Real-Time Digital Signal Processing Laboratory

Lab 0 : Review of the Prerequisites

01/16/2024

T.A. Yongjin Eun, ye2259

TEXAS
The University of Texas at Austin

Outline

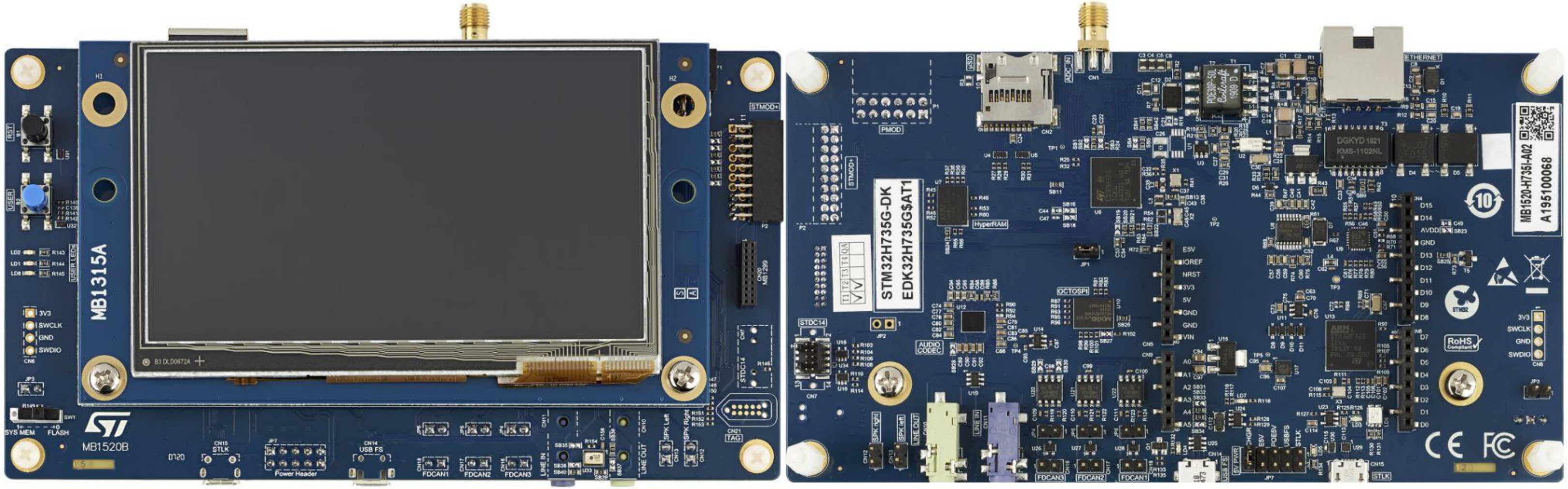
1. **Lab Overview**
   - Hardware Overview
   - Software Overview
   - Labs Overview

2. **Programming Review**
   - **C Programming**
   - **Matlab Programming**
   - **Floating Point**

3. **Linear Systems and Signals Review**

Lab Overview - Hardware

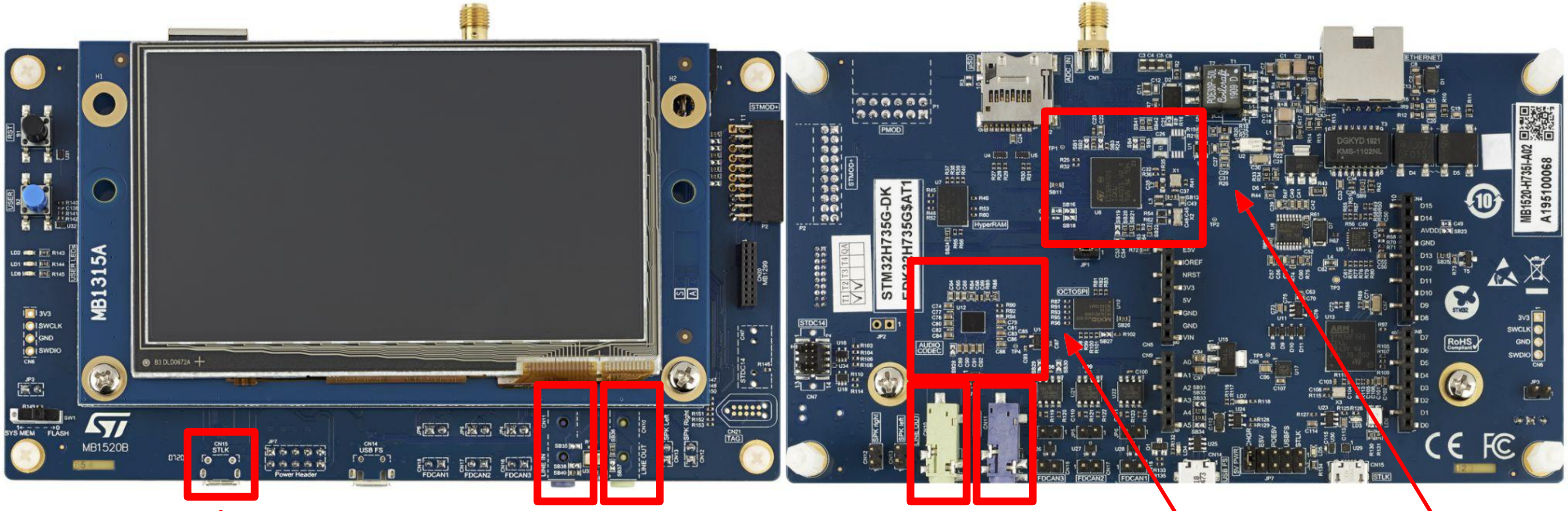
STM32H735G



- Arm® Cortex®-M7 core-based
- Two 16-bit ADCs and one 12-bit ADC, two 12-bit DACs
- Two audio jacks for input/output
- 4.3-inch RGB TFT-LCD display

Lab Overview - Hardware

STM32H735G



USB port for Programming
And Debugging

Signal
Input

Signal
Output

WM8994
Codec
(Cirrus)

STM32H7
MCU

- For more information, click HERE

Lab Overview - Software

STM32CubeIDE Installation Guide

2. Get Software

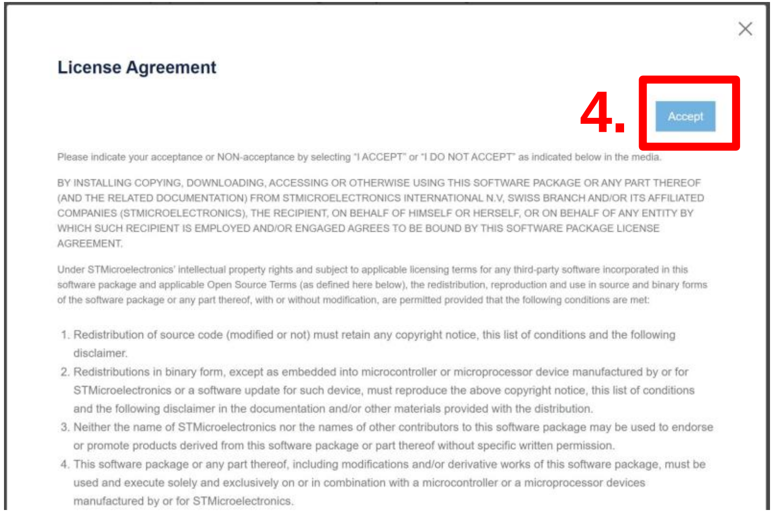
3.

