EEE336 Signal Processing and Digital Filtering

Lecture 2 Fundamentals of Mathematics and Matlab

2_1 Mathematics Review 1

Zhao Wang

Zhao.wang@xjtlu.edu.cn

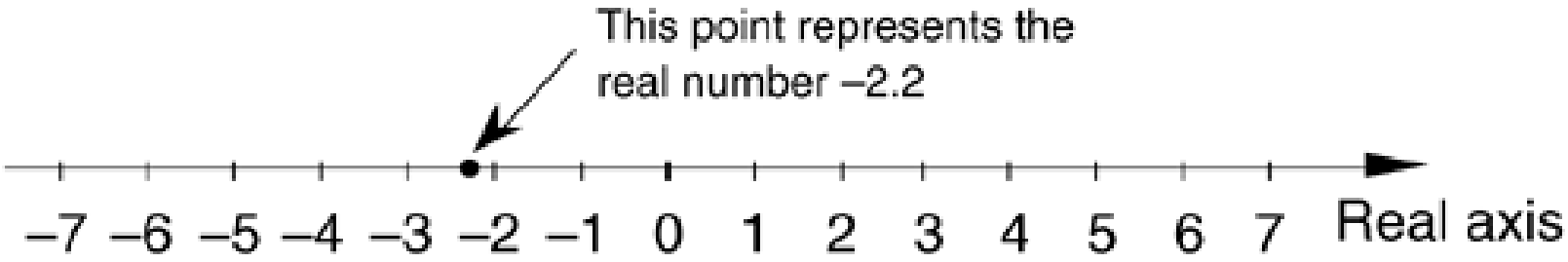
Room EE322

Complex Numbers

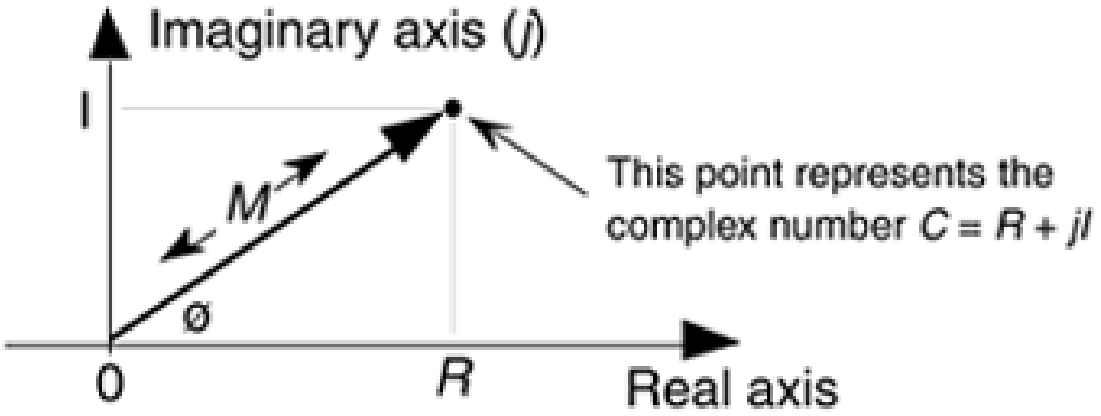
- Real VS. Complex
- Representation of complex numbers
- Euler's formula
- Operations / calculations of complex numbers
  - Addition and subtraction
  - Multiplication and division
  - Conjugate
  - Raising to power and taking roots
  - Logarithms

Graphical representation

- Real numbers: all real numbers correspond to all of the points on the real axis line.



