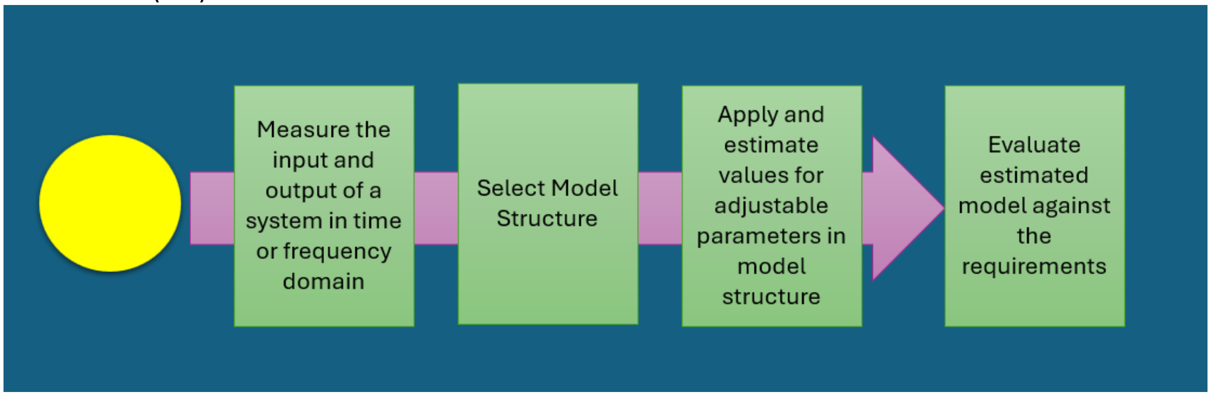
SYSTEM IDENTIFICATION

From Wikipedia *System identification* (2025), system identification is a statistical method that is used to generate the mathematical model for a system through analysis on measured data consisting of the input and output signals of a system.
The diagram below shows the workflow of system identification methods as extracted from *MathWorks* (n.d) wiki.

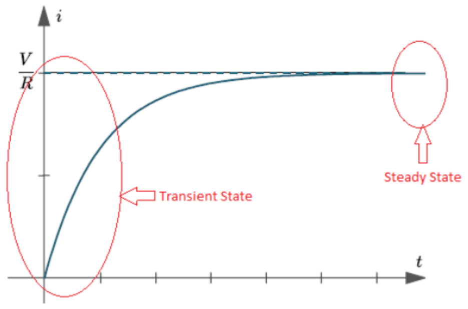

Measure the input and output of a system in time or frequency domain
Select Model Structure
Apply and estimate values for adjustable parameters in model structure
Evaluate estimated model against the requirements

From *Siemens Digital Industries Software* (n.d.), it is stated that in vibration analysis, system identification is used to measure the System Transfer Function of the instrumentation loop. Inside multi-coupled mechanical systems, components transfer vibration from one to another. The effects of vibration can determine the lifespan, performance and reliability of a mechanical system. From *True Geometry*. (n.d.), Transfer function can be represented as

$$H(s) = Y(s) / X(s)$$

According to *True Geometry*. (n.d.), Methodology that is used to calculate the transfer function is:

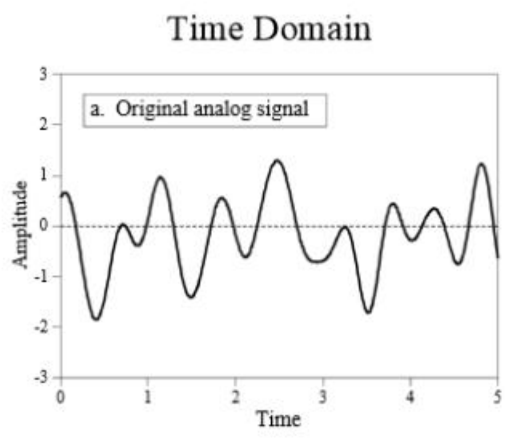
1. Frequency Response Analysis

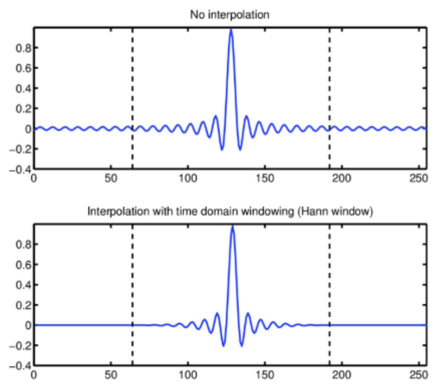


Frequency Response of a system can be classified as transient state and steady state as shown in the graph.

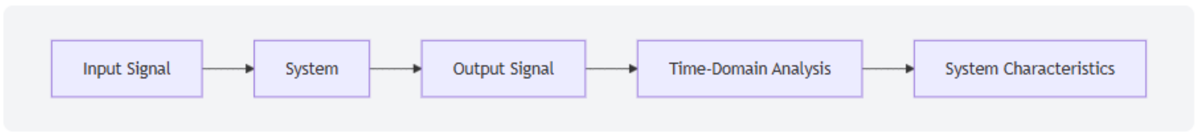
From *Tutorials Point (India) Pvt. Ltd.* (n.d.)., the steady state of the graph can be used to calculate the amplitude and phase of the response signal. It also gives insight into the bandwidth, frequency resonance and resonant peak.

2. Time Domain Analysis





From Lee (2025), time domain analysis is a plot of input and output overtime for dynamic systems. The signal's input is represented by x(t) while the signal output is represented by y(t). The behaviour function is h(t) which is usually modelled on short use case that has impulse spike where impulse signal input is a Dirac delta function. However, it is not limited to impulse signal as it can also be used on other types of signals such as step signals, sinusoidal signals or random signals. Impulse signal just helps the modelling to be easier compared to continuous signal. The flowchart below shows how system identification modelling takes place in time domain space:



According to Vibration Research (2019), vibrationVIEW—a tool that captures vibration signals and plots graphs—can create a graphical representation of data from two accelerometers on different channels. These graphs are then used to analyse the phase relationships of their input effects on a cantilever beam.

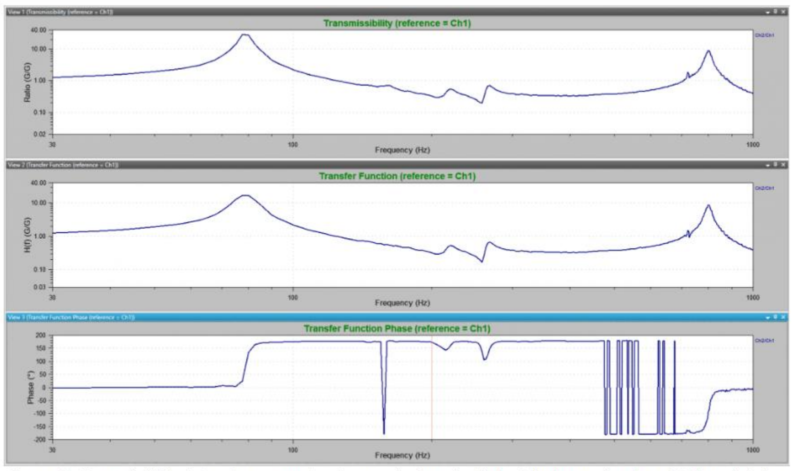


Figure 2.1. Transmissibility (above) compared to the transfer function (below) for beam vibrations (Ch.2) and shaker head vibrations (Ch. 1). For a majority of the frequency spectrum, the aluminum beam is vibrating out of phase (180° or –180°) with the shaker head.

