MATLAB 科学计算语言与应用

Lecturer: 徐岩

Email: xuyan@tju.edu.cn

Tel: 18622996386

Lecture 1

Outline

I. Getting Started

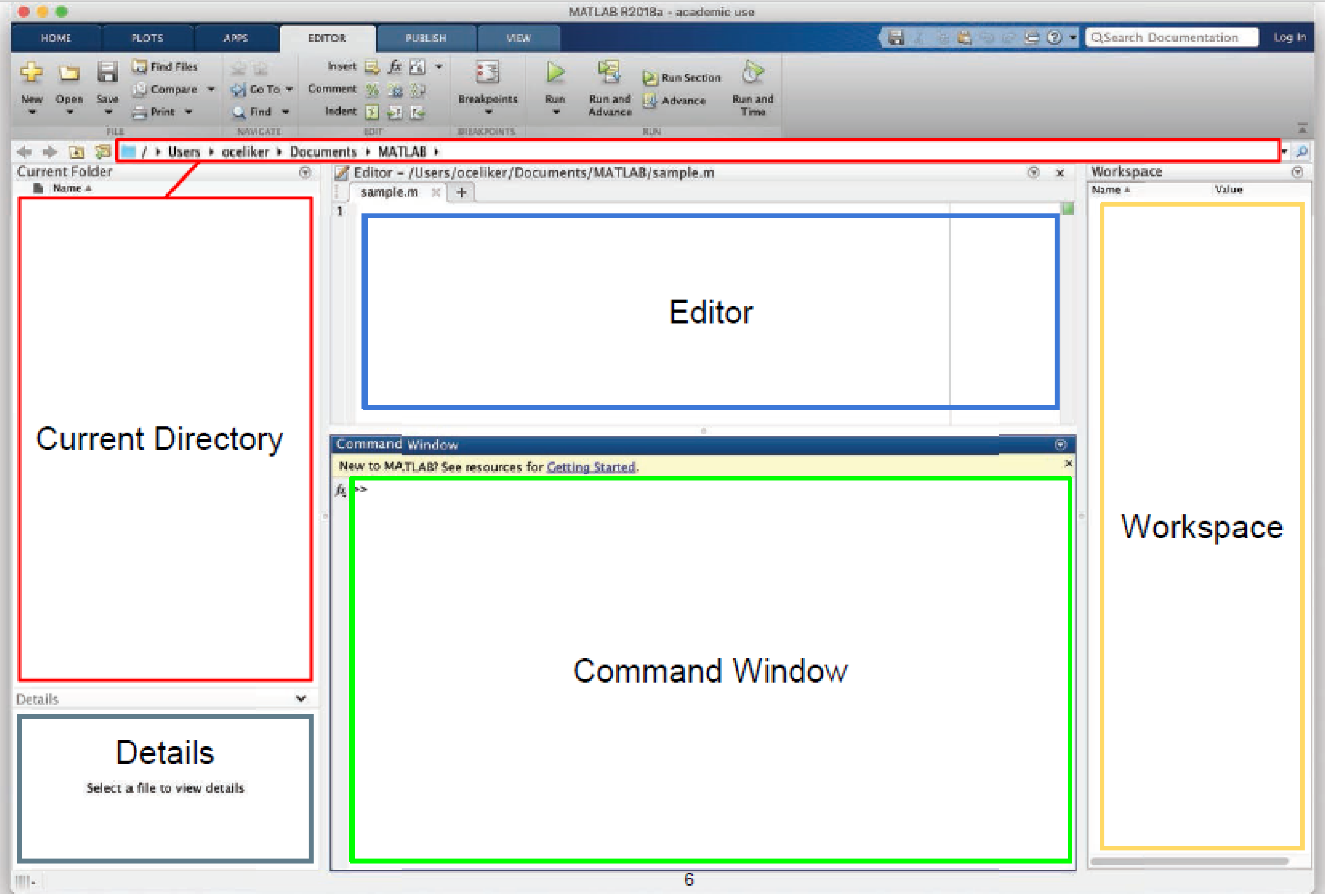
II. Scripts

III. Making Variables

IV. Manipulating Variables

V. Basic Plotting

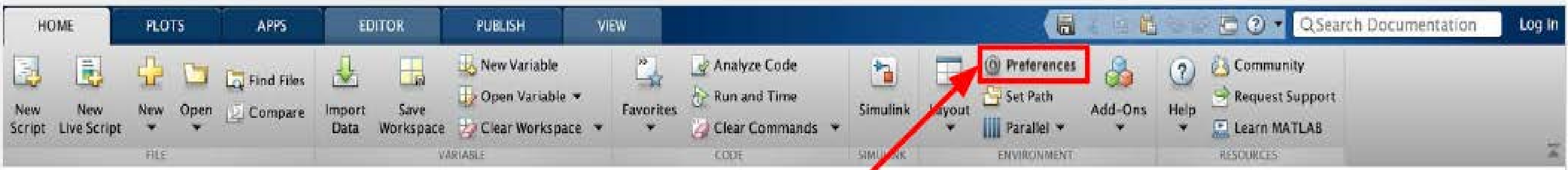
I. Getting Started



New to MATLAB? See resources for Getting Started.

```
>>
```

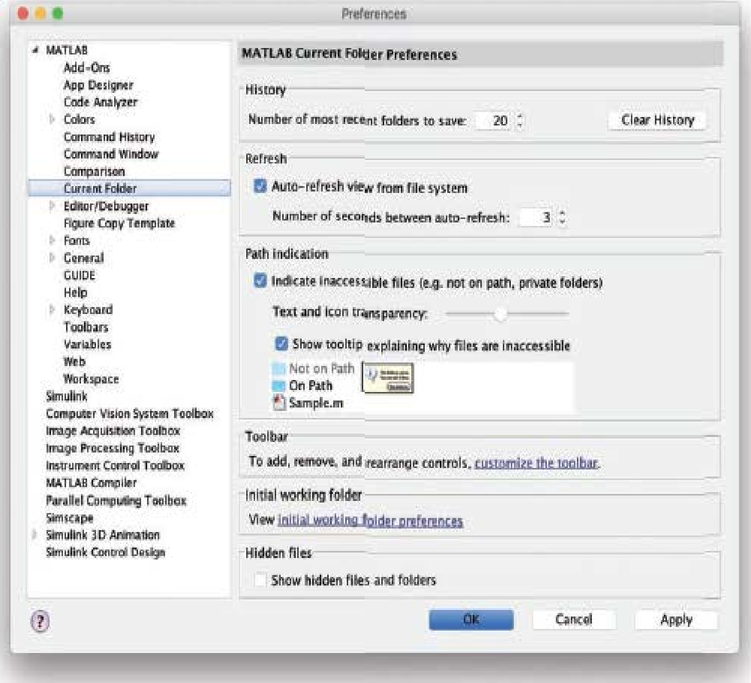
Select a file to view details



Customization

- In the top ribbon, navigate to: Home -> Environment -> Preferences

- Allows you to customize your MATLAB experience (colors, fonts, etc.)



MATLAB Current Folder Preferences

History

Number of most recent folders to save: 20 Clear History

Refresh

☑ Auto-refresh view from file system

Number of seconds between auto-refresh: 3

Path indication

☑ Indicate inaccessible files (e.g. not on path, private folders)

Text and icon transparency:

☑ Show tooltip explaining why files are inaccessible

Toolbar

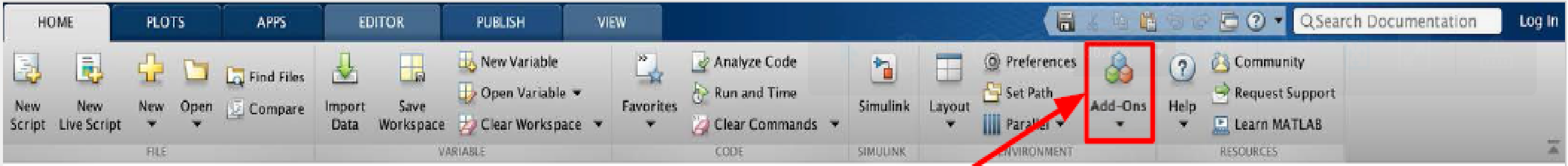
To add, remove, and rearrange controls, customize the toolbar.

