Course: Regulation and control

Lecture 15: Simulation of Mechanical and Electrical systems on MATLAB and Simscape

Lecturer: Chalachew Werku

Why Simulation?

- **The Challenge:** Control theory can be abstract.
- **The Solution:** Simulation!
- **Key Benefits:**
  - 🎯 Visualize complex system behavior.
  - ✅ Validate theoretical predictions.
  - ⚡ Iterate and optimize designs rapidly.
  - 💰 Save costs on physical prototypes.
  - 🔒 Test safely in all conditions.

Our Toolkit: MATLAB, Simulink, Simscape

- **MATLAB:** The Computational Engine
  - For: Mathematical modeling, analysis, and plotting.
  - Analogy: The command center for calculations.
- **Simulink:** The Visual Modeler
  - For: Block diagram simulation of dynamic systems.
  - Analogy: A digital whiteboard for system diagrams.
- **Simscape:** The Physical Modeler
  - For: Modeling physical systems (mechanical, electrical) using components.
  - Analogy: A virtual lab bench with real parts.

Environment Tour: MATLAB

- **Primary Interface:** The Command Window
  - Type commands and see immediate results.
- **Key Areas**:
  - **Workspace:** Shows your variables.
  - **Current Folder:** Manages your files.
  - **Editor:** For writing and saving scripts (.m files).
- **For Control Systems:** We use functions like tf( ), step( ), and bode( ).

Environment Tour: MATLAB

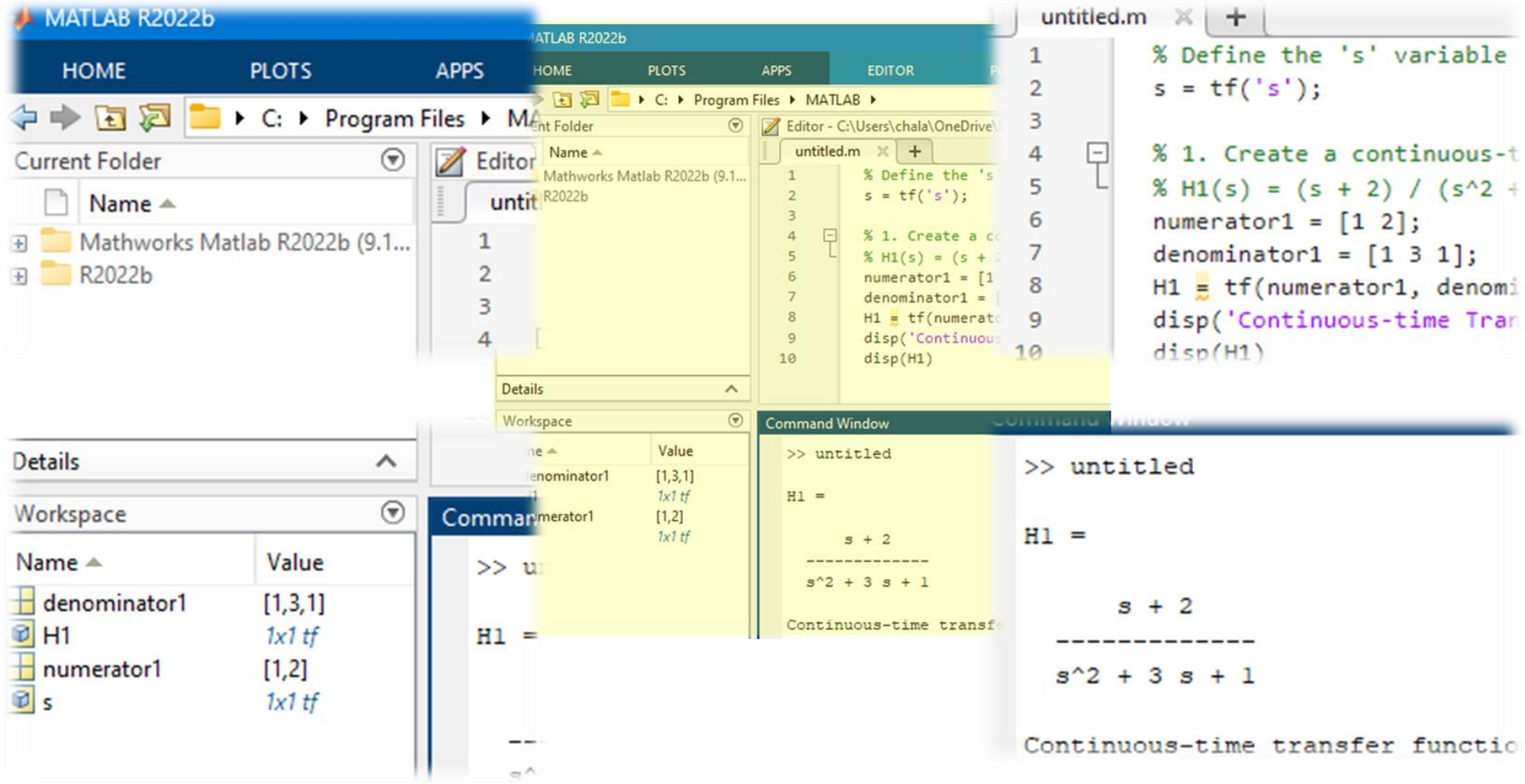


Fig 1: MATLAB Environment [1]

MATLAB workspace - Command Window vs. Editor

Command Window: The "Quick Calculator" / Experiment Zone

- **Purpose:** For executing commands one at a time and seeing immediate results.
- **Best For:**
  - Testing a single function (`step(G)`).
  - Quick calculations (`a = 5+3`).
  - Exploring and debugging.
- **Key Trait: It's ephemeral.** If you close MATLAB, your command history is saved, but the variables and their order of execution can be lost.

