# INTRODUCTION TO SCIENTIFIC & ENGINEERING COMPUTING
# BIL 108E, CRN24023

Dr. S. Gökhan Karaman

Technical University of Istanbul

February 15, 2010



## USING MATLAB

To start from Windows,

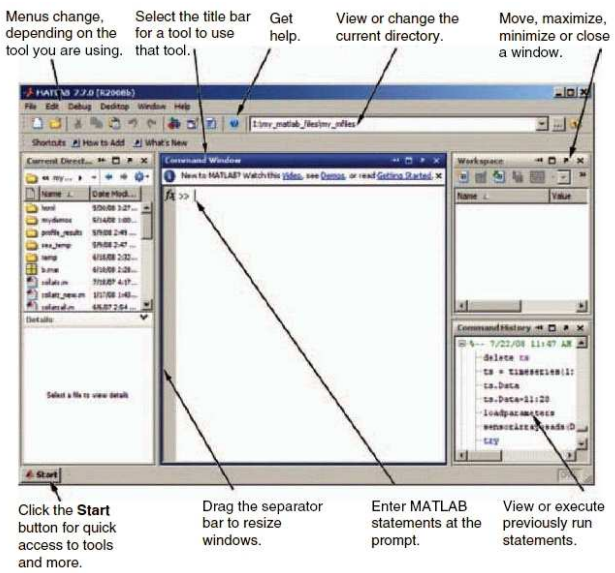
- Double click the Matlab icon.

To start from UNIX,

- type `matlab` at the shell prompt.

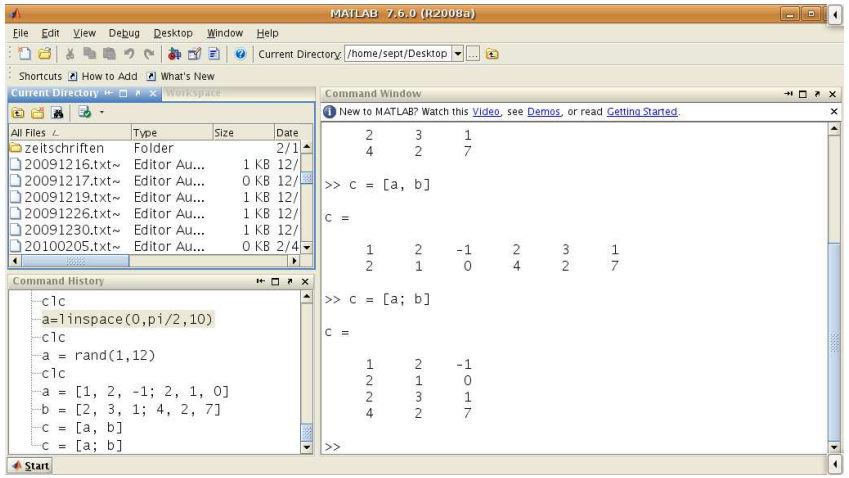


## MATLAB Desktop

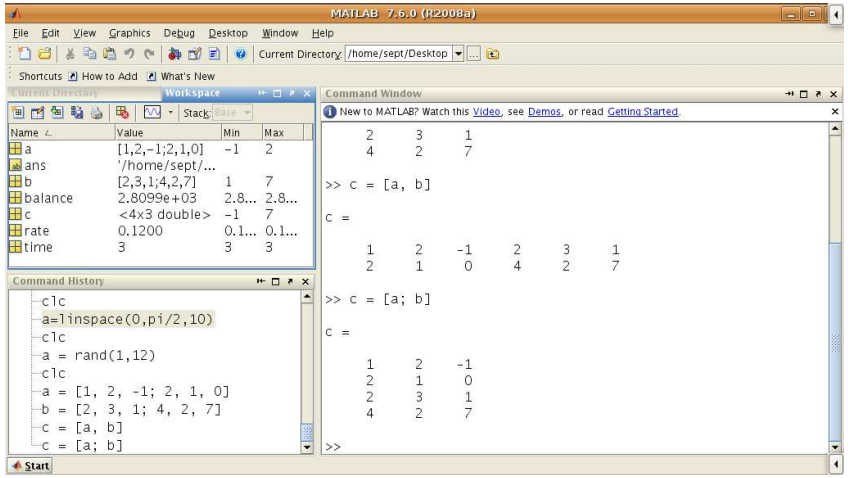





## MATLAB Desktop

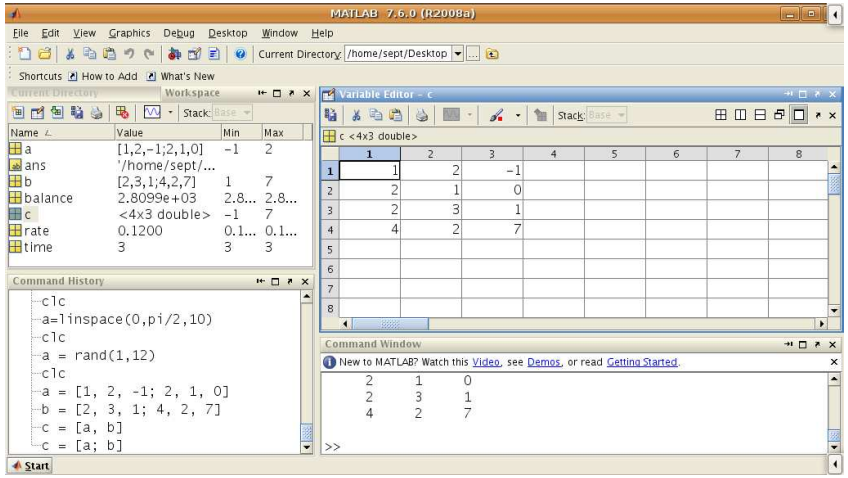


# MATLAB Desktop




Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# MATLAB Desktop




Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# USING MATLAB

To end Matlab session,

- From **File** pulldown menu, select **Exit MATLAB**.
- Enter `exit` or `quit` at the command prompt

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# COMMAND LINE

- $\gg$ indicates the command prompt
- You can edit a MATLAB command before pressing Enter(executing or running) by using various combinations of the Backspace, Left-arrow, Right-arrow, and Del keys.
- You can select (and edit) commands you have entered using Up-arrow and Down-arrow.
- MATLAB has a useful editing feature called smart recall. Just type the first few characters of the command you want to recall. For example, type the characters 2* and press the Up-arrow key. This recalls the most recent command starting with 2*.

# ARITHMETICS



Matlab command prompt can be used as a calculator.

```
>> 8 + 9
>> 24 - 12
>> 8 ^ 2
>> 1 / 16
>> 16 \ 1
```

Backslash means the denominator is to the left of the symbol.

# ARITHMETICS



Period in front of the operators means that the operation is done with single numbers.

```
>> 2 .* 6
>> 1 ./ 8
>> 3 .^ 4
>> 5 .^ 2
```

It is important, when we deal with array of numbers.