Initial working folder

View initial working folder preferences

Hidden files

□ Show hidden files and folders

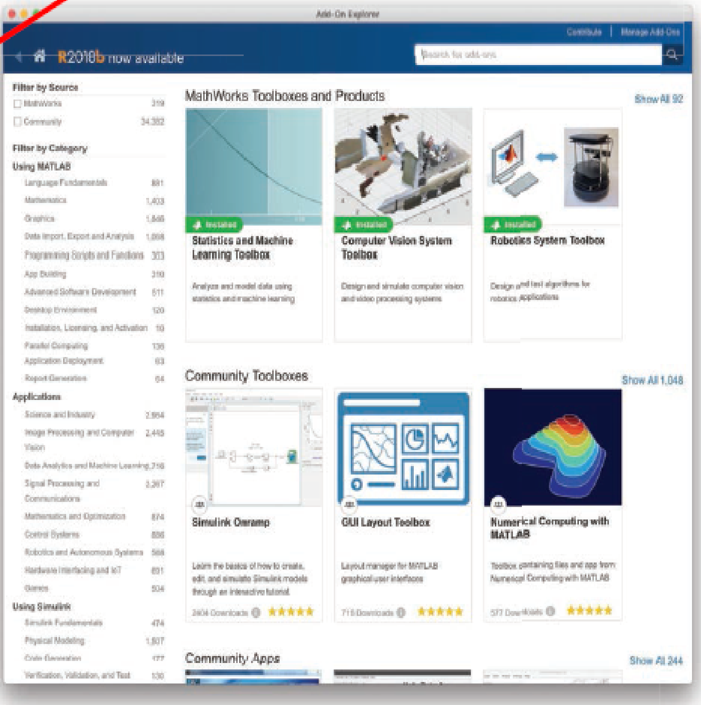


Installing Toolboxes

- In the top ribbon, navigate to: Home -> Environment -> Add-Ons

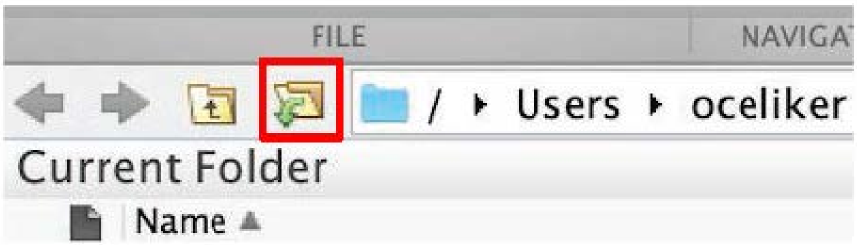
- Allows you to install toolboxes included with your license

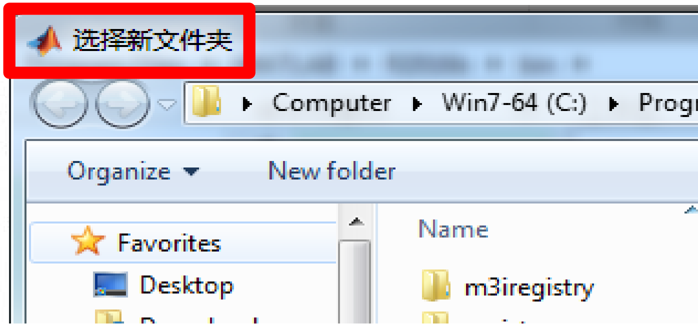
- Recommended toolboxes:
    - Curve Fitting Toolbox
    - Computer Vision System Toolbox
    - Image Processing Toolbox
    - Optimization Toolbox
    - Signal Processing Toolbox
    - and anything related to your field!


Add-On Explorer
Contribute
Manage Add-Ons
R2018b now available
Search for add-ons
Filter by Source
MathWorks 319
Community 34,392
Filter by Category
Using MATLAB
Language Fundamentals
Mathematics
Graphics
Data Import, Export and Analysis
Programming Scripts and Functions
App Building
Advanced Software Development
Desktop Environment
Installation, Licensing, and Activation
Parallel Computing
Application Deployment
Report Generation
Applications
Science and Industry
Image Processing and Computer Vision
Data Analytics and Machine Learning
Signal Processing and Communications
Mathematics and Optimization
Control Systems
Robotics and Autonomous Systems
Hardware Interfacing and IoT
Games
Using Simulink
Simulink Fundamentals
Physical Modeling
Code Generation
Verification, Validation, and Test
MathWorks Toolboxes and Products
Show All 92
Installed
Statistics and Machine Learning Toolbox
Analyze and model data using statistics and machine learning
Installed
Computer Vision System Toolbox
Design and simulate computer vision and video processing systems
Installed
Robotics System Toolbox
Design and test algorithms for robotics applications
Community Toolboxes
Show All 1,048
Simulink Onramp
Learn the basics of how to create, edit, and simulate Simulink models through an interactive tutorial
GUI Layout Toolbox
Layout manager for MATLAB graphical user interfaces
Numerical Computing with MATLAB
Toolbox containing files and app from Numerical Computing with MATLAB
Community Apps
Show All 244

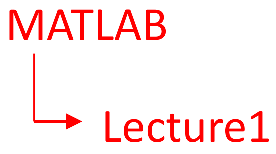
Making Folders

- Use folders to keep your programs organized
- To select a new folder, click "Browse" next to the file path





- In the MATLAB folder, make the following folder structure:



Help/Docs

- help
  - The most important command for learning MATLAB on your own!
- To get info on how to use a function:
  - help sin
    - Help lists related functions at the bottom and links to the documentation
- To get a nicer version of help with examples and easy-to-read description:
  - doc sin
- To search for a function by specifying keywords:
  - doc search sin trigonometric

II. SCRIPTS

Scripts: Overview

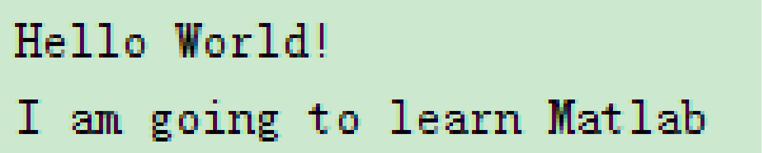
- Scripts are :
  - Collection of commands executed in sequence
  - Written in the Matlab editor
  - Saved as m-files (*.m)
- To create an m-file:
  - From command line: edit Lecture1.m
  - Click the "new script" button on the top left

Scripts: Some notes

- Comment!
  - Anything following a % sign is interpreted as a comment
  - The first continuous comment becomes the script's help file
  - Comment thoroughly to avoid wasting time later
  - Mark beginning of a code block by using %%
- Scripts are static, with no explicit input or output
- All variables created in a script retain their values after script execution

Exercise1-1: Scripts

- Make a script with the name HelloWord.m
- When run, the script should show the following text:



```
Hello World!
I am going to learn Matlab
```

Hint: use disp(…) to display strings.

Strings are written in single quotes. Eg: ‘ This is a string’.

III. MAKING VARIABLES

Variable Types

- Matlab is a “weakly typed” language
  - No need to declare a variable
- Most variables you will deal with are doubles, chars, vectors and matrices
- Other types are also supported, such as complex, symbolic, 16bits or 8bits integers(uint16 or uint8), etc.