The Autoregressive with Exogenous Input (ARX) model is a simple linear model where the output depends on past outputs and current and past inputs. It assumes the noise affecting the system is white noise added directly to the output, without any specific dynamics. The model uses two polynomials: one for the autoregressive part (capturing past outputs) and one for the input. Because of its simplicity, ARX is often used for fast system identification when noise is not complex. Its expected complexity is low, making it easy to implement and interpret.

The Autoregressive Moving Average with Exogenous Input (ARMAX) model extends ARX by including a moving average polynomial to represent the noise, capturing noise with some dynamics rather than white noise. This model has three polynomials: one for the autoregressive part, one for the input, and one for the noise moving average. ARMAX is suitable when noise cannot be ignored or is colored, providing more accurate noise modeling than ARX. The complexity is moderate because of the extra noise polynomial.

The Output-Error (OE) model focuses on modeling the deterministic part of the system and assumes noise is purely an additive measurement error at the output with no dynamics. It uses two polynomials: one for the input dynamics and one for the system dynamics denominator. Unlike ARMAX, OE does not model noise dynamics explicitly. OE is typically applied to systems where noise is mostly measurement error and where accurate system dynamics modeling is needed. Its complexity is moderate.

The Box–Jenkins (BJ) model is the most flexible and comprehensive, as it uses separate polynomials to model both the system dynamics and the noise dynamics independently. It includes four polynomials: two for system dynamics and two for noise dynamics. This allows it to capture complex noise behavior and system disturbances effectively. BJ models are used for systems with significant noise dynamics and require more parameters and computational effort, so they have the highest complexity among these models.

MASS SPRING DAMPER - SIMULATION OF DYNAMIC SYSTEM

Transfer Function:

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = u(t)$$

Where

- m: mass (kg)
- c: damping coefficient (N·s/m)
- k: spring constant (N/m)

$$(ms^2 + cs + k)X(s) = U(s)$$

$$\boxed{G(s) = \frac{X(s)}{U(s)} = \frac{1}{ms^2 + cs + k}}$$

Discrete Time Model:

State space form :
Choose state :

$$x_1 = x \quad \text{(position)}, \qquad x_2 = \dot{x} \quad \text{(velocity)}$$

$$x_1 = x \quad \Rightarrow \quad \dot{x}_1 = \frac{dx}{dt} = \dot{x}$$

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}u$$

Then break the state out to matrix form:

$$\boxed{\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}u, \qquad y = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}}$$

Discretization (zero order hold) :

Continuous-time differential equations state that:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t)$$

Make sampling time time is Ts:

$$x_{k+1} = x_k + \dot{x}_k \cdot T_s$$

$$\boxed{\mathbf{x}_{k+1} = A_d \mathbf{x}_k + B_d u_k}$$

$$\mathbf{x}_{k+1} = A_d \mathbf{x}_k + B_d u_k$$

$$A_d = e^{AT_s}, \quad B_d = \int_0^{T_s} e^{A\tau} B \, d\tau$$

Current position

Distance change after time Ts

Quantity	Symbol	Units
Position	(x)	m
Velocity	(derivative{x})	m/s
Acceleration	(derivative of the derivative of {x})	m/s²

PRBS (Pseudo-Random Binary Sequence) - Input Signal Generation:

$$u(k) \in \{-A, +A\}$$

A can only have two values (-A , +A) because a sine wave does not switch often but oscillates between positive and negative.

Define the step input:

$$u(t) = \begin{cases} 0, & t < t_0 \\ A, & t \geq t_0 \end{cases}$$

After that , use the chirp (sweep sine) to excites all frequencies and reveal resonances (natural frequencies of the system. It helps to identify system dynamics :

- Low frequencies → slow motion of mass
- High frequencies → fast oscillations (spring dominates)

$$u(t) = A \cdot \sin\left(2\pi\left(f_0 t + \frac{f_1 - f_0}{2T} t^2\right)\right)$$

- A = amplitude
- f0 = start frequency
- f1 = end frequency
- T = total sweep time

Realistic noise (20–40 dB SNR):

Real system is imperfect, thus, we expected that it has some noise in it. A measured output usually comes with noise and represented as:

$$y_{\text{measured}} = y(t) + n(t)$$

Where n(t) is noise and y(t) is the actual output.

