

# Math 151A

TA: Bumsu Kim

# Today...

- Introduction
- MATLAB Tutorial
  - License and Installation
  - Basic Syntax
- GitHub Basics

# Introduction

---

- TA: Bumsu Kim
- email: [bumsu@ucla.edu](mailto:bumsu@ucla.edu) (Please use a header like [151A])
- Slides and supplementary materials can be found on my GitHub Repo
  - Link to the repo can be found on BruinLearn
  - Or, use this: <https://github.com/bumsu-kim/151A-1B-Summer22>
- Office Hours → Same Zoom link with Discussion
- We **highly recommend** using **Campuswire** for asking questions
- Polls: <https://pollev.com/bumsukim297>

# MATLAB – License and Installation

---

- Google “ucla matlab” and you’ll find “How to get MATLAB | UCLA Software...”  
[<https://softwarecentral.ucla.edu/matlab-getmatlab>]
- Follow the instructions there – it is much easier than installing C++ IDE (e.g. Visual Studio) on your computer
- Note:
  - You need to create a MathWorks® account using your UCLA email address
  - If you don’t have access to the valid UCLA license, you can use “free trial” (30 days)

# MATLAB – License and Installation

[HOME](#)[PRODUCTS](#)[NEWS](#)[SWC STORE](#)[LICENSING](#)[RESOURCES](#)[ABOUT](#)

## How to Get MATLAB

MATLAB software is available as a self-service installation for individual user computers. If you have installed MATLAB in the past, please re-download and install to access the full set of tools available under the campus license.

### How to get MATLAB for Single User

1. Click this

1. Visit the [UCLA MATLAB Portal](#)

Select 'Sign in to get started' under the Get MATLAB and Simulink section.

2. Create a MathWorks account using your UCLA email address which must have ".ucla.edu" as part of the domain. The MATLAB license is only for current UCLA Students, Faculty and Staff with an ucla.edu email address. (You may be asked to login with your UCLA BOL account to validate your eligibility first.)

3. Log in to your MathWorks account.

4. Click the download button for the current release (you can also download previous releases here).

5. Choose a supported platform and download the installer.

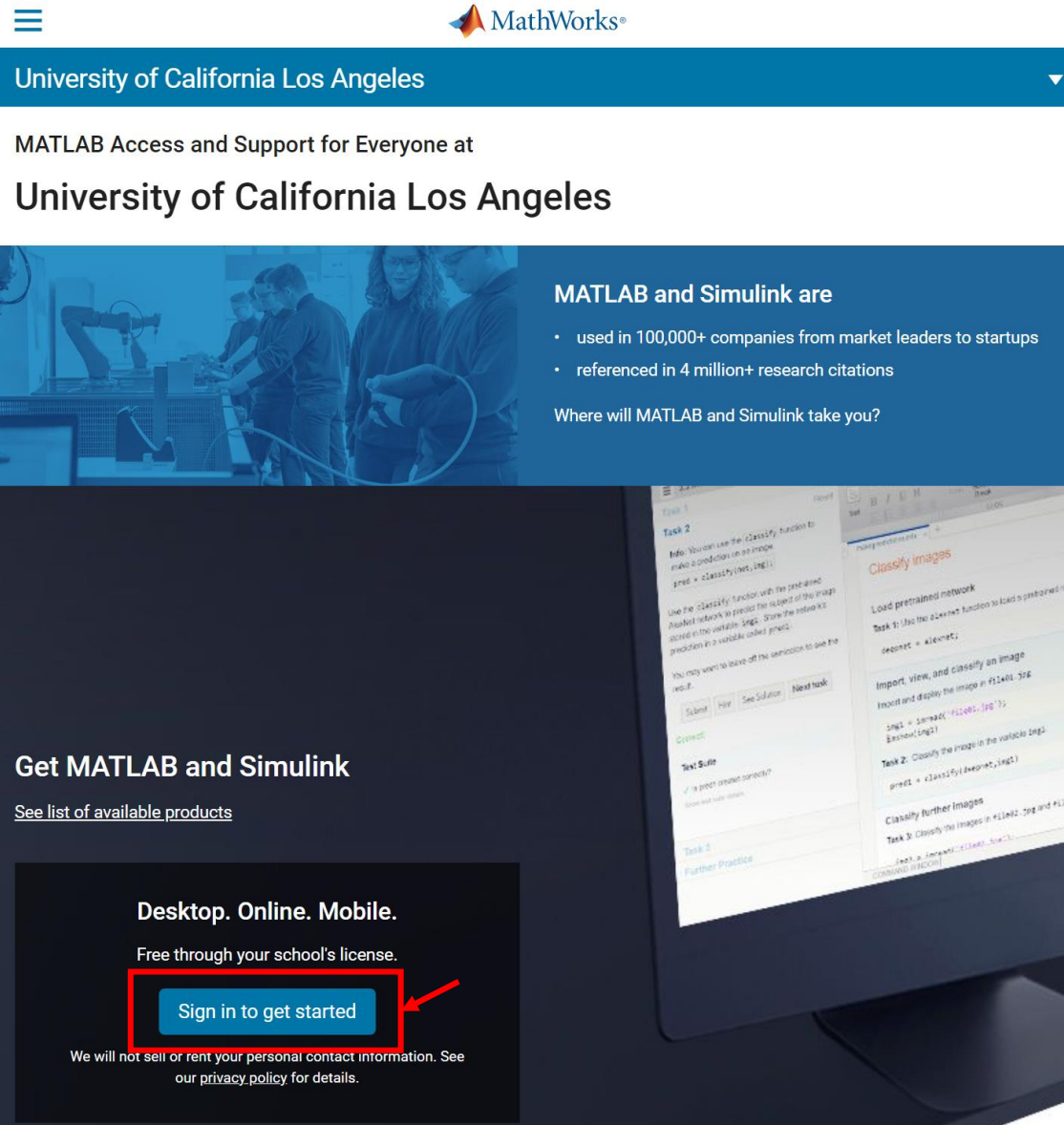


### Recent News

- › [Windows 10 Resources and Slide Deck](#)
- › [Zoom Transition Update](#)
- › [Software Central Debuts New Website](#)
- › [Windows 10 Presentation](#)
- › [Save the Date Windows Server 2016 & System Center 2016 - Microsoft Roadshow](#)



# MATLAB – License and Installation



University of California Los Angeles

MATLAB Access and Support for Everyone at  
University of California Los Angeles

**MATLAB and Simulink are**

- used in 100,000+ companies from market leaders to startups
- referenced in 4 million+ research citations

Where will MATLAB and Simulink take you?

