrix are all zero. Confirm this by finding the eigenvalues of A.

   c. Find a matrix that is nilpotent with order 10.

   E.C. Give (or describe) a matrix that is nilpotent with order 2011.

3. The L_2 condition number of a matrix is given by the 2-norm of the matrix times the 2-norm of the inverse of the matrix (written $||A||_2 \cdot ||A^{-1}||_2$)

   a. Find the condition number of H_5 from problem 1. Look in the help files for the correct command.

   b. Find a value of x so that the following matrix C has L_2 condition number as close to 100 as possible. Guess and check.

$$C = \begin{bmatrix} x & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & x \end{bmatrix}$$

www.dbooks.org

4. In computing an approximating function for a set of data, a spline is often used. The theory of splines is covered in numerical analysis, but for a specific data set, the following system of equations must be solved.

$$\left\{\begin{array}{rcrcrcrcrcl} 0.28S_1 & + & 0.1S_2 & & & & & & & & = -64.65 \\ 0.1S_1 & + & 0.34S_2 & + & 0.07S_3 & & & & & & = -54.81 \\ & & 0.07S_2 & + & 2.16S_3 & + & 1.01S_4 & & & & = -8.43 \\ & & & & 1.01S_3 & + & 2.58S_4 & + & 0.28S_5 & & = -7.92 \\ & & & & & & 0.28S_4 & + & 1.42S_5 & & = -2.78 \end{array}\right\}$$

Find the values for S_1 through S_5 by solving this system in three ways:

a. Using the command rref.

b. Using the command lsolve.

c. Using the matrix inverse.

Chapter 15

Mathcad: Graphing

Here we investigate the graphing capabilities of Mathcad.

15.1 Mathcad: Graphing

Graphing is a great way for interpreting technical and scientific data. Two areas will be discussed: 1) graphical presentation of data, 2) graphical analysis. Mathcad provides several graphing options.

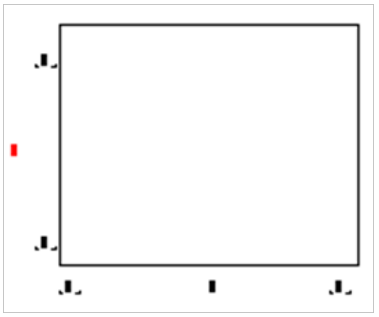
The easiest way to begin a plot is to use the graphing toolbox.
OR
Use the menu options Insert/Graph/X-Y Plot
OR
Shortcut [Shift + 2]

The toolbar looks like



and clicking on the first button creates a blank plot

www.dbooks.org

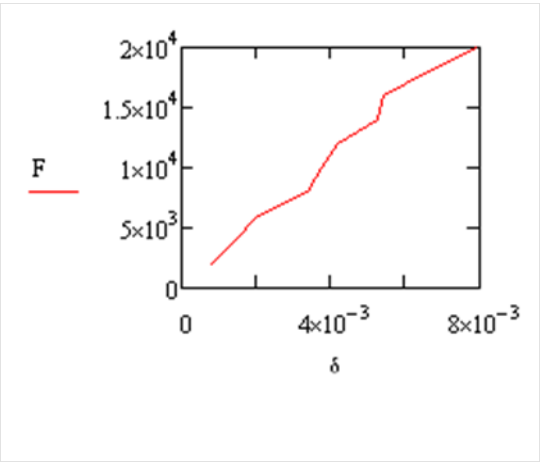


There are six place holders in the empty graph. The centered place holders are used for x-coordinate and y-coordinate data and the other four are for the x and y bounds.

Example 47

An engineer is testing a new material to find its modulus of elasticity, defined as the stress over the strain (or N/m). Plot the data given F vs. δ.

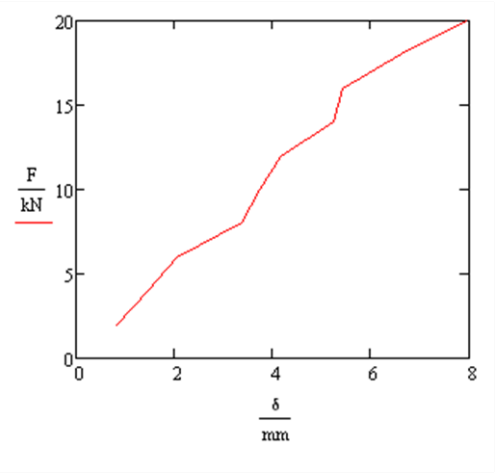
$$F := \begin{pmatrix} 2 \\ 4 \\ 6 \\ 8 \\ 10 \\ 12 \\ 14 \\ 16 \\ 18 \\ 20 \end{pmatrix} \cdot \text{kN} \qquad \delta := \begin{pmatrix} 0.82 \\ 1.47 \\ 2.05 \\ 3.37 \\ 3.75 \\ 4.17 \\ 5.25 \\ 5.44 \\ 6.62 \\ 7.97 \end{pmatrix} \cdot \text{mm}$$



If you wish to make any plot bigger, click on the plot. Grab the bottom right hand handle with the mouse and drag it to make it larger.

