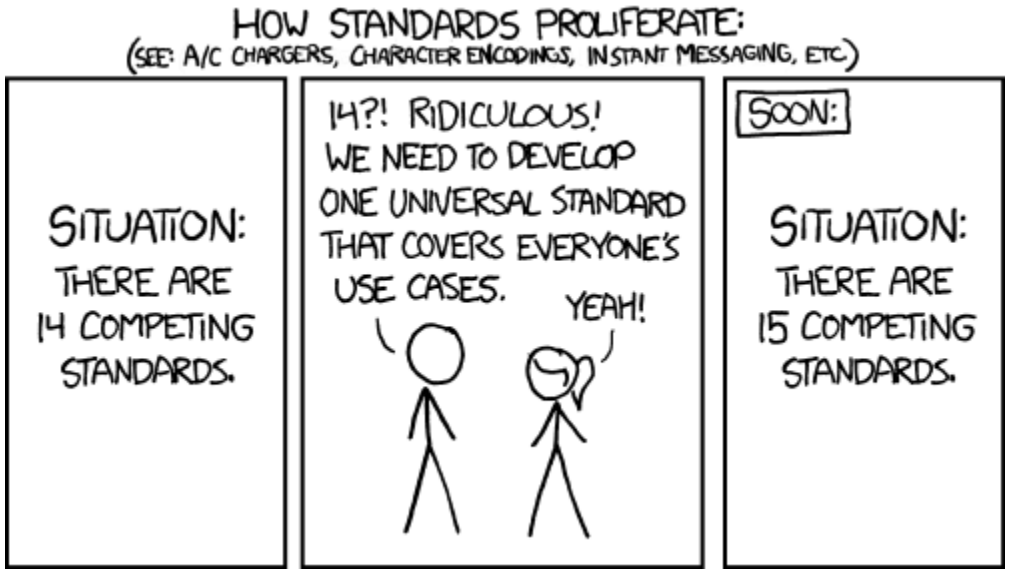
Emergency MATLAB

Computational Methods and Tools

ENG-270

2024 Fall

Why another language?



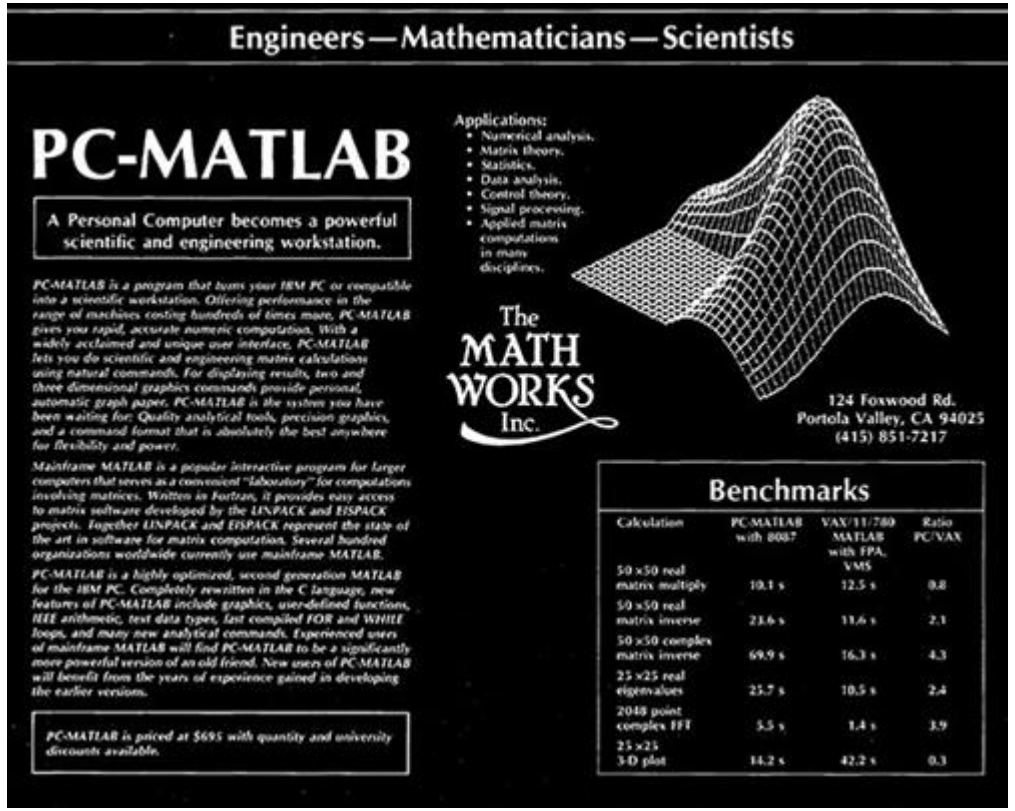
Source: XKCD.

Some languages better represent data and algorithms in specific domains (Domain Specific Languages or DSLs). Also, history matters.