**Get MATLAB and Simulink**

[See list of available products](#)

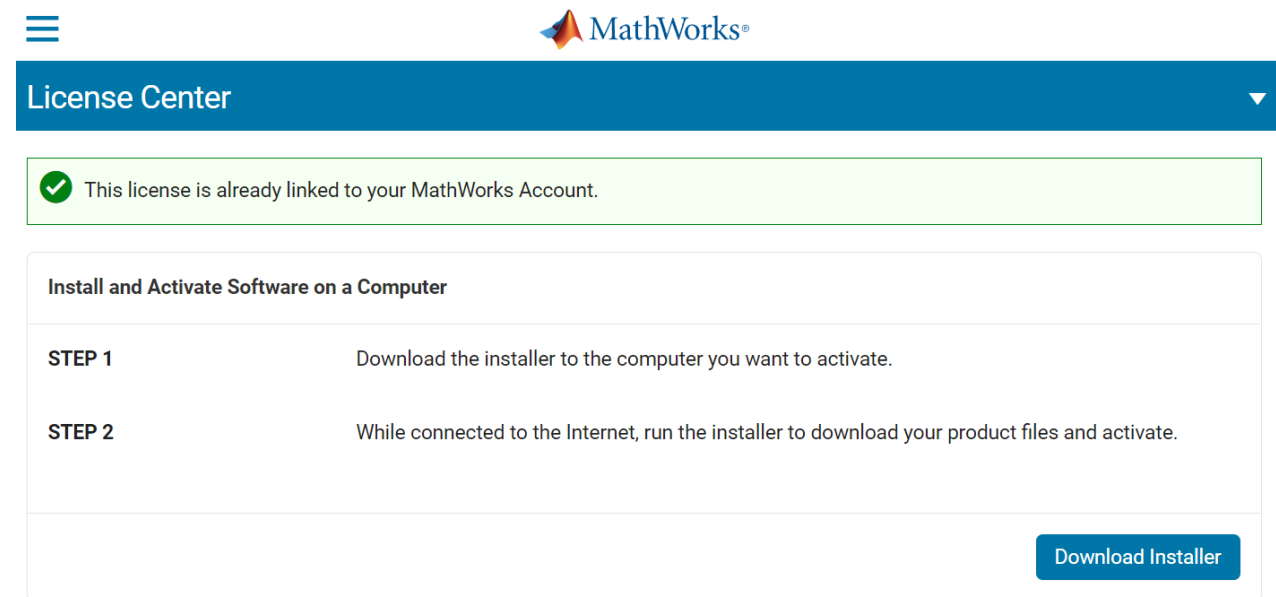
**Desktop. Online. Mobile.**

Free through your school's license.

**Sign in to get started**

We will not sell or rent your personal contact information. See our privacy policy for details.

✓ 2. Click the red box on the left, and create a MathWorks account using your UCLA email address (having “.ucla.edu” domain)



License Center

✓ This license is already linked to your MathWorks Account.

**Install and Activate Software on a Computer**

**STEP 1** Download the installer to the computer you want to activate.

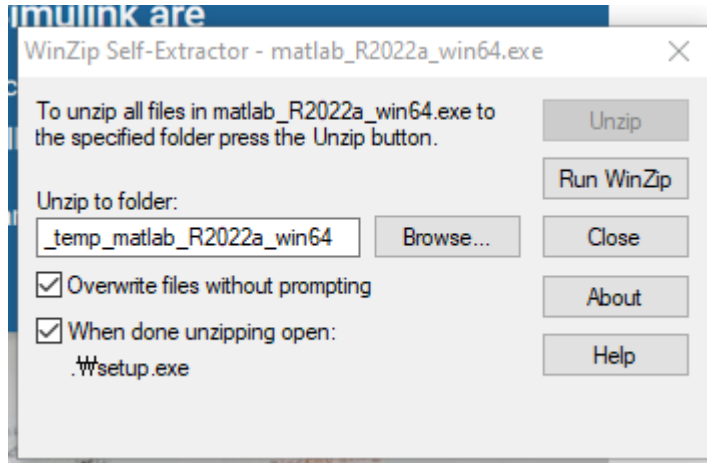
**STEP 2** While connected to the Internet, run the installer to download your product files and activate.

[Download Installer](#)

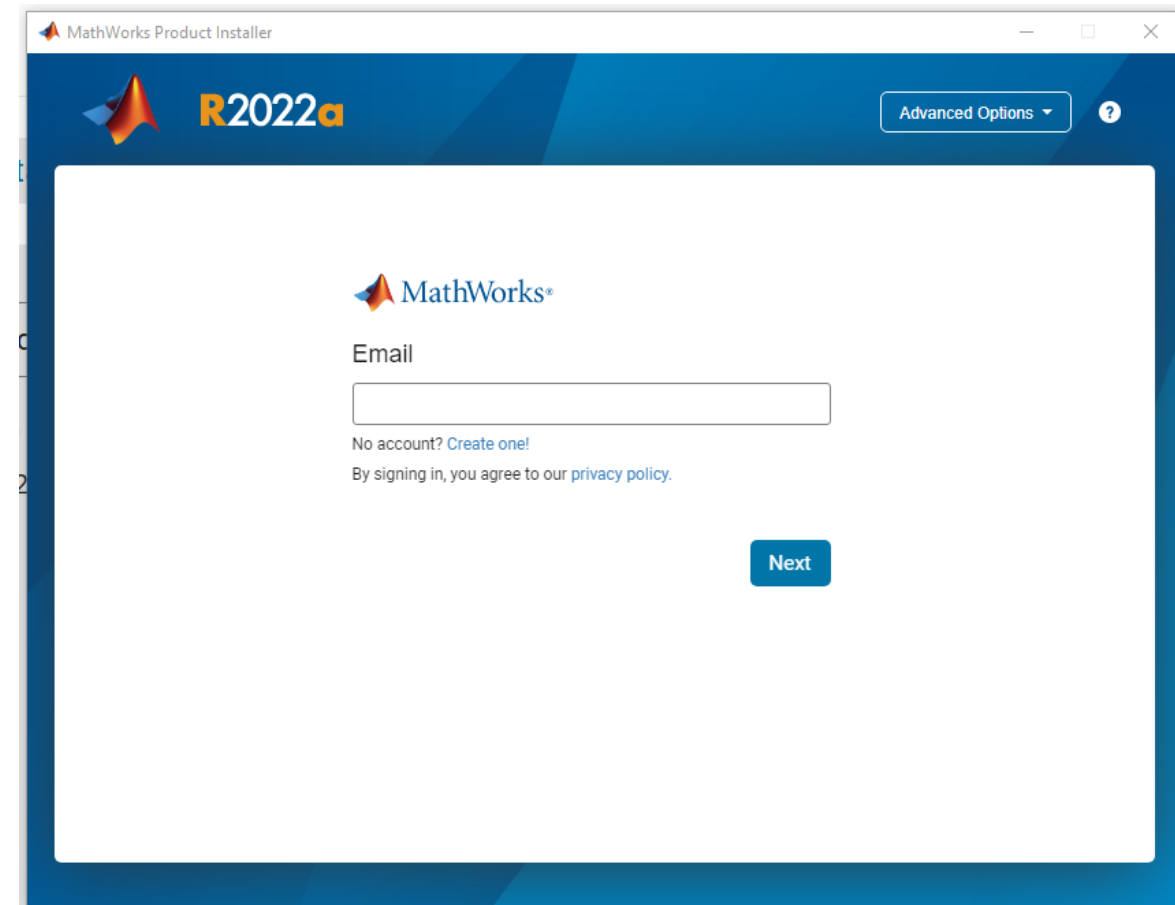
↑ 3. After you get the license you can download the installer, and run it. (You can select any version, e.g. R2022a)