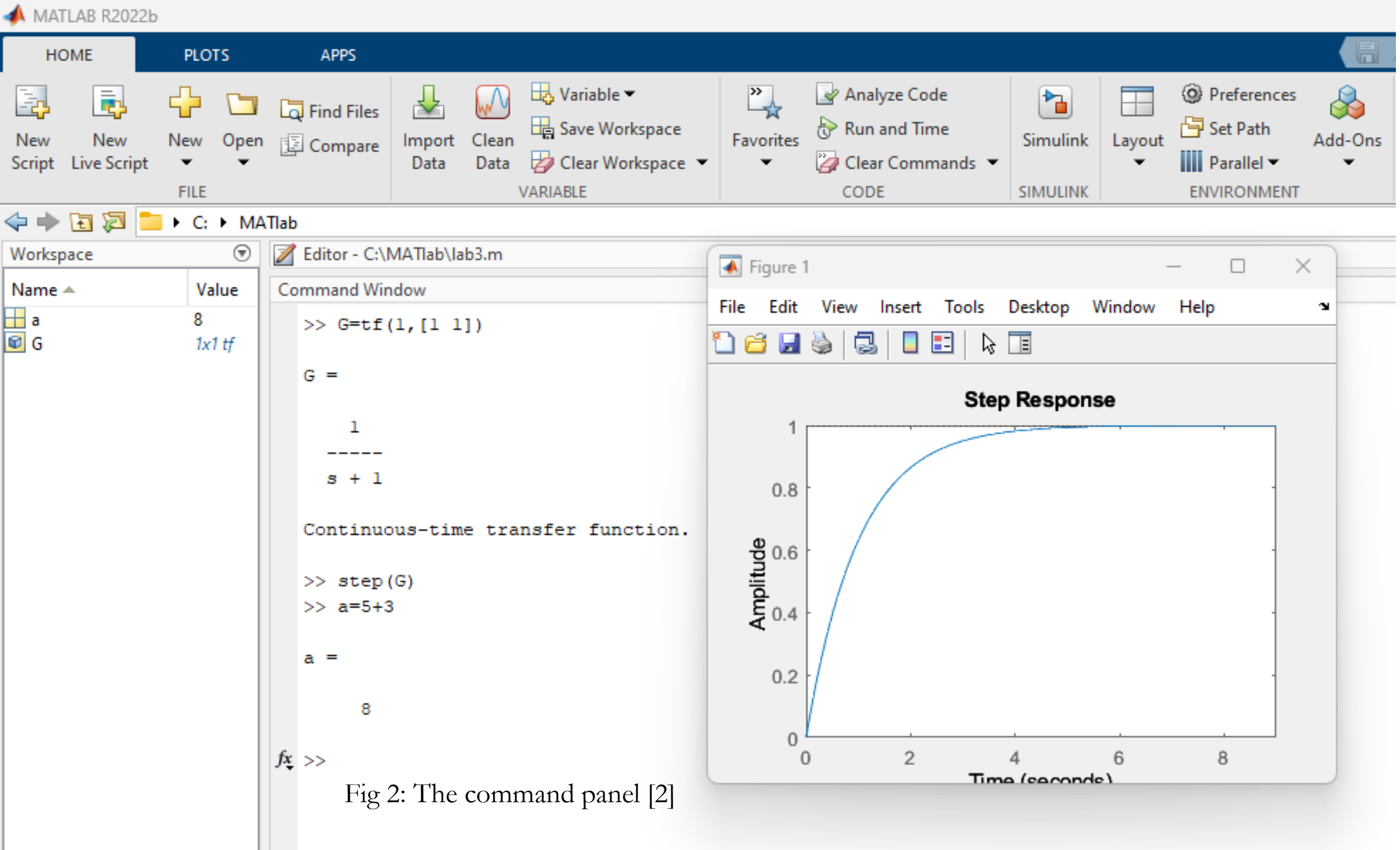


Fig 2: The command panel [2]

MATLAB workspace - Command Window vs. Editor

Editor: The "Script Writer" / Project Zone

- **Purpose:** For writing and saving a sequence of commands in a script file (.m file).
- **Best For:**
  - Writing labs, assignments, and complex procedures.
  - Ensuring reproducibility—you can run the entire script again with one click.
  - Keeping a permanent record of your work.
- **Key Trait:** It's persistent. You save your code, share it, and run it anytime.

MATLAB workspace - Command Window vs. Editor

Summary: When to Use Which?

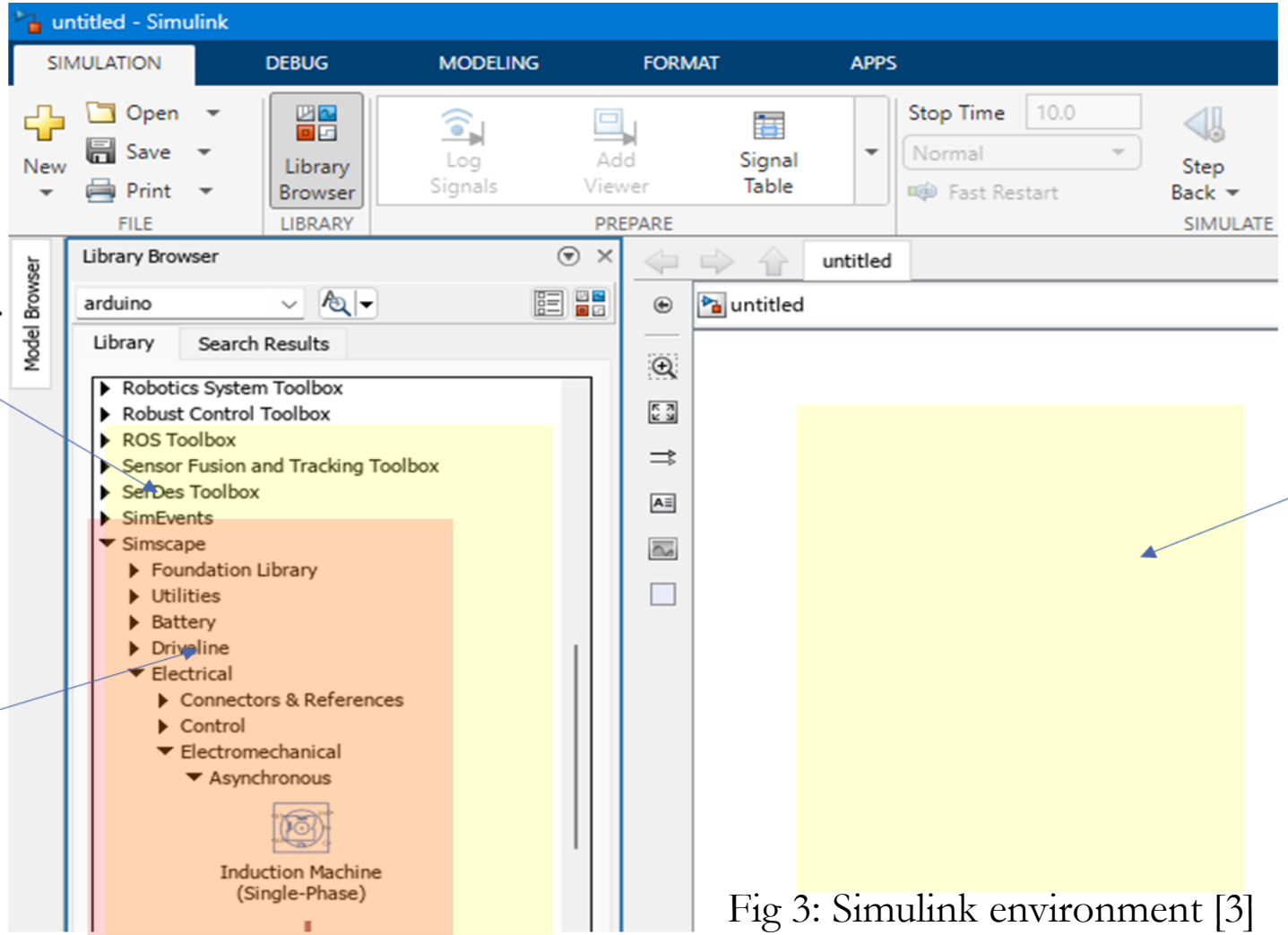
Use the Command Window for...	**Use the Editor for...**
• Quick tests & exploration	• Any graded work or labs
• Checking a variable's value	• Complex, multi-step analysis
• One-off calculations	• Work you need to save and re-run

Pro Tip: You can (and should!) use them together. Test lines in the Command Window, then copy the successful ones into your Editor script.

Environment Tour: Simulink & Simscape

- **Simulink Interface:**
  - **Library Browser:** The toolbox of all available blocks.
  - **Model Canvas:** The white space where you build your diagram.
- **Simulink Philosophy**: Connect input, output, and operation blocks.
- **Simscape**: Found inside the Simulink Library Browser.
  - Specialized libraries for Physical Domains: Mechanical, Electrical, Hydraulic.

Environment Tour: Simulink & Simscape



Library browser

Simscape

Model canvas

Fig 3: Simulink environment [3]

Plotting: Simulink vs. MATLAB Command

Simulink: Real-Time Visualization

Process:

- Drag a Scope block from the library.
- Connect it to any signal wire in your model.
- Run the simulation.
- Double-click the Scope to see the plot.

Best For:

- Debugging a model in real-time.
- Quickly checking signal integrity.
- Interactive, "hands-on" simulation.

Limitations:

- Limited plot customization (colors, labels, etc.).
- Data is "trapped" inside the Scope.

