Islamic University of Technology (IUT)

Organization of Islamic Cooperation (OIC)

Department of Electrical and Electronic Engineering (EEE)

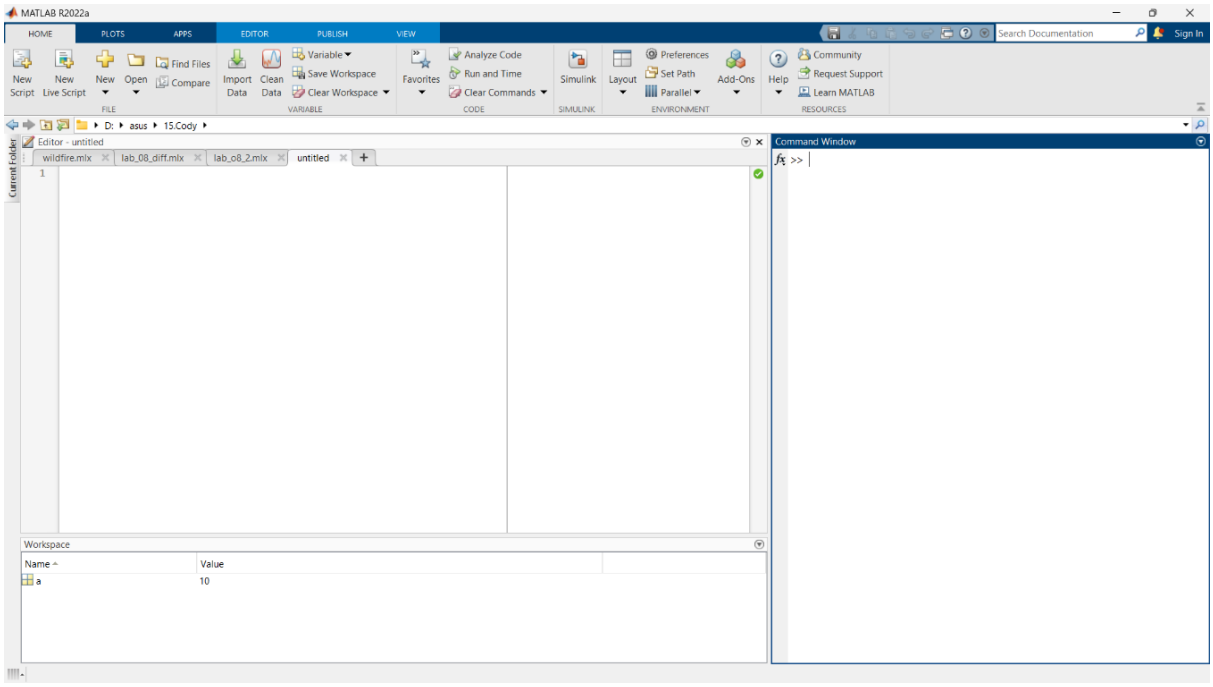
COURSE NO	:	EEE 4416
LAB NO	:	01 (Part – A)
TOPIC	:	FAMILIARIZATION WITH THE MATLAB ENVIRONMENT

Objective:

This lab will provide a general introduction to the MATLAB programming environment. The students will be introduced to the MATLAB interface. They will become familiar with some of the basic data structures used in MATLAB programming and common operations.

MATLAB interface

When you start MATLAB, the "MATLAB Desktop" will appear. Its layout will vary according to the operating system and version of MATLAB. Figure 1 shows version R2022a (mentioned in the top left corner).