Naming Variables

- To create a variable, simply assign a value to a name:

```
myNumberVariable = 3.14
myStringVariable = 'hello world!'
```

- Variable name rules
  - First character must be a LETTER
  - After that, any combination of numbers, letters and _
  - Names are CASE-SENSITIVE

Naming Variables (cont.)

Built-in variables (don't use these names for anything else):

i, j: can be used to indicate complex numbers

a = 2.5 + 3.4i

a = 2.5 + 3.4j

use ii, jj, kk, etc. for loop counters.

pi: has the value 3.1415……

ans: stores the result of the last unassigned value

Inf, -Inf: infinities

NaN: "Not a Number"

Scalars

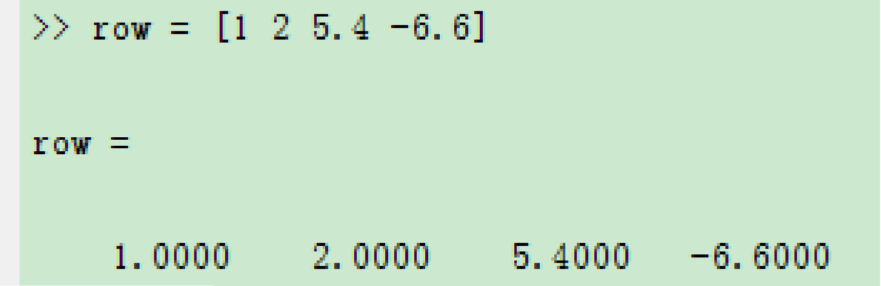
- A variable can be given a value explicitly
  - a = 10
  - Shows up in workspace
- Or as a function of explicit values and existing variables
  - c = 1.3 * 45 – 2 * a
- To suppress output, end the line with a semicolon
  - d = 13/3;

Arrays

- Like other programming languages, arrays are an important part of MATLAB
- Two types of arrays:
  - Matrix of numbers (either double or complex)
  - Cell array of objects (more advanced data structure)

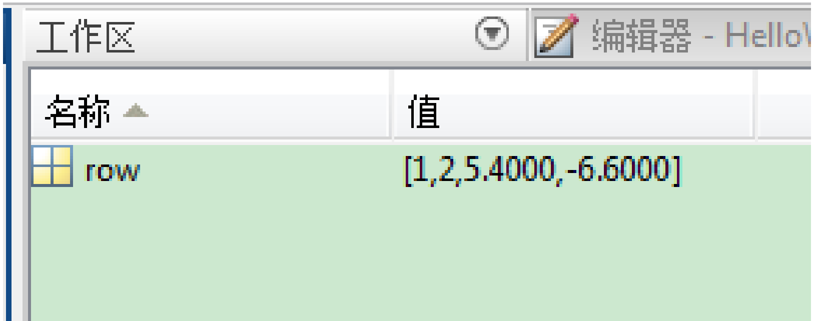
Row vectors

- Row vector: comma- or space-separated values between square brackets[ ]
  - row = [1 2 3 4 5 6]
  - Row = [1, 2, 3, 4, 5, 6]
- Command window:



```
>> row = [1 2 5.4 -6.6]

row =

    1.0000    2.0000    5.4000   -6.6000
```

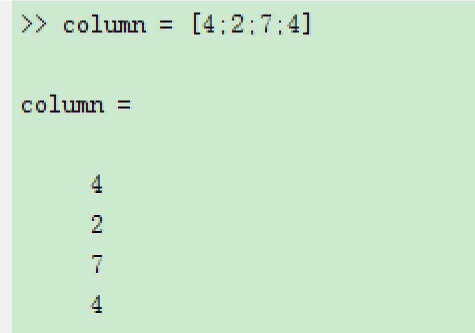
- Workspace:



名称	值
row	[1,2,5.4000,-6.6000]

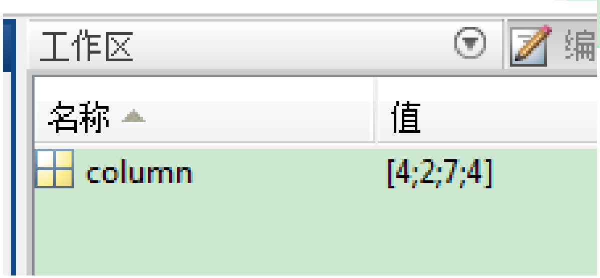
Column vectors

- Column vector: semicolon-separated values between square brackets [ ]
  - col = [1; 2; 3.2; 4; 6; -5.4];
- Command window



```
>> column = [4;2;7;4]

column =

     4
     2
     7
     4
```

- Workspace:

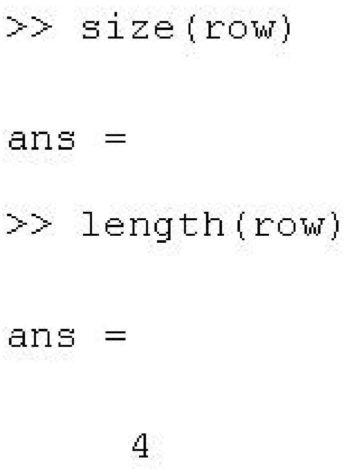


工作区

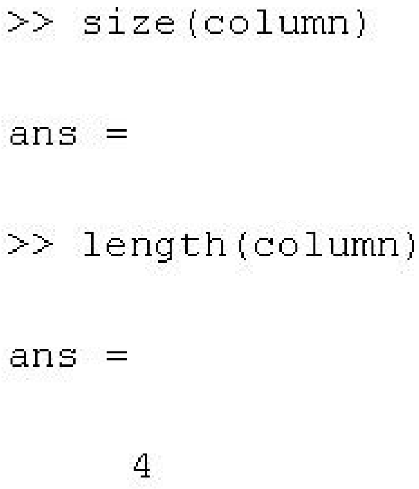
名称	值
column	[4;2;7;4]

Size and length

- You can tell the difference between a row and a column by:
  - Looking in the workspace;
  - Displaying the variable in the command window
  - Using the size function



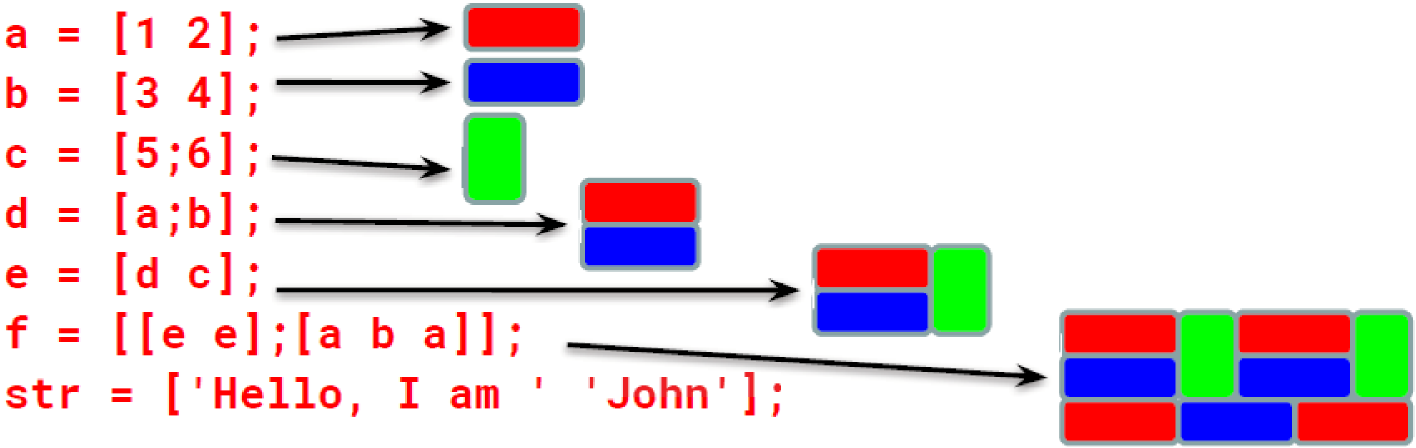
```
>> size(row)

ans =

>> length(row)

ans =

     4
```



```
>> size(column)

ans =

>> length(column)

ans =

     4
```