- Complex numbers: a complex number can be treated as a point on a complex plane



Arithmetic representation

- 1. Rectangular form:

$$C = R + jI$$

- 2. Exponential form:

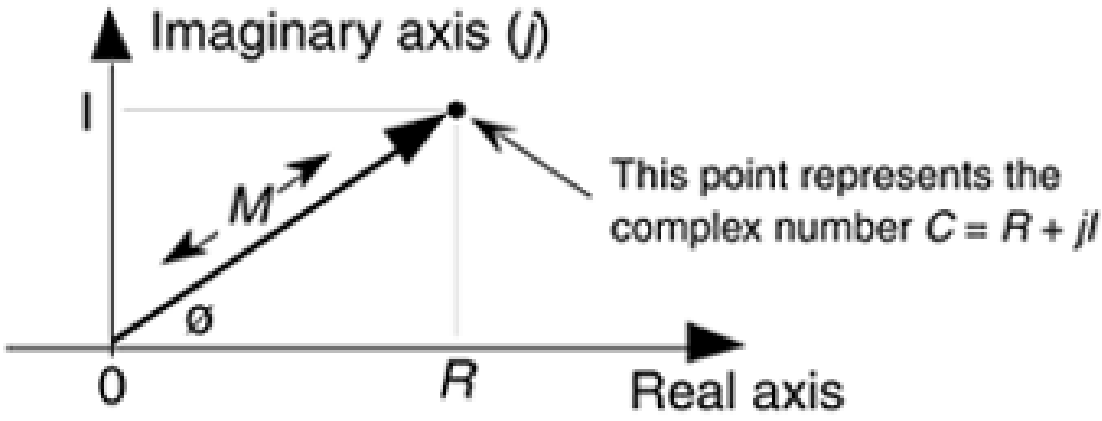
$$C = Me^{j\theta}$$

- 3. Polar form:

$$C = M\angle\theta$$

- 4. Trigonometric form:

$$C = M\cos\theta + jM\sin\theta$$



Imaginary unit 'i' or 'j':

$$i = j = \sqrt{-1}$$

Magnitude M

$$M = |C| = \sqrt{R^2 + I^2}$$

Phase θ

$$\theta = \arctan(I/R)$$

Euler's Formula

- Euler's formula:

$$\boldsymbol{e^{j\theta} = cos\theta + jsin\theta}$$

- Euler's identity:

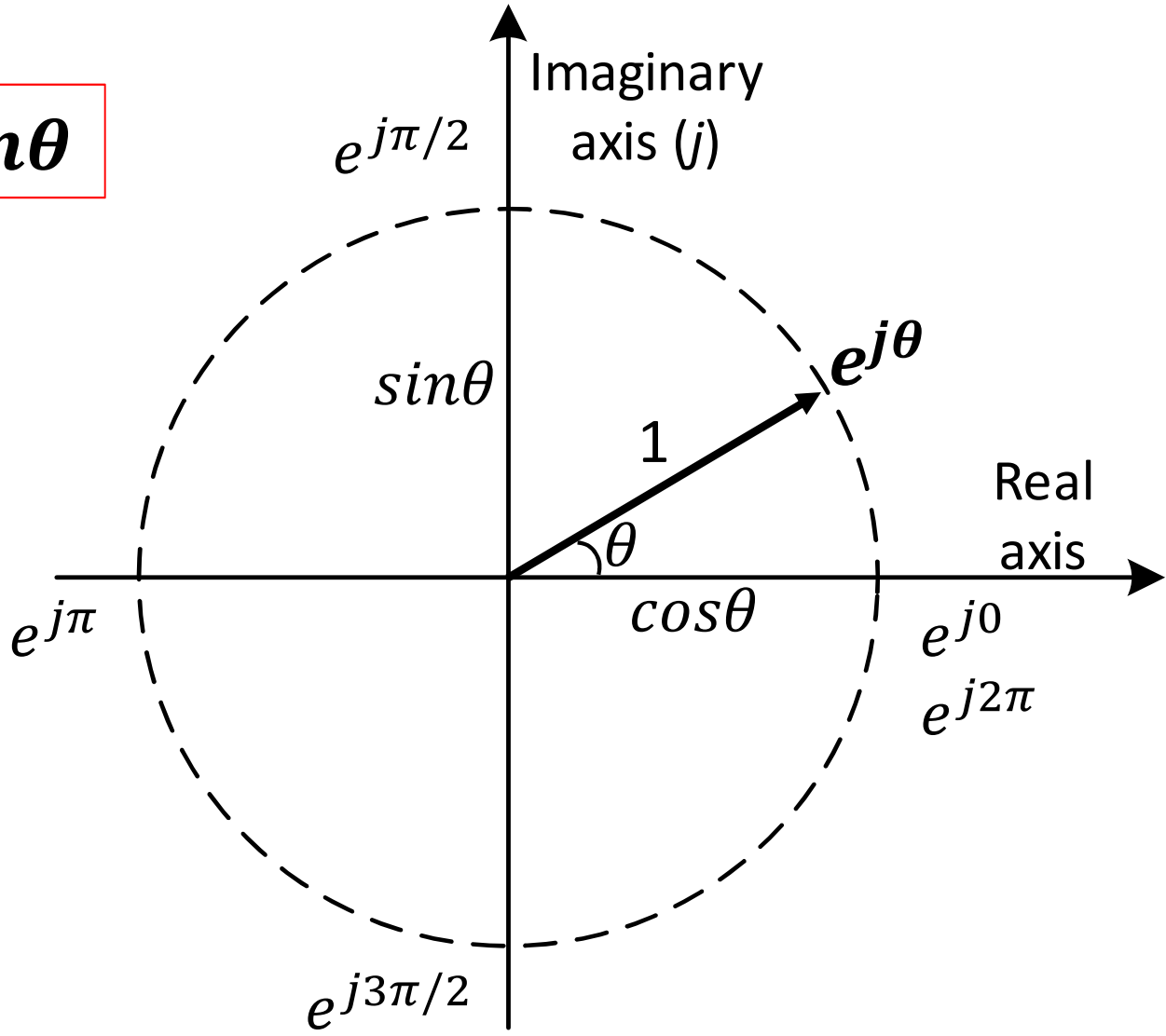
$$\boldsymbol{e^{j\pi} + 1 = 0}$$

- A complex number C can be represented by

$$C = Me^{j\theta} = Me^{j(\theta+2n\pi)}$$

- If the angle $\theta = \omega t$, then we have:

$$C = Me^{j\theta} = Me^{j\omega t}$$

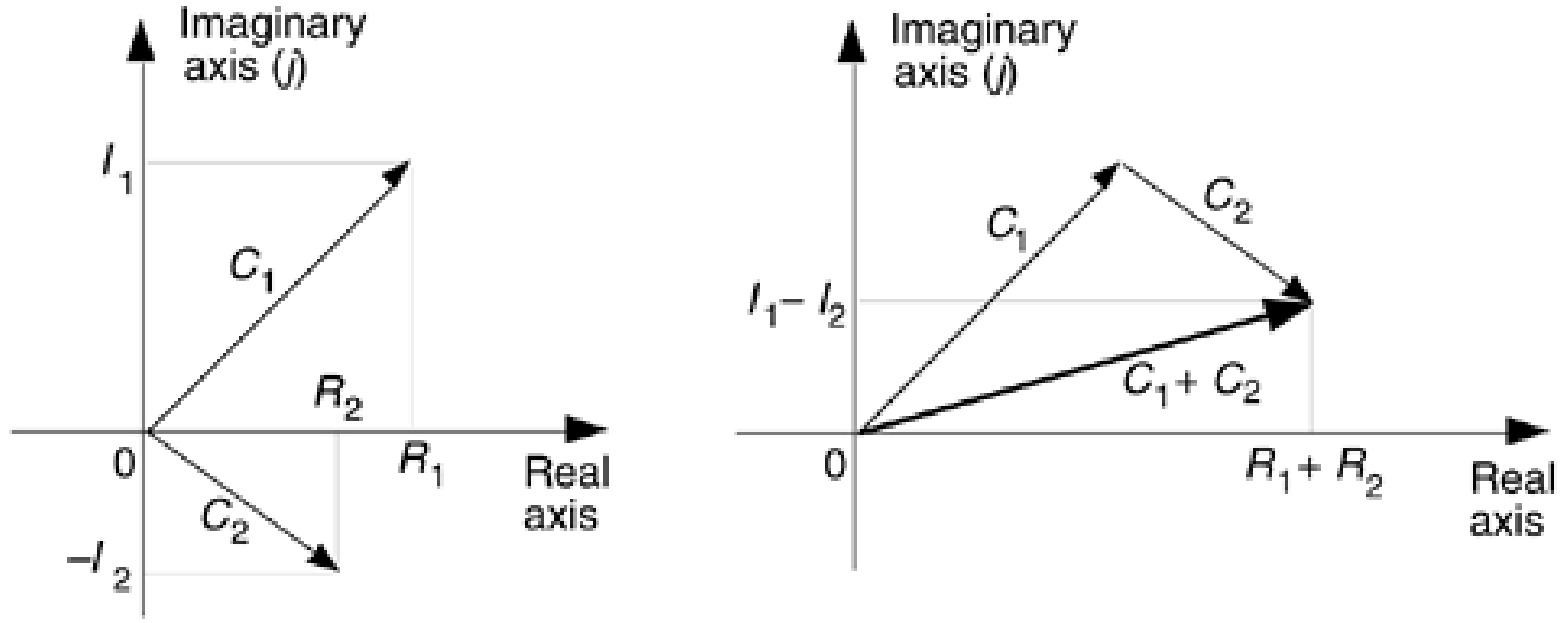


Arithmetic operations

Addition and Subtraction

- Addition (using rectangular form):

$$C_1 + C_2 = (R_1 + jI_1) + (R_2 + jI_2) = (R_1 + R_2) + j(I_1 + I_2)$$



- Subtraction (using rectangular form):

$$C_1 - C_2 = (R_1 - jI_1) + (R_2 - jI_2) = (R_1 - R_2) + j(I_1 - I_2)$$

Arithmetic operations

Multiplication

- Multiply two complex numbers (using rectangular form)

$$C_1C_2 = (R_1 + jI_1)(R_2 + jI_2) = (R_1R_2 - I_1I_2) + j(R_1I_2 + I_1R_2)$$

- Multiply two complex numbers (using exponential form)

$$C_1C_2 = M_1e^{j\theta_1}M_2e^{j\theta_2} = M_1M_2e^{j(\theta_1+\theta_2)}$$

- Scaling (using rectangular form)

$$KC = K(R + jI) = KR + jKI$$

- Scaling (using exponential form)

$$KC = KMe^{j\theta}$$

Arithmetic operations

- Conjugation (in rectangular and exponential forms)

  $C^* = R - jI = Me^{-j\theta}$

- Characteristics of conjugate:
  - If $C = C_1C_2$, then its conjugate C^* is:

    $C^* = (C_1C_2)^* = M_1M_2e^{-j(\theta_1+\theta_2)} = M_1e^{-j\theta_1}M_2e^{-j\theta_2} = C_1^*C_2^*$
  - The product CC^* is:

    $CC^* = Me^{j\theta}Me^{-j\theta} = M^2e^{-j0} = M^2$

Arithmetic operations

Division and inverse

- Division of two complex numbers (in exponential form)

$$\frac{C_1}{C_2} = \frac{M_1 e^{j\theta_1}}{M_2 e^{j\theta_2}} = \frac{M_1}{M_2} e^{j(\theta_1 - \theta_2)} = \frac{M_1}{M_2} \angle \theta_1 - \theta_2$$

