Simulation and Modeling of Dynamic Systems

Dimitrios Karatis 10775, *Electrical and Computer Engineering, AUTH*

***Abstract*—This document explores the simulation and modeling of dynamic systems, focusing on parameter estimation using the least squares method. It examines how mathematical modeling and numerical solvers can be applied to analyze the behavior of a pendulum system under external control inputs. The project highlights key techniques for accurately representing real-world dynamic behaviors, which are essential for control system design.**

***Index Terms*—simulation, dynamic systems, mathematical modeling, control.**

I. Exercise 1

A. First Question: Find the state equations and the transfer function

Consider the system of a simple pendulum with a torque input. After linearization (assuming $\sin(q) \approx q$ for small angles q), its equation of motion is given by:

$$mL^2\ddot{q}(t) + c\dot{q}(t) + mgLq(t) = u(t), \tag{1}$$

where $q(t)$ [rad] represents the angular displacement of the pendulum, m [kg] and L [m] denote the mass and length of the pendulum, respectively, c [N·m·s] is a constant damping coefficient, g [m/sec^2] is the gravitational acceleration, and $u(t)$ [N·m] is the control input torque.

For the state equations:

Given:

$$\ddot{q}(t) + \frac{c}{mL^2}\dot{q}(t) + \frac{g}{L}q(t) = \frac{1}{mL^2}u(t)$$

We define the state variables:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad q = x_1$$

So,

$$x_1 = q(t)$$
$$x_2 = \dot{q}(t)$$

$$\implies \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{c}{mL^2}x_2 - \frac{g}{L}x_1 + \frac{1}{mL^2}u \end{cases}$$

Finally, we have the state-space representation:

$$\dot{\mathbf{x}} = \underbrace{\begin{pmatrix} 0 & 1 \\ -\frac{g}{L} & -\frac{c}{mL^2} \end{pmatrix}}_{A}\mathbf{x} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{mL^2} \end{pmatrix}}_{B}u$$

This document was prepared by Dimitrios Karatis as part of the first assignment for the Simulation and Modeling of Dynamic Systems course at Aristotle University of Thessaloniki.

$$q = \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{C}\mathbf{x} + \underbrace{\begin{pmatrix} 0 \end{pmatrix}}_{D}u$$

For the transfer function:

Transfer function

The original differential equation:

$$\ddot{q}(t) + \frac{c}{mL^2}\dot{q} + \frac{g}{L}q = \frac{1}{mL^2}u(t)$$

Using the Laplace transform:

$$s^2Q(s) + \frac{c}{mL^2}sQ(s) + \frac{g}{L}Q(s) = \frac{1}{mL^2}U(s)$$

$$\Rightarrow Q(s)\left[s^2 + \frac{cs}{mL^2} + \frac{g}{L}\right] = \frac{1}{mL^2}U(s)$$

$$\Rightarrow H(s) = \frac{Q(s)}{U(s)} = \frac{\frac{1}{mL^2}}{s^2 + \frac{cs}{mL^2} + \frac{g}{L}}$$

$$\Rightarrow H(s) = \frac{1}{mL^2s^2 + cs + mgL}$$

B. Second Question: Simulate the system using MATLAB

Next we will simulate the response of a dynamic system described by an Ordinary Differential Equation (ODE) using MATLAB's ODE solver. The system is driven by a sinusoidal input $u(t) = A_0\sin(\omega t)$, with zero initial conditions. We will use the following parameters:

- Mass: $m = 0.75\,\text{kg}$
- Length: $L = 1.25\,\text{m}$
- Damping coefficient: $c = 0.15\,\text{Ns/m}$
- Gravitational acceleration: $g = 9.81\,\text{m/s}^2$
- Input amplitude: $A_0 = 4$
- Input frequency: $\omega = 2\,\text{rad/s}$

The simulation will run for 20 seconds with a time step of $\Delta t = 10^{-4}$ sec to ensure accurate results. Finally, we will plot the system's state variables to visualize its behavior over time.

