Chapter 2. Development Environment and Basic Operations
• 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
Desktop Prior to R2012b

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

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

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

Workspace browser – shows variables defined (created) in workspace

Minimize  Restore  Undock  Close

Command History window – displays previously issued commands

>> commandwindow
>> commandhistory
>> filebrowser
>> workspace
Managing MATLAB Desktop

- Tabs
- Quick Access Toolbar
- Search window
- Toolstrip
- MATLAB Drive
- Current folder
- Command window
- Workspace

% MATLAB commands
>> commandwindow
>> commandhistory
>> filebrowser
>> workspace
Managing MATLAB Desktop

- **Profiler**
- **Simulink**
- **Help Browser**

**Desktop layout**

**GUIDE**

**Add-Ons**

**Help options**
Command History Window

Tap “↑” key
Apps Tab
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

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

In MATLAB/Simulink, you can create favorites to save frequently used commands or scripts. To create a favorite, you can right-click on the command you want to save and select "Create Favorite" from the context menu. This will open the Favorite Command Editor, where you can label the favorite, enter the code, choose a category, and decide whether to add it to the quick access toolbar.

Once created, favorites can be accessed from the Favorites section of the MATLAB interface, allowing quick access to frequently used commands or scripts.
Workspace Window

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.
MATLAB Demos

![MATLAB Demos Screenshot](image)

The image shows a MATLAB Demos page with a sidebar menu and a main content area. The sidebar menu includes categories such as Getting Started with MATLAB, MATLAB Examples, Language Fundamentals, Mathematics, Graphics, and more. The main content area displays various MATLAB Examples, including getting started videos and working in the development environment.
Examples

Documentation

**bin2dec**
Convert text representation of binary number to decimal number

**Syntax**

```matlab
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`. `binarystr` can be a character array or a cell array of character vectors. `bin2dec` ignores any space (“ ”) characters in the input text.

**Examples**

Binary 0101111 converts to decimal 27:

```matlab
bin2dec('0101111')
```

ans =
27
Getting Quick Help

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

Right click
Engineering Computations and Modeling in MATLAB/Simulink

Exploring MATLAB Functions

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

```matlab
>> help ops
>> help elfun
>> help elmat
>> help specfun
>> help randfun
>> 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\backslash b)</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.
# Embedded Functions