# MATLAB – License and Installation

↓ 4. When you run the installer, it will extract the required component...

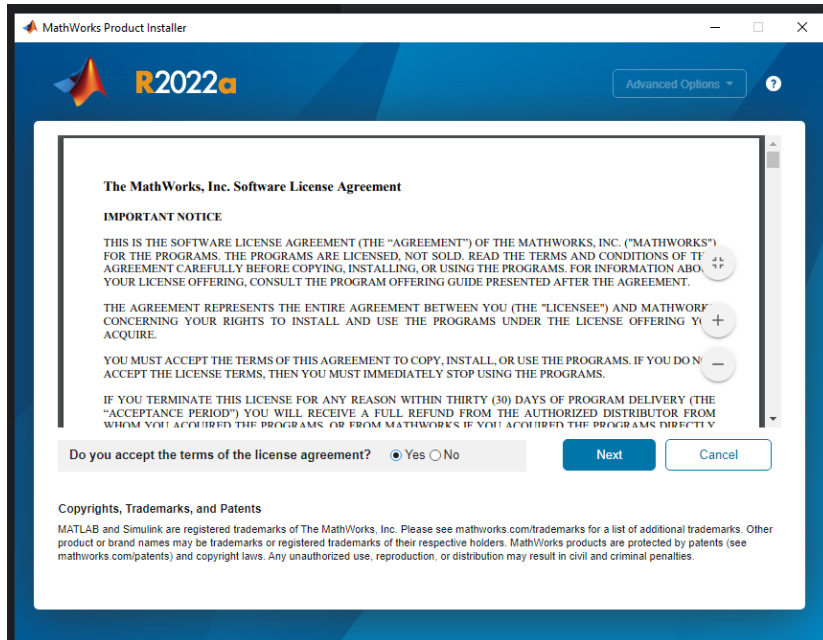


5. And then you will be prompted to log in  
Use your MathWorks account you just created →

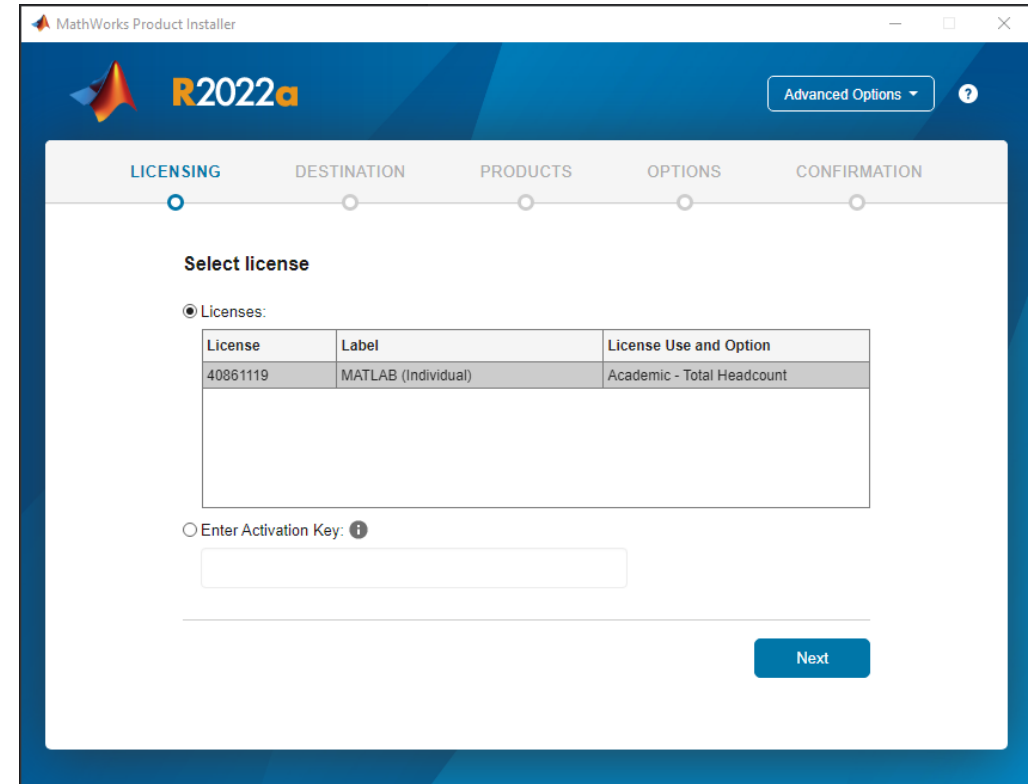


# MATLAB – License and Installation

↓ 6. If you agree and click next,



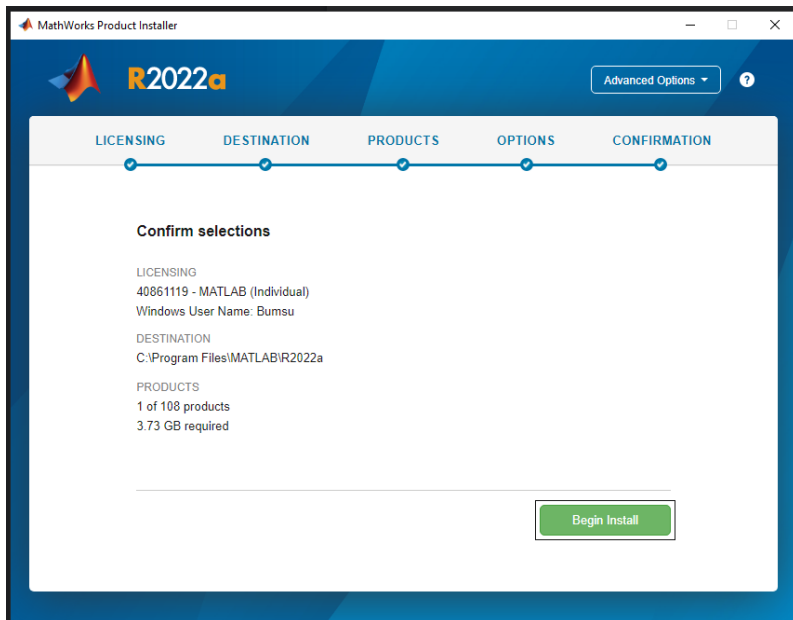
↓ 7. You will find your license (you don't need Activation Key.)  
Click Next again.