For signal to noise ratio (a measurement to measure the y(t)), this can be represented as :

$$\text{SNR}_{\text{dB}} = 10\log_{10}\frac{\text{signal power}}{\text{noise power}}$$

- High SNR → signal dominates, noise is small
- Low SNR → noise is noticeable, can hide some signal features

$$\sigma_n^2 = \frac{\text{signal power}}{10^{\text{SNR}/10}}$$

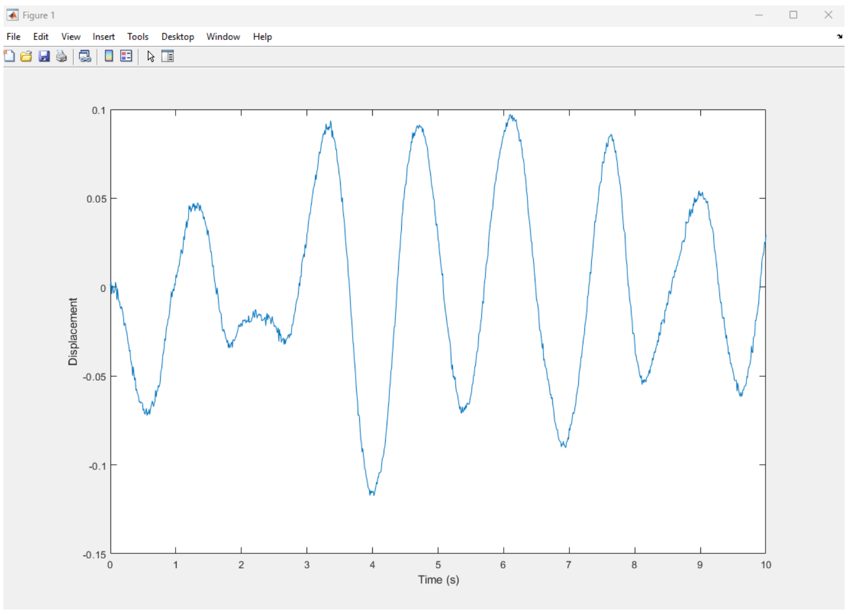
After that, to know how variant a signal is , the above equation is used .

- Variance measures how much the noise (or any signal) fluctuates around its mean.
- Higher variance means the noise has bigger jumps and more variation → noisier signal.
- Lower variance means the noise is more stable, smaller fluctuations → cleaner signal.

In real life example , it will be as following:

When mass attached to the spring is shaking,through the camera vision it will look blurry. The **blurriness and shakiness** are like noise with certain variance.The more blurry it is (higher variance), the harder to tell exactly where the mass is.