# ARITHMETIC OPERATORS



Matlab Operations

| Symbol | Operation | Example | Answer |
|---|---|---|---|
| + | Addition | $z = 4 + 2$ | $z = 6$ |
| − | Subtraction | $z = 4 - 2$ | $z = 2$ |
| / | Right division | $z = 4/2$ | $z = 2$ |
| \ | Left division | $z = 2\backslash 4$ | $z = 2$ |
| * | Multiplication | $z = 4 * 2$ | $z = 8$ |
| ^ | Exponentiation | $z = 4\,\hat{}\,2$ | $z = 16$ |
| Functions such as: sqrt, log | square root log2 | $z =$ sqrt(4) | $z = 2$ |
| | | $z =$ log2(4) | $z = 2$ |

# RELATIONAL OPERATORS



| Operator Name | Operator Symbol | EXAMPLE |
|---|---|---|
| less than | < | x<y |
| less than equal to | <= | a<=22 |
| equal to | == | x==100 |
| not equal to | ~= | x~=10 |
| greater than equal to | => | pi=>3 |
| greater than | > | c>100 |

# OPERATOR PRESEDENCE



| Precedence | Operators |
|---|---|
| 1. | (, ) |
| 2. | ^, .^, ', .' (pure transpose) |
| 3. | + (unary plus), -(unary minus), ~ (NOT) |
| 4. | *, \, /, .*, ./, .\ |
| 5. | + (addition), - (subtraction) |
| 6. | : |
| 7. | >, <, >=, <=, ==, ~= |
| 8. | & (AND) |
| 9. | \| (OR) |

Operator presedence from this table and from left to right.

# NUMBERS



Numbers can be defined in the decimal form

Example;

`1.732050808, -24, 256.0`

In scientific notation e or E could be used to define the exponent. Exponent should be an integer.

The mantissa is multiplied by the power of 10 indicated by the exponent. Example;

`12.35e-3`
$12.35x10^{-3}$

# PRECISION FORMATS



Precision Formats

| MATLAB Instruction | Display | Numerical Output exp(1) |
|---|---|---|
| format short | 4 decimal digits (default) | 2.7183 |
| format long | 16 decimal digits | 2.71828182845905 |
| format short e | 4 decimal digits plus exponent | 2.7183e+000 |
| format long e | 15 decimal digits plus exponent | 2.71828182845904e+000 |
| format bank | 2 decimal digits | 2.72 |
| format + | +, −, 0 (positive, negative, and zero) | + |
| format hex | Hexadecimal | 4005bf0a8b14576a |
| format rat | Rational approximation | 1457/536 |
| format compact | Suppress extra line-feeds | 2.7183 |
| format loose | Puts the extra line-feeds back in | 2.7183 |

# DATA TYPES



Default numeric data type is double precision.

Matlab has 14 data types.

Examples:

`integer, unsigned integer, string, single precision`

# SPECIAL VALUES

Matlab warns you in case of errors, but still gives answer.

Example;

- `1/0`
  `Inf`
- `0/0`
  `NaN`

You can use these symbols in any calculation.

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# VARIABLES

Variable Naming Rules

- It may consist only of the letters a–z, the digits 0–9, and the underscore _.
- It must start with a letter
- The name will be as long as you like but Matlab remembers only the first 63 characters
- Matlab is case sensitive, upper and lower case variables are not the same.

Examples;

`r2d2, x3po, luke_filewalker` RIGHT

`_2d, luke-filewalker, balance$` WRONG

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# Good Naming Techniques

- Camel Caps (dayOfTheWeek, milleniumBug, StarWars)
- using underscore (star_wars, day_of_the_week)

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# RESERVED VARIABLES

List of Reserved Variable Names

| Variable | Description |
|---|---|
| *ans* | Temporary variable that stores the most recent answer. |
| *computer* | Returns the computer type. |
| *version* | MATLAB version. |
| *ver* | Returns the information about the license and version of the MATLAB package installed in your computer. |
| *license* | License information. |
| *pi* | The number $\pi = 3.14159\ldots$. |
| *exp*(1) | The value of $e = 2.71\ldots$. |
| *eps* | Represents the accuracy of floating point, the smallest possible positive number with a magnitude of the order of $10^{-10}$. |
| *realmin* | The smallest real positive number. |
| *realmax* | The largest real positive number. |
| *bitmax* | The largest positive integer, magnitude of $2^{53} - 1$. |
| *flops* | Counts of the floating-point operations. flop(0) starts the count of all algebraic operations such as $+, -, *, /$. |
| *inf* | Represents infinity, (1/0). |
| *nan* | Not a number, undefined (0/0). |
| *i or j* | The value of $\sqrt{-1}$. Denotes the imaginary part of a complex number. |
| *input* | Accepts information via keyboard. |
| *date* | Represents the current date as a string. For example, 25-Jul-00. |
| *clock* | Represents the current date and time as YYMMDDHHMMSS. |
| *beep* | Executes a beep sound. |
| *etime* ($T_f$, $T_i$) | Calculates elapse time in seconds between $T_i$ (initial) and $T_f$ (final). $T_i$ and $T_f$ are in vector form consistins of six elements (year month day hour minute second). |
| *tic, toc* | Measures the time between the `tic` and the `toc`. The `tic` starts the stopwatch, and the `toc` stops the stopwatch and outputs the elapsed time. |
| *cputime* | Total time of MATLAB used in seconds. |
| *Pause* | Stops executing a program momentarily. |
| *Pause*(*n*) | Stops executing a program during *n* seconds. |

# VARIABLES

To do arithmetic operations with the variables we should assign values to variables.

```
>> a = 7;
>> b = 8;
>> c = a + b;

>> t = 12; r = 32;
>> u = t * r;
```

Several commands can be seperated by comma or semicolon and output disabled with semicolon.



# VARIABLES

Names should not duplicate with built-in functions.

```
>> pi = 7;
>> sqrt(pi);
```

pi has a different value then expected.



# VARIABLES

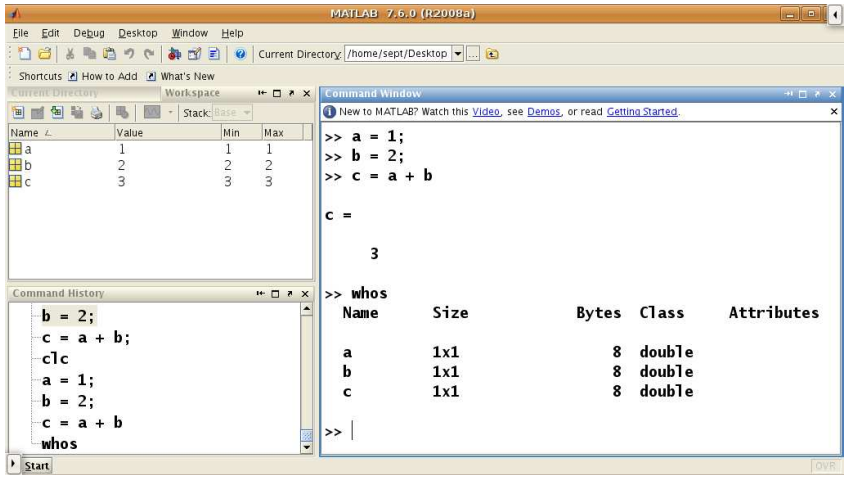
```
>> who
>> whos
>> clear pi
>> whos
>> sqrt(pi)
```

whos represent the locally defined variables and commands in the workspace with size info.