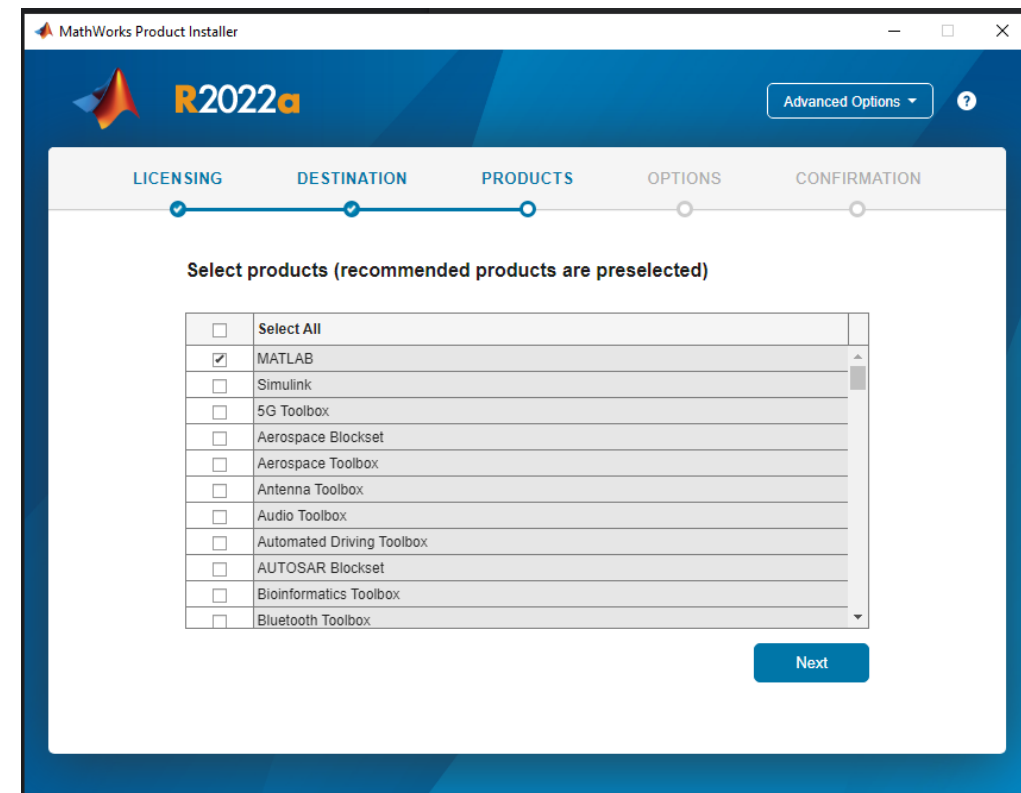


# MATLAB – License and Installation

- 8. Select the destination (installation) folder
  - Recommend to use the default
- 9. Then there appear a lot of tools, but you don't need them at all for this course ↓
  - Later you can install those toolboxes whenever you want
- 10. Click next, and that's it! ↓



It'll take a few minutes ...  
(or more if you selected  
more tools!)



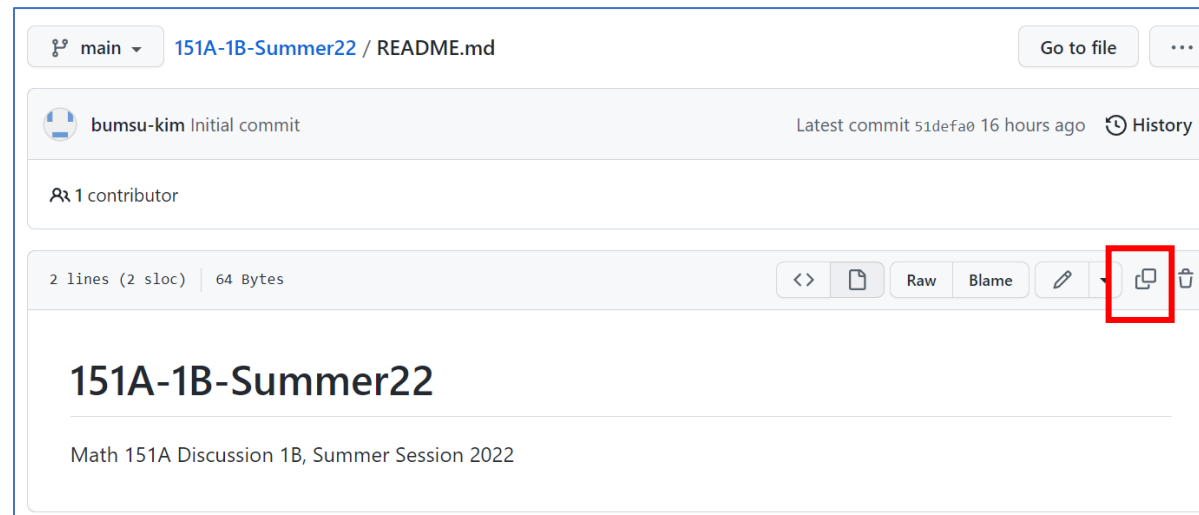
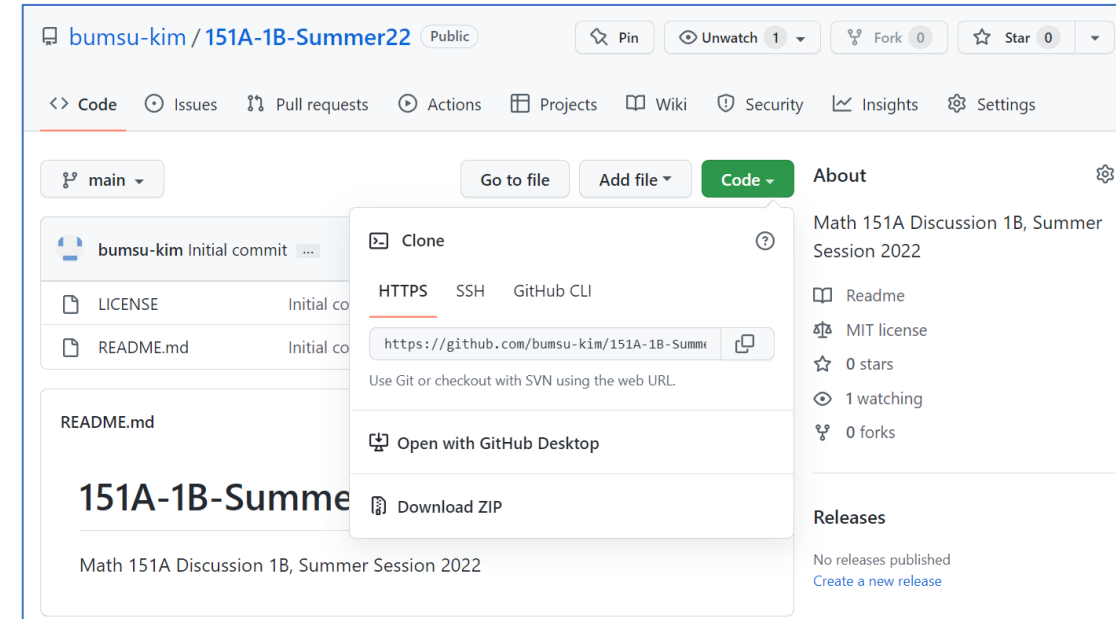
# MATLAB – Basics