For the response of the system and the plots of the system's state variables we have:

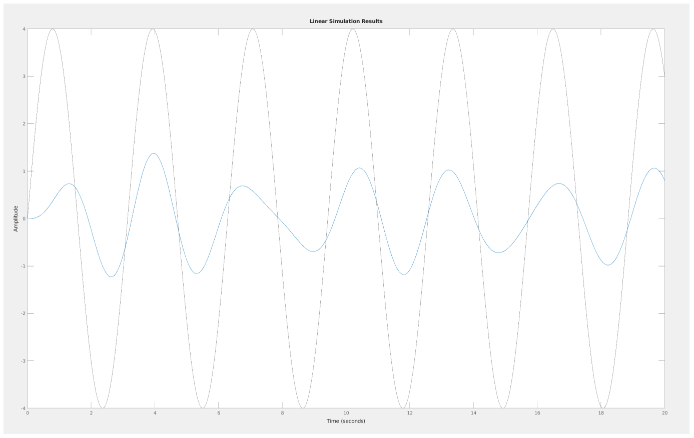


Fig. 1: The response of the system. The plot shows the applied input in gray and the system response in blue

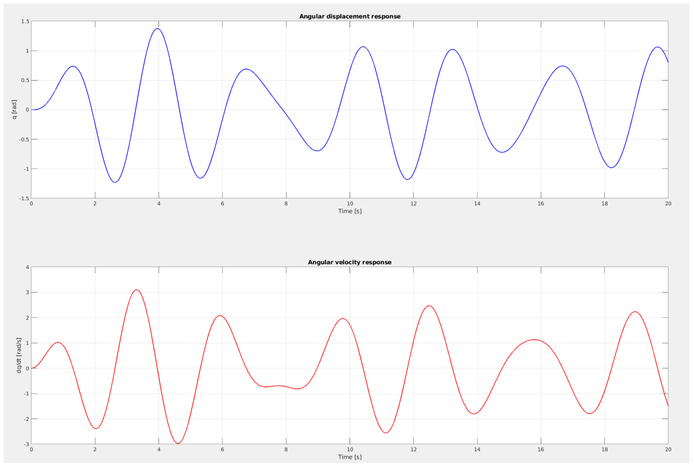


Fig. 2: State system variables

Initially, both signals start from zero, which matches the zero initial conditions. During the first few seconds, the response grows irregularly — this is the transient phase, where the system balances between its natural motion and the input $u(t) = 4sin(2t)$.

Around 8 seconds, we notice a shift in behavior. This is where the transient effects fade, and the system settles into a steady-state response.

The velocity peaks earlier than displacement — this matches the physics of oscillatory motion, where velocity leads displacement in time. The red curve's peaks also become smoother after 8 seconds, confirming the system has settled into periodic behavior.

Detailed calculations as well as code can be found in the thema1_b.m file.

II. EXERCISE 2

Given the system:

$$\ddot{q}(t) + \frac{c}{mL^2}\dot{q}(t) + \frac{g}{L}q(t) = \frac{1}{mL^2}u(t)$$

$$\Rightarrow \ddot{q} = -\frac{c}{mL^2}\dot{q} - \frac{g}{L}q + \frac{1}{mL^2}u$$

We define:

$$\theta_a^* = \begin{bmatrix} a_1 & a_2 \end{bmatrix}^T = \begin{bmatrix} \frac{c}{mL^2} & \frac{g}{L} \end{bmatrix}^T \quad \text{and} \quad \theta_b^* = \begin{bmatrix} b_0 \end{bmatrix}^T = \frac{1}{mL^2}$$

Thus, $\theta^* = \begin{bmatrix} \theta_a^* & \theta_b^* \end{bmatrix}^T = \begin{bmatrix} a_1 & a_2 & b_0 \end{bmatrix}^T = \begin{bmatrix} \frac{c}{mL^2} & \frac{g}{L} & \frac{1}{mL^2} \end{bmatrix}^T$

Also,

$$\Delta = \begin{bmatrix} -\dot{q} & -q & u \end{bmatrix}^T$$

and so the initial system can be written as

$$\ddot{q} = \theta^{*T}\Delta$$

A. q(t) and u(t) measurable

Given that the only measurable signals are the input $\mathbf{u}$ and the output $\mathbf{q}$, the above form cannot be implemented because the derivative of $\mathbf{q}$ is unknown. This problem is overcome by using a stable second-order filter $\frac{1}{\Lambda(s)}$, specifically:

$$\Lambda(s) = s^2 + s(p_1 + p_2) + p_1 p_2 = (s + p_1)(s + p_2)$$

where p_1 and p_2 are poles placed in the left half-plane to ensure stability,

$$\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix}^T = \begin{bmatrix} p_1 + p_2 & p_1 p_2 \end{bmatrix}^T$$