Matrices

- Make matrices like vectors, element by element
  - a = [1 2; 3 4]; → $a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
- By concatenating vectors or matrices(dimension matters)

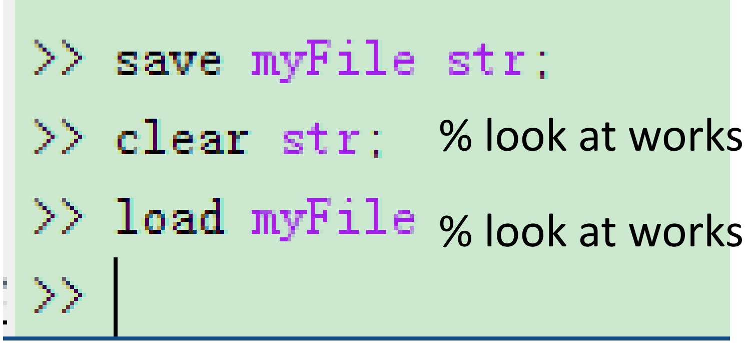


```
a = [1 2];
b = [3 4];
c = [5;6];
d = [a;b];
e = [d c];
f = [[e e];[a b a]];
str = ['Hello, I am ' 'John'];
```

- Strings are character vectors

save/clear/load

- Use save to save variables to a file
  - save myFile a b
  - Saves variables a and b to the file myFile.mat in the current directory
- clear and load



```
>> save myFile str;
>> clear str;      % look at workspace, a and b are gone
>> load myFile     % look at workspace, a and b are back
>> 
```

Exercise1-2: Variables

Get and save the current date and time

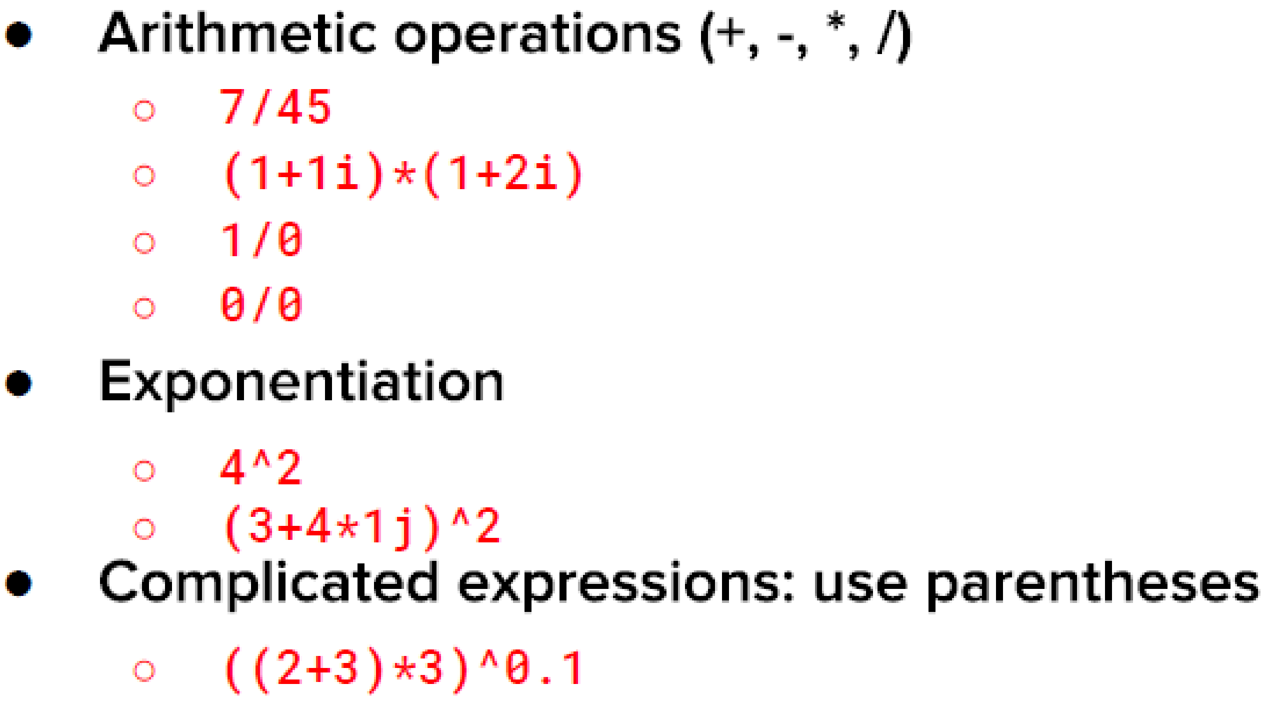
- Create a variable start using the function clock
- What is the size of start? Is it a row or column?
- What does start contain? See help clock
- Convert the vector start to a string. Use the function datestr and name the new variable startString
- Save start and startString into a mat file named startTime

Exercise1-3: Variables II

- In HelloWorld.m, read in variables you saved using load
- Display the following text:
    - ***I started learning MATLAB on [date, time]***
- Hint: use the disp command again
- Remember that strings are just vectors of characters, so you can join 2 strings by making a row vector with the 2 strings as sub-vectors

IV.
MANIPULATING VARIABLES

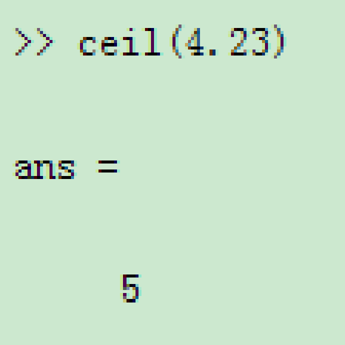
Basic Scalar Operations



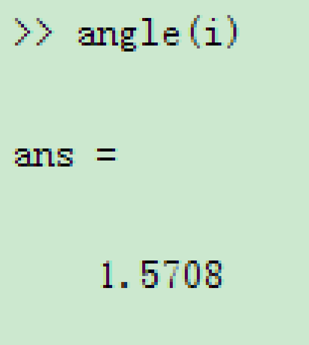
- Arithmetic operations (+, -, *, /)
  - `7/45`
  - `(1+1i)*(1+2i)`
  - `1/0`
  - `0/0`
- Exponentiation
  - `4^2`
  - `(3+4*1j)^2`
- Complicated expressions: use parentheses
  - `((2+3)*3)^0.1`

Built-in Functions

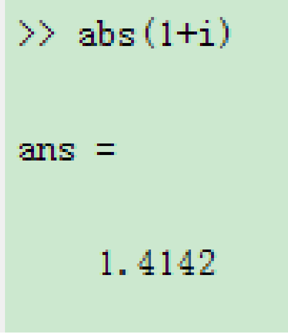
- MATLAB has an <u>enormous</u> library of built-in functions
- Call using parentheses, passing parameters to function
  - `sqrt(2)`
  - `log(2), log10(0.23)`
  - `cos(1.2), atan(-.8)`
  - `exp(2+4*1i)`
  - `round(1.4), floor(3.3), ceil(4.23)`
  - `angle(1i); abs(1+1i);`



```
>> ceil(4.23)

ans =

     5
```



```
>> angle(i)

ans =

    1.5708
```



```
>> abs(1+i)

ans =

    1.4142
```

Exercise1-4: Scalars

HelloWorld script:

- Your learning time constant is 0.75 hrs. Calculate seconds in 0.75 hrs and name this variable tau
- This class lasts 30 hrs. Calculate seconds in 30 hrs and name this variable endOfClass
- This equation describes your knowledge as a function of time t $k = 1 - e^{-t/\tau}$
- How well will you know MATLAB at endOfClass? Name this variable knowledgeAtEnd (use exp)
- Using the value of knowledgeAtEnd, display the phrase:
  - ***At the end of class, I will know X% of MATLAB.***
  - Hint: to convert a number to a string, us num2str

Transpose

- The transpose operator turns a column vector into a row vector, and vice versa
  - `a = [1 2 3 4+i]`
  - `transpose(a)`
  - `a'`
  - `a.'`
- The ' gives the Hermitian-transpose
  - Transposes and conjugates all complex numbers
- For vectors of real numbers .' and ' give same result
  - For transposing a vector, always use .' to be safe

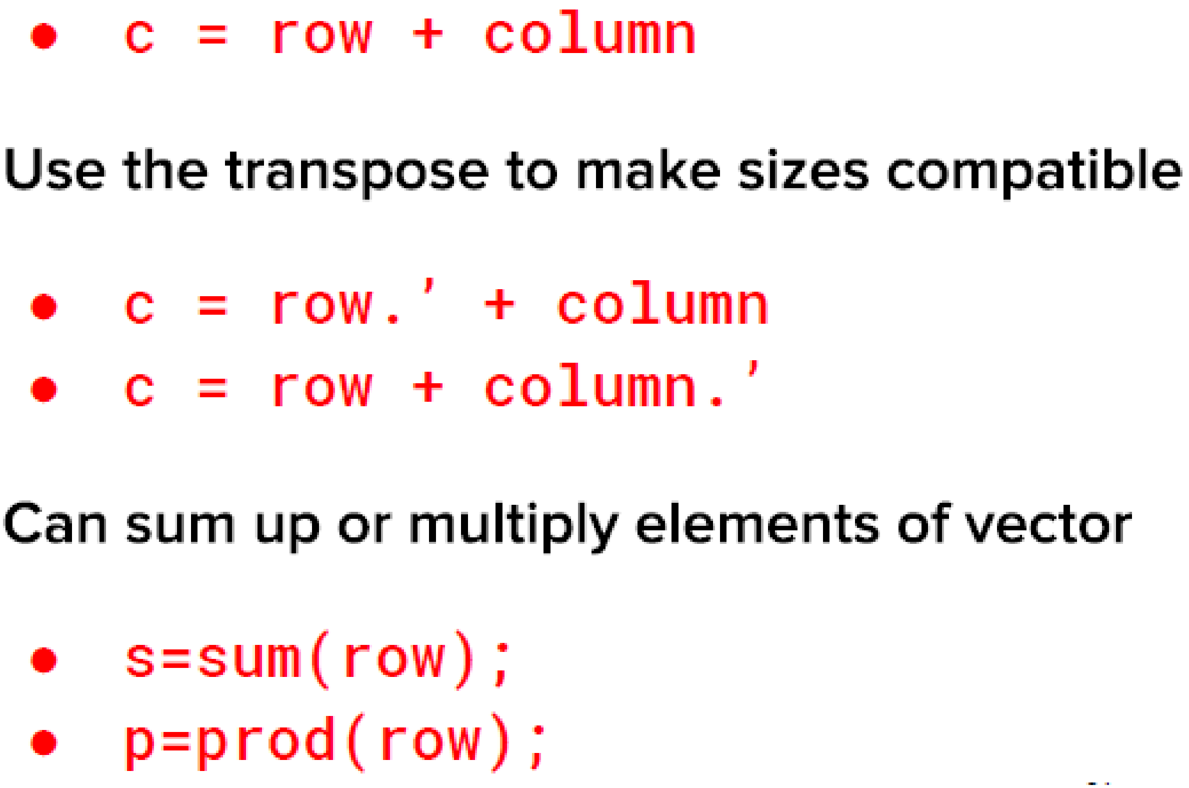
Addition and Subtraction

- Addition and subtraction are element-wise
- Sizes must match (unless one is a scalar):

$$\begin{array}{r}
[12 \quad 3 \quad 32 \quad -11] \\
+[2 \quad 11 \quad -30 \quad 32] \\
\hline
=[14 \quad 14 \quad 2 \quad 21]
\end{array}$$

$$\begin{bmatrix} 12 \\ 1 \\ -10 \\ 0 \end{bmatrix} - \begin{bmatrix} 3 \\ -1 \\ 13 \\ 33 \end{bmatrix} = \begin{bmatrix} 9 \\ 2 \\ -23 \\ -33 \end{bmatrix}$$

Addition and Subtraction



- `c = row + column`

Use the transpose to make sizes compatible

- `c = row.' + column`
- `c = row + column.'`

Can sum up or multiply elements of vector

- `s=sum(row);`
- `p=prod(row);`

Element-wise functions

- All the functions that work on scalars also work on vectors

  -
    ```
    t = [1 2 3];
    f = exp(t);
    ```
    is the same as
    ```
    f = [exp(1) exp(2) exp(3)];
    ```
- If in doubt, check a function's help file to see if it handles vectors element-wise
- Operators (* / ^) have two modes of operation
  - element-wise
  - standard

Element-wise functions

- To do element-wise operations, use the dot: . (.*, ./, .^)
- BOTH dimensions must match (unless one is scalar)!

```
a=[1 2 3];b=[4;2;1];