1. MATLAB

The main components that you will find here are –

i. Menu bar
ii. Directory
iii. Command Window
iv. Workspace
v. Editor (Script or live script)

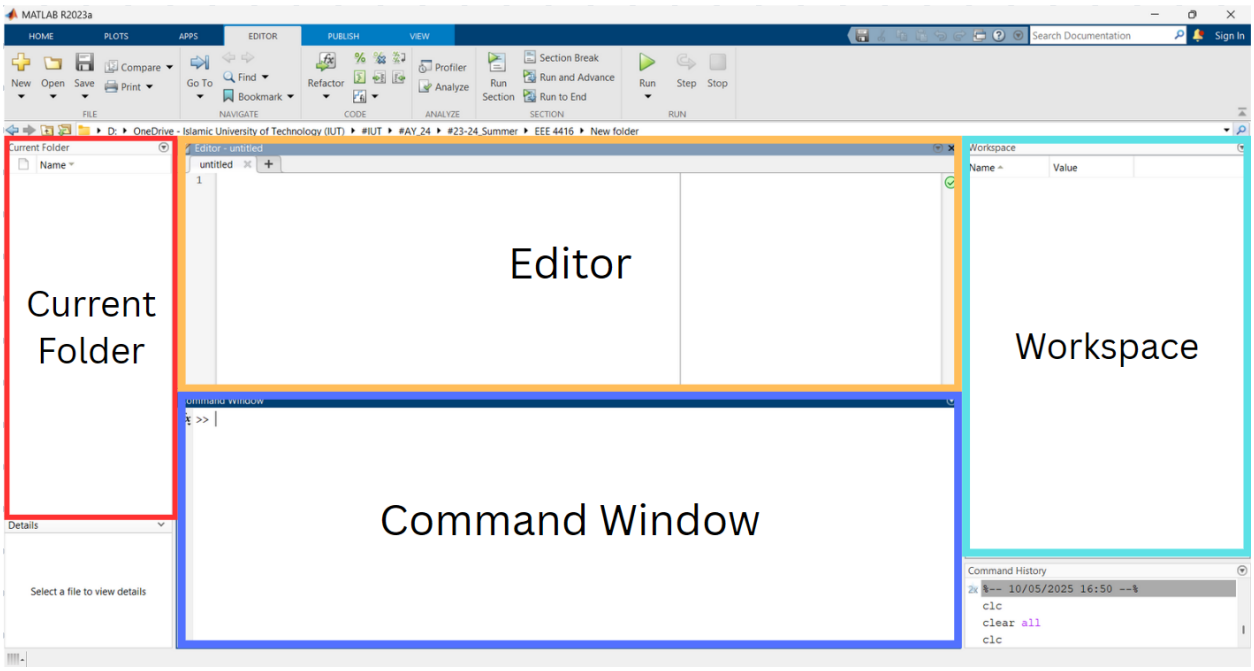
Prepared by: Aseer Imad Keats, Asif Newaz and Salman Hossain

vi. Contents folder
vii. Command history

When working with MATLAB, you may not need all the interface components at once. MATLAB provides options to hide unnecessary panels (e.g., command history), allowing you to declutter your workspace. You can also drag and reposition the components to suit your preferences. Additionally, you can resize the windows—such as enlarging the Editor—to focus on the areas you use most and improve visibility.

Task

- ✓ *Learn how to hide a panel*
- ✓ *learn how to reposition a panel*
- ✓ *learn how to resize a panel*



2. MATLAB interface

Command Window

The Command Window is where you enter MATLAB commands and expressions at the prompt (>>), and where the results of your commands are displayed. If a MATLAB command takes too long to execute, you can abort it by pressing **Ctrl-C**. This will stop the command without quitting MATLAB, giving you control of the Command Window and a fresh prompt.

