Linear Least Squares for Parameter Estimation

1.0 Learning Outcomes....1
2.0 Introduction....2
3.0 Difference Equations....2
4.0 Linear Least Squares for Parameter Estimation....4
5.0 Example: Mass-Spring-Damper System (Simulation)....7
6.0 Example: DC Motor (Measured Data)**Error! Bookmark not defined.**
7.0 Summary....11
References....11
Recommended Citation....11

1.0 Learning Outcomes

After studying this ACE-Lab learning guide, you should be able to demonstrate the Learning Outcomes listed in Table 1. On completion of the ACE-Lab learning guide, revisit Table 1 and confirm whether each Learning Outcome has been achieved.

Table 1: *Learning Outcomes from the Linear Least Squares for Parameter Estimation Learning Guide*

ID	Learning Outcome	Achieved
1.	Convert a discrete-time transfer function into a difference equation using the backward time-shift operator, and explain why difference equations are used to model sampled systems.	☐
2.	Formulate a parameter estimation problem in matrix form and explain how linear least squares (LLS) estimates the model parameters by minimising the sum of squared errors.	☐
3.	Explain the practical requirements for a good estimate, including sufficient data, an exciting input and the effect of measurement noise.	☐
4.	Use MATLAB to estimate first- and second-order models from simulated and measured data, and validate them by comparing the measured and simulated responses.	☐
5.	Convert an estimated discrete-time model into a continuous-time transfer function and interpret its time constant and gain.	☐

2.0 Introduction

The previous Learning Guides assumed that the transfer function of a system was already known, for example from first principles. In practice, the parameters of a real system, such as the friction in a motor or the stiffness of a suspension, are often unknown or uncertain. An important question therefore arises:

How can a mathematical model of a system be obtained from measured input and output data?

One of the most widely used answers is linear least squares (LLS). LLS finds the model parameters that make the model's predictions as close as possible to the measured data, by minimising the sum of the squared differences between them. Building models from measured data in this way is known as system identification.

The identified model is the foundation of model-based design (MBD): controllers, such as PID controllers, are designed and tested on the model in simulation before being implemented on the hardware, which reduces development risk and cost. This Learning Guide introduces difference equations as the model structure, derives the LLS estimate, and then applies it to simulated data from a mass-spring-damper system and to measured data from a DC motor.

Why is Linear Least Squares Important?

LLS is important in control engineering because:

i. It provides a mathematical model of a real system directly from measured data, even when the physical parameters are unknown.
ii. The identified model can be used to design and test a controller in simulation before it is implemented on the hardware.
iii. It is simple to compute, has a single best solution, and can be implemented in a few lines of MATLAB.
iv. The estimated discrete-time model can be converted into a continuous-time transfer function, giving physical insight such as the time constant, natural frequency and damping ratio.

3.0 Difference Equations

A **difference equation** relates the value of a signal at one sampling instant to its values at earlier instants. For a sampled system, it expresses the present output as a linear combination of past outputs and past inputs, all of which are available from measurements. A difference equation is the discrete-time equivalent of a linear differential equation, and it is the natural model structure for a digital controller or for system identification.

Time-Shift Operators

As introduced in the Discrete-Time Modelling Learning Guide, z acts as a time-shift operator, as illustrated in Figure 1. The forward time-shift operator, z, advances a sampled signal f_k by one sampling interval:

$$z\, f_k = f_{k+1} \tag{1}$$

The forward shift requires a future value, which is not available in a real-time system, so it is rarely used in practice. The backward time-shift operator, z^{-1}, delays a signal by one sampling interval, and is used because past values are always available:

$$z^{-1} f_k = f_{k-1} \tag{2}$$

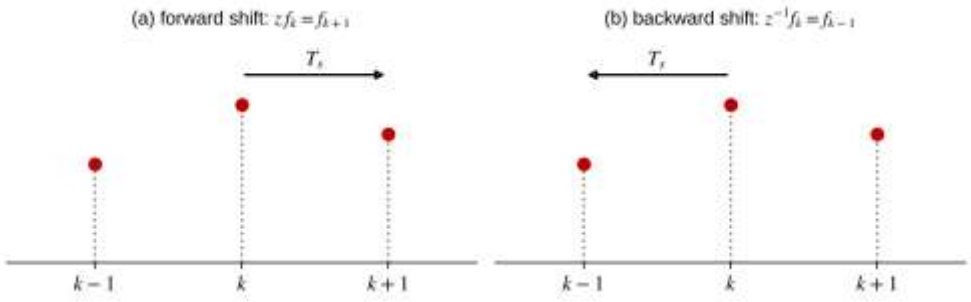


Figure 1: *Forward Time-Shift Operator (a) and Backward Time-Shift Operator (b)*

From Transfer Function to Difference Equation

Example 1: Converting a transfer function into a difference equation

Convert the following discrete-time transfer function into a difference equation:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{z + 0.5}{z^2 - 1.5z + 0.7}$$

Solution:

Dividing the numerator and denominator by the highest power of z, that is z^2, writes the transfer function in terms of z^{-1}:

$$G(z^{-1}) = \frac{z^{-1} + 0.5z^{-2}}{1 - 1.5z^{-1} + 0.7z^{-2}} = \frac{z^{-1}(1 + 0.5z^{-1})}{1 - 1.5z^{-1} + 0.7z^{-2}}$$

The factor z^{-1} in the numerator represents a delay of one sample between the input and the output. Cross-multiplying gives:

$$(1 - 1.5z^{-1} + 0.7z^{-2})y_k = (z^{-1} + 0.5z^{-2})u_k$$

Expanding, and using $z^{-1}y_k = y_{k-1}$ and $z^{-2}y_k = y_{k-2}$, gives:

$$y_k - 1.5y_{k-1} + 0.7y_{k-2} = u_{k-1} + 0.5u_{k-2}$$

Rearranging to make the present output the subject gives the difference

equation:

$$y_k = 1.5y_{k-1} - 0.7y_{k-2} + u_{k-1} + 0.5u_{k-2}$$

The present output depends only on past outputs and past inputs, so it can be calculated in real time.

General Forms

In this Learning Guide, the general second-order difference equation is written as:

$$y_k = -a_1 y_{k-1} - a_2 y_{k-2} + b_0 u_{k-1} + b_1 u_{k-2} \tag{3}$$

which corresponds to the discrete-time transfer function:

$$G(z) = \frac{b_0 z + b_1}{z^2 + a_1 z + a_2} \tag{4}$$

For Example 1, $a_1 = -1.5$, $a_2 = 0.7$, $b_0 = 1$ and $b_1 = 0.5$. Similarly, the general first-order difference equation and transfer function are:

$$y_k = -a_1 y_{k-1} + b_0 u_{k-1} \tag{5}$$

$$G(z) = \frac{b_0}{z + a_1} \tag{6}$$

The coefficients a_1, a_2, b_0 and b_1 are the unknown parameters that LLS is used to estimate.

4.0 Linear Least Squares for Parameter Estimation

Figure 2 shows a sampled-data system. A known input sequence, u_k, is applied through a zero-order hold (ZOH) to the system, and the output is sampled every T_s seconds to give the measured output sequence, y_k. Normally, a known model $G(z)$ is used to predict y_k from u_k. In parameter estimation the problem is reversed: both u_k and y_k are measured, and the unknown coefficients of the difference equation, and hence of $G(z)$, are found from the data. For a second-order model, these are a_1, a_2, b_0 and b_1. The measurements also contain noise, $e(t)$, which the method must cope with.

