**Lab 1 Report**

**Pre-lab Exercise: Sinusoidal Data Generation and Plotting**

**1. Introduction**

The objective of this pre-lab exercise is to refresh MATLAB skills by generating sinusoidal data (sine and cosine waves) and plotting the results, both original and after data truncation. This report details the steps to achieve the intended plots using MATLAB.

**2. Objective**

The goal is to generate two sinusoidal datasets, introduce noise, and then truncate part of the data. Finally, the original and truncated datasets will be plotted to visualize the differences.

1. **Procedure**

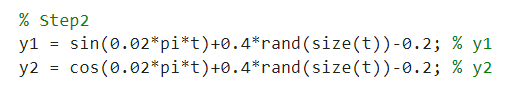
Step 1: Generate Time Vector

A time vector t was created, ranging from 0 to 100 seconds with a step size of 0.1 seconds.



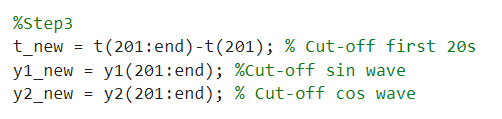
Step 2: Generate Sinusoidal Data

Using the sin and cos functions, two sinusoidal waveforms, y1 and y2, were generated. A uniformly distributed random noise was added to both signals.



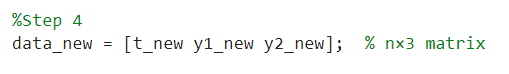
Step 3: Truncate Data

The first 200 samples, corresponding to the first 20 seconds of data, were removed from the dataset. The time vector was adjusted accordingly.



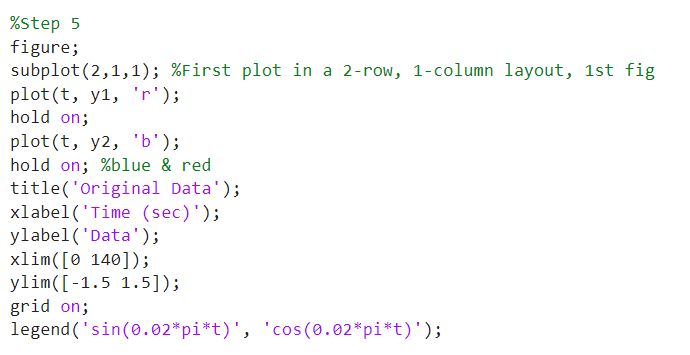
Step 4: Create New Matrix

A new matrix data\_new was created, containing the truncated time, sine, and cosine values.



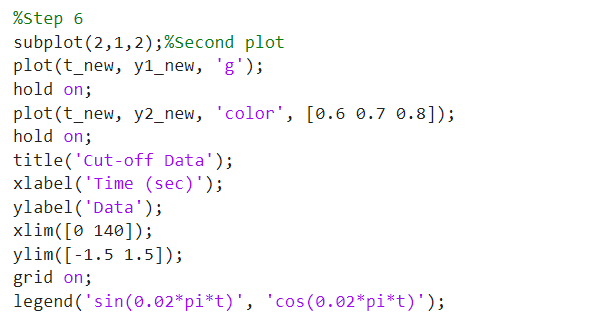
Step 5: Plot the Original Data

The original sinusoidal data was plotted on a figure with two subplots. The first subplot shows the original y1 and y2 data.



Step 6: Plot the Truncated Data

The truncated y1\_new and y2\_new data were plotted on the second subplot using different colors.



1. **Results and Analysis**

Here is the plot this code generated.

Original Data: In the first subplot, the red curve represents the noisy sine wave (sin(0.02\*pi\*t)), and the blue curve represents the noisy cosine wave (cos(0.02\*pi\*t)).

Cut-off Data: The second subplot shows the truncated data. The green and gray lines represent the sine and cosine waves after the first 20 seconds of data have been removed.