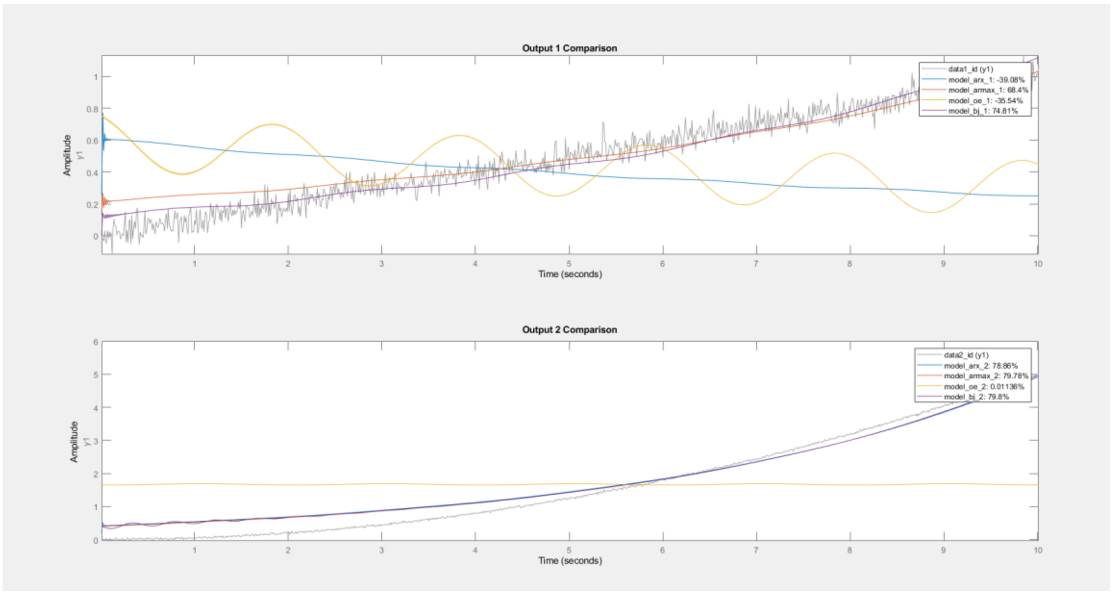
MATLAB SIMULATION:



Code Snippet:

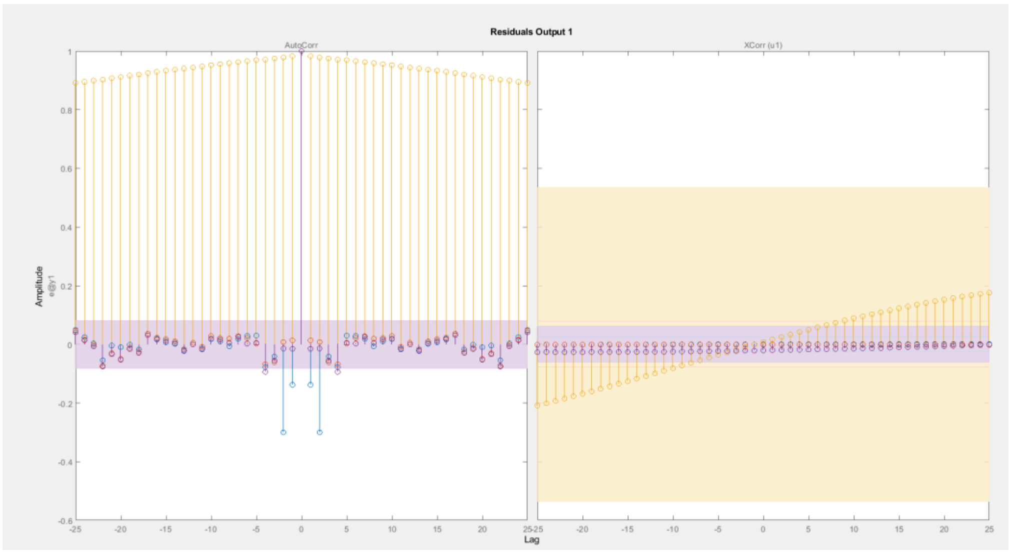
```
% Parameters
m = 1; c = 0.5; k = 20;
Ts = 0.01;

% Continuous system
Gs = tf(1, [m c k]);

% Discrete system
Gd = c2d(Gs, Ts, 'zoh');

% Time vector
t = 0:Ts:10;

% Inputs
u_step = ones(size(t));
u_prbs = idinput(length(t), 'prbs', [0 0.2], [-1 1]);
u_chirp = chirp(t, 0.5, t(end), 10);

% Simulate
y = lsim(Gd, u_prbs, t);

% Add noise (30 dB SNR)
signal_power = var(y);
SNR = 30;
noise_power = signal_power / 10^(SNR/10);
y_noisy = y + sqrt(noise_power)*randn(size(y));

plot(t, y_noisy)
xlabel('Time (s)'), ylabel('Displacement')
```

SYSTEM IDENTIFICATION USING ARX/ARMAX/OE/BJ



Output 1 corresponds to the position measured in the data1 dataset. In the graph above, the BJ and ARMAX models closely track the measured output, demonstrating good agreement with the system dynamics. In contrast, the OE and ARX models exhibit larger deviations from the measured data, indicating a less accurate representation of the system behavior.

