LAB 7: Data-Driven Modeling of DC Servo Motor

Course Number	MENG 3020
Course Title	System Modeling & Simulation
Semester/Year	Fall 2024

Lab/Tutorial Report No.	Lab 7
Report Title	Data-Driven Modeling of DC Servo Motor
Section No.	0NA
Group No.	N/A
Submission Date	November 14, 2024
Due Date	

Student Name	Signature*	Total Mark
Michael McCorkell		49 / 50

LAB 7 Grading Sheet

Student First Name: Michael	**Student Last Name:** McCorkell
Part A: System Identification from Noise-free Data	15 **/15**
Part B: System Identification from Noisy Data	20 **/20**
Post Lab Assignment	9 **/10**
General Formatting: Clarity, Writing style, Grammar, Spelling, Layout of the report	5 **/5**
Total Mark	49 **/50**

LAB 7: Data-Driven Modeling of DC Servo Motor

OBJECTIVES

- To explore the system identification approach to model linear and nonlinear systems
- To understand the nonlinear characteristics of dynamic systems such as dead-zone, saturation and gear backlash and ensure that the system operates in the linear range.
- To apply system identification workflow to understand the importance of data collection, model selection, model fitting and model validation.

INTRODUCTION

System identification approach consists of associating a mathematical model with a measured experimental behavior. Here the system is considered as a **black box** with an input variable and an output variable. The idea is to input a known input variable to the system and to raise the output variable. We then need to find a mathematical model for the system transfer function that will yield the same relationship between input and output. This method provides only a model of the global behavior of the system, without delving into the separate influences of the various performance parameters.

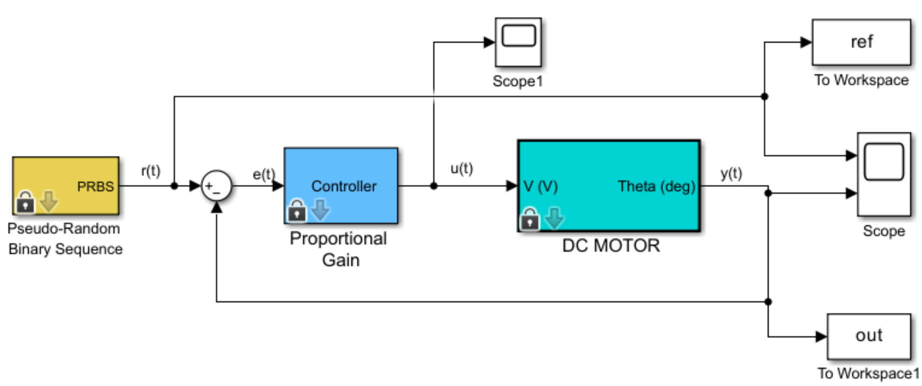
The method provides a model experimentally validated *without* having to write the equations that govern the behavior of the system. The model should be treated with caution in case non-linear phenomenon enter the actual system's behavior. In practice we always perform several tests corresponding to different constraints

Aside from simple cases, identification of the transfer function requires significant computational resources. The software tool will enable us to automate this task by suggesting mathematical models for identification processes that could be complex.

MATLAB provides a toolbox called "**System Identification**" which allows this process to be carried out. We can use *time-domain* and *frequency-domain* input-output data to identify *continuous-time* and *discrete-time transfer functions*, *process models*, and *state-space models*. The toolbox also provides algorithms for embedded online parameter estimation. System Identification Toolbox lets us create models from measured input-output data. We can (a) Analyze and process data, (b) Determine suitable model structure and order, and estimate model parameters, (c) Validate the model accuracy.

Data Driven Modeling of DC Motor

1. **Open** the Simulink model **DC_Motor_System.slx** which is compatible with your MATLAB version.



The model represents closed-loop position control of a DC motor under *proportional controller*.

The **reference signal** is a ***Pseudo-Random Binary Sequence (PRBS)*** signal.

The **output signal** is the ***angular position*** of the motor shaft in degrees.

2. Open the **Block Parameters** by *double-clicking* on the appropriate block. Enter the following values for each block:

a) Pseudo-Random Binary Sequence:

Number of Sample Points = 2000, Switch Constant = 80, Amplitude = 40 degrees

b) Proportional Gain:

Proportional Gain = 1

c) DC Motor:

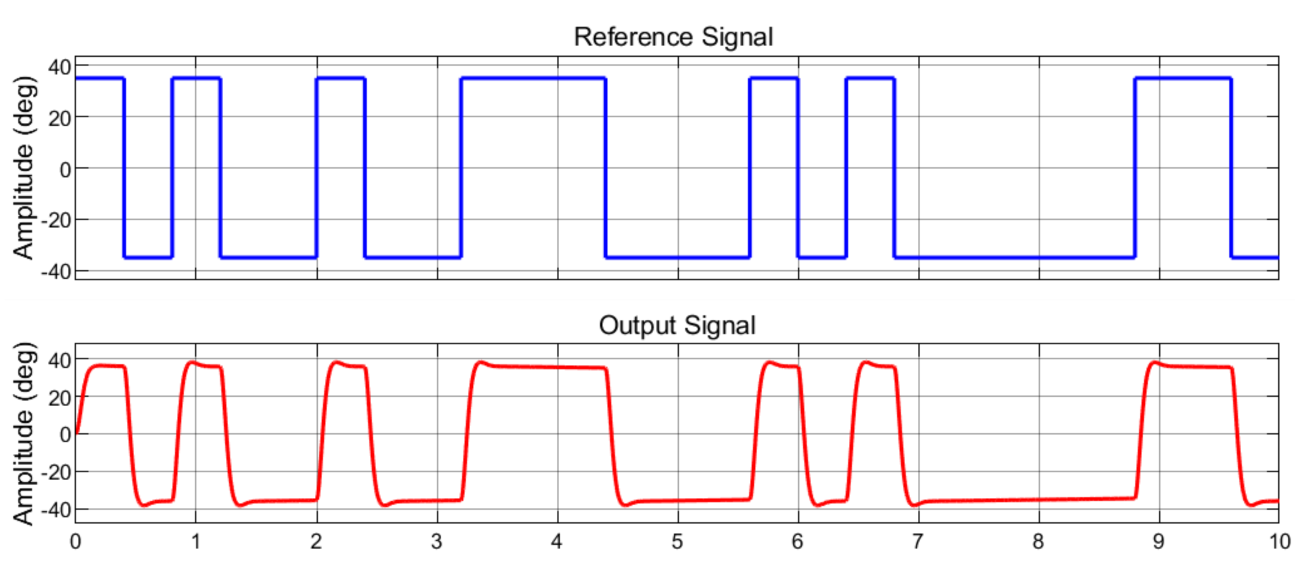
Armature Inductance = 0.15×10^{-3} H, Armature Resistance = 3 Ω,

Motor Inertia = 2.47×10^{-7} oz.In.sec^2, Motor Viscus Friction = 1.2319×10^{-7} N.m.sec,