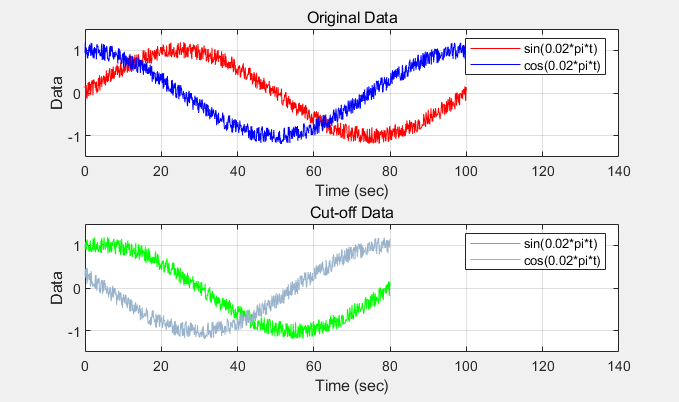


Fig 1

**5. Conclusion**

This pre-lab exercise successfully demonstrated the generation of sinusoidal data in MATLAB, the introduction of noise, and data truncation techniques. Additionally, plotting the results using MATLAB’s plotting functions was accomplished, showcasing both original and truncated datasets.

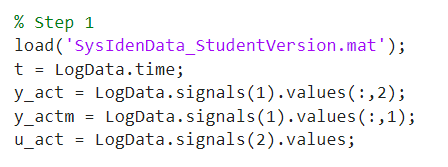
**Lab Exercise: Identifying a Second-Order Discrete-Time Linear Model**

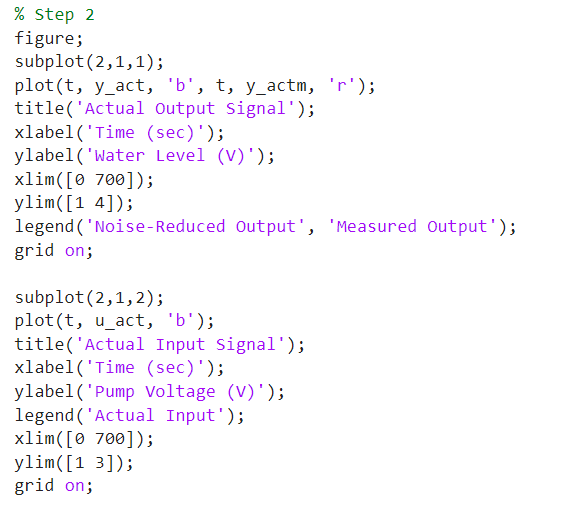
**1. Introduction**

This lab focuses on identifying a second-order discrete-time linear model using pre-collected input-output data from a water tank (W-T) setup. The exercise involves extracting the data, removing offsets, and applying system identification techniques to develop a second-order model, which is then verified by comparing the simulated and actual output.

**2. Data Extraction and Preprocessing**

We first load the pre-collected data (SysIdenData\_StudentVersion.mat) into MATLAB, which contains time (t), the actual output (y\_act), and input signal (u\_act). Both the noise-reduced output (y\_act) and measured output (y\_actm) are plotted for comparison, along with the actual input signal.





Here is the figure generated.

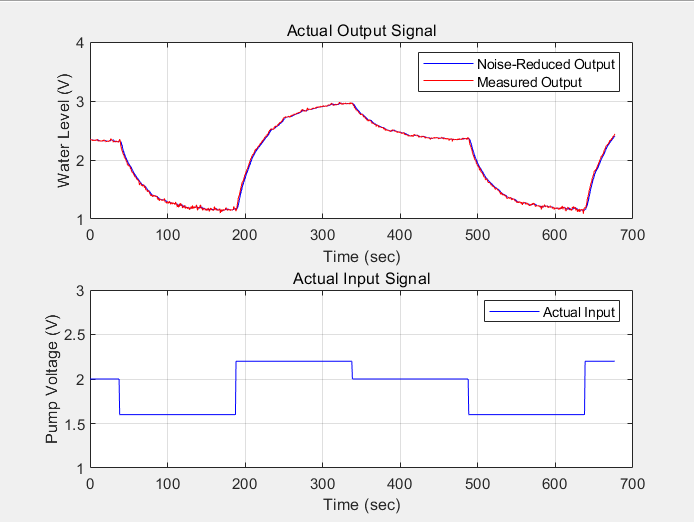
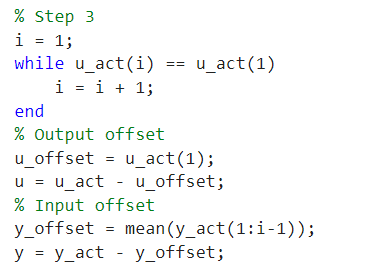