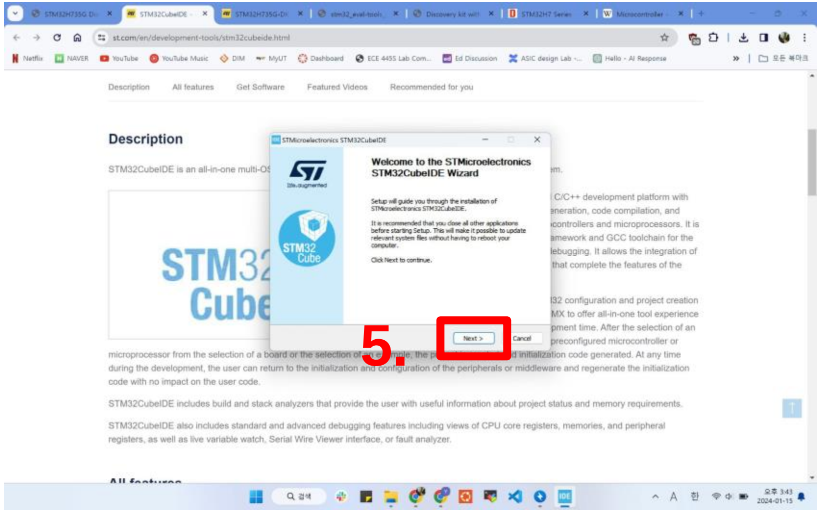
	Part Number	General Description	Latest version	Download	All versions
+	STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	1.14.0	Get latest	Select version
+	STM32CubeIDE-Lnx	STM32CubeIDE Generic Linux Installer	1.14.0	Get latest	Select version
+	STM32CubeIDE-Mac	STM32CubeIDE macOS Installer	1.14.0	Get latest	Select version
+	STM32CubeIDE-RPM	STM32CubeIDE RPM Linux Installer	1.14.0	Get latest	Select version
+	STM32CubeIDE-Win	STM32CubeIDE Windows Installer	1.14.1	Get latest	Select version

1. To download the program, go to STM32CubeIDE
2. Scroll down the Website to find ‘Get Software’
3. Click and Download the OS Version you are using

Lab Overview - Software

STM32CubeIDE Installation Guide



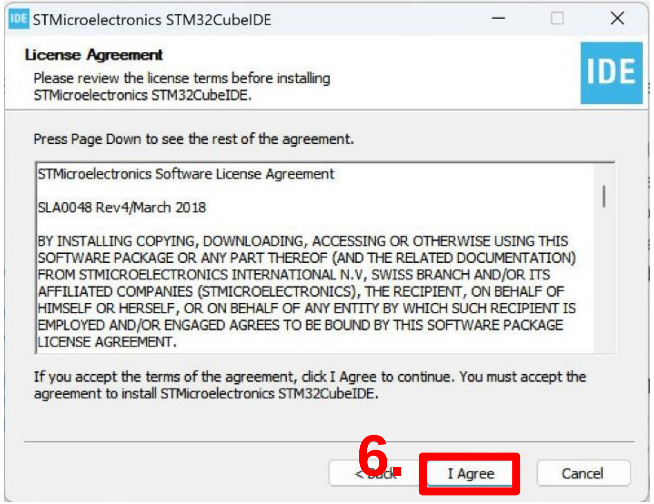


4. Read and Accept the License Agreement. Accepting it will start the download. You will find a ZIP file downloaded.

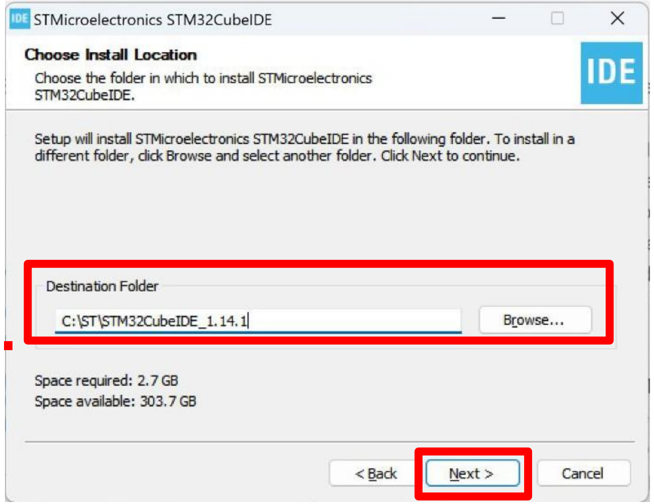
5. Unzip the ZIP file and Run the .exe file inside. You will see the CubeIDE Wizard

Lab Overview - Software

STM32CubeIDE Installation Guide



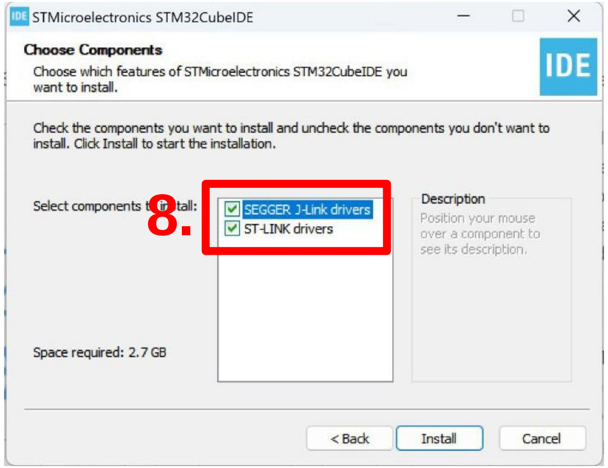
7.