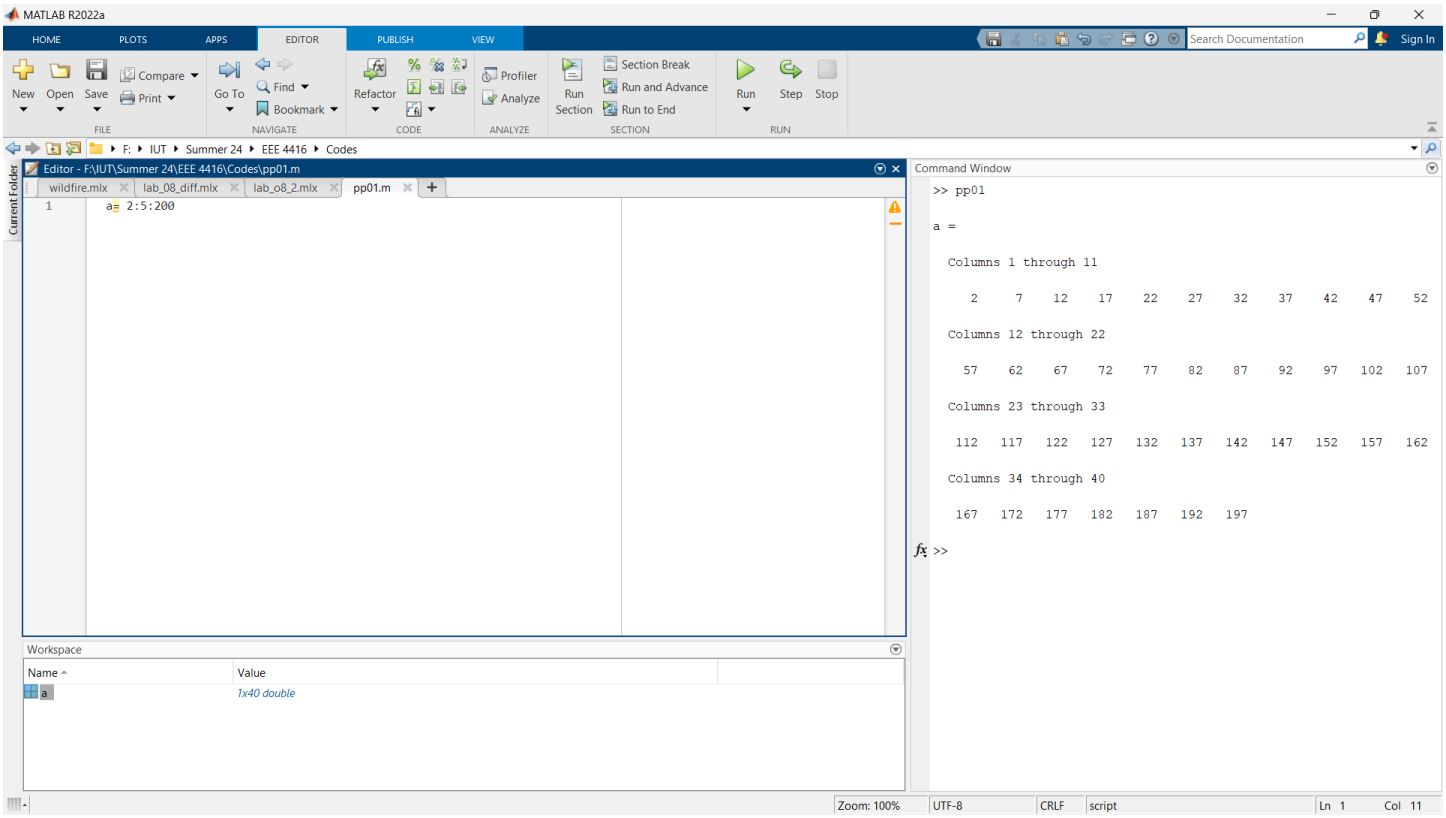
- Used for simple commands to test output
- Used for calling user-defined functions
- Not used for writing the main program

Editor

This is where you write your main code. Once you have written a program, you can execute it using the 'Run' button on the menu bar.

MATLAB provides two types of editors.

i. Script (extension - .m)
ii. Live script (extension - .mlx)



3. MATLAB Scripts

- Use scripts for creating user-defined functions
- Use live scripts for regular tasks

When you run a script, the output will be displayed in the command window. When you run a live script, the output will be displayed on the right-hand side. Live script allows you to create a notebook where detailed instructions, codes, notes, output – everything can be shown, making it more suitable for different tasks.