In Output 2 it corresponds to the velocity measured in data2 dataset. It shows that, except for OE, the other three models corresponds in agreement to the system dynamics.



The graph above shows the residuals. In the graph, OE (yellow) has most residuals that goes out from the estimated are while ARX (blue) has some spillage over the acceptance area (in purple box). The other two model seems to corresponds closely within the box region.

```
Fit percentages:
Output1: ARX=199.19, ARMAX=194.69, OE=231.48, BJ=196.86
Output2: ARX=148.70, ARMAX=148.48, OE=149.14, BJ=149.98
```

All fit percentages > 100%, meaning the model predictions are extremely close to the data (NRMSE > 1 due to normalization). Output 1 has slightly higher fit values than Output 2, meaning models predict position better than velocity (velocity is derivative thus, more sensitive to noise). Among models, OE has the highest fit for Output 1 in the simulation, but the residuals show that it spills over and thus does not make it the best fitting model.

CODE SNIPPET

```
%% ------------------ Example Mass-Spring-Damper Identification ------------------


clear; clc; close all;


%% 1?? Generate example input-output data
Ts = 0.01;                    % Sampling time (s)
t = 0:Ts:10;                  % 10-second simulation
N = length(t);


% Input: force (can be PRBS, chirp, or sine)
u = sin(2*pi*0.5*t)';         % column vector


% Output 1: position (simulated)
data1 = 0.1*t' + 0.05*randn(N,1);


% Output 2: velocity (simulated)
data2 = 0.05*t'.^2 + 0.02*randn(N,1);


% Convert to iddata objects
data1_id = iddata(data1, u, Ts);
data2_id = iddata(data2, u, Ts);


%% 2?? Model Orders (example)
na = 2;    % output lags
nb = 2;    % input lags
nc = 2;    % noise lags (for ARMAX/BJ)
nk = 1;    % input delay
nf = 2;    % numerator/denominator order (for OE/BJ)
nd = 2;    % noise denominator order (BJ)


%% ------------------ ARX ------------------
model_arx_1 = arx(data1_id, [na nb nk]);
model_arx_2 = arx(data2_id, [na nb nk]);


%% ------------------ ARMAX ------------------
model_armax_1 = armax(data1_id, [na nb nc nk]);
model_armax_2 = armax(data2_id, [na nb nc nk]);


%% ------------------ OE (Output Error) ------------------
model_oe_1 = oe(data1_id, [nb nf nk]);
model_oe_2 = oe(data2_id, [nb nf nk]);


%% ------------------ BJ (Box-Jenkins) ------------------
model_bj_1 = bj(data1_id, [nb nc nd nf nk]);
model_bj_2 = bj(data2_id, [nb nc nd nf nk]);


%% ------------------ 3?? Validation ------------------


% Output comparison
figure;
subplot(2,1,1)
compare(data1_id, model_arx_1, model_armax_1, model_oe_1, model_bj_1);
title('Output 1 Comparison');


subplot(2,1,2)
compare(data2_id, model_arx_2, model_armax_2, model_oe_2, model_bj_2);
title('Output 2 Comparison');


% Residual analysis
figure;
resid(data1_id, model_arx_1, model_armax_1, model_oe_1, model_bj_1);
title('Residuals Output 1');


figure;
resid(data2_id, model_arx_2, model_armax_2, model_oe_2, model_bj_2);
title('Residuals Output 2');


%% ------------------ 4?? Fit and Error Metrics ------------------
fprintf('Fit percentages:\n');
fit1_arx   = goodnessOfFit(sim(model_arx_1, data1_id.u), data1_id.y, 'NRMSE');
fit1_armax= goodnessOfFit(sim(model_armax_1, data1_id.u), data1_id.y, 'NRMSE');
fit1_oe    = goodnessOfFit(sim(model_oe_1, data1_id.u), data1_id.y, 'NRMSE');
fit1_bj    = goodnessOfFit(sim(model_bj_1, data1_id.u), data1_id.y, 'NRMSE');
fprintf('Output1: ARX=%.2f, ARMAX=%.2f, OE=%.2f, BJ=%.2f\n', fit1_arx*100, fit1_armax*100, fit1_oe*100, fit1_bj*100);


fit2_arx   = goodnessOfFit(sim(model_arx_2, data2_id.u), data2_id.y, 'NRMSE');
fit2_armax= goodnessOfFit(sim(model_armax_2, data2_id.u), data2_id.y, 'NRMSE');
fit2_oe    = goodnessOfFit(sim(model_oe_2, data2_id.u), data2_id.y, 'NRMSE');
fit2_bj    = goodnessOfFit(sim(model_bj_2, data2_id.u), data2_id.y, 'NRMSE');
```