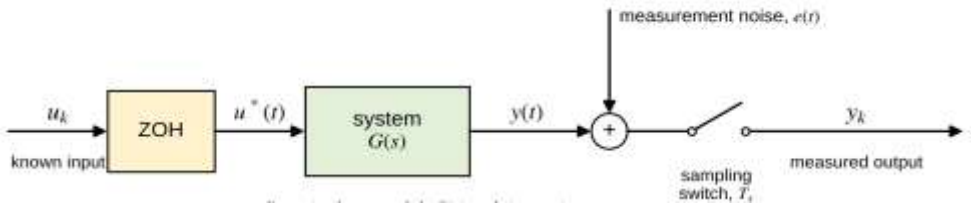


Figure 2: *Sampled-Data System with Measurement Noise*

Initial Considerations

Writing Equation (3) at four successive sampling instants, and including the measurement noise e_k, gives four equations:

$$\begin{aligned} y_k &= -a_1 y_{k-1} - a_2 y_{k-2} + b_0 u_{k-1} + b_1 u_{k-2} + e_k \\ y_{k-1} &= -a_1 y_{k-2} - a_2 y_{k-3} + b_0 u_{k-2} + b_1 u_{k-3} + e_{k-1} \\ y_{k-2} &= -a_1 y_{k-3} - a_2 y_{k-4} + b_0 u_{k-3} + b_1 u_{k-4} + e_{k-2} \\ y_{k-3} &= -a_1 y_{k-4} - a_2 y_{k-5} + b_0 u_{k-4} + b_1 u_{k-5} + e_{k-3} \end{aligned} \tag{7}$$

With four unknowns, four independent equations are, in theory, enough. Ignoring the noise for now, Equation (7) can be written in matrix form:

$$\begin{bmatrix} y_k \\ y_{k-1} \\ y_{k-2} \\ y_{k-3} \end{bmatrix} = \begin{bmatrix} -y_{k-1} & -y_{k-2} & u_{k-1} & u_{k-2} \\ -y_{k-2} & -y_{k-3} & u_{k-2} & u_{k-3} \\ -y_{k-3} & -y_{k-4} & u_{k-3} & u_{k-4} \\ -y_{k-4} & -y_{k-5} & u_{k-4} & u_{k-5} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_0 \\ b_1 \end{bmatrix} \tag{8}$$

which has the form:

$$y = \Phi\, \theta \tag{9}$$

where y is the vector of outputs, Φ is the matrix of past outputs and past inputs, and θ is the vector of unknown parameters. Here, Φ is a square 4×4 matrix, so pre-multiplying both sides by Φ^{-1} gives an estimate of the parameters, denoted by the "hat":

$$\hat{\theta} = \Phi^{-1} y = [\hat{a}_1 \quad \hat{a}_2 \quad \hat{b}_0 \quad \hat{b}_1]^T \tag{10}$$

The Least Squares Estimate

Using only four equations does not work well in practice, for two reasons:

- **Noise:** every measurement contains some noise, so an estimate based on only four samples is very sensitive to it.
- **Lack of information:** if the system is in steady state, successive measurements are almost identical ($y_{k-1} \approx y_{k-2} \approx \cdots$ and $u_{k-1} \approx u_{k-2} \approx \cdots$). The rows of Φ are then almost the same, Φ is ill-conditioned and Φ^{-1} cannot be found reliably. The input must therefore change enough to excite the dynamics of the system.

Instead, many more equations (observations) are used than there are unknowns, and the parameters are chosen to fit all of them as well as possible in a least squares sense. With N observations and p unknown parameters (where $N \gg p$, typically $N > 10p$), y becomes an $N \times 1$ vector and Φ becomes an $N \times p$ observation matrix, whose rows are the observation vectors. For the second-order model, $p = 4$ and:

$$y = \begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k-N+1} \end{bmatrix}, \qquad \Phi = \begin{bmatrix} -y_{k-1} & -y_{k-2} & u_{k-1} & u_{k-2} \\ -y_{k-2} & -y_{k-3} & u_{k-2} & u_{k-3} \\ \vdots & \vdots & \vdots & \vdots \\ -y_{k-N} & -y_{k-N-1} & u_{k-N} & u_{k-N-1} \end{bmatrix}, \qquad \theta = \begin{bmatrix} a_1 \\ a_2 \\ b_0 \\ b_1 \end{bmatrix} \tag{11}$$