- **How it works:** Use Scope blocks connected directly to your signals.
- **Philosophy:** "See the data as it happens" during simulation.

Plotting: Simulink vs. MATLAB Command

MATLAB Command: Post-Processing & Analysis

Process:

- Use a To Workspace block or Data Logging.
- Run the simulation.
- Use the plot function in the Command Window on the saved data.

Best For:

- Creating publication-quality plots.
- Overlaying multiple simulation results.
- Performing calculations on the results before plotting.

Limitations:

- Requires extra steps (logging data).
- Not real-time.

- **How it works:** Send Simulink data to the MATLAB Workspace, then plot using functions like plot( ).
- **Philosophy:** "Analyze and present the data after the simulation."

Plotting: Simulink vs. MATLAB Command

Summary: Which One Should You Use?

Use a **Simulink Scope** when...	Use the **MATLAB Command** when...
• Debugging your model live. • You need a quick, simple look at a signal. • The simulation is running interactively.	• You need **custom, high-quality plots** for a report or presentation. • You want to **overlay multiple runs** or different signals. • You need to **perform calculations** on the data before plotting.

Pro Tip: You can use both! Use Scopes to debug during development, then use Workspace logging and MATLAB plotting to create your final graphs for analysis and reports.

Lab Series Overview

- Our hands-on journey through control theory:
  - Model Physical Systems (Simscape)
  - Create Mathematical Models (MATLAB and Simulink)
  - Analyze Time & Frequency Response (MATLAB)
  - Assess Stability (MATLAB)
  - Design Controllers (Simulink)

Lab 1 - Physical Modeling (Weeks 1-2 Theory)

- **Theoretical Concept:** Modeling Spring-Mass-Damper & RLC Circuits.
- **Lab Goal:** Build these systems using physical components.
- **Tool:** Simscape
- **What we'll do**:
  - Drag and drop physical blocks (springs, resistors, etc.).
  - Apply a step input force/voltage.
  - Observe the natural response.

Lab 1 - Physical Modeling (Weeks 1-2 Theory)

The model

- Type Simulink on MATLAB command or use Simulink icon on home button of the MATLAB window.

1. Simulink→ Blank Model→ Library Browser→ Simscape→ Foundational Library→ Mechanical→ Translational Elements→
2. Drag and drop Mass, Translational Damper and Translational spring and Mechanical translational Reference.
3. Connect them as shown and you can change their value just by double clicking the element

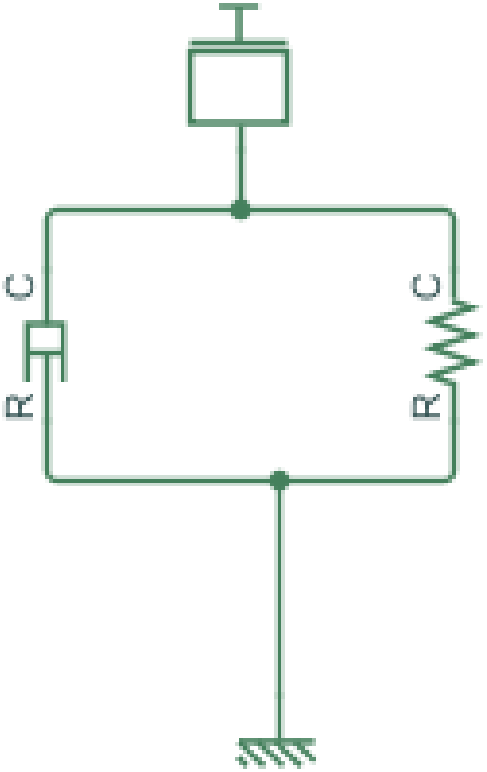


Fig 4: mass-damper-spring on simscape [4]

Lab 1 - Physical Modeling (Weeks 1-2 Theory) mechanical

adding inputs

1. Library Browser→ Simulink→ Sources→ Step

- Adjust step time for step block to 0,
- Here we can not connect the model to the input directly we need an interface add "Simulink-PS Converter" on the input side and "Ideal Force Source" on the model side

2. Library Browser→ Simscape→ Foundational Library→ Utilities→ Network Couplers→ Simulink-PS Converter

3. Foundational Library→ Mechanical→ Mechanical sources→ Ideal Force Source

Lab 1 - Physical Modeling (Weeks 1-2 Theory) mechanical

adding scope

1. Library Browser→ Simulink→ Sinks→ Scope

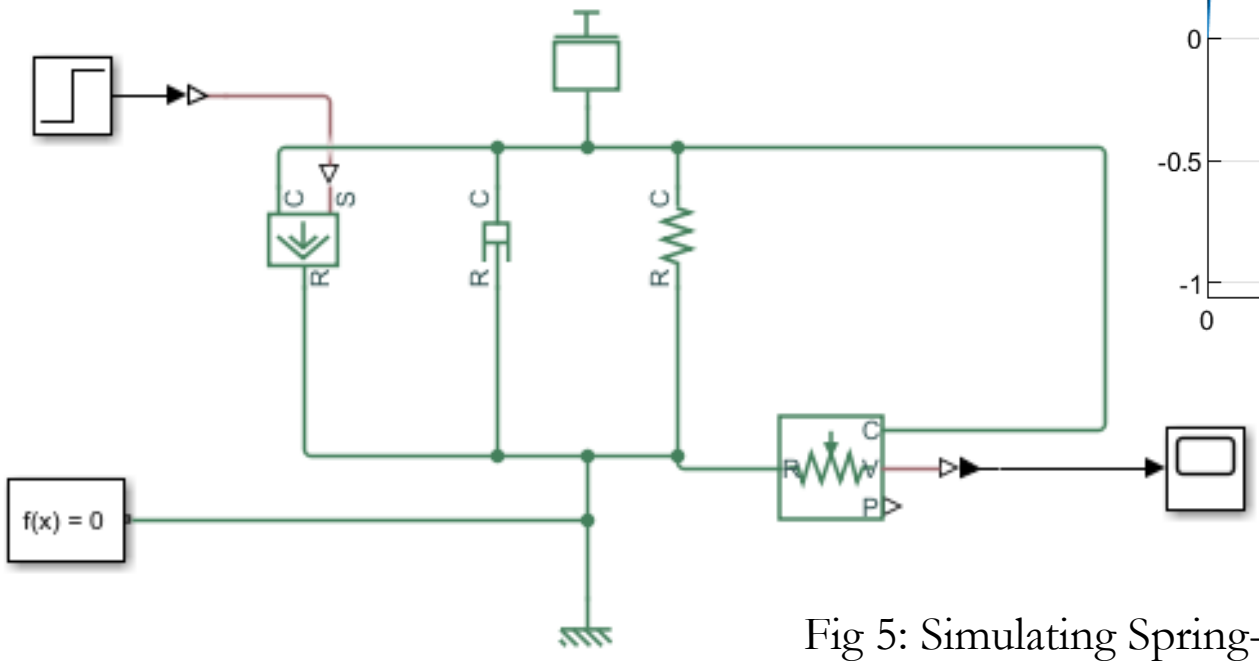
- We need an interface here also, add "PS-Simulink Converter" on the scope side and "Ideal Translational Motion Sensor" on the model side.

