
Lect01 - Matlab Environment

Goals

Explore the Matlab windows

Use Matlab as an advanced calculator

Learn how to save and document your work

Command Window

The Command Window on the right is the main panel where you interact with MATLAB.

You type numbers and commands in this window and hit <Enter>. MATLAB executes the commands and displays results (if requested). In the following the commands that you type are indicated by >>.

Once you hit enter, you can't edit any commands.

You can use the arrow keys to retrieve commands and edit them before hitting enter again.

Arithmetic with MATLAB

Basic operators

```
>> {3-2, 3*2, 3/2, 3^2, 2/0, 0/2}
```

```
ans =
     [1]     [6]    [1.5000]     [9]    [Inf]     [0]
```

Left division. Percent sign is used to indicate a comment. Matlab ignores anything after the % symbol.

```
>> 3\2 % Same as 2/3
```

```
ans =
    0.6667
```

Infinity

```
>> 3*Inf
```

```
ans =
    Inf
```

pi

```
>> 2*pi
```

```
ans =
    6.2832
```

NaN (Not a Number)

```
>> 0/0
```

```
ans =
    NaN
```

Hierarchy of Operations

Just like in mathematics the operations are done in the following order: Left to right doing what is in Parentheses & Exponents first, followed by Multiplication & Division, and then Addition & Subtraction last.

$2+3^2+1/(1+2)$

$2+3^2+1/3 \Rightarrow 2+9+1/3 \Rightarrow 2+9+0.33333 \Rightarrow 11+0.33333 \Rightarrow 11.33333$

```
>> {2+3^2+1/(2+1), 2+3^2+1/2+1}
```

```
ans =
    [11.3333]    [12.5000]
```

Elementary Math Functions

There are functions for almost anything you want to do.

```
>> {sqrt(2), sin(30), asin(pi), cos(3*pi), log(100), log10(100)}  
ans =  
    [1.4142]    [-0.9880]    [1.5708]    [-1]    [4.6052]    [2]
```

Note the trigonometric functions expect arguments in radians.

Algebraic-numeric computations

Let us explore by doing exercises:

```
>> a = 3
a =
    3
>> b = 2
b =
    2
>> {a - b, a / b, a^2}
ans =
    [1]    [1.5000]    [9]
```

```
>> c = a * b
c =
    6
>> d = c^(b+1)
d =
    216
```

Hiding Output

```
>> a = 5; b = 3;
>> c = a * b;
>> d = c^(b+1);
```

Variables a, b, c, d are created but their values are not displayed. We can see value of one or more variables by simply typing their name and hitting Enter key.

```
>> a, b, c, d
a =
    5
b =
    3
c =
    15
d =
    50625
```

Class Activity 1 - Template for saving and documenting your work

Open a new file using Matlab Editor. Lines starting with a % sign are comments explaining what you are doing. Lines starting with two % signs followed by a space are treated as start of new cells. The current cell is highlighted with yellow background. You can execute all commands in the cell by hitting **Ctrl+Enter** keys. The output will appear in the Command window. After completing all exercises use the Publish command from the File menu of the editor window to create a fully formatted file that contains all input and output organized into neat sections.

```
%% Lect01 - Class Activities
% M. A. Bhatti
```

The following commands entered as Initialization cell are useful. **clc** clears all previous entries from the workspace window. **clf** clears the figure window. **close** will close any open figure windows. **clear** will remove all variables from the workspace. **format compact** suppresses excess line feeds to show more output in a single screen.

```
%% Initialization
clc; clf; close; clear
format compact
```

Put each new calculation in a separate cell by starting with two % signs followed by a space and a descriptive title.

```
%% Simple calculations
% Basic operators
{3-2, 3*2, 3/2, 3^2, 2/0, 0/2}
```

```
%% Left division
% Same as 2/3
3\2
```

Greek letters and formatted equations can be entered in the comment cells by enclosing the equation in a set of double \$\$ signs. \TeX codes are used for greek letters and other special mathematical symbols.

```
%% Infinity
% $$ Inf = \infty $$
3*Inf
```

```
%% pi
% $$ pi = \pi $$
2*pi
```

To keep the input and output of a cell together put each line of code in a separate cell by adding a line with two % signs.

```
%% Algebraic-numeric computations
a = 3
%%
b = 2
%%
{a - b, a / b, a^2}
%%
c = a * b
%%
d = c^(b+1)
```

First page of the published file from the template code.

