Chapter 2. Development Environment and Basic Operations
Outline

• 2.1 Introduction
• 2.2 MATLAB Development Environment
  – Prior to R2012b
  – After R2012b
• 2.3 Help System
• 2.4 Basic Operations and Order of Precedence
• 2.5 Basic Functions and Utilities
• 2.6 Managing Work Session
• 2.7 For Advanced Users
  – Changing Preferences
  – Analyzing Code in Current Folder
  – Profiling
• 2.8 Problem-Solving Methodology
Starting MATLAB

MATLAB 7.0.0

MATLAB R2011a

MATLAB R2014b

MATLAB R2019b

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Desktop Prior to R2012b

Start Button – can be used to launch toolboxes, MATLAB tools, and so on

Current Directory browser – shows a list of the files in the current directory

Workspace browser – shows variables defined (created) in workspace

Command Window – the window you type your commands in and get responses

workspace

commandhistory

Command History window – displays previously issued commands

Minimize

Restore

Undock

Close

GUIDE

Profiler

Help Browser

Simulink

File Browser – shows a list of the files in the current directory
Managing MATLAB Desktop

Tabs
Quick Access Toolbar
Search window
Toolstrip
MATLAB Drive

Tabs:
- Home
- Apps
- Files
- Workspace
- Commands
- Environment
- Resources

Quick Access Toolbar:
- New Script
- New Live Script
- New File
- Import Data
- Save Workspace
- Open Workspace
- Variable View
- Clear Workspace
- Favorites
- Clear Commands
- Analyze Code
- Run and Time Code
- Add-Ons
- Help
- Learn MATLAB
- Get Started
- Set Path
- Preferences
- Run Files
- Learn Simulink
- Parallel
- Run
- Add-Ons
- Help
- Learn

Search window:
- Search
- Documentation

Toolstrip:
- MATLAB Drive
- Current folder
- Command window
- Workspace
- Command history
- File browser

Command window:
>> commandwindow
>> commandhistory
>> filebrowser
>> workspace

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Managing MATLAB Desktop

- Profiler
- Simulink
- Help Browser

GUIDE

Desktop layout

Add-Ons

Help options

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Command History Window

Tap “↑” key
Getting More Apps

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.

Engineering Computations and Modeling in MATLAB/Simulink
Typing Commands

- User input
- Result of calculation
- Suppressing the output
- Plotting options available for a selected variable
Path Setting

![MATLAB Path Setting](image)

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Creating Favorites

![Screenshot of MATLAB interface showing navigation to 'ClearStart' favorite command]

- **ClearStart** favorite command is added to the Favorites section.
- The command history includes:
  - `3 * pi / 3.141592 \ 3 - 1 + 2 ^ 2`
  - `z = [1 2 3 4] * 5 / 2 - 1`
  - `x = 3`
  - `x = x + 1`
  - `50 - ans`
  - `R = 10; circumference = 2 * pi * R`
  - `y = 2 * pi * 10`

---

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Workspace Window

Right click
Right click
Right click
Double click
Single click

Workspace

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Array Editor

Use Ctrl to select not adjacent cells

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
On-line Courses, Tutorials & Examples

>> demo

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Examples

bin2dec - Convert text representation of binary number to decimal number

This MATLAB function interprets binarystr, text that represents a binary number, and returns the equivalent decimal number.

Syntax

bin2dec(binarystr)

Description

bin2dec(binarystr) interprets binarystr, text that represents a binary number, and returns the equivalent decimal number. binarystr must represent a nonnegative integer value smaller than or equal to the value returned by flintmax.

Examples

Binary 0101111 converts to decimal 23:

```matlab
ans =
    23
```
Getting Quick Help

Push a Tab key to have your command finished.
Getting Quick Help²

Right click
This subdirectory (\MATLAB\R2018b\toolbox\matlab) contains M-file sources to many of the functions supplied with MATLAB.