clear deletes the defined variable



# MATLAB Desktop



# VARIABLES

`save filename`

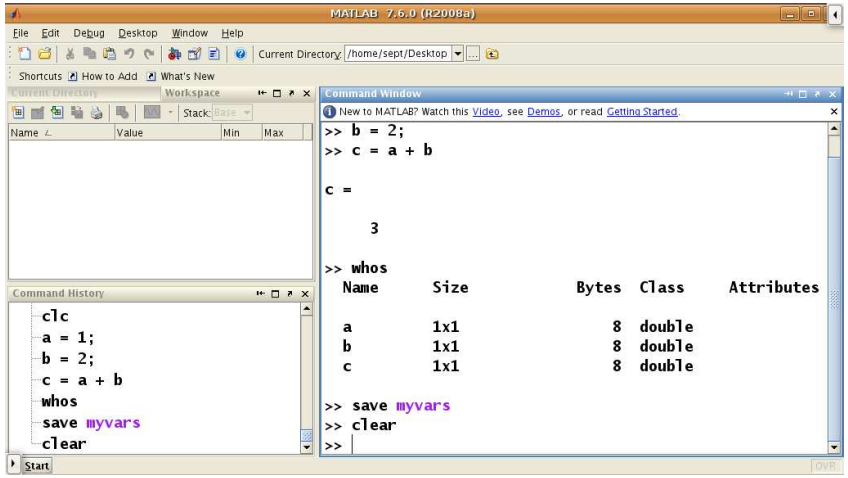
`load filename`

These commands are used to save and load the variables in the current workspace to a file.