Lect01 - Class Activities

M. A. Bhatti

Contents

- [Initialization](#)
- [Simple calculations](#)
- [Left division](#)
- [Infinity](#)
- [pi](#)
- [Hierarchy of Operations](#)
- [Elementary Math Functions](#)
- [Algebraic-numeric computations](#)
- [Hiding Output](#)

Initialization

```
clc; clf; close; clear
format compact
```

Simple calculations

Basic operators

```
{3-2, 3*2, 3/2, 3^2, 2/0, 0/2}
```

```
ans =
     [1]     [6] [1.5000]     [9] [Inf]     [0]
```

Left division

Same as 2/3

```
3\2
```

```
ans =
    0.6667
```

Infinity

```
Inf = ∞
```

```
3*Inf
```

```
ans =
    Inf
```

pi

```
pi = π
```

```
2*pi
```

T_EX Codes

Lowercase Greek letters

α \alpha	ι \iotaota	ϱ \varrhorho
β \beta	κ \kappaappa	σ \sigmasigma
γ \gammagamma	λ \lambdlambda	ς \varsigmasigma
δ \deltadelta	μ \mumu	τ \tautau
ϵ \epsilonpsilon	ν \nunu	υ \upsilonpsilon
ε \varepsilonpsilon	ξ \xixi	ϕ \phiphi
ζ \zetaeta	o o	φ \varphiphi
η \etaeta	π \pipi	χ \chichi
θ \thetatheta	ϖ \varpitheta	ψ \psipsi
ϑ \varthetatheta	ρ \rorho	ω \omegomega

Uppercase Greek letters

Γ \Gamma	Ξ \Xi	Φ \Phi
Δ \Delta	Π \Pi	Ψ \Psi
Θ \Theta	Σ \Sigma	Ω \Omega
Λ \Lambda	Υ \Upsilon	

Miscellaneous symbols

\aleph \aleph	\prime \prime	\forall \forall
\hbar \hbar	\emptyset \emptyset	\exists \exists
∇ \nabla	\neg \neg	
\flat \flat	$\sqrt{\quad}$ \sqrt	
ℓ \ell	\top \top	\natural \natural
\wp \wp	\perp \perp	\sharp \sharp
\Re \Re	\parallel \parallel	\clubsuit \clubsuit
\Im \Im	\angle \angle	\diamond \diamond
∂ \partial	\triangle \triangle	\heartsuit \heartsuit
∞ \infty	\backslash \backslash	\spadesuit \spadesuit
\Box \Box	\diamond \diamond	

Large operators

\sum \sum	\bigcap \bigcap	\prod \prod
\bigcup \bigcup	\coprod \coprod	\int \int
\bigvee \bigvee	\oint \oint	\bigwedge \bigwedge

Binary operations

\pm \pm	\vee \vee	\mp \mp
\wedge \wedge	\setminus \setminus	\uplus \uplus
\oplus \oplus	\sqcap \sqcap	\ominus \ominus
\times \times	\sqcup \sqcup	\otimes \otimes
$*$ \star	\odot \odot	
\diamond \diamond	\wr \wr	\dagger \dagger
\circ \circ	\bigcirc \bigcirc	\ddagger \ddagger
\triangleup \triangleup	\amalg \amalg	
\div \div	\triangledown \triangledown	\leqslant \leqslant
\ll \ll	\gg \gg	\leqslant \leqslant

Relations

\leq \leq	\geq \geq	\equiv \equiv
$<$ \prec	$>$ \succ	\sim \sim
\leqslant \preceq	\succcurlyeq \succeq	\approx \approx
\ll \ll	\gg \gg	\asymp \asymp
\subset \subset	\supset \supset	\approx \approx
\subseteq \subseteq	\supseteq \supseteq	\cong \cong
\sqsubseteq \sqsubseteq	\sqsupseteq \sqsupseteq	
\in \in	\ni \ni	\propto \propto
\vdash \vdash	\dashv \dashv	\models \models
\smile \smile	\mid \mid	\doteq \doteq
\frown \frown	\parallel \parallel	\perp \perp
\sqsubset \sqsubset	\sqsupset \sqsupset	

