Model Identification And Data Analysis II

Christian Rossi

Academic Year 2023-2024

Abstract

The course encompasses a diverse array of topics, including non-parametric system identification utilizing the subspace-based state-space approach, Kalman Filter applications for prediction, virtual-sensing, and gray-box system identification. It also covers the analysis and design of closed-loop systems using the minimum-variance approach, as well as non-linear system identification involving parametric nonlinear fitting, NARMAX models, and the optimal design of basis functions using principal component analysis. Furthermore, the course includes frequency-domain parametric estimation of models from data and extends into recursive system identification, which broadens the scope to encompass time-varying systems.

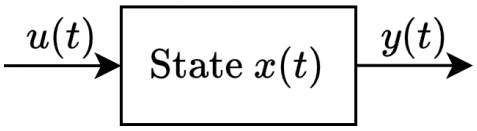
Contents

CHAPTER 1

Non-parametric system modeling

1.1 System representations

The discrete-time dynamic linear system is represented by the following block diagram:



1.1.1 State space representation

State space (or internal) representation describes a system where the state $\mathbf{x}(t)$ comprises internal variables known as states. The system can be represented as:

$$\begin{cases} \mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}u(t) \\ y(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{D}u(t) \end{cases}$$

Here, $\mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}u(t)$ is termed the state equation, a difference equation, and $y(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{D}u(t)$ is the output equation. In the case of a single input single output system, $\mathbf{F}$ (state matrix) is an $n \times n$ matrix, $\mathbf{G}$ (input matrix) is an $n \times 1$ column vector, $\mathbf{H}$ (output matrix) is a $1 \times n$ row vector, and $\mathbf{D}$ (input-output matrix) is a scalar.

Definition (*Strictly proper system*)**.** A system is termed strictly proper if the input doesn't directly affect the output.

Property 1.1.1. The system is strictly proper if the matrix $\mathbf{D}$ is zero.

Property 1.1.2. The system is stable if the eigenvalues of the matrix $\mathbf{F}$ lie within the unit circle.

The state space representation of the system is not unique. We can transform it as follows:

$$\begin{cases} \mathbf{F}' = \mathbf{T}\mathbf{F}\mathbf{T}^{-1} \\ \mathbf{G}' = \mathbf{T}\mathbf{G} \\ \mathbf{H}' = \mathbf{H}\mathbf{T}^{-1} \\ \mathbf{D}' = \mathbf{D} \end{cases}$$

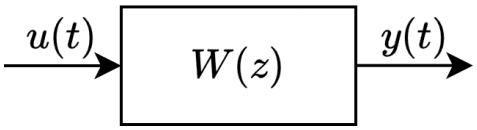
These representations are equivalent for all invertible square matrices $\mathbf{T}$, where $\mathbf{T}$ is an $n \times n$ matrix.

1.1.2 Transfer function representation

The transfer function (or external) representation is derived from the input/output time domain difference equation by utilizing the delay operator z^{-1}:

$$z^{-1}\mathbf{x}(t) = \mathbf{x}(t-1) \qquad z^{1}\mathbf{x}(t) = \mathbf{x}(t+1)$$

In general we have:



In general, the transfer function can be represented as:

$$W(z) = \frac{B(z)}{A(z)}z^{-k} = \frac{b_0 + b_1 z^{-1} + \cdots + b_p z^{-p}}{a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}} z^{-k}$$

Property 1.1.3. The system is strictly proper if the time delay k is greater than or equal to one.

1.1.3 Impulse response representation

This representation, also known as convolution of the input with the impulse response, relies on the impulse in discrete time, which has a value of one at time zero and null values at all other instants.

The response of the system at successive instants starting from zero constitutes the impulse response $\omega(t)$:

$$\text{impulse response} = \{\omega(0), \omega(1), \omega(2), \omega(3), \dots\}$$

The output of the system can be expressed as:

$$y(t) = \sum_{k=0}^{+\infty} \omega(k) u(t-k)$$

Thus, the output $y(t)$ is the convolution of the generic input signal $u(t)$ with the impulse response coefficients.

Filters The filter:

$$W(z) = \frac{z^{-1} + \frac{1}{2}z^{-2}}{1 + \frac{1}{3}z^{-1}}$$

is termed an Infinite Impulse Response filter.

The filter:

$$W(z) = z^{-1} + \frac{1}{2}z^{-2} + \frac{1}{3}z^{-3}$$

is referred to as a Finite Impulse Response filter.

1.2 Transformations

There are six possible transformations between the representations discussed earlier.

1.2.1 State space to transfer function

Given a state space representation, we can derive the transfer function representation using the formula:

$$W(z) = \mathbf{H}(z\mathbf{I} - \mathbf{F})^{-1}\mathbf{G} - \mathbf{D}$$

Alternatively, we can directly transform the state equations using the delay operator.

1.2.2 Transfer function to state space

The transformation from transfer function to state space is known as realization. The main challenge of this method is that the state space representation is not unique, leading to infinitely many equivalent realizations of a transfer function into a state space model. Assuming a monic denominator, the transfer function is given by:

$$W(z) = \frac{b_0 z^{n-1} + b_1 z^{n-2} + \cdots + b_{n-1}}{z^n + a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_n}$$

With this representation, we can find the state space representation as follows:

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 1 \\ -a_n & -a_{n-1} & \cdots & -a_1 \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} b_{n-1} \\ b_{n-2} \\ \cdots \\ b_0 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

1.2.3 Transfer function to impulse response

We can transform the transfer function into impulse response representation by performing a long polynomial division between the numerator and the denominator of the transfer function. From the quotient of the long division, we obtain $\omega(t)$ for the impulse response, and then we can express the representation in terms of the impulse response.

1.2.4 Impulse response to transfer function

Given a discrete-time signal $s(t)$, where $s(t) = 0$ if $t < 0$, the Z-transform of the signal $s(t)$ is defined as:

$$\mathcal{Z}[s(t)] = \sum_{t=0}^{+\infty} s(t) z^{-t}$$

It can be shown that:

$$W(z) = \mathcal{Z}[\omega(t)] = \sum_{t=0}^{+\infty} \omega(t) z^{-t}$$

However, in practice, this formula cannot be directly used due to the requirement of having infinite values of the impulse response and the necessity for the impulse response to be noise-free.

1.2.5 State space to impulse response

The impulse response can be determined starting from the state space representation with the initial conditions:

$$\begin{cases} x(0) = 0 \\ y(0) = 0 \end{cases}$$

Then, the system simulation is run starting from $t = 1$ onwards. The generic pattern for the impulse response calculation is:

$$\omega(t) = \begin{cases} \mathbf{D} & \text{if } t = 0 \\ \mathbf{HF}^{t-1}\mathbf{G} & \text{otherwise} \end{cases}$$

1.2.6 Impulse response to state space

The transformation from impulse response representation to state space is a crucial task in subspace-based state space system identification methods, such as the 4SID method. This method is a black-box system identification approach.

1.3 Constructive noise-free 4SID algorithm

The 4SID algorithm is utilized to derive an estimated state-space model based on a truncated impulse response acquired from system measurements.

Beginning with a finite dataset of impulse response values, the algorithm operates under the assumption of a flawless, noise-free measurement of the impulse response. While the initial discussion centers on the algorithm's application to impulse response experiments for clarity, it's important to note its versatility to accommodate various types of input excitations.

The algorithm operates as follows:

1. Construct the Hankel matrix in increasing order and assess its rank. Cease increasing the order when encountering the first matrix lacking full rank. At this point, the rank value denotes the system's order, denoted as n.

2. Select $\mathbf{H}_{n+1}$ (the first non-full rank matrix with a rank of n). Decompose $\mathbf{H}_{n+1}$ into two rectangular matrices of dimensions $(n+1)\times n$ and $n\times(n+1)$. The former matrix becomes $\mathbf{O}_{n+1}$, while the latter becomes $\mathbf{R}_{n+1}$, signifying the extended $(n+1)$ observability and reachability matrices, respectively.

3. Derive $\hat{\mathbf{H}}$, $\hat{\mathbf{G}}$, and $\hat{\mathbf{F}}$ from $\mathbf{O}_{n+1}$ and $\mathbf{R}_{n+1}$:

$$\mathbf{O}_{n+1} = \begin{bmatrix} \mathbf{H} \\ \mathbf{HF} \\ \mathbf{HF}^2 \\ \vdots \\ \mathbf{HF}^{n-1} \\ \mathbf{HF}^n \end{bmatrix} \qquad \mathbf{R}_{n+1} = \begin{bmatrix} \mathbf{G} \\ \mathbf{FG} \\ \mathbf{F}^2\mathbf{G} \\ \vdots \\ \mathbf{F}^{n-1}\mathbf{G} \\ \mathbf{F}^n\mathbf{G} \end{bmatrix}$$

   Form this, $\mathbf{H}$ and $\mathbf{G}$ are directly extracted as the first row of $\mathbf{O}_{n+1}$ and the first column of $\mathbf{R}_{n+1}$, respectively. To estimate $\hat{\mathbf{F}}$, focus on $\mathbf{O}_{n+1}$ (the same can be applied to

$\mathbf{R}_{n+1}$). Formulate two submatrices, $\mathbf{O}_1$ and $\mathbf{O}_2$, by excluding the last and first rows of $\mathbf{O}$, respectively:

$$\mathbf{O}_1 = \begin{bmatrix} \mathbf{H} \\ \mathbf{HF} \\ \mathbf{HF}^2 \\ \vdots \\ \mathbf{HF}^{n-1} \end{bmatrix} \qquad \mathbf{O}_2 = \begin{bmatrix} \mathbf{HF} \\ \mathbf{HF}^2 \\ \vdots \\ \mathbf{HF}^{n-1} \\ \mathbf{HF}^n \end{bmatrix}$$

Notably, $\mathbf{O}_1$ and $\mathbf{O}_2$ are square matrices owing to the shift invariance property. Consequently, $\mathbf{O}_2 = \mathbf{O}_1 \cdot \mathbf{F}$. As $\mathbf{O}_1$ is square and invertible (due to the full rank of $\mathbf{H}_n$ with a row removed), $\hat{\mathbf{F}}$ is determined as:

$$\hat{\mathbf{F}} = (\mathbf{O}_1)^{-1}(\mathbf{O}_2)$$

It's important to note that we've derived the state space matrices by commencing with a noise-free impulse response experiment.

1.4 Parametric 4SID algorithm

The parametric 4SID method comprises the following steps:

1. *Data collection*: gather a dataset consisting of input-output pairs:

$$\{u(1), u(2), \ldots, u(N)\} \quad \{y(1), y(2), \ldots, y(N)\}$$

2. *Model selection*: choose a parametric model, denoted as $\mathcal{M}$, which relates the output $y(t)$ to the input $u(t)$:

$$\mathcal{M} : y(t) = W(z, \vartheta) \cdot u(t)$$

3. *Performance index definition*: define a performance index to evaluate the goodness-of-fit of the chosen model. For instance, utilize the sample variance of the output error generated by the model:

$$J(\vartheta) = \frac{1}{N} \sum_{t=1}^{N} (y(t) - W(z, \vartheta)u(t))^2$$

This index serves to rank the effectiveness of different models. Specifically, if $J(\vartheta_1) < J(\vartheta_2)$, then $\mathcal{M}(\vartheta_1)$ is deemed superior to $\mathcal{M}(\vartheta_2)$.

4. *Optimization*: minimize the performance index with respect to the model parameter ϑ:

$$\hat{\vartheta}_N = \underset{\vartheta}{\operatorname{argmin}}\, J(\vartheta)$$

Consequently, $\mathcal{M}(\hat{\vartheta}_N)$ represents the optimally estimated model.

1.5 Constructive 4SID algorithm

4SID for noise-free data was initially discovered in the 1960s; however, its applicability was limited due to its inability to handle noisy data effectively. It was later rediscovered in the 1990s with the advent of Singular Value Decomposition, a pivotal tool in numerical algebra.

Considering the practical scenario, let's address the real-world problem where impulse response experiments are affected by noise.

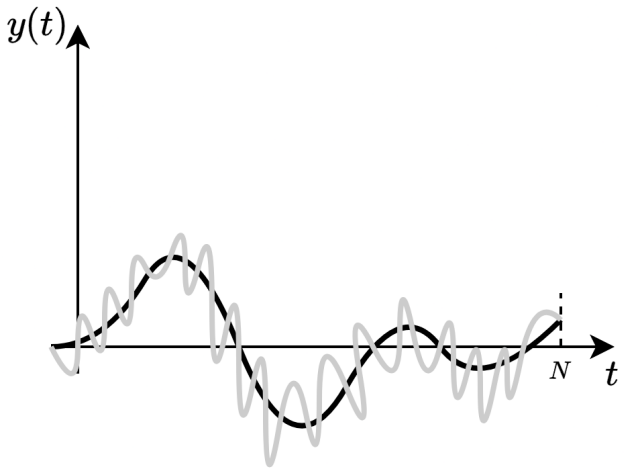


Figure 1.1: Real and ideal impulse response to a step input

In this context, each recorded impulse response data point consists of the true value along with a noise component:

$$\tilde{\omega}(t) = \omega(t) + \eta(t) \qquad t = 0, 1, 2, \ldots, N$$

Thus, the dataset is structured as follows:

$$\{\tilde{\omega}(0), \tilde{\omega}(1), \tilde{\omega}(2), \ldots, \tilde{\omega}(N)\}$$

Here, N represents a minimum of 100 samples.