6. Click ‘I Agree’ to proceed
7. Use the Default Destination Folder or Browse for a specific folder. Click Next

Lab Overview - Software

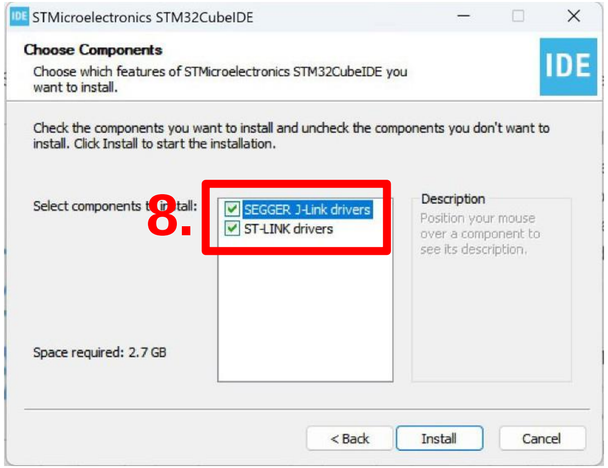
STM32CubeIDE Installation Guide



8. Make sure to check both ‘SEGGER J-Link drivers’ and ‘ST-LINK drivers’. Click Install
9. Installation will be completed within few minutes

Lab Overview - Software

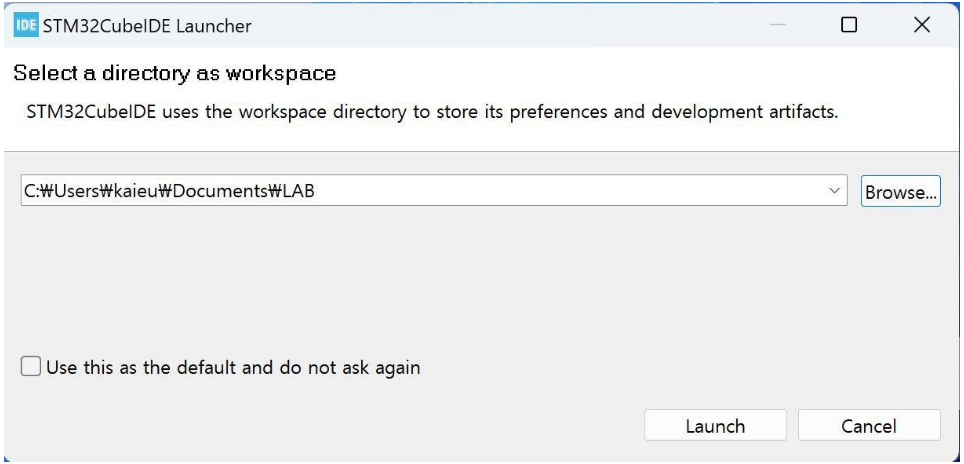
STM32CubeIDE Installation Guide



8. Make sure to check both ‘SEGGER J-Link drivers’ and ‘ST-LINK drivers’. Click Install
9. Installation will be completed within few minutes

Lab Overview - Software

STM32CubeIDE Installation Guide

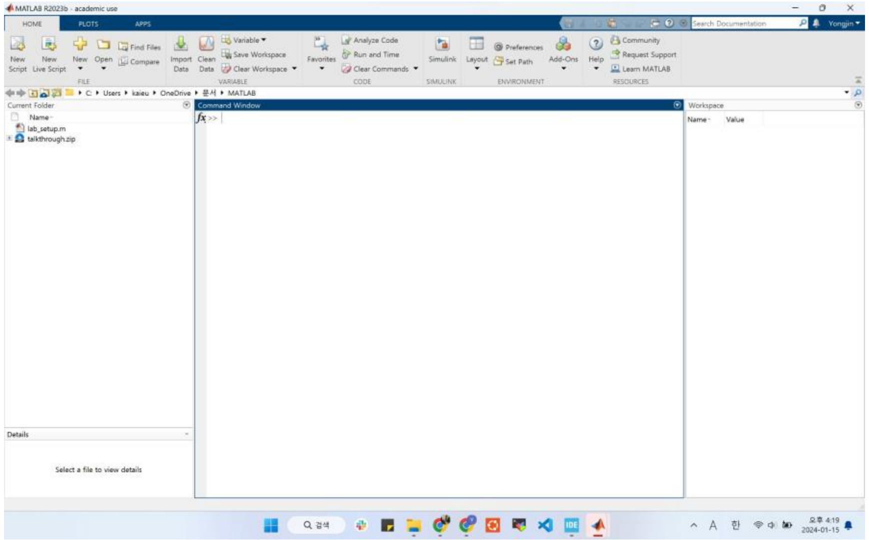


When opening the STM32CubeIDE for the first time, you will be prompted to choose the workspace directory. Browse and choose the convenient workspace location.

Make sure that the directory path only has english! STM32CubeIDE will give errors when there are non-english paths

Lab Overview - Software

MATLAB



You will need Matlab for HWs and Labs.

For the installation of Matlab, refer to the UT Wiki or the Course Website.

Lab Overview - Software

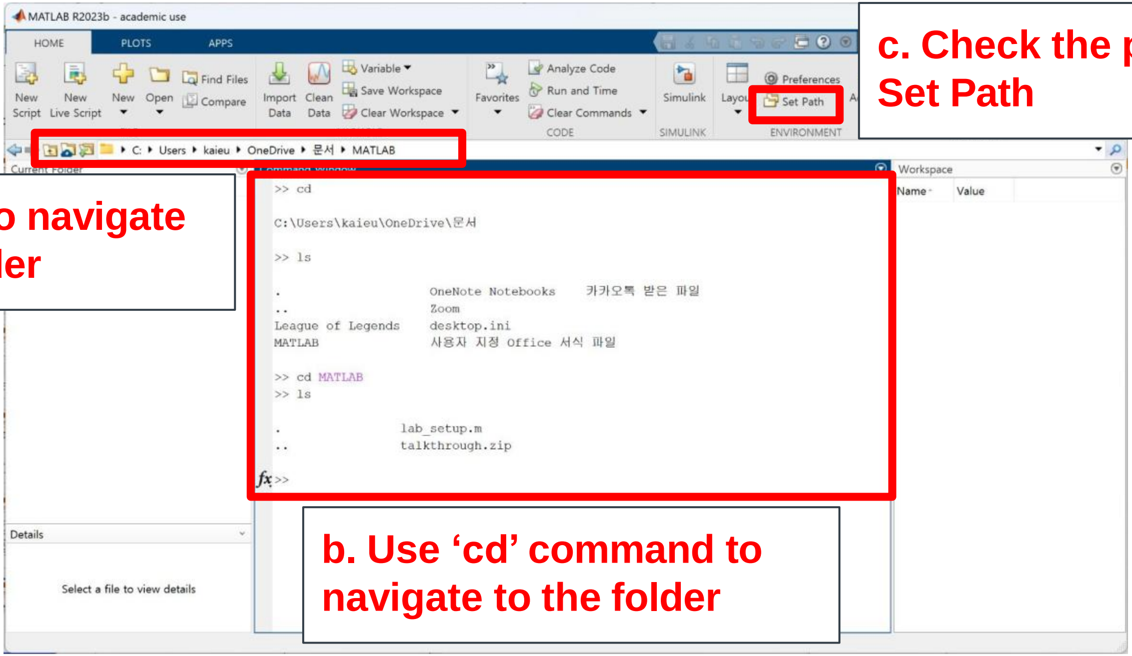
Using MATLAB to setup Starter Code

For each lab, you would need to generate a new project in CubeIDE with the starter codes. Using Matlab and the instructions in Here, you can simplify the process of making projects.