Negated relations

$\nless \backslash \text{not} <$ $\nless \backslash \text{not} >$ $\neq \backslash \text{not} =$
 $\nleq \backslash \text{not} \leq$ $\ngeq \backslash \text{not} \geq$ $\nequiv \backslash \text{not} \text{equiv}$
 $\nprec \backslash \text{not} \prec$ $\nsucc \backslash \text{not} \succ$ $\nsim \backslash \text{not} \sim$
 $\npreceq \backslash \text{not} \preceq$ $\nsucceq \backslash \text{not} \succeq$ $\nsimeq \backslash \text{not} \simeq$
 $\nsubset \backslash \text{not} \subset$ $\nsupset \backslash \text{not} \supset$ $\napprox \backslash \text{not} \approx$
 $\nsubseteq \backslash \text{not} \subseteq$ $\nsupseteq \backslash \text{not} \supseteq$ $\ncong \backslash \text{not} \cong$
 $\nsubsetseq \backslash \text{not} \subsetseq$ $\nsupsetseq \backslash \text{not} \supsetseq$ $\nasymp \backslash \text{not} \asymp$

Arrows

$\leftarrow \backslash \text{leftarrow}$ $\longleftarrow \backslash \text{longleftarrow}$ $\uparrow \backslash \text{uparrow}$
 $\Leftarrow \backslash \text{Leftarrow}$ $\Longleftarrow \backslash \text{Longleftarrow}$ $\Uparrow \backslash \text{Uparrow}$
 $\rightarrow \backslash \text{rightarrow}$ $\longrightarrow \backslash \text{longrightarrow}$ $\downarrow \backslash \text{downarrow}$
 $\Rightarrow \backslash \text{Rightarrow}$ $\Longrightarrow \backslash \text{Longrightarrow}$ $\Downarrow \backslash \text{Downarrow}$
 $\leftrightarrow \backslash \text{leftrightarrow}$ $\longleftrightarrow \backslash \text{longleftrightarrow}$ $\Updownarrow \backslash \text{updownarrow}$
 $\Leftrightarrow \backslash \text{Leftrightarrow}$ $\Longleftrightarrow \backslash \text{Longleftrightarrow}$ $\Updownarrow \backslash \text{Updownarrow}$
 $\mapsto \backslash \text{mapsto}$ $\nearrow \backslash \text{nearrow}$ $\searrow \backslash \text{searrow}$
 $\leftharpoonup \backslash \text{leftharpoonup}$ $\rightharpoonup \backslash \text{rightharpoonup}$ $\swarrow \backslash \text{swarrow}$
 $\leftharpoondown \backslash \text{leftharpoondown}$ $\rightharpoondown \backslash \text{rightharpoondown}$ $\nwarrow \backslash \text{nwarrow}$
 $\rightrightarrows \backslash \text{rightrightarrows}$

Openings

$[\backslash \text{lbrack}$ $\lfloor \backslash \text{lfloor}$ $\lceil \backslash \text{lceil}$ $\{ \backslash \text{lbrace}$

Closings

$] \backslash \text{rbrack}$ $\rfloor \backslash \text{rfloor}$ $\rceil \backslash \text{rceil}$ $\} \backslash \text{rbrace}$

Alternate names

$\neq \backslash \text{ne}$ or $\backslash \text{neq}$ (same as $\backslash \text{not} =$)
 $\leq \backslash \text{le}$ (same as $\backslash \text{leq}$)
 $\geq \backslash \text{ge}$ (same as $\backslash \text{geq}$)
 $\{ \backslash \{$ (same as $\backslash \text{lbrace}$)
 $\} \backslash \}$ (same as $\backslash \text{rbrace}$)
 $\rightarrow \backslash \text{to}$ (same as $\backslash \text{rightarrow}$)
 $\leftarrow \backslash \text{gets}$ (same as $\backslash \text{leftarrow}$)
 $\ni \backslash \text{owns}$ (same as $\backslash \text{ni}$)
 $\wedge \backslash \text{land}$ (same as $\backslash \text{wedge}$)
 $\vee \backslash \text{lor}$ (same as $\backslash \text{vee}$)
 $\neg \backslash \text{Inot}$ (same as $\backslash \text{neg}$)
 $| \backslash \text{vert}$ (same as $|$)
 $\| \backslash \text{Vert}$ (same as $\|$)

Misc

∴ \vdots

⋯ \cdots

⋮ \ddots

-- (hyphen)

— (en dash)

— (em dash)

å \aa

Å \AA

ø \o

Ø \O

ł \l

Ł \L

ß \ss

§ \S

¶ \P

† \dag

‡ \ddag

© \copyright

£ \pounds

… \ldot

Variable names

All names must start with a letter.