MATLAB

- Dynamically-typed
- Originally interpreted; JIT* since R2016a
- Proprietary

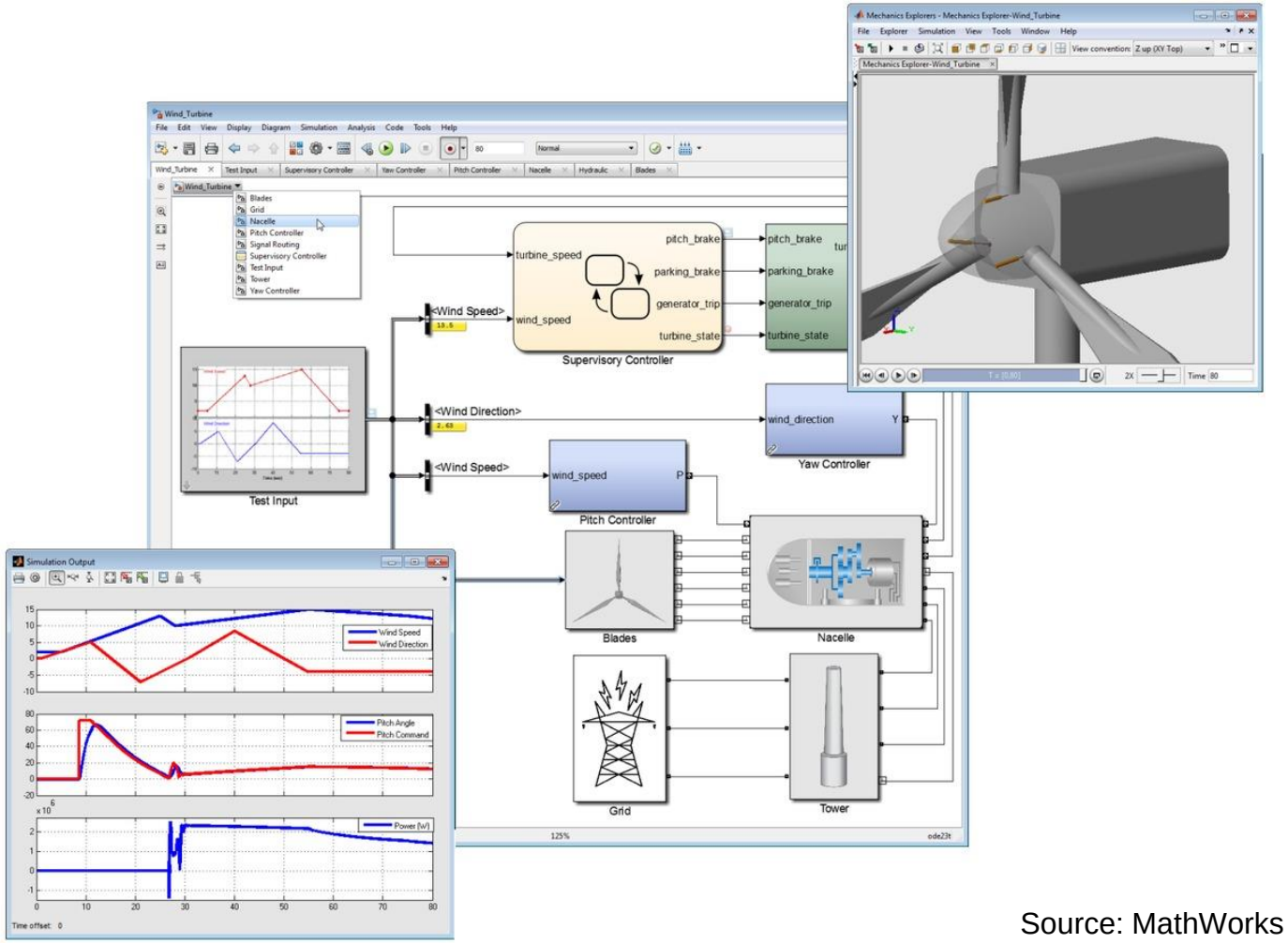
- Started 1970s
- Influenced many languages
- Continues to expand/evolve as it incorporates developments in other languages



Source: MathWorks

*just-in-time compiled

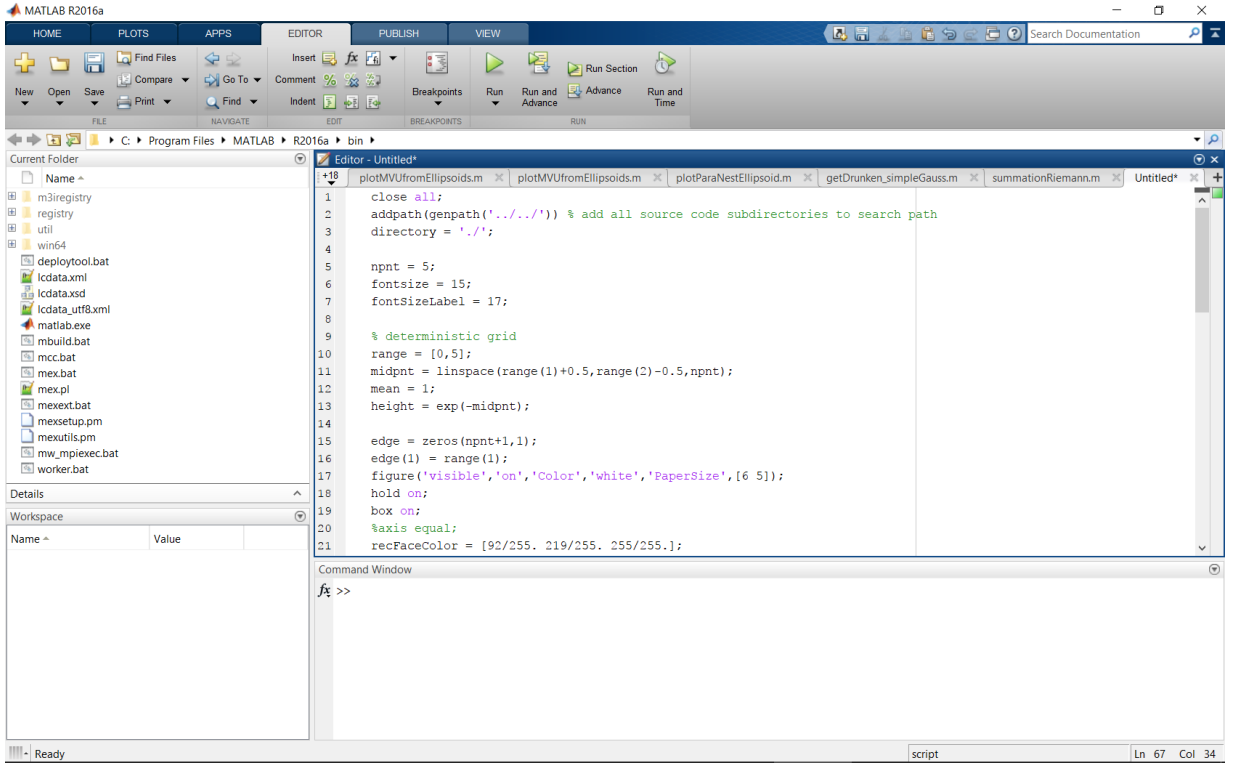
Simulink



Source: MathWorks

IDE and workflow

https://www.cdslab.org/matlab/notes/preliminary-foundations/matlab-for-beginners/index.html