>> help ops
>> help elfun
>> help specfun
>> help randfun
>> help elmat
>> help strfun
# Arithmetic Operations

## Matrix form

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>MATLAB Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>exponentiation, $a^b$</td>
<td>$a^b$</td>
</tr>
<tr>
<td>*</td>
<td>multiplication, $ab$</td>
<td>$a*b$</td>
</tr>
<tr>
<td>/</td>
<td>right division, $a/b$</td>
<td>$a/b$</td>
</tr>
<tr>
<td>\</td>
<td>left division, $a\backslash b$</td>
<td>$a\backslash b$</td>
</tr>
<tr>
<td>+</td>
<td>addition, $a+b$</td>
<td>$a+b$</td>
</tr>
<tr>
<td>-</td>
<td>subtraction, $a-b$</td>
<td>$a-b$</td>
</tr>
</tbody>
</table>

## Element wise form

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operation</th>
<th>MATLAB Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>exponentiation, $a^b$</td>
<td>$a.^b$</td>
</tr>
<tr>
<td>*</td>
<td>multiplication, $ab$</td>
<td>$a.*b$</td>
</tr>
<tr>
<td>/</td>
<td>right division, $a/b$</td>
<td>$a./b$</td>
</tr>
<tr>
<td>\</td>
<td>left division, $a\backslash b$</td>
<td>$a./b$</td>
</tr>
</tbody>
</table>

Order of precedence: Left to right, then top to bottom.
### Trigonometric functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sin )</td>
<td>sine</td>
</tr>
<tr>
<td>( \text{asin} )</td>
<td>inverse sine</td>
</tr>
<tr>
<td>( \cos )</td>
<td>cosine</td>
</tr>
<tr>
<td>( \text{acos} )</td>
<td>inverse cosine</td>
</tr>
<tr>
<td>( \tan )</td>
<td>tangent</td>
</tr>
<tr>
<td>( \text{atan} )</td>
<td>4-quadrant inverse tangent</td>
</tr>
<tr>
<td>( \text{atan2} )</td>
<td>4-quadrant inverse tangent</td>
</tr>
<tr>
<td>( \sec )</td>
<td>secant (1/cos(x))</td>
</tr>
<tr>
<td>( \text{asec} )</td>
<td>inverse secant</td>
</tr>
<tr>
<td>( \csc )</td>
<td>cosecant (1/sin(x))</td>
</tr>
<tr>
<td>( \text{acsc} )</td>
<td>inverse cosecant</td>
</tr>
<tr>
<td>( \cot )</td>
<td>cotangent (1/tan(x))</td>
</tr>
<tr>
<td>( \text{acot} )</td>
<td>inverse cotangent</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{sind} )</td>
<td>sine</td>
</tr>
<tr>
<td>( \text{asind} )</td>
<td>inverse sine</td>
</tr>
<tr>
<td>( \text{cosd} )</td>
<td>cosine</td>
</tr>
<tr>
<td>( \text{acosd} )</td>
<td>inverse cosine</td>
</tr>
<tr>
<td>( \tand )</td>
<td>tangent</td>
</tr>
<tr>
<td>( \text{atand} )</td>
<td>inverse tangent</td>
</tr>
<tr>
<td>( \text{secd} )</td>
<td>secant</td>
</tr>
<tr>
<td>( \text{asecd} )</td>
<td>inverse secant</td>
</tr>
<tr>
<td>( \cscd )</td>
<td>cosecant</td>
</tr>
<tr>
<td>( \text{acscd} )</td>
<td>inverse cosecant</td>
</tr>
<tr>
<td>( \cotd )</td>
<td>cotangent</td>
</tr>
<tr>
<td>( \text{acotd} )</td>
<td>inverse cotangent</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Funcion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sinh )</td>
<td>hyperbolic sine</td>
</tr>
<tr>
<td>( \text{asinh} )</td>
<td>inverse hyperbolic sine</td>
</tr>
<tr>
<td>( \cosh )</td>
<td>hyperbolic cosine</td>
</tr>
<tr>
<td>( \text{acosh} )</td>
<td>inverse hyperbolic cosine</td>
</tr>
<tr>
<td>( \tanh )</td>
<td>hyperbolic tangent</td>
</tr>
<tr>
<td>( \text{atanh} )</td>
<td>inverse hyperbolic tangent</td>
</tr>
<tr>
<td>( \text{sech} )</td>
<td>hyperbolic secant</td>
</tr>
<tr>
<td>( \text{asech} )</td>
<td>inverse hyperbolic secant</td>
</tr>
<tr>
<td>( \text{csch} )</td>
<td>hyperbolic cosecant</td>
</tr>
<tr>
<td>( \text{acsch} )</td>
<td>inverse hyperbolic cosecant</td>
</tr>
<tr>
<td>( \text{coth} )</td>
<td>hyperbolic cotangent</td>
</tr>
<tr>
<td>( \text{acoth} )</td>
<td>inverse hyperbolic cotangent</td>
</tr>
</tbody>
</table>