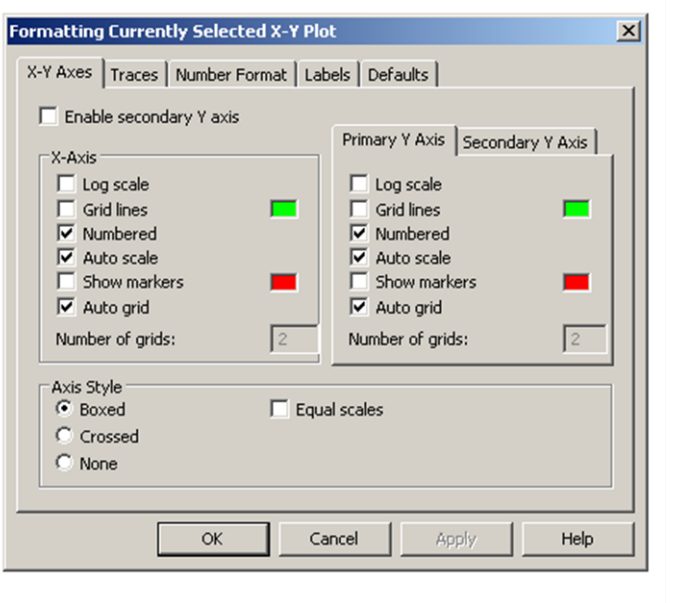
If you want to reposition a plot, click on the plot, point to the outside edge of the area until you see the hand, hold the mouse button down and drag the plot to a new location.

Note: Mathcad always plots in base units. So the plot in example 47 is in Newtons for the F and in meters for the δ. Here we show the enlarged plot with the units removed. As you see the plot now shows the values in the above tables with a unit on the axis. The center place holders have the data value divided by the unit F/kN and δ/mm.

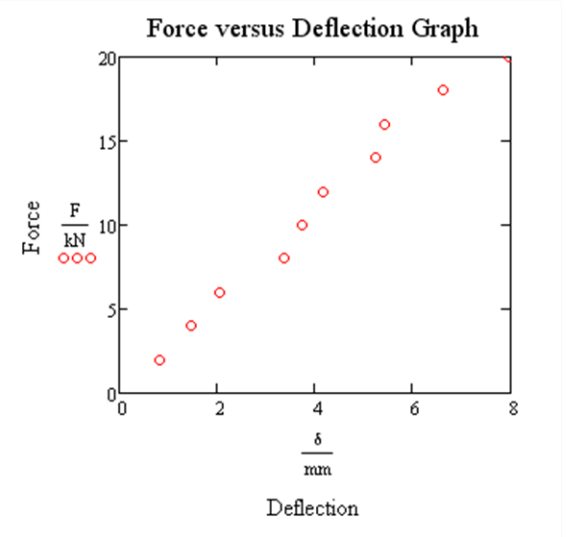


However, in example 47 suppose we really want to see just the data points, not the connecting lines. To do this, double click the graph to bring up the formatting dialog box.

www.dbooks.org



Go to the TRACES tab to add a symbol and remove the line. As you can see from the other tabs in this dialog box, you make many changes to the way the graph is formatted. A possible, reformatted graph appears below.



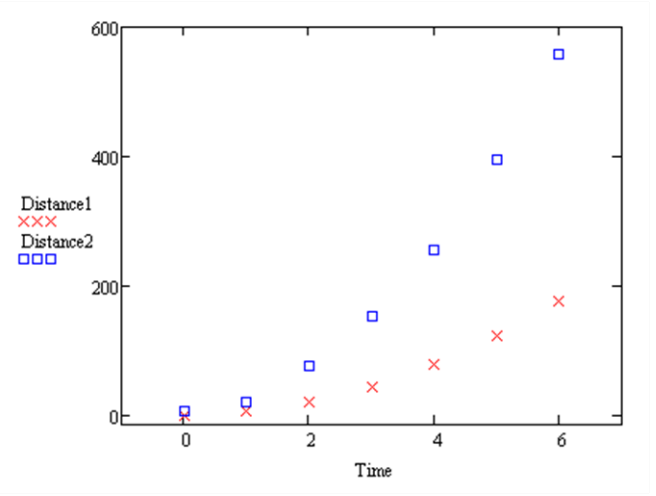
How about plotting multiple curves on the same plot? We demonstrate this in the following example.

Example 48

Begin by entering the data for the variables Time, Distance1 and Distance2.

$$\text{Time} := \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \quad \text{Distance1} := \begin{pmatrix} 0 \\ 4.9 \\ 19.6 \\ 44.1 \\ 78.4 \\ 122.5 \\ 176.4 \end{pmatrix} \quad \text{Distance2} := \begin{pmatrix} 5 \\ 19.8 \\ 76.8 \\ 153.3 \\ 256.2 \\ 394.5 \\ 559.2 \end{pmatrix}$$

Open a new plot and put Time in the x-axis. For the y-axis, enter Distance1 and then type a comma (important!!). This will move you to a new line where you can type Distance2. When you are done, click outside the graph to see the plot. To customize the plot, double click the plot and use the Formatting dialog box. See if you can match the plot below:

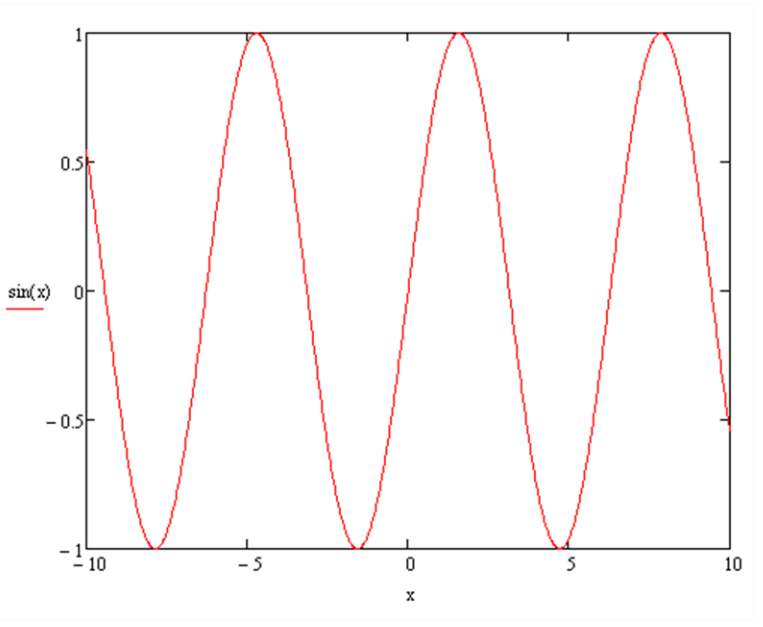


If you want to see a plot of a predefined function, you can use a **Quickplot**.

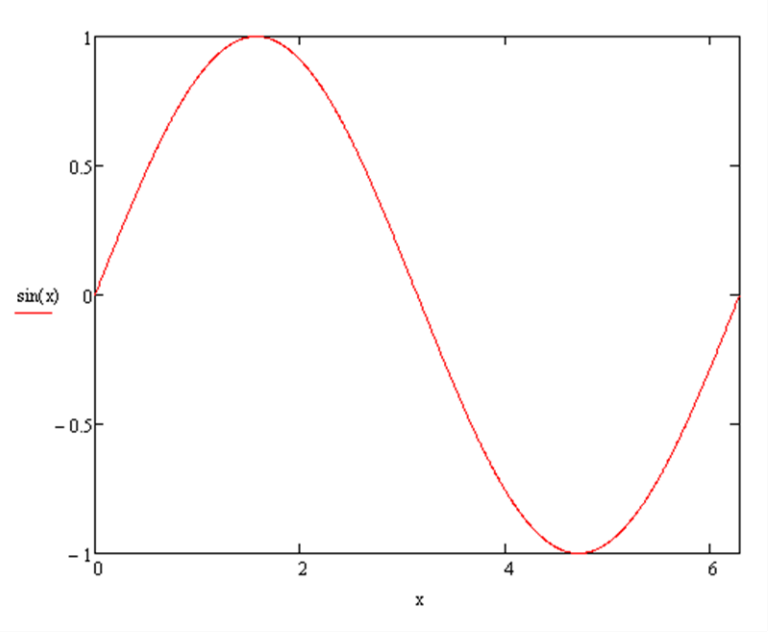
Example 49

Simply open a new plot, enter the variable x on the x-axis and a function, say sin(x) on the y-axis. The default plot has x from -10 to 10.

www.dbooks.org



In example 49 if you want to change the x values, first single click on the plot. The numbers -10 and 10 appear at the bottom. One at a time simply click on them and type in the new values 0 and 2π (using the Greek toolbar).



Now for a more advanced example...

Example 50

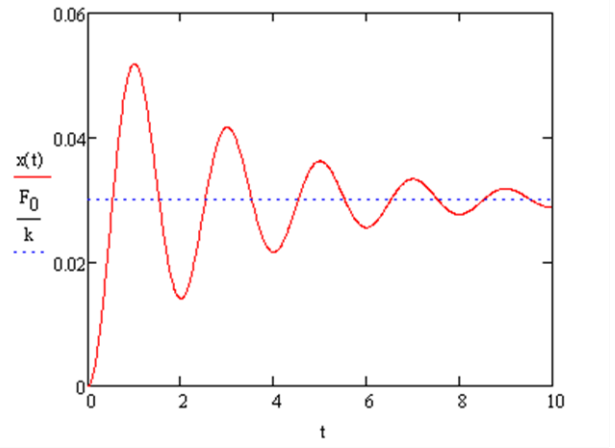
Here is the vibration response of a damped signal of freedom system to a step input.

$$F_0 := 30 \quad k := 1000 \qquad t_0 := 0 \qquad \zeta := 0.1 \quad \omega_n := 3.16$$

$$\omega_d := \omega_n \cdot \sqrt{1 - \zeta^2} \qquad mass := \frac{k}{(\omega_n)^2}$$

$$x(t) := \frac{F_0}{mass \cdot \omega_d} e^{-\zeta \cdot \omega_n \cdot t} \cdot \left[\int_{t_0}^{t} e^{\zeta \cdot \omega_n \cdot \tau} \cdot \sin\left[\omega_d \cdot (t - \tau)\right] d\tau \right]$$

with resulting plot:



The last example may be difficult to reproduce. Mastering how to input all of these parts into Mathcad takes time. Be patient.

www.dbooks.org

Chapter 15 Exercises

1. The double logistic curve is defined by

$$y1(x) = \text{sign}(x-5) \cdot \left(1 - \exp\left[-\left(\frac{x-5}{50}\right)^2\right]\right)$$

and a scaled logistic curve is defined by

$$y2(x) = \frac{5}{3 + 3\exp\left(-\frac{x}{10}\right)} - \frac{5}{6}$$

   a. Plot $y1(x)$ and $y2(x)$ on the same graph (adjusting x from -100 to 100). Format $y1(x)$ as a thick dotted line and $y2(x)$ as a thick dashed curve. Add an appropriate title.

   b. Use the trace functionality (right click on the graph) to find when each plot reaches a y value of $\frac{1}{2}$. Write a sentence below the plot giving this information.

   E.C. Use solve or given/find blocks to check your answers to part b.

2. Enter the Analemma data from the exercise in chapter 4 into a table. Plot the table using the matrix column button to plot one column on the x-axis and one on the y-axis. Make sure to adjust the data so the plot is a closed figure and add an appropriate title.

3. In this exercise, you will plot some parametric equations for t from 0 to 6π.

   a. First set t by using the range variable (which is both on the matrix toolbar or just type the semicolon). Make sure to use a small step size to make a smooth curve.

   Plot each of the following curves with descriptive titles.

   b. Cardioid: $x(t) = 2\cos t - \cos 2t, y(t) = 2\sin t - \sin 2t.$

   c. Astroid: $x(t) = \cos^3 t, y(t) = \sin^3 t.$

   d. Hypotrochoid:
   $x(t) = 2\cos t + 5\cos\left(\frac{2}{3}t\right), y(t) = 2\sin t - 5\sin\left(\frac{2}{3}t\right).$

   e. Epicycloid:
   $x(t) = 6.5\cos(t) - \cos(6.5t)$
   $y(t) = 6.5\sin(t) - \sin(6.5t)$

4. Reproduce the plot in example 50.

Chapter 16

Mathcad: Curve Fitting

Now that we know how to plot, let's look at how to fit some curves to data and then plot the results.

16.1 Mathcad: Curve Fitting

Data sets are by definition a discrete set of points. We can plot them, but we often want to view trends, interpolate new points and (carefully!) extrapolate for prediction purposes. Linear regression is a powerful tool, although sometimes the data would be better fit by another curve. Here we show how to do linear and quadratic regression.

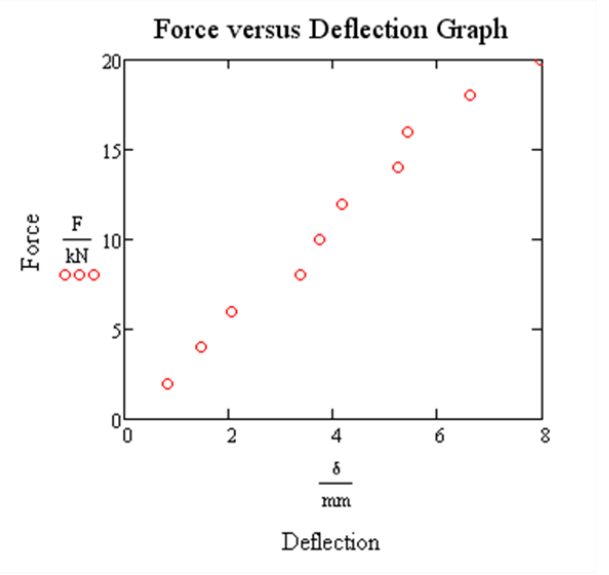
Example 51

Consider the following data set from example 47 in the previous chapter.

$$F := \begin{pmatrix} 2 \\ 4 \\ 6 \\ 8 \\ 10 \\ 12 \\ 14 \\ 16 \\ 18 \\ 20 \end{pmatrix} \cdot \text{kN} \qquad \delta := \begin{pmatrix} 0.82 \\ 1.47 \\ 2.05 \\ 3.37 \\ 3.75 \\ 4.17 \\ 5.25 \\ 5.44 \\ 6.62 \\ 7.97 \end{pmatrix} \cdot \text{mm}$$

A plot of the data suggests a linear relationship:

www.dbooks.org

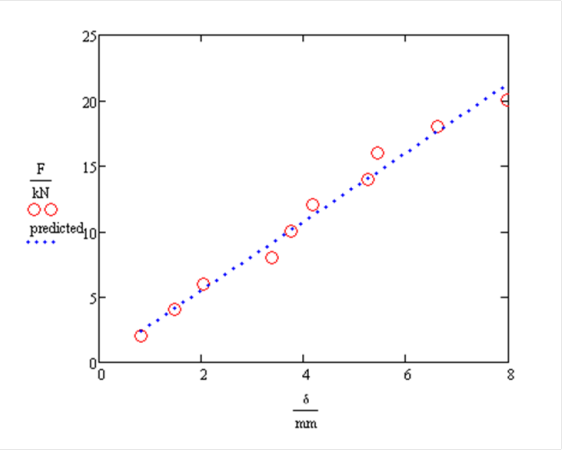


Let's find the least squares line. We consider two methods to do this.