Works with three types of files:

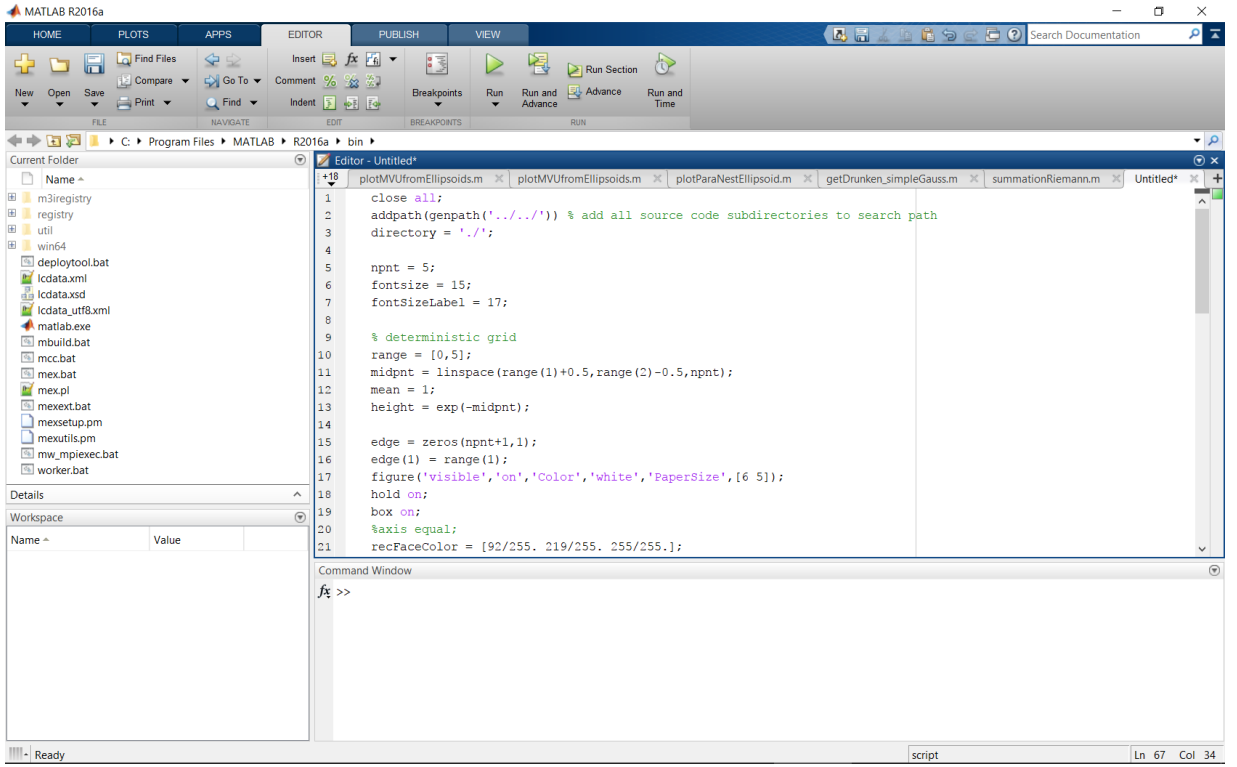
- scripts (.m)
- functions (.m)
- data (.mat)

Get help:

- help *functionname*
- doc *functionname*

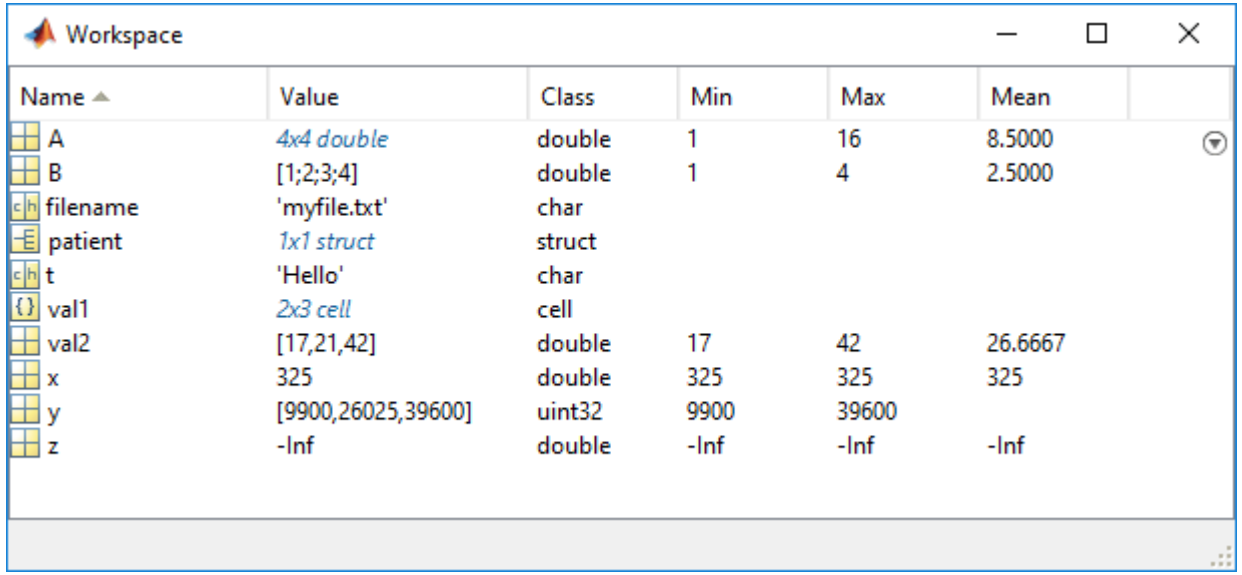
Run one section at a time

https://www.cdslab.org/matlab/notes/preliminary-foundations/matlab-for-beginners/index.html