Workspace

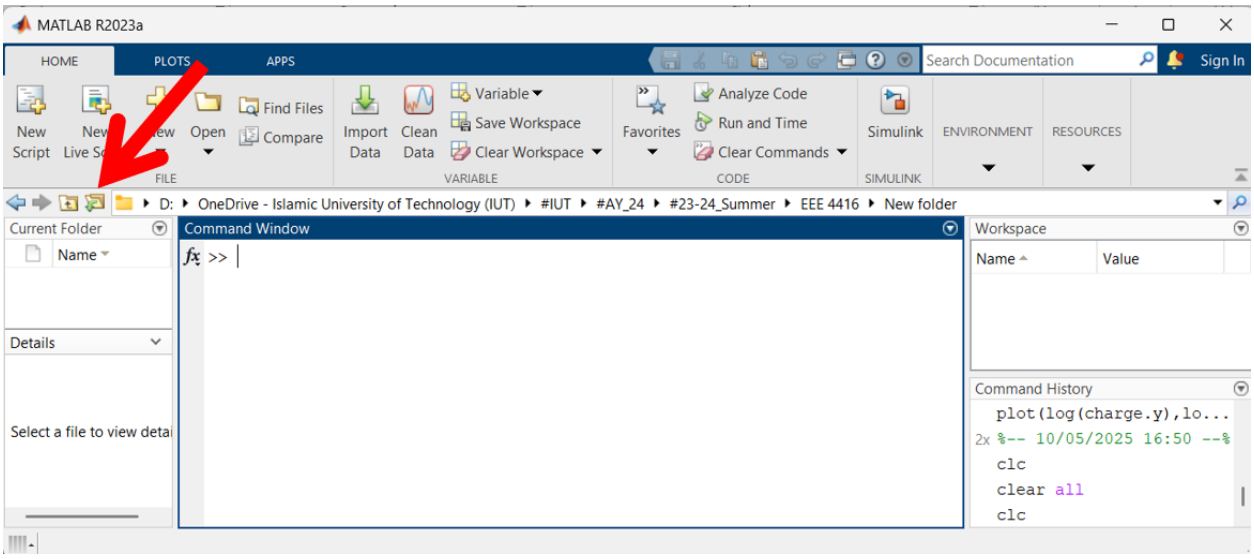
The workspace contains the variables created during a MATLAB session. The Workspace Browser displays these variables along with some related information. By double-clicking a variable in the Browser, the Array Editor opens, allowing you to view and sometimes modify the variable's properties.

Directory

This is where your files are saved. You can change your directory very easily to save your files in your desired folder.

Prepared by: Aseer Imad Keats, Asif Newaz and Salman Hossain

To change the current directory, hover your mouse over the File icon with the green arrow, indicated by the red arrow, and click. A window will appear showing the folder structure on your computer. Select the desired folder and click OK. The current folder name, displayed next to the file icon, will update to your selection.



Content Folder
You can see all the files saved in your current directory in this window.