1. Download Zip File and Setup Script to the convenient location. If the script file does not download directly, make a new file named ‘lab_setup.m’ and copy/paste the contents to the file.
2. Next, you have to make sure that both files are accessible to Matlab. You can do it by navigating to the folder location using Matlab GUI ‘Current Folder’ or through commanding ‘cd’ on the command window. For those who are unfamiliar with ‘cd’ command, refer to Linux cd command. One easy way is to just move the files to the default Matlab folder. You can check it at the Matlab Toolbar : Home - Environment - Set Path. The first path on the list would be the current directory.

Lab Overview - Software

Using MATLAB to setup Starter Code



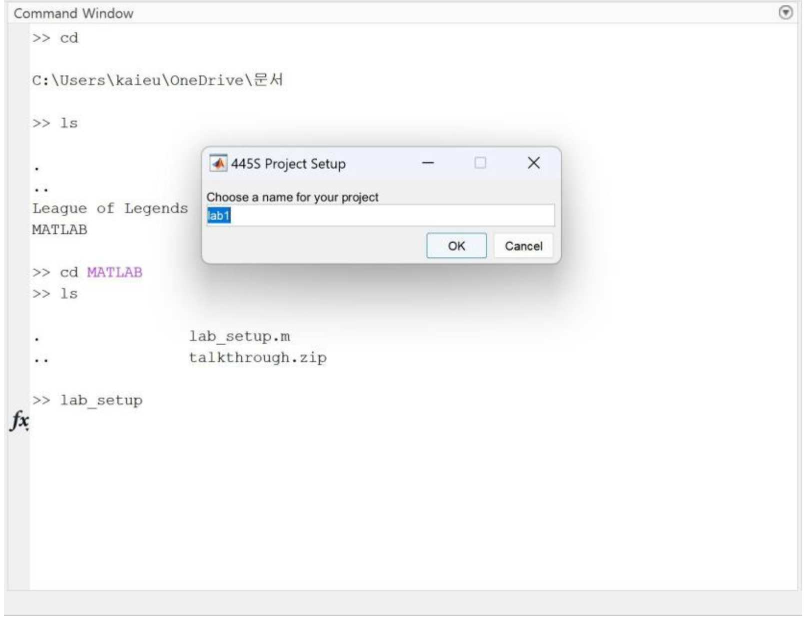
c. Check the path inside Set Path

a. Use GUI to navigate to the folder

b. Use ‘cd’ command to navigate to the folder

Lab Overview - Software

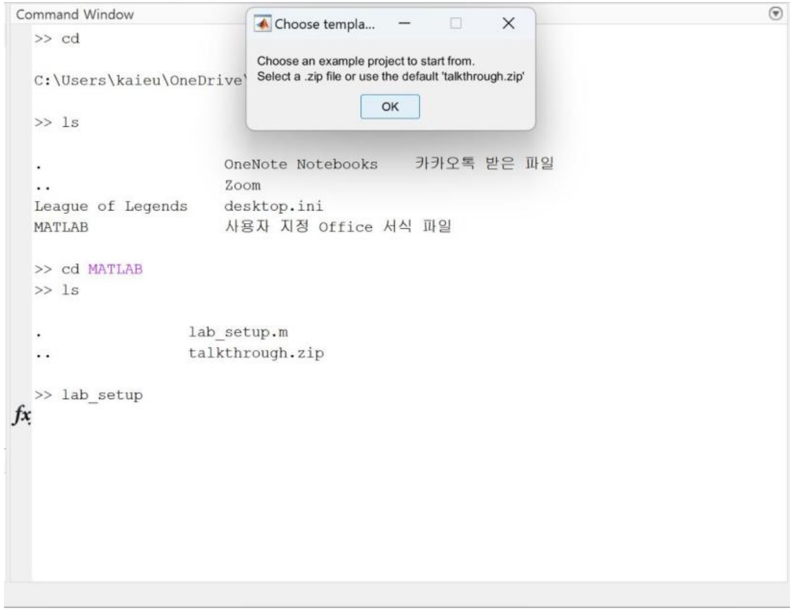
Using MATLAB to setup Starter Code



3. Run lab_setup.m by commanding ‘lab_setup’ at the Command Window. A pop-up window will ask you for the name for the project. Choose the name you want.

Lab Overview - Software

Using MATLAB to setup Starter Code

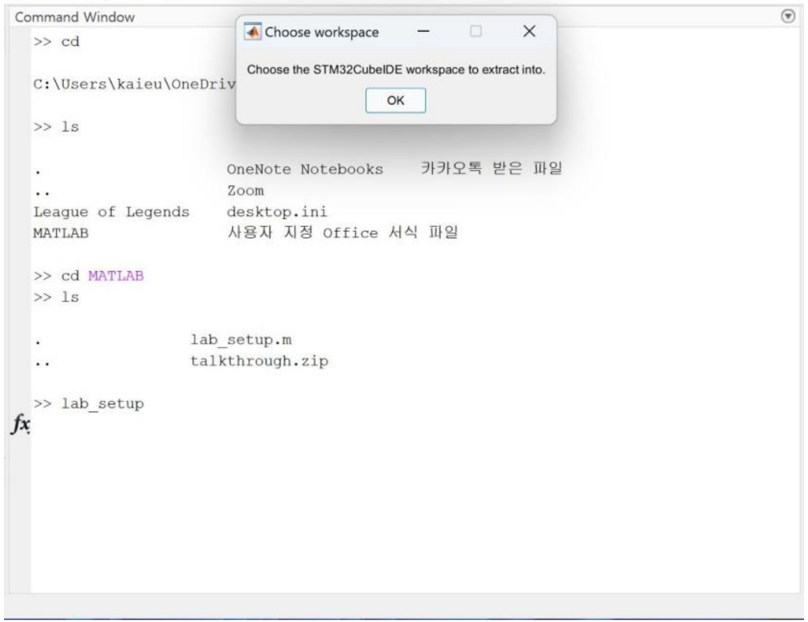


4. Next, the window will ask to choose an example .zip file for the project to start from. Choose the ‘talkthrough.zip’ file you downloaded.