Method 1

$$b := \text{intercept}\left(\frac{\delta}{\text{mm}}, \frac{F}{\text{kN}}\right) \qquad M := \text{slope}\left(\frac{\delta}{\text{mm}}, \frac{F}{\text{kN}}\right) \qquad \text{predicted} := M \cdot \frac{\delta}{\text{mm}} + b$$

Now put them together on the same plot:



Method 2

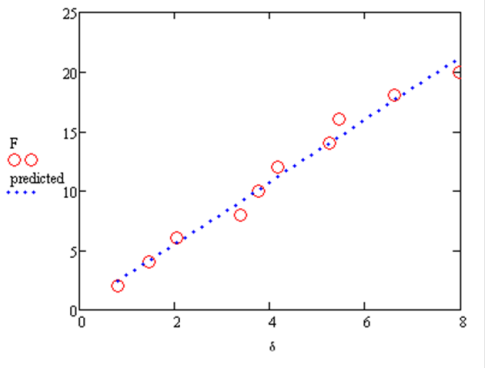
This method uses the command **linfit**. It is a bit awkward here, but will be useful when we do higher order regression. We do have to first remove the units from our variables (a limitation of **linfit**)

$$f(x) := \begin{pmatrix} 1 \\ x \end{pmatrix}$$

$$Coeff := linfit(\delta, F, f)$$

$$b := Coeff_0 \qquad M := Coeff_1 \qquad predicted := M \cdot \delta + b$$

Now put them together on the same plot:

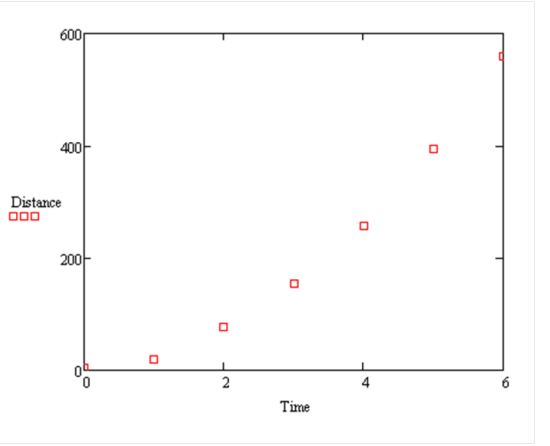


Example 52 Consider the following data set:

$$Time := \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \qquad Distance := \begin{pmatrix} 5 \\ 19.8 \\ 76.8 \\ 153.3 \\ 256.2 \\ 394.5 \\ 559.2 \end{pmatrix}$$

Plotted:

www.dbooks.org



The plot of the data suggests a quadratic fit, i.e. a curve $y = a + bx + cx^2$. How do we find this quadratic curve?

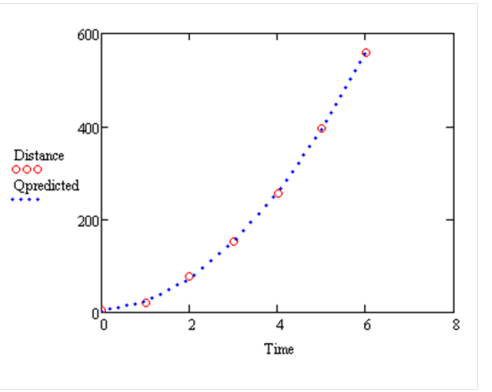
Here's how we implement this in Mathcad:

$$f(x) := \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}$$

Qcoeffs := linfit(Time, Distance, f)

$$Qpredicted := Qcoeffs_0 + Qcoeffs_1 \cdot Time + Qcoeffs_2 \cdot Time^2$$

Now plot them together:



We can compute the R-squared value to see the correlation between Distance and Qpredicted.

RR := corr(Distance, Qpredicted)

RR = 1

RR=1 means a great fit.

www.dbooks.org

Chapter 16 Exercises

1. The volume (in m^3) and pressure (in kPa) of a gas at a constant temperature is recorded in the table as

Volume	**Pressure**
1	2494
2	1247
3	831
4	623
5	499
6	416

   a. Find the least squares line using the intercept and slope functions and plot it along with the data points. Add an appropriate title.

   b. Use the linfit command to find the second-order approximating polynomial. Plot it along with the data points and add an appropriate title.

   c. Use the linfit command to find the third-order approximating polynomial. Plot it along with the data points and add an appropriate title.

2. Resistance (in ohms) and current (in amps) are related through the equation I=V/R. Data was collected from a circuit with unknown constant voltage and is shown in the table.

Resistance(R)	**Current(I)**
10	11.11
15	8.04
25	6.03
40	2.77
65	1.97
100	1.51

   a. Plot R (x-axis) versus I (y-axis). What kind of relationship do you see?

   b. Plot 1/R (x-axis) versus I (y-axis). Now, what kind of relationship do you see?

   c. Use intercept and slope (or use linfit) to calculate the slope and intercept of the line in part **b**.

   d. Approximate the current when the resistance is 80 ohms. You can use the trace capability here.

   e. Create a new plot with the data points from part b, a line with slope and intercept from part c, and the interpolated point (with a different symbol) from part d. Label and title appropriately.

