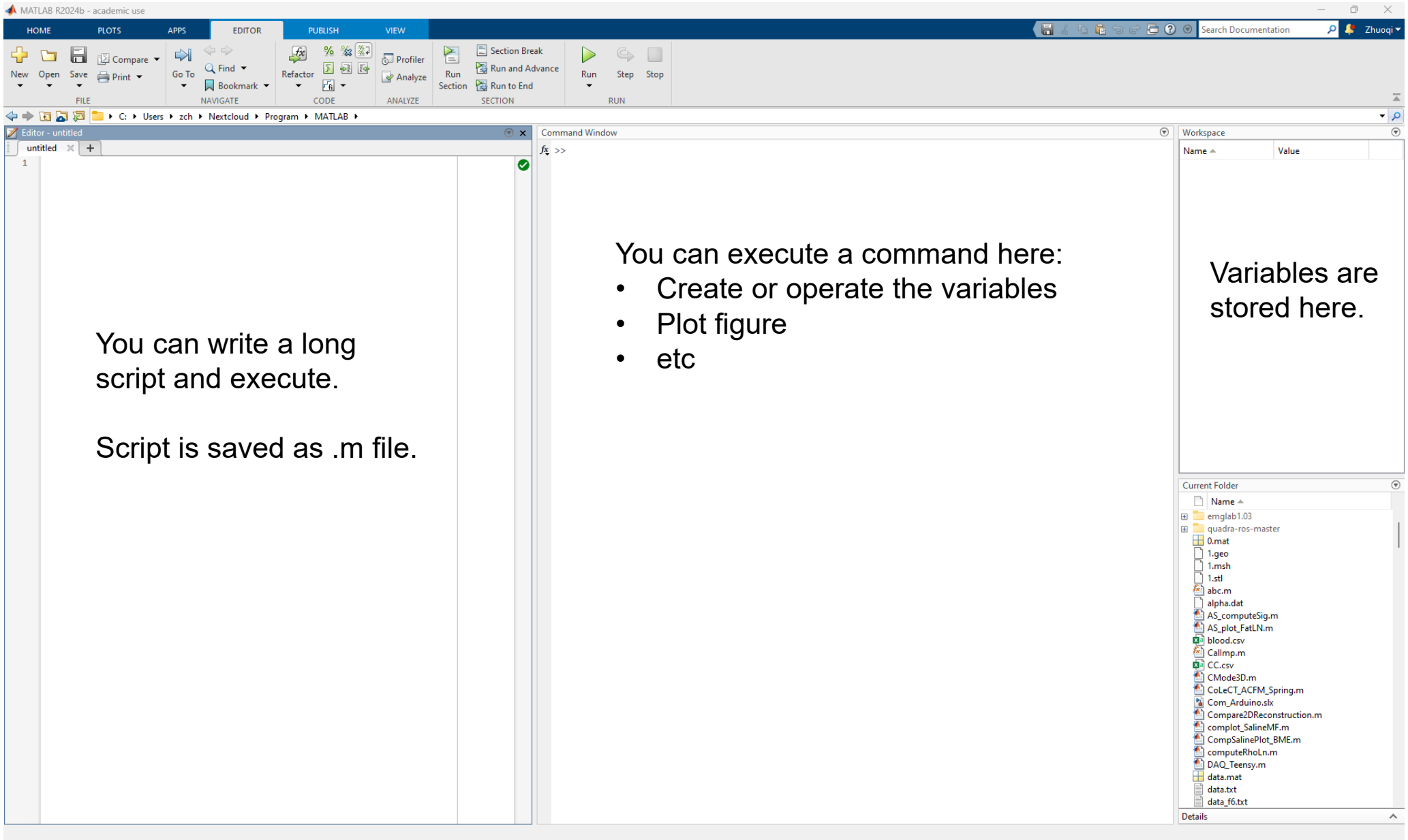
Matlab tutorial (Basics)

Zhuoqi **Cheng**
zch@mmmi.sdu.dk
SDU Robotics

Matlab



You can write a long script and execute.

Script is saved as .m file.

You can execute a command here:

- Create or operate the variables
- Plot figure
- etc

Variables are stored here.

SDU

Some tips and basic commands

→ Matlab is an interpreted language without compiling.
→ When you use it to run a script, it executes line by line. It will **NOT** tell you an error somewhere in your script (such as a typo) until it executes that line.
→ Matlab has many ready functions and toolboxes, by the company or by individual. When you download and install Matlab, remember to select the **Signal Processing** toolbox.
→ Variables in the workspace should be saved manually. Otherwise, they will be deleted after Matlab is close.