Like using the REPL in Python

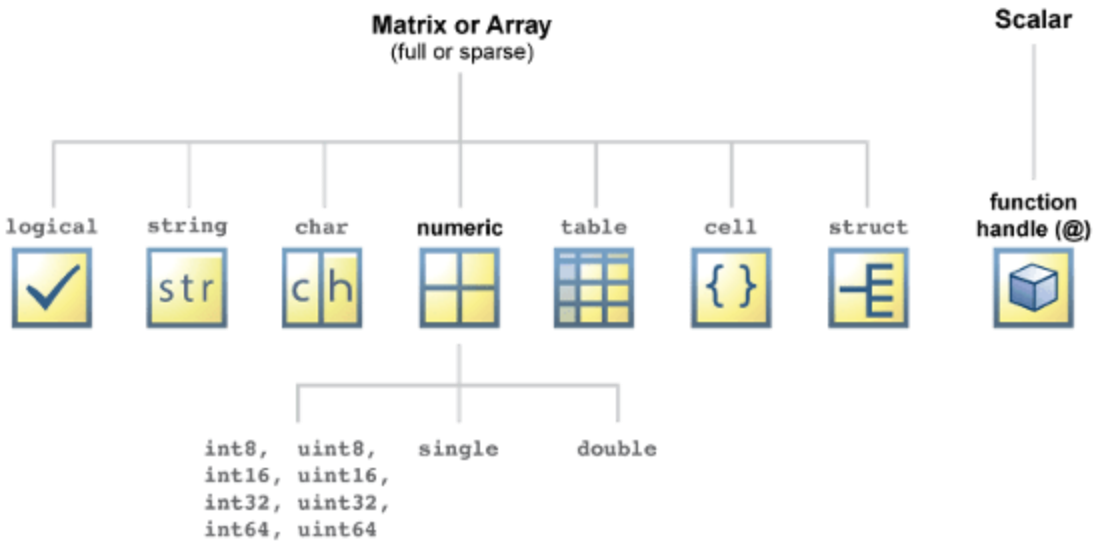
Workspace



- “Workspace” == namespace
- Save variables with *save*
- Load variables with *load*

Variables (data types / objects)

Most everything is a matrix or array



Similarities:

MATLAB	Python	C
cell array	list	
containers.Map	dictionary	
structure		struct

- Default is a matrix (1D or 2D)
- *N*D arrays are also possible
- Cell arrays are most similar to Python lists, but cell arrays can be multidimensional
- Most numerical computations should be handled through vectorized operations with matrices or arrays

1D arrays are row-wise by default:

```
a = 1:5;
size(a)
```

```
ans =

   1   5
```

```
>> [1 2 3]

ans =

     1     2     3
```

```
>> [1, 2, 3]

ans =

     1     2     3
```

```
>> [1; 2; 3]

ans =

     1
     2
     3
```

More examples:

```
>> C = {'lat', 'lon', 'alt'; 57.3, 6.35, 0}

C =

  2×3 cell array

    {'lat'    }    {'lon'   }    {'alt'}
    {[57.3000]}    {[6.3500]}    {[  0]}

>> S = struct('lat', 57.3, 'lon', 6.35, 'alt', 0)

S =

  struct with fields:

    lat: 57.3000
    lon: 6.3500
    alt: 0

>> M = containers.Map({'lat', 'lon', 'alt'}, [57.3, 6.35, 0])

M =

  Map with properties:

       Count: 3
     KeyType: char
   ValueType: double
```

Extract contents:

```
>> C(1,:)

ans =

  1×3 cell array

    {'lat'}    {'lon'}    {'alt'}

>> S.lat

ans =

   57.3000

>> S.('lat')

ans =

   57.3000

>> M('lat')

ans =

   57.3000
```

Array indexing

MATLAB arrays are 1-indexed rather than 0-indexed (Python and C). I.e., the first index is 1 instead of 0.

Generally, you should use (), except {} to extract single element of *cell arrays*.

```
a = 1:10;
disp(a(2:5));
disp(a(end));
disp(a(9:end));
disp(a(1:length(a) > 5));
```

```
2   3   4   5
10
9   10
6    7    8    9   10
```

```
C = {1, 2, 3};
```

```
>> C{1}

ans =

     1

>> C(1)

ans =

  1×1 cell array

    {[1]}

>> C(1:2)

ans =

  1×2 cell array

    {[1]}    {[2]}
```

Math

Operations are vectorized (like NumPy – which was inspired by MATLAB)

MATLAB

```
>> disp([1; 2] + [3; 4])
     4
     6

>> 1:10 < 5

ans =

  1×10 logical array

   1   1   1   1   0   0   0   0   0   0
```

Python

```
>>> import numpy as np
>>> print(np.array([1, 2]) + np.array([3, 4]))
[4 6]
>>> np.arange(10) < 5
array([ True,  True,  True,  True,  True, False, False, False, False,
       False])
```

“Gotchas” – differences from Python

- Element-wise operators – prefix with period (“.”)
- Not operator is ~ instead of !
- Not equal to is ~= instead of !=

```
>> 1:5 < 3

ans =

  1×5 logical array

   1   1   0   0   0
```

```
>> ~(1:5 < 3)

ans =

  1×5 logical array

   0   0   1   1   1
```

```
>> a = [1, 2]

a =

     1     2


>> b = [3; 4];

b =

     3
     4
```

Matrix multiplication (dot/inner product):

```
>> a * b

ans =

    11
```

Element-wise multiplication

```
>> a .* b

ans =

     3     6
     4     8

>> a .* b'

ans =

     3     8
```

Characters and strings

Use single quotes to define character arrays.