Live Script

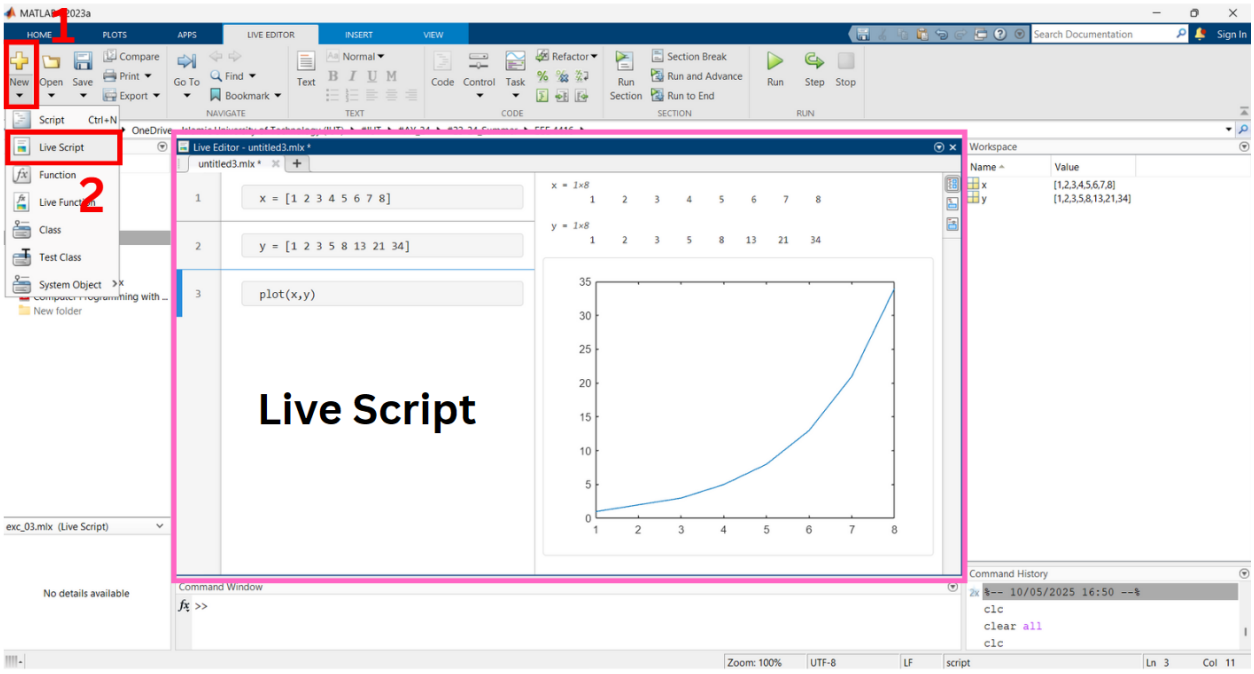
In MATLAB, Live Scripts provide an interactive environment that allows you to write and execute code while also viewing the results immediately in the same document. This format is especially useful for visualizing the outputs of your code, such as graphs and plots, alongside the actual code and comments.

To create a Live Script in MATLAB (versions R2016a and later), follow these steps:

1. Click on the **New** button (highlighted by the red arrow in the image).
2. Select **Live Script** from the dropdown menu.
3. This will open a new editor window where you can write MATLAB code.

A Live Script combines code with formatted text, allowing for a more dynamic way to document your work. In the image, you can see an example where:

- The **Editor** section contains the MATLAB code (e.g., x = 1:8 and y = [1 2 3 5 8 13 21 34]).
- The **right panel** shows the output of the code (in this case, a plot generated from the plot(x, y) command).



Section Break and 'Run and Advance'

In the MATLAB Live Editor, the **Run and Advance** command is used to execute a section of your script and then automatically move the cursor to the next section. This feature is particularly useful for executing code in blocks, allowing you to step through your script efficiently without manually navigating between sections.

Section Breaks:

To begin using **Run and Advance**, you need to insert **Section Breaks** in your script. Section breaks divide your code into logical blocks, allowing you to run each block independently. In the figure, the **Section Break** button (highlighted in red) is used to create these divisions. You can also use the shortcut **Ctrl + Alt + Enter** to insert a section break.

Run and Advance:

Once you have set up section breaks, you can use the **Run and Advance** command to execute the current section and automatically move to the next one. In the figure, the code is divided into sections, and lines 1, 3, and 5 are executed. The results of each section are shown accordingly in the **right panel** (the plot).

To run a section and advance to the following:

1. Place the cursor in the section you want to execute.
2. Click the **Run and Advance** button (highlighted in green in the figure) or press **Ctrl + Shift + Enter**.

This allows you to step through the script, running sections one after the other without needing to manually click between sections. It's especially useful for testing or debugging, as each section's output is displayed immediately after execution, making it easier to analyze results.

Introduction to Simulink

Simulink is a graphical programming environment integrated with MATLAB, used primarily for modeling, simulating, and analyzing dynamic systems. It is widely used in fields like control systems, signal processing, communications, and embedded systems.