a.*b  , a./b  , a.^b → all errors

a.*b.', a./b.', a.^(b.') → all valid
```

Operators

- Multiplication can be done in a standard way or element-wise
- Standard multiplication (*) is matrix product
  - Remember from linear algebra: inner dimensions must MATCH!!
- Standard exponentiation (^) can only be done on square matrices or scalars
- Left and right division (/ \) is same as multiplying by inverse
  - Our recommendation: for now, just multiply by inverse (more on this later)

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} * \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = 11$$

$$1\times3 * 3\times1 = 1\times1$$

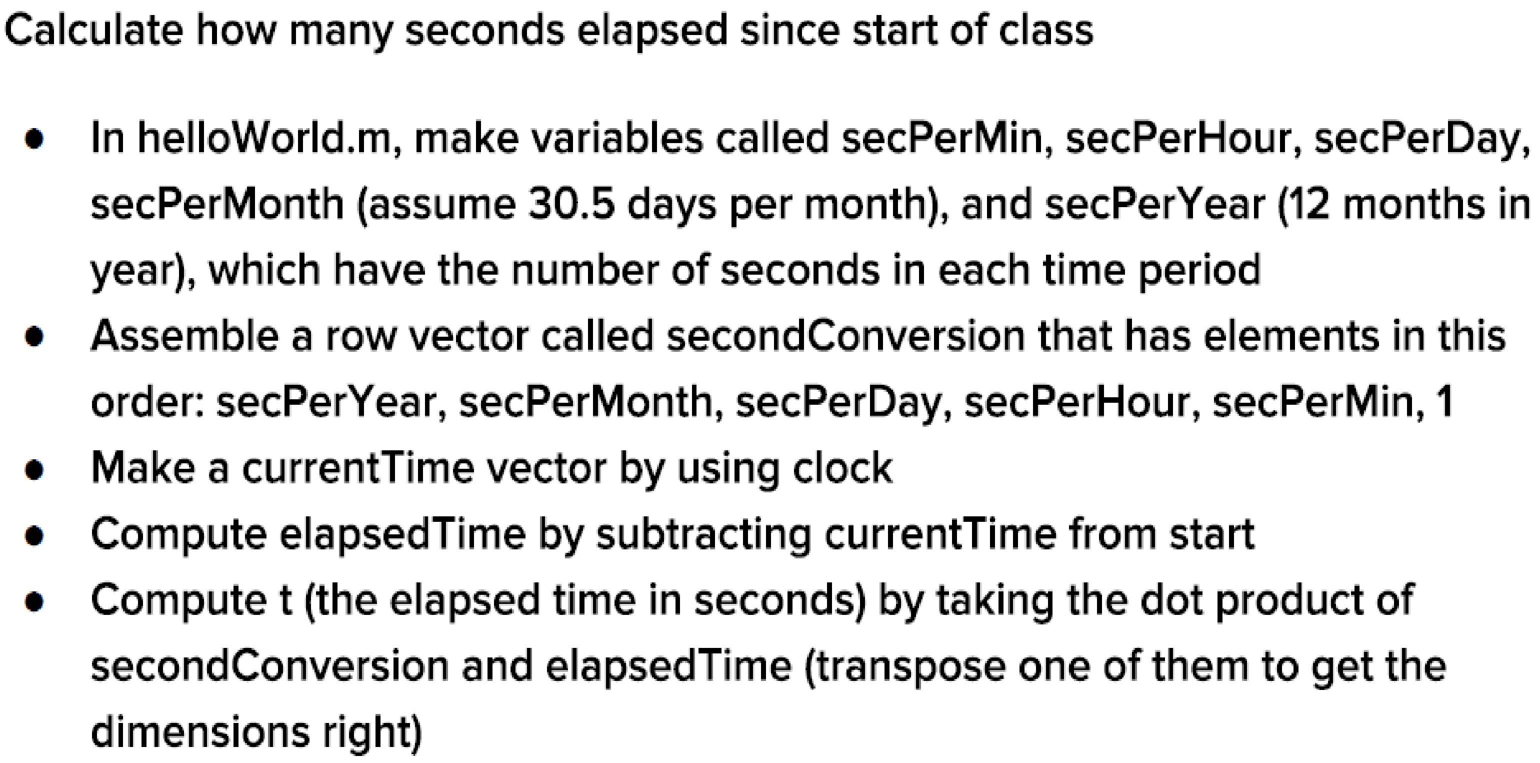
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \wedge 2 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Must be square to do powers

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 9 \\ 6 & 12 & 18 \\ 9 & 18 & 27 \end{bmatrix}$$

$$3\times3 * 3\times3 = 3\times3$$

Exercise1-5: Vector Operations



Calculate how many seconds elapsed since start of class

- In helloWorld.m, make variables called secPerMin, secPerHour, secPerDay, secPerMonth (assume 30.5 days per month), and secPerYear (12 months in year), which have the number of seconds in each time period
- Assemble a row vector called secondConversion that has elements in this order: secPerYear, secPerMonth, secPerDay, secPerHour, secPerMin, 1
- Make a currentTime vector by using clock
- Compute elapsedTime by subtracting currentTime from start
- Compute t (the elapsed time in seconds) by taking the dot product of secondConversion and elapsedTime (transpose one of them to get the dimensions right)

Exercise1-5: Vector Operations

Display the current state of your knowledge

- Calculate currentKnowledge using the same relationship as before, and the t we just calculated:

$$k = 1 - e^{-t/\tau}$$

- Display the following text:
  At this time, I know X% of MATLAB

Automatic Initialization

- Initialize a vector of **ones**, **zeros**, or **random** numbers
  - » `o=ones(1,10)`
    - ➢ Row vector with 10 elements, all 1
  - » `z=zeros(23,1)`
    - ➢ Column vector with 23 elements, all 0
  - » `r=rand(1,45)`
    - ➢ Row vector with 45 elements (uniform (0,1))
  - » `n=nan(1,69)`
    - ➢ Row vector of NaNs (representing uninitialized variables)

Automatic Initialization

- To initialize a linear vector of values use **linspace**
  - `a=linspace(0,10,5)`
    - Starts at 0, ends at 10 (inclusive), 5 values

- Can also use colon operator (**:**)
  - `b=0:2:10`
    - Starts at 0, increments by 2, and ends at or before 10
    - Increment can be decimal or negative
  - `c=1:5`
    - If increment is not specified, default is 1

- To initialize logarithmically spaced values use **logspace**
    - Similar to **linspace**, but see **help**

Exercise1-6: Vector Functions

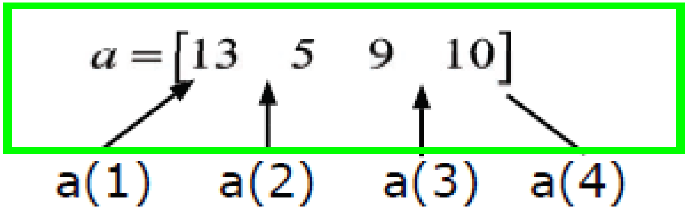
Calculate your learning trajectory

- In helloWorld.m, make a linear time vector `tVec` that has 10,000 samples between 0 and `endOfClass`
- Calculate the value of your knowledge (call it `knowledgeVec`) at each of these time points using the same equation as before:

$$k = 1 - e^{-t/\tau}$$

Vector Indexing

- MATLAB indexing starts with **1**, not **0**
  - ➢ We will not respond to any emails where this is the problem.
- a(n) returns the nth element



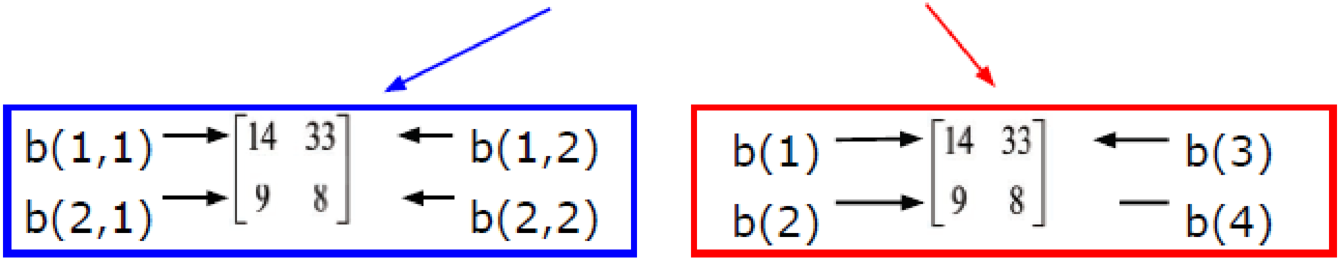
$a = [13 \quad 5 \quad 9 \quad 10]$

- The index argument can be a vector. In this case, each element is looked up individually, and returned as a vector of the same size as the index vector.

```
» x=[12 13 5 8];
```

Matrix Indexing

- Matrices can be indexed in two ways
  - using **subscripts** (row and column)
  - using linear **indices** (as if matrix is a vector)
- Matrix indexing: subscripts or linear indices



b(1,1) → $\begin{bmatrix} 14 & 33 \\ 9 & 8 \end{bmatrix}$ ← b(1,2)

b(2,1) → ← b(2,2)

b(1) → $\begin{bmatrix} 14 & 33 \\ 9 & 8 \end{bmatrix}$ ← b(3)

b(2) → — b(4)

- Picking submatrices

```
» A = rand(5) % shorthand for 5x5 matrix
```

Advanced Indexing 1

- To select rows or columns of a matrix, use the :

$$c = \begin{bmatrix} 12 & 5 \\ -2 & 13 \end{bmatrix}$$

```
» d=c(1,:);          d=[12 5];
» e=c(:,2);          e=[5;13];
» c(2,:)=[3 6];   %replaces second row of c
```

Advanced Indexing 2

- MATLAB contains functions to help you find desired values

```
» vec = [5 3 1 9 7]
```

- To get the minimum value and its index (similar for `max`):

```
» [minVal,minInd] = min(vec);
```

- To find the indices of specific values or ranges