Use cell arrays to hold a series of character arrays. (Or struct)

```
string = 'abc';
array_of_strings = {'abc', 'bcd'};
```

Function	Description	Example
`strcmp`	compare strings	`strcmp('abc', 'abc')`
`strcat` or `[ ]`	join strings	`stcat('abc', 'def')`
`str2num`	convert strings to numbres	`str2num('5')`
`num2str`	convert numbers to strings (also see `sprintf`)	`num2str(5)`
`sprintf`	format strings	`sprintf('Hello %s', 'world')`
`lower~/~upper`	change case	`lower('a')`
`findstr`	find shorter string in longer string	`findstr('b', 'abc')`

Control structures

Largely similar to Python, C

MATLAB

```
nrows = 4;
ncols = 6;
A = ones(nrows,ncols);

for c = 1:ncols
    for r = 1:nrows

        if r == c
            A(r,c) = 2;
        elseif abs(r-c) == 1
            A(r,c) = -1;
        else
            A(r,c) = 0;
        end

    end
end
```

```
>> class(A)

ans =

    'double'

>> A

A =

     2    -1     0     0     0     0
    -1     2    -1     0     0     0
     0    -1     2    -1     0     0
     0     0    -1     2    -1     0
```

Python

```
import numpy as np

nrows = 4
ncols = 6
A = np.ones((nrows, ncols))

for c in range(ncols):
    for r in range(nrows):

        if r == c:
            A[r,c] = 2
        elif abs(r - c) == 1:
            A[r,c] = -1
        else:
            A[r,c] = 0
```

```
>>> A.dtype
dtype('float64')
>>> A
array([[ 2., -1.,  0.,  0.,  0.,  0.],
       [-1.,  2., -1.,  0.,  0.,  0.],
       [ 0., -1.,  2., -1.,  0.,  0.],
       [ 0.,  0., -1.,  2., -1.,  0.]])
```

Functions

	MATLAB	Python

Single return value	`function y = square(x)`
	`  y = x.^2;`
	`end`
		`def square(x):`
		`  y = x**2`
		`  return y`

Multiple return values	`function [m,s] = stat(x)`
	`  n = length(x);`
	`  m = sum(x)/n;`
	`  s = sqrt(sum((x-m).^2/n));`
	`end`
		`from math import sqrt`
		`def stat(x):`
		`  n = len(x)`
		`  m = sum(x)/n;`
		`  s = sqrt(sum(map(lambda x: (x-m)**2, x))/n)`
		`  return m, s`

	`out = stat(1:10)`	`out = stat(range(1, 11))`
	only first argument captured *need to explicitly capture multiple outputs*	*return value is a tuple*
Anonymous function	`power = @(x, y) x.^y`	`power = lambda x, y: x**y`

Where to define functions

Before Release 2024a:

- at the end of the script
- in separate “function files” – each function in a separate .m file with the same name as the function

Since Release 2024a:

- can include anywhere in the script – but only accessible to this script (not command line) *“Local functions are only visible within the file where they are defined. They are not visible to functions in other files, and cannot be called from the Command Window.”*

Function files

Each function must be defined in its own file.
A function called only by one other function can be defined in the latter function's file.

Example: main.m called the function stat().

example directory

```
.
├── main.m
├── square.m
└── stat.m
```

square.m

```
function y = square(x)
    y = x.^2;
end
```

stat.m

```
function [m,s] = stat(x)
    n = length(x);
    m = sum(x)/n;
    s = sqrt(sum((x-m).^2/n));
end
```

example directory

```
.
├── main.m
└── stat.m
```

stat.m

```
function [m,s] = stat(x)
    n = length(x);
    m = sum(x)/n;
    s = sqrt(sum(square(x-m)/n));
end

function y = square(x)
  y = x.^2;
end
```

Paths

- List of paths determines method resolution order
- query order with *path*
- find function with *which –all functionName*
- add path with *addpath()*

```
                MATLABPATH

       C:\Program Files\MATLAB\R2017a\toolbox\matlab\datafun
       C:\Program Files\MATLAB\R2017a\toolbox\matlab\datatypes
       C:\Program Files\MATLAB\R2017a\toolbox\matlab\elfun
       C:\Program Files\MATLAB\R2017a\toolbox\matlab\elmat
       C:\Program Files\MATLAB\R2017a\toolbox\matlab\funfun
       C:\Program Files\MATLAB\R2017a\toolbox\matlab\general
       C:\Program Files\MATLAB\R2017a\toolbox\matlab\iofun
       C:\Program Files\MATLAB\R2017a\toolbox\matlab\lang
       C:\Program Files\MATLAB\R2017a\toolbox\matlab\matfun
...
```

File I/O

Use `fileparts` and `fullfile` to deconstruct and construct file names.