### Rounding and Remainder

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{fix} )</td>
<td>round towards zero</td>
</tr>
<tr>
<td>( \text{floor} )</td>
<td>round towards minus infinity</td>
</tr>
<tr>
<td>( \text{ceil} )</td>
<td>round towards plus infinity</td>
</tr>
<tr>
<td>( \text{round} )</td>
<td>round towards nearest integer (( \text{round}(X,N) ) rounds to ( N ) digits)</td>
</tr>
<tr>
<td>( \text{mod} )</td>
<td>modulus (signed remainder after division)</td>
</tr>
<tr>
<td>( \text{rem} )</td>
<td>remainder after division</td>
</tr>
<tr>
<td>( \text{sign} )</td>
<td>signum (returns 1 if the element is greater than 0, 0 - equals 0, and -1 - less than 0)</td>
</tr>
</tbody>
</table>
Examples of Using Functions

```matlab
>> tand(90)
ans =
    Inf
>> sin([0,0.5*pi,pi])
ans =
    0  1.0000  0.0000
>> floor([4.2 -4.2])
ans =
    4   -5
>> fix([4.3 -4.3])
ans =
    4   -4
>> mod(12,5)
ans =
    2
>> exp(1)
ans =
    2.7183
>> abs(3+4i)
ans =
    5
>> angle(2+2j)*180/pi
ans =
    45
>> bin2dec('101')
ans =
    5
>> num2str(pi,3)
ans =
    3.14
>> lower('MATLAB')
ans =
    matlab
>> km2deg(111.2)
ans =
    1.0000
>> clock
ans =
    1.0e+03 *
    2.0140  0.0110  0.0080  0.0230  0.0380  0.0536
>> datenum(2014,11,15,10,15,40)
ans =
    2.3592e+05
>> date
ans =
    26-Apr-2015
```
Variable Names

The valid variable name may contain 1–63 characters (defined by the `namelengthmax` function of MATLAB). It must start with a letter, but may use letters, digits, and underscore after the first letter (remember, MATLAB is case-sensitive). The only expectation is that the variable name cannot be one of 20 keywords (type in `iskeyword` to see what these keywords are).

### Examples of valid names:

```
Speed = 25
Speed = 25
>> speed = 30
speed = 30
>> SPEED_of_AC = 35
SPEED_of_AC = 35
>> speeD2 = 2*Speed
speeD2 = 50
```

### Invalid names:

```
x6 6x
lastValue end
n_factorial n!
```

Try to give your variables the meaningful names, so that anyone who looks at your code understands what it does.

You may check existence of variable, function, folder, or class by typing in

```
>> exist name
```
Embedded Functions²

Exponential functions

exp - exponential
log - natural logarithm
log10 - common (base-10) logarithm
log2 - base-2 logarithm and dissect floating point number
pow2 - base-2 power and scale floating point number
realpow - power that will error out on complex result
reallog - natural logarithm of real number
realsqrt - square root of number greater than or equal to zero
sqrt - square root
hypot - square root of sum of squares
nextpow2 - next higher power of 2

Complex functions

abs - absolute value
angle - phase angle
complex - construct complex data from real and imaginary parts
conj - complex conjugate
imag - complex imaginary part
real - complex real part
unwrap - unwrap phase angle
isreal - true for real array
cplxpair - sort numbers into complex conjugate pairs
Embedded Functions

Conversion functions

- `bin2dec` - converts binary string to decimal integer
- `dec2bin` - converts decimal integer to a binary string
- `dec2hex` - converts decimal integer to IEEE hexadecimal string
- `hex2dec` - converts hexadecimal string to decimal integer
- `num2hex` - converts singles and doubles to IEEE hexadecimal strings
- `hex2num` - converts hexadecimal number string to double-precision number
- `base2dec` - converts base-B string to decimal integer
- `dec2base` - converts decimal integer to base-B string

Different useful functions

- `factorial` - factorial function
- `rand` - generates uniformly distributed random numbers
- `randn` - generates normally distributed random numbers
- `randi` - generates uniformly distributed pseudorandom integers
- `lower` - convert string to lowercase
- `upper` - convert string to uppercase
- `int2str` - convert integer to string
- `num2str` - convert number to string
- `str2num` - convert string to number
Mapping Toolbox Functions