```
» ind = find(vec == 9); vec(ind) = 8;
» ind = find(vec > 2 & vec < 6);
```

  - ➢ find expressions can be very complex, more on this later
  - ➢ When possible, **logical indexing** is faster than find!
  - ➢ E.g., `vec(vec == 9) = 8;`

Exercise1-7: Vector Functions

When will you know 50% of MATLAB?

- First, find the index where **knowledgeVec** is closest to 0.5. Mathematically, what you want is the index where the value of ~ $\left|knowledgeVec-0.5\right|$ is at a minimum (use **abs** and **min**)
- Next, use that index to look up the corresponding time in **tVec** and name this time **halfTime**
- Finally, display the string:
  Convert **halfTime** to days by using secPerDay. I will know half of MATLAB after X days

V. BASIC PLOTTING

Plotting

- Example

```
» x=linspace(0,4*pi,10);
» y=sin(x);
```

- Plot values against their index

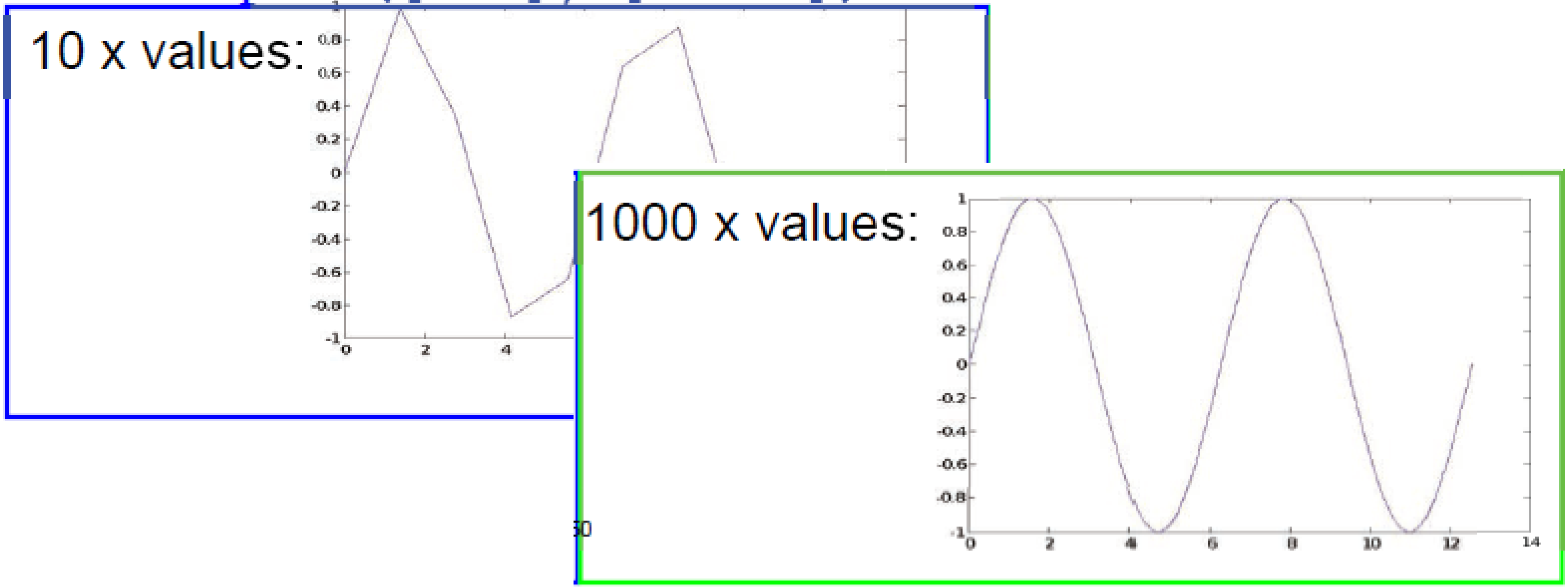
```
» plot(y);
```

- Usually we want to plot y versus x

```
» plot(x,y);
```

What does plot do?

- **plot** generates dots at each (x,y) pair
  and then connects the dots with a line
- To make plot of a function look smoother, evaluate at more points
  - » `x=linspace(0,4*pi,1000);`
  - » `plot(x,sin(x));`
- x and y vectors must be same size or else you'll get an error
  - » `plot([1 2], [1 2 3])`


10 x values:

1000 x values:

Axis

- Built-in axis modes (see `doc axis` for more modes)
  - » `axis square`
    - ➢ makes the current axis look like a square box
  - » `axis tight`
    - ➢ fits axes to data
  - » `axis equal`
    - ➢ makes x and y scales the same
  - » `axis xy`
    - ➢ puts the origin in the lower left corner (default for plots)
  - » `axis ij`
    - ➢ puts the origin in the upper left corner (default for matrices/images)

Axis Label, limit
figure title, legend