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

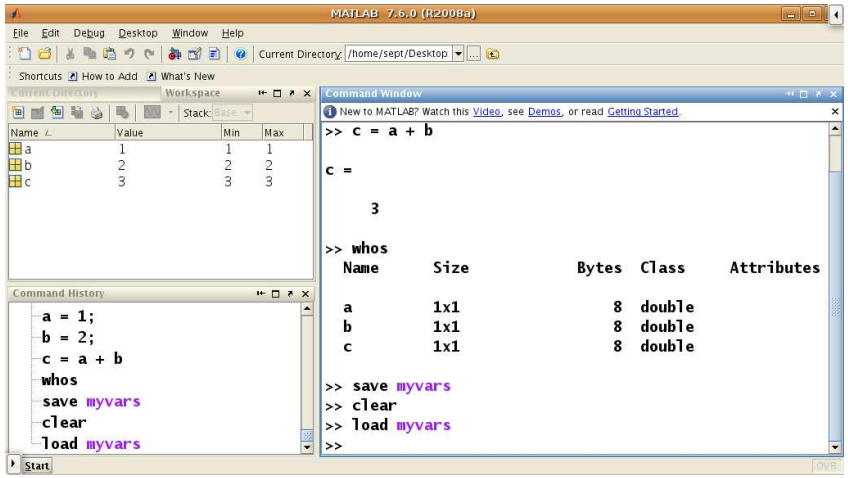
# MATLAB Desktop




Introduction to Scientific and Engineering Computing, BIL108E

Karaman

# MATLAB Desktop




Introduction to Scientific and Engineering Computing, BIL108E

Karaman

# GENERAL FUNCTIONS

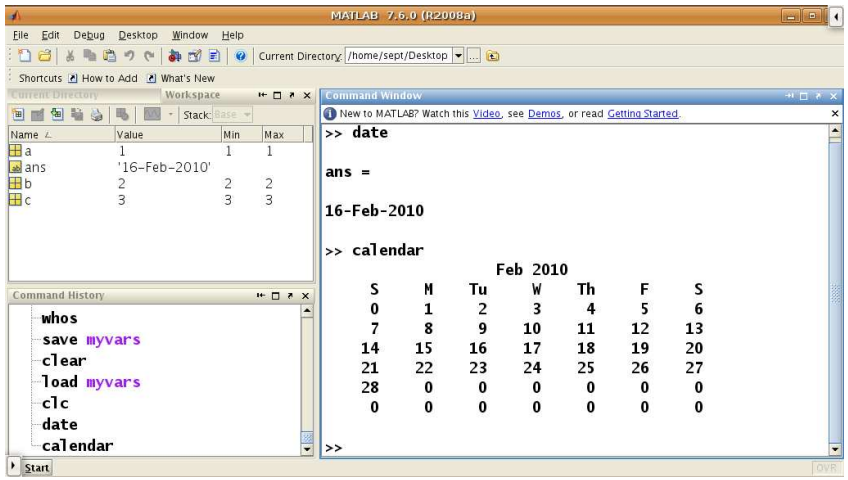
`date, calendar`

`clc` clear command window

`clf` clear figure window

`help`

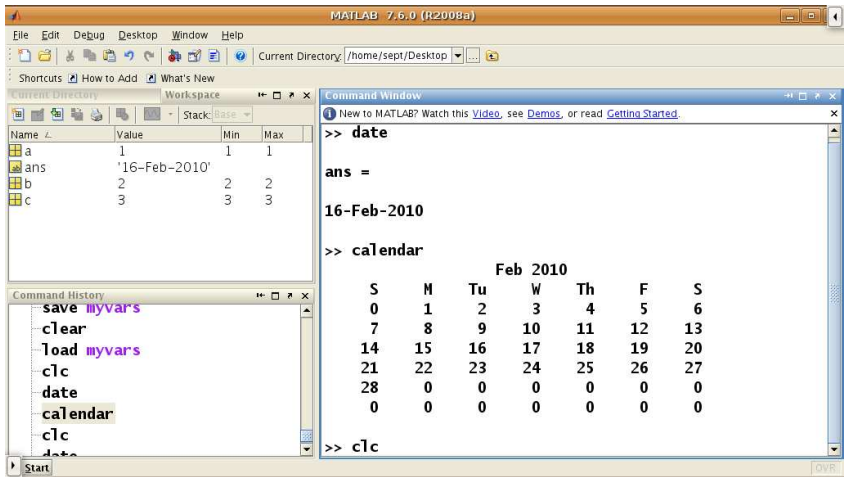
# MATLAB Desktop




Introduction to Scientific and Engineering Computing, BIL108E

Karaman

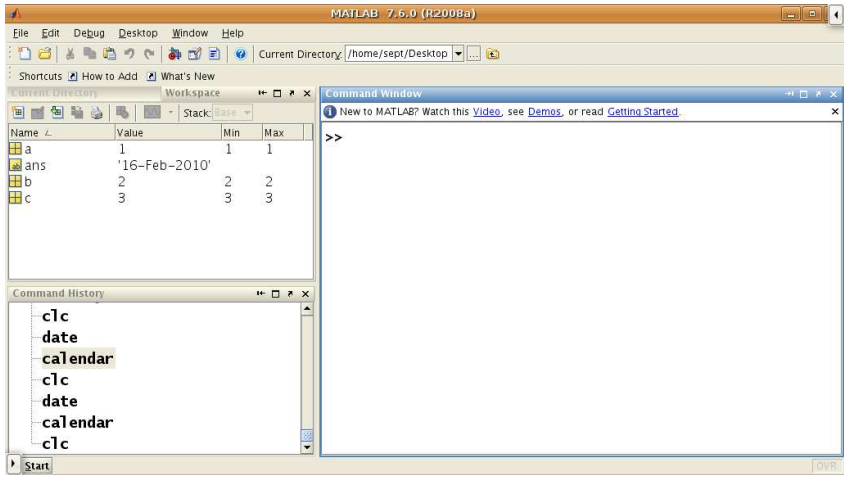
# MATLAB Desktop




Introduction to Scientific and Engineering Computing, BIL108E

Karaman

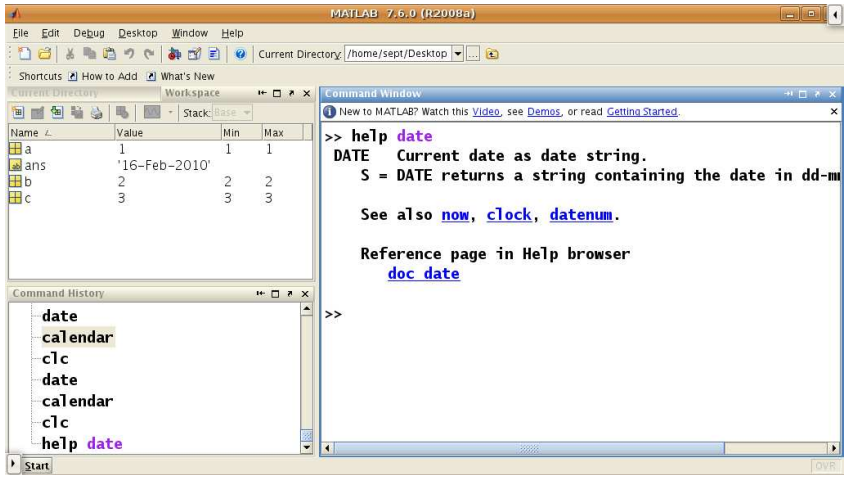
# MATLAB Desktop




Introduction to Scientific and Engineering Computing, BIL108E

Karaman

# MATLAB Desktop



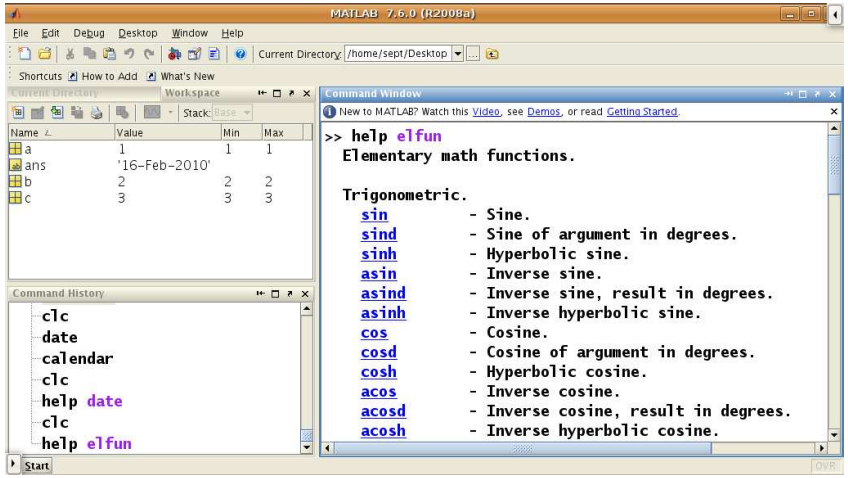
# BUILT–IN FUNCTIONS

MATLAB offers a wealth of built-in math functions that can be quite helpful for many computational problems

- Elementary MATLAB functions (help elfun)
  - Trigonometric functions
  - Exponential functions
  - Complex functions
  - Rounding and remainder functions
- Specialized MATLAB functions (help specfun)
  - Specialized math functions
  - Number theoretic functions
  - Coordinate transformations

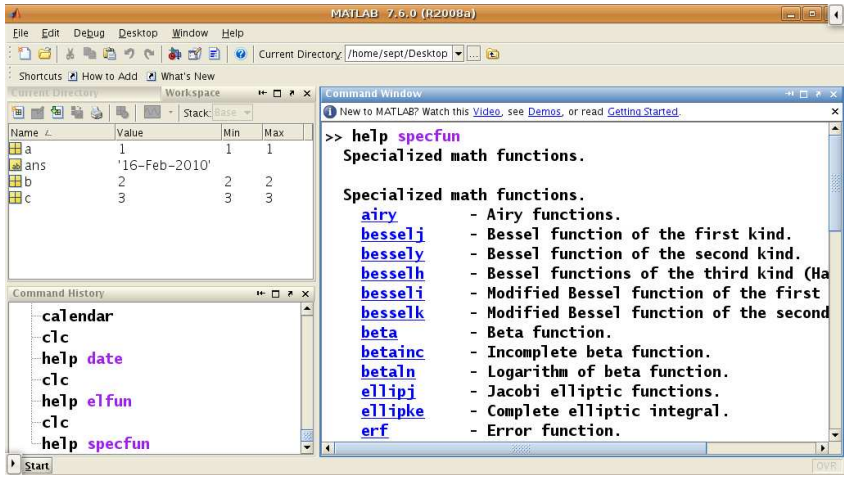
Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# MATLAB Desktop




Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# MATLAB Desktop




Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# MATHEMATICAL FUNCTIONS

| | |
|---|---|
| abs | Absolute value |
| acos, acosh | Inverse cosine and inverse hyperbolic cosine |
| acot, acoth | Inverse cotangent and inverse hyperbolic cotangent |
| acsc, acsch | Inverse cosecant and inverse hyperbolic cosecant |
| angle | Phase angle |
| asec, asech | Inverse secant and inverse hyperbolic secant |
| asin, asinh | Inverse sine and inverse hyperbolic sine |
| atan, atanh | Inverse tangent (two quadrant) and inverse hyperbolic tangent |
| atan2 | Inverse tangent (four quadrant) |
| bessel | Bessel function |
| ceil | Round up |
| conj | Complex conjugate |
| cos, cosh | Cosine and hyperbolic cosine |
| cot, coth | Cotangent and hyperbolic cotangent |
| csc, csch | Cosecant and hyperbolic cosecant |

# MATHEMATICAL FUNCTIONS cont'd

| | |
|---|---|
| erf | Error function |
| exp | Exponential |
| fix | Round toward zero |
| floor | Round down |
| gamma | Gamma function |
| imag | Imaginary part |
| log | Natural logarithm |
| log2 | Dissect floating point numbers into exponent and mantissa |
| log10 | Common logarithm |
| mod | Modulus (signed remainder after division) |
| rat | Rational approximation |
| real | Real part |
| rem | Remainder after division |
| round | Round toward nearest integer |
| sec, sech | Secant and hyperbolic secant |
| sign | Signum function |
| sin, sinh | Sine and hyperbolic sine |
| sqrt | Square root |
| tan, tanh | Tangent and hyperbolic tangent |

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# LINEAR EQUATIONS

$$2x - y = 4$$

$$-x + 2y = 3$$

$$\begin{pmatrix} 2 \\ -1 \end{pmatrix} x + \begin{pmatrix} -1 \\ 2 \end{pmatrix} y = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

Column picture

Scalars are not enough to define this kind of data.