Thus, by filtering both sides of the equation $\ddot{q} = \theta^{*T}\Delta$ with $\frac{1}{\Lambda(s)}$, after calculations, we end up with an expression of the form:

$$q = \theta_\lambda^T J, \text{where } J = \begin{bmatrix} -\frac{\Delta n-1^T(s)}{\Lambda(s)}q & \frac{\Delta m^T(s)}{\Lambda(s)}u \end{bmatrix}^T,$$

$$\theta\lambda = \begin{bmatrix} \theta_a^{*T} - \lambda^T & \theta_b^{*T} \end{bmatrix},$$

$$\Delta_{n-1}(s) = \begin{bmatrix} s & 1 \end{bmatrix}^T \text{ and}$$

$$\Delta_m(s) = 1$$

so finally we have the linearly parameterized system:

$$q = \theta_\lambda^T J, \text{where}$$

$$J = \begin{bmatrix} -\frac{\begin{bmatrix} s & 1 \end{bmatrix}^T}{s^2+(p_1+p_2)s+p_1p_2}q & +\frac{1}{s^2+(p_1+p_2)s+p_1p_2}u \end{bmatrix}^T \text{ and}$$

$$\theta\lambda = \begin{bmatrix} \frac{c}{mL^2} - \lambda_1 & \frac{g}{L} - \lambda_2 & \frac{1}{mL^2} \end{bmatrix}^T$$

Next, we will use the least squares method to estimate the unknown parameters L, m, c.

It is easy to see that the general form in which our system needs to be expressed is:

$$\mathbf{y} = \boldsymbol{\theta}^T \boldsymbol{\varphi}$$

where, $\boldsymbol{\theta}^T = \boldsymbol{\theta}_\lambda^T$ and $\boldsymbol{\varphi} = J$.

We know that the only available measurements are from $\mathbf{u}, \mathbf{q}$. Let us assume that we have N measurements, including both input and output. Thus, Z_N will represent the dataset of the system, i.e., input and output measurements for time instances up to N.

We will need to use a norm of e (where

$$e(t, \boldsymbol{\theta}) = y(t) - \hat{y}(t, \boldsymbol{\theta})$$

is the prediction error), which takes the form:

$$V_N(\boldsymbol{\theta}, Z_N) = \frac{1}{N} \sum_{t=1}^{N} l(e(t, \boldsymbol{\theta}))$$

and specifically, for the least squares algorithm, we will use the norm $l(e) = \frac{1}{2}e^2$.

Next, we construct a vector

$$\mathbf{Y}(t) = [y(1) \quad y(2) \quad \dots \quad y(N)]^T \in \mathbb{R}^N$$

which results from the output measurements, and a matrix $\boldsymbol{\Phi}$ consisting of the components of the vector $\boldsymbol{\varphi}$ for each measurement we have. That is, $\varphi_i(j)$ represents the j-th measurement of the i-th component of the vector $\boldsymbol{\varphi}$. Thus:

$$\boldsymbol{\Phi} = \begin{bmatrix} \varphi_1(1) & \varphi_2(1) & \varphi_3(1) \\ \vdots & \vdots & \vdots \\ \varphi_1(N) & \varphi_2(N) & \varphi_3(N) \end{bmatrix} \in \mathbb{R}^{N \times 3}$$

The error will therefore take the following form:

$$\mathbf{e} = \mathbf{Y} - \boldsymbol{\theta}^{\boldsymbol{T}} \boldsymbol{\Phi}$$

which is now in vector form due to the number of measurements. I will use the error metric for minimization and application of the algorithm:

$$\boldsymbol{\theta}_0 = \arg\min_{\boldsymbol{\theta}} V_N(\boldsymbol{\theta}, Z_N)$$

Thus, the function V_N becomes:

$$V_N = \frac{||\mathbf{Y} - \boldsymbol{\Phi\theta}||}{2N}$$

This function is convex, so it has a unique minimum, say at $\boldsymbol{\theta}_0$, which I can find by solving the equation:

$$\frac{\partial V_N}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0} = 0 \Leftrightarrow ...\boldsymbol{\theta}_0 = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi} \boldsymbol{Y}$$

Thus, the solution to this system, i.e., the values I find for $\boldsymbol{\theta}_0$, are the estimated values of the parameters we are interested in, namely L, m, c. The vector $\boldsymbol{\theta}_0$ represents the set of parameters with values that minimize the prediction error metric and therefore are the most realistic approximations of the actual values of these quantities.