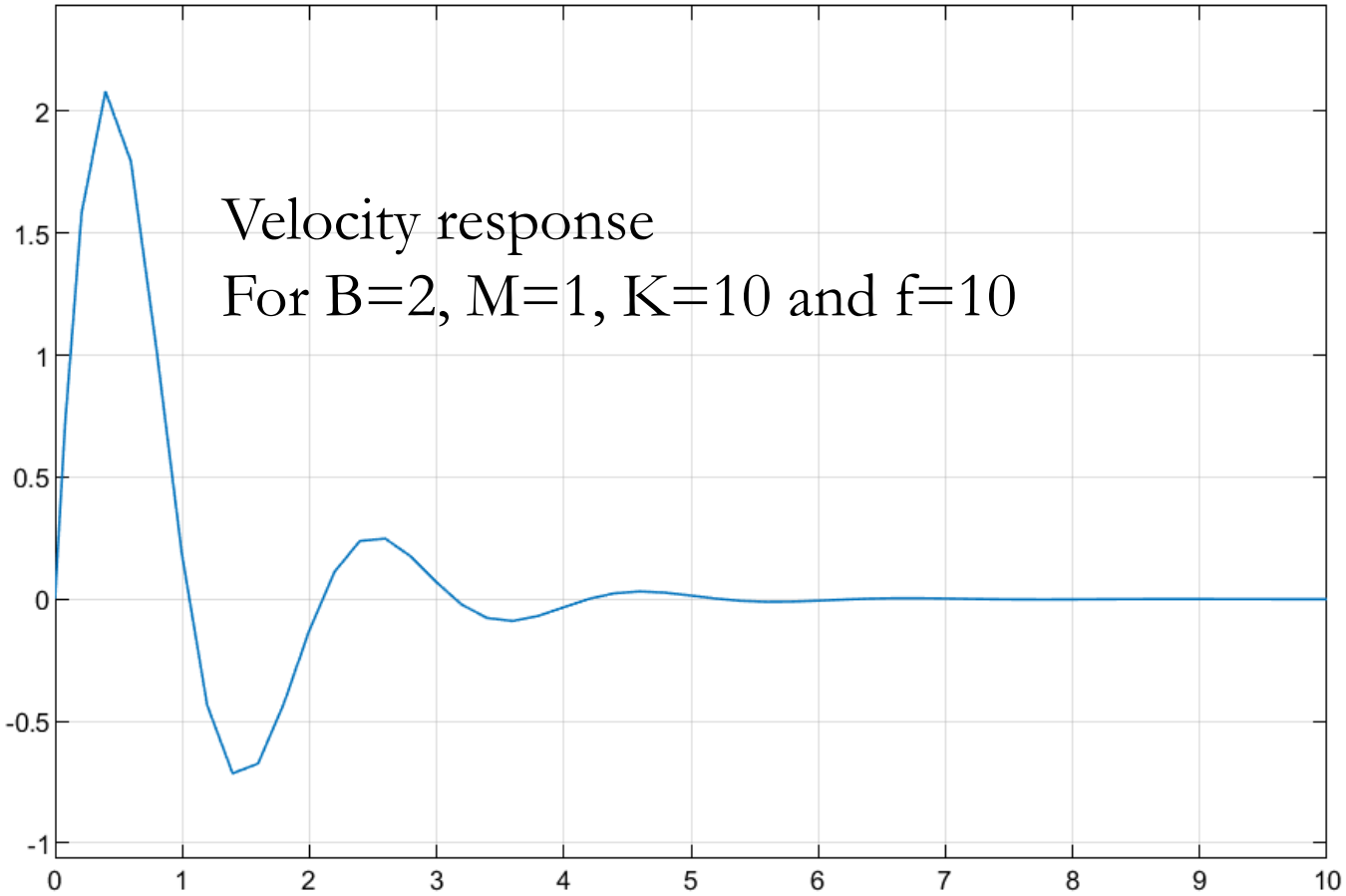
2. Library Browser→ Simscape→ Foundational Library→ Utilities→ Network Couplers→ PS-Simulink Converter

3. Foundational Library→ Mechanical→ Mechanical sensor→ Ideal Translational Motion Sensor

Finally you need to add Solver-configuration always

Double click the scope and press run button





Velocity response
For B=2, M=1, K=10 and f=10

Fig 5: Simulating Spring- mass- damper on simscape [5]

Lab 1 - Physical Modeling (Weeks 1-2 Theory) electrical

The model

1. Simulink→ Blank Model→ Library Browser→ Simscape→ Foundational Library→ Electrical→ Electrical Elements→
2. Drag and drop Inductor, Capacitor, Resistor and Electrical Reference.
3. Connect them as shown and you can change their value just by double clicking the element

Fig 6: R-L-C on simscape [6]

Lab 1 - Physical Modeling (Weeks 1-2 Theory) electrical

adding inputs and output

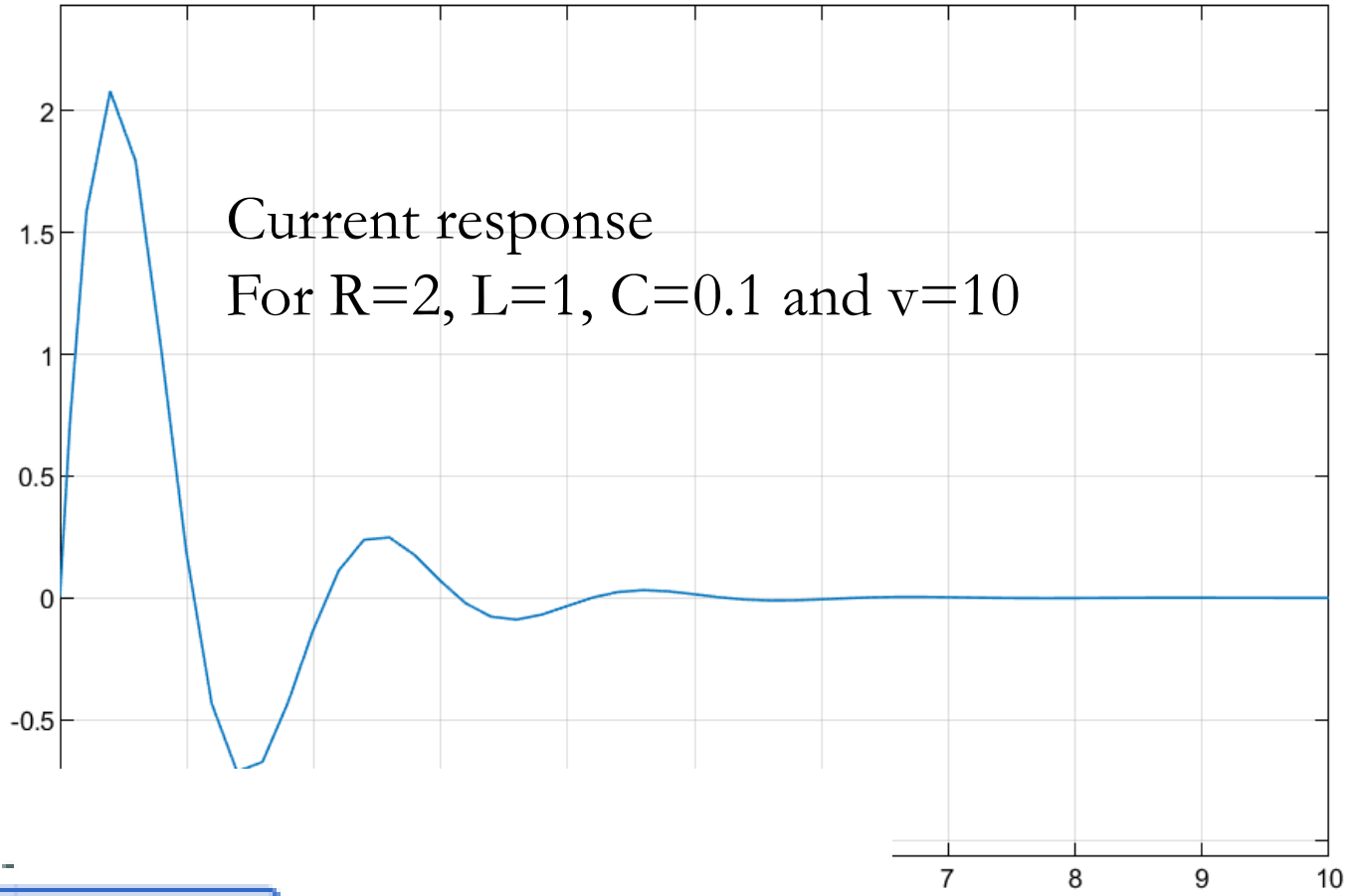
1. Foundational Library→ Electrical→ Electrical Elements→ Electrical sources→ DC Voltage Source

- Connect the scope via current sensor and the interface

2. Foundational Library→ Electrical→ Electrical Elements→ Electrical sensor→ Current → sensor

- Be sure to connect the sensor in series with the elements and finally connect solver configuration.