They may contain letters, numbers and the underscore (_).

Names are case sensitive.

There are certain keywords you can't use. Use the **iskeyword** function for a list of keywords.

```
>> iskeyword
ans =
    'break'
    'case'
    'catch'
    'classdef'
    'continue'
    'else'
    'elseif'
    'end'
    'for'
    'function'
    'global'
    'if'
    'otherwise'
    'parfor'
    'persistent'
    'return'
    'switch'
    'try'
    'while'
```

Which of these names are allowed in Matlab?

1. test
2. Test
3. if
4. my-book
5. my_book
6. var?
7. 1stgroup
8. group_one
9. zzaAbc
10. z34wAwy #12
11. sin
12. log

The following variable names are allowed but should be avoided because of obvious name conflicts with the built-in functions.

1. sin
2. log

Array and Matrices

Using Matlab as a glorified calculator is OK, but its real strength is in matrix manipulations.

Defining row and column vectors

A vector is defined by placing a sequence of numbers within square braces.

```
>> v = [3 1 12]

v =
     3     1    12
```

This creates a row vector called "v". If you do not want to print out the result put a semi-colon at the end of the line.

```
>> v = [3 1 12];
```

If you want to view the vector just type its name.

```
>> v

v =
     3     1    12
```

You can also separate entries in a row by commas.

```
>> w = [3, 1, 7, -21, 5, 6]

w =
     3     1     7   -21     5     6
```

You can define a column vector by separate entries by semicolon.

```
>> p = [3; 1; 7; -21; 5; 6]

p =
     3
     1
     7
   -21
     5
     6
```

You can change rows into columns and vice versa by using the matrix Transpose operation. In Matlab the transpose is defined using an apostrophe (').

```
>> p = w'

p =
     3
     1
     7
   -21
     5
     6
```

Instead of first creating a row matrix and then taking its transpose, the following line directly defines a column vector without using semicolons.

```
>> p = [3 1 7 -21 5 6]';
```

A common task is to create a large vector with numbers that fit a repetitive pattern. Matlab can define a set of numbers with a common increment using colons. For example, to define a vector whose first entry is 1, the second entry is 2, the third is three, up to 8 you enter the following.

```
>> v = [1:8]
```

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

If you wish to use an increment other than one that you have to define the start number, the value of the increment, and the last number. For example, to define a column vector that starts with 2 and ends in 4 with steps of .25 you enter the following.

```
>> v = [2:.25:4]'
```

```
v =
 2.0000
 2.2500
 2.5000
 2.7500
 3.0000
 3.2500
 3.5000
 3.7500
 4.0000
```

You can use negative increments. For example, to define a vector with the numbers from 0 to -4 in steps of -1 we do the following.

```
>> u = [0:-1:-4]
```

```
u =
     0    -1    -2    -3    -4
```

Be careful. Matlab, like any other computer program, is dumb and will do strange things if you tell it to. For example, it obviously is impossible to define a vector with the numbers from 0 to 4 in steps of -1.

```
>> u = [0:-1:4]
```

```
u =
Empty matrix: 1-by-0
```

Defining matrices

Defining a matrix is similar to defining a vector. To define a matrix, you can treat it like a column of row vectors.