So, in our example the estimated parameters will come from the $\boldsymbol{\theta}_\lambda^T = \left[\frac{c}{mL^2} - \lambda_1 \quad \frac{g}{L} - \lambda_2 \quad \frac{1}{mL^2}\right]$ $.So$

$$L_{\text{estim}} = \frac{g}{\boldsymbol{\theta_0}(2) + \lambda_2}$$
$$m_{\text{estim}} = \frac{1}{(L_{\text{estim}}^2) \cdot \boldsymbol{\theta_0}(3)}$$
$$c_{\text{estim}} = (m_{\text{estim}} \cdot L_{\text{estim}}^2) \cdot (\boldsymbol{\theta_0}(1) + (\lambda_1))$$

In order to find the $\boldsymbol{\theta_0}$ we will need the $\boldsymbol{\Phi}$ and $\boldsymbol{Y}$, or in our case $\boldsymbol{Q}$.

For the $\boldsymbol{\Phi}$ we have that $\boldsymbol{\Phi} = \boldsymbol{J}$. As for the $\boldsymbol{Q}$, we can have it, by sampling the values of the $q(t)$ that we get from the ODE simulation every 0.1 seconds.

That being said, in my approach i used a second order filter with $p1 = p2 = 0.5$ (> 0, in order for the system to be stable).

Detailed calculations as well as code can be found in the thema2_b.m file.

RESULTS

```
--- Original System Parameters ---
Mass (m): 0.75 kg
Length (L): 1.25 m
Damping coefficient (c): 0.15 N*m*sec

--- Estimated System Parameters (Least-Squares) ---
Estimated Mass (m_estim): 0.75 kg
Estimated Length (L_estim): 1.24 m
Estimated Damping coefficient (c_estim): 0.15 N*m*sec
```

Fig. 3: Actual vs Estimated parameters

The estimated mass and damping coefficient match exactly with their original values, indicating high accuracy in their estimation. However, the estimated length is slightly lower than the original length, showing a small estimation error. This minor discrepancy could be due to model simplifications, or numerical approximations in the estimation process. Overall, the Least-Squares method provides highly accurate parameter estimations for the system and for the filter that was selected.

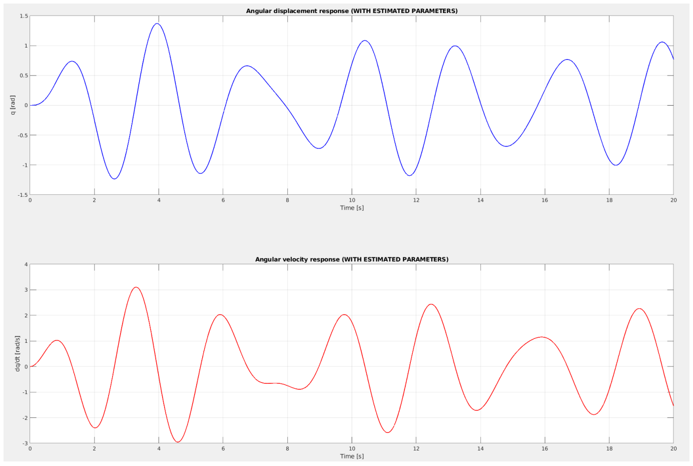


Fig. 4: Response and state system variables using estimates

As observed, the new system response ($q\_estim(t)$, in blue) and the system state variables are nearly the same as before. This outcome is expected since the estimated parameters closely match the actual values of the system.

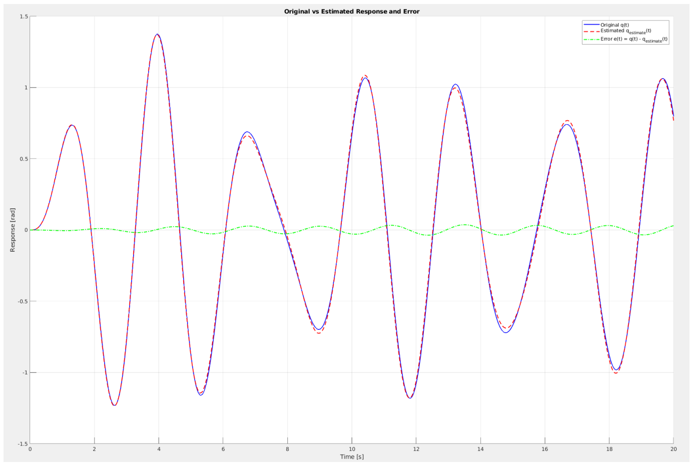


Fig. 5: Error: q(t) - q_est(t)

The Fig.5 compares the original system response $q(t)$ (blue) with the estimated response $q_{\text{estimate}}(t)$ (red), obtained using the Least-Squares estimation method. The two curves align almost perfectly, indicating a highly accurate estimation of the system parameters. The error $e(t) = q(t) - q_{\text{estimate}}(t)$ (green dashed line) remains small and oscillates around zero, confirming that the estimated parameters effectively capture the system dynamics. Overall, the results validate the effectiveness of the estimation process.