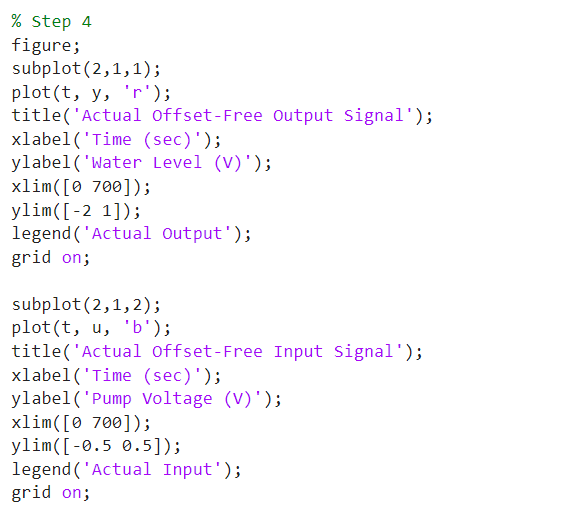
Fig 2

3. **Removing Offset from Data**

Offsets are detected and removed from both input and output signals to create offset-free data. This ensures that the model identifies the correct system dynamics.



After removing the offsets, we plot the offset-free signals.



Here is the figure generated.

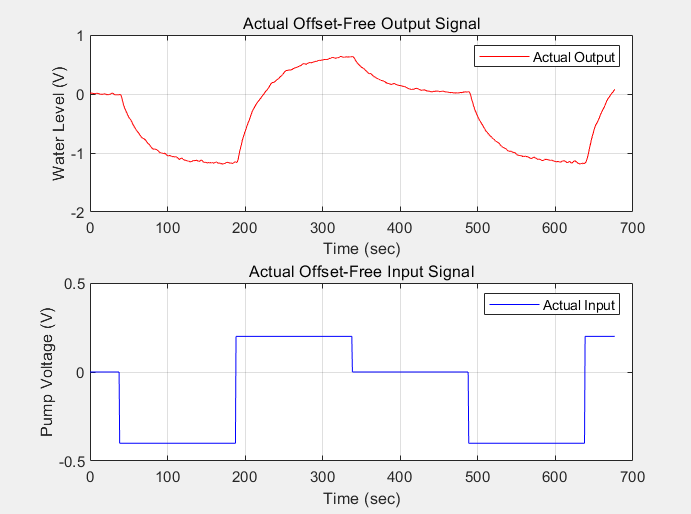
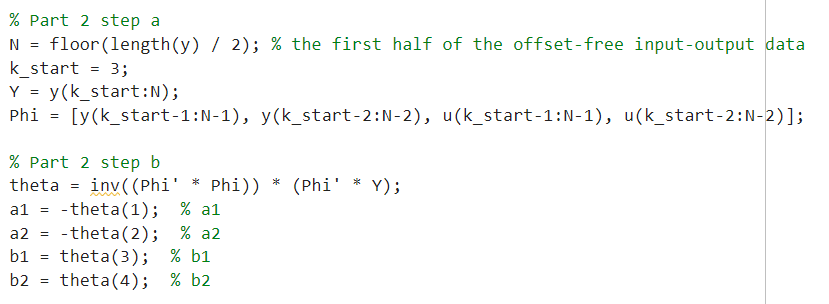