- Division of two complex numbers (in rectangular form)

$$\frac{C_1}{C_2} = \frac{R_1 + jI_1}{R_2 + jI_2} = \frac{R_1 + jI_1}{R_2 + jI_2} \cdot \frac{R_1 - jI_1}{R_2 - jI_2} = \frac{(R_1 R_2 + I_1 I_2) + j(R_2 I_1 - R_1 I_2)}{R_2^2 + I_2^2}$$

- Inverse of a complex number (in exponential form)

$$\frac{1}{C_2} = \frac{1}{M_2 e^{j\theta_2}} = \frac{1}{M_2} e^{-j\theta_2} = \frac{1}{M_2} \angle - \theta_2$$

- Inverse of a complex number (in rectangular form)

$$\frac{1}{C_2} = \frac{1}{R_2 + jI_2} = \frac{R_1 - jI_1}{R_2^2 + I_2^2}$$

Arithmetic operations

- The k^{th} power of a complex number $C = Me^{j\theta}$

  $C^k = (Me^{j\theta})^k = M^k e^{jk\theta}$

- The k^{th} root of a complex number $C = Me^{j\theta}$
  - Since $C = Me^{j\theta} = Me^{j(\theta+2n\pi)} = Me^{j(\theta+n360^\circ)}$
  - Its roots are:

    $$\sqrt[k]{C} = \sqrt[k]{Me^{j(\theta+n360^\circ)}} = \sqrt[k]{M}e^{j\frac{\theta+n360^\circ}{k}}$$

  - The value of n can be 0, 1, 2, 3, …, k-1.

2_1 Wrap up

- In this section, complex number and the Euler's formula are reviewed;
- Several basic operations of complex numbers are reviewed;
- They are very useful in this module.