TEXAS
The University of Texas at Austin
WHAT STARTS HERE CHANGES THE WORLD

Lab Overview - Software

Using MATLAB to setup Starter Code

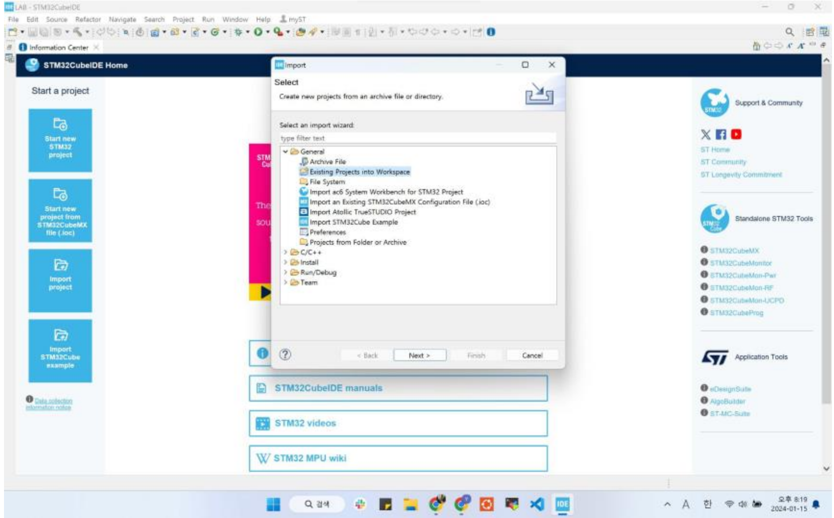


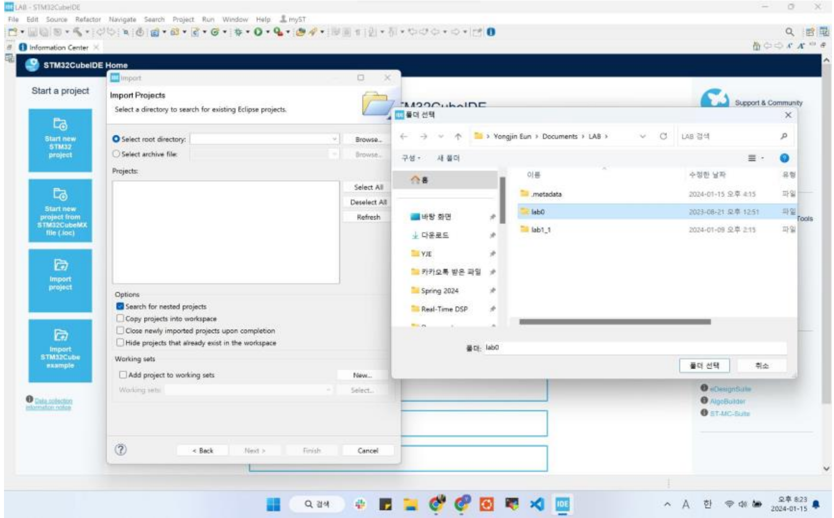
5. **Finally choose the STM32CubeIDE workspace to extract into. Choose the workspace you decided to use on the first time opening the CubeIDE.**

You will see the lab folder created under the workspace.

Lab Overview - Software

Using MATLAB to setup Starter Code





6. Open CubeIDE. Go to File - Import - Existing Projects into Workspace

Select the root directory as the directory you set in Matlab and click Finish

Lab Overview - Labs

- **Lab 0 - Review of Prerequisites**
  - H/W Overview, Programming Reviews
- **Lab 1 - Overview of Hardware and Software Tools**
- **Lab 2 - Generating Cosine and Sine Waves (2 Weeks)**
  - Sinusoidal Generation with Different Methods
- **Lab 3 - Digital Filters (3 Weeks)**
  - FIR / IIR Filter Design

Lab Overview - Labs

- **Lab 4 - Pseudo-Random Binary Sequences and Data Scramblers**
  - Pseudonoise sequence, Data scrambler / descrambler
- **Lab 5 - Digital Data Transmission by Baseband Pulse Amplitude Modulation (PAM) (3 Weeks)**
- **Lab 6 - Quadrature Amplitude Modulation (QAM)**
- **Lab 7 - Vocoder and Guitar Effects**
  - Vocoder, Flanger, Distortion

Programming Review - C

Data Types

- C language does not provide exact rules for data types
    - Example: int
        - Must be at least 16 bits, but can be larger
        - Must include the range [−32,767, +32,767]. Often, -32,768 is included as well.
- Toolchain used in lab provides exact specifications for data types
    - int16_t : 16-bit signed integer
    - uint32_t : 32 bit unsigned integer
    - float32_t : 32 bit (single precision) floating point

Programming Review - C

Pointers and functions

- Example: measuring clock cycles

```
uint32_t* systick = (uint32_t*) 0xe000e018;
void tic(void)
{
    t = *systick;
}
uint32_t toc(void)
{
    return t - *systick;
}
```

Declaring a pointer ‘systick’ and initializing as given address, ‘0xe000e018’

Sets value of ‘t’ as a value pointed by ‘systick’

Returns the difference between ‘t’ and the value pointed by ‘systick’

Programming Review - C

Arrays and For Loops

- Example: Cosine Lookup Table

```
int16_t table[16];
float32_t amplitude;
float32_t omega0 = 0.0576;

for (uint32_t n = 0; n < 16; n+=1)
{
    amplitude = arm_sin_f32(n * omega0);
    table[n] = OUTPUT_SCALE_FACTOR * amplitude;
}
```

Declares an array, ‘table’, of 16 elements. Each elements are of ‘int_16t’ type.

‘arm_sin_f32’ is an ARM math function that computes sine function for float32 type

Scale ‘amplitude’ with scaling factor and store it the array, ‘table’

Programming Review - C

Linear Buffers and Circular Buffers

- Example: Linear Buffer Shifting

```
float32_t x[16] = {0};
   …
void linear_buffer_shift(float32_t new_element)
{
   for (uint32_t n = 15; n >0; n-=1)
   {
      x[n] = x[n-1];
   }
   x[0] = new_element;
}
```

x15 = x14
x14 = x13
⋮
x1 = x0
x0 ←

→ Declares an array, ‘x’, of 16 elements and initialize all elements to 0

→ Using for loop, the array is shifted to the right.

→ First element of ‘x[0]’ is replaced with a ‘new_element’

Programming Review - C

Linear Buffers and Circular Buffers

- Example: Circular Buffer Shifting

```
float32_t x[16] = {0};
int32_t position = 0;
    …
void circular_buffer_shift(float32_t new_element)
{
    position = (position-1) % 16
    x[position] = new_element
}
```

0 15
14
13
.
.
.
1
0

Declares an array, ‘x’, of 16 elements and initialize all elements to 0

‘position’ is the index of the buffer for inserting new element

Using modulo operation, update the index

Element of ‘x[position]’ is replaced with a ‘new_element’

Programming Review - Matlab

Data Types in MATLAB

- By default, every variable in MATLAB is a 2d array of double precision floating point values
- To use another datatype, call the corresponding function (single(), int16(), uint32(), etc)
- View the size and type of variables using the whos() function
- By default, **i** and **j** represent $\sqrt{-1}$. Be careful naming variables that overwrite these!
- Data automatically become complex if any multiple of **j** is added, e.g. 1-3j

Programming Review - Matlab

Vector and Array Operations

- Matlab Array index starts at **'1'** unlike C or other languages!

```
A = [1,2,3
     4,5,6];
B = [7,8,9
     3,2,1];
disp(A);
```

```
     1     2     3
     4     5     6
```

```
disp(A(1, 1))
```

```
     1
```

```
disp(A(0, 1))
```

```
Index in position 1 is invalid. Array indices must be positive integers or logical values.
```