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# VECTORS

A vector is an ordered list of numbers (one-dimensional). In MATLAB they can be represented as a row-vector or a column-vector (1 x n) or (n x 1).

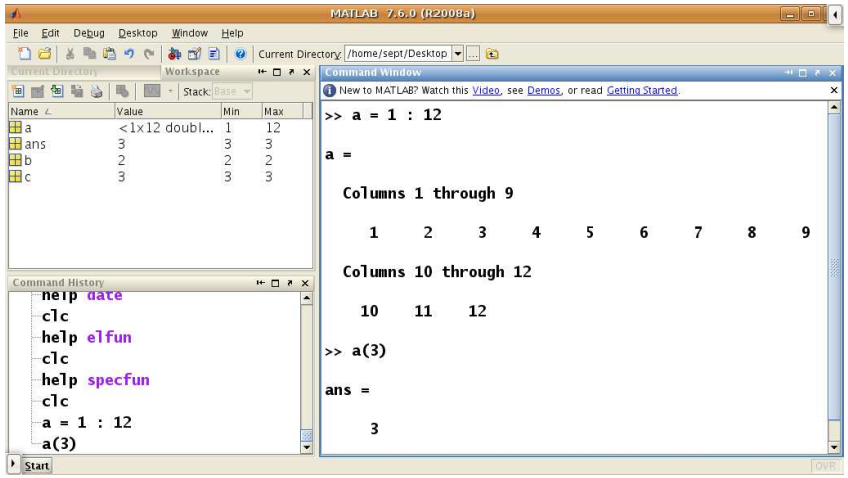
Simple vector definition;
*COLON* **:** is used to define row vectors in Matlab.

```
>> a = 1 : 12;
>> size(a)
```

`size` command shows dimension of the variable, here the size of vector *a*.

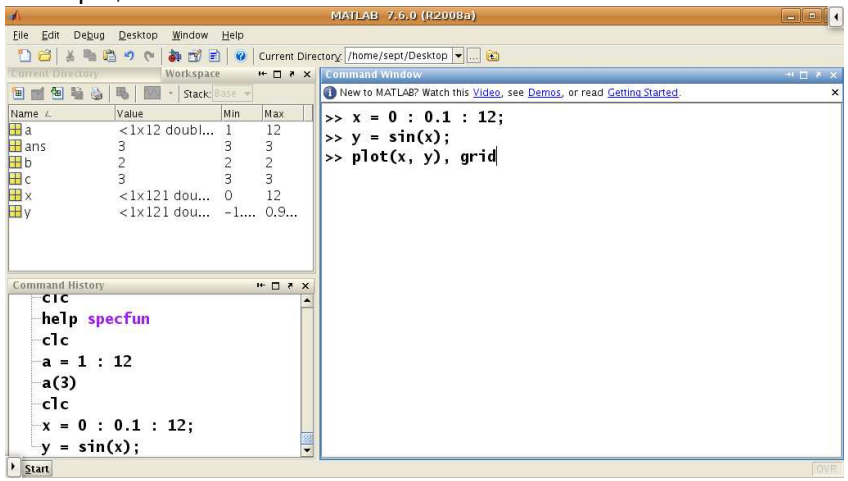
Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# MATLAB Desktop



# VECTORS

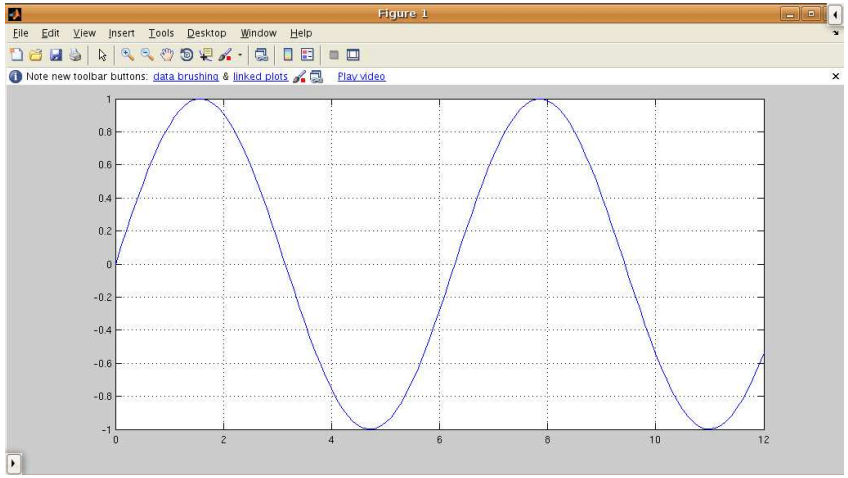
Example;




Introduction to Scientific and Engineering Computing, BIL108E

Karaman

# VECTORS

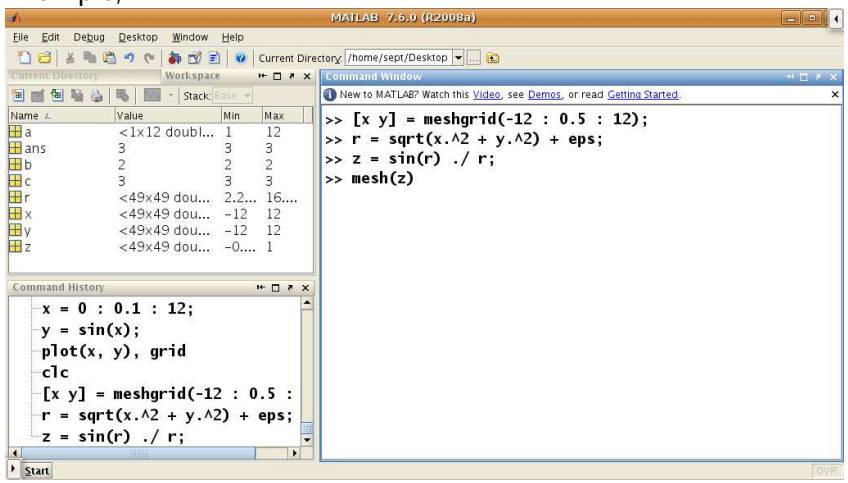



Introduction to Scientific and Engineering Computing, BIL108E

Karaman

# EXAMPLES

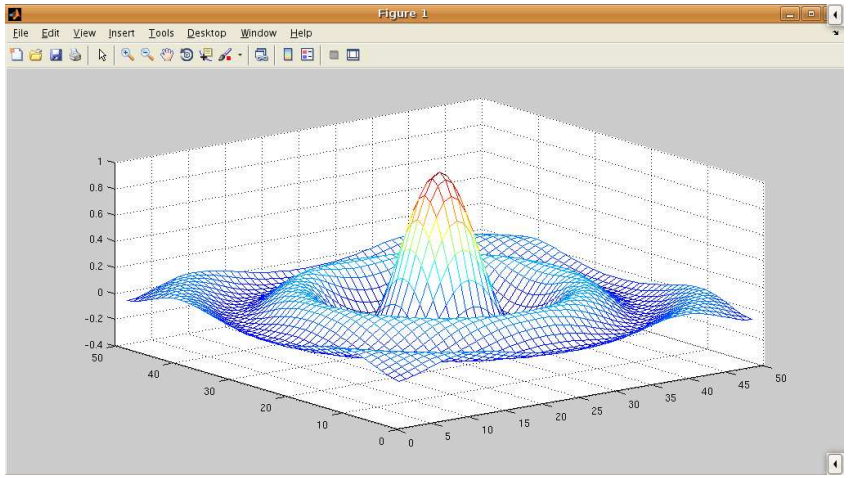
Example;