EEE336 Signal Processing and Digital Filtering

Lecture 2 Fundamentals of Mathematics and Matlab

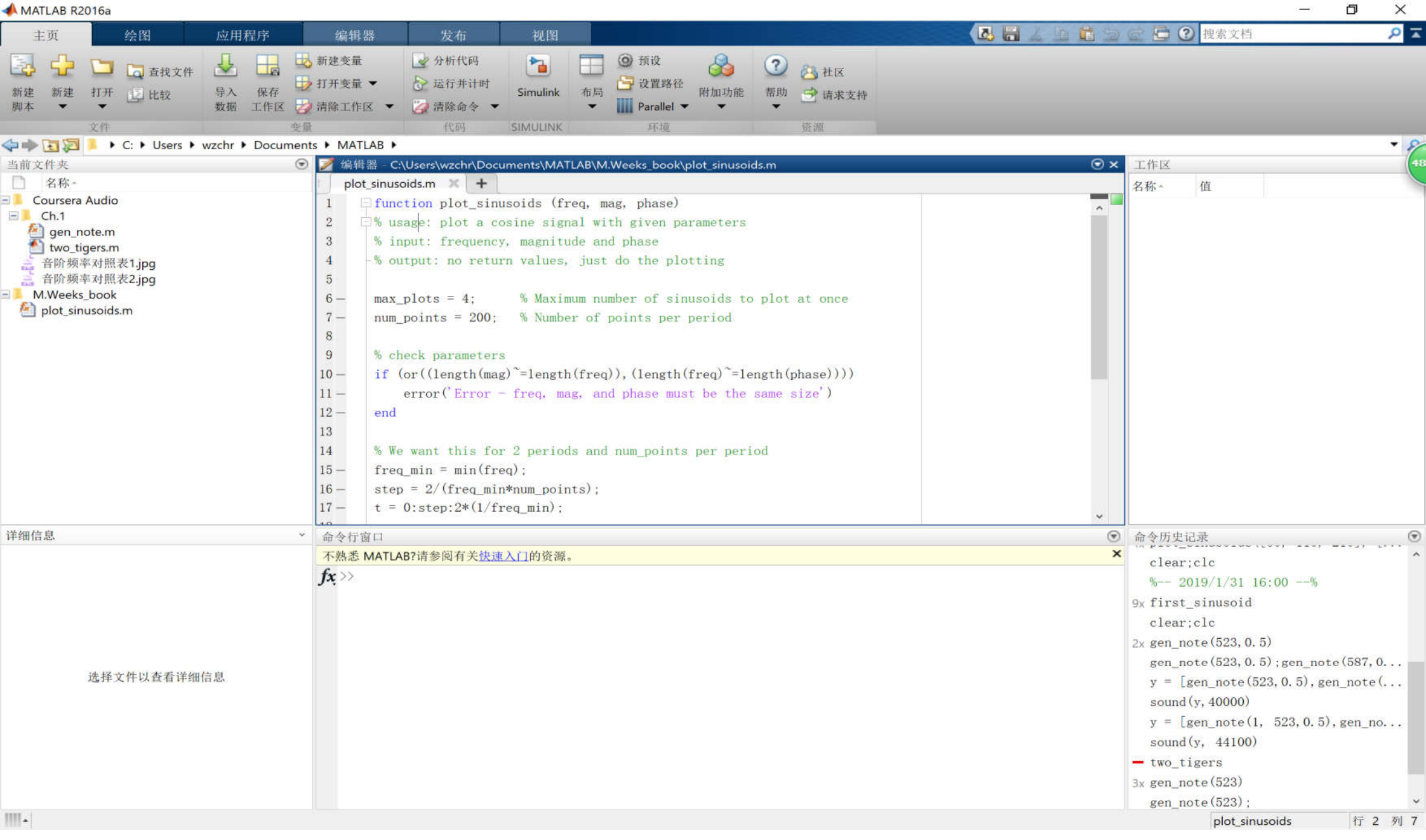
2_2 Matlab Review 1

Zhao Wang

Zhao.wang@xjtlu.edu.cn

Room EE322

Programming environment



Variables

- A variable can be used without declaration;
- A variable can store integer, floating-points, or even complex;
- A semicolon “;” is often put at the end of a command;
- In Matlab, both “i” and “j” represent the complex unit (the square root of -1);
- Name your variables with enough information
  - Usually, uppercase for matrices and lowercase for scalars;
  - “`CamelCase`” or “`complex_var_naming`” are preferred.

Scalars, vectors and matrices

- Create:
  - a single value: scalar;
  - several values at one: vector;
    - Row: `A = [1 2 3 4]` or `B = [1, 2, 3, 4]`
    - Column: `C = [1; 2; 3; 4]` or `D = [1 2 3 4]'`
  - Or even a matrix;
    - Eg: `E = [1 2; 3 4; 5 6]`
- To access one of the values, use index starting from 1
  - For vectors: `A(1)` `D(4)` `D(end)`
  - For matrix: `E(2,3)` `E(:,1)` `E(1,:)`
- To manipulate the matrices
  - To add a row or column: `E(:,3) = [9; 9; 9]` `E(4,:) = [8, 8]`
  - To remove a row or column: `E(:,3) = []` `E(2,:) = []`

Conditional statement

- Syntax:

```
if expression
        statements;
elseif expression
        statements;
else
        statements;
end
```

- expression:
  - scalars => use "**==**";
  - arrays or matrices => use "**isequal**";
  - strings => use "**strcmp**".

Loops

- With loop control statements, you can repeatedly execute a block of code.
  - `for` statements loop a specific number of times, and keep track of each iteration with an incrementing index variable.
  - `while` statements loop as long as a condition remains true.

- Tips
  - It is a good idea to indent the loops for readability, especially when they are nested (that is, when one loop contains another loop);
  - You can programmatically exit a loop using a `break` statement, or skip to the next iteration of a loop using a `continue` statement.