Programming Review - Matlab

Vector and Array Operations

- Example: Element-wise product of Two matrices using for loops

```
A = [1,2,3
     4,5,6];
B = [7,8,9
     3,2,1];

for i_row = 1:2
    for i_col = 1:3
        C(i_row,i_col) = A(i_row,i_col) * B(i_row,i_col);
    end
end

disp(C)
```

```
     7    16    27
    12    10     6
```

Programming Review - Matlab

Vector and Array Operations

- Example: Simpler Element-wise product of Two matrices

A 2 x 3
B 2 x 3

```
A = [1,2,3
     4,5,6];
B = [7,8,9
     3,2,1];

C_2 = A .* B; % .* returns the element-wise product of two matrices

disp(C_2)
```

```
     7    16    27
    12    10     6
```

B'

* [M x N] * [N x M]

```
C_3 = A * B'; % Transpose using ' will allow Matrix Multiplication
disp(C_3)
```

2x3 ⇒ 3x2

```
    50    10
   122    28
```

```
C_4 = A * B; % Using * on matrices will perform Matrix Multiplication
```

```
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 operate on each element of the matrix
individually, use TIMES (.*) for elementwise multiplication.
```

Related documentation

```
% Note that this operation is impossible as the dimensions are incorrect.
% A = 2 x 3 , B = 2 x 3
```

A.^2

Programming Review - Matlab

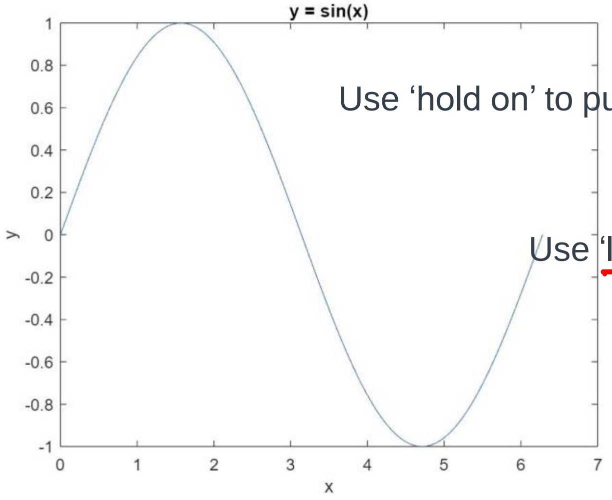
Creating Plots in Matlab

```
x = linspace(0,2*pi);
y1 = sin(x);
plot(x,y1)
title('y = sin(x)')
xlabel('x')
ylabel('y')
```

Use 'plot' to represent continuous signals

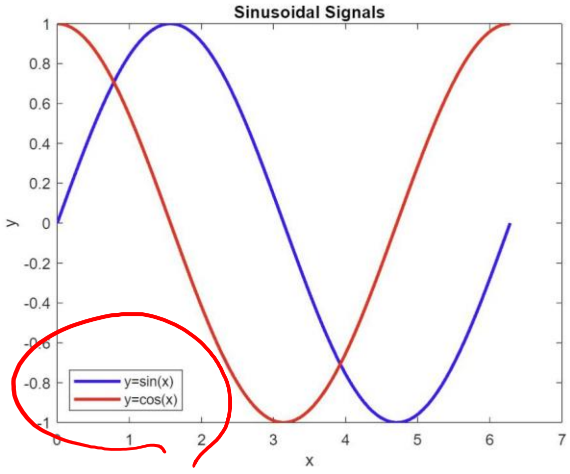
Use 'title', 'xlabel', 'ylabel' to label plots and axes

```
plot(x,y1,'LineWidth', 2, 'Color','b')
hold on
plot(x,y2, 'LineWidth', 2, 'Color','r')
hold off
legend('y=sin(x)','y=cos(x)', 'Location','southwest')
title('Sinusoidal Signals')
xlabel('x')
ylabel('y')
```



Use 'hold on' to put multiple plots on one figure

Use 'legend' to add legend to plots

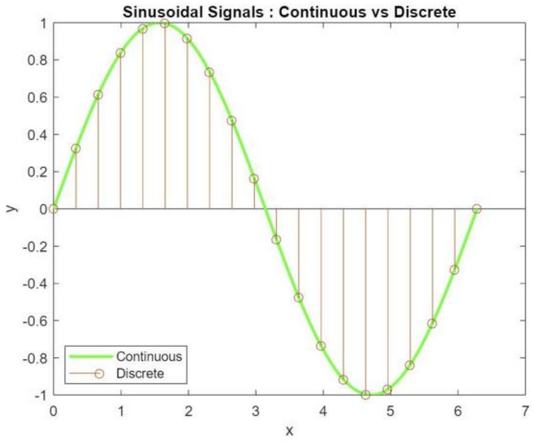


Programming Review - Matlab

Creating Plots in Matlab

```
x2 = linspace(0,2*pi,20);
y3 = sin(x2);
plot(x, y1, 'LineWidth', 2, 'Color', 'g');
hold on
stem(x2, y3)
hold off
legend('Continuous','Discrete', 'Location','southwest')
title('Sinusoidal Signals : Continuous vs Discrete')
xlabel('x')
ylabel('y')
```

Use 'stem' to represent discrete signals



Use the incantation `set(0,'DefaultFigureWindowStyle','Docked')` to keep all figures in the same window but make an new tab for each

If you are unsure of the syntax of functions inside Matlab, search at [Matlab Support](). They have a very detailed description of functions in their website with examples.

Programming Review - Floating Point

Floating - Point

Why do we need a floating point?

To represent **real numbers / decimal fractions** with **a wide range of magnitudes.** Broader range with fractional parts.

Floating-point numbers provide **flexibility** and **a wide range of representations**, along with **efficient computational compatibility with hardwares.**

Two Main parts of Floating-Point Numbers

3. sign : $\pm$

1. **Significand (Mantissa) :** Digits of Number
2. **Exponent :** Where the decimal point is placed.