Angle conversions
- formats angle strings
- converts angles units or encodings
- converts degrees to deg:min, deg:min:sec and to radians
- converts deg:min:sec to deg, deg:min, rad, [deg min sec] matrix
- converts [deg min sec] matrix to deg:min:sec
- converts radians to degrees, deg:min and deg:min:sec
- converts strings to angles in degrees

Distance conversions
- converts degrees to kilometers, nautical miles, statute miles
- formats distance strings
- converts distance units
- converts kilometers to degrees, nautical miles, radians, and statute miles
- converts nautical miles to degrees, kilometers, radians and statute miles
- converts radians to kilometers, nautical miles and statute miles
- converts statute miles to degrees, kilometers, nautical miles and radians

Time conversions
- converts hrs:min:sec to hrs:min, hours, sec, [hrs min sec] matrix
- converts hours to hrs:min, hrs:min:sec and seconds
- converts [hrs min sec] matrix to hrs:min:sec
- converts time from seconds to hrs:min, hrs:min:sec and hours
- formats time strings
- converts time units or encodings

Coordinate conversions
- transforms Cartesian coordinates to polar/cylindrical or spherical
- transforms polar/cylindrical or spherical coordinates to Cartesian
Useful Constants

pi - the numerical equivalent to π=3.14159265..., the ratio of a circle’s circumference to its diameter

i and j - imaginary unit, √-1 (for complex numbers you may use an asterisk as in –2+3*I or have it implied as in –2+3i)

eps - floating-point relative precision of your computer, 2^-52

realmin - smallest floating-point number, 2^-1022

realmax - largest floating-point number, (2^-ε)2^1023

inf - infinity, ∞, results from operations like division by zero and overflow (i.e., exceed realmax), which lead to results too large to represent as conventional floating-point values

NaN - undefined numerical result, Not-a-Number, results from expressions like 0/0, inf/inf or Inf-Inf, as well as from arithmetic operations involving a NaN (also, if n is complex with a zero real part, then n/0 returns a value with a NaN real part)

NaT - is the representation for Not-a-Time, a value that can be stored in a datetime array to indicate an unknown or missing datetime value
Open Source Paradigm

```matlab
>> open(which('fzero.m'))
```

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
MATLAB Editor

New Script  Open m-file

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
You can save a Live Script as a MATLAB live code file (live script), or m-file (ignoring the output part).

Right-clicking on any m-file appearing in the Current Folder panel opens a menu giving that gives you an the option of opening this m-file as a Live script rather than a regular Script.

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Examples of Live Scripts

Create URL address

3. URL: [http://weather.wpc.noaa.gov/region/naco/]

Read source (HTML) code

5. SourceFile = webread(url);
6. dlmwrite('WeatherData.txt', SourceFile, 'delimiter', ' ');

Find the beginning and end of table data

8. PatternString = 'KX';
9. start = regexpi( SourceFile, PatternString, 'start' );
10. end = regexpi( SourceFile, PatternString, 'end' );

Read data in

11. text = SourceFile( start : end );

Plot data

13. plot(x, y), xlabel('Time (s)'), ylabel('Amplitude (V)');

Air Mass and Solar Radiation

As light from the sun passes through the earth's atmosphere, some of the solar radiation will be absorbed. The air mass is a function of solar elevation ($\alpha$). As shown in the diagram below, it is a measure of the length of the path of light through the atmosphere ($Y$) relative to the shortest possible path ($\tan\alpha$) when the sun's elevation is 90°.

The larger the air mass, the less radiation reaches the ground. The air mass can be calculated from the equation:

$$AM = \frac{1}{\cos(90 - \alpha) + 0.58572 \cdot (0.67955 + \log2\sin(\alpha)) - 1.6354};$$

Then the solar radiation (in W/m²) reaching the ground can be calculated from the empirical equation:

$$srad = 1.353 \cdot 10^9 \cdot \frac{AM}{0.8}\; srad.$$
### Managing Work Session