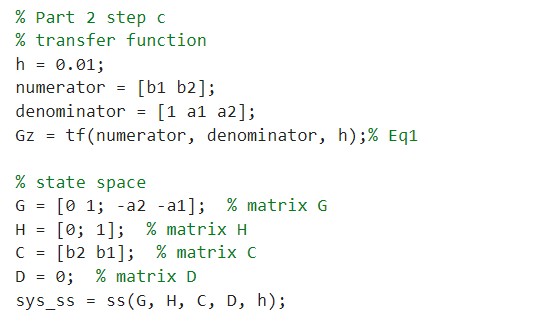
Fig 3

**4.Second-Order Model Identification**

Using the first half of the offset-free data, a second-order model is identified by constructing the matrix Φ and solving for the model parameters , , and .

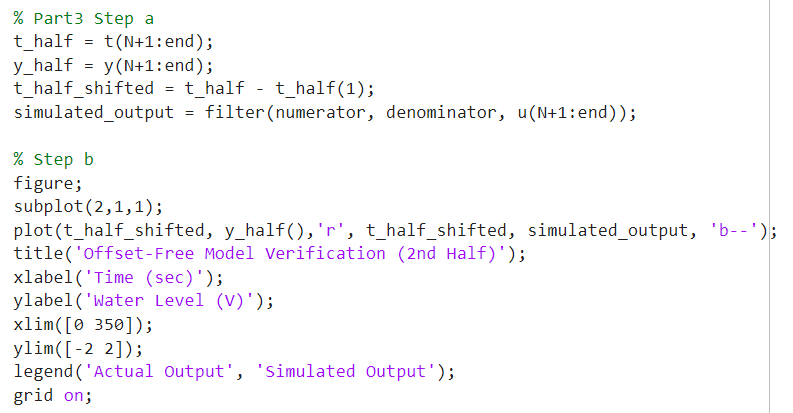


The second-order transfer function and state-space model are derived based on the identified parameters:

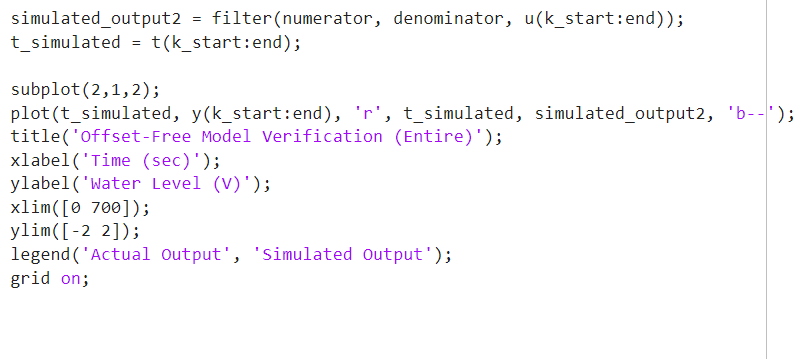


**5. Model Validation**

To validate the model, the second half of the data is used for simulation, and the simulated output is compared to the actual offset-free output.



The entire dataset is then used to simulate the model and compare it with the actual output across the full time period.



Here is the generated figure.