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

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

You can also treat it like a row of column vectors.

```
>> B = [ [1 2 3]' [2 4 7]' [3 5 8]']
```

```
B =
```

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

Accessing elements within a vector

You can view individual entries in this vector. For example to view the third entry of the vector `wt` defined earlier just type in the following.

```
>> wt(3)
```

```
ans =
    -5
```

Matlab will allow you to look at specific parts of the vector. The notation is the same as that used to create the vector.

```
>> wt(1:3)'
```

```
ans =
 -2.0000  -3.5000  -5.0000
```

```
>> wt(1:2:4)
```

```
ans =
    -2
    -5
```

Matrix operations

For the most part Matlab follows the standard notation used in linear algebra. We will see later that there are some extensions to make some operations easier. For now, though, both addition subtraction are defined in the standard way.

```
>> u = [0:-1:-4]; v = [2:0.5:4]; w = u + v
```

```
w =
    2.0000    1.5000    1.0000    0.5000         0
```

```
>> u - v
```

```
ans =
 -2.0000  -3.5000  -5.0000  -6.5000  -8.0000
```

Note when a variable is not assigned to a calculation, Matlab automatically assigns a generic variable `ans` to it.

```
>> wt = ans'
```

```
wt =
-2.0000
-3.5000
-5.0000
-6.5000
-8.0000
```

Scalar multiplication is defined in the standard way. Also note that scalar division is defined in a way that is consistent with scalar multiplication.

```
>> -2*u

ans =
    0     2     4     6     8

>> v/3

ans =
    0.6667    0.8333    1.0000    1.1667    1.3333

3*v-u/2

ans =
    6     8    10    12    14
```

You will need to be careful. These operations can only be carried out when the dimensions of the vectors allow it. You will likely get used to seeing the following error message which follows from adding two vectors whose dimensions are different.

```
>> u + v'

??? Error using ==> plus
Matrix dimensions must agree.
```

Matrix multiplication for matrices of compatible sizes is defined using the usual multiplication symbol.

```
>> X = w'*v

X =
    4.0000    5.0000    6.0000    7.0000    8.0000
    3.0000    3.7500    4.5000    5.2500    6.0000
    2.0000    2.5000    3.0000    3.5000    4.0000
    1.0000    1.2500    1.5000    1.7500    2.0000
         0         0         0         0         0

>> wt*w

ans =
   -17.5000

>> w*v

??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Matrix power and division

For square matrices Matlab defines a matrix exponentiation (^) operation called the matrix power. X^p is X to the power p , if p is a scalar. If p is an integer, the power is computed by repeated squaring. If the integer is negative, X is inverted first. For other values of p , the calculation involves eigenvalues and eigenvectors.

```
>> X^3

ans =
 1.0e+003 *
    0.6250    0.7813    0.9375    1.0938    1.2500
    0.4688    0.5859    0.7031    0.8203    0.9375
    0.3125    0.3906    0.4688    0.5469    0.6250
    0.1563    0.1953    0.2344    0.2734    0.3125
         0         0         0         0         0
```

```
>> w^3
```

```
??? Error using ==> mpower
Matrix must be square.
```

Matlab defines two different operators with symbols similar to the usual division. The slash (/) defines matrix right division. B/A is roughly the same as $B \cdot \text{inv}(A)$. More precisely, $B/A = (A \backslash B)'$ where the backslash (\) or matrix left division is defined as follows. If A is a square matrix, $A \backslash B$ is roughly the same as $\text{inv}(A) \cdot B$, except it is computed in a different way. If A is an n -by- n matrix and B is a column vector with n components, or a matrix with several such columns, then $X = A \backslash B$ is the solution to the equation $AX = B$ computed by Gaussian elimination. If A is an m -by- n matrix with $m \approx n$ and B is a column vector with m components, or a matrix with several such columns, then $X = A \backslash B$ is the solution in the least squares sense to the under- or overdetermined system of equations $AX = B$. These operations will be discussed in more detail when discussing solution of systems of equations.

```
>> X/w
```

```
ans =
 3.3333
 2.5000
 1.6667
 0.8333
 0
```

```
>> X \ w
```

```
??? Error using ==> mldivide
Matrix dimensions must agree.
```

Array operations

There are many times where we want to do an operation to every element in a vector or matrix. Matlab allows these "element-wise" operations and calls them array operations. For example, suppose you want to multiply each entry in vector u with its corresponding entry in vector v . In other words, suppose you want to find $u(1) \cdot v(1)$, $u(2) \cdot v(2)$, and so on. Since the symbol "*" is already used to indicate the usual matrix multiplication, the programmers who came up with Matlab decided to use the symbols ".*" to do this. Similarly you can put a period in front of / and ^ to tell Matlab that you want the operations to take place on each entry of the vector.