Double click the scope and press run button



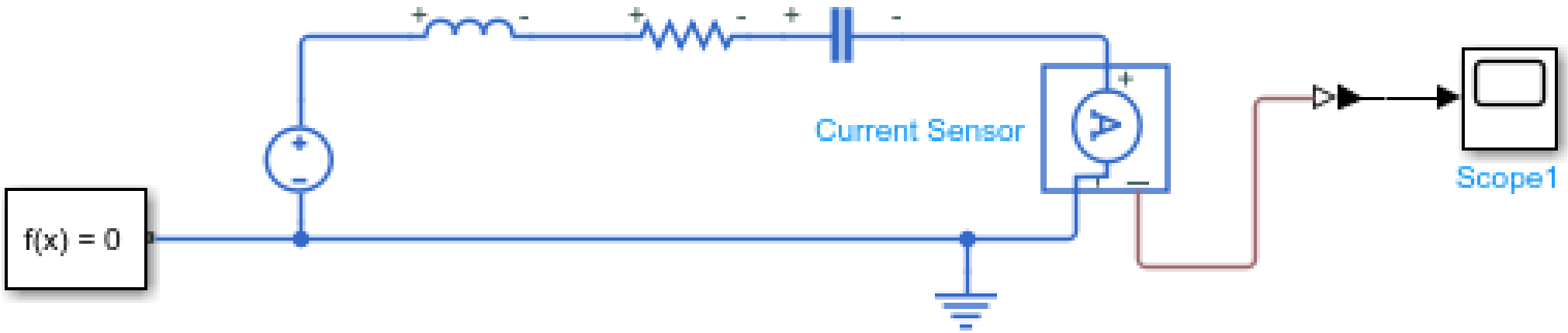


Fig 7: Simulating Spring- mass- damper on simscape [7]

Physical vs Transfer function

- We can get the same response if we use transfer function.

1. Simulink→ Continuous→ drag and drop Transfer Fcn block double click the block to adjust the values set numerator coefficients to [1 0] and denominator to [1 2 10] this represents $\frac{V(s)}{F(s)} = \frac{s}{s^2+2s+10}$
2. And directly connect step block to the input of Transfer Fcn and scope at the output.

Why Won't It Run? Troubleshooting Simscape

The Missing Solver Configuration

- **Error:** "This model does not have any Solver Configuration block."
- **Why:** Simscape needs this block to know how to solve the physics.
- **Fix:** Always add a Solver Configuration block from the Simscape Utilities library to the top level of your model.

Why Won't It Run? Troubleshooting Simscape

The Floating Node / Incomplete Circuit

- **Error:** "No connection made between port X and any other port." or "Initialization failure."
- **Why:** Physical domains need a reference (like electrical ground or mechanical reference). An unconnected port is a "floating node."
- **Fix:** Always add a reference block:
  - **Electrical:** Electrical Reference (Ground)
  - **Mechanical (Translational):** Mechanical Translational Reference
  - **Mechanical (Rotational):** Mechanical Rotational Reference

Why Won't It Run? Troubleshooting Simscape

The Unit Mismatch

- **Error:** Unexpected results, or errors about incompatible units.
- **Why:** You connected a rotational velocity (rad/s) output to a translational velocity (m/s) input.
- **Fix:** Use the correct converter block.
  - **Rotational to Translational:** Use a Lead Screw block.
  - **Physical Signal to Simulink Signal:** Use a PS-Simulink Converter.
  - **Simulink Signal to Physical Signal:** Use a Simulink-PS Converter.

Why Won't It Run? Troubleshooting Simscape

The Impossible Initial Condition

- **Error:** Simulation fails during initialization or immediately.
- **Why:** The solver can't find a consistent starting state. Example: two rigidly connected mechanical sources fighting each other, or a capacitor with an initial voltage that violates Kirchhoff's laws.
- **Fix:**
  - Check for conflicting sources.
  - Use appropriate initial conditions in blocks like springs, capacitors, and masses.

Quick-Fix Checklist

- ✅ Solver Configuration block added?
- ✅ Reference/Ground block connected?
- ✅ PS-Simulink Converters used for Scopes?
- ✅ Simulink-PS Converters used for Sources?
- ✅ Initial conditions make physical sense?

How to Compare Plots on the Same Graph

MATLAB

Command	What it does	When to use it
hold on	**Locks** the current plot for adding more lines.	When you want to **compare** multiple responses on the same graph.
hold off	**Unlocks** the plot (default state).	After you finish building a multi-line plot, to clean up for the next one.

- **Good Practice:** Use hold off after you are done adding to a plot to avoid accidentally cluttering future figures.