A rectangular matrix cannot be inverted directly. The least squares estimate, which minimises the sum of the squared errors between y and $\Phi\theta$, is:

$$\hat{\theta} = [\Phi^T \Phi]^{-1} \Phi^T y \tag{12}$$

For a first-order model, Equation (5), $p = 2$ and:

$$\Phi = \begin{bmatrix} -y_{k-1} & u_{k-1} \\ -y_{k-2} & u_{k-2} \\ \vdots & \vdots \\ -y_{k-N} & u_{k-N} \end{bmatrix}, \qquad \theta = \begin{bmatrix} a_1 \\ b_0 \end{bmatrix} \tag{13}$$

The terms in Equation (12) have the following meanings:

Table 2: *Terms in the Least Squares Estimate*

Term	Dimension	Meaning
y, output vector	$N \times 1$	The measured outputs to be fitted
Φ, observation matrix	$N \times p$	Each row contains the past outputs and inputs for one observation
$\Phi^T \Phi$, information matrix	$p \times p$	Represents how much information the data contain about each parameter
$[\Phi^T \Phi]^{-1}$	$p \times p$	Proportional to the covariance of the estimates: larger diagonal entries mean more uncertain parameters
$[\Phi^T \Phi]^{-1} \Phi^T$	$p \times N$	The (Moore–Penrose) pseudo-inverse of Φ

LLS is the matrix equivalent of fitting the “best” straight line through a set of points, as illustrated in Figure 3. Two points are enough to draw a line, but with many noisy points, many lines could be drawn. The least squares line is the one that minimises the sum of the squared deviations, $\sum d_i^2$, of the points from the line. Because the cost is quadratic in the parameters, it has a single minimum, which Equation (12) finds directly. The estimate is unbiased when the measurement noise has zero mean and is uncorrelated from sample to sample. A full derivation of Equation (12) is given by Hsia (1977).

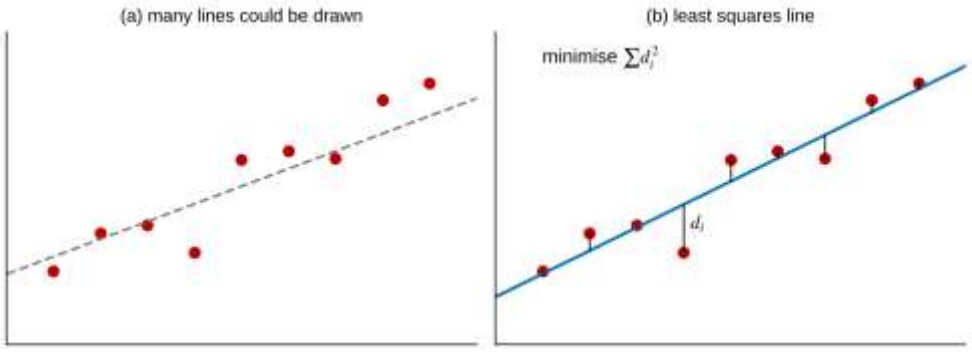


Figure 3: *Fitting a Line through Noisy Data Points (a), and the Least Squares Line that Minimises the Sum of the Squared Deviations (b)*

Choosing the Model Order

The model order is chosen from the form of the measured response. A response with no overshoot can often be modelled as first order, whereas a response with overshoot and oscillation needs at least a second-order model. Note that a second-order system with a damping ratio $\zeta \geq 1$ also has no overshoot, and it may be approximated well by a first-order model.