Input and output

- Input (from keyboard)
  - **x = input(prompt)** displays the text in prompt and waits for the user to input a value and press the return key;
  - **str = input(prompt,'s')** returns the entered text as a string;

- Output (to screen)
  - **disp(X)** displays the value of variable X without printing the variable name;
  - **sprintf(formatSpec,A1,...,An)** formats data intro string (and displays on screen);
  - **fprintf(formatSpec,A1,...,An)** directly display on screen without creating a string;

Plotting

Line Plots	Pie Charts, Bar Plots, and Histograms	Discrete Data Plots	Polar Plots	Contour Plots	Vector Fields	Surface and Mesh Plots		Polygons	Animation
plot	area	stairs	polar	contour	quiver	surf	mesh	fill	animatedline
plot3	pie	stem	rose	contourf	quiver3	surfc	meshc	fill3	comet
loglog	pie3	stem3	compass	contour3	feather	surfl	meshz	patch	comet3
semilogx	bar	scatter	ezpolar	contourslice	streamslice	ezsurf	waterfall		
semilogy	barh	scatter3		ezcontour	streamline	ezsurfc	ezmesh		
errorbar	bar3	spy		ezcontourf	streamribbon	ribbon	ezmeshc		
ezplot	bar3h	plotmatrix			streamtube	pcolor			
ezplot3	histogram				coneplot				

Functions

- Functions in Matlab are stored as separate .m files which have the same names as the functions.

```
function plot_sinusoid ( freq, mag, phase )
% stored as "plot_sinusoid.m"
```

- Syntax:

```
function [y1,...,yN] = myfun(x1,...,xM)
```

  – Declares the function name, inputs, and outputs

```
function y = average(x)
if ~isvector(x)
   error('Input must be a vector')
end
y = sum(x)/length(x);
end
```

```
>>z = 1:99;
>>average(z)
```

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

```
>>values = [12.7, 45.4, 98.9, 26.6, 53.1];
>>[ave,stdev] = stat(values)
```

More useful Matlab functions

Name	usage
ones	Create array of all ones
zeros	Create array of all zeros
linspace	Generate linearly spaced vector
rand	Uniformly distributed random numbers
randn	Normaly distributed random numbers
size	Array dimensions
length	Length of largest array dimension
reshape	Reshape array
sort	Sort array elements
circshift	Shift array circularly
ctranspose	Complex conjugate transpose

2_2 Wrap_up

- Matlab IDE
- Fundamentals
  - Variables
  - Scalars, vectors and matrices
    - Access, manipulate, information
  - Conditional statements (**if**, **else**)
  - Loops (**for**, **while**)
  - Input and output (**input**, **disp**, **sprintf**)
  - Plotting (**plot**, **stem**)
  - Functions

EEE336 Signal Processing and Digital Filtering

Lecture 2 Fundamentals of Mathematics and Matlab

2_3 Matlab Review 2 -- Matlab & Audio

Zhao Wang

Zhao.wang@xjtlu.edu.cn

Room EE322

Read and Write Audio Files

- Reading:
  - From saved Matlab workspace

    ```
    load handel.mat
    ```

  - From saved audio file

    ```
    [y, Fs] = audioread ('handel.wav' );
    ```

- Writing:
  - To audio file

    ```
    audiowrite('handel.wav',y,Fs)
    ```

  - To save a workspace

    ```
    save ('newHandel.mat', 'y', 'Fs')
    ```

Audio file's information

- Get Information About Audio File
  - **>> info = audioinfo('handel.wav')**
  - **info** contains the information about audio file, returned as a *structure*. info can contain the fields as shown in the example:

```
info =
Filename: 'C:\...\handel.wav'
CompressionMethod: 'Uncompressed'
NumChannels: 1
SampleRate: 8192
TotalSamples: 73113
Duration: 8.9249
Title: []
Comment: []
Artist: []
BitsPerSample: 16
```

Play an audio file

- Play by external audio player
- Play in Matlab by **sound()**

```
>> y1 = audioread('handel.wav');
>> [y1, Fs] = audioread('handel.wav');
>> sound(y1)
>> sound(y1, Fs)

>> [y2, Fs] = audioread('piano.wav');
>> sound(y2)
>> sound(y2, Fs)
```