→ Good habit: place '**;**' at end of a statement to suppresses display of value (try to type 'a = 0' and 'a = 0;')

Basic commands

% used to denote a comment
... continues the statement on next line when a command is too long.

SDU

To find out how to use a function, use command '**help**'.
For example, **help disp**

Matlab’s Workspace

save – save workspace vars to *.mat file.
load – load variables from *.mat file.
clear all – clear workspace vars.
close all – close all figures
clc – clear screen
clf – clear figure

Numbers

By default, MATLAB stores all numeric values as double-precision floating point.

Mathematical functions: **sqrt(x)**, **exp(x)**, **cos(x)**, **sin(x)**, **tan(x)**, **sum(x)**, **log(x)**, etc.
Operations: **+**, **-**, ***, **/**
Power: **^**
Round toward negative infinity: **floor(x)**
Constants: **pi**, etc.

Generate random number: **rand(x)**

Absolute: **abs(x)**

Arrays and Matrices

v = [-2 3 0 4.5 -1.5]; % length 5 row vector.
v = v'; % transposes v.
v(1); % first element of v.
v(2:4); % entries 2-4 of v.
v([3,5]); % returns entries 3 & 5.
v=[4:-1:2]; % same as v=[4 3 2];

Example:
a=1:3; b=2:4;
c=[a b]; → c = [1 2 3 2 3 4];
c=[a; b]; → c = [1 2 3;
2 3 4];

Arrays and Matrices (2)

x = linspace(-pi,pi,10); % creates 10 linearly-spaced elements from –pi to pi.

Example:
A = [1 2 3; 4 5 6]; % creates 2x3 matrix
A(1,2) % the element in row 1, column 2.
A(:,2) % the second column.
A(2,:) % the second row.

Arrays and Matrices (3)

A+B, **A-B**, **2*A**, **A*B**	% matrix addition, matrix subtraction, scalar multiplication, matrix multiplication
A.*B	% element-by-element multiply.
A’	% transpose of A
det(A)	% determinant of A
cross(A, B)	% cross product

Creating special matrices

diag(v)	% change a vector v to a diagonal matrix.
diag(A)	% get diagonal of A.
eye(n)	% identity matrix of size n.
zeros(m,n)	% m-by-n zero matrix.
ones(m,n)	% m*n matrix with all ones.

Logical Conditions

== (equal)
~= (not equal)
<, **>**, **<=**, **>=** (greater or less)
~ (not)
& (element-wise logical and)
| (element-wise logical or)

Matrix/vector operations

length(v)	% determine length of vector.
size(A)	% determine size of matrix.
norm(A), norm(v)	% determine norm of matrix or vector.
fliplr(v)	% flip array left to right

For loops

```
x = 0;
for i=1:2:5       % start at 1, increment by 2
    x = x+i;        % end with 5.
end
```

This computes x = 0+1+3+5=9.

While loops

```
x=7;
while (x >= 0)
    x = x-2;
end;
```

This computes x = 7-2-2-2-2 = -1.

break – terminates execution of for and while loops. For nested loops, it exits the innermost loop only.

If statements

```
if (x == 3)
    disp('The value of x is 3.');
elseif (x == 5)
    disp('The value of x is 5.');
else
    disp('The value of x is not 3 or 5.');
end;
```

Switch statement

```
switch face
case {1}
      disp('Rolled a 1');
case {2}
      disp('Rolled a 2');
otherwise
      disp('Rolled a number >= 3');
end
```

NOTE: Unlike C, ONLY the SWITCH statement between the matching case and the next case, otherwise, or end are executed. *(So breaks are unnecessary.)*