Part A: System Identification from Noise-free Data

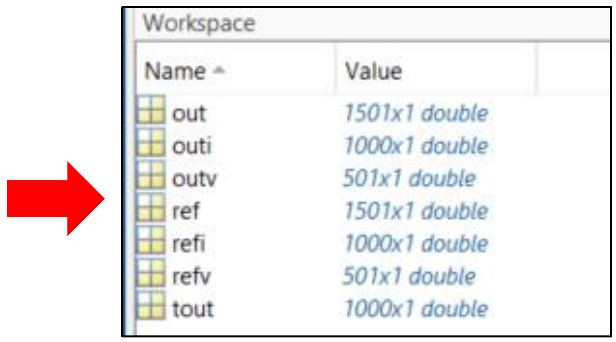
3. In DC Motor block set **Additive Noise = 0**.
4. **Run** the simulation for **10 seconds.** Open both **Scope** blocks. You will see the input-output data and the control signal. Ensure that the control signal is **not saturated**. Decrease the amplitude of input signal if required. Provide the graph below:

Noise-free Input-Output Data

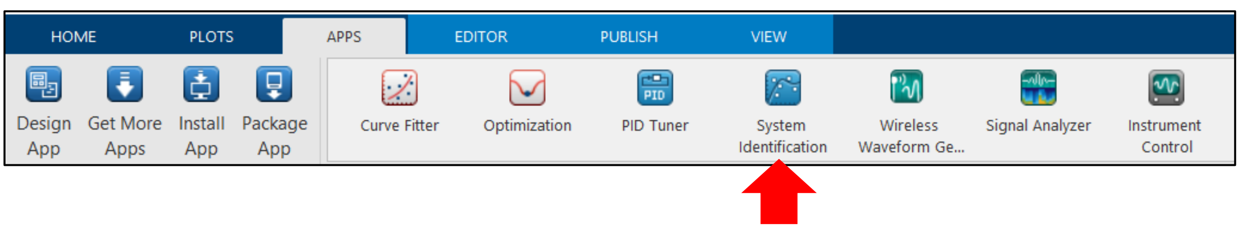


5. The input and output data will be saved in variables **ref** and **out** that are available in **MATLAB Workspace**.
6. **Run** the following code in **MATLAB** to create the **identification** and **validation** datasets in the **Workspace**.

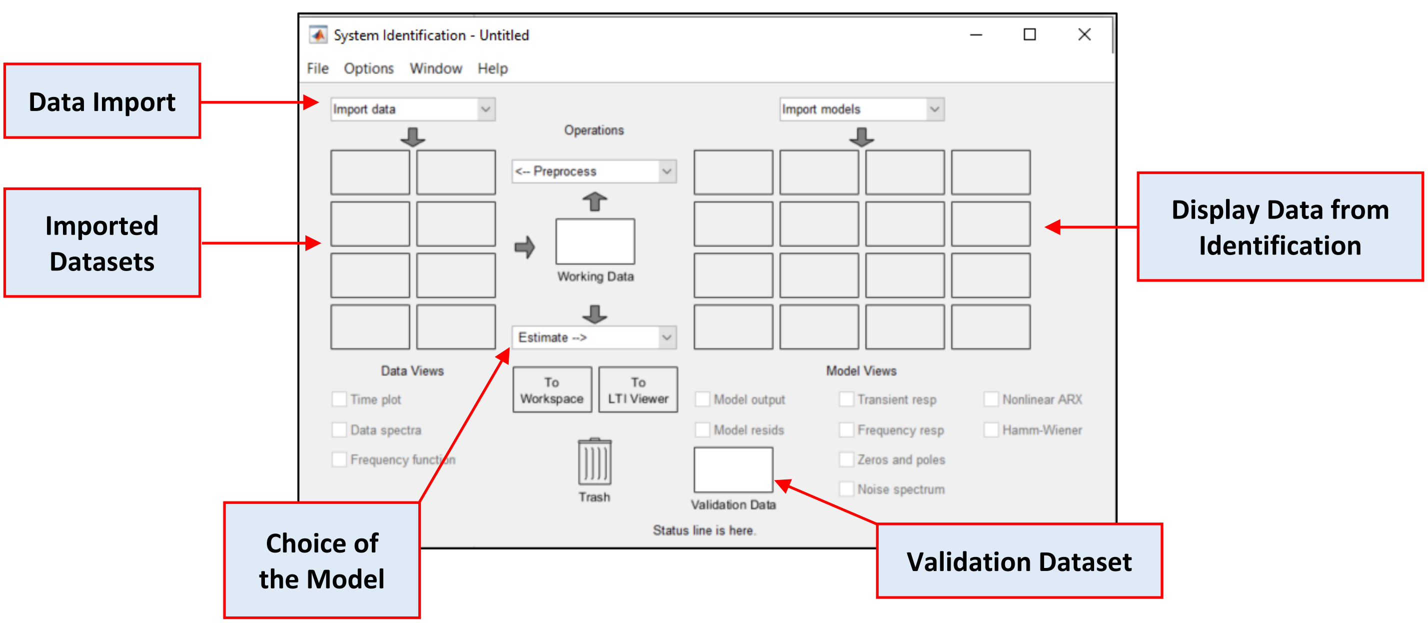
```
% identification dataset
outi = out(1:1000);
refi = ref(1:1000);

% Validation dataset
outv = out(1001:end);
refv = ref(1001:end);
```



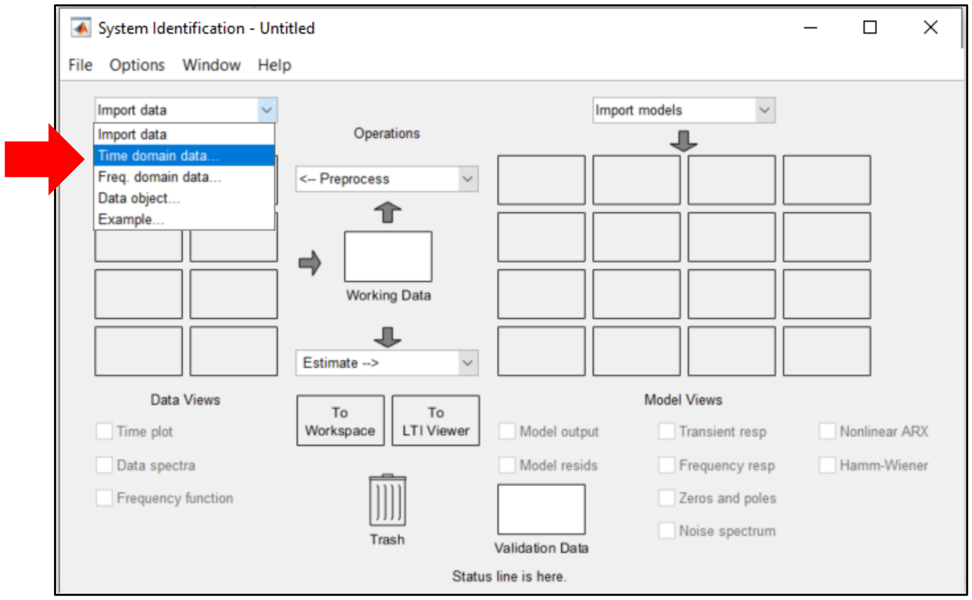
7. In **MATLAB Window**, **APPS** tab, click on the **System Identification** app to run the "**System Identification**" toolbox.