1.5.1 Algorithm

Here's how the algorithm functions:

1. Construct the Hankel matrix $\tilde{\mathbf{H}}_{qd}$ using the full dataset, we arrange the data in the following manner:

$$\tilde{\mathbf{H}}_{qd} = \begin{bmatrix} \tilde{\omega}(1) & \tilde{\omega}(2) & \tilde{\omega}(3) & \cdots & \tilde{\omega}(d) \\ \tilde{\omega}(2) & \tilde{\omega}(3) & \tilde{\omega}(4) & \cdots & \tilde{\omega}(d+1) \\ \tilde{\omega}(3) & \tilde{\omega}(4) & \tilde{\omega}(5) & \cdots & \tilde{\omega}(d+2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \tilde{\omega}(q) & \tilde{\omega}(q+1) & \tilde{\omega}(q+2) & \cdots & \tilde{\omega}(q+d-1) \end{bmatrix}$$

   This matrix is diagonal with dimensions $q \times d$, where it's assumed that $q < d$. Given the relation $q + d - 1 = N$, it follows that $q = N + 1 - d$. That graphically becomes:

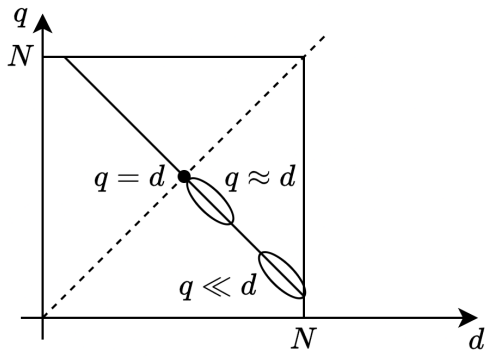


Figure 1.2: Graphical representation of the matrix dimension

When $q \ll d$, estimation quality is compromised but computational efficiency is maximized. When $q \approx d$, estimation quality is optimized but computational complexity increases. The optimal choice lies where $q > \frac{1}{2}d$, balancing estimation quality and computational efficiency. It's worth noting that there's only a minor sensitivity to the specific values of q and d as long as they adhere to this inequality.

2. Singular Value Decomposition of the matrix $\tilde{\mathbf{H}}_{qd}$ is given by:

$$\tilde{\mathbf{H}}_{qd} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T$$

Here, $\tilde{\mathbf{H}}_{qd}$ is a $q \times d$ matrix, $\tilde{\mathbf{U}}$ is a $q \times q$ unitary matrix, $\tilde{\mathbf{S}}$ is a $q \times d$ matrix, and $\tilde{\mathbf{V}}$ is a $d \times d$ unitary matrix. The matrix $\tilde{\mathbf{S}}$ is defined as:

$$\tilde{\mathbf{S}} = \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_q & \cdots & 0 \end{bmatrix}$$

Here, $\sigma_1, \sigma_2, \cdots, \sigma_q$ are the singular values of matrix $\tilde{\mathbf{H}}_{qd}$, which are real positive numbers sorted in decreasing order: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$.

3. Plot the singular values and identify the system part of the matrix.

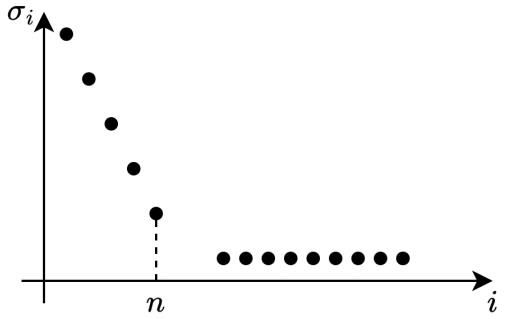


Figure 1.3: Graphical representation of the ideal singular values

In the ideal scenario depicted above, there's a clear jump between the constant and decreasing parts of the graph, indicating the estimated order of the system n. However, in real-world scenarios, the graph might look more like this:

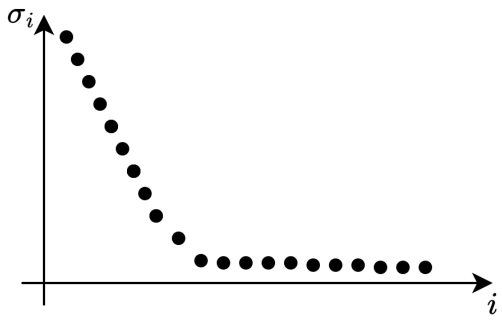


Figure 1.4: Graphical representation of the real singular values

In such cases, we don't see a distinct jump but rather a knee in the graph. The order of the system can lie at multiple points within this knee. Once we select an order, say n, we can cut the plot into two parts: the left containing the singular values of the system and the right containing the singular values of the noise. Typically, $n \ll q$.

We then proceed to cut the Hankel matrix at n to obtain:

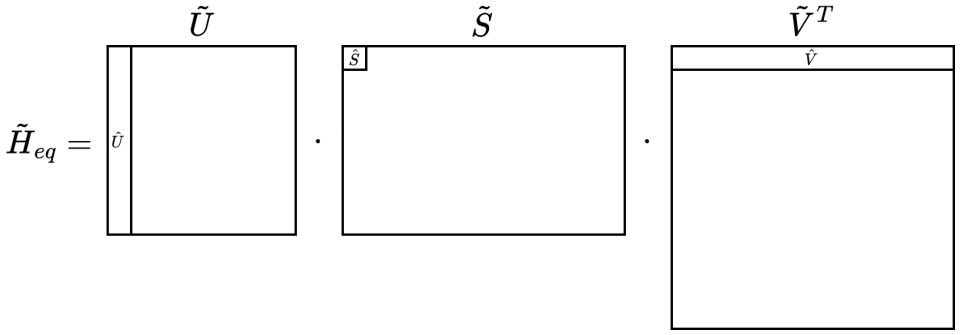


Figure 1.5: Hankel matrix cutting

Now we define the reduced matrix:

$$\hat{\mathbf{H}}_{qd} = \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^T$$

At this stage, we have:

$$\tilde{\mathbf{H}}_{qd} = \hat{\mathbf{H}}_{qd} + \mathbf{H}_{\text{res}_{qd}}$$

Where the rank of $\tilde{\mathbf{H}}_{qd}$ and $\mathbf{H}_{\text{res}_{qd}}$ is q, while the rank of $\hat{\mathbf{H}}_{qd}$ is n, indicating a significant rank reduction from q to n.

4. Estimation of matrices $\hat{\mathbf{F}}$, $\hat{\mathbf{G}}$, and $\hat{\mathbf{H}}$ from the clean matrix $\tilde{\mathbf{H}}_{qd}$. We start by decomposing $\hat{H}_{qd}$ as follows:

$$\hat{\mathbf{H}}_{qd} = \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{V}}^T = \hat{\mathbf{U}}\sqrt{\hat{\mathbf{S}}}\sqrt{\hat{\mathbf{S}}}\hat{\mathbf{V}}^T$$

We define:

$$\hat{\mathbf{O}} = \hat{\mathbf{U}}\sqrt{\hat{\mathbf{S}}}$$
$$\hat{\mathbf{R}} = \sqrt{\hat{\mathbf{S}}}\hat{\mathbf{V}}^T$$

Therefore:

$$\hat{\mathbf{H}}_{qd} = \hat{\mathbf{O}} \cdot \hat{\mathbf{R}}$$

Where $\hat{\mathbf{O}}$ and $\hat{\mathbf{R}}$ are the extended observability and controllability matrices of the system, respectively. The first row of $\hat{\mathbf{O}}$ and the first column of $\hat{\mathbf{R}}$ represent $\hat{\mathbf{H}}$ and $\hat{\mathbf{G}}$, respectively.

Next, we formulate two submatrices, $\mathbf{O}_1$ and $\mathbf{O}_2$, by excluding the last and first rows of $\hat{\mathbf{O}}$, respectively. It's worth noting that $\mathbf{O}_1$ and $\mathbf{O}_2$ are square matrices due to the shift invariance property.

Consequently, $\mathbf{O}_2 = \mathbf{O}_1 \cdot \hat{\mathbf{F}}$. Since $\mathbf{O}_1$ is rectangular and not directly invertible, $\hat{\mathbf{F}}$ can be determined as:

$$\mathbf{O}_2 = \mathbf{O}_1 \cdot \hat{\mathbf{F}} \rightarrow \mathbf{O}_1^T\mathbf{O}_2 = \mathbf{O}_1\mathbf{O}_1^T \cdot \hat{\mathbf{F}} \rightarrow \hat{\mathbf{F}} = \left(\mathbf{O}_1^T\mathbf{O}_1\right)^{-1}\mathbf{O}_1^T\mathbf{O}_2$$

Property 1.5.1. The optimality of the 4SID algorithm stems from the Singular Value Decomposition, which achieves an optimal reduction in rank.

CHAPTER 2

Frequency domain system identification

2.1 Introduction

The method at hand is straightforward and intuitive, making it a staple in control applications. It follows a classic parametric approach:

1. Experiment design, data collection, and preprocessing.
2. Selection of parametric model family $\mathcal{M}(\boldsymbol{\theta})$.
3. Definition of performance index $J(\boldsymbol{\theta})$.
4. Optimization: $\hat{\boldsymbol{\theta}} = \text{argmin}_{\boldsymbol{\theta}}\, J(\boldsymbol{\theta})$.

In frequency domain system identification, the overarching approach involves:

- Conducting a series of experiments with single-sinusoid excitations.
- Extracting a single data point representing the system's frequency response from each experiment.
- Employing optimization techniques to match the measured frequency response with the modeled frequency response, thereby determining the system parameters.

2.2 Experiment design and data gathering

In designing our experiment, the first step is to carefully select a set of excitation frequencies. These frequencies play a crucial role in characterizing the behavior of the system under study. We typically consider the following frequencies:

- The Nyquist frequency, denoted as ω_N.
- The maximum explored frequency, denoted as ω_H.

It's common practice to ensure that the maximum explored frequency ω_H is significantly lower than the Nyquist frequency ω_N.

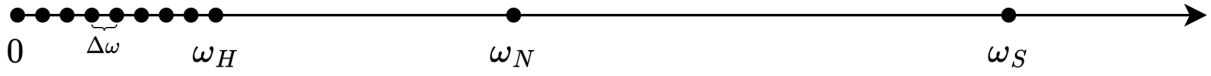


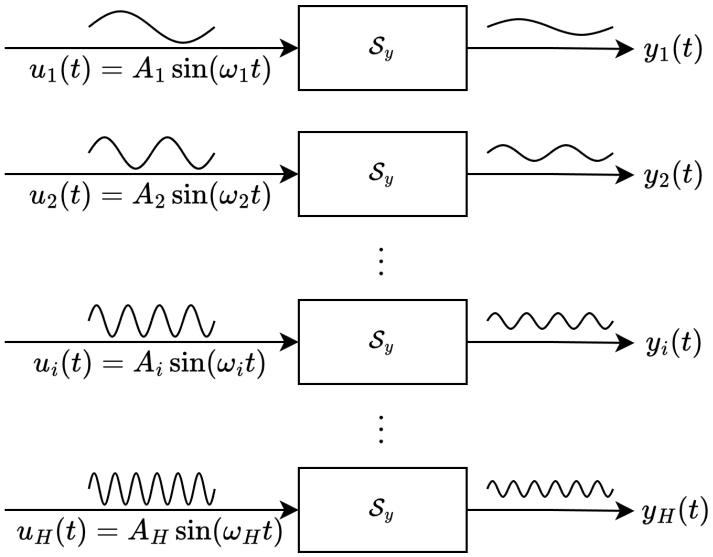
Figure 2.1: Physical system

Within our experiment design, $\Delta\omega$ represents the step size between two consecutive explored frequencies. Typically, this step size remains constant, facilitating an even distribution of frequencies from ω_1 to ω_H. Hence, the set of explored frequencies can be represented as $\{\omega_1, \ldots, \omega_H\}$.

Our primary objective is to identify an effective model that accurately represents the behavior of the system across this range of frequencies.

2.2.1 Experiments design

To comprehensively explore the behavior of the system, we conduct H independent input-output experiments. Each experiment involves applying a specific excitation signal to the system and observing its response.