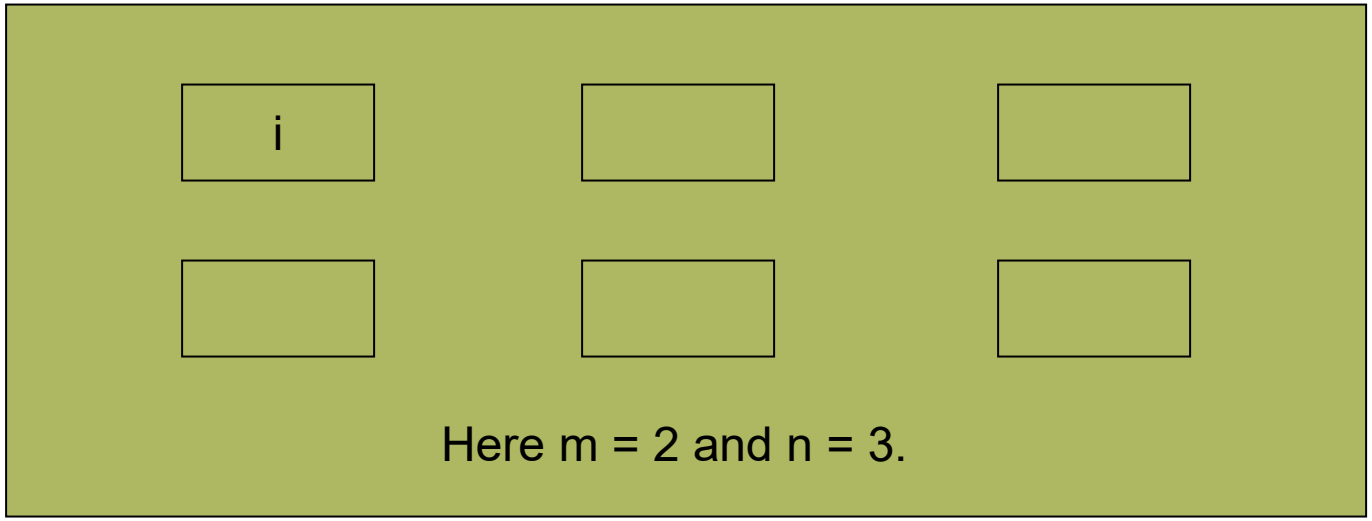
Graphics

```
x = linspace(-1,1,10);
y = sin(x);
plot(x,y);              % plots y vs. x.
plot(x,y,'k-');         % plots a black line of y vs. x.

hold on;                % put several plots in the same figure window.
figure(x);              % open new figure window.
```

Graphics (2)

subplot(m,n,i) % Makes an m*n array for plots. Will place plot in the i^{th} position.



Example:

```
% Generate and plot a sine wave
fs = 1000; % Sampling frequency
t = 0:1/fs:1; % Time vector
f = 5; % Frequency (Hz)
x1 = sin(2*pi*f*t);
x2 = cos(2*pi*f*t);

subplot(2,1,1)
plot(t, x1);
xlabel('Time (s)');
ylabel('Amplitude');
title('Sine Wave');

subplot(2,1,2)
plot(t, x2,'k');
hold on; % Try to remove this line
plot([0,1], [0, 0], 'r--')
xlabel('Time (s)');
ylabel('Amplitude');
title('Cosine Wave');
```

Graphics (3)

plot3(x,y,z) % plot 2D function.
mesh(x,y_ax,z_mat) – surface plot.

axis([xmin xmax ymin ymax]) – change axes
title('My title'); - add title to figure;
xlabel, ylabel – label axes.
legend – add key to figure.

Exercise

→ Create a vector v which = [2,4,6,8,10] (try to use a ‘for’ loop)
→ Create a matrix m which = [0, 1, 2, 3, 4; 5, 6, 7, 8, 9; 0, 0.1, 0.2, 0.3, 0.4; 0.5, 0.6, 0.7, 0.8, 0.9]
→ Define tv which is the transpose of v
→ Calculate dot multiply res which m*tv
→ Plot res using * dots and in red color