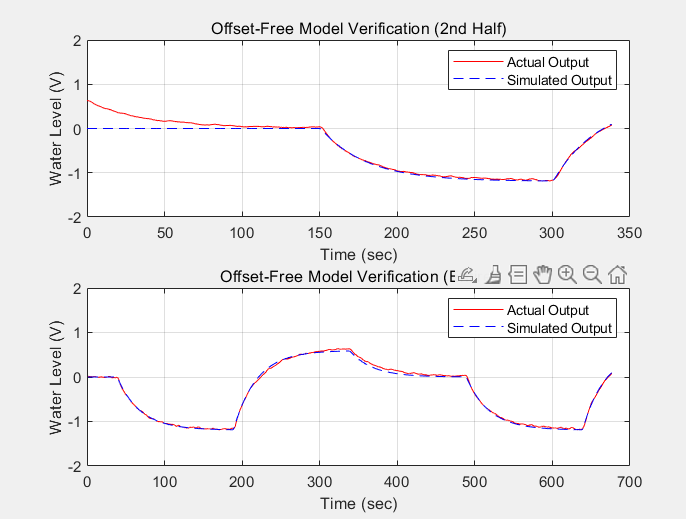


Fig 4

**6. Conclusion**

This lab exercise demonstrated how to extract data, remove offsets, and identify a second-order discrete-time linear model from input-output data. The model was validated using both the second half and the entire dataset, showing close agreement between the simulated and actual outputs. This suggests the model effectively captures the system dynamics of the water tank.

**Post-lab Exercise: Comparison of First and Second-Order Models**

**1. Introduction**

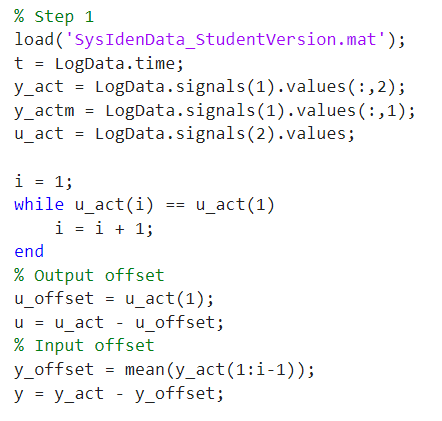
The goal of this post-lab exercise is to compare the accuracy of first-order and second-order discrete-time linear models in system identification. The comparison is based on the Mean Squared Error (MSE) between the simulated model responses and the actual system output.

**2. Model Identification**

We follow the same procedure used in the previous lab exercise to identify both first-order and second-order models using the offset-free input (u) and output (y). The two models are compared by simulating their outputs and comparing them to the actual system response.

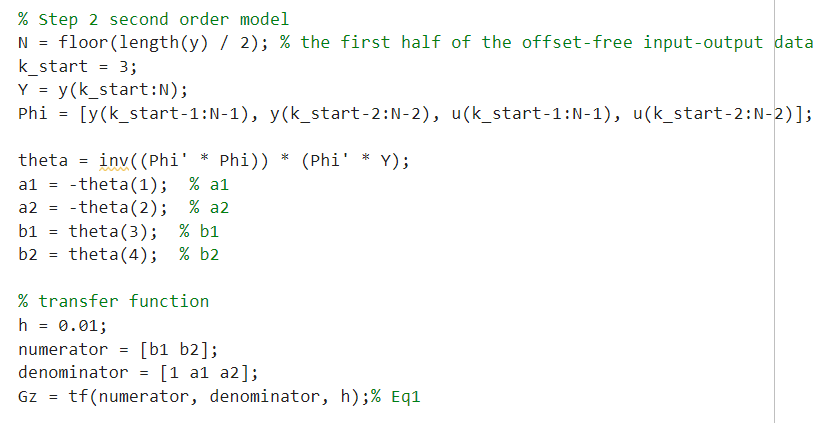
Step 1: Data Extraction

The first step involves loading the pre-collected data (SysIdenData\_StudentVersion.mat), extracting the time, input, and output signals, and removing the offset from the signals.



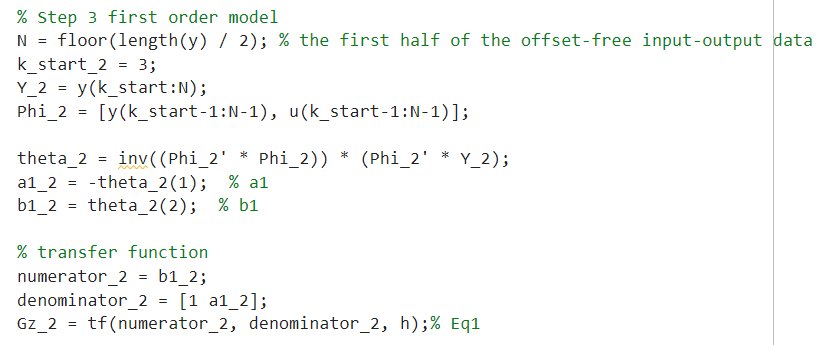
Step 2: Second-Order Model Identification

Using the offset-free input-output data, the second-order model parameters are identified by creating the matrix Φ and solving the system of equations. The transfer function and state-space models are then derived.