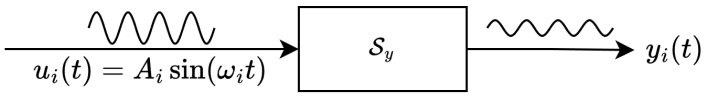
Upon completion of all experiments, we obtain H independent datasets. The i-th dataset comprises input-output pairs:

$$\{u_i(1), u_i(2), \ldots, u_i(N)\} \quad \{y_i(1), y_i(2), \ldots, y_i(N)\}$$

Choice of the amplitudes While the amplitudes of the sinusoidal inputs can be uniform across experiments, it's common practice for them to decrease gradually. This adjustment ensures that the amplitudes align with the system's actuation limits, thus providing a more realistic representation of operating conditions.

2.2.2 Data gathering

Let's delve into the specifics of the i-th experiment.



For our analysis, we operate under the assumption that the system is linear and time-invariant. This assumption allows us to employ the theorem of frequency response for linear time-invariant systems

Theorem 2.2.1 (Frequency response for linear time invarinat systems)**.** *If the input is a sinusoid of frequency ω_i, the output must also be a sinusoid of frequency ω_i, albeit with potentially different amplitude and phase.*

However, in real-world scenarios, the output $y_i(t)$ deviates from a perfect sinusoid due to various non-ideal factors:

- Measurement noise affecting the output.
- Internal noises within the system.
- Small non-linear effects, which can often be considered negligible.

Note that we are looking for a linear approximation of the system even if the system is slightly nonlinear. Let's call $\hat{y}_i(t)$ the perfect ideal sinusoid approximation of $f_i(t)$, that is the result of the experiment without the noise. We are trying to solve a classical modelling problem where our model for $y_i(t)$ is:

$$\hat{y}_i(t) = B_i \sin(\omega_i t + \varphi_i)$$

Or:

$$\hat{y}_i(t) = a_i \sin(\omega_i t) + b_i \cos(\omega_i t)$$

They are equivalent models of a sinusoid, both with two parameters to be estimated.

Identification of parameters with amplitude-amplitude model We opt for the second model due to its linearity with respect to the parameters, facilitating parameter estimation. We estimate $\hat{a}_i$ and $\hat{b}_i$ by treating it as a classical supervised parametric identification problem, with the following performance index:

$$J_N(a_i, b_i) = \frac{1}{N} \sum_{t=1}^{N} \left(y_i(t) - (a_i \sin(\omega_i t) + b_i \cos(\omega_i t))\right)^2$$

This index represents the sample variance of the modeling error. The parameters are then obtained by minimizing this index:

$$\left\{\hat{a}_i, \hat{b}_i\right\} = \underset{a_i, b_i}{\operatorname{argmin}} J_N(a_i, b_i)$$

Since the model is linear with respect to the parameters, $J_N(a_i, b_i)$ is also linear. We can therefore minimize the performance index by solving the following system of equations:

$$\begin{bmatrix} \sum_i^N \sin^2(\omega_i t) & \sum_i^N \sin(\omega_i t)\cos(\omega_i t) \\ \sum_i^N \sin(\omega_i t)\cos(\omega_i t) & \sum_i^N \cos^2(\omega_i t) \end{bmatrix} \cdot \begin{bmatrix} a_i \\ b_i \end{bmatrix} = \begin{bmatrix} \sum_i^N y_i(t)\sin(\omega_i t) \\ \sum_i^N y_i(t)\cos(\omega_i t) \end{bmatrix}$$

Thus, the solution becomes:

$$\begin{bmatrix}\hat{a}_i\\ \hat{b}_i\end{bmatrix} = \begin{bmatrix}\sum_i^N \sin^2(\omega_i t) & \sum_i^N \sin(\omega_i t)\cos(\omega_i t)\\ \sum_i^N \sin(\omega_i t)\cos(\omega_i t) & \sum_i^N \cos^2(\omega_i t)\end{bmatrix}^{-1} \cdot \begin{bmatrix}\sum_i^N y_i(t)\sin(\omega_i t)\\ \sum_i^N y_i(t)\cos(\omega_i t)\end{bmatrix}$$

This represents the Least Squares solution to the identification problem.

Identification of parameters with amplitude-phase model We consider the expression for $\hat{y}_i(t)$ in the amplitude-phase model:

$$\hat{y}_i(t) = \hat{B}_i \sin\left(\omega_i t + \hat{\varphi}_i\right) = \hat{B}_i \sin(\omega_i)\cos(\hat{\varphi}_i) + \hat{B}_i \cos(\omega_i)\sin(\hat{\varphi}_i)$$

Matching this expression to the amplitude-amplitude model, we find:

$$\begin{cases}\hat{B}_i \cos(\hat{\varphi}_i) = \hat{a}_i\\ \hat{B}_i \sin(\hat{\varphi}_i) = \hat{b}_i\end{cases}$$

This leads to:

$$\frac{\hat{a}_i}{\hat{b}_i} = \frac{\cos(\hat{\varphi}_i)}{\sin(\hat{\varphi}_i)} \rightarrow \hat{\varphi}_i = \arctan\left(\frac{\hat{b}_i}{\hat{a}_i}\right)$$

And we can also obtain $\hat{B}_i$ as:

$$\hat{B}_i = \frac{\frac{\hat{a}_i}{\cos(\hat{\varphi}_i)} + \frac{\hat{b}_i}{\sin(\hat{\varphi}_i)}}{2}$$

Thus, the two representations are equivalent.

In the end, we have:

$$y_i(t) \approx \hat{y}_i(t) = \hat{B}_i \sin(\omega_i t + \hat{\varphi}_i)$$

This cleaning procedure is repeated for all H signals. After completing this process, we acquire H complex numbers:

$$\left\{\hat{B}_i, \hat{\varphi}_i\right\} \rightarrow \frac{\hat{B}_i}{A_i} e^{j\hat{\varphi}_i}$$

We have H complex numbers, each representing an estimated value of the frequency response of the system. According to the frequency response theorem, if the input is $A_i \sin(\omega_i t)$, then the output is $\hat{B}_i \sin\left(\omega_i t + \hat{\varphi}\right)$.

These H numbers can be plotted in amplitude and phase.

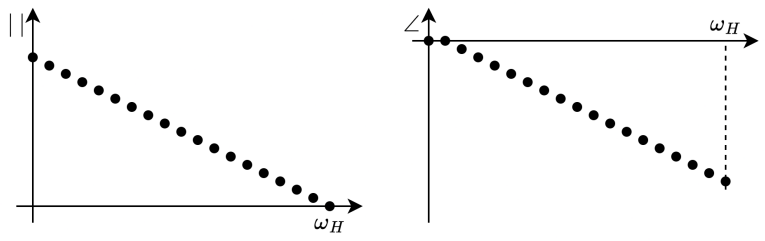


Figure 2.2: Amplitude and phase of samples H

2.2.3 Remarks

Selection of the maximum explored bandwidth Determining a meaningful bandwidth for the control system is crucial. Often, we are provided with the bandwidth of the control system, denoted as ω_C. A prudent selection for the maximum explored bandwidth would be:

$$\omega_H \approx 3 \times \omega_C$$

This choice ensures that the explored bandwidth sufficiently covers the relevant frequency range for the control system, enabling comprehensive analysis and effective model development.

Precision In certain cases, achieving higher precision in the model, particularly in specific frequency ranges such as around resonances, becomes essential.

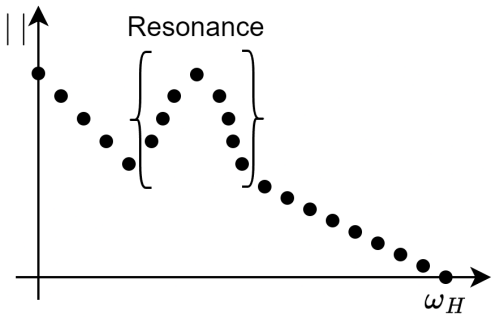


Figure 2.3: Resonance

Here, our aim is to concentrate on the region with the resonance for greater accuracy.

To address this requirement, we employ non-uniform weights. Within the critical zone, frequencies ω_i carry more weight than those outside this area. Consequently, the performance index takes the form:

$$\tilde{J}_N(\boldsymbol{\theta}) = \frac{1}{H} \sum_{i=1}^{H} \gamma_i \left(\left| W(e^{j\omega_i}, \boldsymbol{\theta}) - \frac{\hat{B}_i}{A_i} e^{j\hat{\varphi}_i} \right|^2 \right)$$

Here, γ_i denotes the weight assigned to the frequency ω_i.

Alternatively, a similar effect can be achieved by oversampling the critical frequency range.

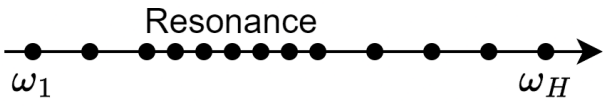


Figure 2.4: Oversampling

By conducting numerous experiments in the critical region, we obtain a non-uniformly spaced distribution, thereby enhancing precision where it is most needed.

Single experiment In some cases, instead of conducting multiple individual sinusoidal experiments, a single, extended experiment is performed wherein $u(t)$ is a sinusoidal sweep. This results in a slowly time-varying frequency sinusoidal signal that spans from ω_1 to ω_H. The performance index is computed as:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(\left|\hat{W}(e^{j\omega_i}) - W(e^{j\omega_i}, \boldsymbol{\theta})\right|^2\right)$$

This index quantifies the deviation between the measured frequency response $\hat{W}(e^{j\omega_i})$ and the model-based response $W(e^{j\omega_i}, \boldsymbol{\theta})$ across the range of frequencies considered.

2.3 Model selection

For our model, we opt for the transfer function representation:

$$\boldsymbol{\theta} : W(z, \boldsymbol{\theta}) = \frac{b_0 + b_1 z^{-1} + \cdots + b_P z^{-p}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} z^{-1}$$

Here, $\boldsymbol{\theta}$ denotes the vector encompassing all parameters of $A(z)$ and $B(z)$, respectively. It's worth noting that the determination of appropriate values for p and n can be accomplished using classic cross-validation techniques.

2.4 Performance index selection

Our chosen performance index operates in the frequency domain and is computed as follows:

$$J_M(\boldsymbol{\theta}) = \frac{1}{H}\sum_{i=1}^{H}\left(\left|W(e^{j\omega_i}, \boldsymbol{\theta}) - \frac{\hat{B}_i}{A_i}e^{j\hat{\varphi}_i}\right|^2\right)$$

This index evaluates the discrepancy between the transfer function $W(e^{j\omega_i}, \boldsymbol{\theta})$ and the estimated frequency response $\frac{\hat{B}_i}{A_i}e^{j\hat{\varphi}_i}$ across the range of frequencies considered.

2.5 Optimization

Our objective is to minimize $J_N(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}}_H = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J_N(\boldsymbol{\theta})$$

Typically, $J_N(\boldsymbol{\theta})$ isn't a quadratic function of $\boldsymbol{\theta}$, necessitating iterative optimization techniques for minimization. The ultimate outcome of this optimization process is our model $\mathcal{M}(\hat{\boldsymbol{\theta}}_N)$, expressed as:

$$\mathcal{M}(\hat{\boldsymbol{\theta}}_N) : y(t) = W(z, \hat{\boldsymbol{\theta}}_H)u(t)$$

CHAPTER 3

Software sensing with Kalman Filter

3.1 Introduction

The Kalman Filter operates based on the state space representation provided:

$$\begin{cases} \mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t) + \mathbf{v}_1(t) \\ \mathbf{y}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) + \mathbf{v}_2(t) \end{cases}$$

In this representation, two distinct noises are considered: $v_1(t)$ and $v_2(t)$. It's important to note that this model is typically provided in a white-box method, where the system dynamics are known or assumed.

Kalman Filter usage Given a dataset:

$$\{u(1), u(2), \ldots, u(N)\}, \{y(1), y(2), \ldots, y(N)\}$$

Where the current time is N, we can address the following tasks:

1. *Predicting output k steps ahead.* The objective is to forecast the output at k steps ahead:

   $$\hat{y}(N+k \mid N)$$

   This task can also be tackled using ARMAX models, resulting in the same prediction.

2. *Predicting state k steps ahead.* The aim is to estimate the state at k steps ahead:

   $$\hat{\mathbf{x}}(N+k \mid N)$$

   Unlike ARMAX models, which can only predict the output, this problem cannot be resolved using them.

3. *Filtering the present state.* The goal is to determine the state at the current time N:

   $$\hat{\mathbf{x}}(N \mid N)$$

   This constitutes a filtering challenge rather than a predictive one. ARMAX models are incapable of solving this.

