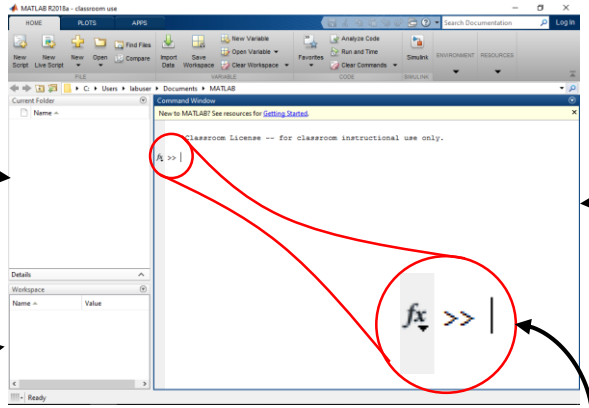
Brief Introduction to Matlab

(Benjamin Walter METU-NCC SPE Presentation 5.10.2019)

The Matlab Window (R2018)



Current Folder
data files, scripts, etc on computer's drive

Workspace
variables, data, functions in computer memory

Command Window
type commands here

Command Prompt
type commands after the >>
Note: it may take a while for this to appear when Matlab starts on campus computers…

Command prompt is like "calculator mode."

```
>>  2 + 5
 ans =
       7
```

```
>>  cos(15)
 ans =
      -0.7597
```

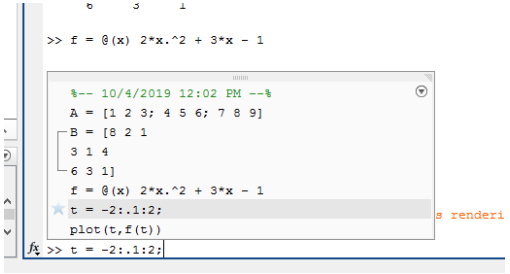
Store results in variables.

```
>>  2 + 5
 ans =
       7

>> ans + 3
 ans =
       10
```

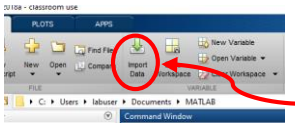
```
>>  a = 2
 a =
       2

>> b = a^a + 3
 b =
       7
```

Press ⇧ *<up arrow>* on your keyboard to see your command history

Repeat a command from the past!

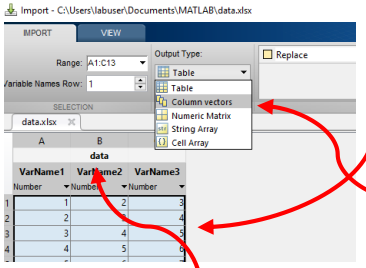


Importing Data to Matlab with GUI (R2018)



Easiest method: use import tool

Select data file from drive:
.txt .csv .xls .xlsx .ods



Highlight data from file to import

Choose format of data after import:
table, series of vectors, matrix, …
(probably matrix)

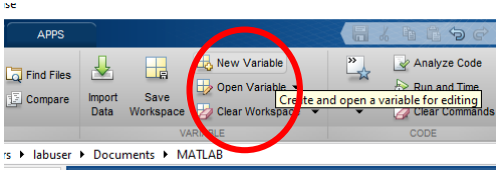
Click to change name of imported data and name of columns (if table)

Large dataset --- may be easier to use direct command:

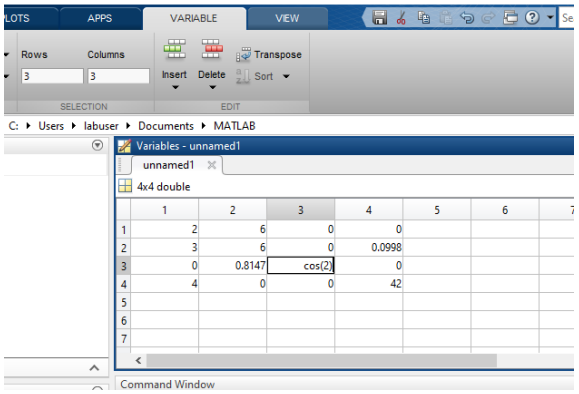
>> readmatrix(<*file*>)

>> readtable(<*file*>)

Creating and Editing Variables with GUI (R2018)



Graphical Interface for Editing:
-- Double-click variable in *Workspace*
or
-- Select *"Open Variable"* from ribbon



Now you can enter values into the data variable, just like Excel.