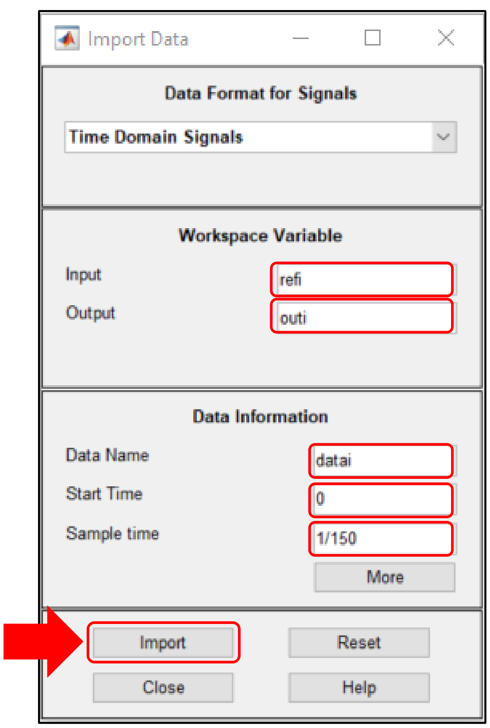
The **System Identification User Interface** window will be opened as below. It is possible to import data in Time-domain, Frequency-domain and Data object. In our example the data from experiment is a **Time-domain** data.

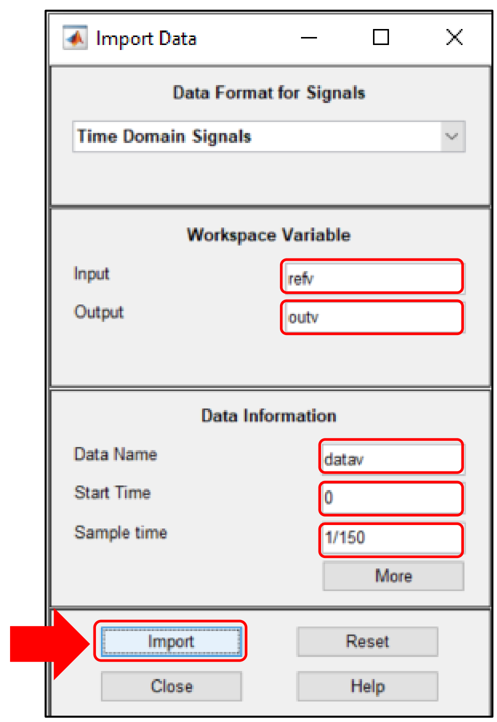


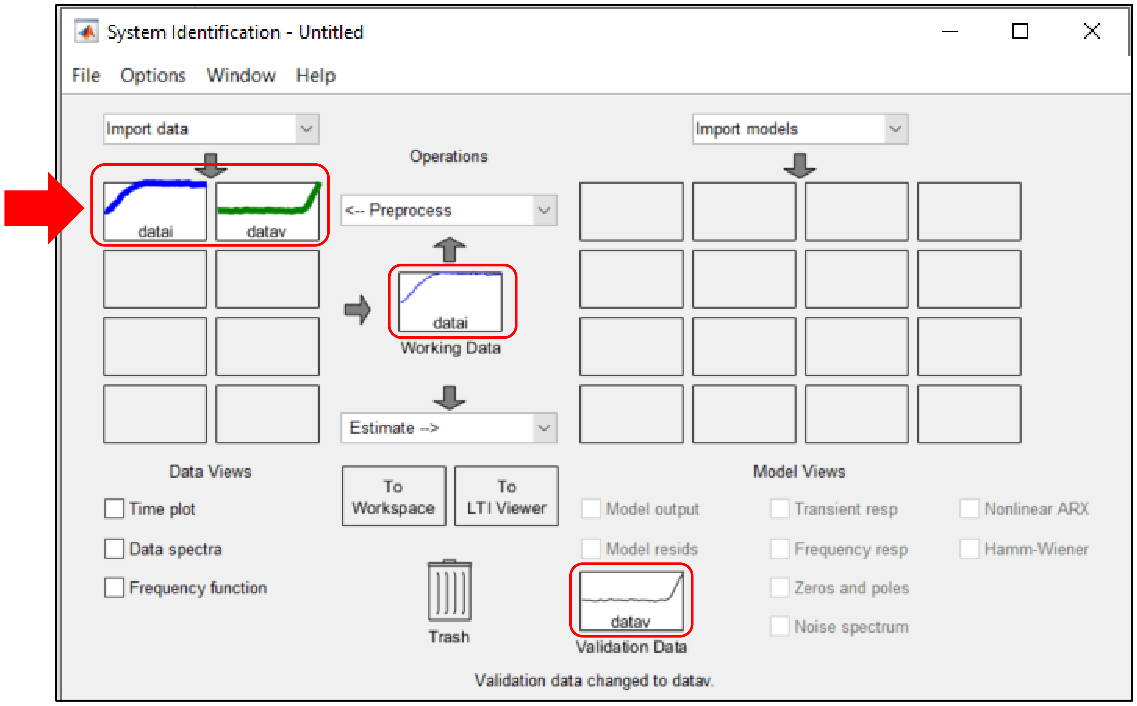
8. Select **Import data > Time domain data**. This command opens the **Import Data** window to define the data necessary for identification and validation.



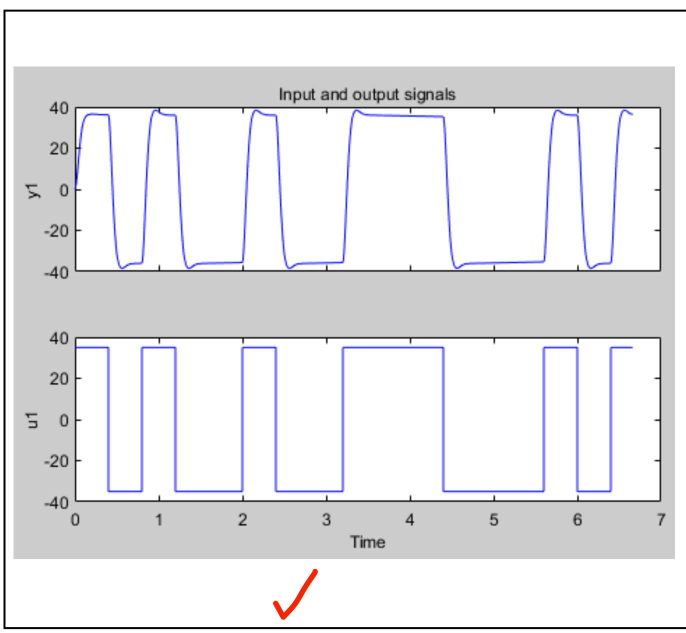
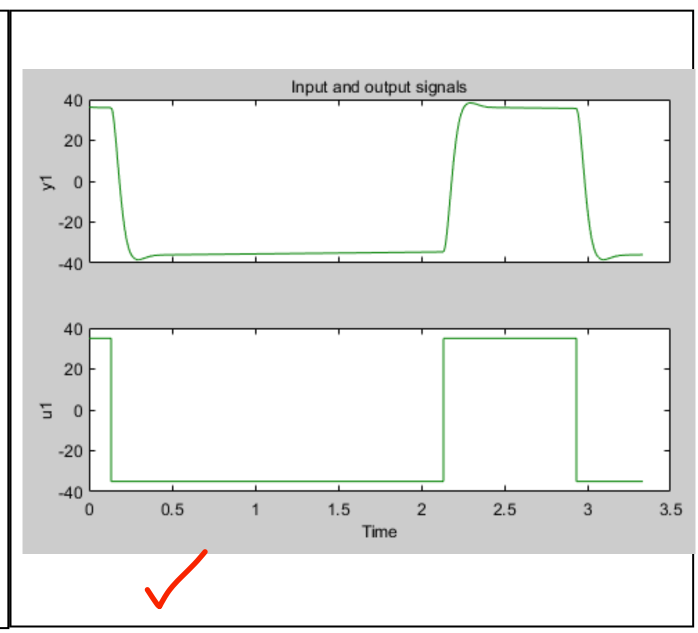
Complete the **Import Data** window as shown below to create the *identification dataset* and *validation dataset*. Each time clicking **Import** the data will appear in the **System Identification** window.