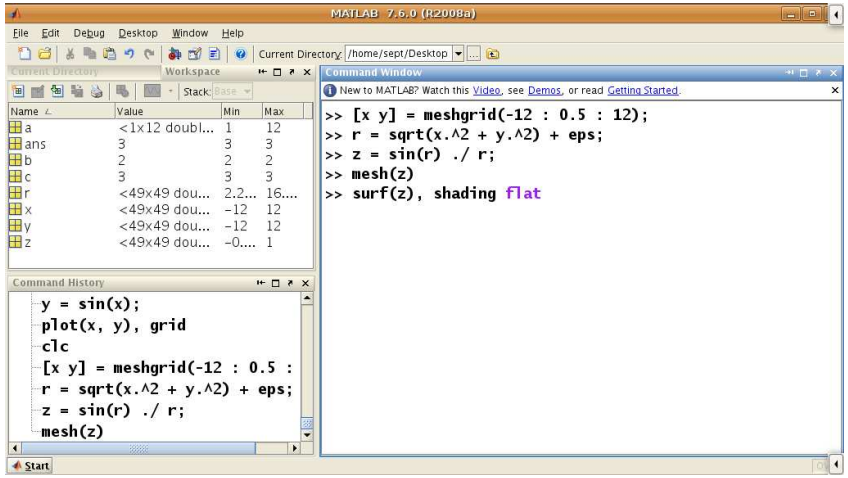

Introduction to Scientific and Engineering Computing, BIL108E

Karaman

# EXAMPLES

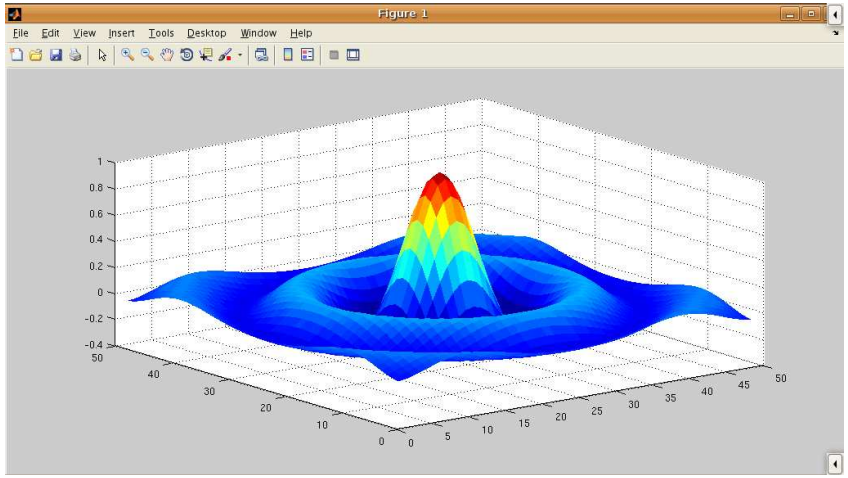


# EXAMPLES




Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# EXAMPLES




Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# EXAMPLES

Given : Balance 2000 USD, 12% rate per year

Find : Bank Balance after 3 years?

Formula : $balance(1 + r)^n$

First write down a rough algorithm.

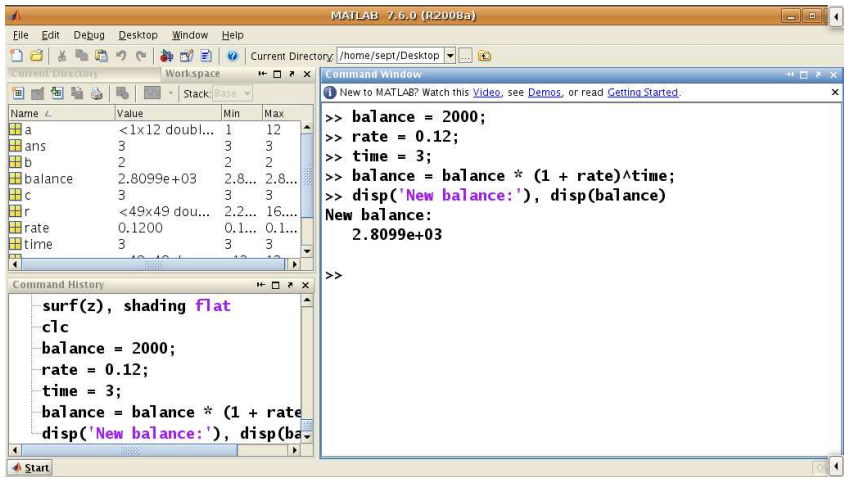
1. Get the data into Matlab.
2. Calculate the balance after 3 years
3. Display the new balance

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# EXAMPLES

```
balance = 2000; % USD
rate = 0.12; % bank rate
time = 3; % years
balance = balance * (1 + rate)^time;
disp('New balance:');
disp(balance)
```

# MATLAB Desktop





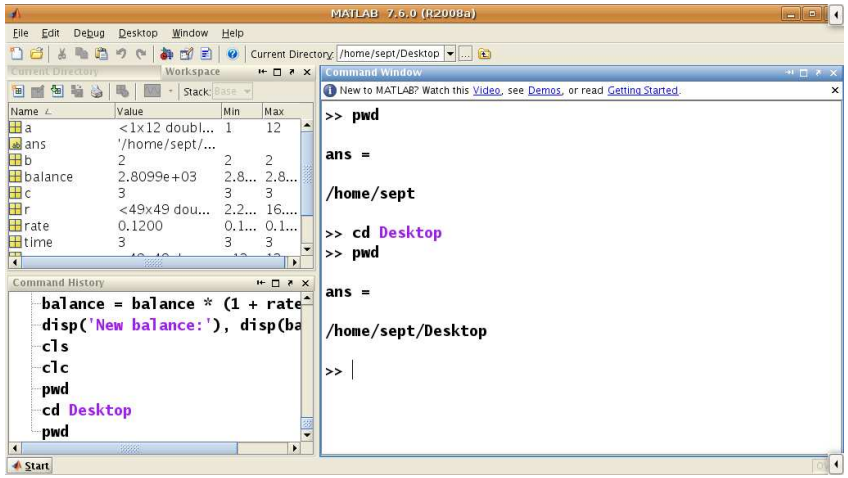

# RUNNING SCRIPT FILES

- Use file extension **.mat** for Workspace files and can be opened from Matlab Desktop File menu.
- Use file extension **.m** for function files

- `pwd` shows the current directory of matlab
- `cd` change directory to run the script file
- change directory with Matlab desktop Directory Browser.