How to Compare Plots on the Same Graph

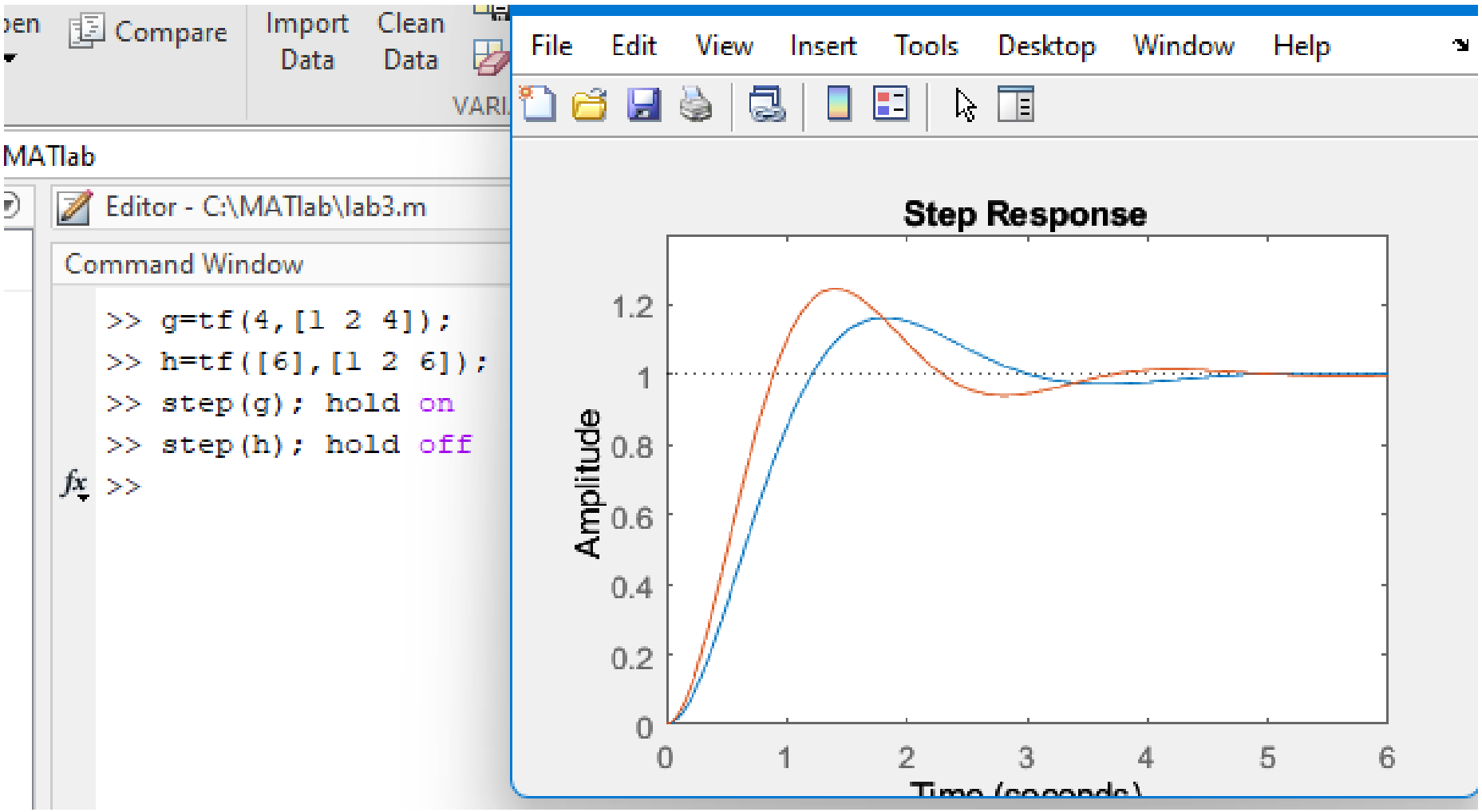


Fig 8: Comparing plots on MATLAB [8]

Mathematical Models & Time Response (Weeks 3-4 Theory)

- **Theoretical Concept:** Transfer Functions, State-Space, Step Response.
- **Lab Goal:** Convert the physical model into math and analyze its transient response.
- **Tool:** MATLAB & Simulink
  - What we'll do:
  - Define a Transfer Function. `tf` and `zpk`
  - Use `step()` to simulate the response.
  - Measure **Rise Time** and **Settling Time**.

Mathematical Models & Time Response (Weeks 3-4 Theory) functions

$$G(s) = \frac{s+2}{s^2+5s+6}$$

- Tf : `G=tf([1 2] , [1 5 6]);`
- Zpk: `G=zpk([-2], [-2 -3], [1]);`
- If you run the code on the right you will get performance metrices for G.

```
S = stepinfo(G);
disp('Step Response Characteristics:');
fprintf('Rise Time: %.3f s\n', S.RiseTime);
fprintf('Settling Time: %.3f s\n',
S.SettlingTime);
fprintf('Overshoot: %.2f%%\n', S.Overshoot);
fprintf('Peak: %.3f\n', S.Peak);
fprintf('Peak Time: %.3f s\n', S.PeakTime);
```

The MATLAB "Fresh Start" Command

Why clc; close all; clear;?

- This command is used at the top of scripts to reset the MATLAB environment, ensuring your code runs consistently every time.

clear

- **What it does:** Erases all variables from the Workspace.
- **Why it's important:**
  - Prevents old variables from interfering with your new script.
  - Avoids using stale or incorrect data from a previous run.
  - Without it: Your script might use a variable G from yesterday, leading to confusing, incorrect results.

The MATLAB "Fresh Start" Command
close all

- **What it does:** Closes all open Figure windows (plots).
- **Why it's important:**
  - Prevents a cluttered desktop with dozens of old plots.
  - Ensures new plots (like step or bode) are created in a fresh window, making them easy to find.
  - **Without it:** Your new step response plot might be hidden behind five old ones.

The MATLAB "Fresh Start" Command
clc

- **What it does:** Clears the Command Window.
- **Why it's important:**
  - Provides a clean output space for your new script's results, warnings, and disp commands.
  - Makes it easier to read the relevant output for your current task.
  - **Without it:** The output from your previous work clutters the window, making it hard to see what's happening now.

Lab 3 - Stability Analysis (Weeks 5-7 Theory)

- **Theoretical Concept:** System Poles, Routh-Hurwitz Criterion.
- **Lab Goal:** Determine if a system is stable and how its performance changes with gain.
- **Tool:** MATLAB
- **What we'll do**:
  - Plot poles and zeros with `pzplot( )`.

Frequency Domain Analysis (Weeks 8-9 Theory)

- **Theoretical Concept:** Bode Plots, Nyquist Plots, Stability Margins.
- **Lab Goal:** Analyze system performance in the frequency domain.
- **Tool:** MATLAB
- **What we'll do:**
  - Generate Bode and Nyquist plots. [`bode ( ) ,nyquist( )`]
  - Use `margin( )` to automatically calculate Gain and Phase Margin.

Lab 5 - Root Locus (Weeks 10-11 Theory)

- **Theoretical Concept:** How closed-loop poles move with changing gain.
- **Lab Goal:** Visualize the root locus and connect it to system performance.
- **Tool:** MATLAB
- **What we'll do:**
  - Plot the root locus with `rlocus( )`.

Lab 6 - Controller Design (Weeks 12-13 Theory)

- **Theoretical Concept:** PID Control, Lead/Lag Compensators.
- **Lab Goal:** Design a controller to improve system performance.
- **Tool:** Simulink and MATLAB
- **What we'll do:**
  - Build a feedback control loop with a plant and PID controller.
  - Tune the PID gains using Ziegler-Nichols rules.

Lab 6 - Controller Design (Weeks 12-13 Theory)

MATLAB

$G(s) = \frac{1}{s^3+6s^2+5s}$ an exercise from lecture 12

- With Zn we have got kp=18, ki=12.86, kd=6.3
- By using MATLAB
- Run the code below you will get kp=10.8, ki=1.79, kd=9.47

```
G(s)=tf([1],[1 6 5 0]);

C=pidtune(G,'PID')
```

•

Lab 6 - Controller Design (Weeks 12-13 Theory)

SIMULINK

Open Simulink

Drag and drop from Library Browser or just double click on the canvas and hit enter

- Step, sum, pid, Transfer fcn, scope
- Double click the sum you will get something like |++ change it to |+- we are making positive feedback into negative
- Connect the elements as shown in the next slide.

Lab 6 - Controller Design (Weeks 12-13 Theory)