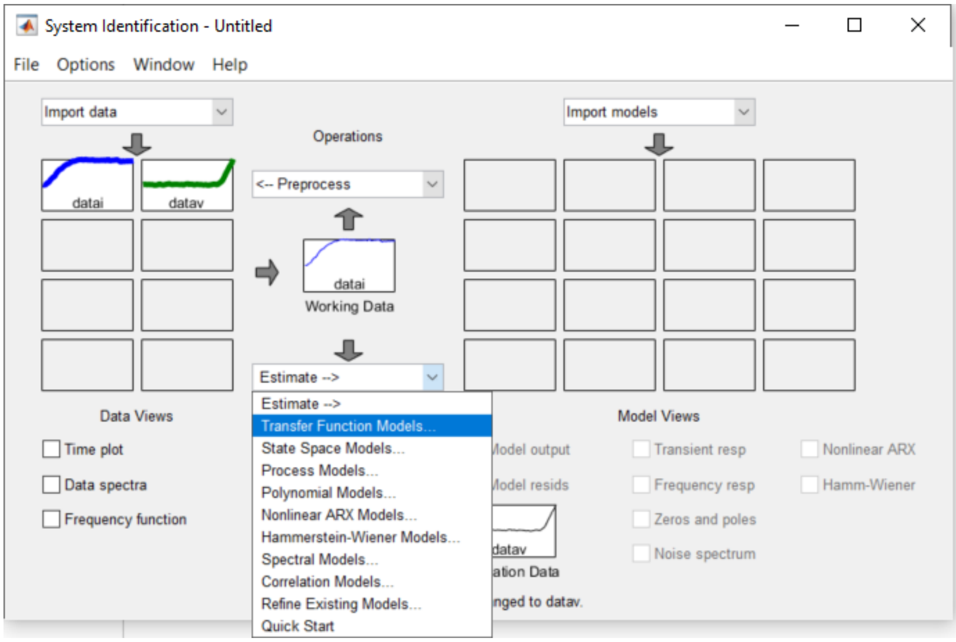




9. *Drag and drop* the **datai** in the **Working Data** box and the **datav** in the **Validation Data** box.
10. We can plot the identification and validation datasets by selecting the desired dataset and deselecting the others and then *check* the box **Time plot.** Select **datai** and deselect **datav**. *Check* the box **Time plot**. The identification dataset will be shown. Select **datav** and deselect **datai**. *Check* the box **Time plot**. The validation dataset will be shown. Provide the graphs below:

Noise-free Identification Dataset	Noise-free Validation Dataset
	

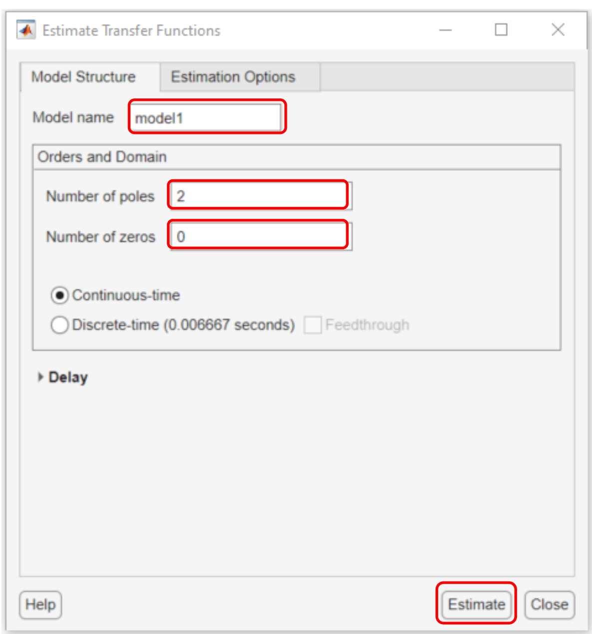
11. Select the desired model type as **Transfer Function Models**.



12. The window that opens allows us to choose the desired type of transfer function. Here we use a *second order* transfer function with **2 poles** and **no zero** and **no delay**. Set the model structure as below and click **Estimate**.

    Model name = model1, **Number of poles** = 2, **Number of zeros** = 0,

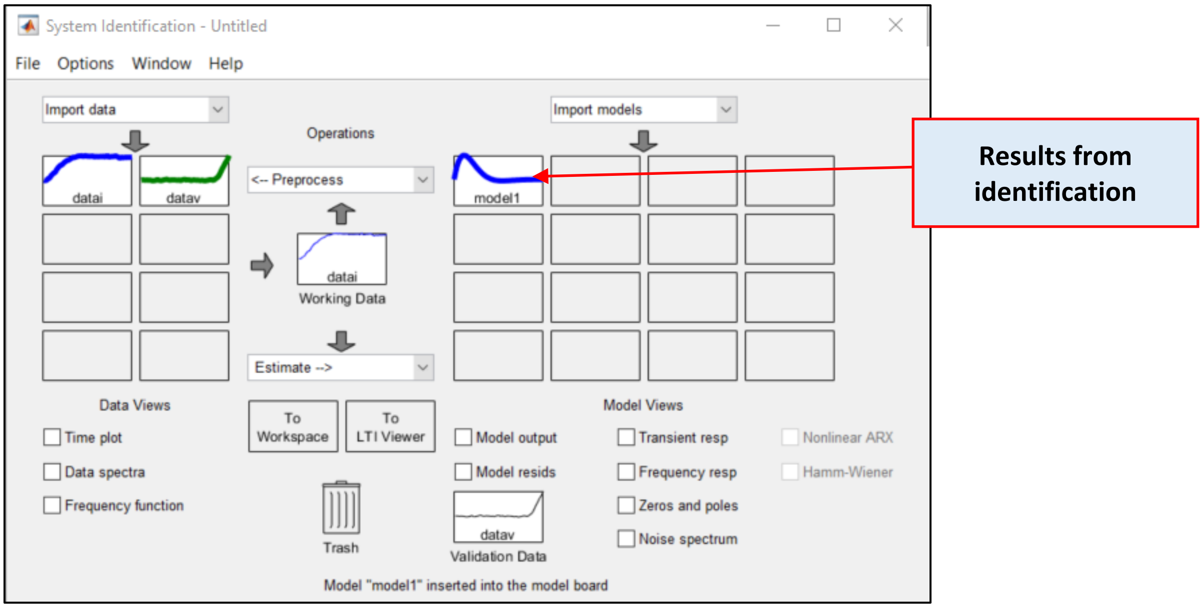
    Check the **Continuous-time** model.



13. The system **Identification Progress** window will appear, which shows the selected algorithm and the status of the estimation. Provide the **percent of fit to estimation data** and the **FPE** values below:

Fit to estimated data = 99.2%
Final Prediction Error (FPE) = 0.0750438

14. The identified model **model1** will appear in the **Identification Results** chart. The result of the identification is available by right clicking the mouse on the **model1** box representing the identification results.