# MATLAB Desktop






# `linspace` FUNCTION

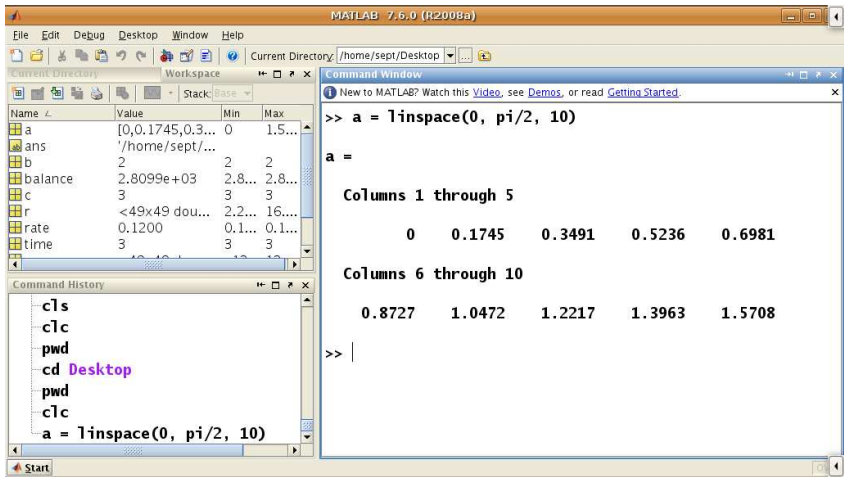
**The function `linspace` creates a vector of equally spaced values.**

Example;

`linspace(0 , pi/2, 20)`

creates a vector with the size of 1 x 20 from 0 to $pi/2$

# MATLAB Desktop






# TRANSPOSE OF A VECTOR

`a = [ 1 2 3 5 8 13 ]` is a row vector.

"[" and "]" used to define a vector.

To generate the column vector, transpose the vector.

`b = [ 1 2 3 5 8 13 ]' (')` apostrophe is used to transpose vector a. Size of a is 1 x n

Size of b is n x 1

In mathematics a column vector is shown as $a^i$
and a row vector is shown as $a_j$.
A matrix could be shown as $a_{ij}$, $ora^i_j$



# SUBSCRIPT

`rand(i,j)` command is used for creating a random variable matrix with values between 0 and 1.

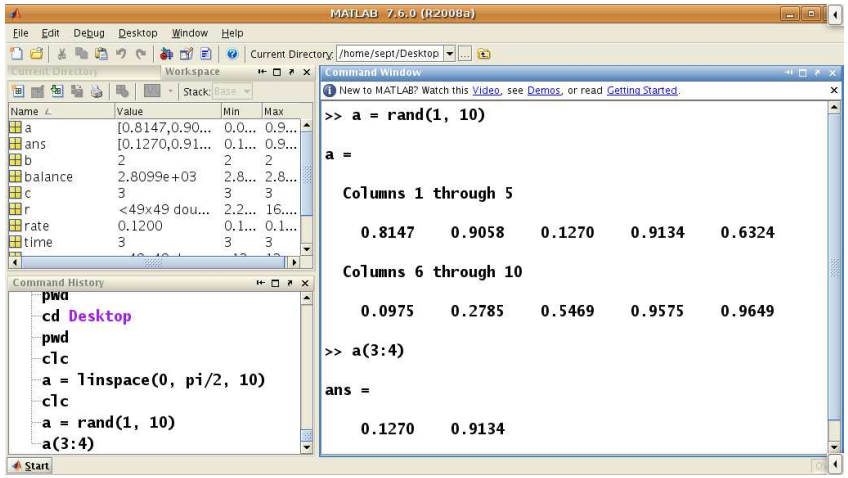
`a=rand(1,12)` creates a 1x12 row vector.

a(3) gives the third element of the vector

a(3:5) gives the elements between 3 and 5



# MATLAB Desktop



# MATRIX

A matrix is a rectangular array of numbers (multidimensional). In MATLAB, a two-dimensional matrix is defined by its number of rows and columns (n x m) or (m x n).

A matrix can be created like a vector.

Examples;

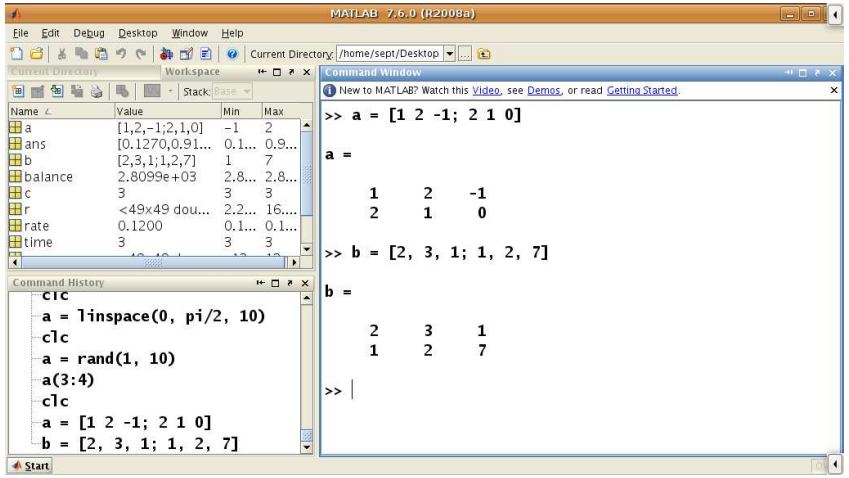
- a = [1 2 -1; 2 1 0]
- b = [2, 3, 1 ; 4, 2, 7]

A matrix can also be constructed from other matrices.

- c = [a , b]
- c = [a ; b]

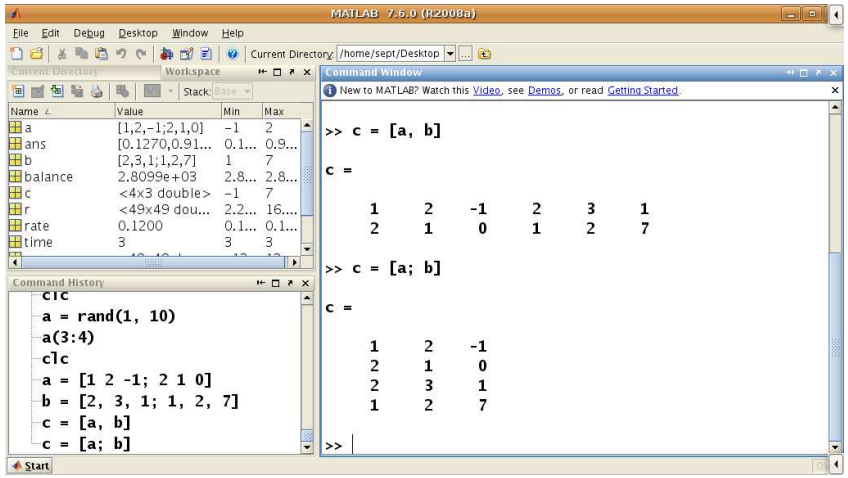


# MATLAB Desktop






# MATLAB Desktop






# Capturing Output

`diary` *FILENAME*

It creates a file with the name *FILENAME* and appends all the output to this file till we end it with the command

`diary off`

# VERTICAL MOTION UNDER GRAVITY



## EXAMPLE: VERTICAL MOTION UNDER GRAVITY

If a stone is thrown vertically upward with an initial speed $u$ its vertical displacement $s$ after an elapsed time $t$ is given by the formula $s = ut - gt^2/2$, where $g$ is the acceleration due to gravity. Air resistance is ignored.