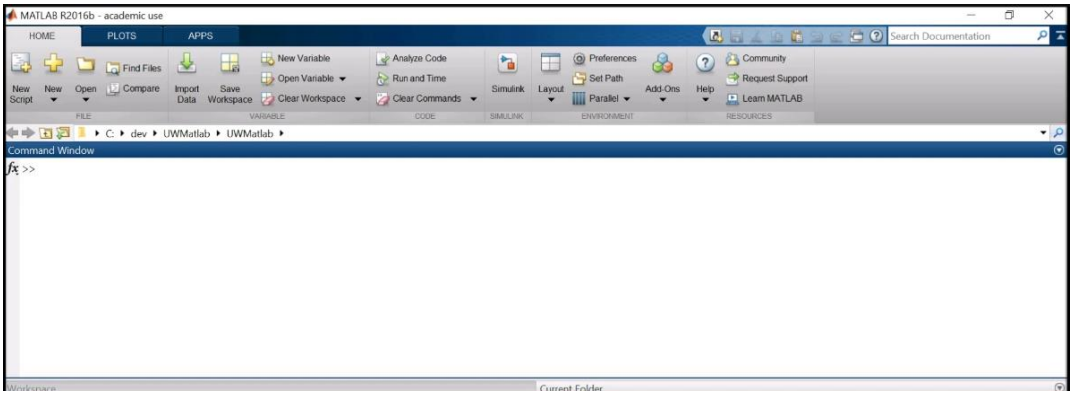
Key Features of Simulink:

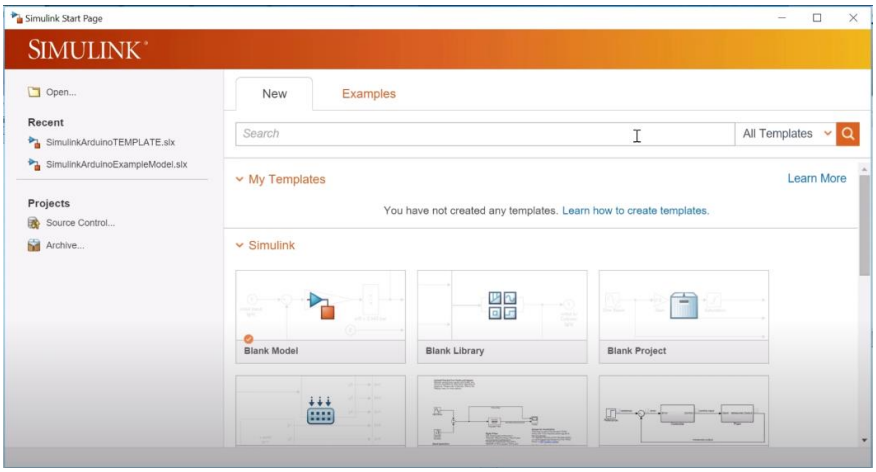
- Block Diagram Interface: Instead of writing code, you build models by connecting blocks that represent mathematical operations, system components, or functions.
- Simulation: You can simulate the behavior of your system over time and visualize how it responds to different inputs.
- Integration with MATLAB: Simulink works seamlessly with MATLAB, allowing you to use MATLAB scripts, functions, and data within your models.
- Code Generation: You can generate C, C++, HDL, or PLC code from your models, which is especially useful for deploying on hardware (e.g., microcontrollers, FPGAs).