15. Right-click on the **model1** box and find the estimated transfer function model of the **DC motor** system. Provide the model and the validation criteria below:

Estimated Transfer Function Model = $\dfrac{869}{s^2+42.53s+847.4}$

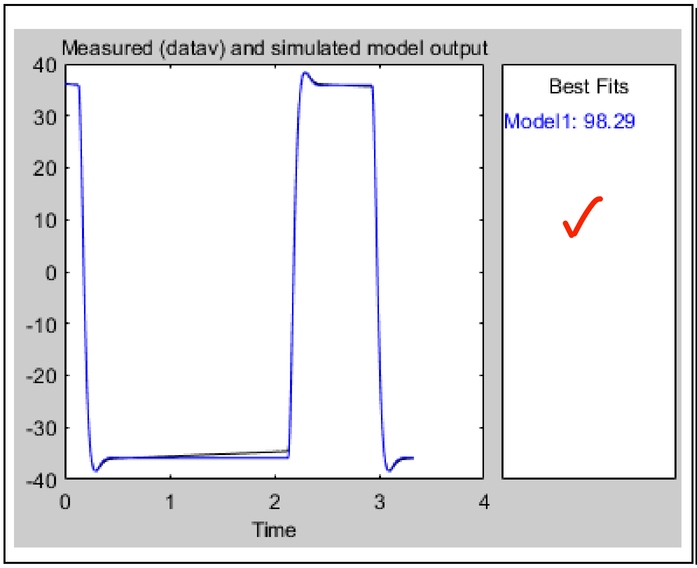
Fit to estimated data = 99.2%

FPE = 0.07504

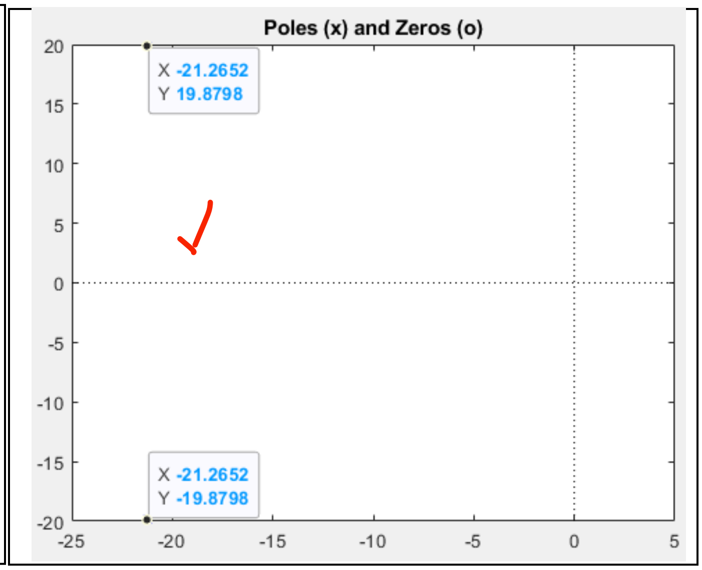
MSE = 0.0743

16. Select **model1** and check the **Model output** box to view the overlap of the *validation data* and the *model* resulting from the identification. Provide the graph with the **Best fit** results below. Check the **Zero and poles** box to view the pole-zero map. Provide the graph below.

Cross-Validation Results for Noise-free Dataset



Pole-Zero Plot for Noise-free Dataset

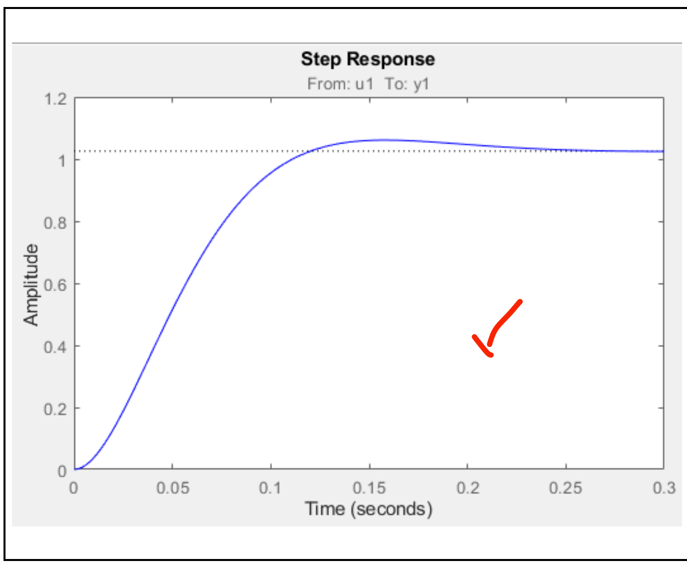


17. Enter the pole and zero values below:

Poles = -21.2652 ± 19.8798i
Zeros = No Zero

18. Drag the **model1** box to **To LTI Viewer** box to plot the unit-step response of the system. Provide the step response graph below:

Unit-Step Response (Noise-free Data)