This is meant for raw numeric data entry – no complicated formulas allowed!

Free online Matlab tutorial course by MathWorks (Matlab company)
https://www.mathworks.com/learn/tutorials/matlab-onramp.html

Basic Data Manipulation in Matlab

Basic data types: Vectors and Matrices.
-- Entered using […] comma or space separates row entries
semicolon or new-line separates rows.

```
>> v = [ 2 3 4 5 ]
 v =
      2  3  4  5

>> w = [ 4, 6, 8, 10 ]
 w =
      4  6  8  10
```

```
>> A = [ 1 2 ; 3 4 ]
 A =
      1  2
      3  4

>> B = [ 1  2
        3  4 ]
```

Important basic operator for making vectors -- colon:

<start> : <end> *sequence of numbers from <start> to <end>*
<start> : <step> : <end> *sequence from <start> to <end> by <step>*

```
>> v = 2 : 5
 v =
      2  3  4  5

>> w = 0 : 0.2 : 2
 w =
      0  0.2  0.4  0.6  0.8  1.0
```

```
>> A = [ 1 : 3 ; 4 : 2 : 8 ; -1 : 1]
 A =
      1  2  3
      4  6  8
     -1  0  1
```

Note: Ending command with semicolon suppresses Matlab output

```
>> v = 2 : 5 ;

>> w = 0 : 0.2 : 2 ;
```

```
>> A = [ 1 : 3 ; 4 : 2 : 8 ; -1 : 1] ;

>> B = [ 1 : 100 ; 101 : 200 ] ;
```

Refer to specific **elements** of vectors and matrices using (..) *"index notation"*

```
>> v = [ 7  11  13  17 ]
 v =
      7  11  13  17

>> v(2)
 ans =
      11
```

element 2 of v

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

>> A(2 , 3)
 ans =
      6
```

element in row 2, column 3 of A

Use index notation (..) to change elements as well as view them.

```
>> v = [ 7  11  13  17 ]
 v =
      7  11  13  17

>> v(2) = 25
 v =
      7  25  13  17
```

```
>> A = [ 4  5  6 ; 7  8  9 ; 10 11 12 ] ;

>> A(2 , 3) = 0
 A =
      4  5  6
      7  8  0
     10  11  12
```

Combine with : to cut out or change chunks of vectors or matrices.

```
>> v = [ 7  11  13  17  19 ]
 v =
      7  11  13  17  19

>> v(2 : 4)
 ans =
      11  13  17
```

```
>> A = [ 4  5  6 ; 7  8  9 ; 10 11 12 ] ;

>> A( 2 : 3 , 1 : 2)
 ans =
      5  6
      8  9
```

Colon on its own is a wildcard – for returning entire rows or columns.

```
>> A( :  , 1)      ← all of column 1
>> A( 2 , : )      ← all of row 2
```

Indexing with conditional statements gives values satisfying specific properties.

```
>> v( v <= 15 )    ← only the values of v that are ≤ 15
>> A( sin(A) ~= 0 )  ← only the values of A whose sin ≠ 0
```

Combine matrices using […]

```
>> A = [ 1 2 ; 3 4 ] ;

>> B = [ 5 6 ; 7 8 ] ;

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

```
>> A = [ 1 2 ; 3 4 ] ;

>> B = [ 5 6 ];

>> [ A ; B ]
 ans =
      1  2
      3  4
      5  6
```

Examples

```
>> A  =  [ A   A( : , 1)  ]      ← add a copy of the first column to end of A
>> A  =  [ A(end, :) ; A ]       ← add a copy of the last row to start of A
>> A  =   A( : , [3 1 2] )       ← reorder the columns of A
>> A(:, 3)  =  A(:, 1) + A(:, 2)   ← column 3 of A is column 1 + column 2
>> A(1, :)  =  A(1, :) + 20      ← add 20 to each element in row 1
>> A(A<3)  =  A(A<3) * 2         ← double all elements of A that are less than 3
>>  [ 1  2  3 ] +  1             ← add 1 to vector [ 1  2  3 ]
>>  [ 1  2  3 ] + [ 1 ; 5 ; 7 ]  ← matrix with three rows:  add 1,  add 5,  add 7
```

Basic Operations in Matlab

Operations: + - * / ^ (plus, minus, times, division, power)