Example Use Cases:

- Designing a PID controller for an industrial system
- Simulating a DC motor and its control system
- Testing a communication system model before hardware implementation

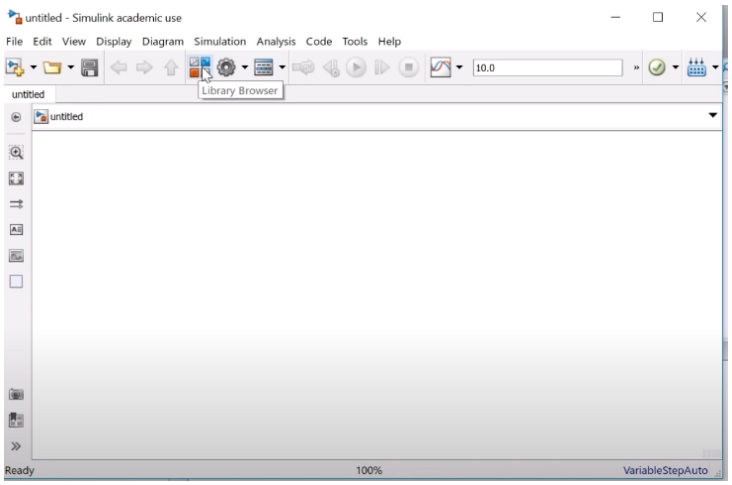
To open Simulink, on the ‘Home’ tab, click on ‘Simulink’.



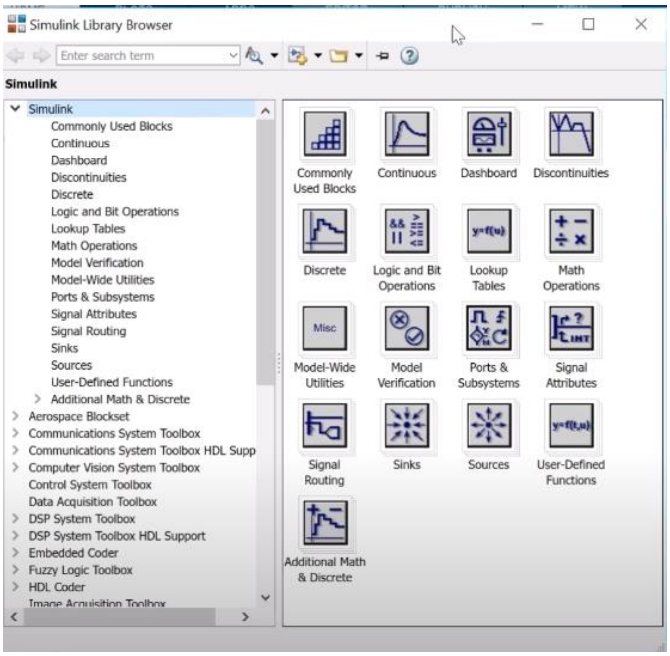


Prepared by: Aseer Imad Keats, Asif Newaz and Salman Hossain

When Simulink launches, the **Start Page** will appear. Click on **Blank Model** to begin a new project.



To add blocks: Click on the **Library Browser** or Double-click anywhere within the blank model space to search for the block you need.