From Estimated Parameters to Transfer Functions

Once the parameters have been estimated, they are substituted into Equation (4) or (6) to give the discrete-time transfer function. This can then be converted into a continuous-time transfer function, for example using `d2c` in MATLAB, which reverses the ZOH discretisation covered in the Discrete-Time Modelling Learning Guide. The continuous-time form gives physical insight through the standard forms:

$$G(s) = \frac{K}{\tau s + 1} \qquad \text{or} \qquad G(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{14}$$

For a first-order model, the conversion can also be done by hand. Comparing Equation (6) with the ZOH discretisation of $K/(\tau s + 1)$ gives:

$$\tau = \frac{-T_s}{\ln(-a_1)}, \qquad K = \frac{b_0}{1 + a_1} \tag{15}$$

5.0 Example: Mass-Spring-Damper System (Simulation)

In this example, data are generated by simulating a mass-spring-damper (suspension) system in Simulink, and LLS is used to recover its model. Because the true model is known, the accuracy of the estimate can be checked. The true system has $M = 200$ kg, $B = 10$ N·s/m and $K = 5$ N/m, giving:

$$G(s) = \frac{K/M}{s^2 + (B/M)s + K/M} = \frac{0.025}{s^2 + 0.05s + 0.025} \tag{16}$$

The Simulink model is shown in Figure 4. A pulse generator (period 500 s, 50% duty cycle) provides the input, band-limited white noise with noise power n is added to the output to represent measurement noise, and the time, input and output are saved to the MATLAB workspace with a sample time of 1 s. The simulation runs for 1000 s, and the model is saved as `susp`.