4. *Gray-box system identification*: although not the primary focus of the Kalman Filter, it can provide auxiliary benefits in system identification.

3.1.1 Software sensing

Consider a generic system with $u(t)$ as inputs (actuators or control variables) and $y(t)$ as outputs (sensors, measurable variables). The system also contains internal states $x(t)$.

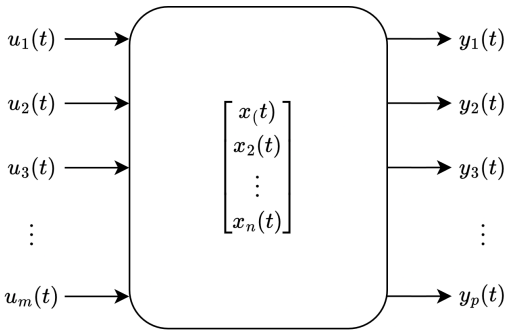


Figure 3.1: Generic multiple input multiple output system

The depicted system is multiple input multiple output with m inputs, p outputs, and n states. Typically, the number of states is much greater than the number of outputs ($n \gg p$). This is due to sensor cost considerations and the need to minimize their number.

Not all states are usually measured, but having the full state vector available is valuable for:

- Designing control algorithms.
- Monitoring the system, including fault detection and predictive maintenance.

This problem can be addressed by adding physical sensors or by developing software sensing algorithms.

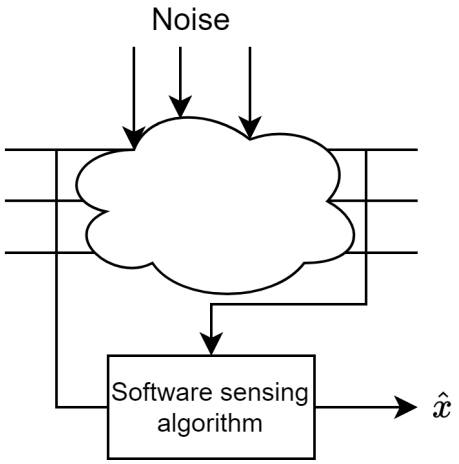


Figure 3.2: Software sensing

The software for estimation typically runs on another device with appropriate configuration.

Issues with software sensing One must verify the observability of full states from the outputs to ascertain the feasibility of software sensing. If feasible, it's crucial to assess the level of noise in the estimation. The estimation error is deemed acceptable if it's comparable to sensor noise.

3.1.2 Software sensing application

Designers often face the decision between physical and software sensing. When physical sensing is not an option, software sensing becomes the only viable choice. However, when both options are available, a balance must be struck between development and hardware costs.

The development cost remains fixed, while the sensor cost varies based on the number of sensors required:

- For small-volume applications, installing physical sensors may be more cost-effective.
- Conversely, for high-volume applications, software sensing tends to be the more economical choice.

In some instances, both methods may be employed for redundancy, particularly in safety-critical applications.

3.2 Kalman Filter for prediction

Let's examine a straightforward system devoid of external influences, characterized by linearity and time-invariance.

3.2.1 System description

We're dealing with a multiple input multiple output system denoted as $\mathcal{S}$, described by the following equations:

$$\mathcal{S}: \begin{cases} \mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t) + \mathbf{v}_1(t) \\ \mathbf{y}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) + \mathbf{v}_2(t) \end{cases}$$

In this configuration, there are n states, p inputs, and m outputs.

First White Noise The vector White Noise $\mathbf{v_1}(t)$ characterizes internal disturbances and minor model inaccuracies, with zero mean and variance V_1:

$$\mathbf{v}_1(t) = \begin{bmatrix} v_{11}(t) \\ v_{12}(t) \\ \vdots \\ v_{1n}(t) \end{bmatrix}$$

It's termed model noise. Its properties are:

1. Each element of the $\mathbf{v}_1(t)$ matrix has an expected value of zero:

$$\mathbb{E}\left[\mathbf{v}_1(t)\right] = 0$$

2. The covariance matrix of $\mathbf{v}_1(t)$ is determined as:

$$V_1 = \mathbb{E}\left[\mathbf{v}_1(t)\mathbf{v}_1(t)^T\right]$$

   This matrix is square, symmetric, and positive semi-definite.

3. Covariance is zero:

$$\mathbb{E}\left[\mathbf{v}_1(t)\mathbf{v}_1(t-\tau)^T\right] = 0 \qquad \forall t, \forall \tau \neq 0$$

Second White Noise The vector White Noise $\mathbf{v}_2(t)$ characterizes internal disturbances and minor model inaccuracies, with zero mean and variance V_2:

$$\mathbf{v}_2(t) = \begin{bmatrix} v_{21}(t) \\ v_{22}(t) \\ \vdots \\ v_{2p}(t) \end{bmatrix}$$

It's termed sensor noise. Its properties are:

1. Each element of the $\mathbf{v}_2(t)$ matrix has an expected value of zero:

$$\mathbb{E}\left[\mathbf{v}_2(t)\right] = 0$$

2. The covariance matrix of $\mathbf{v}_2(t)$ is determined as:

$$V_2 = \mathbb{E}\left[\mathbf{v}_2(t)\mathbf{v}_2(t)^T\right]$$

   This matrix is square, symmetric, and assumed to be positive definite, a requirement for applications involving the Differential Riccati Equation.

3. Covariance is zero:

$$\mathbb{E}\left[\mathbf{v}_2(t)\mathbf{v}_2(t-\tau)^T\right] = 0 \qquad \forall t, \forall \tau \neq 0$$

Assumptions Given the dynamical nature of the system, certain assumptions about initial conditions are necessary. We assume that:

- The expected value of the initial state $\mathbf{x}(1)$ is $\mathbf{x}_0$:

$$\mathbb{E}\left[\mathbf{x}(1)\right] = \mathbf{x}_0$$

- The expected value of the outer product of the difference between the initial state and x_0 with its transpose is P_0, where P_0 is non-negative:

$$\mathbb{E}\left[\left(\mathbf{x}(1) - \mathbf{x}_0\right)\left(\mathbf{x}(1) - \mathbf{x}_0^T\right)\right] = \mathbf{P}_0 \geq 0$$

This approach provides a probabilistic description of the initial state, encompassing both its mean value and variance. If $\mathbf{P}_0 = 0$, it implies precise knowledge of the initial state.

Correlation between the two White Noises We make the assumption that the correlation between $\mathbf{v}_1(t)$ and $\mathbf{v}_2(t-\tau)$ is given by:

$$\mathbb{E} = \left[\mathbf{v}_1(t)\mathbf{v}_2(t-\tau)^T\right] = \begin{cases} 0 & \text{if } \tau \neq 0 \\ V_{12} & \text{if } \tau = 0 \end{cases}$$

Typically, $V_{12} = 0$.

Correlation between the White Noises and the initial state It's assumed that both White Noises $\mathbf{v}_1(t)$ and $\mathbf{v}_2(t)$ are uncorrelated with the initial state $\mathbf{x}(1)$. This technical assumption ensures the theoretical optimality of the Kalman Filter.

3.2.2 One step predictor

We utilize the following solution:

$$\begin{cases} \hat{\mathbf{x}}(t+1 \mid t) = \mathbf{F}\hat{\mathbf{x}}(t \mid t-1) + \mathbf{K}(t)\mathbf{e}(t) \\ \hat{\mathbf{y}}(t+1 \mid t) = \mathbf{H}\hat{\mathbf{x}}(t \mid t-1) \\ \mathbf{e}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t \mid t-1) \\ \mathbf{K}(t) = \left(\mathbf{F}\mathbf{P}(t)\mathbf{H}^T + V_{12}\right)\left(\mathbf{H}\mathbf{P}(t)\mathbf{H}^T + V_2\right)^{-1} \\ \mathbf{P}(t+1) = \left(\mathbf{F}\mathbf{P}(t)\mathbf{F}^T + V_1\right) - \left(\mathbf{F}\mathbf{P}(t)\mathbf{H}^T + V_{12}\right)\left(\mathbf{H}\mathbf{P}(t)\mathbf{H}^T + V_2\right)^{-1}\left(\mathbf{F}\mathbf{P}(t)\mathbf{H}^T + V_{12}\right)^T \end{cases}$$

Each equation serves a distinct purpose:

1. *State equation*: predicts the next state based on the current state and the system's dynamics.
2. *Output equation*: predicts the next output based on the predicted state.
3. *Error equation*: computes the prediction error between the actual output and the predicted output.
4. *Gain equation*: computes the optimal Kalman gain.
5. *Differential Riccati Equation*: computes the covariance matrix of the one-step prediction error of the states.

To ensure the solution's validity, we require initial conditions:

$$\begin{cases} \hat{\mathbf{x}}(1 \mid 0) = \mathbf{x}_0 \\ \mathbf{P}(1) = \mathbf{P}_0 \end{cases}$$

Differential Riccati Equation and gain structure The structure of $\mathbf{K}(t)$ and the Differential Riccati Equation is block-wise. The three essential blocks involved are:

- State block: $\mathbf{F}\mathbf{P}(t)\mathbf{F}^T + V_1$.
- Output block: $\mathbf{H}\mathbf{P}(t)\mathbf{H}^T + V_2$.
- Mix block: $\mathbf{F}\mathbf{P}(t)\mathbf{H}^T + V_{12}$.

Differential Riccati Equation The Differential Riccati Equation a specialized type of nonlinear matrix difference equation. It represents an autonomous system devoid of inputs:

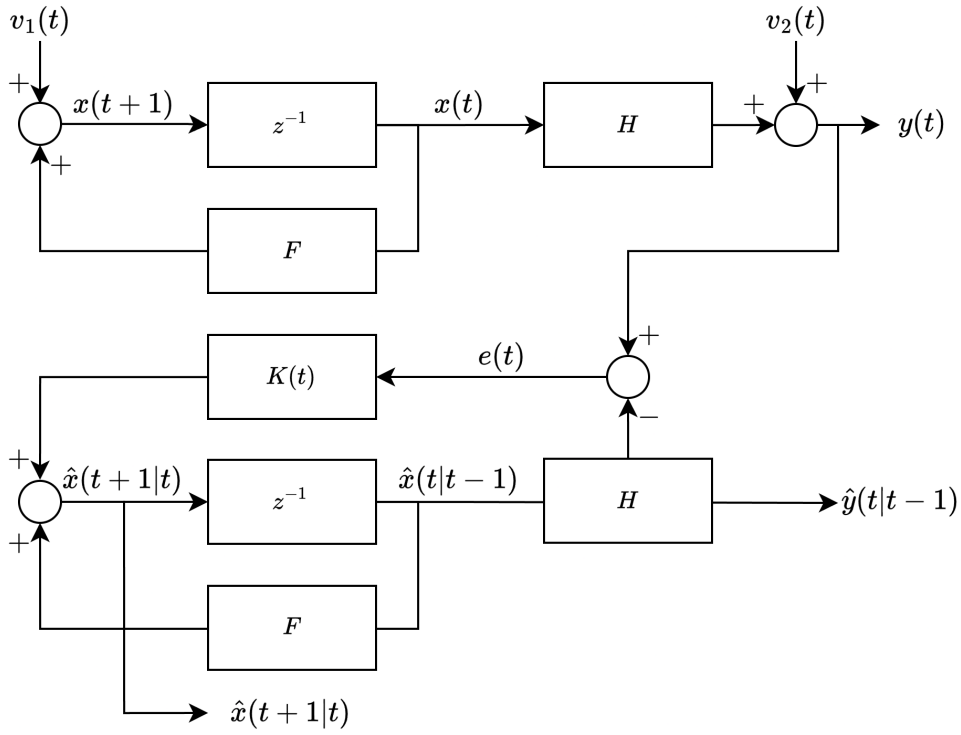
$$\begin{cases} \mathbf{P}(t+1) = f(\mathbf{P}(t)) \\ \mathbf{P}(1) = \mathbf{P}_0 \end{cases}$$

To ensure the existence of the Differential Riccati Equation at all time instants, $\mathbf{H}\mathbf{P}(t)\mathbf{H}^T + V_2$ must be invertible. Given our assumption that V_2 is positive semi-definite, the full matrix becomes positive definite and thus invertible.

Prediction error The matrix $\mathbf{P}(t)$ is a symmetric $n \times n$ matrix representing the covariance matrix of the one-step prediction error of the states:

$$\mathbf{P}(t) = \mathbb{E}\left[\left(\mathbf{x}(t) - \hat{\mathbf{x}}(t \mid t-1)\right)\left(\mathbf{x}(t) - \hat{\mathbf{x}}(t \mid t-1)\right)^T\right] = \mathrm{Var}\left[\mathbf{x}(t) - \hat{\mathbf{x}}(t \mid t-1)\right]$$

Graphical representation The block diagram of the system and the Kalman Filter is depicted below:



The structure of the Kalman Filter is straightforward and intuitive:

- It mimics the system's structure, excluding the two unmeasurable noises.
- It computes the real output and the estimated output, generating the output error:

$$\mathbf{e}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t \mid t-1)$$

- It corrects the state equation in a feedback loop using a given $\mathbf{K}(t)$ applied on $\mathbf{e}(t)$.

