

MATLAB PROGRAMMING

AIST2010 Lecture 3P

MATLAB — MATRIX LABORATORY

MATLAB allows easy analysis and visualization of huge numerical data

- CUHK has campus-wide licence since this year

- Including MATLAB Online for use in browser

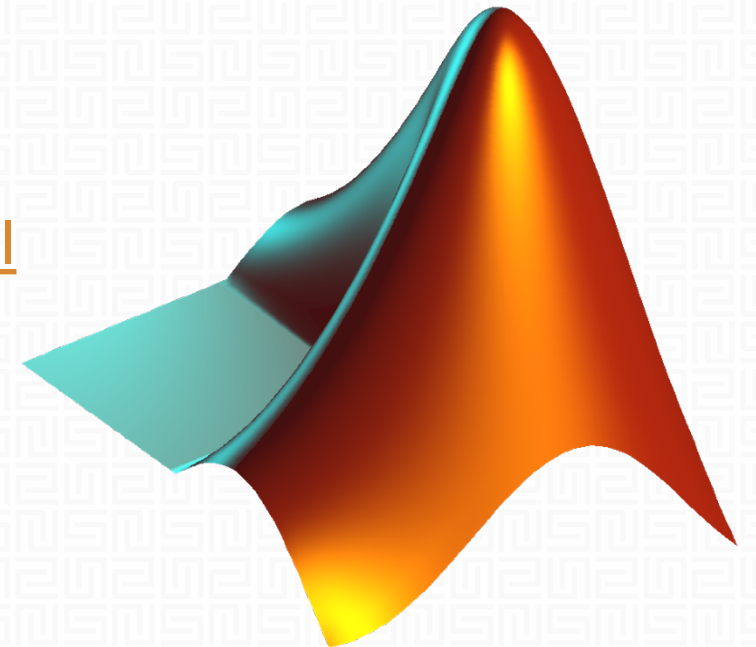
<https://matlab.mathworks.com>

- 5GB storage on MATLAB Drive

<https://www.mathworks.com/products/matlab-drive.html>

MATLAB files

- *MATLAB scripts *.m*: a list of commands, or functions
- *LiveScript *.mlx*: scripts + published text + graphs
- *Data file *.mat*: variables in workspace



BASIC MATLAB OPERATIONS

The .m script file is an ordinary text file of programme code

- Syntax is like many common languages: C, python, ...
- If in path, typing the .m file name will execute it
- Semicolon at the end of a statement suppresses the output
- Comments start with %

You can execute commands by typing at the prompt:

Get started here:

- <https://www.mathworks.com/help/matlab/getting-started-with-matlab.html>

```
Command Window
>> 2+2

ans =

    4

>> sin(pi/2)

ans =

    1

fx >>
```

ARRAYS, VECTORS AND MATRICES

MATLAB handles variables best as arrays

- Scalar, row vector, column vector, matrix
- MATLAB has dynamic typing
- Case sensitive and usual identifier rules
- Array can be assignment targets
 - `[M I] = max(y) % M = 6; I = 1`

Finding the number of elements

- Vectors: `length(y)`
- Matrices: `size(x, dim)`

Transpose of vectors/matrices

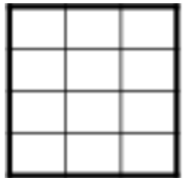
- Changing from row to column: $x^T = x'$

Image from: <https://matlabacademy.mathworks.com>

Matrix 

Values are stored in m rows and n columns.
(m -by- n)

$x = [1 \ 2 \ 3; 4 \ 5 \ 6]$



4-by-3

Row vector

Values are stored in 1 row and n columns.
(1-by- n)

$y = [6 \ 5 \ 4]$



1-by-3

Column vector

Values are stored in n rows and 1 column.
(n -by-1)

$z = [1; 2; 3]$



4-by-1

Scalar

A single value is stored.
(1-by-1)



1-by-1

ARRAY CREATION

Generating matrices with constant values

- `a = zeros(3); % a = [0 0 0; 0 0 0; 0 0 0];`
- `b = ones(2, 4); % b = [1 1 1 1; 1 1 1 1];`
- `c = rand(3, 4); % random numbers in (0,1) of 3x4 matrix`