Problems continue on the next page...

3. Suppose that a population, P, grows over time, t, according to exponential growth with the following data.

Time(years)	**Population(thousands)**
1	1.61
2	2.51
3	2.17
4	6.14
5	6.81
6	17.53
7	16.36
8	25.04
9	39.61
10	55.50

We have that the population is therefore growing according to the equation $P = P_0 e^{kt}$. Taking a logarithm, we can convert this into $\ln(P) = \ln(P_0) + kt$ (i.e. $\ln(P)$ is linear in t).

   a. Plot the data with t on the horizontal axis and $\ln(P)$ on the vertical axis.

   b. Find the slope and intercept for the data in part a and then create a new plot with this regression line and the data in part a.

www.dbooks.org

Chapter 17

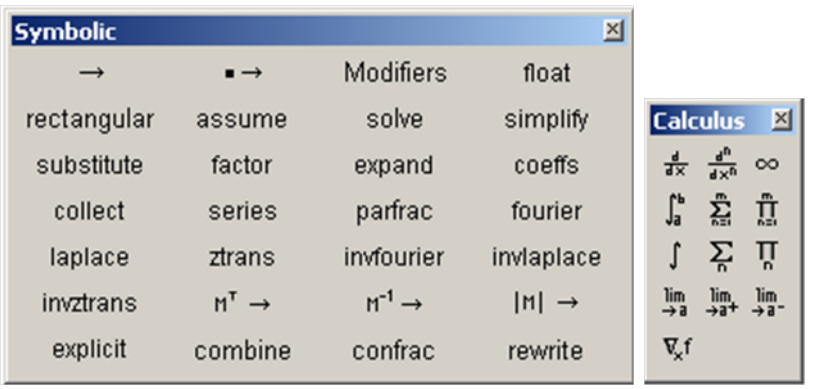
Mathcad: Calculus and Symbolics

Up to this point, we have investigated the numeric capabilities of Mathcad. In this section, we learn about the calculus and symbolic capabilities of Mathcad.

17.1 Mathcad: Calculus

Using the calculus toolbar

First, open both the calculus and the the symbolic toolbar with File > Toolbars > Calculus, and File > Toolbars > Symbolic.



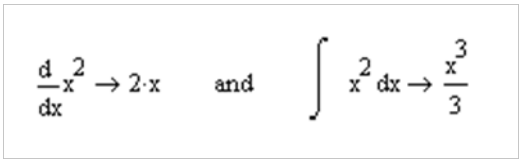
Many of the key concepts from a two-semester sequence of Calculus courses can be found on the calculus toolbar: limits, derivatives (including the gradient), integrals (definite and indefinite), summations and products. The symbol for infinity is also on this toolbar. The symbolic toolbar may look a bit more complicated (it is). We will cover many of the symbolic toolbar buttons in the

next section, but in order to complete many of the symbolic calculations using the calculus toolbar, we will need the "Symbolic Evaluation" (the $\longrightarrow$ button) from the symbolic toolbar too. Of course, there is a shortcut for symbolic evaluation: Ctrl + . (hold the Ctrl key and press the period).

We start with some simple examples.

Example 53 **Compute the derivative and integral of x^2.**

By hand, we know the answers are $2x$ and $x^3/3+C$. Using Mathcad, we use the symbolic differentiation  and indefinite integral  buttons. In each case, we end the calculation with the symbolic evaulation button $\rightarrow$.



Note that the $+c$ part of the indefinite integral is not shown.

17.2 Mathcad: Symbolics

Here we look at many of the other capabilities of the symbolic toolbar. The "solve" button was discussed in a previous chapter.

Example 54 **Expand $(x+2)^4$, factor x^3+3x^2+3x+1, find the Taylor polynomial of order 8 for $\cos x$, find the partial fraction expansion of $\dfrac{1}{x^3-x}$, and find the Laplace transform of t^2-1.**

In each case, we enter the expression, select the button from the symbolic toolbar, make any alterations (like in the Taylor polynomial question - adding the comma and 8) and click outside the expression

www.dbooks.org

$$(x+2)^4 \text{ expand } \rightarrow x^4 + 8\cdot x^3 + 24\cdot x^2 + 32\cdot x + 16$$

$$x^3 + 3\cdot x^2 + 3\cdot x + 1 \text{ factor } \rightarrow (x+1)^3$$

$$\cos(x) \text{ series}, 8 \rightarrow 1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720}$$

$$\frac{1}{x^3 - x} \text{ parfrac } \rightarrow \frac{1}{2\cdot(x-1)} - \frac{1}{x} + \frac{1}{2\cdot(x+1)}$$

$$t^2 - 1 \text{ laplace } \rightarrow -\frac{s^2 - 2}{s^3}$$

In the next example, we point out the difference between symbolic evaluation and approximation.