```
xlabel( ‘label’ )
ylabel( ‘label’ )
```

```
xlimit([xmin xmax])
ylimit([ymin ymax])
```

```
title( ‘figure title’ )
```

```
legend( ‘str1’ ,’ str2’ ,....)
```

Copy/Paste Figures

- Figures can be pasted into other apps (word, ppt, etc)
- *Edit → copy options→ figure copy template*

pro d ppt

- *E* ure
- P st

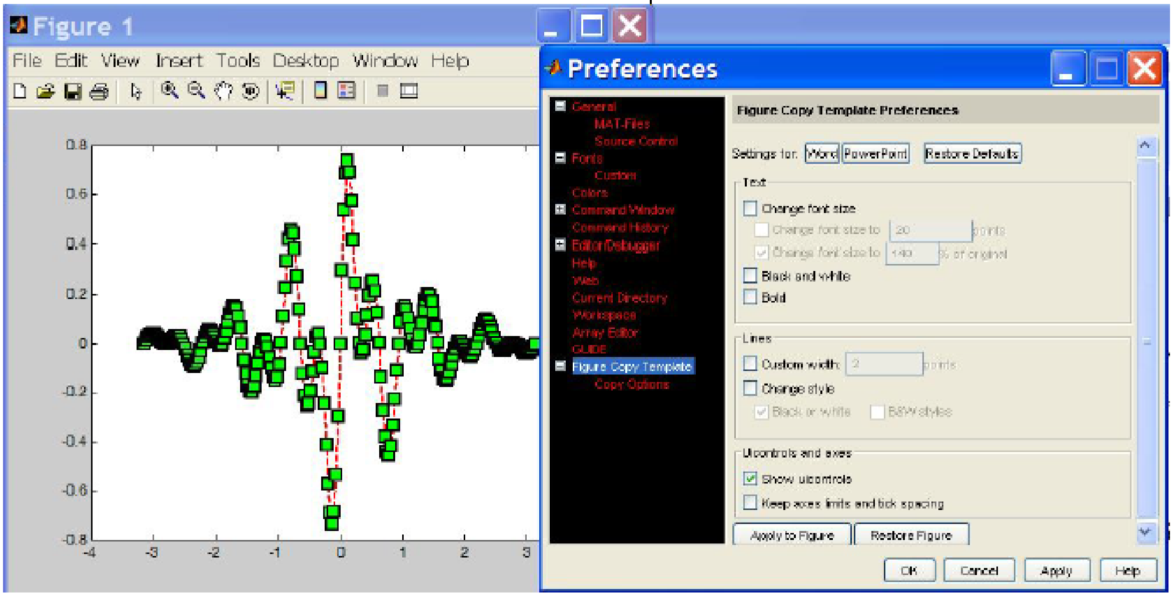


Figure Copy Template Preferences

Settings for: Word PowerPoint Restore Defaults

Text
- ☐ Change font size
  - ☐ Change font size to 20 points
  - ☑ Change font size to 140 % of original
- ☐ Black and white
- ☐ Bold

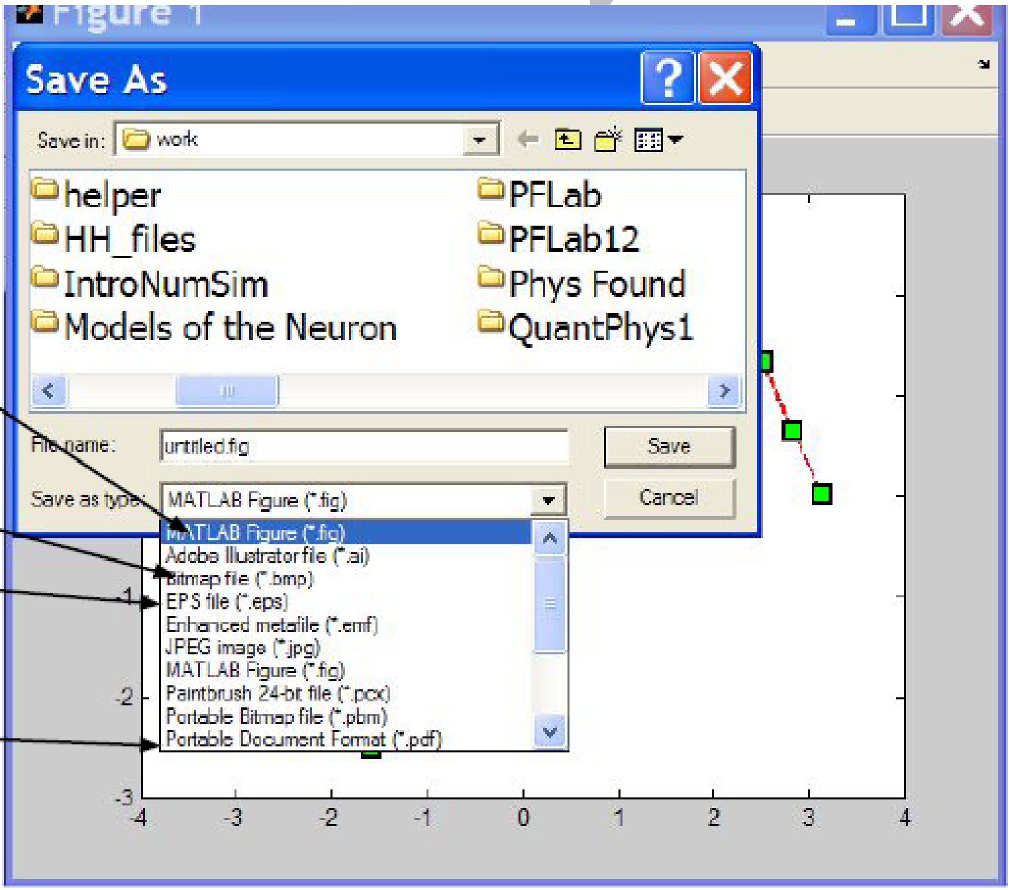
Lines
- ☐ Custom width: 2 points
- ☐ Change style
  - ☑ Black or white ☐ B&W styles

Uicontrols and axes
- ☑ Show uicontrols
- ☐ Keep axes limits and tick spacing

Saving Figures

Figures can be saved in many formats.

.



Save in: work

File name: untitled.fig

Save as type: MATLAB Figure (*.fig)

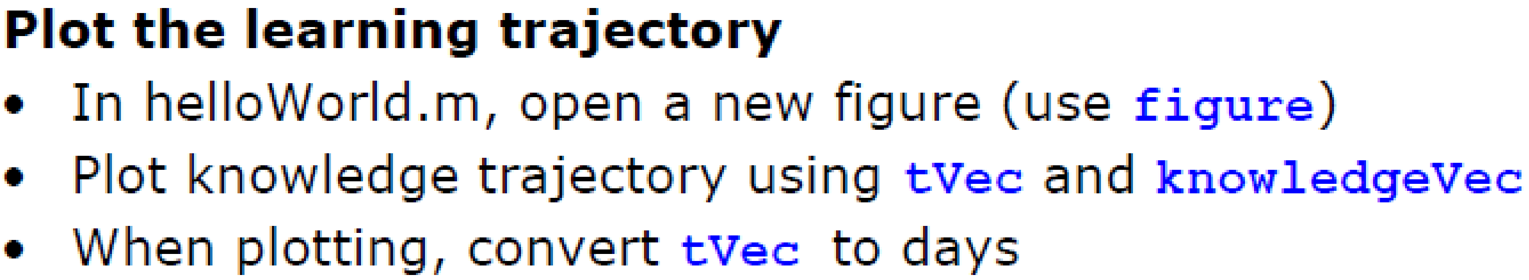
.fig preserves all information

.bmp uncompressed image

.eps high-quality scaleable format

.pdf compressed image

Exercise1-8: Plotting



Plot the learning trajectory

- In helloWorld.m, open a new figure (use `figure`)
- Plot knowledge trajectory using `tVec` and `knowledgeVec`
- When plotting, convert `tVec` to days

柱状图

bar(x, width, ‘mode’)

width: 竖条宽度，>1则竖条重叠；

mode：默认分组式’grouped’； ‘stacked’堆栈式

bar3(x, width, ‘mode’) %三维柱状图

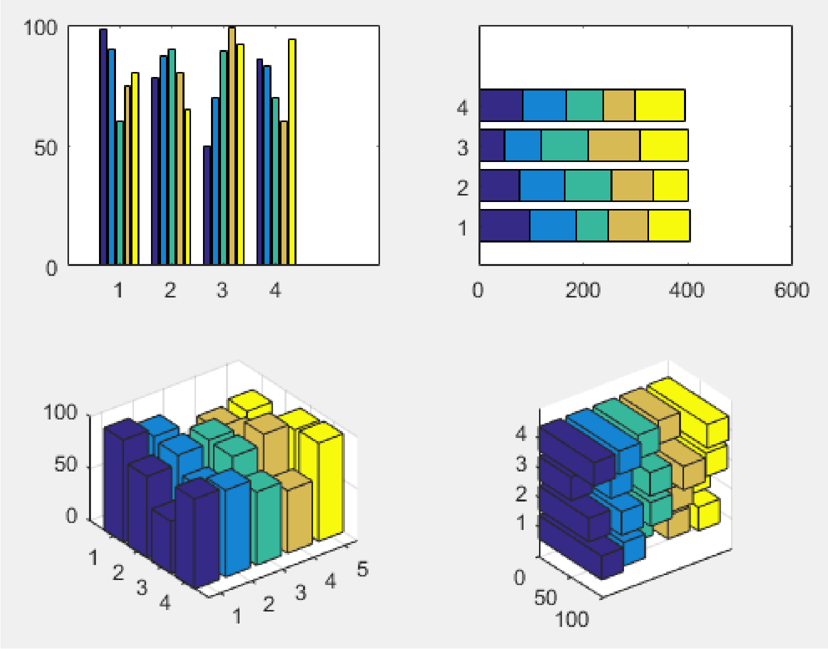
barh(x, width, ‘mode’), barh3(x, y, width, ‘mode’)

柱为水平

例：已知某班4位同学，在5门课程考试中取得如下表所示成绩，分别用垂直柱状图、水平柱状图、三维垂直柱状图和三维水平柱状图显示成绩。

	课程1	课程2	课程3	课程4	课程5
学生1	98	90	60	75	80
学生2	78	87	90	80	65
学生3	50	70	89	99	92
学生4	86	83	70	60	94

```
>> x=[98 90 60 75 80;78 87 90 80 65;50 70 89 99 92;86 83 70 60 94]
>> subplot(2,2,1);
>> bar(x);
>> subplot(2,2,2);barh(x,'stacked');
>> subplot(2,2,3);bar3(x);
>> subplot(2,2,4);bar3h(x);
```



100
50
0
1
2
3
4
4
3
2
1
0
200
400
600
100
50
0
1
2
3
4
1
2
3
4
5
4
3
2
1
0
50
100