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>format</code></td>
<td>- default format with 4 decimal digits, same as <code>short</code></td>
</tr>
<tr>
<td><code>format short</code></td>
<td>- short, fixed-decimal format with 4 digits after the decimal point (ADP)</td>
</tr>
<tr>
<td><code>format long</code></td>
<td>- long, fixed-decimal format with 15 and 7 digits ADP for double and single values, resp.</td>
</tr>
<tr>
<td><code>format shortE</code></td>
<td>- short scientific floating-point format with 4 digits ADP</td>
</tr>
<tr>
<td><code>format longE</code></td>
<td>- long scientific floating-point format with 15 and 7 digits ADP for double and single values, respectively</td>
</tr>
<tr>
<td><code>format shortEng</code></td>
<td>- the same as <code>shortE</code>, but with three digits dedicated to store exponent</td>
</tr>
<tr>
<td><code>format longEng</code></td>
<td>- the same as <code>longE</code>, but with three digits dedicated to store exponent at the expense of having only 14 and 6 digits ADP for double and single values, resp.</td>
</tr>
<tr>
<td><code>format shortG</code></td>
<td>- short or <code>shortE</code>, whichever is more compact</td>
</tr>
<tr>
<td><code>format longG</code></td>
<td>- long or <code>longE</code>, whichever is more compact</td>
</tr>
<tr>
<td><code>format hex</code></td>
<td>- hexadecimal format</td>
</tr>
<tr>
<td><code>format +</code></td>
<td>- the symbols +, -, and blank are printed for positive, negative, and zero elements, while imaginary parts are ignored</td>
</tr>
<tr>
<td><code>format bank</code></td>
<td>- fixed format for dollars and cents (2 decimal digits)</td>
</tr>
<tr>
<td><code>format rat</code></td>
<td>- approximation by ratio of small integers</td>
</tr>
<tr>
<td><code>format compact</code></td>
<td>- suppresses excess line feeds to show more output in a simple screen</td>
</tr>
<tr>
<td><code>format loose</code></td>
<td>- adds extra line feeds to make output more readable</td>
</tr>
</tbody>
</table>

---

*All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.*
## Managing Work Session

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>home</td>
<td>moves the cursor to the upper left corner of the Command window</td>
</tr>
<tr>
<td>clc</td>
<td>clears the Command window and homes the cursor</td>
</tr>
<tr>
<td>exist('name')</td>
<td>determines if a file or variable exists having the name 'name'</td>
</tr>
<tr>
<td>who</td>
<td>lists the variables currently in memory</td>
</tr>
<tr>
<td>whos</td>
<td>lists the current variables and sizes, and indicates if they have imaginary parts</td>
</tr>
<tr>
<td>clear</td>
<td>removes all variables from memory (workspace)</td>
</tr>
<tr>
<td>clear var1 var2</td>
<td>removes the variables var1 and var2 from memory</td>
</tr>
<tr>
<td>clearvars -except a b</td>
<td>clears all variables except for those specified following the -except flag</td>
</tr>
<tr>
<td>close all</td>
<td>closes all open figure windows</td>
</tr>
<tr>
<td>close('name')</td>
<td>closes the named figure window</td>
</tr>
<tr>
<td>quit</td>
<td>stops MATLAB</td>
</tr>
<tr>
<td>: (colon)</td>
<td>generates an array having regularly spaced elements</td>
</tr>
<tr>
<td>, (comma)</td>
<td>separates elements of an array</td>
</tr>
<tr>
<td>; (semicolon)</td>
<td>suppresses screen printing also denotes a new row in an array</td>
</tr>
<tr>
<td>... (ellipsis)</td>
<td>continues a line</td>
</tr>
</tbody>
</table>

Also, see **pwd** (identifies current folder), **dir** or **ls** (lists folder contents), **what** (lists MATLAB files in folder), **cd** (changes current folder), **which** (locates functions and files), **fileparts** (gets parts of file name), and **memory** (displays memory information)
>> preferences
MATLAB saves statements that run in the Command Window to the history file, History.xml, located in the directory \AppData\Roaming\MathWorks\MATLAB\R2018b (type `prefdir` in the Command Window to see what the directory containing preferences, history, and layout files is). The History.m file is loaded when MATLAB starts. It stores a maximum of 25,000 commands (the default value), deleting the oldest entries as needed to maintain that size.
Toolbars / Colors Preferences