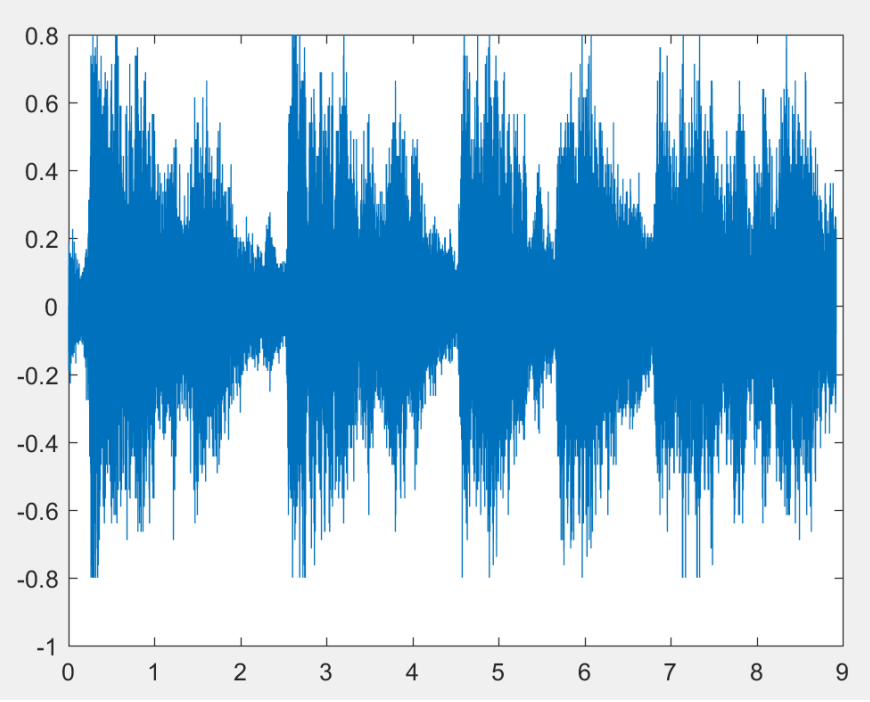
Plot Audio Data

- Create a vector **t** the same length as **y**, that represents elapsed time.

```
>> [y, Fs] = audioread ('handel.wav')
>> info = audioinfo('handel.wav')
>> t = 0:1/Fs:info.Duration;
>> t = t(1:end-1);
```

- Plot the audio data as a function of time

```
>> plot(t, y)
```



An example

- Generate a sinusoidal signal
  - With given frequency, amplitude and phase;
    - Freq = 523 Hz; A = 1; phase = 0;
  - Lasting for certain duration with certain time-spacing;
    - L = 1 s; time-spacing = 10e-4 s;
  - Plot the sinusoidal signal;
    - Using the function ‘plot’;
  - Play the sound of the signal;
    - Using the function ‘sound’;
  - Use this function to play a simple song;
    - Call the function with different frequency to play “do, re, mi”

2_3 Wrap_up

- Matlab is a powerful language for the dealing with digital signals, including audio signals;
- Several frequently used functions are briefly introduced; be familiar with them cause we will need them later in many examples and practices;
- Complete the Programming Exercise 1.

Some useful tips

- If something does not work, check the dimensions of the data. It may work with the transpose;
- If a plot does not show up, the data could be an empty matrix;
- If something calls for the discrete Fourier transform (DFT), try using the fast Fourier transform (FFT);
- Your programs may be slow, especially when you see “busy” in the status bar, so be patient;
- Try to avoid the usage of loop, that will reduce the speed of your programs. Think about using arrays / matrices instead;
- Use the graphical user interface to highlight a section of code, and comment it out (or uncomment it later);
- When debugging, use the breakpoint command to stop a program, and check values.