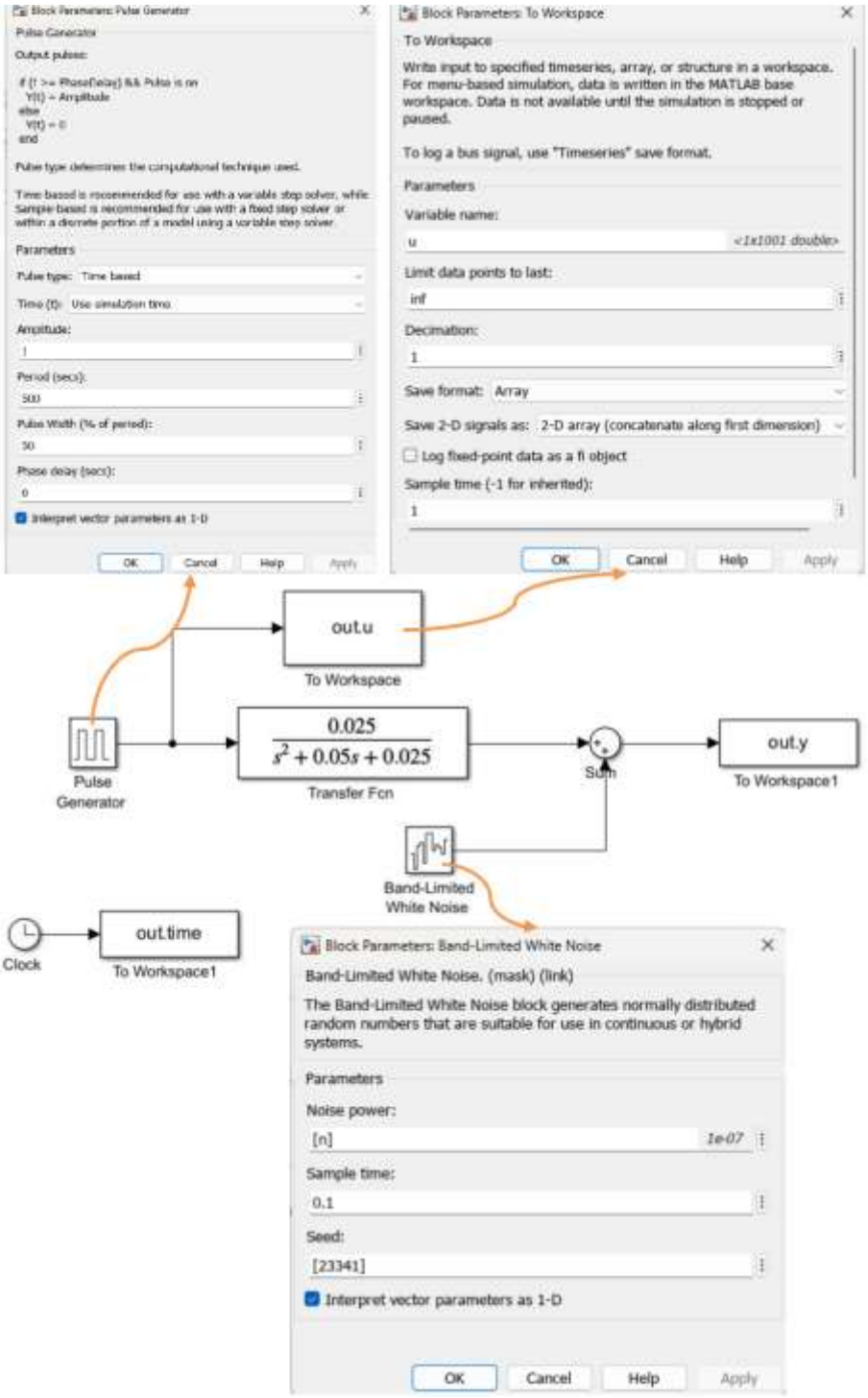


Figure 4: *Simulink Model of the Mass-Spring-Damper System, Used to Generate Input–Output Data*

MATLAB Script: estimating a second-order model of the mass-spring-damper system using LLS

```
clear; close all; clc;

%% Simulate the suspension system (representing the real system)
n = 1e-7;                           % noise power
sim('susp');
t = ans.time';  u = ans.u';  y = ans.y';

figure;
plot(t, u, t, y, 'LineWidth', 2.5); grid on;
legend('Input', 'Output', 'Location', 'Best');
xlabel('Time (s)'); ylabel('Amplitude');
title('Input and Output Signals'); set(gca, 'FontSize', 25);

%% Parameter estimation using linear least squares
N = length(u);
Y   = y(3:N)';                      % output vector
% observation matrix
Phi = [-y(2:N-1)' -y(1:N-2)' u(2:N-1)' u(1:N-2)'];
theta_hat = (Phi'*Phi) \ (Phi'*Y);  % LLS estimate

%% Estimated model
Ts = 1;                             % sampling interval [s]
% discrete-time
sysCSd = tf(theta_hat(3:4)', [1 theta_hat(1:2)'], Ts);
sysCS  = d2c(sysCSd)                % continuous-time

%% True model, for comparison
G  = tf(0.025, [1 0.05 0.025])      % M = 200, B = 10, K = 5
Gd = c2d(G, Ts);
[numd, dend] = tfdata(Gd, 'v');
theta = [dend(2:3), numd(2:3)]';    % true parameters (column)
Em = norm(theta - theta_hat)        % parameter estimation error

%% Simulate the estimated model with the same input
y_hat = zeros(size(y));
for k = 3:N
    phi_k = [-y_hat(k-1), -y_hat(k-2), u(k-1), u(k-2)];
    y_hat(k) = phi_k * theta_hat;
end

figure;
plot(t, y, t, y_hat, '--', 'LineWidth', 2.5); grid on;
legend('Measured', 'Simulated', 'Location', 'Best');
xlabel('Time (s)'); ylabel('Amplitude');
title('Measured vs Simulated Response');
set(gca, 'FontSize', 25);
```