A few functions:

sin	cos	tan	*etc*	(trigonometric functions – *in radians*)
log	log10	exp	sqrt	(natural logarithm and log base 10, exponential, sqrt)
abs	max	min	ceil	(absolute value, maximum, minimum, ceiling)

Adding and multiplying vectors or matrices by numbers applies to all elements.

```
>> A + 2   ← adds 2 to each element
>> A / 4   ← divides each element by 4
```

Adding and multiplying vectors or matrices by other vectors or matrices attempts matrix operation.

```
>> A + B   ← adds matrices
>> A * B   ← multiplies matrices
```

Will fail unless sizes of A and B are compatible!

To multiply or divide **elementwise**, use the "dot" versions of the operator .* ./ .^ :

```
>> v = [ 1 2 3 ] ;
>> w = [ 4 5 6 ] ;
>> v * w
Error using  *
Incorrect dimensions for matrix mult.
```

```
>> v = [ 1 2 3 ] ;
>> w = [ 4 5 6 ] ;
>> v .* w
ans =
     4   10   18
```

Anonymous Functions in Matlab

To define your own algebraic functions in Matlab use the format

@(*<vars>*) *<function expression>*

```
>> f = @(x)  x.^2 + 3*x + 1 ;
>> f(2)
 ans =
        11
```

```
>> g = @(x,y)  x.^2 – y.^2 + x.*y ;
>> g(2,3)
 ans =
         1
```

To apply these functions to vectors or matrices you **must** use "dot" versions of operators!

```
>> [x, y] = meshgrid(-1:0.1:1, -1:0.1:1) ;   % mesh of points filling [0,1]x[0,1]
>> f = @(x,y) x.^2 + y.^2 ;                  % distance from (x,y) to (0,0)
>> circ_x = x( f(x,y) <= 1 ) ;               % keep only the points (x,y) which
>> circ_y = y( f(x,y) <= 1 ) ;               %     are inside the circle of radius 1
```

Basic Graphing in Matlab

Basic data plotting commands:

2D Plots

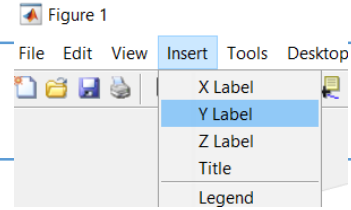
```
>> scatter ( <x> , <y> )      ← plot points
>> plot  ( <x> , <y> )        ← plot points connected by lines (curves)
-------------------------------------------------------------------
>> fplot ( <f(x)> )           ← plot function (on interval [-5,5])
>> fplot ( <x(t)>, <y(t)>)    ← plot parametric curve (on -5 ≤ t ≤ 5)
```

3D Plots

```
>> meshgrid ( <x> , <y> )     ← creates grid of sample points for evaluating f(x,y)
>> mesh ( <x> , <y> , <z> )   ← plot points connected by mesh of lines
>> surf   ( <x> , <y> , <z> ) ← plot points connected by shaded polygons
-------------------------------------------------------------------
>> fsurf   ( <f(x,y)> )                    ← plot surface (on interval [-5,5 for x and y])
>> fsurf   ( <x(u,v)> , .. , <z(u,v)> )    ← plot parametric surface
>> fplot3 ( <x(t)>, <y(t)>, <z(t)>)        ← plot parametric curve (on -5 ≤ t ≤ 5)
```

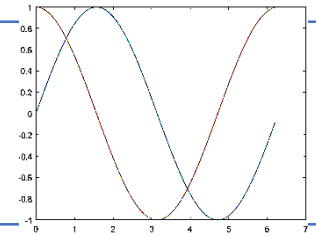
Using any of these commands will open a "Figure Window" containing your graph.

The easiest way to add axis labels, graph title, etc. is to use the toolbar on the figure window.