In essence, the Kalman Filter aims to track the real system with a feedback correction based on the output error. This concept, known as a state observer, predates Kalman. Kalman's contribution lies in providing the formula for the optimal correction gain $\mathbf{K}(t)$.

The optimal gain $\mathbf{K}(t)$ plays a crucial role:

- If $\mathbf{K}(t)$ is too small, it underutilizes the information in $\mathbf{e}(t)$.
- If $\mathbf{K}(t)$ is too large, it amplifies noises excessively, potentially leading to instability.

Since $\mathbf{K}(t)$ is an $n \times p$ matrix, empirical tuning becomes impractical.

3.3 Extensions

We will now explore the potential application of the one-step ahead predictor formula with the Kalman Filter in various contexts beyond its original domain.

3.3.1 Multi-step prediction

Assuming that $\hat{\mathbf{x}}(t+1 \mid t)$ is known, the k-step ahead prediction can be straightforwardly computed. Given $\hat{\mathbf{x}}(t+1 \mid t)$, we can derive:

$$\begin{cases} \hat{\mathbf{x}}(t+k \mid t) = \mathbf{F}^{k-1}\hat{\mathbf{x}}(t+1 \mid t) \\ \hat{\mathbf{y}}(t+k \mid t) = \mathbf{H}\hat{\mathbf{x}}(t+k \mid t) \end{cases}$$

3.3.2 Filtering

The filtering process, denoted as $\hat{\mathbf{x}}(t \mid t)$, aims to estimate the current state of the system given available measurements up to time t. Assuming that $\hat{\mathbf{x}}(t+1 \mid t)$ is known, we can compute $\hat{\mathbf{x}}(t \mid t)$ using different formulations based on the properties of the system matrix.

Straightforward solution If the matrix $\mathbf{F}$ is invertible, we have a straightforward solution:

$$\hat{\mathbf{x}}(t \mid t) = \mathbf{F}^{-1}\hat{\mathbf{x}}(t+1 \mid t)$$

Filter formulation In special cases where $\mathbf{F}$ is not invertible, we use the filter formulation of the Kalman Filter:

$$\begin{cases} \hat{\mathbf{x}}(t \mid t) = \mathbf{F}\hat{\mathbf{x}}(t-1 \mid t-1) + \mathbf{K}_0(t)\mathbf{e}(t) \\ \hat{\mathbf{y}}(t \mid t-1) = \mathbf{H}\hat{\mathbf{x}}(t \mid t-1) \\ \mathbf{e}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t \mid t-1) \\ \mathbf{K}_0(t) = \left(\mathbf{P}(t)\mathbf{H}^T\right)\left(\mathbf{H}\mathbf{P}(t)\mathbf{H}^T + V_2\right)^{-1} \\ \mathbf{P}(t+1) = \left(\mathbf{F}\mathbf{P}(t)\mathbf{F}^T + V_1\right) - \left(\mathbf{F}\mathbf{P}(t)\mathbf{H}^T + V_{12}\right)\left(\mathbf{H}\mathbf{P}(t)\mathbf{H}^T + V_2\right)^{-1}\left(\mathbf{F}\mathbf{P}(t)\mathbf{H}^T + V_{12}\right)^T \end{cases}$$

The initial condition for the filter is $\hat{\mathbf{x}}(1 \mid 1) = \mathbf{x}_0$. The equation for $\mathbf{K}_0(t)$ is referred to as the filter gain. This formula is valid only when $V_{12} = 0$, which is typically true in practice.

Gains There's a distinction between two gains when $V_{12} = 0$:

- Prediction gain: $\mathbf{K}(t) = \left(\mathbf{F}\mathbf{P}(t)\mathbf{H}^T + V_{12}\right)\left(\mathbf{F}\mathbf{P}(t)\mathbf{H}^T + V_2\right)^{-1}$.
- Filter gain: $\mathbf{K}_0(t) = \left(\mathbf{P}(t)\mathbf{H}^T\right)\left(\mathbf{H}\mathbf{P}(t)\mathbf{H}^T + V_2\right)^{-1}$.

The primary difference lies in the initial $\mathbf{F}$, which is absent in the filter gain equation.

3.3.3 Exogenous input

Graphically, the Kalman Filtering process with an eXogenous input $\mathbf{G}\mathbf{u}(t)$ is illustrated as follows:

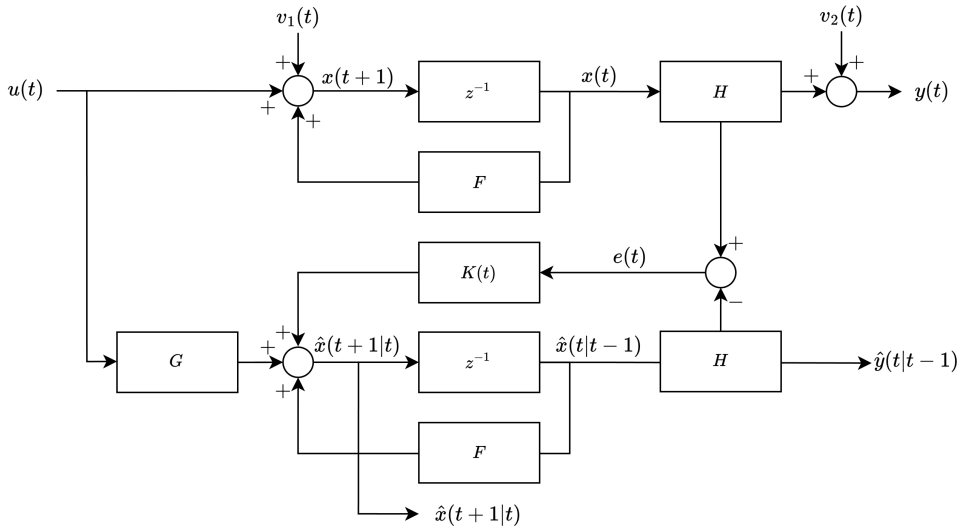


Figure 3.3: Kalman Filtering with eXogenous input

Notably, the $\mathbf{K}(t)$ and $\mathbf{P}(t)$ terms remain unchanged from the formulas without $\mathbf{Gu}(t)$. This intuitively makes sense because $\mathbf{Gu}(t)$ does not introduce additional uncertainty into the system; rather, it represents a deterministic component added to the system. Consequently, the state prediction error $\mathbf{P}(t)$ remains unaffected.

3.3.4 Time-variant systems

Consider a system represented by the following equations, where all matrices are time dependent:

$$\begin{cases} \mathbf{x}(t+1) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}(t)\mathbf{u}(t) + \mathbf{v}_1(t) \\ \mathbf{y}(t) = \mathbf{H}(t)\mathbf{x}(t) + \mathbf{v}_2(t) \end{cases}$$

In this scenario, even with time-variance in the system matrices, the Kalman Filter equations remain identical to those of time-invariant systems.

3.4 Asymptotic solution of the Kalman Filter

In this section, we delve into the asymptotic or steady-state solution of the Kalman Filter.

It's important to note that although the system $\mathcal{S}$ may be linear time-invariant, the Kalman Filter itself is a linear time-variant system due to the presence of the gain $\mathbf{K}(t)$. This introduces two significant challenges:

1. Ensuring and verifying the asymptotic stability of the Kalman Filter becomes notably intricate. Even when all eigenvalues of the matrix $\mathbf{F}(t)$ remain strictly inside the unit circle at any given time t, stability isn't guaranteed. Consequently, assessing stability, especially for higher-order systems, can be exceptionally complex.

2. There's a computational challenge. With each sampling instance, we need to update the computations for $\mathbf{K}(t)$ and $\mathbf{P}(t)$. Notably, solving the Differential Riccati Equation necessitates inverting a $p \times p$ matrix.

These inherent difficulties often lead to the adoption and implementation of the asymptotic version of the Kalman Filter in practical applications.

3.4.1 Time invariant Kalman Filter

In this simplified version, the fundamental concept is that if: $\mathbf{P}(t) \to \bar{\mathbf{P}}$, then:

$$\mathbf{K}(t) \to \bar{\mathbf{K}}$$

Thus, we can utilize this constant correction gain $\bar{\mathbf{K}}$ to convert the Kalman Filter into a linear time-invariant system.

Before delving into the existence of $\bar{\mathbf{K}}$, let's examine the asymptotic stability of the Kalman Filter, assuming the existence of $\bar{\mathbf{K}}$.

Consider the core state equation of the Kalman Filter:

$$\hat{\mathbf{x}}(t+1 \mid t) = \left(\mathbf{F} - \bar{\mathbf{K}}\mathbf{H}\right) \hat{\mathbf{x}}(t \mid t-1) + \bar{\mathbf{K}}\mathbf{y}(t)$$

Here, the state matrix of the Kalman Filter is $\mathbf{F} - \bar{\mathbf{K}}\mathbf{H}$. Hence, the Kalman Filter is asymptotically stable if and only if all eigenvalues of $\mathbf{F} - \bar{\mathbf{K}}\mathbf{H}$ lie strictly within the unit circle. Particularly:

- The stability of the system $\mathcal{S}$ is contingent upon the matrix $\mathbf{F}$.
- The stability of the Kalman Filter hinges on $\mathbf{F} - \bar{\mathbf{K}}\mathbf{H}$.

Thus, the Kalman Filter can achieve asymptotic stability even if the system $\mathcal{S}$ itself is unstable.

3.5 Differential Riccati Equation equilibrium

Since $\mathbf{K}(t) = \left(\mathbf{F}\mathbf{P}(t)\mathbf{H}^T + V_{12}\right)\left(\mathbf{H}\mathbf{P}(t)\mathbf{H}^T + V_2\right)^{-1}$, then a constant gain $\bar{\mathbf{K}}$ is attainable only if the Differential Riccati Equation possesses an equilibrium point $\bar{\mathbf{P}}$. In addressing the equilibrium of the Differential Riccati Equation, let's delve into the Algebraic Riccati Equation given by:

$$\bar{\mathbf{P}} = \left(\mathbf{F}\bar{\mathbf{P}}\mathbf{F}^T + V_1\right) - \left(\mathbf{F}\bar{\mathbf{P}}\mathbf{H}^T + V_{12}\right)\left(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^T + V_2\right)^{-1}\left(\mathbf{F}\bar{\mathbf{P}}\mathbf{H}^T + V_{12}\right)^T$$

This nonlinear matrix static equation, known as Algebraic Riccati Equation, is crucial for determining the existence of a steady-state solution for the Differential Riccati Equation.

To address the equilibrium of the Algebraic Riccati Equation and subsequently the Differential Riccati Equation, we need to prove three critical aspects:

1. *Existence*: demonstrating that the Algebraic Riccati Equation possesses semi-definite positive solutions.
2. *Convergence*: proving the convergence of the Discrete Differential Riccati Equation to $\bar{\mathbf{P}}$.
3. *Stability*: verifying the asymptotic stability of the corresponding Kalman Filter, which is equivalent to ensuring all eigenvalues of $\mathbf{F} - \bar{\mathbf{K}}\mathbf{H}$ lie strictly inside the unit circle.

Although these proofs are challenging, we can rely on two key theorems that provide sufficient conditions for the requested proofs:

Theorem 3.5.1 (First asymptotic Kalman Filter)**.** *If $V_{12} = 0$ and the system is asymptotically stable, then:*

- *Algebraic Riccati Equation possesses one and only one semi-positive solution $\bar{\mathbf{P}} \geq 0$.*

- *The corresponding* $\bar{\mathbf{K}}$ *ensures the asymptotic stability of the Kalman Filter.*
- *Differential Riccati Equation converges to* $\bar{\mathbf{P}}$ *for all* $\mathbf{P}_0 \geq 0$.

In the equation for the state:

$$\mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t) + \mathbf{v}_1(t) \qquad \mathbf{v}_1(t) \sim WN(0, V_1)$$

to introduce another noise source, we require a special type of controllability from the noise $\mathbf{v}_1(t)$ since it serves as an input for the states $\mathbf{x}(t)$. We can reformulate the same equation as:

$$\mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t) + \mathbf{\Gamma}_\omega(t) \qquad \omega \sim WN(0, \mathbf{I})$$

Here, $\mathbf{\Gamma}_\omega(t)$ represents the factorization of V_1, such that $\mathbf{\Gamma}\mathbf{\Gamma}^T = V_1$.