SIMULINK

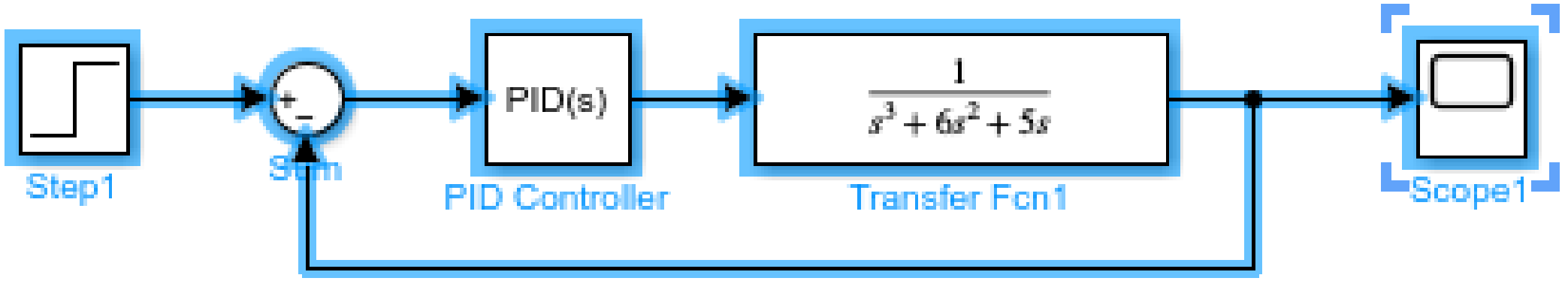
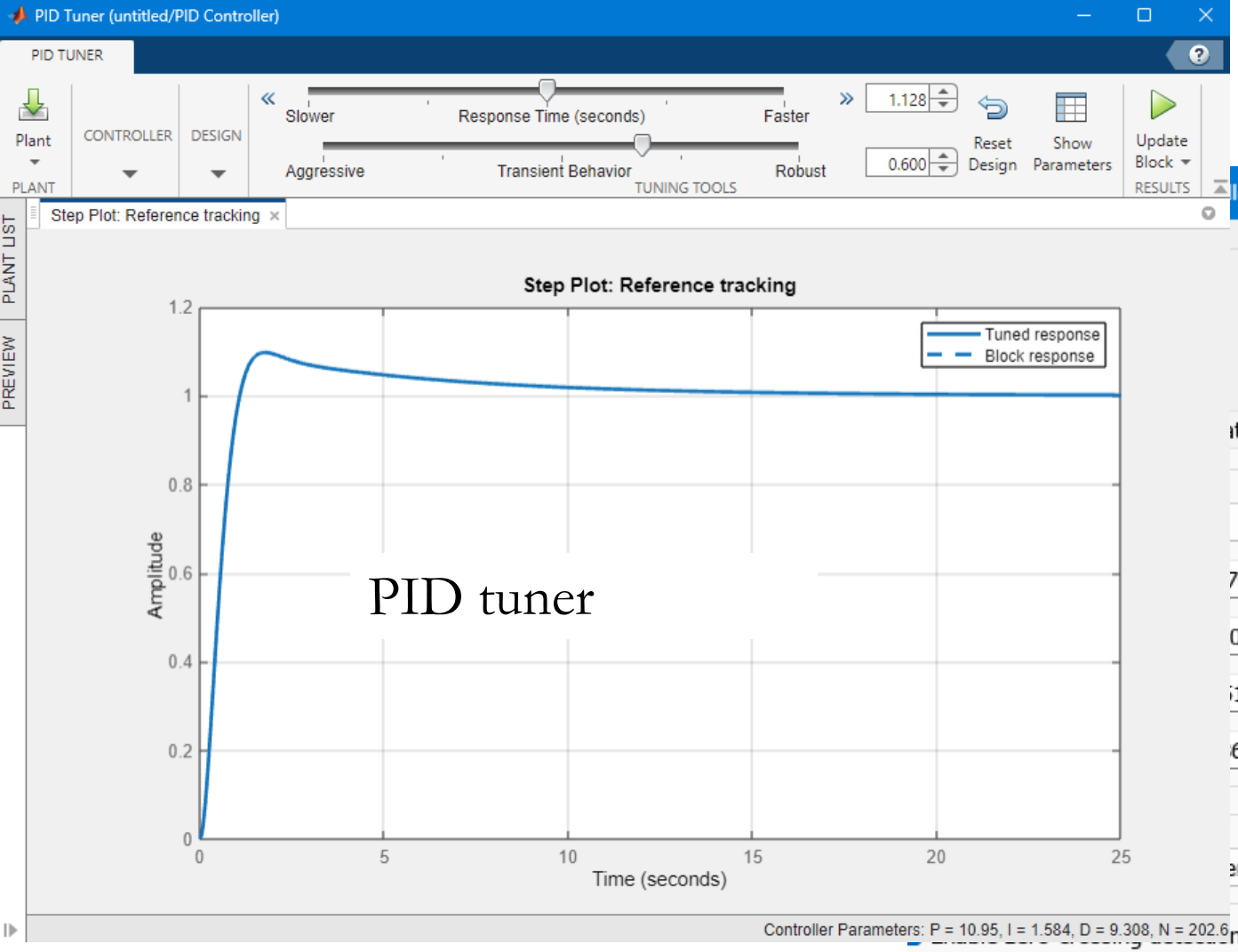


Fig 9: Block diagram on Simulink [9]

- Double click on the pid block→ scroll down and press tune→ PID tuner→ adjust speed vs robustness → update Block



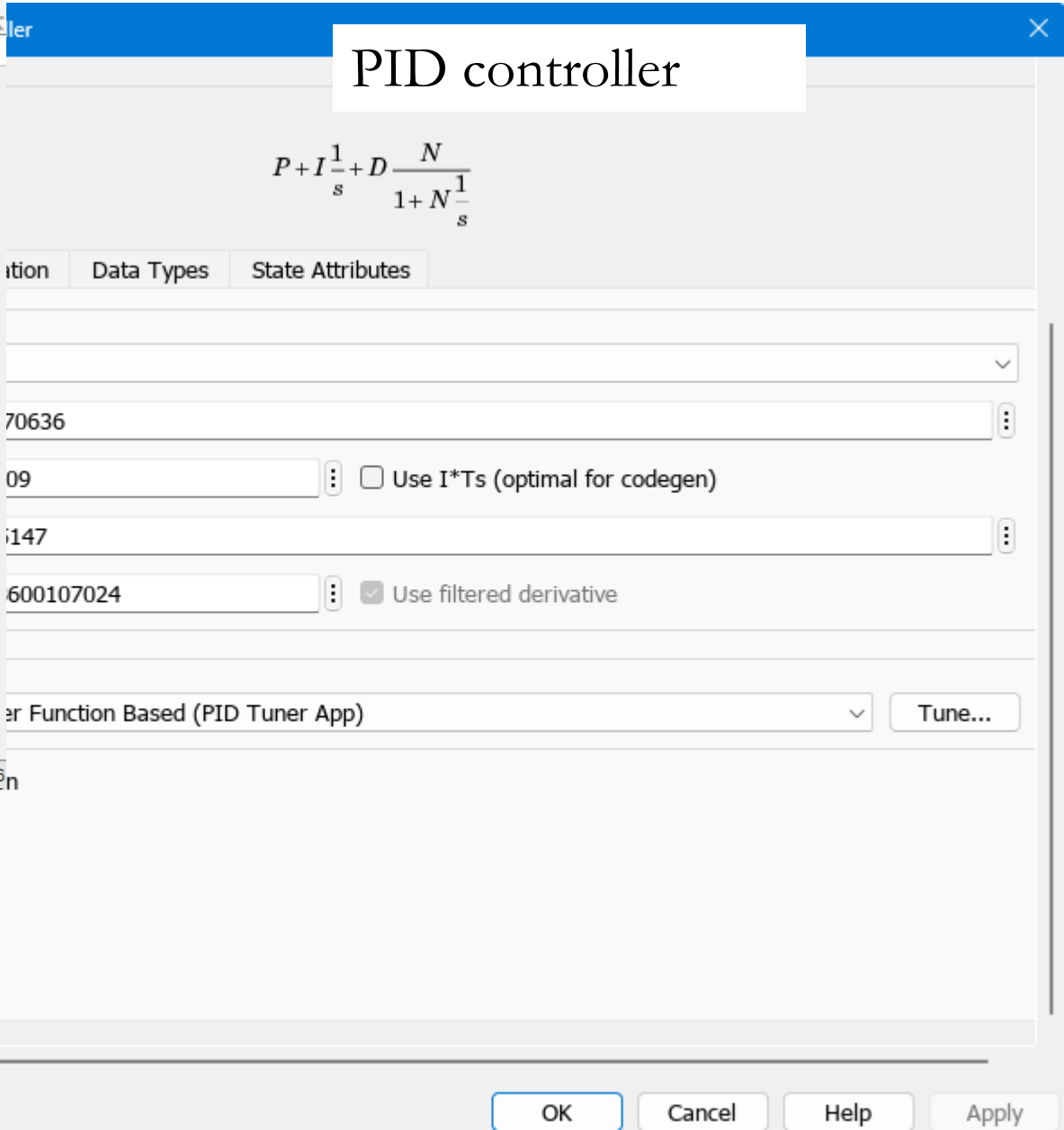


Fig 10: Simulink PID tuner [10]