Preferences

MATLAB Toolbox Preferences
- Toolbar: Quick Access
- Layout:
  - [ ] Show Label
  - [ ] Show Window
  - [ ] Show Switch
  - [ ] Show Toolbar
  - [ ] Show Window

Preferences

MATLAB Colors Preferences
- Desktop tool colors
  - [ ] Use system colors
  - [ ] Background
  - [ ] Text

MATLAB syntax highlighting colors
- Keywords
- Comments
- [ ] Unmatched strings
- [ ] Unmatched strings
- [ ] Syntax errors
- [ ] Invalid type

Syntax highlighting sample
```
% create a file for output
f = fopen('testfile.txt', 'w');
for i=1:10
    fprintf(f, '66.2f
end
```

Command Window sample
```
>> samplefunction
Link to sample: link
Warning: Min value set to 0
> In samplefunction at 1
Error using samplefunction
Invalid type
>>
```

Restore Default Colors

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Variable Editor / Figure Copy Preferences

- MATLAB Variables Preferences
  - Format: Default array format: short
  - Editing: Move selection after Enter
    - Direction: Down
  - International number handling: Decimal separator for exporting numeric data via system clipboard

- MATLAB Figure Copy Template Copy Options Preferences
  - Clipboard format: Metafile (may lose information)
  - Background color: Use figure color
  - Size: Match figure screen size

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Current Folder Window

MATLAB R2014b

HOME | PLOTS | APPS

FILE | HOME | USER | Oleg | Documents | MATLAB |

Current Folder

- New Folder
- New File
- Reports
  - Code Analyzer Report
  - TODO/FIXME Report
  - Help Report
  - Contents Report
  - Dependency Report
  - Coverage Report

Workspace

- Name
- Value
- Size
- Class

Select a file to view details

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted, unless for course participation, in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the Publisher and/or Author. Contact the American Institute of Aeronautics and Astronautics, Professional Development Programs, 12700 Sunrise Valley Drive, Suite 200, Reston, VA 20191-5807.
Annotating a file with the comments including text TODO, FIXME, or a string of your choice makes it easier to find areas of your code that you intend to improve, complete, or update later.

Displays potential errors and problems, as well as opportunities for improvement in your code through messages.

Checks the help component of your MATLAB scripts.
Current Folder Reports

Allows creating the `Contents.m` file providing a summary of the programs in a particular folder. (The `help`, `doc`, and `ver` functions refer to `Contents.m` files to display information about folders.)

```
>> help codetools
```

Shows dependencies among MATLAB code files in a current folder.

Determine how much of a file ran when you profiled it (when you run the Profiler on a file, some code might not run, e.g. a block containing an if statement).
Profiling is a way to measure where a program spends time. Once you identify which functions are consuming the most time, you can determine why you are calling them. Then, you could look for ways to minimize their use and thus improve performance by:

- Avoiding unnecessary computation, which can arise from oversight
- Changing your algorithm to avoid costly functions
- Avoiding recomputation by storing results for future use

When profiling spends most of its time on calls to a few built-in functions, you have probably optimized the code as much as you can.
1) **Think** about the main goal you want to achieve.
2) Define required **inputs** and desired **outputs**.
3) Collect all **necessary information**.
4) Start with a **simplified version** of the problem and clearly state your assumptions.
5) Draw a **sketch** of the problem and think of **names for your variables**.
6) Check the **dimensions and units** of your variables.
7) Think about how you are going to proceed toward a solution and write down the **general steps**, maybe even labeling them (which will further be converted to the comments line in your code).
8) Make sure your program is robust enough to handle a variety of the **expected and unexpected (wrong or missing) inputs** as well as interface properly with any applications you use.
9) Start writing code, checking its pieces if appropriate by displaying **intermediate results** as needed.
10) Devote some time to develop a concise, yet informative output structure, which later on will include producing informative **graphics**.
11) Check the outputs of your program on some specific cases with known (**expected**) results.
12) Perform a “reality check” of the outputs your program produces in a **general case**.
13) If your algorithm involves some iterative computations (such as, optimization), try to **limit the number of cycles** first to be sure your program works properly.
14) Do not start batch computing unless you are absolutely sure your program runs properly for a **single dataset** (having lots of pretty output does not necessarily mean it is *not* garbage).
15) After debugging your program, make sure to **suppress** displacing all unnecessary **intermediate results**, so that your program produces only those outputs you actually need.
The End of Chapter 2

Questions?