```
>> u = [0:-1:-4]; v = [2:0.5:4]; w = u .* v

w =
     0    -2.5000    -6.0000   -10.5000   -16.0000

>> u.\v
```

```

ans =
    Inf    -2.5000    -1.5000    -1.1667    -1.0000
>> u./v
ans =
     0    -0.4000    -0.6667    -0.8571    -1.0000
>> w.^3
ans =
  1.0e+003 *
     0    -0.0156    -0.2160    -1.1576    -4.0960

```

The built-in math functions also perform operations element-wise. If you pass a vector to a predefined math function, it will return a vector of the same size, and each entry is found by performing the specified operation on the corresponding entry of the original vector.

```

>> su = sin(u*pi/180)

su =
     0    -0.0175    -0.0349    -0.0523    -0.0698
>> asin(su)*180/pi

ans =
     0     -1     -2     -3     -4

```

The ability to work with these vector functions is one of the advantages of Matlab. Now complex operations can be defined that can be done quickly and easily. In the following example a very large vector is defined and can be easily manipulated. Notice that the commands have a ";" at the end of the line to suppress resulting large output. Only the first 10 elements of y are shown.

```

>> x = [0:0.1:100];
>> y = sin(x).*x ./ (1+cos(x));
>> y(1:10)

ans =
  Columns 1 through 7
     0    0.0050    0.0201    0.0453    0.0811    0.1277    0.1856
  Columns 8 through 10
     0.2555    0.3382    0.4347

```

Use of array operations to evaluate and plot functions

Array operations are useful for generating tables of values by evaluating a given function at specified intervals. These values can then be plotted using the **plot** command in Matlab.

Example

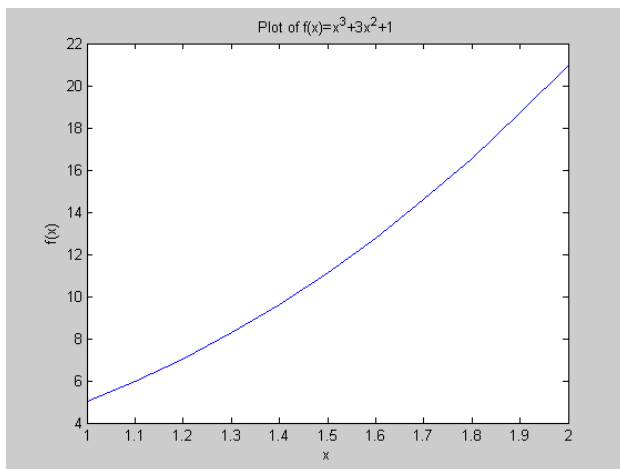
Evaluate the following function for x values indicated.

$f(x) = x^3 + 3x^2 + 1$ for x from 1 to 2 in steps of 0.1.

```
x = [1:0.1:2];
fx = x.^3 + 3* x.^2 + 1;
fx(1:10)

ans =
  Columns 1 through 7
    5.0000    5.9610    7.0480    8.2670    9.6240   11.1250   12.7760
  Columns 8 through 10
    14.5830   16.5520   18.6890
```

```
plot(x,fx)
xlabel('x')
ylabel('f(x)')
title('Plot of f(x)=x^3+3x^2+1')
```



Class Activity 2 - Evaluating and plotting functions of single variables

1. Evaluate the following functions for values of the independent variables indicated.

$$f(y) = \sin(\cos^{-1}(y)) \text{ for } y \text{ value of } \sqrt{3}/2.$$

$$y(u) = 1/u + u^3 / (u^4 + 5u \sin(u)) \text{ for } u \text{ values of } 1, \pi/2, 3\pi, \text{ and } 10\pi.$$

2. Evaluate and plot the following functions for values of the independent variables indicated.

$$g(x) = (\sin x)^2 \text{ for } x \text{ from } 0 \text{ to } 3\pi \text{ in steps of } 0.2\pi.$$

$$p(t) = \log(t^2) \text{ for } t \text{ from } 0.1 \text{ to } 2 \text{ in steps of } 0.1.$$