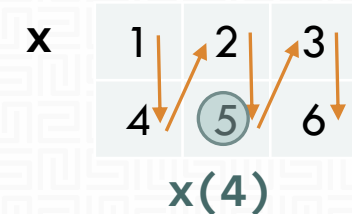
Generating a range of numbers

- `d = 5:2:15; % interval of 2: d = [5 7 9 11 13 15];`
- `e = linspace(10, 20, 4);`
% 4 evenly distributed elements:
% `e = [10.0000 13.3333 16.6667 20.0000];`

ARRAY ELEMENTS

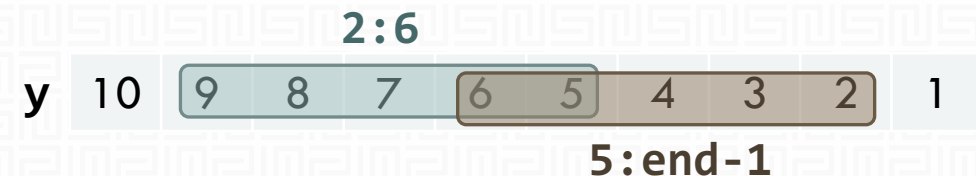
Indexing in array or matrix

- `x = [1 2 3; 4 5 6]; x(4)` % numbered down columns: output is 5
- `x(1,:)` % whole row: output is [1 2 3]



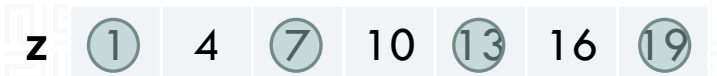
Slicing: select part of matrix using colon

- `y = 10:1; y(2:6)` % output is [9 8 7 6 5]
- `y(5:end-1)` % output is [6 5 4 3 2]



Logical indexing: select elements satisfying condition

- `z = 1:3:20; z(mod(z,2)~=0)` % output is [1 7 13 19] non-even number



CONDITIONS AND LOOPS

if-elseif-else

```
▪ if x > 0
    disp('positive');
elseif x < 0
    disp('negative');
    x = abs(x);
else
    disp('zero!');
end
```

AND/OR/NOT

```
▪ && || ~
```

for loop

```
▪ for i = 1:10
    disp(i*2);
end
```

while loop

```
▪ while x > 1
    x = x-1
    % no x--, no x-=1
end
```

Change of flow

```
▪ break/continue are available
```

PLOT

Basic plot: two axes must share the same number of elements

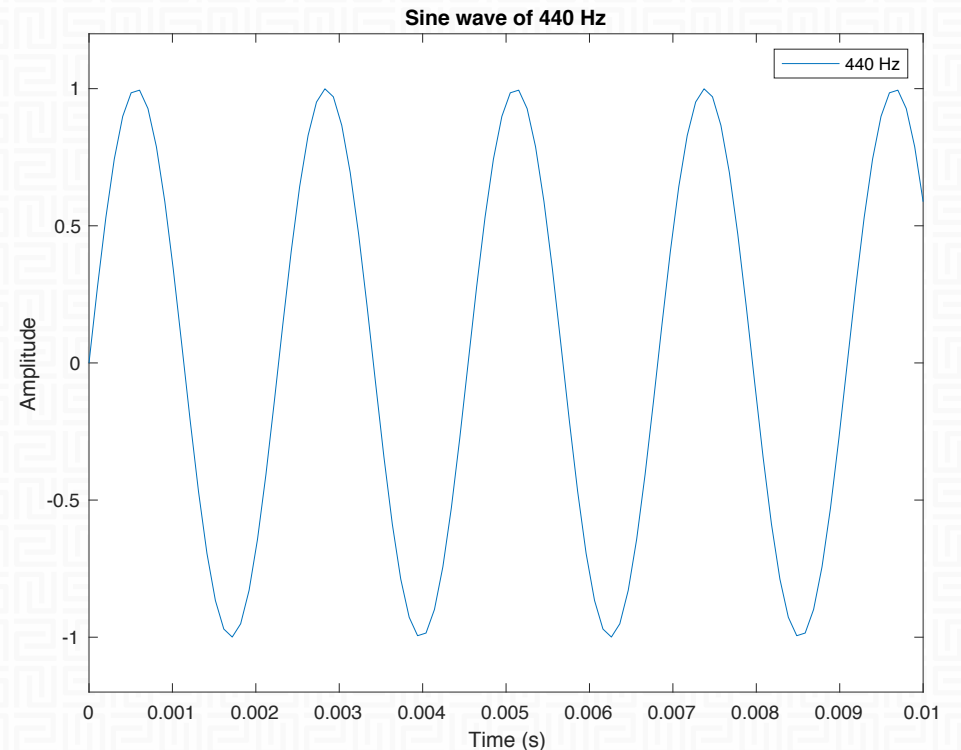
```
▪ t = linspace(0, 1, 44100); % 44100 elements from 0 to 1  
  % x as function of t, 44100 elements  
  x = sin(2*pi*440*t);  
  plot(t, x)
```