Programming Review - Floating Point

Floating - Point

Significand	Exponent	Scientific Notation	Fixed-Point Representation
1.5	4	$1.5 * 10^{4}$	15000
- 2.001	2	$- 2.001 * 10^{2}$	-200.1
5	-3	$5 * 10^{-3}$	0.005
6.667	-11	6.667e-11	0.00000000006667

1) https://floating-point-gui.de/formats/fp/

Programming Review - Floating Point

Floating - Point

Format	Total Bits	Significand Bits	Exponent Bits	Smallest Number	Largest Number
Single	32	23 + 1 sign	8	$1.2 * 10^{-38}$	$3.4 * 10^{38}$
Double	64	52 + 1 sign	11	$2.2 * 10^{-308}$	$1.8 * 10^{308}$

31 8 22 0

exponent significand

sign

$$Value = (-1)^{\text{sign}} \times 2^{(E-127)} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right)$$

1) https://floating-point-gui.de/formats/fp/

Programming Review - Floating Point

Floating - Point

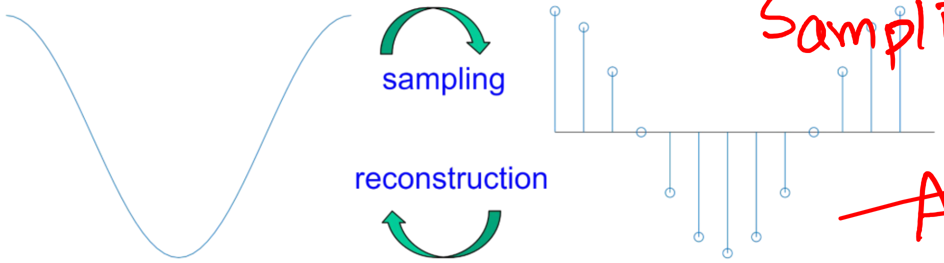
- **Integer and other fixed-point data types** have the **same spacing** between all possible values
  - Useful for representing natural numbers or the value read from ADC
- **Distance between adjacent floating point numbers is variable**
  - Useful for representing filter coefficients
  - Less build up of error for sequences of operations compared to fixed point
- Single precision (near 1.0) is roughly equivalent to 8 decimal points
  - Largest possible value is about 3.4×10^{38}
  - Near zero, the smallest distance between values is about 1.4×10^{-45}

Linear Systems and Signals Review

Sampling and Reconstruction

- **Sampling (continuous-time to discrete-time conversion)**
  - $x[n] = x(nTs)$
- **Reconstruction (discrete-time to continuous-time conversion)**
  - A continuous-time signal $x(t)$ with frequencies no higher than f_{max} can be reconstructed exactly from its samples $x[n] = x(nTs)$ if samples are taken at a sampling rate $f_s > 2f_{max}$

Shannon – Nyquist Sampling Theorem.



~~Aliasing~~

Linear Systems and Signals Review

Transforms

Preview of the Different Transforms we will use in this class

- Laplace Transform
- Z-Transform
- Fourier Transform
- Discrete-Time Fourier Transform (DTFT)
- Fourier Series
- Discrete Fourier Transform (DFT)

rect(t) sinc(f)

All of these have certain properties in common (with a few caveats)

- Multiplication ↔ Convolution
- Stretch ↔ Contract

Linear Systems and Signals Review

Transforms

Laplace

$$\begin{tikzcd}[column sep=small] & H(s) \arrow[dl, leftrightarrow] \arrow[dr, "s = j\omega"] & \\ h(t) \arrow[rr, leftrightarrow] & & H(j\omega) \end{tikzcd}$$

Fourier

Z transform

$$\begin{tikzcd}[column sep=small] & H(z) \arrow[dl, leftrightarrow] \arrow[dr, "z = e^{j\hat{\omega}}"] & \\ h[n] \arrow[rr, leftrightarrow] & & H(e^{j\hat{\omega}}) \end{tikzcd}$$

DTFT

Linear Systems and Signals Review

Fourier Transform

Basic Properties of Fourier Transform

1. **Linearity**

$$a\,x(t) + b\,y(t) \iff a\,X(w) + b\,Y(w)$$

2. **Time Shifting**

$$y(t) = x(t - t_0) \iff Y(w) = e^{-jwt_0} X(w)$$

3. **Frequency Shifting**

$$e^{jw_0 t}\,x(t) \iff X(w - w_0)$$

4. **Time Scaling**

$$x(at) \iff \frac{1}{|a|} X\left(\frac{w}{a}\right)$$

1\. $1 \xRightarrow{\mathcal{F}} \delta(f)$

2\. $\cos at \Rightarrow \frac{1}{2}\delta\left(f - \frac{a}{2\pi}\right) + \frac{1}{2}\delta\left(f + \frac{a}{2\pi}\right)$

3\. $\sin at \Rightarrow \frac{1}{2j}\delta\left(f - \frac{a}{2\pi}\right) + \frac{1}{2}\delta\left(f + \frac{a}{2\pi}\right)$

4\. $\text{rect}(t) \xRightarrow{\mathcal{F}} \text{sinc}(f)$

$\delta(f - f_0) * X(f) = X(f - f_0)$

Linear Systems and Signals Review

Fourier Transform

Example 1. What is a Fourier Transform of

$$x(t) = (1 + \cos(2\pi t))(rect(t))$$

$$\mathcal{F}(x(t)) = \underline{\mathcal{F}(1+\cos 2\pi t)} * \underline{\mathcal{F}(rect(t))}$$

$$\{\underline{\mathcal{F}(1)} + \underline{\mathcal{F}(\cos 2\pi t)}\} * \qquad \rightarrow \mathrm{sinc}(f)$$

$$\{\delta(f) + \tfrac{1}{2}\delta(f-1) + \tfrac{1}{2}\delta(f+1)\} * \mathrm{sinc}(f)$$

$$= \mathrm{sinc}(f) + \tfrac{1}{2}\mathrm{sinc}(f-1) + \tfrac{1}{2}\mathrm{sinc}(f+1)$$

Linear Systems and Signals Review

Fourier Transform

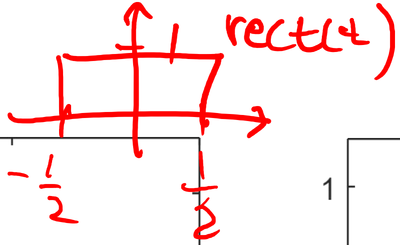
Example 1. What is a Fourier Transform of