---

- This course is not about programming languages, but algorithms
- But, we still need to know the basic components and syntax of MATLAB!
- Let's first talk about GitHub
  - I will share course materials through my GitHub repo, as announced
- Three options:
  - Git on Command Line → If you know this, you don't need the other options
  - GitHub Website → Easy to use, not easy to keep track of the latest version
  - GitHub Desktop App → Easiest for syncing, but need to install

# GitHub on Website and GitHub Desktop App

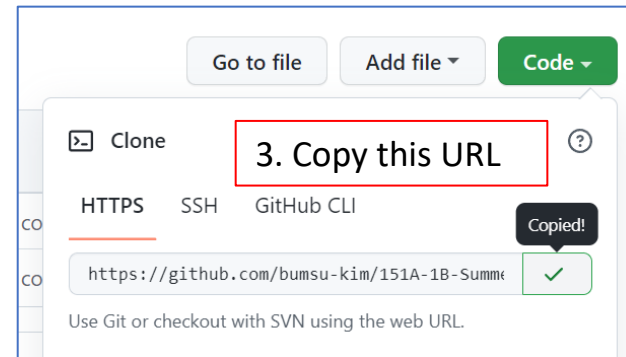
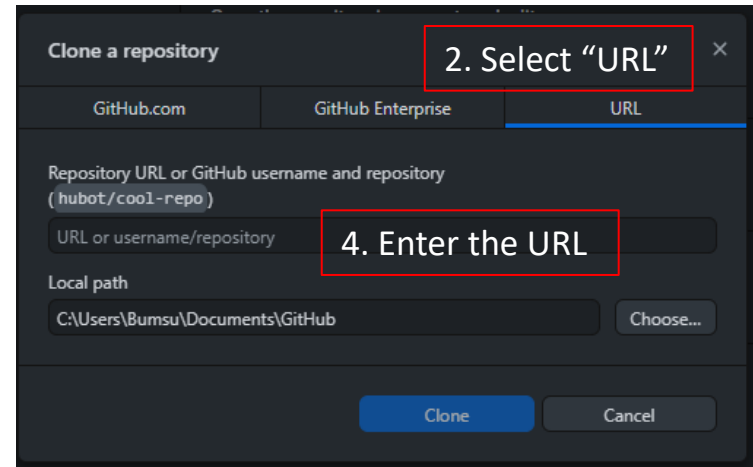
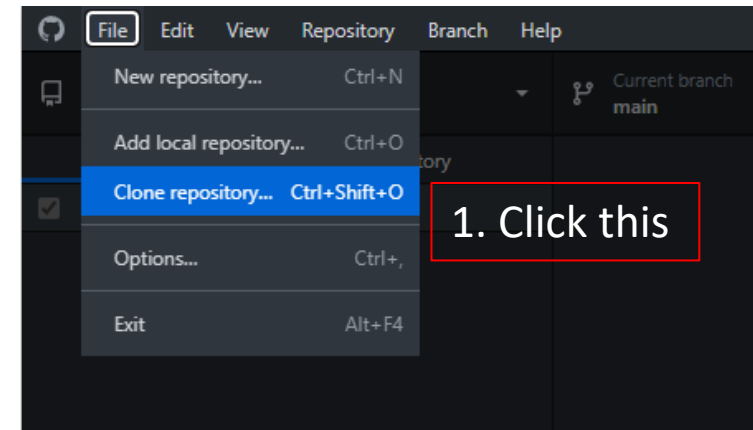
- 1. GitHub Website (e.g. <https://github.com/bumsu-kim/151A-1B-Summer22>)
  - Click the green box (“Code”)
  - Then you can “download ZIP”
- Pros: Very easy to download the whole repo
- Cons: need to download the whole repo every time
- You can also open a file and copy its content, too.



“Copy raw contents”  
Button

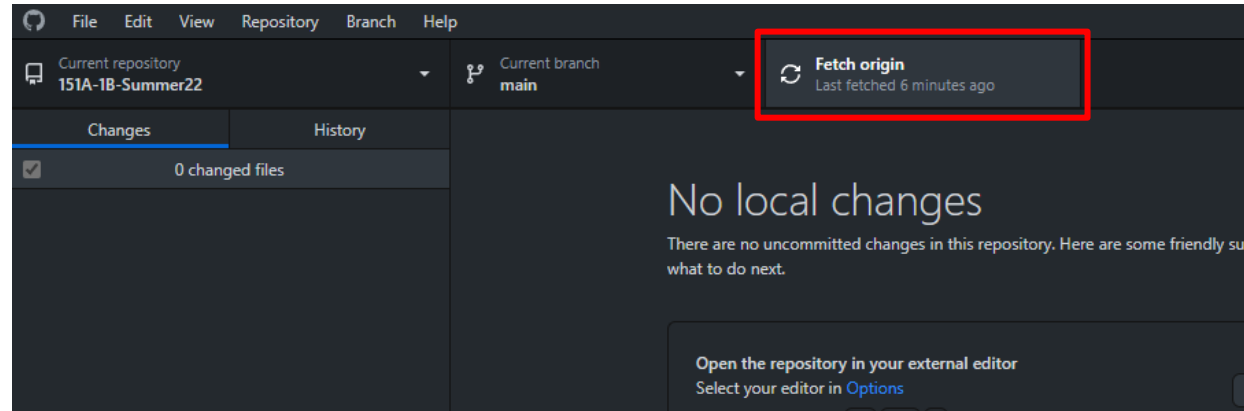
# GitHub on Website and GitHub Desktop App