B. x(t) and u(t) measurable

At the start of this exercise, we arrived at the following system representation:

$$\ddot{q} = \theta^{*T}\Delta$$

with

$$\theta^* = \begin{bmatrix}\theta_a^* & \theta_b^*\end{bmatrix}^T = \begin{bmatrix}a_1 & a_2 & b_0\end{bmatrix}^T = \begin{bmatrix}\frac{c}{mL^2} & \frac{g}{L} & \frac{1}{mL^2}\end{bmatrix}^T$$

and

$$\Delta = \begin{bmatrix}-\dot{q} & -q & u\end{bmatrix}^T \tag{2}$$

Now all quantities of vector Δ are measurable. However, $\ddot{q}$ is not measurable, so we will apply linear parameterization again, this time a little differently (by using a first order filter):

$$\Lambda(s) = s + p, \quad s = \Lambda(s) - p \tag{3}$$

Using:

$$J = \begin{bmatrix}J_1 & J_2 & J_3\end{bmatrix} = \begin{bmatrix}-\frac{1}{\Lambda(s)} & -\frac{1}{\Lambda(s)} & \frac{1}{\Lambda(s)}\end{bmatrix}\begin{bmatrix}sq(s)\\ q(s)\\ u(s)\end{bmatrix} \tag{4}$$

$$= \begin{bmatrix}-\frac{1}{\Lambda(s)} & -\frac{1}{\Lambda(s)} & \frac{1}{\Lambda(s)}\end{bmatrix}\begin{bmatrix}k(s)\\ q(s)\\ u(s)\end{bmatrix} \tag{5}$$

Since:

$$\ddot{q} = \theta^{*T}\Delta \Rightarrow (\dot{q})^{(1)} = \theta^{*T}\Delta \Rightarrow \dot{k(t)} = \theta^{*T}\Delta, where \tag{6}$$

$$k(t) = \dot{q(t)} \Rightarrow k(s) = sq(s) \tag{7}$$

Taking the Laplace Transform of the equation (6), we have:

$$sk(s) = \theta^{*T}\Delta(s) \tag{8}$$

And by applying the filter:

$$\frac{sk(s)}{\Lambda(s)} = \frac{\theta^{*T}\Delta(s)}{\Lambda(s)} \Rightarrow \frac{sk(s)}{\Lambda(s)} = \theta^{*T}J \tag{9}$$

Now we define:

$$z = \frac{s\kappa(s)}{\Lambda(s)} \Rightarrow z = \frac{\Lambda(s) - p}{\Lambda(s)}k(s) = k - \frac{p}{\Lambda(s)}k \tag{10}$$

$$\Rightarrow z = k + pJ_1 \Rightarrow k = z - pJ_1 \tag{11}$$

Since:

$$z = \theta_1^{*T}J_1 + \theta_2^{*T}J_2 + \theta_3^{*T}J_3 \tag{12}$$

From the equation (11) we get:

$$k = (\theta_1^{*T} - p)J_1 + \theta_2^{*T}J_2 + \theta_3^{*T}J_3 \tag{13}$$

So, finally we have:

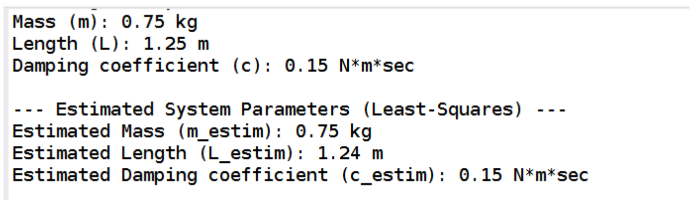
$$k = \Theta^{*T}J, \text{where } \Theta = \begin{bmatrix}\theta_1 - p & \theta_2 & \theta_3\end{bmatrix}^T \Rightarrow \tag{14}$$

$$\Theta = \begin{bmatrix}\alpha_1 - p & \alpha_2 & \alpha_3\end{bmatrix}^T \Rightarrow \Theta = \begin{bmatrix}\frac{c}{mL^2} - p & \frac{g}{L} & \frac{1}{mL^2}\end{bmatrix}^T \tag{15}$$

Unlike before, I now applied a first-order filter with $p = 0.5$ (> 0, ensuring system stability).

Detailed calculations as well as code can be found in the thema2_a.m file.