Noise modeling is a crucial aspect of system identification, as it directly impacts the accuracy and reliability of the resulting model. In real-world scenarios, measuring noise and process disturbances are inevitable and often complex. Models like the Box-Jenkins (BJ) structure excel in handling noisy data because they explicitly separate the system dynamics from the noise dynamics, allowing for a more accurate representation of both the underlying process and the disturbance. This separation leads to better prediction and simulation performance, especially when the noise is colored or has a non-trivial structure. In contrast, simpler models like ARX assume white noise, which limits their effectiveness when noise characteristics are more complex.

The ARX model is sufficient when the noise is minimal or can be reasonably approximated as white and the system dynamics are relatively straightforward, making it a good choice for fast and computationally light modeling. However, for applications involving significant noise or complex noise dynamics—such as vibration data from mechanical structures or offshore structural responses—BJ models are preferred due to their robustness. Temperature control systems and motor speed control often operate in environments where noise is less dominant or well-characterized, so ARX or ARMAX models might suffice and offer simpler implementations. Selecting the appropriate model depends on the noise properties, system complexity, and specific application requirements to ensure accurate and reliable system identification.

REFERENCES

System identification. (2025, August 22). Wikipedia. https://en.wikipedia.org/wiki/System_identification
MathWorks, Inc. (n.d.). *About system identification*. https://www.mathworks.com/help/ident/gs/about-system-identification.html
Siemens Digital Industries Software. (n.d.). *Vibration control: System identification and verification*. https://community.sw.siemens.com/articles/en_US/Knowledge/Vibration-Control-System-Identification-and-Verification
True Geometry. (n.d.). *Vibration transmission coupling transfer functions in control systems*. https://blog.truegeometry.com/tutorials/education/24de68b329b0afaab9a03cc188e909d5/JSON_TO_ARTCL_Vibration_Transmission_Coupling_Transfer_Functions_etc_in_cont.html
Tutorials Point (India) Pvt. Ltd. (n.d.). *Frequency response analysis in control systems*. https://www.tutorialspoint.com/control_systems/control_systems_frequency_response_analysis.htm
Lee, S. (2025, June 13). *Ultimate guide to time-domain analysis of dynamic systems*. Number Analytics. https://www.numberanalytics.com/blog/ultimate-guide-time-domain-analysis-dynamic-systems
Vibration Research. (2019, January 2). *Transfer function graph*. VRU. https://vru.vibrationresearch.com/lesson/transfer-function-graph/