- 2. GitHub Desktop App (<https://desktop.github.com/>)
    - Download the app from the link
  - Pros: Need to install
  - Cons: very easy to sync
- 
- 1. After you install it, “clone” a repo
  - 2. For my repo for this course, click “URL”
  - 3. Go to my repo on your browser and copy the URL
  - 4. Enter this to the app and clone

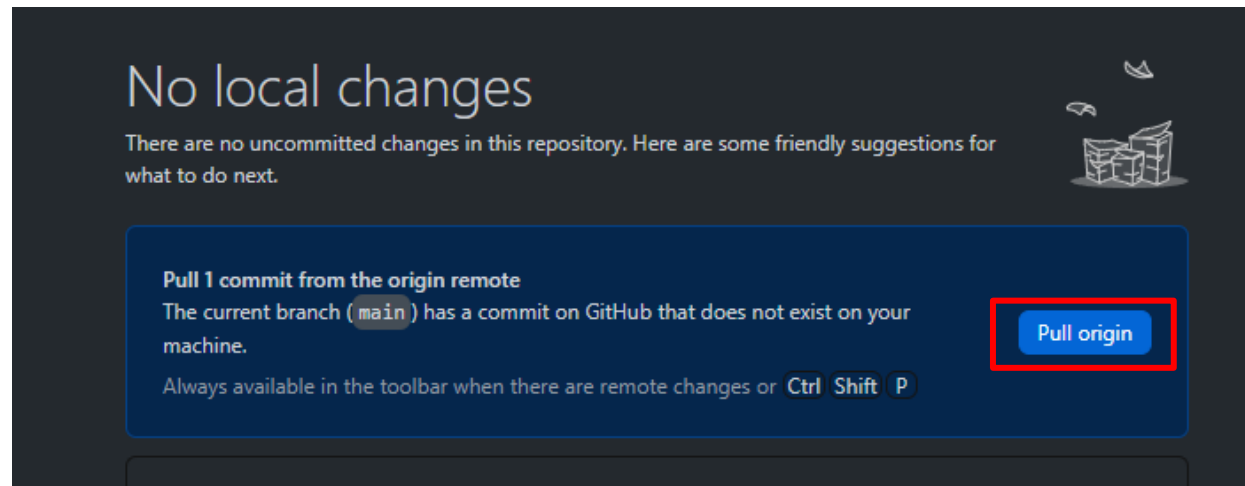


# GitHub on Website and GitHub Desktop App

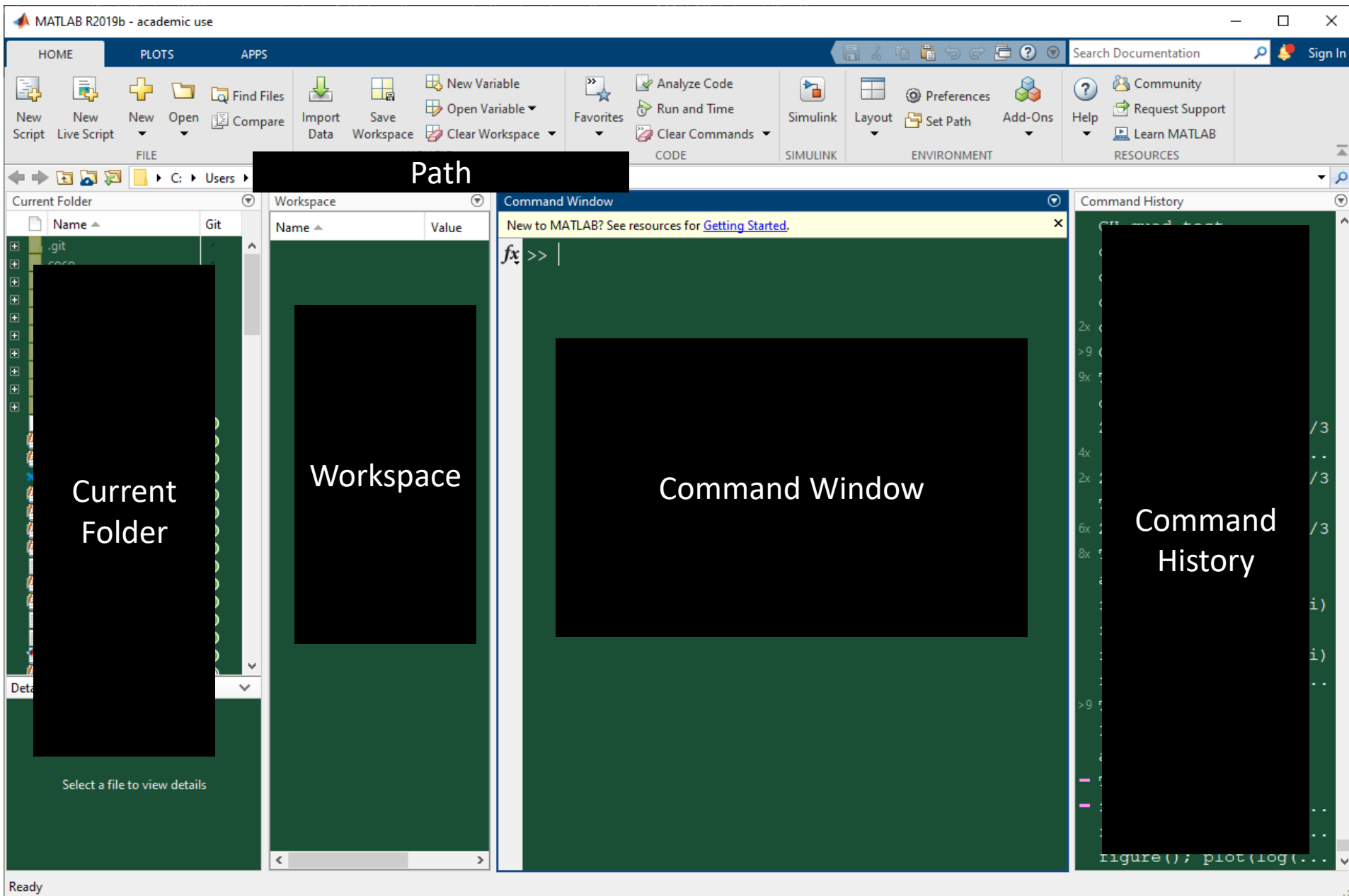
- Syncing in GitHub Desktop is very easy
- After you clone the repo, just click “Fetch origin”



- If the repo has an update, you will see an option to “pull”