## Trigonometric functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sin(X)</td>
<td>sine</td>
</tr>
<tr>
<td>asin(X)</td>
<td>inverse sine</td>
</tr>
<tr>
<td>cos(X)</td>
<td>cosine</td>
</tr>
<tr>
<td>acos(X)</td>
<td>inverse cosine</td>
</tr>
<tr>
<td>tan(X)</td>
<td>tangent</td>
</tr>
<tr>
<td>atan(X)</td>
<td>inverse tangent</td>
</tr>
<tr>
<td>atan2(Y,X)</td>
<td>4-quadrant inverse tangent</td>
</tr>
<tr>
<td>sec(X)</td>
<td>secant (1/cos(x))</td>
</tr>
<tr>
<td>asec(X)</td>
<td>inverse secant</td>
</tr>
<tr>
<td>csc(X)</td>
<td>cosecant (1/sin(x))</td>
</tr>
<tr>
<td>acsc(X)</td>
<td>inverse cosecant</td>
</tr>
<tr>
<td>cot(X)</td>
<td>cotangent (1/tan(x))</td>
</tr>
<tr>
<td>acot(X)</td>
<td>inverse cotangent</td>
</tr>
<tr>
<td>sind(X)</td>
<td>sine</td>
</tr>
<tr>
<td>asind(X)</td>
<td>inverse sine</td>
</tr>
<tr>
<td>cosd(X)</td>
<td>cosine</td>
</tr>
<tr>
<td>acosd(X)</td>
<td>inverse cosine</td>
</tr>
<tr>
<td>tand(X)</td>
<td>tangent</td>
</tr>
<tr>
<td>atand(X)</td>
<td>inverse tangent</td>
</tr>
<tr>
<td>atan2d(Y,X)</td>
<td>4-q. inv. tangent</td>
</tr>
<tr>
<td>secd(X)</td>
<td>secant</td>
</tr>
<tr>
<td>asecd(X)</td>
<td>inverse secant</td>
</tr>
<tr>
<td>cscd(X)</td>
<td>cosecant</td>
</tr>
<tr>
<td>acscd(X)</td>
<td>inverse cosecant</td>
</tr>
<tr>
<td>cotd(X)</td>
<td>cotangent</td>
</tr>
<tr>
<td>acotd(X)</td>
<td>inverse cotangent</td>
</tr>
<tr>
<td>sinh(X)</td>
<td>hyperbolic sine</td>
</tr>
<tr>
<td>asinh(X)</td>
<td>inverse hyperbolic sine</td>
</tr>
<tr>
<td>cosh(X)</td>
<td>hyperbolic cosine</td>
</tr>
<tr>
<td>acosh(X)</td>
<td>inverse hyperbolic cosine</td>
</tr>
<tr>
<td>tanh(X)</td>
<td>hyperbolic tangent</td>
</tr>
<tr>
<td>atanh(X)</td>
<td>inverse hyperbolic tangent</td>
</tr>
<tr>
<td>sech(X)</td>
<td>hyperbolic secant</td>
</tr>
<tr>
<td>asech(X)</td>
<td>inverse hyperbolic secant</td>
</tr>
<tr>
<td>csch(X)</td>
<td>hyperbolic cosecant</td>
</tr>
<tr>
<td>acsch(X)</td>
<td>inverse hyperbolic cosecant</td>
</tr>
<tr>
<td>coth(X)</td>
<td>hyperbolic cotangent</td>
</tr>
<tr>
<td>acoth(X)</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>fix(X)</td>
<td>rounds towards zero</td>
</tr>
<tr>
<td>floor(X)</td>
<td>rounds towards minus infinity</td>
</tr>
<tr>
<td>ceil(X)</td>
<td>rounds towards plus infinity</td>
</tr>
<tr>
<td>round(X)</td>
<td>rounds towards nearest integer (round(X,N) rounds to N digits)</td>
</tr>
<tr>
<td>mod(a,m)</td>
<td>executes a modulo operation (signed remainder after division of a by m)</td>
</tr>
<tr>
<td>rem(a,b)</td>
<td>computes a remainder after division of a by b</td>
</tr>
<tr>
<td>sign(X)</td>
<td>returns the signum function of X (returns 1 if the element is greater than 0, 0 - equals 0, and -1 - less than 0)</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.
Examples of Using Functions

>> 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 =
   7.3592e+05

>> date
ans =
   26-Apr-2015
Variable Names

>> Speed=25
Speed =
 25
>> speed=30
speed =
 30
>> SPEED_of AC=35
SPEED_of AC =
 35
>> speeD2=2*Speed
speeD2 =
 50

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).

<table>
<thead>
<tr>
<th>Examples of valid names:</th>
<th>Invalid names:</th>
</tr>
</thead>
<tbody>
<tr>
<td>x6</td>
<td>6x</td>
</tr>
<tr>
<td>lastValue</td>
<td>end</td>
</tr>
<tr>
<td>n_factorial</td>
<td>n!</td>
</tr>
</tbody>
</table>

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

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

#### Exponential functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp(X)</td>
<td>exponential</td>
</tr>
<tr>
<td>log(X)</td>
<td>natural logarithm</td>
</tr>
<tr>
<td>reallog(X)</td>
<td>natural logarithm for nonnegative real arrays</td>
</tr>
<tr>
<td>log10(X)</td>
<td>common (base-10) logarithm</td>
</tr>
<tr>
<td>log2(X)</td>
<td>base-2 logarithm</td>
</tr>
<tr>
<td>pow2(X)</td>
<td>base-2 power</td>
</tr>
<tr>
<td>realpow(X,Y)</td>
<td>power that will error out on complex result</td>
</tr>
<tr>
<td>sqrt(X)</td>
<td>square root</td>
</tr>
<tr>
<td>realsqrt(X)</td>
<td>square root for nonnegative real arrays</td>
</tr>
<tr>
<td>hypot(A,B)</td>
<td>square root of sum of squares (hypotenuse)</td>
</tr>
<tr>
<td>nthroot(X)</td>
<td>real (n)th root of real numbers</td>
</tr>
<tr>
<td>nextpow2(X)</td>
<td>next higher power of (2)</td>
</tr>
</tbody>
</table>