Other plot settings

```
▪ title('Sine wave of 440 Hz')  
  ylabel('Amplitude')  
  xlabel('Time (s)')  
  legend('440 Hz')  
  ylim([-1.2 1.2])  
  xlim([0 0.01])
```

Other plots

▪ <https://www.mathworks.com/products/matlab/plot-gallery.html>



HOLD ON AND SUBPLOTS

MATLAB by default opens a new figure for every plot command

To retain current plot and draw over it

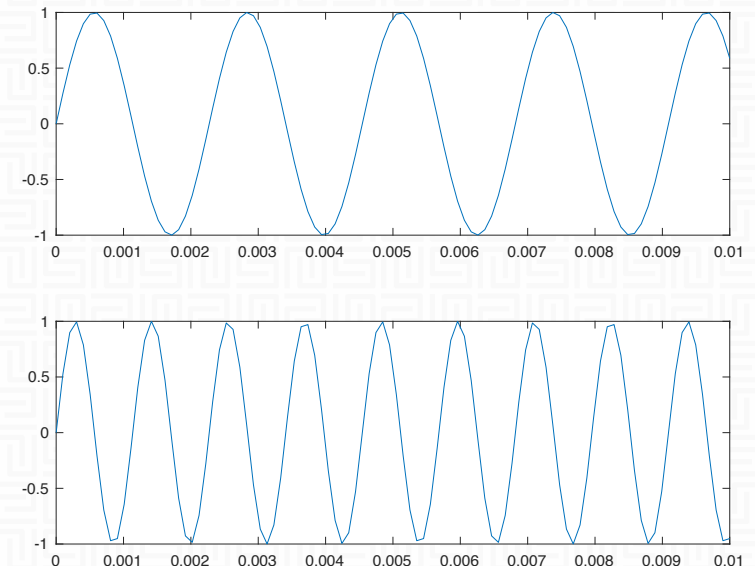
- `plot(t, x);`
hold on **% overlap**
`plot(t, x2);`
hold off **% no overlap**

To create the next plot in new figure window

- `plot(t, x);`
`figure; % new window`
`plot(t, x2);`

To organize plots into the same figure in columns

- **% index 1 in 2row1col**
`subplot(2, 1, 1)`
`plot(t, x)`
% index 2 in 2row1col
`subplot(2, 1, 2)`
`plot(t, x2)`



SOUND AS ARRAY

You can output the sine array as a sound

```
▪ t = linspace(0, 1, 44100); % 1 sec of time, 44100 samples  
  x = sin(2*pi*440*t); % sine wave of 440 Hz  
  sound(x, 44100); % play sound in Fs=44100Hz
```

Reading a sound file into array

```
▪ [y, Fs] = audioread(filename); % supporting .wav .mp3 .m4a ...
```