- deconstruct with `fileparts`

```
[filepath,name,ext] = fileparts("/home/jsmith/myfile.txt")
```

```
filepath =
"/home/jsmith"

name =
"myfile"

ext =
".txt"
```

- construct with `fullfile`

```
fullfile(filepath, strcat(name, ext))
```

```
ans =

  "/home/jsmith/myfile.txt"
```

Check files in directory / check if file exists:

- `dir`: list folder (directory) contents
- `isfile(filename)`: check if file exists

Example: "num1.csv"

```
col1,col2,col3
0.0,0.1,0.2
0.3,0.4,0.5
```

fscanf and **fprintf**

Gotchas:

- Note that transposed dimensions have to be provided for `fscanf` and
- the tranpose of the matrix must be provided to `fprintf`.

```
fid = fopen('num1.csv');
header = split(fgetl(fid), ',');
A = fscanf(fid, '%f,%f,%f', [3, Inf])'; % note transpose
fclose(fid);
```

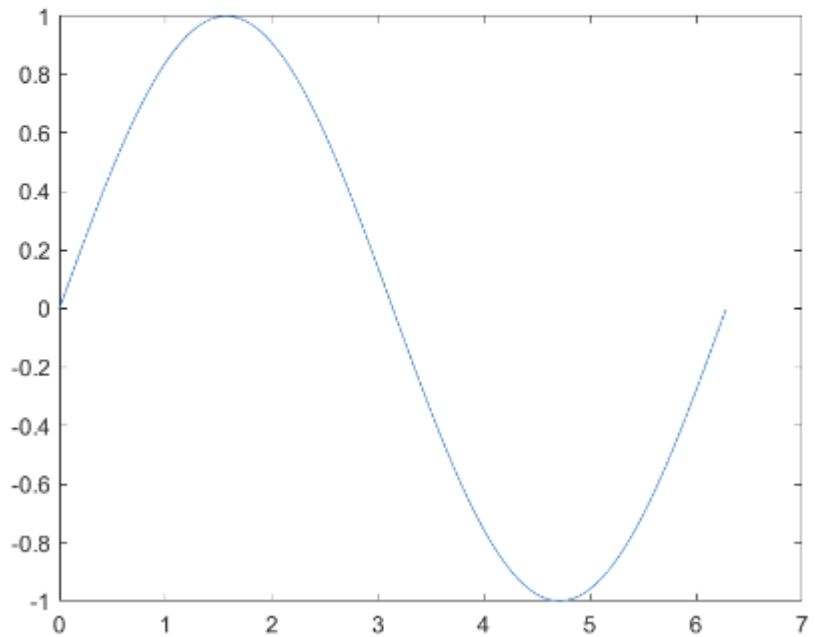
```
fout = fopen('num1_out.csv', 'w');
fprintf(fout, 'col1,col2,col3\n');
fscanf(fout, '%f,%f,%f', A'); % note transpose
fclose(fout);
```

Graphics

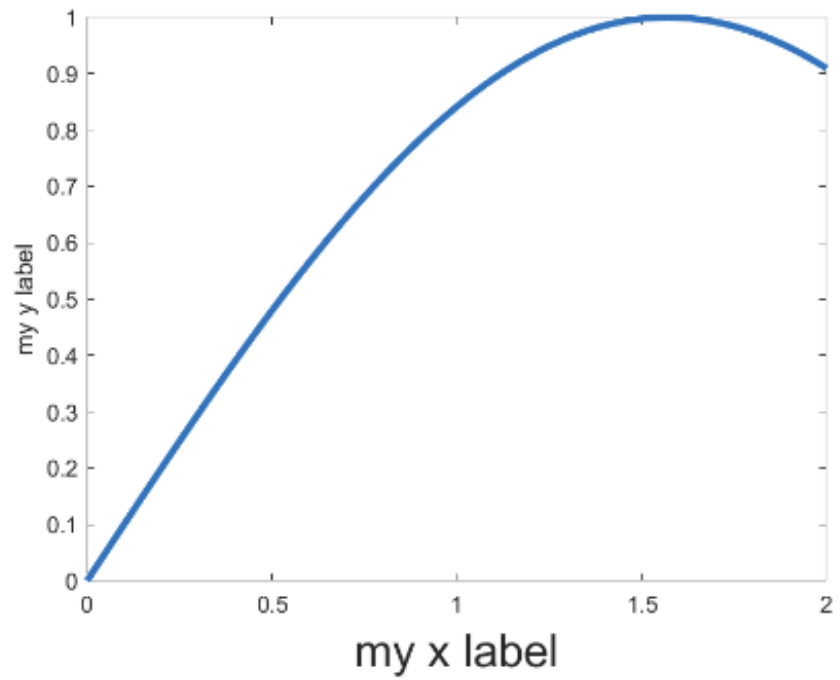
Line Plots	Scatter and Bubble Charts	Data Distribution Plots	Discrete Data Plots	Geographic Plots	Polar Plots	Contour Plots	Vector Fields	Surface and Mesh Plots	Volume Visualization	Animation	Images
plot	scatter	histogram	bar	geoplot	polarplot	contour	quiver	surf	streamline	animatedline	image
plot3	scatter3	histogram2	barh	geoscatter	polarhistogram	contourf	quiver3	surfc	streamslice	comet	imagesc
stairs	bubblechart	pie	bar3	geobubble	polarscatter	contour3	feather	surfl	streamparticles	comet3	
errorbar	bubblechart3	pie3	bar3h		polarbubblechart	contourslice		ribbon	streamribbon		
area	swarmchart	scatterhistogram	pareto		compass	fcontour		pcolor	streamtube		
stackedplot	swarmchart3	swarmchart	stem		ezpolar			fsurf	coneplot		
loglog	spy	swarmchart3	stem3					fimplicit3	slice		
semilogx		wordcloud	stairs					mesh			
semilogy		bubblecloud						meshc			
fplot		heatmap						meshz			
fplot3		parallelplot						waterfall			
fimplicit		plotmatrix						fmesh			