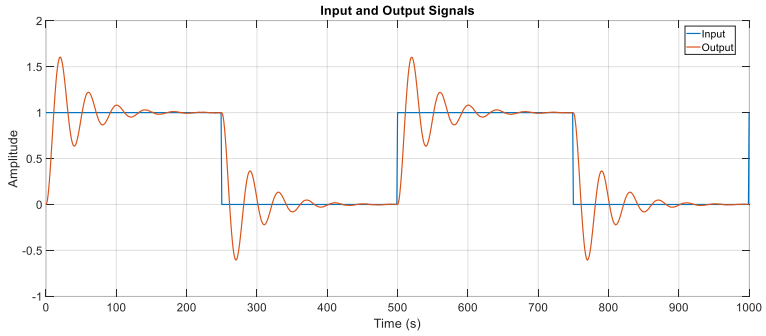
Results: the estimated continuous-time model is close to the true model:

$$G_{est}(s) = \frac{-0.001036s + 0.02523}{s^2 + 0.05305s + 0.02522} \quad \text{compared with} \quad G(s) = \frac{0.025}{s^2 + 0.05s + 0.025}$$

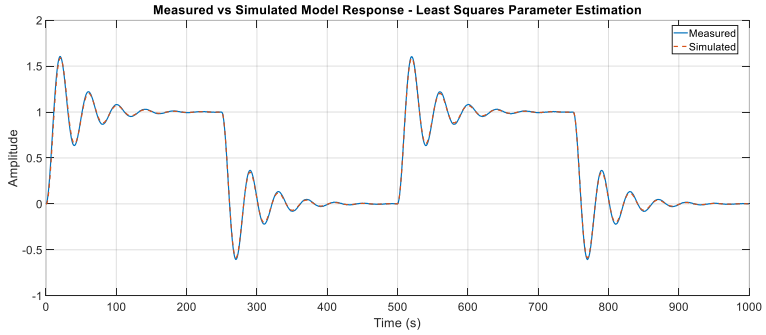
The parameter estimation error, E_m, is approximately 0.004. The small s term in the numerator of the estimated model is caused by the measurement noise. Note that `theta` must be a column vector, the same shape as `theta_hat`; otherwise MATLAB expands the subtraction into a

4×4 matrix and E_m is meaningless.

The simulated input and output data:



The measured output compared with the output of the estimated model, for the same input:



Further Investigation

Investigate how the noise power and the sampling interval affect the accuracy of the estimated model:

- **Noise power:** repeat the estimation for $n = 1 \times 10^{-6}$, 1×10^{-5} and 1×10^{-4}. Record $\hat{\theta}$ and E_m, and plot E_m against n.
- **Sampling interval:** repeat the estimation for $T_s = 0.5$, 2 and 5 s (changing the sample time of the To Workspace blocks and `Ts` in the script). Compare each estimated continuous-time model with the true model, and plot the measured and simulated outputs. Explain how T_s affects the quality of the estimate.

7.0 Summary

This Learning Guide introduced difference equations as discrete-time models that relate the present output of a system to its past outputs and inputs, using the backward time-shift operator z^{-1}. Linear least squares (LLS) was then derived as a way to estimate the unknown parameters of a difference equation from measured input–output data, by minimising the sum of the squared errors: $\hat{\theta} = [\Phi^T \Phi]^{-1} \Phi^T y$. Using many more observations than unknowns, and an input that excites the dynamics of the system, reduces the effect of measurement noise.

LLS was applied to simulated data from a mass-spring-damper system, where the estimated model closely matched the known true model, and to measured data from a DC motor, giving a first-order model with a gain of 8.77 RPM/V and a time constant of 0.52 s. The estimated discrete-time models were converted into continuous-time transfer functions for physical insight. The key progression introduced in this guide can be summarised as:

measured input–output data → observation matrix → LLS estimate → $G(z)$ → $G(s)$

The identified models form the basis of model-based controller design in the Learning Guides that follow.

References

Hsia, T. C. (1977). *System Identification: Least-Squares Methods*. Lexington, MA: Lexington Books.

Recommended Citation

If you have used this learning guide in your teaching, learning or research, please cite it using the reference below, adapting the format as required by your preferred referencing style.

Pickering, J. E. (2026). Linear Least Squares for Parameter Estimation. ACE-Lab. Teaching Notes: ACE-Lab Learning Guide, Version 1.0. https://zenodo.org/records/23325212