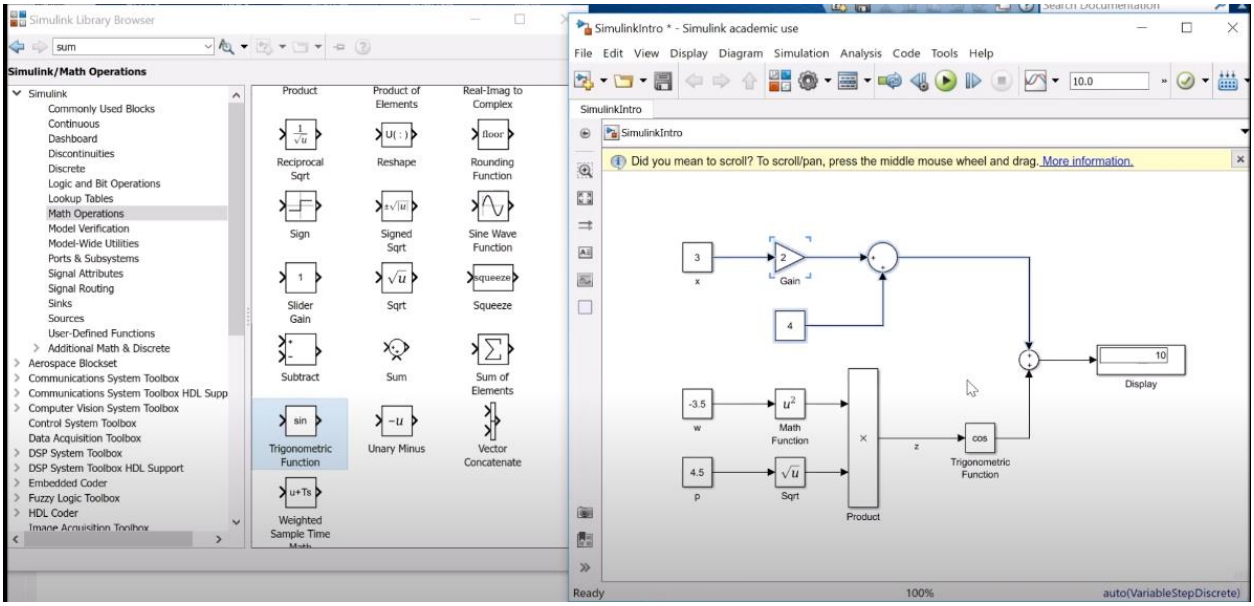


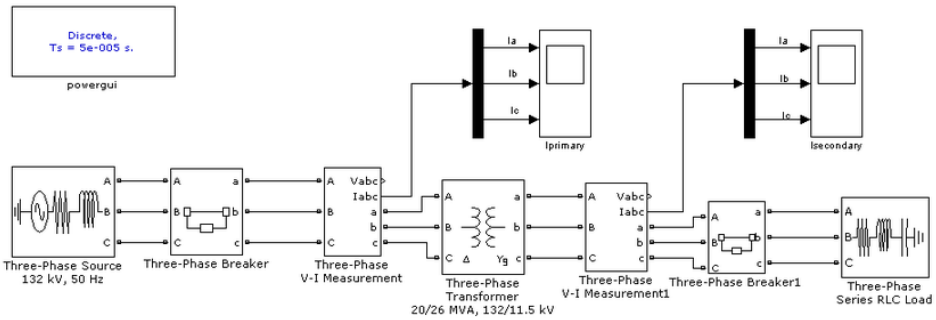
The Library Browser in Simulink provides a vast collection of blocks and toolboxes, particularly useful for modeling and simulating systems in electrical and electronics engineering. Some of the most relevant categories include:

- Simulink Standard Blocks – Essential for signal generation, math operations, logic control, and system visualization (e.g., Sine Wave, Scope, Gain, Switch).
- Simscape – Includes resistors, capacitors, inductors, diodes, and switches for simulating electrical and electronics circuits.
- Electrical Toolbox – Voltage/current sensors, meters, and circuit breakers.
- Power Systems Toolbox – Provides tools for simulating three-phase systems, transformers, transmission lines, power electronics, and grid components.
- Control System Toolbox – Modeling feedback systems in power electronics and automation.

- Signal Processing Toolbox – For filtering, signal transformation, and analysis in applications like power quality monitoring and fault detection.
- Simscape Electrical Specialized Power Systems – Ideal for detailed modeling of power generation, distribution networks, renewable energy systems (e.g., solar, wind), and motor drives.
- Measurement and Instrumentation Blocks – Includes ammeters, voltmeters, scopes, and other tools for monitoring voltage, current, and power in simulated systems.

These toolboxes allow electrical and electronics engineers to simulate everything from **basic analog circuits** to **complex power systems**, **motor control systems**, and **renewable energy integration**, all in a virtual environment before actual hardware implementation.





You can build and simulate complete electrical systems, such as transmission lines, within Simulink. The simulation environment allows data collection, visualization, and in-depth analysis, making it a powerful tool for modeling, testing, and verifying complex systems in a virtual environment.

Prepared by: Aseer Imad Keats, Asif Newaz and Salman Hossain