Step Response Specifications

Rise time = 0.0764

Peak time = 0.1581

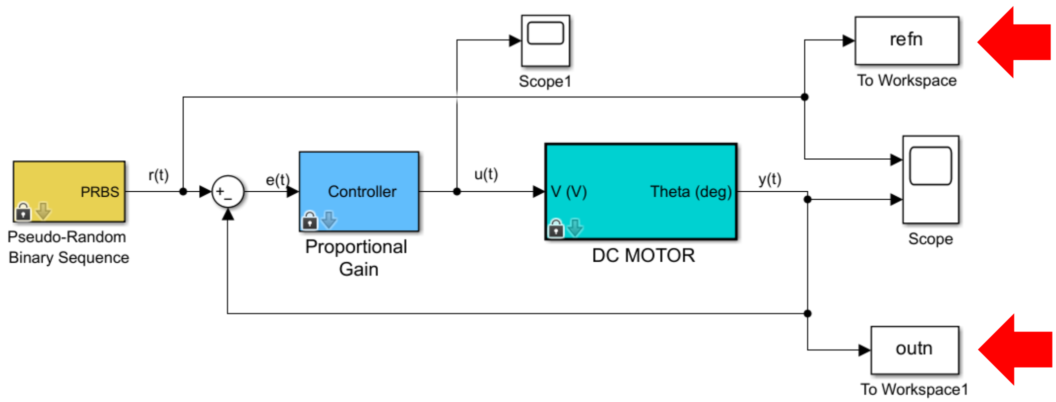
Overshoot = 3.4717

Settling time = 0.2018

Steady-state value = 1.0255

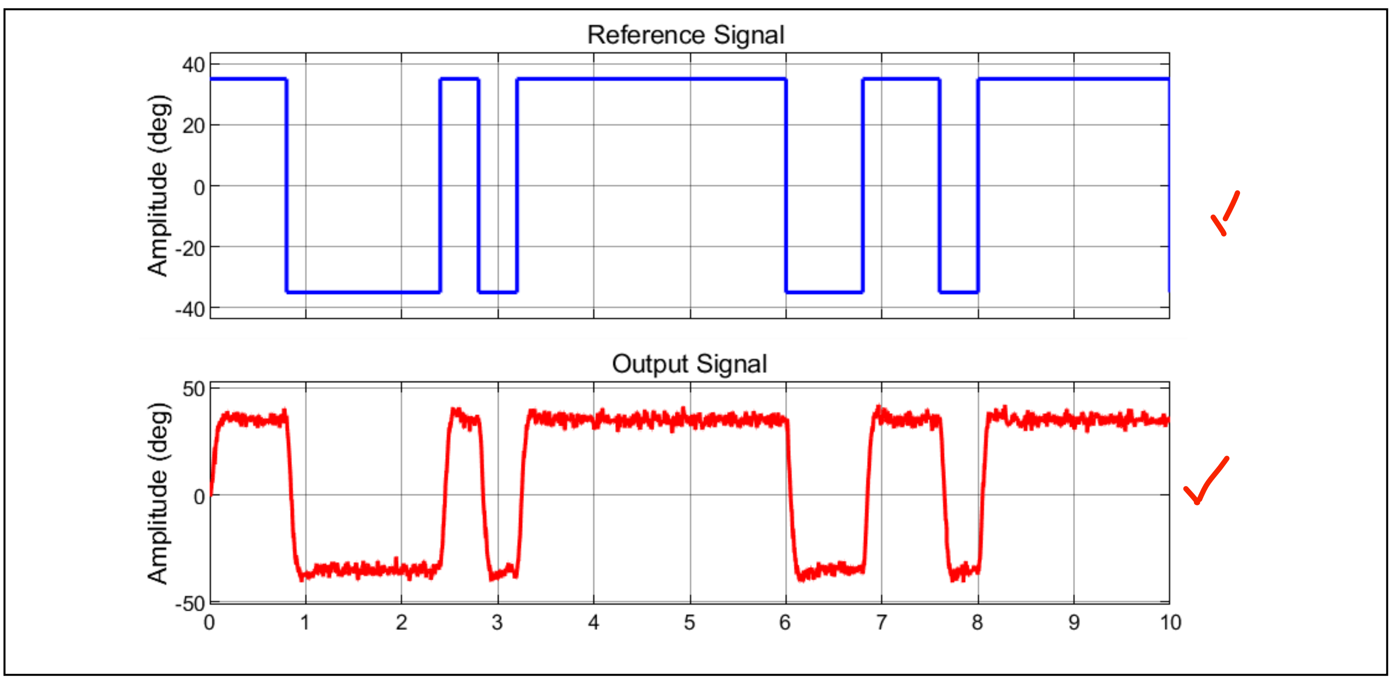
Part B: System Identification from Noisy Data

19. In DC Motor block set **Additive Noise = 1**.
20. Change the input and output data name to **refn** and **outn** to distinguish between the noisy and noise-free datasets in **Part A** and **Part B**.



21. **Run** the simulation for **10 seconds.** Open both **Scope** blocks. You will see the input-output data and the control signal. You will clearly see the noise effect on the output data and the control signal. Ensure that the control signal is **not saturated**. Decrease the amplitude of input signal if required. Provide the graph below:

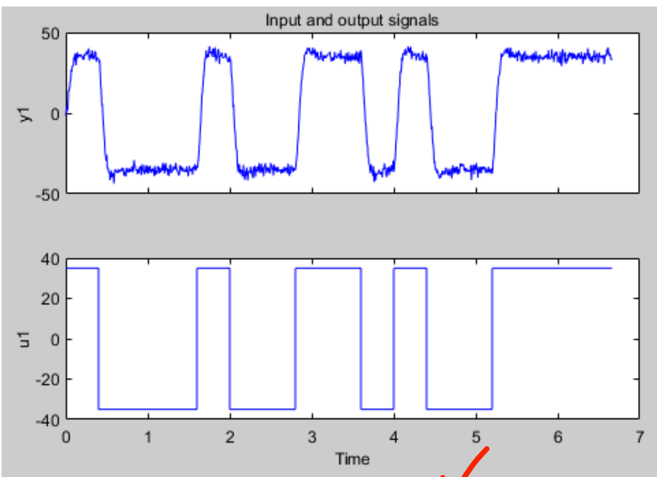
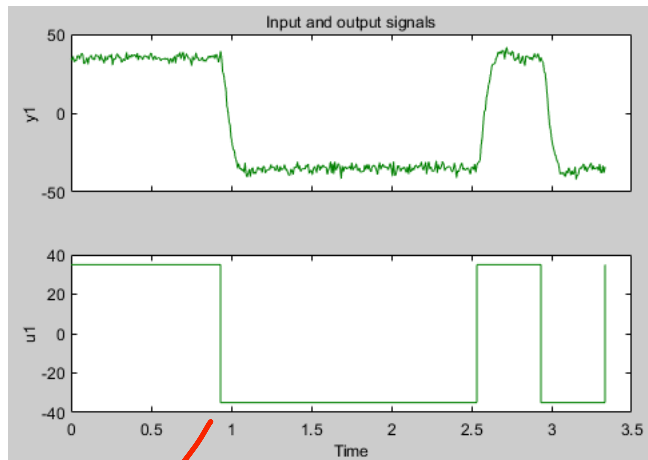
Noisy Input-Output Data



22. **Run** the following code in **MATLAB** to create the new **identification** and **validation** datasets in the **Workspace**.

```
% identification dataset
outni = outn(1:1000);
refni = refn(1:1000);

% Validation dataset
outnv = outn(1001:end);
refnv = refn(1001:end);
```

23. Repeat the **Steps 7** to **Step 10** to plot the identification and the validation datasets for **noisy data**. Provide the graphs below:

Noisy Identification Dataset	Noisy Validation Dataset
	

24. Repeat the **Steps 11** to **Step 15** to identify the model. Name the model as **model2**. Right-click on the **model2** box and find the estimated transfer function model of the **DC motor** system from **noisy data.** Provide the new model and the validation criteria below:

Estimated Transfer Function Model = $\dfrac{828.5}{s^2+40.6s+825}$

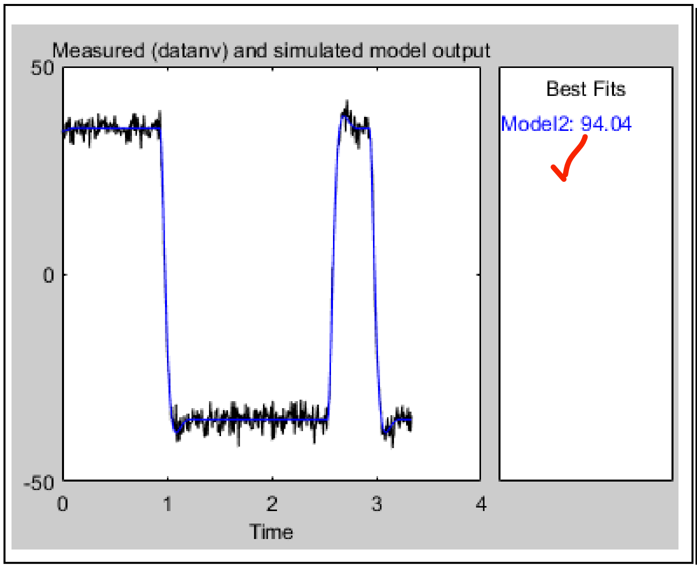
Fit to estimated data = 93.93%

FPE = 4.288

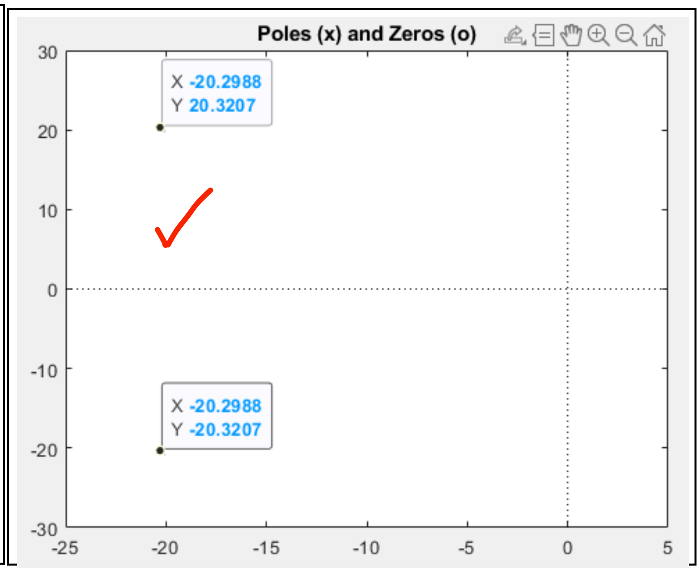
MSE = 4.245

25. Select **model2** and deselect **model1**. Check the **Model output** box to view the overlap of the *validation data* and the *model* resulting from the identification of noisy system. Provide the graph with the **Best fit** results below. Check the **Zero and poles** box to view the pole-zero map. Provide the graph below.