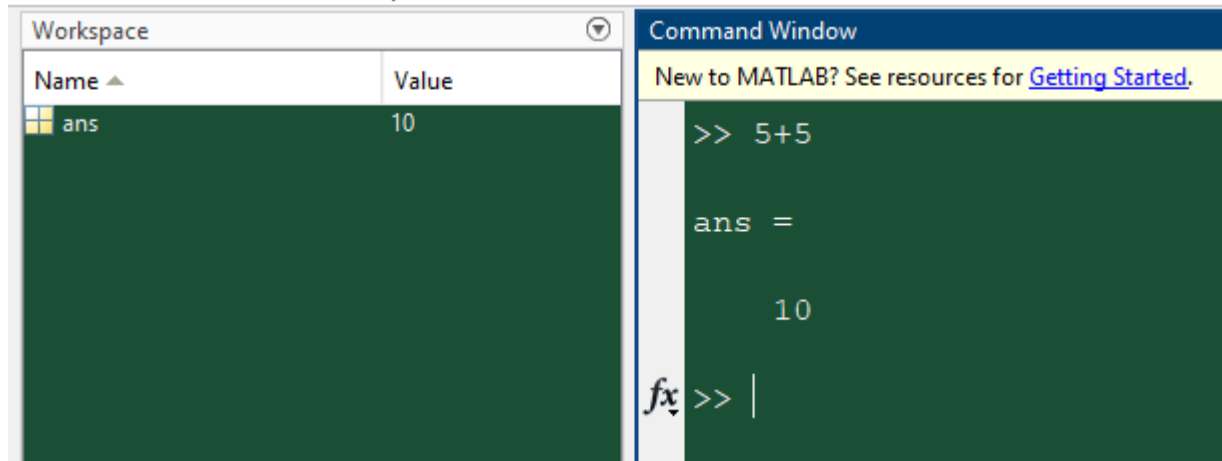


# MATLAB – Basics



# MATLAB – Basics

- MATLAB is like a super fancy calculator
- For instance, you can type an expression like “5+5” in Command Window



- Basic arithmetic operations: +, -, \*, / (same with C++)
- But “%” is NOT the integer operator – use “mod” function instead
- Use ^ for exponentiation (e.g.  $2^3 = 8$ ,  $3^2 = 9$ )

# MATLAB – Basic Arithmetic Operations

- Comparison between C++ and MATLAB arithmetic operations

Operation	C++	MATLAB
Addition, Subtraction, Multiplication	+, -, *	+, -, *
Floating-point Division $a \div b$	a/b	a/b
Integer Division $a \div b$	a/b	fix(a/b) or floor(a/b) <sup>1</sup>
Remainder after integer division $a \div b$	a%b	mod(a,b) or rem(a,b) <sup>2</sup>
Exponentiation $a^b$	std::pow(a,b) <sup>3</sup>	a^b

1. fix rounds the number towards 0, and floor rounds towards  $-\infty$
2. % is used to add *comments* in MATLAB
3. C++ does not have an exponent operator. You need to #include <cmath> for std::pow.



# MATLAB – Basic Matrix Manipulations

- In fact, MATLAB stands for “**MAT**rix **LAB**oratory”
- Very powerful software for numeric computing involving matrices
- Defining vectors/matrices is very easy – you don’t need any additional library, just type:

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

v =

     1     2     3

>> A = [ 11, 12; 21, 22]

A =

    11    12
    21    22
```

- Even scalars are considered matrices in MATLAB (1-by-1)
- [ ... ] defines a matrix
- Separators:
  - (white space) or , for row
  - ; for column

# MATLAB – Basic Matrix Manipulations

- Vector/Matrix Operations are similar, but the dimensions must agree!
  - + : addition
  - - : subtraction
  - \* : matrix multiplication (  $[m \times n] * [n \times p] = [m \times p]$  )
  - / : Don't use this for vector/matrix. Recall that in undergrad linear algebra, dividing by a matrix is not defined in general. (But in fact, it can be defined in *some* sense. Will be covered if necessary.)
- Entry-wise operation is also allowed – use “.” in front of the operator

# MATLAB – Basic Matrix Manipulations

- Entry-wise operation is also allowed – use “.” in front of the operator

```
>> v = [1, 4, 9]; w = [3, 3, 3];  
>> v+w  
  
ans =  
  
     4     7    12  
  
>> v*w  
Error using *  
Incorrect dimensions for matrix multiplication. Check that the number of columns in  
the first matrix matches the number of rows in the second matrix. To perform  
elementwise multiplication, use '.*'.  
  
>> v.*w  
  
ans =  
  
     3    12    27
```

# MATLAB – Basic Matrix Manipulations

- Entry-wise operation is also allowed – use “.” in front of the operator

```
>> v = [1, 4, 9]; w = [3, 3, 3];  
>> v+w  
  
ans =  
  
     4     7    12  
  
>> v*w  
Error using *  
Incorrect dimensions for matrix multiplication. Check that the number of columns in  
the first matrix matches the number of rows in the second matrix. To perform  
elementwise multiplication, use '.*'.  
  
>> v.*w  
  
ans =  
  
     3    12    27
```

Use ; to suppress output

Entry-wise multiplication

*You can also use .^ for entry-wise exponentiation:*

```
>> v.^(0.5)  
  
ans =  
  
     1     2     3
```

↑ *In this case, it computes the square root of each entry*

# MATLAB – Basic Matrix Manipulations

- (IMPORTANT) MATLAB uses **1-based** indexing;
  - Recall that C++ (and Python) uses 0-based indexing
- Entry access: use parenthesis ( ) (instead of square brackets [ ])

```
>> v = [10, 20, 30]

v =

    10    20    30

>> v(1)

ans =

    10

>> v(0)
Array indices must be positive integers or logical values.
```

# MATLAB – Basic Matrix Manipulations

- Entry access: use parenthesis ( ) (instead of square brackets [ ])
- For matrices, use  $(row\_idx, col\_idx)$  to access an entry
- Colon (:) *extracts* all the entries in that dimension

```
>> A(1,:)

ans =

    11    12    13

>> A(:,2)

ans =

    12
    22
```

The first row of A  
= [ A(1,1), A(1,2), A(1,3) ]

The second column of A  
= [ A(1,2) ; A(2,2) ]

```
>> A = [ 11, 12, 13; 21, 22, 23]

A =

    11    12    13
    21    22    23

>> A(1,2)

ans =

    12

>> A(2,3)

ans =

    23
```

# MATLAB – Basic Matrix Manipulations

- Special vectors/matrices
  - Arithmetic Sequences (produces a ROW VECTOR)
    - $a:b \rightarrow [a, a+1, a+2, \dots, b]$  if  $(b-a)$  is an integer.
    - $a:n:b \rightarrow [a, a+n, a+2*n, \dots, b]$  if  $(b-a)$  is a multiple of  $n$
- (i.e.,  $a:b$  is equivalent to  $a:1:b$ )
- In general,  $a:n:b = [a, a+n, \dots, a+kn]$  where  $k = \text{fix}((b-a)/n)$  (can be empty!)

Rounds a number towards 0