We can assert that the state is controllable from noise $\mathbf{v}_1(t)$ if and only if the controllability matrix is full rank:

$$\operatorname{rank}(\mathbf{R}) = \operatorname{rank}\left(\begin{bmatrix}\mathbf{\Gamma} & \mathbf{F}\mathbf{\Gamma} & \mathbf{F}^2\mathbf{\Gamma} & \cdots & \mathbf{F}^n\mathbf{\Gamma}\end{bmatrix}\right) = n$$

Here, n denotes the dimension of the state space.

Theorem 3.5.2 (Second asymptotic Kalman Filter)**.** *If* $V_{12} = 0$*, the pair* $(\mathbf{F}, \mathbf{H})$ *is fully observable, and the pair* $(\mathbf{F}, \mathbf{\Gamma})$ *is fully controllable, then:*

- *Algebraic Riccati Equation possesses one and only one solution* $\bar{\mathbf{P}} \geq 0$.
- *The corresponding* $\bar{\mathbf{K}}$ *ensures the asymptotic stability of the Kalman Filter.*
- *Differential Riccati Equation converges to* $\bar{\mathbf{P}}$ *for all* $\mathbf{P}_0 \geq 0$.

These theorems are particularly valuable in practice as they allow us to bypass the intricate convergence analysis of the Differential Riccati Equation for systems with $n > 1$.

3.6 Kalman Filter with non-White Noise

The conventional formulas of Kalman Filters are typically employed under the assumption that both $\mathbf{v}_1(t)$ and $\mathbf{v}_2(t)$ are White Noises. However, this assumption can be overly restrictive, as in many real-world scenarios, noise is not white. A common workaround to this limitation is a general technique known as model extension.

3.7 Extended Kalman Filter

Consider a state-space model with nonlinear dynamics:

$$\mathcal{S} : \begin{cases} \mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{v}_1(t) \\ \mathbf{y}(t) = h(\mathbf{x}(t)) + \mathbf{v}_2(t) \end{cases}$$

Here, $f(\cdot)$ and $h(\cdot)$ are nonlinear functions of class C^1 or higher.

The Extended Kalman Filter is a variant of the Kalman Filter designed to handle nonlinear dynamics and measurements. It approximates the nonlinear system by linearizing it at each

time step around the current estimate of the state. To illustrate the idea, let's consider the following block diagram of the Extended Kalman Filter:

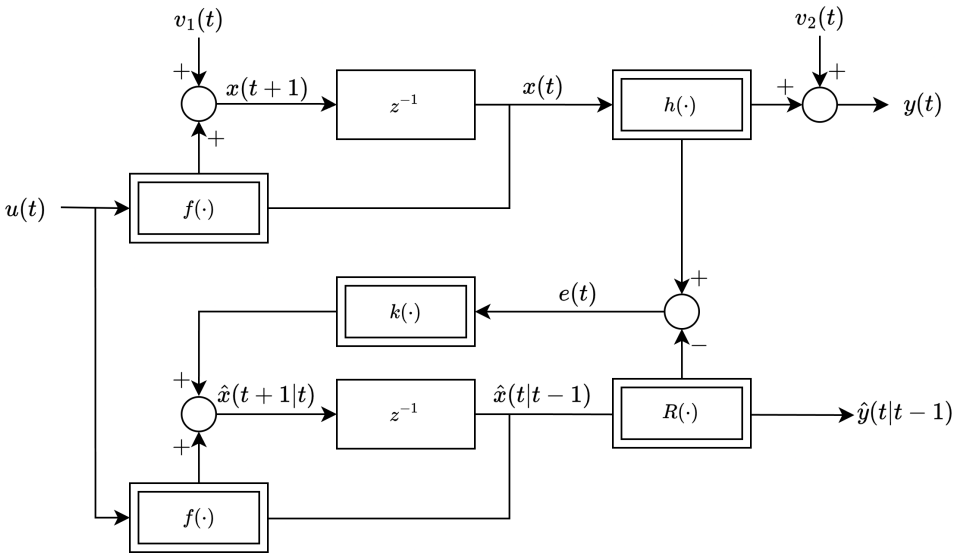


Figure 3.4: Extended Kalman Filter block scheme

To establish the feedback correction gain, we have two distinct options:

1. The gain is structured as a nonlinear function, denoted as $\mathbf{K}(\cdot)$, yielding an output of $\mathbf{K}(\mathbf{e}(t))$.

2. Alternatively, the gain takes the form of a linear time-variant block akin to the Kalman Filter, producing an output of $\mathbf{K}(t)\mathbf{e}(t)$.

The Extended Kalman Filter embraces the latter approach, albeit less intuitive, it proves more efficacious by leveraging established theory from traditional Kalman Filters.

To easily compute the gain $\mathbf{K}(t)$ using the Kalman Filter formula, we utilize the usual equations. To implement this formula, we require a method for computing $\mathbf{F}(t)$ and $\mathbf{H}(t)$ at each sampling instant:

$$\mathbf{F}(t) = \frac{\partial f(\mathbf{x}(t), \mathbf{u}(t))}{\partial \mathbf{x}(t)} \qquad \mathbf{H}(t) = \frac{\partial h(\mathbf{x}(t))}{\partial \mathbf{x}(t)}$$

These derivatives are assessed at $\mathbf{x}(t) = \hat{\mathbf{x}}(t \mid t-1)$. Hence, $\mathbf{F}(t)$ and $\mathbf{H}(t)$ become time-varying matrices acquired through the local linearization of the nonlinear system around the predicted state vector from the preceding step.

In practice, the Extended Kalman Filter aims to convert a nonlinear, time-invariant system into a linear, time-varying system. Subsequently, it employs the Kalman Filter methodology on this transformed linear system.

The general procedure for the Extended Kalman Filter at the current time t is outlined as follows:

1. Utilize the most recent state prediction, denoted as $\hat{\mathbf{x}}(t \mid t-1)$.

2. Employ $\hat{\mathbf{x}}(t \mid t-1)$ to compute $\mathbf{F}(t)$ and $\mathbf{H}(t)$.

3. Calculate $\mathbf{K}(t)$ and update the Differential Riccati Equation, thereby determining $\mathbf{P}(t+1)$.

4. Compute $\hat{\mathbf{x}}(t+1 \mid t)$ for the subsequent iteration.

This iterative procedure must be repeated for each sampling instant.

Remarks The Extended Kalman Filter stands out for its efficacy in handling nonlinear systems. However, due to its inherent nature as a nonlinear, time-variant system, it lacks a steady-state solution. Consequently, several implications arise:

- Achieving asymptotic stability is challenging or infeasible, necessitating extensive empirical validation.
- Computational burden: with each sampling instance, recalculations of $\mathbf{F}(t)$, $\mathbf{H}(t)$, $\mathbf{P}(t)$, and $\mathbf{K}(t)$ are required.

While the Extended Kalman Filter finds widespread application, it encounters limitations in scenarios involving safety critical systems and applications, and mission critical systems and applications. The exploration of meta-parameters within the Kalman Filter structure represents a new avenue in Machine Learning.

CHAPTER 4

Black-box software sensing

4.1 Introduction

We've successfully tackled software sensing using the Kalman Filter. The key characteristics of the Kalman Filter include:

- It necessitates a white-box model of the system.
- While theoretically a training dataset isn't required, practical tuning $\mathbf{v}_1$ and $\mathbf{v}_2$ often involves experimentation and data.
- It operates as a constructive method built upon a feedback correction framework.
- Theoretically, it can estimate states that are entirely unmeasurable.

Alternatively, the same software sensing problem can be approached using a black-box methodology, by redefining the problem as a black-box system identification challenge.

4.2 Problem definition

We commence with a training dataset comprising:

$$\{u(1), u(2), \ldots, u(N)\}$$
$$\{y(1), y(2), \ldots, y(N)\}$$
$$\{x(1), x(2), \ldots, x(N)\}$$

In this methodology, during the design and training phase, complete state measurements are required. Therefore, initially, more sensors than the final system configuration are necessary. Now, the goal is to identify a system with the given input-output relationship.

Model We seek a model with the depicted architecture:

$$\hat{x}(t|t) = \boxed{\mathcal{S}_{ux}(z)} \cdot u(t-1) + \boxed{\mathcal{S}_{yx}(z)} y(t)$$

That is:

$$\hat{x}(t) = \mathcal{S}_{ux}(\mathcal{Z}, \boldsymbol{\theta})u(t-1) + \mathcal{S}_{yx}(\mathcal{Z}, \boldsymbol{\theta})y(t)$$

Here, the transfer functions $\mathcal{S}_{ux}$ and $\mathcal{S}_{yx}$ need to be determined. To achieve this, we can employ the standard parametric approach:

$$J_N(\boldsymbol{\theta}) = \frac{1}{N}\sum_{t=1}^{N}\left(x(t) - \hat{x}(t)\right)^2$$

The optimization step involves:

$$\hat{\boldsymbol{\theta}}_N = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J_N(\boldsymbol{\theta})$$

Once the software sensing algorithm has been devised, physical state sensors can be removed. The above approach can also be implemented using alternative system identification methods such as 4SID.

4.3 Nonlinear systems

In cases where the system exhibits nonlinearity, we can still apply a similar approach, but now necessitating a nonlinear software sensor.

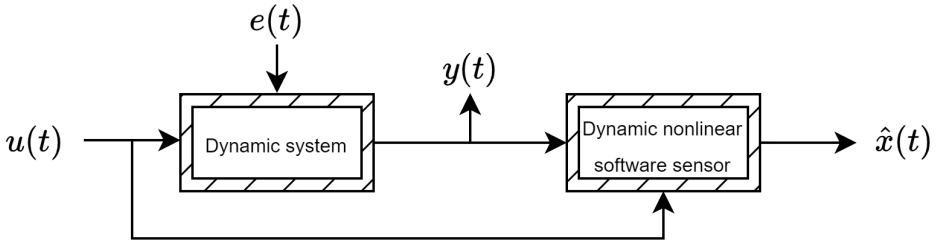


Figure 4.1: Nonlinear software sensor

There exist four potential architectures commonly employed to address nonlinear problems.

4.3.1 Neural Network architecture

In this framework, a dynamical Neural Network is employed:

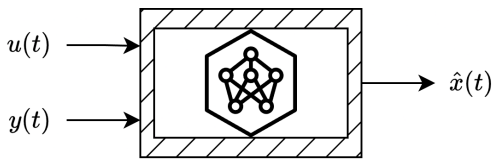
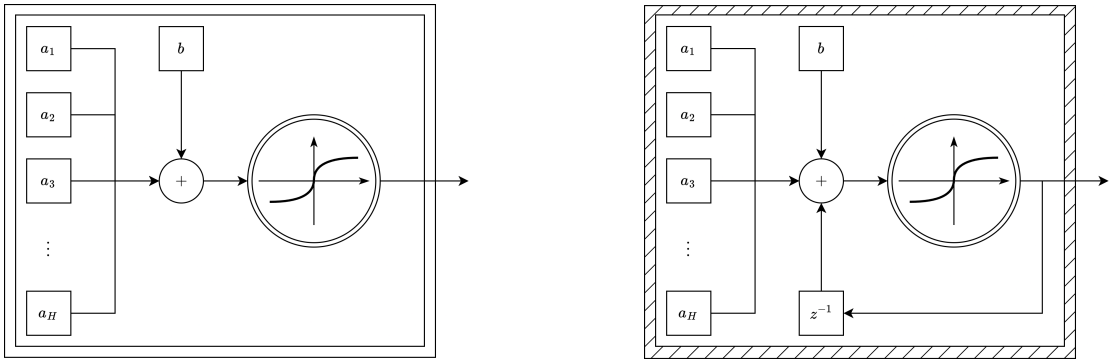


Figure 4.2: Neural Network

Each neuron within the network can be either static or dynamic:



(a) Static Neuron (b) Dynamic Neuron

Figure 4.3: Neurons

This architecture is intuitive and versatile, but not widely used in practice due to:

- High computational demand for training and optimization.
- Lack of guarantee regarding convergence and stability of the resulting Neural Network.

4.3.2 Finite Impulse Response filter architecture

The software sensor is divided into a static nonlinear part and a dynamic nonlinear part utilizing a non-recursive Finite Impulse Response Filter-like scheme:

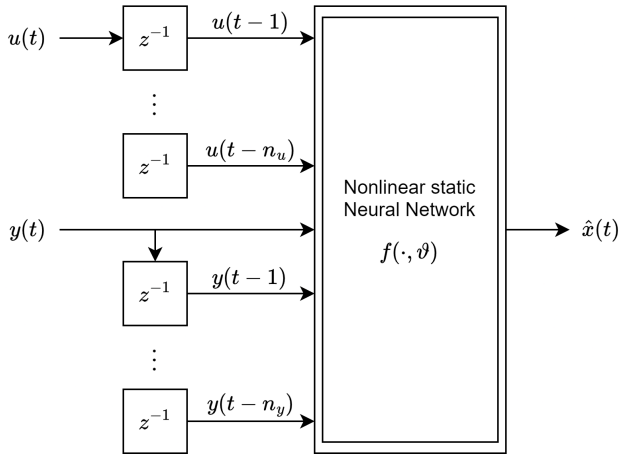


Figure 4.4: Input and output splitting

The inputs and outputs are treated as linear dynamic systems, while the block represents a nonlinear static parametric system. Consequently, training is solely focused on the static nonlinear Neural Network component.