You may need to pause before playing the next sound

```
▪ pause(1);
```

Other useful tools: audioplayer, audiowrite

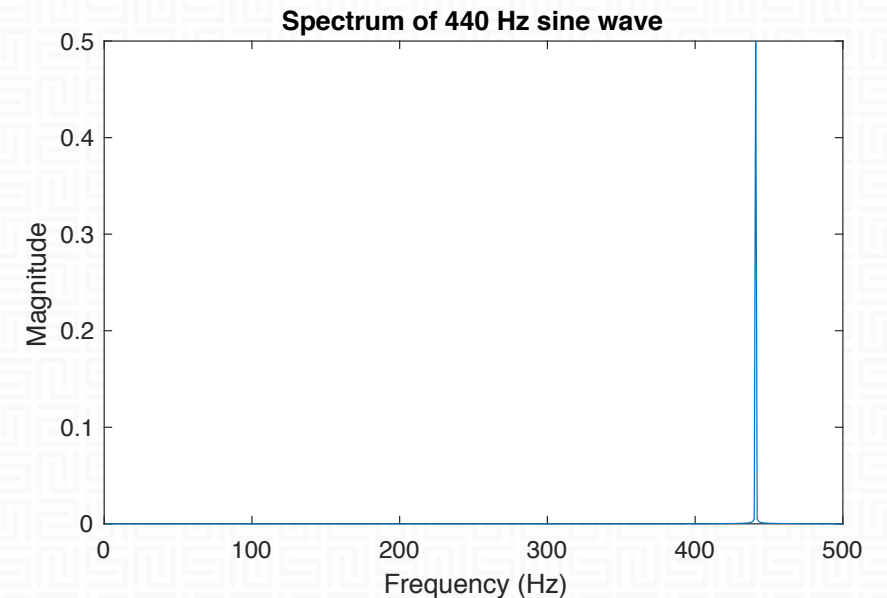
FFT IN MATLAB

Default implementation of FFT: `fft()`

- <https://www.mathworks.com/help/matlab/ref/fft.html>
- `y = fft(x, n)` % **n point FFT of x**
 - If **x** is a vector (1 row/column), return FFT of vector
 - If **x** is a matrix, return FFT of each column in columns

Plotting the spectrum

- Magnitude `m = abs(y);`
- Plot `m` and adjust plotting parameters



STFT IN MATLAB

To obtain spectrogram data of a 440 Hz sine wave of 1 second:

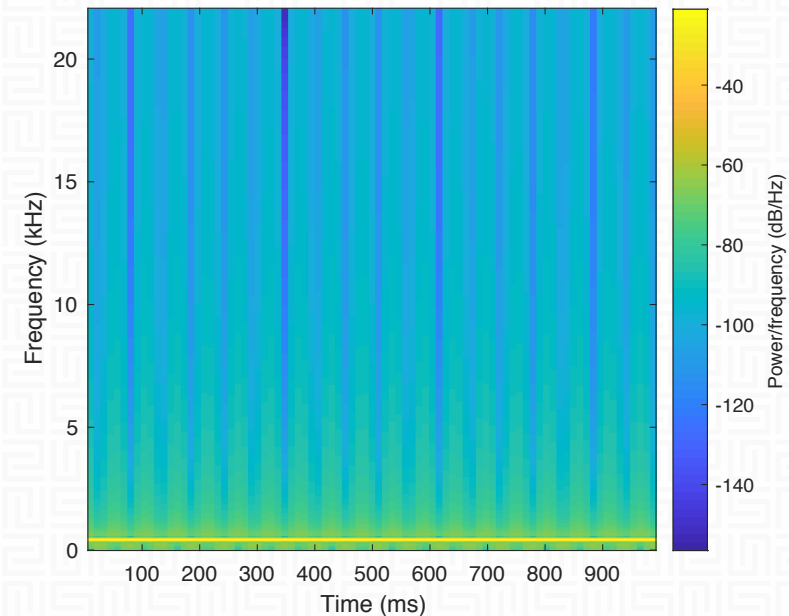
- `t = linspace(0, 1, 44100);`
`x = sin(2*pi*440*t);`
`w = hamming(1024); % Hamming window of 1024`
`s = spectrogram(x,w,512,1024,44100,'yaxis');`

Here S is a 513×85 matrix

- 85 FFT frames (window size=1024, overlap=512)
- 513 frequency bins (half of 1024-point FFT is mirrored)

To plot the spectrogram

- `spectrogram(x, ...); % without output arguments, plot in current figure`



GETTING HELP

Basic documentations

- `help mesh` % show help text of function *mesh*
- `doc LogLog` % show detailed documentation of function *LogLog*

Online tutorials available to subscribers only (including CUHK)

- <https://matlabacademy.mathworks.com>
- <https://matlabacademy.mathworks.com/artifacts/quick-reference.html?course=mlbe&language=en&release=R2019a>

READ FURTHER

*Chapter 2, “Basic audio processing”, **Speech and Audio Processing—A MATLAB-based Approach***

*Chapter 7, “Audio analysis”, **Speech and Audio Processing—A MATLAB-based Approach***