#### Complex functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(X)</td>
<td>absolute value and complex magnitude</td>
</tr>
<tr>
<td>angle(Z)</td>
<td>phase angle in the interval ([-\pi,\pi])</td>
</tr>
<tr>
<td>complex(A,B)</td>
<td>constructs complex data from the real and imaginary parts</td>
</tr>
<tr>
<td>conj(Z)</td>
<td>complex conjugate</td>
</tr>
<tr>
<td>imag(X)</td>
<td>imaginary part of complex number</td>
</tr>
<tr>
<td>real(X)</td>
<td>real part of complex number</td>
</tr>
<tr>
<td>unwrap(X)</td>
<td>unwraps the radian phase angles (whenever the jump between consecutive angles is greater than or equal to (\pi) radians, <strong>unwrap</strong> shifts the angles by adding multiples of (\pm 2\pi) until the jump is less than (\pi))</td>
</tr>
<tr>
<td>isreal(X)</td>
<td>determines whether (X) is real</td>
</tr>
<tr>
<td>cplxpair(Z)</td>
<td>sort numbers into complex conjugate pairs</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.*
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 \( \pi = 3.14159265... \), the ratio of a circle’s circumference to its diameter

i and j  - imaginary unit, \( \sqrt{-1} \) (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, \( (2^{-\varepsilon})2^{1023} \)

inf  - infinity, \( \infty \), 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 \( \text{Inf-Inf} \), 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

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.
Open Source Paradigm

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

Example of using LaTeX math constructs:

```
function [x, eval, flag, output] = fzero(FunFunction, x, options, varargin)
%FZERO Single-variable nonlinear zero finding.

% X = FZERO(FUN,X0) tries to find a zero of the function FUN near X0.
% If X0 is a vector, it first finds an interval containing X0 where the
% function values of the interval endpoints differ in sign, then searches
% that interval for a zero.  FUN is a function handle.  FUN accepts real
% scalar input X and returns a real scalar function value F, evaluated
% at X.  The value X returned by FZERO is near a point where FUN changes
% sign (if FUN is continuous), or NaN if the search fails.

% X = FZERO(FUN,X0), where X0 is a vector of length 2, assumes X0 is a
% finite interval where the sign of FUN(X0(1)) differs from the sign of
% FUN(X0(2)).  An error occurs if this is not true. Calling FZERO with a
% finite interval guarantees FZERO will return a value near a point where
% FUN changes sign.

% X = FZERO(FUN,X0), where X0 is a scalar value, uses X0 as a starting
% guess.  FZERO looks for an interval containing a sign change for FUN and
% containing X0.  If no such interval is found, NaN is returned.

% In this case, the search terminates when the search interval
```
MATLAB Editor

New Script  Open m-file
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 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

```matlab
url = 'http://weather.washington.edu/cgi-bin/sounding?region=nacon&TYPE=TEXT...'
WMO_ID = 72493;
Y = 2019; M = '07'; D = '29'; T = '12'; % Read data for 7/29/2019 at 12Z
```

Read source (HTML) code

```matlab
SourceFile = urlread(url);
sWrite('Index.htm', SourceFile, 'delimiter', '');
```

Find the beginning and end of table data

```matlab
Pattern = 'X'; k = strfind(SourceFile, Pattern);
start = (end - length(Pattern));
pos = strfind(SourceFile, '
\n\n', 'once');
endPos = pos+1;
```

Read data in

```matlab
c = textscan(SourceFile(start:end), 'delimiter', ',');
```

Plot data

```matlab
plot([c(1), c(2)/1000, ...], grid, xlabel('T, OCC'), ylabel('W, km'), figure
plot([c(1), c(2)/1000, ...], grid, xlabel('W, km'), ylabel('L, km'),
```

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 \(X\), 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

\[
AM = \frac{1}{\cos(90 - \alpha)} + 0.5077 \times (6.97955 + \text{deg2rad}(\alpha)) - 1.6354
\]

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

\[
\text{Srad} = 1.353 \times 0.54^{\text{AirMass}}
\]

![Diagram of solar radiation and air mass](image.png)
Managing Work Session

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>format</code></td>
<td>- default format with 4 decimal digits, same as short</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>- <code>short</code> or <code>shortE</code>, whichever is more compact</td>
</tr>
<tr>
<td><code>format longG</code></td>
<td>- <code>long</code> 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>
### Managing Work Session