```
>> 1:10  
ans =  
     1     2     3     4     5     6     7     8     9    10  
  
>> 1:2:10  
ans =  
     1     3     5     7     9  
  
>> pi:10  
ans =  
  3.1416  4.1416  5.1416  6.1416  7.1416  8.1416  9.1416
```

# MATLAB – Basic Matrix Manipulations

- Special vectors/matrices
- Zero matrices, Identity matrices and matrices of ones

```
>> A = zeros(3)
```

```
A =
```

```
    0    0    0
    0    0    0
    0    0    0
```

```
>> A = zeros(2,3)
```

```
A =
```

```
    0    0    0
    0    0    0
```

```
>> ones(3)
```

```
ans =
```

```
    1    1    1
    1    1    1
    1    1    1
```

```
>> ones(2,3)
```

```
ans =
```

```
    1    1    1
    1    1    1
```

```
>> eye(3)
```

```
ans =
```

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

```
>> eye(2,3)
```

```
ans =
```

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



# MATLAB – Scripts and Functions

- You can write **scripts** and **functions** for more complex tasks

```
A = rand(3); % 3-by-3 square matrix of random entries
disp('A is ')
disp(A);      % displays A

x = [1;0;0]; % column vector

disp('A*x is ');
disp(A*x);    % displays A*x = (First column of A)

v = A(:);      % vectorize A
disp('A (vectorized) is');
disp(v);

n = size(v); % size of v (3*3 = 9)

for i = 2:n % for i running from 2 to n
    v(i) = v(i) + v(i-1); % what is the result?
end

disp('cumulative sum is ');
disp(v);

% Plot v
figure(); plot(v); % multiple commands in a single line is allowed
```

You can find this script  
“week1.m” on the GitHub repo

# MATLAB – Scripts and Functions

- Running “week1.m” in the command window:

1. Press “Run [▶]” button in the editor (shortcut: F5)

or,

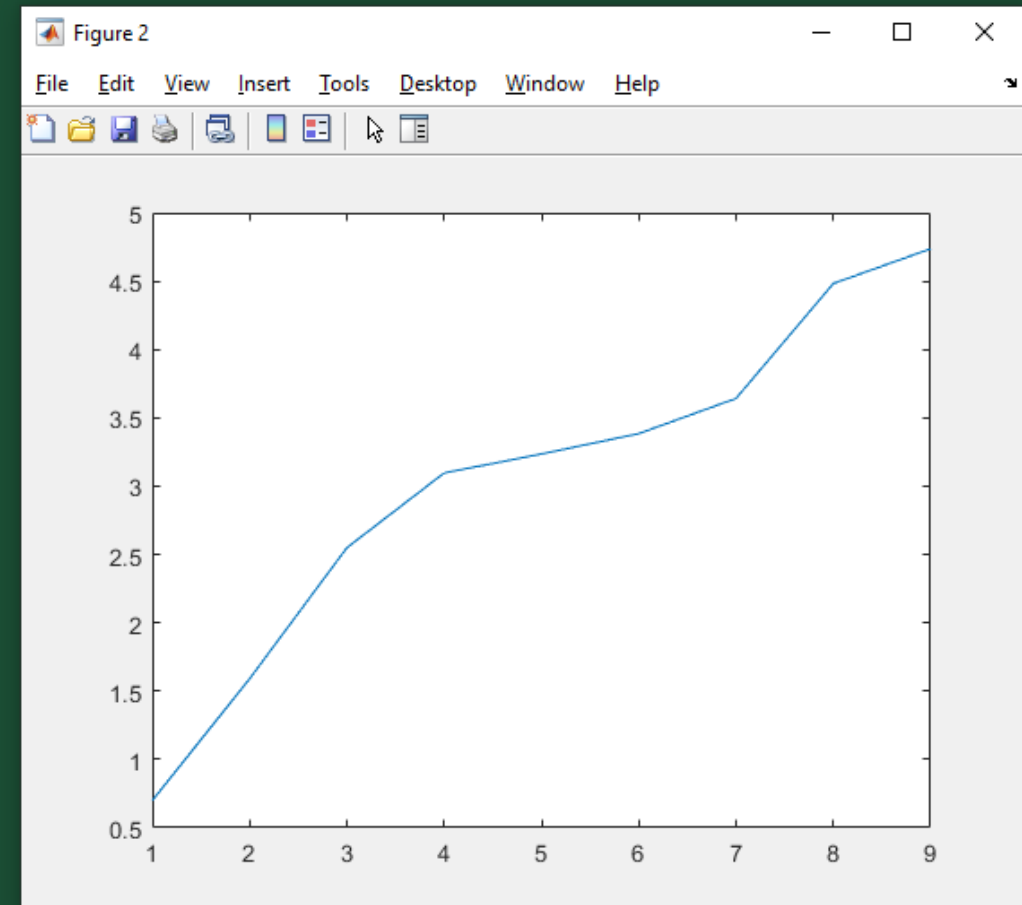
2. type “week1” on the command window

```
>> week1  
A is  
    0.6991    0.5472    0.2575  
    0.8909    0.1386    0.8407  
    0.9593    0.1493    0.2543
```

```
A*x is  
    0.6991  
    0.8909  
    0.9593
```

```
A (Vectorized) is  
    0.6991  
    0.8909  
    0.9593  
    0.5472  
    0.1386  
    0.1493  
    0.2575  
    0.8407  
    0.2543
```

```
cumulative sum is  
    0.6991  
    1.5900  
    2.5493  
    3.0965  
    3.2351  
    3.3844  
    3.6419  
    4.4826  
    4.7369
```



# MATLAB – Scripts and Functions

- You can write scripts and functions for more complex tasks
- Machine epsilon = upper bound on the relative error due to rounding

```
1  e = machine_eps(100); % run machine_eps function with max_iter = 100
2
3  function eps = machine_eps(max_iter)
4  eps = 1;
5  for i = 1:max_iter
6      if ( 1 + eps/2 == 1)
7          break;
8      else
9          eps = eps / 2;
10     end
11 end
12 disp(['machine eps = ', num2str(eps)]);
13 disp([ ' = 2^(-' , num2str(i-1), ' )']);
14 % or you can use "fprintf" if you prefer
15 % fprintf('machine eps = %.4e = 2^(-%d)\n', eps, i-1);
16 end
```

Simple script including the implementation of the function *machine\_eps*