Results



```
Mass (m): 0.75 kg
Length (L): 1.25 m
Damping coefficient (c): 0.15 N*m*sec

--- Estimated System Parameters (Least-Squares) ---
Estimated Mass (m_estim): 0.75 kg
Estimated Length (L_estim): 1.24 m
Estimated Damping coefficient (c_estim): 0.15 N*m*sec
```

Fig. 6: Actual vs Estimated parameters

When q, $\dot{q}$, and u were measurable, a first-order filter $(s + 0.5)$ was used, yielding a mean squared error (MSE) of 0.00040373. However, when only q and u were available, a second-order filter $(s + 0.5)(s + 0.5)$ was necessary since $\dot{q}$ was missing, yielding a MSE of 0.00040040.

The slightly higher MSE (0.00040373) in the case where $\dot{q}$ was directly measured may seem unexpected at first. However, as it appears, this is influenced by the choice of filter. For instance, when I experimented with a first-order filter of $(s + 0.5)$, the MSE decreased. So it is evident that as the pole moves closer to zero, the MSE decreases. Despite this minor difference, both methods resulted in nearly identical estimated parameters.

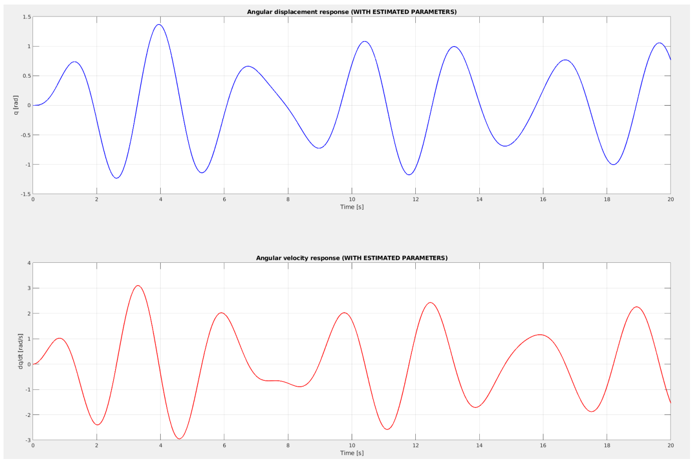


Fig. 7: Response and state system variables using estimates

As observed, the new system response ($q\_estim(t)$, in blue) and the system state variables are nearly the same as before. This outcome is expected since the estimated parameters are the same as the estimated parameters in the previous question.

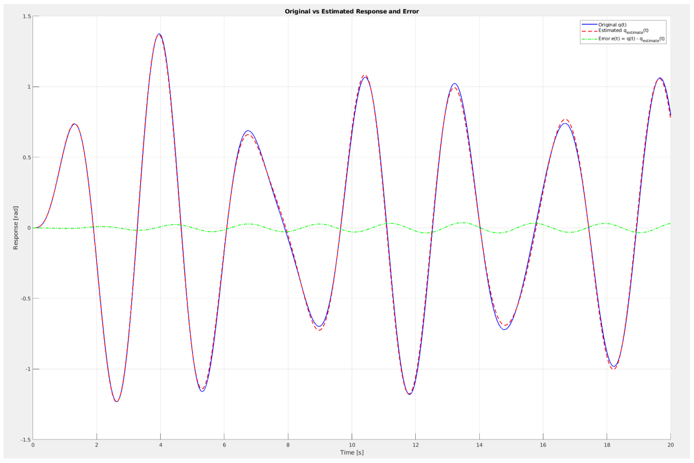


Fig. 8: Error: q(t) - q_est(t)

The Fig.8 compares the original system response $q(t)$ (blue) with the estimated response $q_{\text{estimate}}(t)$ (red), obtained using the Least-Squares estimation method. The two curves seem to be almost identical, indicating a highly accurate estimation of the system parameters. The error $e(t) = q(t) - q_{\text{estimate}}(t)$ (green dashed line) remains small and oscillates around zero, confirming that the estimated parameters effectively capture the system dynamics. Overall, the results validate the effectiveness of the estimation process.

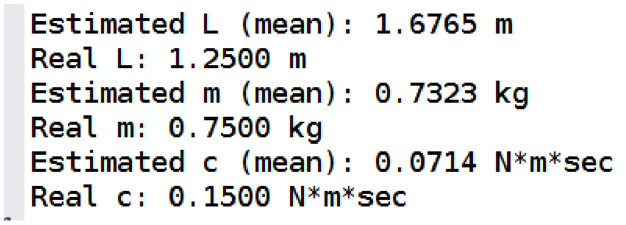
Compared to the previous case, it's clear that having access to sampled values of $\dot{q}$ can enhance the accuracy of parameter estimation—provided that a suitable filter is used alongside it.