The training process is limited to the static nonlinear Neural Network, ensuring stability by design.

It's important to note that in the case of multiple input multiple output systems, where:

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \quad y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_p(t) \end{bmatrix} \quad x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

As a result, the input-output size can be significant.

4.3.3 Infinite Impulse Response filter architecture

To address the challenge of large mapping input-output sizes, we can adopt the previous architecture but incorporate recursion, resembling an Infinite Impulse Response filter scheme:

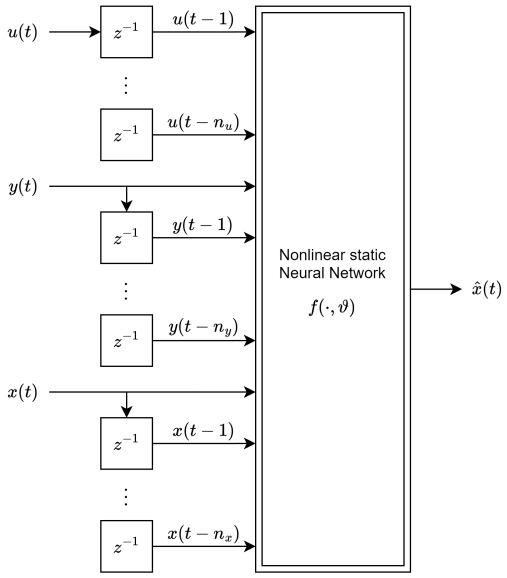


Figure 4.5: Training input and output splitting with recursion

This architecture features a recursive state component that considers past output values. The advantage lies in the potential reduction of n_u and n_y dimensions. However, a notable disadvantage arises during production, as $\mathbf{x}(t)$ is no longer available after training. Consequently, it must be substituted with feedback on $\hat{\mathbf{x}}(t)$:

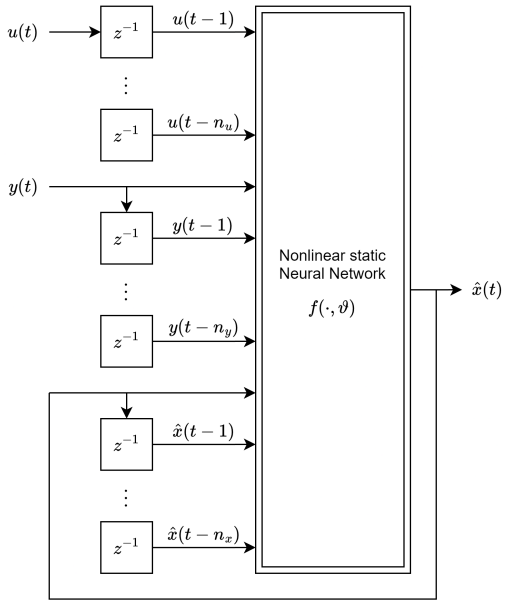


Figure 4.6: Production input and output splitting with recursion

This feedback mechanism on $\hat{\mathbf{x}}(t)$ introduces the potential for system instability.

4.3.4 Meta architecture

This architecture represents a modification incorporating prior knowledge of the system gleaned from the previous three architectures. The concept involves splitting the system into two distinct parts:

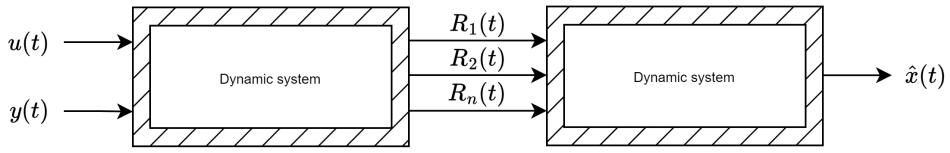


Figure 4.7: Split architecture

The first block serves as a preprocessing stage, wherein $u(t)$ and $y(t)$ are utilized to construct a set of new signals known as regressors $\mathbf{R}(t)$.

The preprocessing block is built using a priori knowledge, so it is not built using system identification but with know-how on $u(t)$ and $y(t)$. The preprocessing block is constructed using a priori knowledge, obviating the need for system identification. Typically, the regressors $\mathbf{R}(t)$ are considerably fewer than $u(t)$ and $y(t)$, leading to a substantial reduction in the size of the second block.

If the regressors are judiciously chosen, the second block may remain static. This approach offers the advantage of greatly simplifying the estimation process of the second block.

4.4 Software sensing methodologies comparison

White-box models require a physical system model and, in practice, a training dataset. They offer interpretability, allow retuning, and provide good software sensor accuracy, including sensing unmeasurable states. In contrast, black-box models do not need a physical model but require training data. They lack interpretability and retuning capability but achieve very good software sensor accuracy without sensing unmeasurable states.

Feature	**White box**	**Black box**
Physical model of the system	Required	Not required
Training dataset	Required in practice	Required
Interpretability	Yes	No
Retuning	Yes	No
Software sensor accuracy	Good	Very good
Software sensing of unmeasurable states	Yes	No

CHAPTER 5

Gray-box system identification

5.1 Introduction

The Kalman Filter serves as a powerful tool for prediction and filtering, primarily employed for software sensing variable estimation. However, its utility extends beyond these applications, as it can be readily repurposed for gray-box system identification.

It's essential to recognize that while this capability represents a valuable side benefit, it is not the primary objective of the Kalman Filter.

5.2 Online gray box system identification

In gray-box system identification, we begin with a model constructed using a white-box approach, which relies on first principles:

$$\mathcal{S}: \begin{cases} \mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}) + \mathbf{v}_1(t) \\ \mathbf{y}(t) = h(\mathbf{x}(t), \boldsymbol{\theta}) + \mathbf{v}_2(t) \end{cases}$$

Here, $f(\cdot)$ and $h(\cdot)$ represent known equations with some parameters unknown, denoted by the parameter vector $\boldsymbol{\theta}$ having physical significance.

The objective is to estimate the parameters of $\boldsymbol{\theta}$ from a given dataset. This task can be tackled using the Kalman Filter technique along with a state extension trick. Consequently, the new system becomes:

$$\mathcal{S}: \begin{cases} \mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}(t)) + \mathbf{v}_1(t) \\ \boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \mathbf{v}_{\boldsymbol{\theta}}(t) \\ \mathbf{y}(t) = h(\mathbf{x}(t), \boldsymbol{\theta}(t)) + \mathbf{v}_2(t) \end{cases}$$

The extended state vector $\mathbf{x}_E(t)$ comprises the previous states along with the value of $\boldsymbol{\theta}(t)$:

$$\mathbf{x}_E(t) = \begin{bmatrix} \mathbf{x}(t) \\ \boldsymbol{\theta}(t) \end{bmatrix}$$

5.2.1 Parameters equation

The equation $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \mathbf{v}_{\boldsymbol{\theta}}(t)$ is a fictitious equation necessitated by the state extension. The fundamental dynamical relationship remains $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t)$, representing the equation of a constant quantity. This is appropriate since our objective is to estimate a constant parameter. Notably, this equation represents a trivially unstable system (non-asymptotically stable). However, this instability poses no issue as the Kalman Filter is capable of handling non-asymptotically stable systems.

The inclusion of the noise $\mathbf{v}_{\boldsymbol{\theta}}(t)$ is crucial; without it, the equation becomes steady-state, and the Kalman Filter does not alter the initial condition. By introducing $\mathbf{v}_{\boldsymbol{\theta}}(t)$, we instruct the Kalman Filter to iteratively converge to the correct value of $\boldsymbol{\theta}$, without overly relying on the initial condition.

Noise definition The primary challenge lies in defining the noise. Typically, the following assumption is made:

$$\mathbf{v}_{\boldsymbol{\theta}} \sim WN(0, \mathbf{V}_{\boldsymbol{\theta}})$$

This noise is assumed to be independent of other noises.

A common simplifying assumption is often made:

$$\mathbf{V}_{\boldsymbol{\theta}} = \begin{bmatrix} \lambda_{\boldsymbol{\theta}}^2 & 0 & 0 & 0 \\ 0 & \lambda_{\boldsymbol{\theta}}^2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda_{\boldsymbol{\theta}}^2 \end{bmatrix}$$

In practice, all noise characteristics are condensed into a single parameter $\lambda_{\boldsymbol{\theta}}^2$. This parameter is empirically tuned, serving as a tuning parameter. A large value of $\lambda_{\boldsymbol{\theta}}^2$ yields rapid convergence but substantial post-convergence oscillation, while a small value results in slower convergence but minimal post-convergence oscillation.

This approach is particularly useful when $\boldsymbol{\theta}$ varies, as the algorithm remains active. Applying the Kalman Filter to the extended system facilitates simultaneous estimation of the state vector (software sensing) and the unknown parameters (system identification).

5.3 Offline grey box identification

In many applications, we have some prior knowledge of a possible model for a system, denoted as $\mathcal{M}(\boldsymbol{\theta})$, which depends on certain physical parameters. The model, expressed in state-space representation, is given by:

$$\mathcal{M}(\boldsymbol{\theta}) : \begin{cases} \dot{\mathbf{x}}(t) = \mathbf{F}(\boldsymbol{\theta})\mathbf{x}(t) + \mathbf{G}(\boldsymbol{\theta})\mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{H}(\boldsymbol{\theta})\mathbf{x}(t) + \mathbf{D}(\boldsymbol{\theta})\mathbf{u}(t) \end{cases}$$

Our objective is to estimate the parameters $\boldsymbol{\theta}$ based on input and output data.

5.3.1 Simulation Error Method

One approach to parameter estimation is the Simulation Error Method. We select an initial estimate $\bar{\boldsymbol{\theta}}$ and simulate the model $\mathcal{M}\left(\bar{\boldsymbol{\theta}}\right)$. The accuracy of the estimate is assessed by comparing the simulated output $\mathbf{y}_{\text{sim}}(t)$ with the actual measured output $\mathbf{y}(t)$. This comparison is

formulated mathematically as:

$$\boldsymbol{\theta}_{\text{opt}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmin}} \operatorname{RMS}(\mathbf{y}(t) - \mathbf{y}_{\text{sim}}(t, \boldsymbol{\theta})) = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmin}} \operatorname{RMS}(\mathbf{y}(t) - \mathcal{M}(\boldsymbol{\theta})u(t))$$

Here, $\operatorname{RMS}(\cdot)$ denotes the Root Mean Square error, and $\boldsymbol{\Theta}$ represents the set of all possible parameter values. Since the optimization problem is generally non-convex and does not have a linear relationship with the parameter vector $\boldsymbol{\theta}$, finding an optimal solution requires specialized numerical techniques.

Data quality For effective identification, the dataset $\mathcal{D}$ must be informative. The length of the dataset is less important than ensuring it excites all relevant system dynamics. The quality of the identified parameters directly depends on the richness of the data.

Common signal types used for system identification include:

- *Sine sweep or multiple sine experiments*: these signals cover a broad frequency range, providing useful information in both time and frequency domains. The frequency sweeps from an initial value f_{init} to a final value f_{end}.
- *Pseudo-Random Binary Signal*: a two-level signal that switches values at random intervals. PRBS provides a nearly flat frequency spectrum up to a specified cutoff frequency f_{max}, making it effective for system identification.

Typically, the optimization process is performed on at least one dataset and validated on a separate dataset to ensure robustness. The validation must be conducted in both the time and frequency domains.

Optimization The grey-box identification optimization follows these steps:

Algorithm 1 Grey-box optimization

1: $\mathbf{F} = f(\boldsymbol{\theta})$
2: $\mathbf{G} = f(\boldsymbol{\theta})$
3: $\mathbf{H} = f(\boldsymbol{\theta})$
4: $\mathbf{D} = f(\boldsymbol{\theta})$
5: sys = ss($\mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{D}$) ▷ Construct the state-space system
6: y_sim = lsim(sys, id_data.u, id_data.time) ▷ Simulate system response
7: cost = rms(id_data_y − y_sim) ▷ Compute cost function
8: **return** cost

Since the optimization is non-convex, it requires specialized solvers. We implement the optimization in MATLAB, using two different solvers: `fmincon` (a gradient-based solver) and `particleswarm` (a global optimization method). These solvers offer different advantages, and their performance is compared to determine the best approach for parameter estimation.