We would like to compute the value of $s$ over a period of about 12.3 seconds at intervals of 0.1 seconds, and plot the distance versus time graph over this period.

# VERTICAL MOTION UNDER GRAVITY



```
% Assign the data (g, u, and t) to MATLAB variables
% Calculate the value of s according to the formula
% Plot the graph of s against t
% Stop
```

# VERTICAL MOTION UNDER GRAVITY



This plan may seem trivial and a waste of time to write down. Yet you would be surprised how many beginners, preferring to rush straight to the computer, start with step 2 instead of step 1. It is well worth developing the mental discipline of structure–planning your program first. You can even use cut and paste to plan as follows:



1. Type the structure plan into the Editor
2. Paste a second copy of the plan directly below the first.
3. Translate each line in the second copy into a MATLAB statement or statements
4. Finally, paste all the translated MATLAB statements into the Command Window and run them
5. If necessary, go back to the Editor to make corrections

# VERTICAL MOTION UNDER GRAVITY

```
% Vertical motion under gravity
g = 9.81; % acceleration due
          % to gravity
u = 60; % initial velocity in
        % metres/sec
t = 0 : 0.1 : 12.3;          % time in seconds
s = u * t - g / 2 * t .^ 2; % vertical displacement
                             % in metres
plot(t, s), title( 'Vertical motion under gravity' )
xlabel( 'time' ), ylabel( 'vertical displacement' )
grid
disp( [t' s'] )              % display a table
```

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# VERTICAL MOTION UNDER GRAVITY

1. Anything in a line following the symbol % is ignored by MATLAB and may be used as a comment (description).
2. The statement `t = 0 : 0.1 : 12.3` sets up a vector.
3. The formula for $s$ is evaluated for every element of the vector $t$, making another vector.
4. The expression `t.^ 2` squares each element in `t`. This is called an array operation and is different from squaring the vector itself.

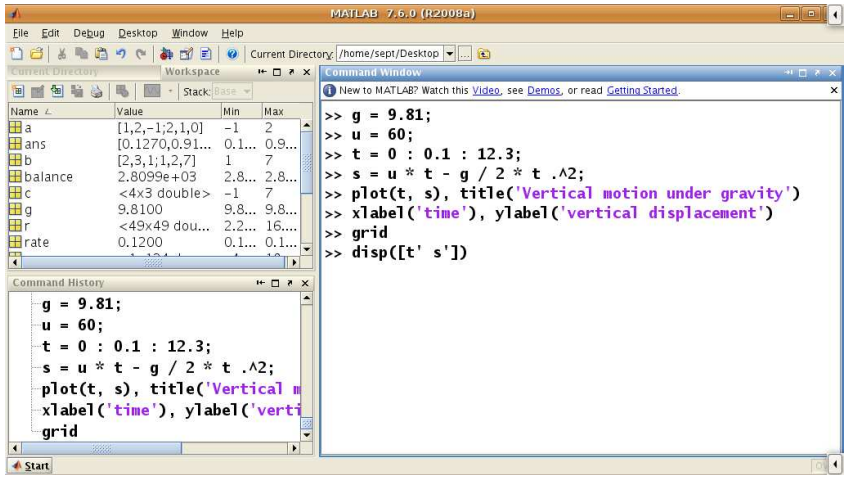
Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# VERTICAL MOTION UNDER GRAVITY

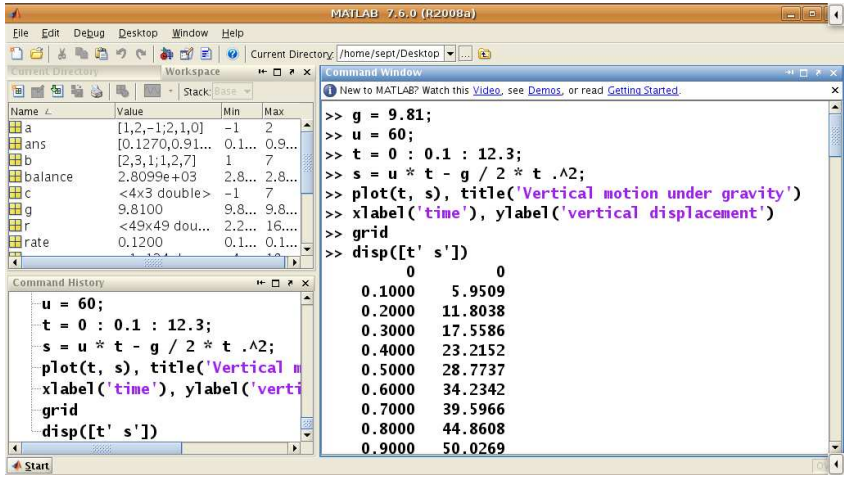
5. More than one statement can be entered on the same line if the statements are separated by commas.
6. A statement or group of statements can be continued to the next line with an ellipsis of three or more dots `(...)`.
7. The statement `disp([t' s'])` first transposes the row vectors $t$ and $s$ into two columns and constructs a matrix from them, which is then displayed.

Introduction to Scientific and Engineering Computing, BIL108E
Karaman

# VERTICAL MOTION UNDER GRAVITY



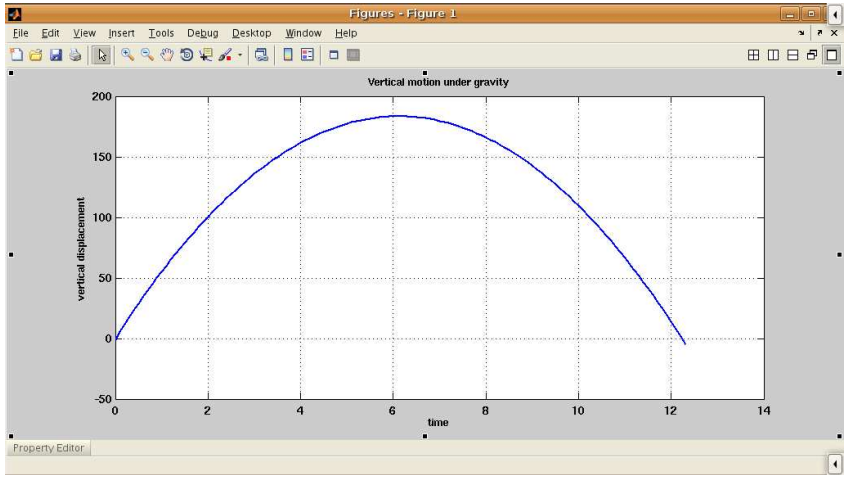
# VERTICAL MOTION UNDER GRAVITY




Introduction to Scientific and Engineering Computing, BIL108E

Karaman

# VERTICAL MOTION UNDER GRAVITY




Introduction to Scientific and Engineering Computing, BIL108E

Karaman

# References

References for Week 2

1. Brian Hahn, Daniel T.Valentine, Essential Matlab for Engineers and Scientists, Elsevier, 2010.
2. Misza Kalechman, Practical Matlab Basics for Engineers, CRC Press, 2009.