Example 55 The Error Function

The error function $f(t)$ comes up in many applications. It is defined as

$$f(t) := \frac{2}{\sqrt{\pi}} \cdot \int_0^t e^{-x^2}\, dx$$

If we use the standard symbolic evaluation symbol $\longrightarrow$, then we will get the answer in terms of the built in error function "erf"

$$f(1) \rightarrow \text{erf}(1)$$

If we instead use the evaluation equals sign (just =), we get the approximation

$$f(1) = 0.843$$

Chapter 17 Exercises

In these exercises, you need to use the calculus and symbolic keyword toolbars.

1. Compute the following limits.
   a. $\lim_{x \to 0} \dfrac{\sin x}{x}$
   b. $\lim_{x \to 0^+} \dfrac{1}{x^2}$
   c. $\lim_{x \to \infty} \arctan x$
2. Find the following derivatives.
   a. If $f(x) = \dfrac{1}{(x-1)(x-2)}$, find $f'(x)$ and $f''(x)$.
   b. If $f(x) = \sqrt[3]{x^2 - 1}$, find $f'(x)$.
   c. If $f(x, y) = e^{x^2+y^2}$, find $\dfrac{\partial f}{\partial x}$, $\dfrac{\partial f}{\partial y}$, and $\dfrac{\partial^2 f}{\partial x \partial y}$.
3. Find the following integrals.
   a. The area under the bell curve between -1 and 1 represents the percentage of a population that is one standard deviation away from the mean (or 68.3%). Show this is true by approximating the integral
   $$\sqrt{\frac{1}{2\pi}} \int_{-1}^{1} e^{-x^2/2}\, dx.$$
   b. The area between the cardioid $r = 1 + \cos\theta$ and the circle $r = 1$ is given by the polar integral
   $$\int_{-\pi/2}^{\pi/2} \int_{1}^{1+\cos\theta} r\, dr d\theta.$$
   Find the exact value of this integral using the exact variables r and θ.
   c. The volume of the solid bounded by the sphere of radius 1 (centered at the origin) and the cone $\phi = \pi/3$ is given by
   $$\int_0^{2\pi} \int_0^{\pi/3} \int_0^1 \rho^2 \sin\phi\, d\rho d\phi d\theta.$$
   Find the exact value of this integral, using the exact variables ρ, ϕ and θ.
4. Find the partial fraction decomposition of each of the following.
   a. $\dfrac{x^3 + 1}{x^2 - 1}$
   b. $\dfrac{1}{2x^6 - x^4 - x^2}$

www.dbooks.org

5. Find the following sums and products.
   a. $\sum_{i=1}^{10} i^2$ and $\prod_{i=1}^{10} i^2$.
   b. $\sum_{i=1}^{n} i^1, \sum_{i=1}^{n} i^2, \sum_{i=1}^{n} i^3$ and $\sum_{i=1}^{n} i^4$
   (your answers should be in terms of n).
   c. $\sum_{i=0}^{100} \frac{1}{i!}$ and $\sum_{j=1}^{1000} \frac{(-1)^{j+1}}{j}$.

   E.C. Do the numbers in part c look familiar? What are they?

6. Find the following Taylor Polynomials (i.e. series expansions).
   a. The 8th order polynomial for e^{2x}.
   b. The 6th order polynomial for $\cos(x^2)$.

Appendix

This section contains tables, plots, etc. that are referred to in various locations throughout the text and the exercises.

Table 1 GDPAndDeficitByYear.xlsx - used in Chapter 6 exercises

Columns with year, GDP (in billions), Deficit (% of GDP)

1910	33.4	-0.11	1935	73.3	4.12	1960	526.4	-0.48	1985	4217.5	5.03
1911	34.3	-0.12	1936	83.8	4.76	1961	544.8	0.65	1986	4460.1	4.96
1912	37.4	0.01	1937	91.9	2.84	1962	585.7	1.22	1987	4736.4	3.16
1913	39.1	0.02	1938	86.1	1.42	1963	617.8	0.77	1988	5100.4	3.04
1914	36.5	0.2	1939	92.2	2.32	1964	663.6	0.89	1989	5482.1	2.78
1915	38.7	0.56	1940	101.4	3.02	1965	719.1	0.2	1990	5800.5	3.81
1916	49.6	0.31	1941	126.7	3.73	1966	787.7	0.47	1991	5992.1	4.49
1917	59.7	1.82	1942	161.9	12.04	1967	832.4	1.04	1992	6342.3	4.58
1918	75.8	11.88	1943	198.6	28.05	1968	909.8	2.77	1993	6667.4	3.83
1919	78.3	16.86	1944	219.8	22.35	1969	984.4	-0.33	1994	7085.2	2.87
1920	88.4	-0.68	1945	223	24.07	1970	1038.3	0.27	1995	7414.7	2.21
1921	73.6	-0.91	1946	222.2	9.06	1971	1126.8	2.04	1996	7838.5	1.37
1922	73.4	-0.68	1947	244.1	-1.32	1972	1237.9	1.89	1997	8332.4	0.26
1923	85.4	-0.66	1948	269.1	-4.33	1973	1382.3	1.08	1998	8793.5	-0.79
1924	86.9	-0.73	1949	267.2	-1.48	1974	1499.5	0.41	1999	9353.5	-1.34
1925	90.6	-0.47	1950	293.7	0.43	1975	1637.7	3.25	2000	9951.5	-2.37
1926	96.9	-0.67	1951	339.3	-2.3	1976	1824.6	4.04	2001	10286.2	-1.25
1927	95.5	-0.98	1952	358.3	-0.06	1977	2030.1	2.64	2002	10642.3	1.48
1928	97.4	-0.68	1953	379.3	1.52	1978	2293.8	2.58	2003	11142.1	3.39
1929	103.6	-0.46	1954	380.4	0.49	1979	2562.2	1.59	2004	11867.8	3.48
1930	91.2	-0.96	1955	414.7	0.37	1980	2788.1	2.65	2005	12638.4	2.52
1931	76.5	0.17	1956	437.4	-1.21	1981	3126.8	2.53	2006	13398.9	1.85
1932	58.7	2.78	1957	461.1	-1.15	1982	3253.2	3.93	2007	14077.6	1.14
1933	56.4	3.27	1958	467.2	0.01	1983	3534.6	5.88	2008	14441.4	3.18
1934	66	3.11	1959	506.6	1.59	1984	3930.9	4.72	2009	14258.2	9.91
									2010	14623.9	10.64