Comparing the tuners

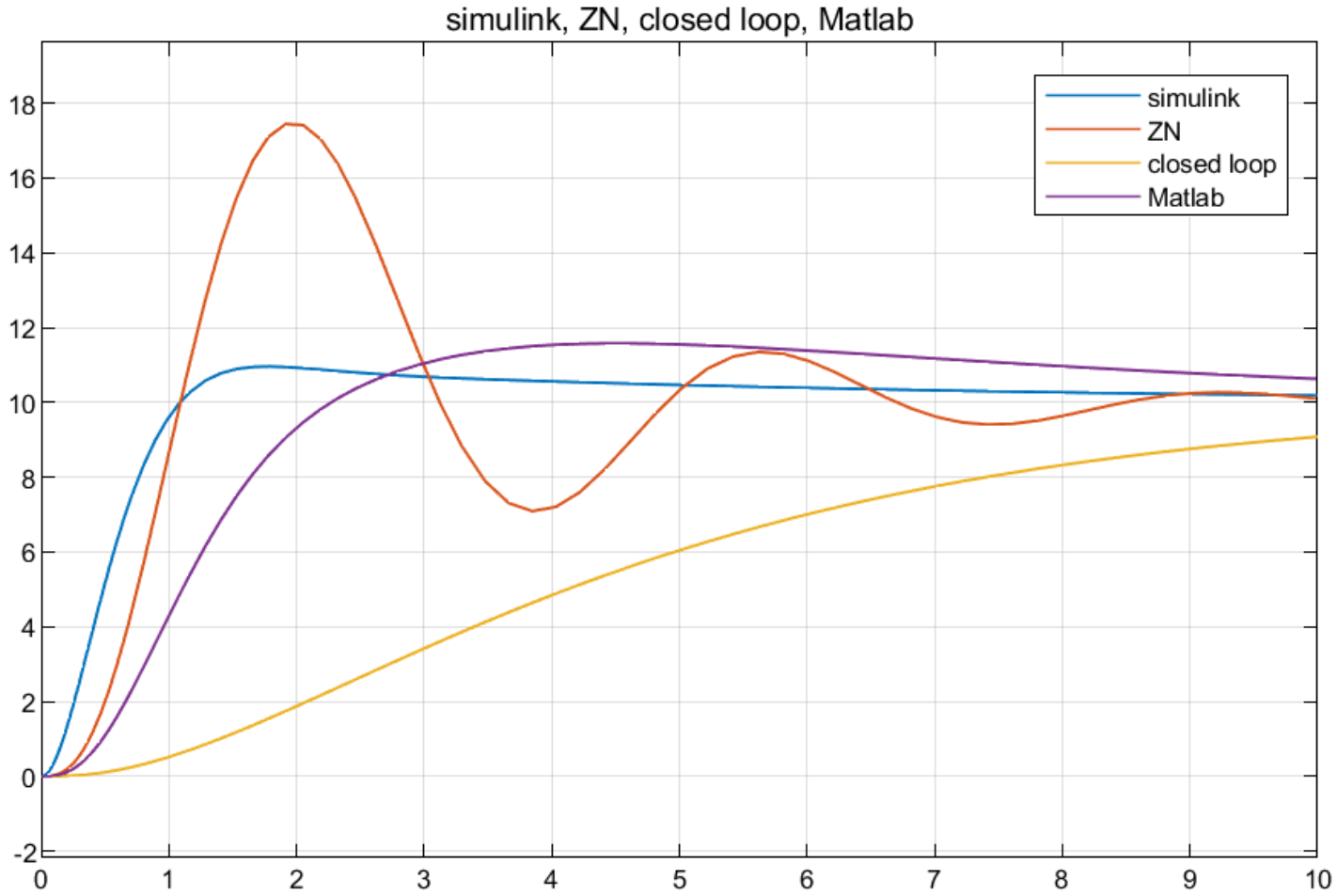
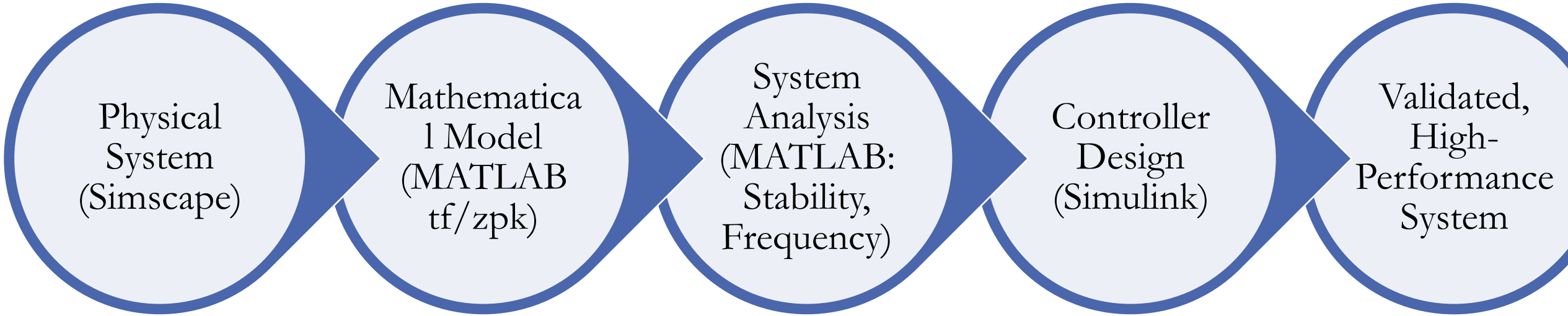


Fig 11: System response of a PID controller with different tuning methods [11]

The Complete Workflow



What You've Learned

- You can now transition a concept from a physical system to a working controller.
- **Skills Gained:**
  - Physical and mathematical modeling.
  - Comprehensive system analysis (Time & Frequency).
  - Practical controller design and tuning.

Conclusion & Next Steps

- Simulation is a powerful tool that makes theory tangible.
- These skills are directly applicable in industries like robotics, aerospace, and automation.
- Continue Exploring: Try new systems, more complex controllers, and real-world hardware integration.

References

[1] Chalachew Werku, 2025, MATLAB Environment, Self-created

[2] Chalachew Werku, 2025, The command panel, Self-created

[3] Chalachew Werku, 2025, Simulink environment, Self-created

[4] Chalachew Werku, 2025, mass-damper-spring on simscape, Self-created

[5] Chalachew Werku, 2025, Simulating Spring- mass- damper on simscape, Self-created

[6] Chalachew Werku, 2025, R-L-C on simscape, Self-created

[7] Chalachew Werku, 2025, Simulating Spring- mass- damper on simscape, Self-created

[8] Chalachew Werku, 2025, Comparing plots on MATLAB, Self-created

[9] Chalachew Werku, 2025, Block diagram on Simulink, Self-created

[10] Chalachew Werku, 2025, Simulink PID tuner, Self-created

[11] Chalachew Werku, 2025, System response of a PID controller with different tuning methods, Self-created

- MATLAB documentation
- Norman S. Nise – Control Systems Engineering (2015, Wiley)