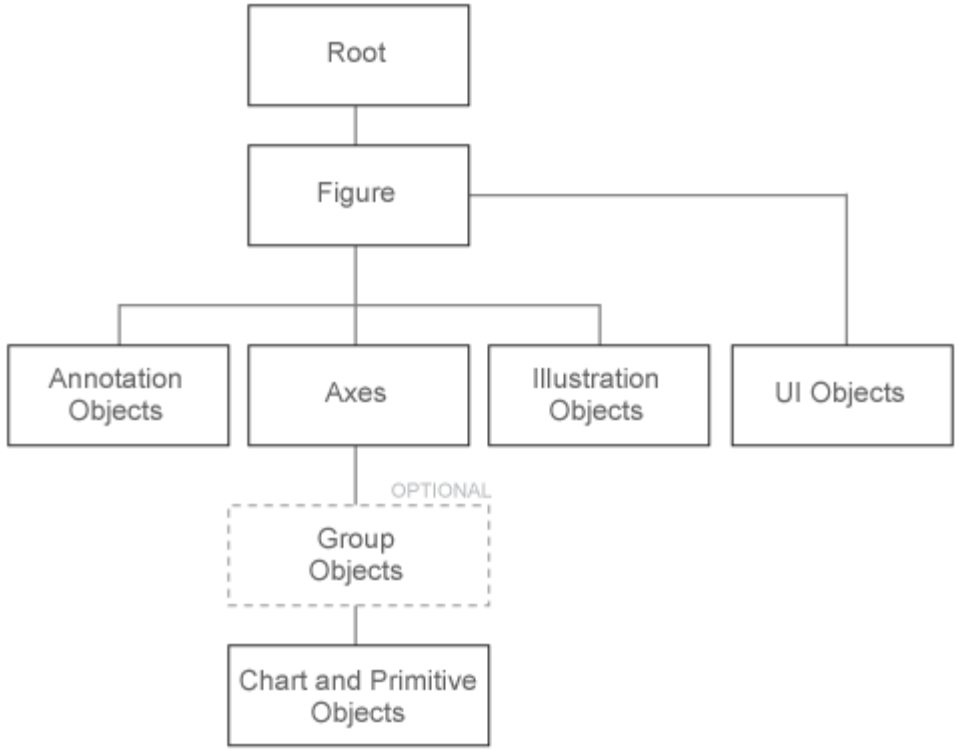
Source: MathWorks

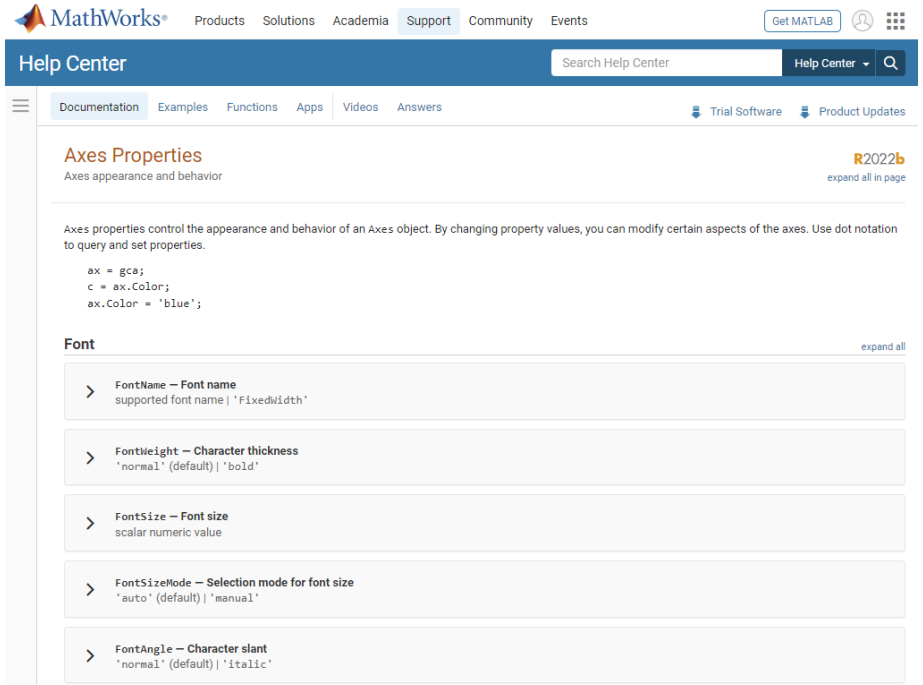
```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```