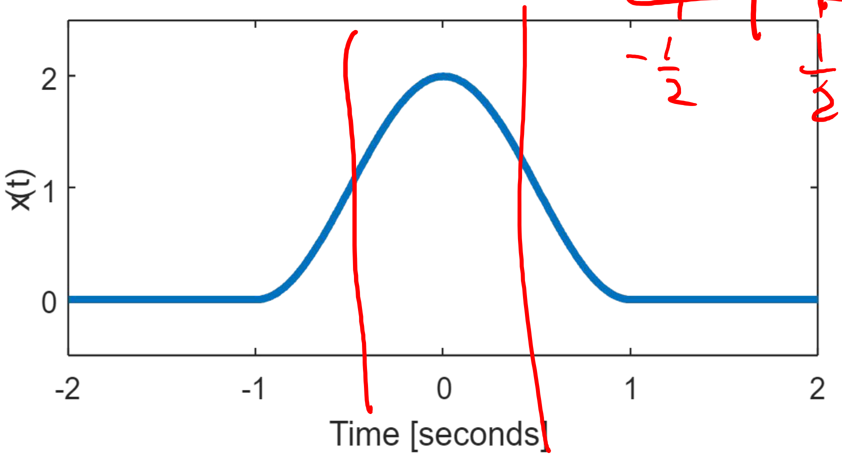
$$x(t) = (1 + \cos(2\pi t))(rect(t))$$

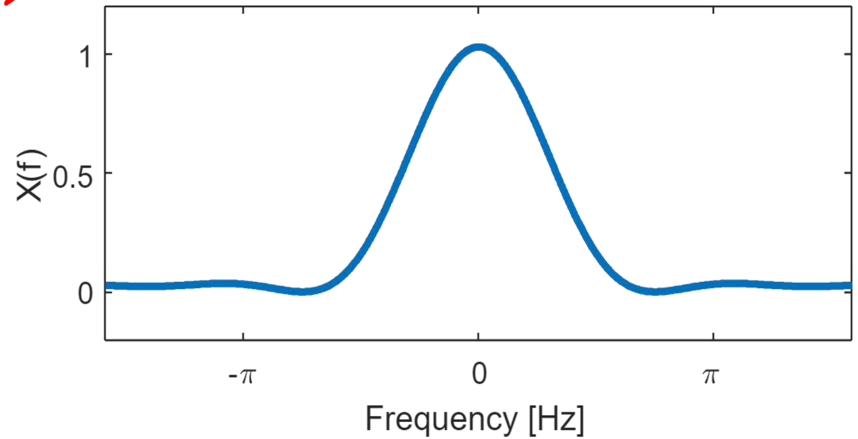
$$\begin{aligned}
\mathcal{F}(x(t)) &= \mathcal{F}((1 + \cos(2\pi t))(rect(t))) \\
&= \mathcal{F}(1 + \cos(2\pi t)) * \mathcal{F}(rect(t)) \\
&= \{\mathcal{F}(1) + \mathcal{F}(\cos(2\pi t))\} * \mathcal{F}(rect(t)) \\
&= \left\{\delta(f) + \frac{1}{2}\delta(f-1) + \frac{1}{2}\delta(f+1)\right\} * sinc(f) \\
&= \frac{1}{2}sinc(f-1) + sinc(f) + \frac{1}{2}sinc(f+1)
\end{aligned}$$

Linear Systems and Signals Review

Fourier Transform







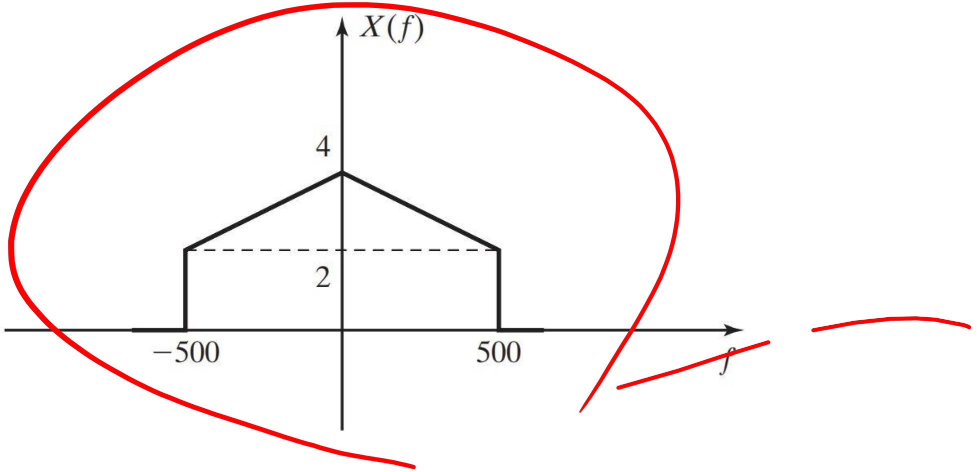
$$x(t) = (1 + \cos(2\pi t))\,(rect(t))$$

$$\mathcal{F}(x(t)) = \frac{1}{2}sinc(f-1) + sinc(f) + \frac{1}{2}sinc(f+1)$$

Linear Systems and Signals Review

Fourier Transform

Example 2. The Fourier transform of a signal is as the following figure. Determine and sketch the Fourier transform of the signal $x_1(t) = -x(t) + x(t)\,\cos(3000\pi t) + 2\,x(t)\,\cos^2(5000\pi t)$



Proakis, J. G., & Salehi, M. (2014). Chapter 2. Signals and Linear Systems. Problem 2. 47. In *Fundamentals of Communication Systems*. essay, Pearson.

Linear Systems and Signals Review

Fourier Transform

Example 2. $x_1(t) = -x(t) + x(t)\,\cos(3000\pi t) + 2\,x(t)\,\cos^2(5000\pi t)$

$\mathcal{F}(x_1(t)) = -\underline{\mathcal{F}(x(t))} + \mathcal{F}(\underline{x(t)\cdot\cos(3000\pi t)})$

$+ 2\cdot\mathcal{F}(\underline{x(t)\cdot\cos^2(5000\pi t)})$

$= -X(f) + \underline{\mathcal{F}(x(t)) * \mathcal{F}(\cos 3000\pi t)}$

$+ 2\,\mathcal{F}(x(t)) * \mathcal{F}(\underline{\cos^2 5000\pi t})$

$\underbrace{\frac{X(f-1500)}{2} + \frac{X(f+1500)}{2}}$

$\cos^2 ax = \frac{1+\cos 2ax}{2}$

Linear Systems and Signals Review

Fourier Transform

Example 2. $$x_1(t) = -x(t) + x(t)\,\cos(3000\pi t) + 2\,x(t)\,\cos^2(5000\pi t)$$

$$\begin{aligned} x_1(t) &= -x(t) + \cos(3000\pi t)x(t) + 2\cos^2(5000\pi t)x(t) \\ &= -x(t) + \cos(3000\pi t)x(t) + (1 + \cos(10000\pi t))x(t) \end{aligned}$$

$$\begin{aligned} \mathcal{F}(x_1(t)) &= \mathcal{F}(-x(t)) + \mathcal{F}(\cos(3000\pi t)x(t)) + \mathcal{F}((1 + \cos(10000\pi t))x(t)) \\ &= -X(f) + \mathcal{F}(\cos(3000\pi t)) * X(f) + \mathcal{F}(1 + cos(10000\pi t)) * X(f) \\ &= -X(f) + \left\{\frac{1}{2}\delta(f - 1500) + \frac{1}{2}\delta(f + 1500)\right\} * X(f) + \mathcal{F}(1 + cos(10000\pi t)) * X(f) \\ &= -X(f) + \frac{1}{2}X(f - 1500) + \frac{1}{2}X(f + 1500) + \left\{\delta(f) + \frac{1}{2}\delta(f - 5000) + \frac{1}{2}\delta(f + 5000)\right\} * X(f) \\ &= \cancel{-X(f)} + \frac{1}{2}X(f - 1500) + \frac{1}{2}X(f + 1500) + \cancel{X(f)} + \frac{1}{2}X(f - 5000) + \frac{1}{2}X(f + 5000) \\ &= \frac{1}{2}X(f - 1500) + \frac{1}{2}X(f + 1500) + \frac{1}{2}X(f - 5000) + \frac{1}{2}X(f + 5000) \end{aligned}$$

Linear Systems and Signals Review

Fourier Transform

Example 2.