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>home</strong></td>
<td>moves the cursor to the upper left corner of the Command window</td>
</tr>
<tr>
<td><strong>clc</strong></td>
<td>clears the Command window and homes the cursor</td>
</tr>
<tr>
<td><strong>exist(’name’)</strong></td>
<td>determines if a file or variable exists having the name ’name’</td>
</tr>
<tr>
<td><strong>who</strong></td>
<td>lists the variables currently in memory</td>
</tr>
<tr>
<td><strong>whos</strong></td>
<td>lists the current variables and sizes, and indicates if they have imaginary parts</td>
</tr>
<tr>
<td><strong>clear</strong></td>
<td>removes all variables from memory (workspace)</td>
</tr>
<tr>
<td><strong>clear var1 var2</strong></td>
<td>removes the variables var1 and var2 from memory</td>
</tr>
<tr>
<td><strong>clearvars</strong></td>
<td>clears all variables except for those specified following the -except flag</td>
</tr>
<tr>
<td><strong>close all</strong></td>
<td>closes all open figure windows</td>
</tr>
<tr>
<td><strong>close(’name’)</strong></td>
<td>closes the named figure window</td>
</tr>
<tr>
<td><strong>quit</strong></td>
<td>stops MATLAB</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 Preferences

Select an element in the tree to set its preferences.

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 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.
Variable Editor / Figure Copy Preferences

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: 

Preferences

MATLAB Figure Copy Template Copy Options Preferences

Clipboard format
- Metafile (may lose information)
- Bitmap
- Preserve information (metafile if possible)

Figure background color
- Use figure color
- Transparent background
- Force white background

Size
- Match figure screen size

Select this option to copy the figure as it appears on screen, or leave it unchecked to use the Print Preview settings to determine its size.
Current Folder Window

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 Reports

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

Checks the help component of your MATLAB scripts.

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.
Current Folder Reports

```
>> 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).

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.)
Code Profiling for Improving Performance

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.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Calls</th>
<th>Total Time</th>
<th>Self Time*</th>
<th>Total Time Plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadingJpeg</td>
<td>1</td>
<td>6.04 s</td>
<td>1.19 s</td>
<td></td>
</tr>
<tr>
<td>legend</td>
<td>5</td>
<td>1.36 s</td>
<td>0.03 s</td>
<td></td>
</tr>
<tr>
<td>scr_private\legend\GUI\UsingMATLABClasses</td>
<td>5</td>
<td>1.33 s</td>
<td>0.13 s</td>
<td></td>
</tr>
<tr>
<td>...nc\GUI\UsingMATLABClasses\make_legend</td>
<td>5</td>
<td>1.204 s</td>
<td>0.022 s</td>
<td></td>
</tr>
<tr>
<td>xlabel</td>
<td>12</td>
<td>1.041 s</td>
<td>0.867 s</td>
<td></td>
</tr>
<tr>
<td>quantizer</td>
<td>36</td>
<td>0.857 s</td>
<td>0.823 s</td>
<td></td>
</tr>
<tr>
<td>Legend_Legend\Legend_Legend</td>
<td>5</td>
<td>0.707 s</td>
<td>0.079 s</td>
<td></td>
</tr>
<tr>
<td>Legend_dcSetup</td>
<td>5</td>
<td>0.549 s</td>
<td>0.133 s</td>
<td></td>
</tr>
<tr>
<td>ylabel</td>
<td>34</td>
<td>0.546 s</td>
<td>0.512 s</td>
<td></td>
</tr>
<tr>
<td>subplot</td>
<td>39</td>
<td>0.517 s</td>
<td>0.022 s</td>
<td></td>
</tr>
<tr>
<td>gra_private\subpict\GUI\UsingMATLABClasses</td>
<td>39</td>
<td>0.495 s</td>
<td>0.320 s</td>
<td></td>
</tr>
<tr>
<td>Legend_dcMethod</td>
<td>20</td>
<td>0.379 s</td>
<td>0.005 s</td>
<td></td>
</tr>
</tbody>
</table>

*Self Time refers to the time spent in a function, excluding time spent in any of its sub-functions.
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?