III. Exercise 3

A. *First Question: Adding white noise to the samples*

For this exercise, I based my analysis on the setup from Question 2b, where only $q(t)$ and $u(t)$ are assumed to be measurable.

Next, we introduced white Gaussian noise with zero mean and a standard deviation of 0.1 to the samples q(t) and u(t). To improve accuracy, each parameter was estimated 100 times, and the final values were determined as the average of these estimates. The results are as follows:



```
Estimated L (mean): 1.6765 m
Real L: 1.2500 m
Estimated m (mean): 0.7323 kg
Real m: 0.7500 kg
Estimated c (mean): 0.0714 N*m*sec
Real c: 0.1500 N*m*sec
```

Fig. 9: Estimates when adding noise

After adding white Gaussian noise with a mean of zero and a standard deviation of 0.1 to the samples, the estimated parameters deviated from the true values. The length L was overestimated (1.6765 m compared to the true 1.25 m), the mass m was slightly underestimated (0.7323 kg vs. 0.75 kg), and the damping coefficient c was significantly underestimated (0.0714 N·m·sec vs. 0.15 N·m·sec). This suggests that the noise affected the system's dynamic response, particularly impacting the damping estimate, which depends more on velocity data — inherently noisier due to differentiation.

Detailed calculations as well as code can be found in the thema3_a.m file.

B. Second Question: Accuracy of the estimated parameters vs Sampling period

After that, we will study the effect of varying the sampling period $T_s > 0$ on the accuracy of the estimated parameters. We will generate plots showing the estimation error of each parameter as a function of the sampling period and analyze the results.

For that reason, I used a vector of sampling times: [0.001, 0.01, 0.1]. Additionally, the system's poles were set at p1 = p2 = 0.5, in order for the system to be stable.

The plot below shows the absolute estimation error for each parameter as a function of the sampling period:

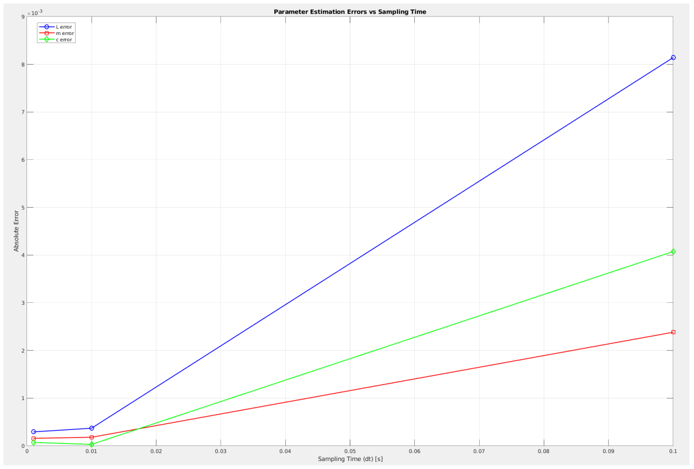


Fig. 10: Absolute estimation error vs Sampling time

From the graph, we observe that as the sampling period increases, the estimation errors for L, m, and c also increase. This happens because larger sampling intervals result in less accurate representations of the system's dynamics, leading to greater deviations in the estimated parameters. The blue curve (for L) shows the steepest rise, indicating that the length parameter is most sensitive to sampling period changes, whereas the red and green curves (for m and c) grow more gradually.

Detailed calculations as well as code can be found in the thema3_b.m file.

C. Third Question: Input amplitude vs Sampling period

Next, we will study the effect of varying the input amplitude A_0 on the accuracy of the estimated parameters, while keeping the sampling period fixed at $T_s = 0.1\,\text{sec}$. We will create plots showing the estimation error of each parameter as a function of the input amplitude and analyze the results.

For that reason, I used a vector of amplitudes: [0.01, 0.1, 1, 10, 100]. Additionally, the system's poles were set at p1 = p2 = 0.5, in order for the system to be stable.