```
x = 0:pi/100:2*pi;
y = sin(x);
h = plot(x,y);
ax = gca; % current axes
ax.XLim = [0, 2];
ax.XLabel.String = 'my x label';
ax.XLabel.FontSize = 20;
ax.YLabel.String = 'my y label';
h.LineWidth = 3;
```

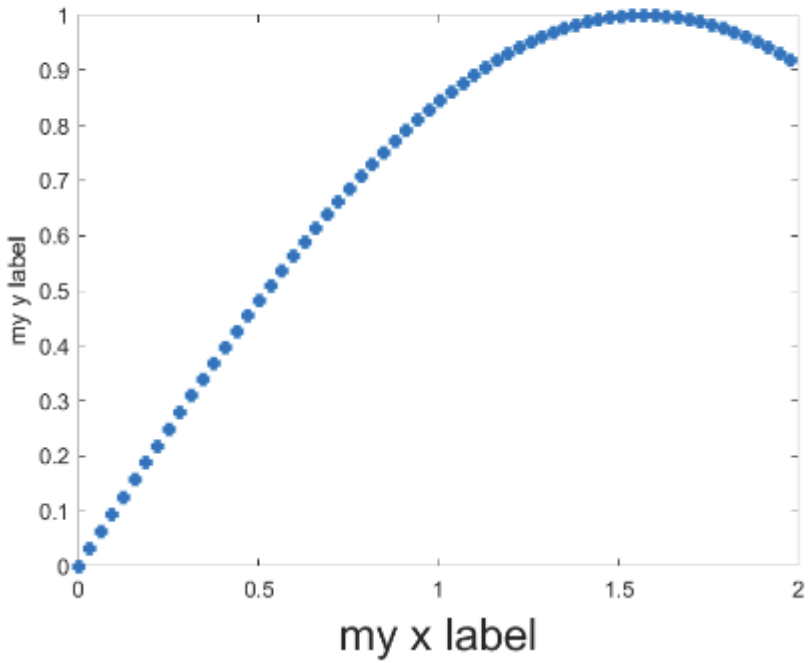




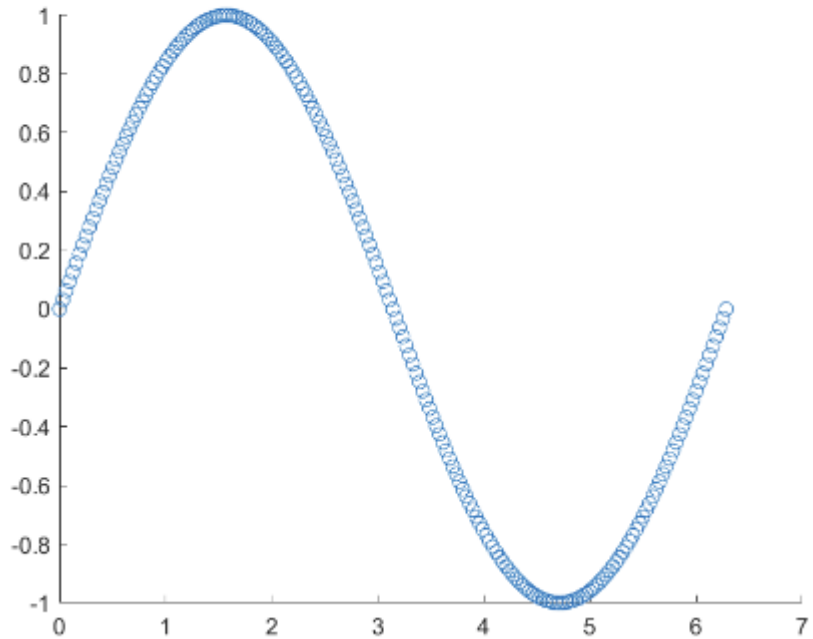


Source: MathWorks

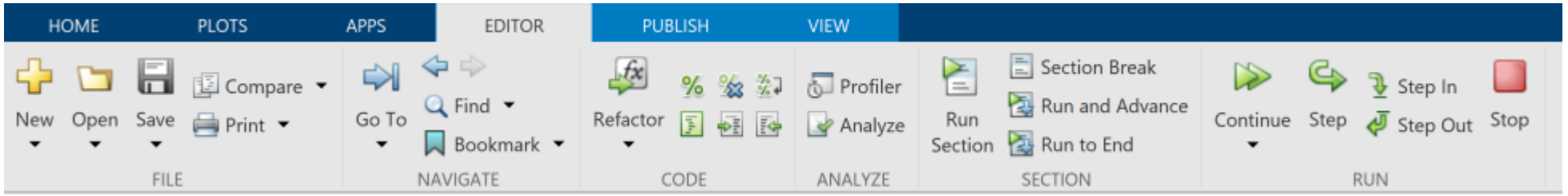
```
h.LineStyle = 'none';
h.Marker = '+';
```

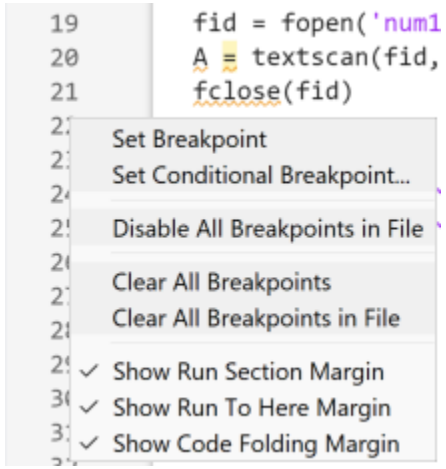


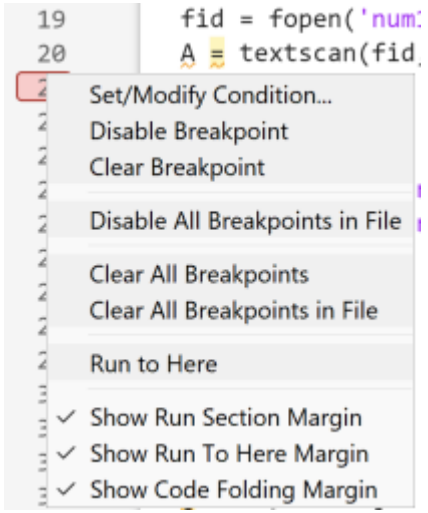

```
scatter(x,y)
```



Debugger


HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
New
Open
Save
Compare
Print
FILE
Go To
Find
Bookmark
NAVIGATE
Refactor
CODE
Profiler
Analyze
ANALYZE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Continue
Step
Step In
Step Out
Stop
RUN


19 fid = fopen('num1
20 A = textscan(fid,
21 fclose(fid)
Set Breakpoint
Set Conditional Breakpoint...
Disable All Breakpoints in File
Clear All Breakpoints
Clear All Breakpoints in File
Show Run Section Margin
Show Run To Here Margin
Show Code Folding Margin


19 fid = fopen('num
20 A = textscan(fid
Set/Modify Condition...
Disable Breakpoint
Clear Breakpoint
Disable All Breakpoints in File
Clear All Breakpoints
Clear All Breakpoints in File
Run to Here
Show Run Section Margin
Show Run To Here Margin
Show Code Folding Margin

Examples

Exercises on sieprog island

- éolienne: hint, partial solutions
- oiseaux: hint, solutions
- suisse: solutions

Modern additions

- functional programming, part 1, part 2
- object-oriented programming