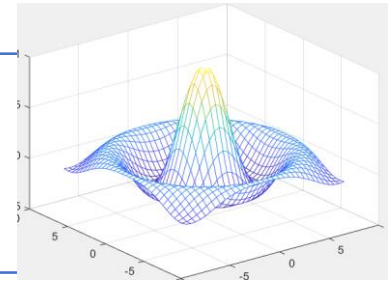


Use "hold" command to add new plots to an existing figure.

```
>> x = 0 : 0.1 : 2*pi ;
>> plot (x, sin(x))
>> hold on
>> plot (x, cos(x))
>> hold off
```



```
>> [x ,y]  =  meshgrid (-8 : 0.5 : 8) ;
>> d  =  @(x,y)  sqrt(x.^2 + y.^2) ;
>> f  =  @(x,y) sin(d(x,y)) ./ d(x,y) ;
>> mesh (x, y, f(x,y))
>>  % surf( x, y, f(x,y))
```



Basic Programming in Matlab

Conditional Statements.

```
if ( <expression> )
   <statements>
end
```

```
if ( <expression> )
   <statements>
else
   <statements>
end
```

```
if ( <expression> )
   <statements>
elseif ( <expression> )
   <statements>
else
   <statements>
end
```

Logical Operators.

>	greater than
<	less than
==	equal to

>=	greater than or equal to
<=	less than or equal to
~=	not equal to

&	AND
|	OR
~	NOT

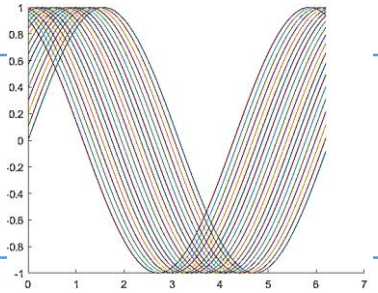
Loops.

```
for ( <var> = <vector> )
   <statements>
end
```

```
while ( <expression> )
   <statements>
end
```

Examples.

```
>> x = 0 : 0.1 : 2*pi ;
>> hold on
>> for ( k = 0 : 0.1 : 2 )
       plot( x , sin(x + k) )
   end
```



Using Loops to Compute Factorial

```
>> x = 1 ;
>> for ( k = 1 : 10 )
       x = x * k ;
   end
```

```
>> x = 1 ;              k = 1 ;
>> while ( k <= 10 )
       x = x * k ;      k = k + 1 ;
   end
```

(Note: Combine multiple commands per line using semicolon.)

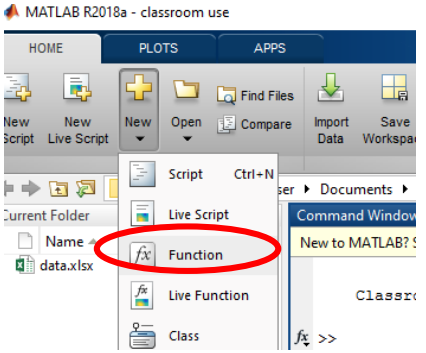
Nice tutorial video with more detail: https://www.youtube.com/watch?v=zr_aB7V79DE

Creating Simple Functions

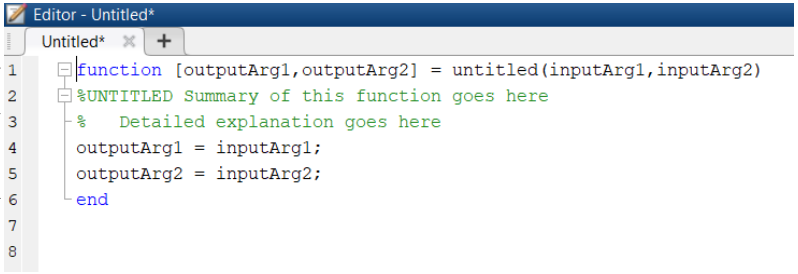
Programs are stored in function files

Create new function:
Select "New" → "Function"

Edit existing function:
Double-click a function in "Current Folder".



- Matlab function files have **.m** extension



Top line is function declaration. **function** *<output>* = **name** (*<input>*)
outputArg results returned by function *(usually delete one and rename)*
untitled current name of function *(usually rename this)*
inputArg data which is passed to the function *(usually rename this)*

```
>>  [ X , Y ] = untitled( x , y )        % use function in Matlab
```

Next two lines are **comments** giving information about function *(optional)*
first line is FUNCTION_NAME and a one line description of function
second line is a more detailed summary

Function code with for loops and whatnot goes after the "**function**" declaration and before the "**end**" declaration.

Final line is **end** declaration.
when function reaches end, the value of "outputArg" well be returned

Functions use their own memory space (separate from the workspace)
-- only see the data which is passed to them
-- will not overwrite or change any variables on the workspace

Free online "basic coding in Matlab" tutorial by MathWorks:
https://learntocode.mathworks.com/