www.dbooks.org

Table 2 UnemploymentByYear.xlsx - used in Chapter 6 exercises

Values are given in percentages.

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2000	4	4.1	4	3.8	4	4	4	4.1	3.9	3.9	3.9	3.9
2001	4.2	4.2	4.3	4.4	4.3	4.5	4.6	4.9	5	5.3	5.5	5.7
2002	5.7	5.7	5.7	5.9	5.8	5.8	5.8	5.7	5.7	5.7	5.9	6
2003	5.8	5.9	5.9	6	6.1	6.3	6.2	6.1	6.1	6	5.8	5.7
2004	5.7	5.6	5.8	5.6	5.6	5.6	5.5	5.4	5.4	5.5	5.4	5.4
2005	5.3	5.4	5.2	5.2	5.1	5	5	4.9	5	5	5	4.9
2006	4.7	4.8	4.7	4.7	4.6	4.6	4.7	4.7	4.5	4.4	4.5	4.4
2007	4.6	4.5	4.4	4.5	4.4	4.6	4.6	4.6	4.7	4.7	4.7	5
2008	5	4.8	5.1	5	5.4	5.5	5.8	6.1	6.2	6.6	6.9	7.4
2009	7.7	8.2	8.6	8.9	9.4	9.5	9.4	9.7	9.8	10.1	10	10
2010	9.7	9.7	9.7	9.9	9.7	9.5	9.5	9.6	9.6	9.6	9.8	

Index

Mathcad
- :=, Assignment Equals, 74
- =, Evaluation Equals, 72
- ≡, Global Equals, 75
- =, Symbolic Equals, 75
- Reference matrix entries, 95
- Aligning Regions, 78
- Formatting Results, 79
- Function Listing, 87
- Highlighting Regions, 78
- Insert\Delete Matrix Row or Column, 94
- Matrices, 91
- Plot Customization, 105
- Quickplot, 107
- Reference matrix column, 96
- Reference matrix row, 96
- Symbolic Evaluation, 119
- Toolbars
  - Calculus, 118
  - Graphing, 103
  - Matrix, 91
  - Symbolic, 118
- Units, 79
- User-defined Functions, 88

Mathcad Functions
- corr, 115
- intercept, 112
- linfit, 113
- slope, 112
- augment, 93
- expand, 119
- factor, 119
- given\find block, 82
- laplace, 119
- lsolve, 99
- parfrac, 119
- rref, 98
- series, 119
- solve block, 84
- stack, 94
- submatrix, 97
- absolute value, 88
- square root, 88

Matlab
- Block Commenting, 5
- Block Matrices, 15
- Built-in Help, 3
- colon (:), 10
- Command History, 2
- Command Window, 2
- Directory, 3
- dot operations (.∗, ./, .∧), 14
- Editor, 4
- Left Division, 20
- Logical Operators, 51
- Matrix Definition, 9
- Relational Operators, 51
- Running Code, 6
- semicolon (;), 3, 10

Matlab Functions
- `clc`, 3
- `clear`, 3
- `close`, 3
- `cross`, 18
- `ctrl + c`, 3
- `det`, 18
- `disp`, 44
- `dot`, 18
- `exp`, 24
- `eye`, 19
- `find`, 26
- `fliplr`, 26
- `flipud`, 26

www.dbooks.org

for, 54
fprintf, 44
function, 37
hold, 32
if\elseif, 53
input, 43
interp1, 64
inv, 18
length, 16
linspace, 11
log,log10, 24
max, 17
mean, 25
median, 25
meshgrid, 27
min, 17
nargin, 40
nargout, 40
num2str, 45
ones, 11
plot, 30
polyfit, 63
prod, 18
rref, 19
sin, sind, 24
size, 15
sortrows, 26
sort, 26
sqrt, 24
std, 25
subplot, 33
sum, 18
switch\case, 53
text, 31
title, 31
type, 40
while, 54
xlim, 31
ylim, 31
zeros, 11
ginput, 46
legend, 46
polyval, 66
xlsread, 46