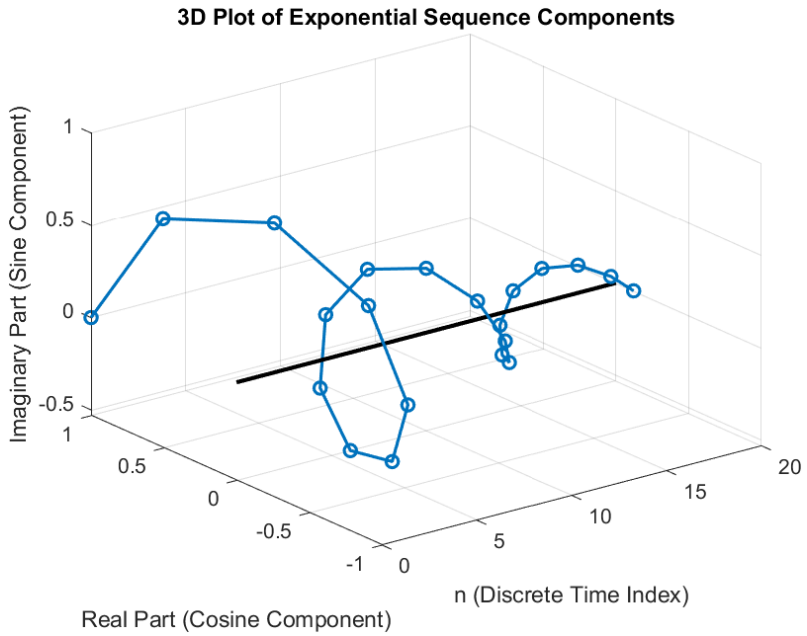
Exercise

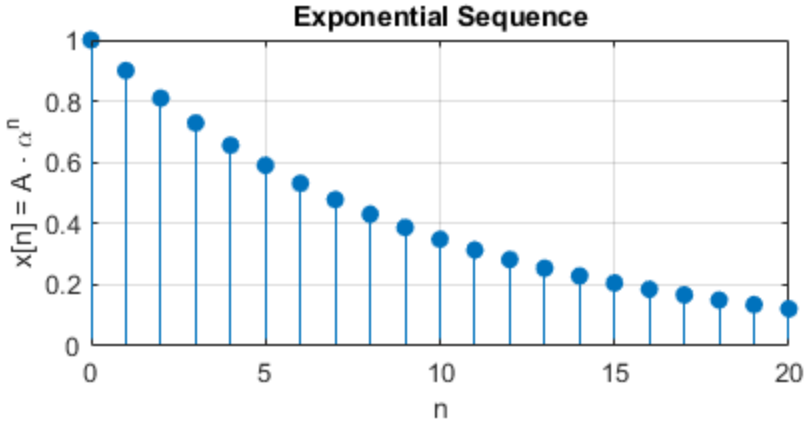
Create and plot an exponential sequence
$x(n) = 0.9^n,\ n \geq 0$

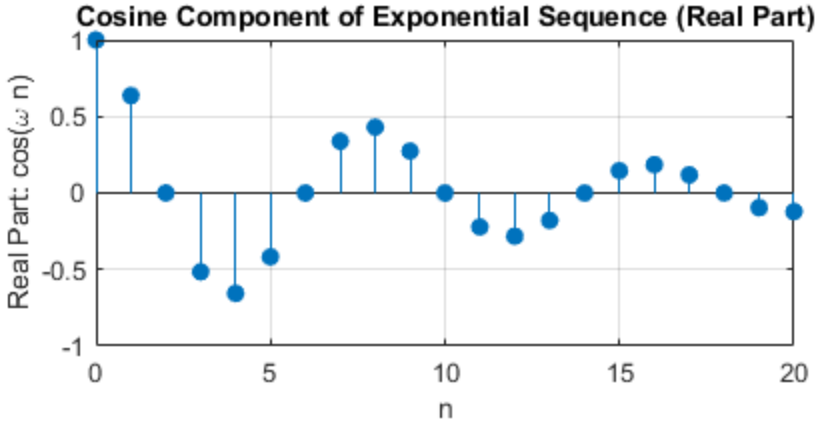
Decompose the sequence to a complex exponential
$x(n) = e^{j\omega_0 n} = \cos(\omega_0 n) + j\,\sin(\omega_0 n)$

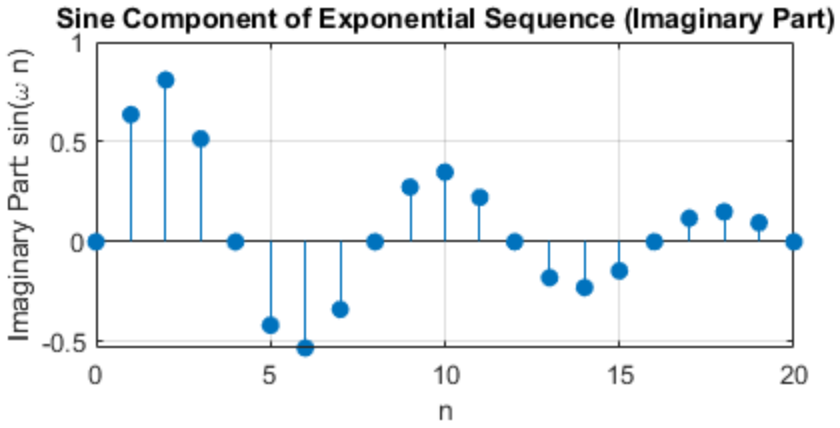
Plot the real part (cosine)
And the imagine part (sine)

3D plot of the complex exponential









SDU