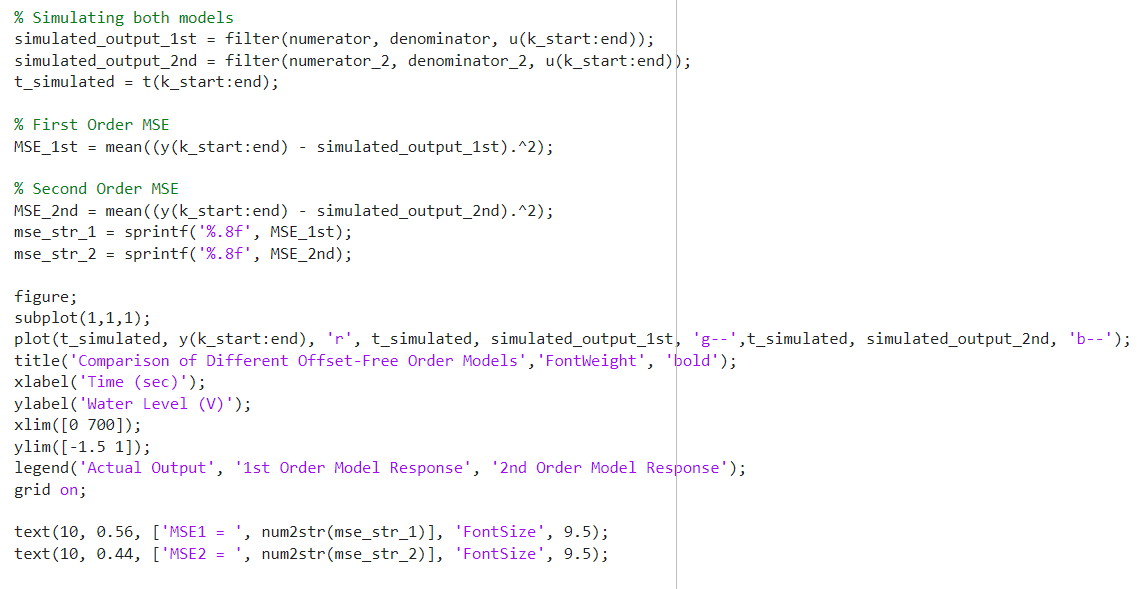
Step 3: First-Order Model Identification

Similarly, a first-order model is identified using the same dataset, but with a simplified equation structure.



**3. Model Simulation and Comparison**

After identifying both models, we simulate the system response using the entire dataset for both first and second-order models. The simulated responses are then compared with the actual output, and the MSE for each model is calculated.



**4. Results and Discussion**

The comparison between the two models is visualized in the plot, with the MSE values for both models displayed on the graph.



Fig 5

First-order model MSE: 0.00096910

Second-order model MSE: 0.00091656

The second-order model demonstrates a slightly lower MSE compared to the first-order model, indicating a marginally better fit to the actual data. Both models track the actual output with reasonable accuracy, but the second-order model captures the system dynamics with more precision due to the inclusion of an additional parameter. The smaller MSE for the second-order model suggests that it provides a more accurate representation of the system's behavior, especially during more complex transitions.

**5. Conclusion**

The second-order model, with a lower MSE (0.00091656), offers better accuracy compared to the first-order model (0.00096910). While the difference is small, the second-order model is preferable for applications requiring precision. However, the first-order model remains a viable option when simplicity and computational efficiency are prioritized.