The plot below shows the absolute estimation error for each parameter as a function of the input amplitude:

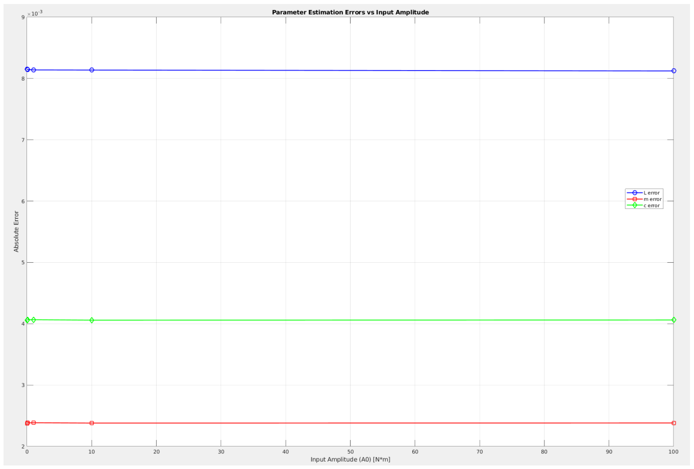


Fig. 11: Absolute estimation error vs Input Amplitude

The plot illustrates the absolute errors in the estimated parameters L, m, and c as a function of the input amplitude A_0 of the excitation signal $u = A_0 \sin(\omega t)$. The results show that the estimation errors remain nearly constant across the range of input amplitudes, indicating that increasing A_0 does not significantly improve parameter estimation accuracy. The error in L (blue) remains the highest, followed by c (green) and m (red), suggesting that L is the most sensitive parameter to estimation inaccuracies.

Detailed calculations as well as code can be found in the thema3_c.m file.

Also, below we can see the each diagram separately.

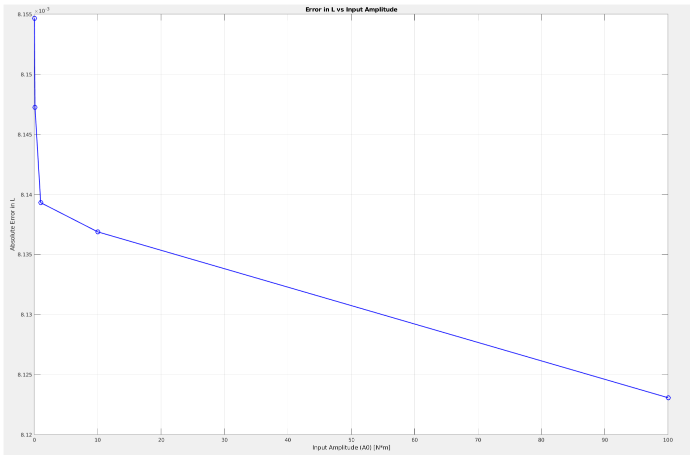


Fig. 12: Absolute estimation error vs Input Amplitude for L

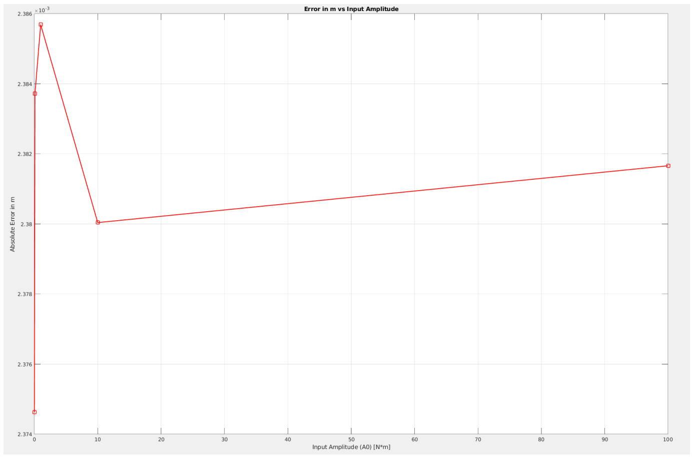


Fig. 13: Absolute estimation error vs Input Amplitude for m

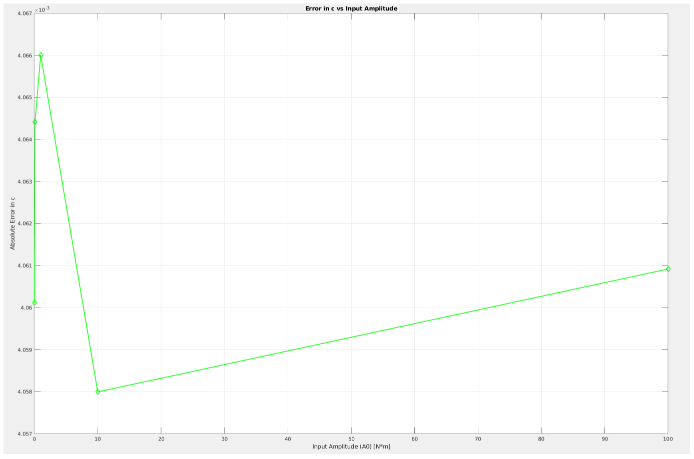


Fig. 14: Absolute estimation error vs Input Amplitude for c