

Introduction to Matlab

Adapted from

<http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml>

Accessing Matlab

- MIT students with Athena account:
 - On Athena, type “add Matlab”
 - Then type “Matlab &”
- WI personnel with a Fladda account:
 - On Fladda, type “Matlab &”
- Genome center personnel with a ____ account
 - On _____, type “Matlab &”

Matlab will open in a new window.

If you do not have access to one
of these accounts, or other access
to Matlab

Contact Jeanne Darling, the course
secretary, at darling@psrg.lcs.mit.edu
and request an account

Conventions used in this document:

- **Examples** will be highlighted in red, with a “**Try this!**” on the slide
- Things you should type into MATLAB are *italicized*
- MATLAB’s display in response is in **bold**
- Notes are displayed in blue

Getting started: matrices

- A matrix is a rectangular array of numbers, with some number of rows and columns
- A 1×1 matrix (1 row by 1 column) is a single number, also called a scalar or a constant
- A $1 \times n$ matrix (1 row by n columns) is called a vector, or a row vector
- An $n \times 1$ matrix (n rows by 1 column) is called a vector, or a column vector

Matlab allows you to work with entire matrices quickly and easily

Entering matrices: Enter an explicit list of matrix elements

- Separate the elements of each row with blanks or commas
- Indicate the end of a row with a semicolon
- Surround the entire list of elements with square brackets, []

Entering matrices: Enter an explicit list of matrix elements cont.

Example:

If you type

```
my_matrix = [1 2 3; 4 5 6; 7 8 9]
```

a matrix with the name `my_matrix` is now in memory. Matlab will display your matrix as follows:

```
my_matrix =
```

```
 1  2  3  
 4  5  6  
 7  8  9
```

Try this!

Entering matrices: Loading data from a file

- Data can be loaded into Matlab from a text file: simply type *load myfile.txt*, and the data in *myfile.txt* will be stored in a matrix called *myfile*.
- Data can be loaded from a saved Matlab file: type *load myfile.mat*. Mat files can contain a number of matrices, each will be loaded into Matlab
- NOTE: The directory Matlab looks in for data files is by default the directory you were in when you started the program. To look elsewhere, you must specify the directory!

Entering matrices: Loading data from a file cont.

Example:

Create a text file that has an array of numbers in it. In your main window, type:

```
emacs mytest.txt &
```

enter some numbers

```
47, 65, 98, 102, 451
```

```
921, 25, 89, 194, 87
```

```
2, 10, 74, 66, 200
```

then press *Ctrl-x* then *Ctrl-c* to exit emacs. You will be asked if you want to save the file - say *yes*.

Go into Matlab and type *load mytest.txt*

Then type *whos*

The matrix and its size will be displayed.

Try this!

Accessing data in a matrix

Individual elements in a matrix are referred to by what are called subscripts: the row and column that a particular element is in.

from the previous example, $\text{mytest}(2,4) = 194$

mytest =

47	65	98	102	451
921	25	89	194	87
2	10	74	66	200

That is, it's the element in the second row, fourth column. When using subscripts, the row is always listed first, then the column.

Accessing data in a matrix

Example:

(Make sure you've completed the previous example, and loaded mytest.txt)

Type *mytest(3,2)*

Matlab will display

ans =

10

Type *myvariable = mytest(1,4)*

Matlab will display

myvariable =

102

Try this!

Note: Here we've assigned the value 102 to the variable named **myvariable**. If you don't assign a variable name, Matlab puts the answer to a query in the variable **ans**. This gets overwritten every time you type something without assigning it to a name, so beware!

Accessing data in a matrix: The colon operator

The expression `1:10` is a row vector containing the integers from 1 through 10

`1:10 = [1 2 3 4 5 6 7 8 9 10]`

Subscript expressions involving colons indicate portions of a matrix

`mytest(1, 2:4)` equivalent to `mytest(1,[2 3 4])`

refers to the first row, columns 2, 3, and 4 of matrix `mytest`.

`mytest(2:3, 5)`

refers to rows 2 and 3, column 5

The **colon by itself** refers to ALL elements in a row or column of a matrix, the keyword **end** refers to the LAST row or column.

`mytest(3, :)`

refers to all elements of row 3

Accessing data in a matrix: The colon operator cont.

Example:

Type *mytest(1, 2:5)*

Matlab displays

ans =

65 98 102 451

Type *mytest(:, 4:5)*

Matlab displays

ans =

102 451

194 87

66 200

Type *myvariable = mytest(end, 1:3)*

Matlab displays

myvariable =

2 10 74

Try this!

Expressions: Variables

- Variable names consist of a letter, followed by any number of letters, digits, or underscores.
- Only the first 31 characters of a variable name are used.
- MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are *not* the same variable.
- To view the matrix assigned to any variable, simply enter the variable name. To view all variables currently assigned, type *who*. To view with sizes, type *whos*.

Expressions: Operators

Operator:

+

-

* OR *

/ OR ./

^ OR .^

'

()

Function:

Addition

Subtraction

Multiplication

Division

Power ($x^2 = x*x$)

Transpose (interchange rows and columns)

Precedence

Expressions cont.

Example:

Operators work exactly as you would expect when using two scalars, or a matrix and a scalar

Type $2+2$

ans =

4

Type $2*3$

ans =

6

Type 3^3

ans =

27

Type $my_matrix+20$

ans =

21 22 23

24 25 26

27 28 29

Type $my_matrix/2$

ans =

0.5 1.0 1.5

2.0 2.5 3.0

3.5 4.0 4.5

Try This!

Expressions cont.

Be careful when you're working with TWO matrices! Addition and subtraction work as you would expect, but multiplication, division and exponents don't!

Example:

Type $my_matrix + my_matrix$

ans =

2	4	6
8	10	12
14	16	18

Each element in the first matrix is added to the corresponding element in the second - same for subtraction

Type $my_matrix * my_matrix$

ans =

30	36	42
66	81	96
102	121	150

NOT each element of the first matrix times the corresponding element of the second!

Expressions cont.

In order to get element by element multiplication, division or exponents, use the `.*`, `./` and `.^` operators.

Example:

Type `my_matrix.*my_matrix`

ans =

1	4	9
16	25	36
49	64	81

NOW each element in the first matrix is multiplied by the corresponding element in the second - same for division and exponent

Try this!

Expressions cont.

The transpose operator: `'` (single quote) This operator inverts the rows and columns of a matrix.

Example:

Type *mytest*'

ans =

```
47  921  2
65   25 10
98   89 74
102 194 66
451  87 200
```

Try this!

Note that what was the first ROW of mytest is now the first column. What was the second row is now the second column, and the third row is now the third column.

Functions

Matlab has built in functions to perform just about any common mathematical operation. These functions include `log()`, `sqrt()`, `abs()`, `exp()` {take the natural log, square root, absolute value, and e to the power}

The functions can be used with either a scalar or a matrix as an argument.

Example:

Type `log(my_matrix)`

ans =

0	0.6931	1.0986
1.3863	1.6094	1.7918
1.9459	2.0794	2.1972

Type `log(9265)`

ans =

9.1721

Try this!

Matlab online help:

Matlab's online documentation is excellent. Go to:

<http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml>

This has information on all the included functions, plus the environment etc.

After this intro, you should be able to:

- Enter in a matrix or variable manually
- Load a matrix from a file
- Access data in a matrix: specific elements as well as portions of the matrix
- See what variables you have in memory
- Use basic mathematical operations on matrices and scalars
- Use some basic functions on matrices and scalars