Cross-Validation Results for Noisy Dataset



Pole-Zero Plot for Noisy Dataset



26. Enter the pole and zero values below:

Poles = -20.2988 ± 20.3207i

Zeros = No zero

27. Compare the validation results of noisy dataset in **Step 24** with the noise-free dataset in **Step 15**. Explain how the additive noise effects the estimated transfer function model and the validation criteria.
Compare the poles' location in **model1** and **model2** from **Step 17** and **Step 26**. Are they almost in the same location?

Additive noise in system identification affects the predicted transfer function model's accuracy and reliability. Noise increases variability, which can change the underlying system response and make the estimated model less accurate. In fact, this results in a transfer function that may not accurately portray the system's underlying dynamics, particularly if the noise is significant in comparison to the signal. This issue is typically addressed with model validation criteria such as fit percentages and error metrics, which may indicate decreasing accuracy when the model incorporates noisy input.

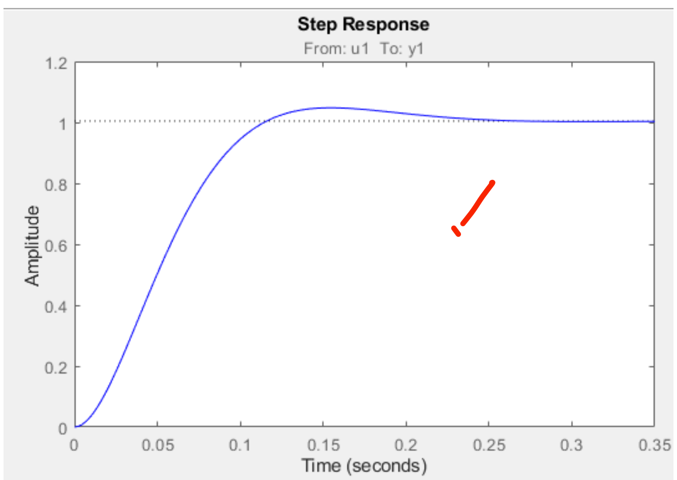
The poles in the two variations are near each other, but they are not identical. This slight variance might be due to differences in how the models react to noise. Since poles represent the stability and inherent responsiveness of the system, slight changes in pole position imply that noise has some effect on the predicted dynamics of the system. Even little differences can have an impact on the model's predictability, even though comparable dynamic behavior is typically inferred when poles in various models are close to one another.

28. Compare the cross-validation results in **Step 16** and **Step 25**. Which model has better percentage of fit on the validation data? **Model1** or **Model2**? Check the cross-validation graph of noisy system. Did the identified model from noisy dataset successfully capture the dynamics of the system? If yes, discuss why the model from the noisy dataset has less percentage of fit? Provide ideas to improve the percent of fit for noisy dataset.

Compared to Model 2, Model 1 more accurately depicts the system's behavior on the validation dataset. The model's fit is jeopardized by the noise, though, if the noisy system's cross-validation graph displays notable deviations. A poor fit percentage might result from the model capturing both the real system dynamics and some noise-related features in a noisy dataset.
When the data is noisy, the model tends to include components of the noise into its structure, resulting in "overfitting" rather than accurately describing the system dynamics. This results in a model that performs well on noisy data but poorly on unseen or less noisy data, yielding a lower fit percentage. Noisy input might obscure the system's true links, making generalization more difficult and lowering the model's predictive ability.
Improvement for noisy data: Noise Filtering, Regularization, Data Segmentation, Increasing Model Order, Ensemble Methods

29. Drag the **model2** box to **To LTI Viewer** box to plot the unit-step response of the system. Provide the step response graph below:

Unit-Step Response (Noisy Data)



Step Response Specifications

Rise time = 0.0748

Peak time = 0.1543

Overshoot = 4.3347

Settling time = 0.2076

Steady-state value = 1.0043

30. Compare the step response specifications of the model from noisy dataset in **Step 29** and the noise-free dataset in **Step 18**. How close the values are? What can you conclude about the accuracy of the two models?

Overall, both models are quite close, but Model 1 is marginally better in terms of stability and accuracy based on these metrics.

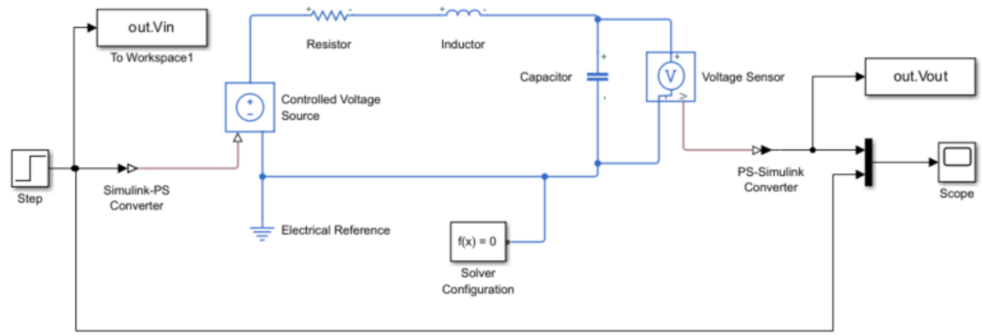
Model 1 is likely a slightly more accurate representation of the system's true behavior, as it fits the validation data better and has a bit less overshoot.
Model 2, while close in performance, may be slightly affected by the noise in the data, leading to a slightly higher overshoot and lower validation fit.

Post Lab Assignment

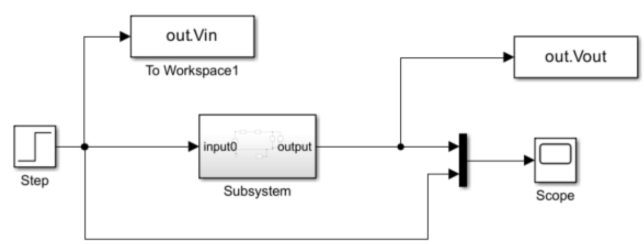
1. Build the following system using **Simscape**. Set the component values as $R = 1\Omega,\ L = 2H$ and $C = 3F$.

   Open the **Model Settings** window and set the **Solver** to **ode23t**.

   Run the simulation for **50 seconds** and provide the **Scope** plot.



Create a **Subsystem** as shown below and suppose that it is an unknown system.



Set the **sample time** of the **To Workspace** blocks to **0.01 seconds**.

First, create a **step input** starting from 0, with an **initial value** of 0 and a **final value** of 2.

Run the simulation and collect the input and output data. Provide the **Scope** plot. The input and output data will be saved in **Workspace** as `out.Vin` and `out.Vout` variables. Save this data as the **identification data**:

```
In_i = out.Vin;
Out_i = out.Vout;
```

Next, create a **step input** starting from 1, with an **initial value** of -1 and a **final value** of 3.

Run the simulation and collect the input and output data. Provide the **Scope** plot. The input and output data will be saved in **Workspace** as `out.Vin` and `out.Vout` variables. Save this data as the **validation data**:

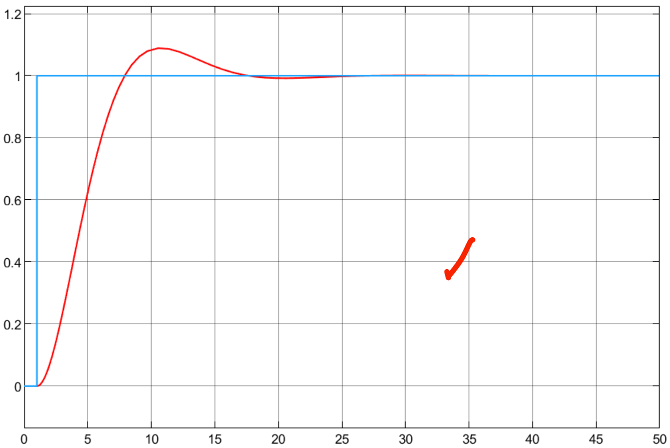
```
In_v = out.Vin;
Out_v = out.Vout;
```

Estimate a **transfer function model** for this system using the **System Identification Toolbox App** by selecting appropriate numbers of poles and zeros.

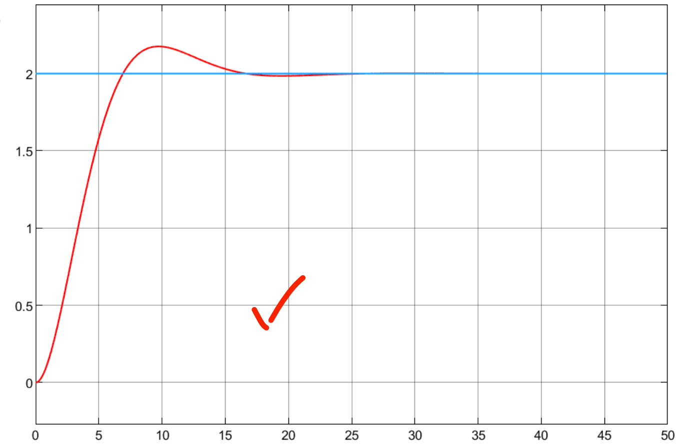
Provide the estimated model, all validation criteria results (FPE, MSE, Fit to the estimated data), cross-validation plot and the pole-zero map. Discuss the validity of the identified model based on the results.

Provide the Simulink file, all the required graphs and the results.

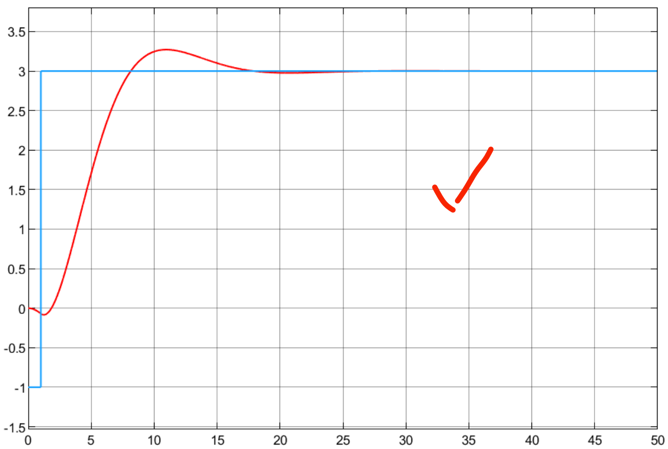
a)



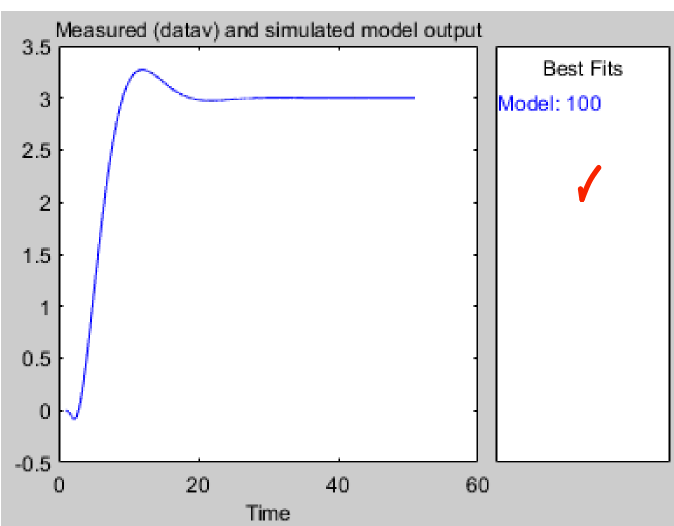
b)



c)

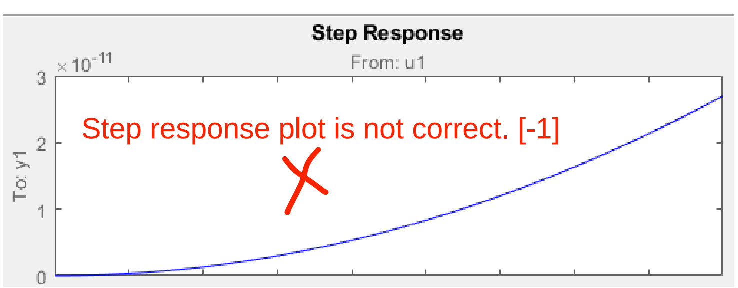


d)

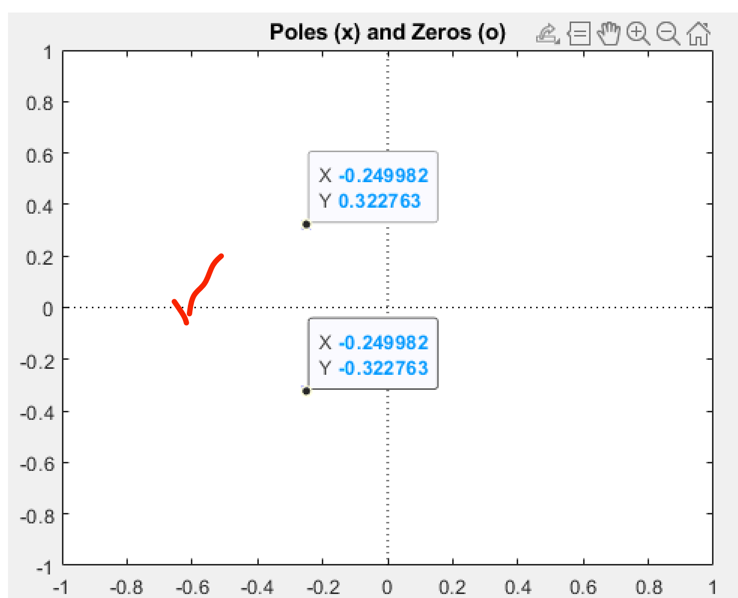



```
            0.1667
y1:  ---------------------
     s^2 + 0.5 s + 0.1667
```

```
Status:
Estimated using TFEST on time domain data "datai".
Fit to estimation data: [100;100]% (stability enforced)
FPE: 7.973e-41, MSE: 2.703e-11
```



Step response plot is not correct. [-1]



It has a 100% Fit so the validity is strong