CHAPTER 6

Minimum variance control

6.1 Introduction

The process of designing a control system typically unfolds in three sequential steps:

1. *System modeling*: this involves creating a mathematical representation of the system under consideration.
2. *Estimation of unmeasurable variables* (software sensing): certain variables within the system may not be directly measurable and need to be estimated through software-based techniques.
3. *Control algorithm design*: here, the actual control algorithm that governs the system's behavior is developed.

In the realm of minimum variance control, the mathematical techniques employed bear strong resemblance to those utilized in system identification and software sensing. Minimum variance control, essentially an optimization endeavor centered around an ARMAX system, shares mathematical commonalities with these areas.

Indeed, Minimum Variance Control can be viewed as a versatile tool for optimizing stochastic feedback systems, offering a broad spectrum of applications in the realm of control engineering.

Problem The problem of Minimum Variance Control is set around a standard ARMAX representation of the system:

$$y(t) = \frac{B(z)}{A(z)}u(t-k) + \frac{C(z)}{A(z)}e(t)$$

Here:

$$\begin{cases} A(z) = 1 + a_1 z^{-1} + \cdots + a_n z^{-n} \\ B(z) = b_0 + b_1 z^{-1} + \cdots + b_p z^{-p} \\ C(z) = 1 + c_1 z^{-1} + \cdots + c_m z^{-m} \end{cases}$$

We assume that the transfer function $\frac{C(z)}{A(z)}$ is in canonical form, and k is the pure delay from $u(t)$ to $y(t)$ ($k \geq 1$). Also, we assume $b_0 \neq 0$ to prevent changes in the actual delay. Additionally, we assume that $\frac{B(z)}{A(z)}$ is minimum phase.

Controlling a non-minimum phase system is challenging because immediate reactions may lead to incorrect decisions. Designing control systems for non-minimum phase systems requires specific design tools. Minimum variance control cannot be used for non-minimum phase systems. Instead, there exists a Generalized Minimum Variance Control that can be employed, even for minimum phase systems.

The objective of the control system is to track a desired output reference signal:

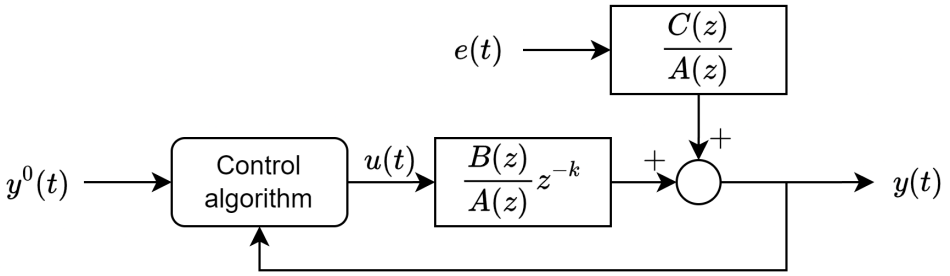


Figure 6.1: Control system

Ideally, $y(t)$ should exactly follow $y^0(t)$.

Formal description In a formal manner, we can define the problem as an optimal control problem: find the signal $u(t)$ such that the performance index J is minimized:

$$\min J = \mathbb{E}\left[\left(y(t) - y^0(t)\right)^2\right]$$

That is the variance of the tracking error between $y(t)$ and $y^0(t)$.

We attempt to solve this problem with the following technical assumptions:

1. $y^0(t)$ and $e(t)$ are uncorrelated.
2. $y^0(t)$ is unpredictable (worst-case assumption). This implies that we have no preview of the future desired output behavior. The best prediction we can make at time t is:

$$\hat{y}^\circ(t+k \mid t) = y^0(t)$$

6.2 General problem

Given the ARMAX system:

$$\mathcal{S} : y(t) = \frac{B(z)}{A(z)}u(t-k) + \frac{C(z)}{A(z)}e(t) \qquad e(t) \sim WN(0, \lambda^2)$$

We have the following assumptions:

- $b_0 \neq 0$.
- $B(z)$ has all the roots strictly inside the unit circle.
- $\frac{C(z)}{A(z)}$ is in canonical representation.
- $y^0(t) \perp e(t)$.

- $y^0(t)$ is unpredictable, so $\hat{y}^\circ(t+k \mid t) = y^0(t)$.

Our goal is to minimize:

$$\min J = \mathbb{E}\left[\left(y(t) - y^0(t)\right)^2\right]$$

Optimality condition The primary method to solve this problem is to split $y(t)$ into predictor and error. Remembering that the prediction error is defined as $\varepsilon(t) = y(t) - \hat{y}(t \mid t-k)$, we have:

$$y(t) = \hat{y}(t \mid t-k) + \varepsilon(t)$$

We can now compute the performance index as:

$$\begin{aligned} J &= \mathbb{E}\left[\left(\hat{y}(t \mid t-k) + \varepsilon(t) - y^0(t)\right)^2\right] \\ &= \mathbb{E}\left[\left(\left(\hat{y}(t \mid t-k) - y^0(t)\right) + \varepsilon(t)\right)^2\right] \\ &= \mathbb{E}\left[\left(\hat{y}(t \mid t-k) - y^0(t)\right)^2\right] + \underbrace{\mathbb{E}\left[\varepsilon(t)^2\right]}_{\text{not } u(t)\text{-dependent}} + 2\underbrace{\mathbb{E}\left[\left(\hat{y}(t \mid t-k) - y^0(t)\right)\varepsilon(t)\right]}_{0} \end{aligned}$$

As a result, minimizing J with respect to $u(t)$ is equivalent to minimizing the expected value of:

$$\min \mathbb{E}\left[\left(\hat{y}(t \mid t-k) - y^0(t)\right)^2\right]$$

This is minimum when:

$$\hat{y}(t \mid t-k) = y^0(t)$$

Optimal predictor To find the optimal predictor, we perform k steps of the long division between $C(z)$ and $A(z)$, obtaining a quotient $E(z)$ and the residual $R(z) = \tilde{R}(z)z^{-k}$. Thus, we have:

$$\frac{C(z)}{A(z)} = E(z) + \frac{\tilde{R}(z)z^{-k}}{A(z)}$$

The optimal predictor of an ARMAX system is then given by:

$$\hat{y}(t+k \mid t) = \frac{B(z)E(z)}{C(z)}u(t-k) + \frac{\tilde{R}(z)}{C(z)}y(t)$$

To ensure the optimality condition $\hat{y}(t+k \mid t) = y^0(t+k)$, we impose that $\hat{y}(t+k \mid t) = y^0(t)$:

$$y^0(t) = \frac{B(z)E(z)}{C(z)}u(t-k) + \frac{\tilde{R}(z)}{C(z)}y(t)$$

We can now isolate $u(t)$ to find the controller algorithm:

$$u(t) = \frac{1}{B(z)E(z)}\left(C(z)y^0(t) - \tilde{R}(z)y(t)\right)$$

This formula represents the general Minimum Variance Control controller.

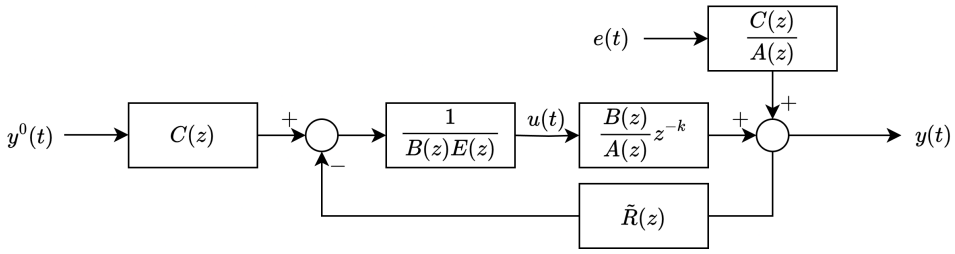


Figure 6.2: Full control system scheme

6.3 Control system analysis

We'll delve into assessing both the stability and performance of the Minimum Variance Control system.

6.3.1 Stability

To assess the stability of the Minimum Variance Control system, we can apply stability criteria for feedback systems. Considering the closed-loop transfer function $L(z)$, defined as:

$$L(z) = \frac{1}{B(z)E(z)} \frac{B(z)z^{-k}}{A(z)} \tilde{R}(z)$$

We then have:

$$\chi_x = L_N(z) + L_D(z) = B(z)\tilde{R}(z)z^{-k} + B(z)E(z)A(z) = B(z) \underbrace{\left(\tilde{R}(z)z^{-k} + E(z)A(z)\right)}_{C(z)}$$

Given that the roots of $B(z)$ are strictly inside the unit circle (due to the assumption of a minimum phase system) and the roots of $C(z)$ are strictly inside the unit circle (owing to the assumption that $\frac{C(z)}{A(z)}$ is in canonical representation), the Minimum Variance Control system is guaranteed to be asymptotically stable.

6.3.2 Performance

The control system can be expressed as a function of its two inputs:

$$y(t) = F_{y^0y}(z)y^0(t) + F_{ey}(z)e(t)$$

The transfer function from $y^0(t)$ to $y(t)$ is:

$$F_{y^0y}(z) = C(z)\frac{\frac{1}{B(z)E(z)}\frac{B(z)}{A(z)}z^{-k}}{1 + \frac{1}{B(z)E(z)}\frac{B(z)}{A(z)}z^{-k}\tilde{R}(z)} = z^{-k}$$

And the transfer function from $e(t)$ to $y(t)$ is:

$$F_{ey}(z) = \frac{\frac{C(z)}{A(z)}}{1 + \frac{1}{B(z)E(z)}\frac{B(z)}{A(z)}z^{-k}\tilde{R}(z)} = E(z)$$

Hence, the closed-loop behavior of the Minimum Variance Controller system is:

$$y(t) = y^0(t)z^{-k} + E(z)e(t) = y^0(t-k) + E(z)e(t)$$

That has a very simple closed loop behavior.

This indicates perfect tracking, albeit with a delay of k steps due to the pure delay and the unpredictability of $y^0(t)$. Moreover, $E(z)e(t)$ represents the minimum amount of noise (the prediction error) making the Minimum Variance Control system optimal. Therefore, achieving better tracking with lower noise is impossible.

6.4 Generalized Minimum Variance Control

The primary limitations of Minimum Variance Control are:

- It can only be applied to minimum phase systems.
- There is no moderation of the effort and intensity of the control action $u(t)$.
- t lacks the capability to design a specific behavior or relationship between $y^0(t)$ and $y(t)$.

To address these limitations, Generalized Minimum Variance Control extends Minimum Variance Control.

6.4.1 Performance indexes comparison

For Minimum Variance Control, the performance index is:

$$J = \mathbb{E}\left[\big(y(t) - y^0(t)\big)^2\right]$$

However, for Generalized Minimum Variance Control, the performance index is more comprehensive:

$$J = \mathbb{E}\left[\big(y(t) - \mathbf{P}(z)y^0(t) + \mathbf{Q}(z)u(t)\big)^2\right]$$

Here, the additional term $\mathbf{Q}(z)u(t)$ penalizes the usage of control effort, and $\mathbf{P}(z)$ establishes a reference model between $y^0(t)$ and $y(t)$. It's noteworthy that Generalized Minimum Variance Control reduces to simple Minimum Variance Control when $\mathbf{P}(z) = 1$ and $Q(z) = 0$.

Reference model matrix The matrix $\mathbf{P}(z)$ serves as the reference model for the closed-loop system's desired behavior. In general, we envision the ideal scenario as: It's important to note that in some systems, the optimal behavior isn't always achieved when $\mathbf{P}(z) = 1$. In such cases, a low-pass filter $\mathbf{P}(z)$ might be employed to establish a new target for the closed loop.

APPENDIX A

Transfer functions and matrices

A.1 Observability and controllability

Let's consider the system described by:

$$\begin{cases} \mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}u(t) \\ y(t) = \mathbf{H}\mathbf{x}(t) \end{cases}$$

Definition (*Fully observable*)**.** A system is fully observable from the output $y(t)$ if and only if its observability matrix, denoted as $\mathbf{O}$, is full rank, meaning its rank equals the number of states n.

The observability matrix is defined as:

$$\mathbf{O} = \begin{bmatrix} \mathbf{H} \\ \mathbf{HF} \\ \mathbf{HF}^2 \\ \vdots \\ \mathbf{HF}^{n-1} \end{bmatrix}$$

Observability, which solely depends on $\mathbf{F}$ and $\mathbf{H}$, indicates that by observing the output $y(t)$, we can deduce the behavior of the system's state $\mathbf{x}(t)$.

Definition (*Fully controllable*)**.** A system is fully controllable (or reachable) from the input $u(t)$ if and only if its controllability matrix, denoted as $\mathbf{R}$, is full rank, also equal to the number of states n.

The reachability matrix is defined as:

$$\mathbf{R} = \begin{bmatrix} \mathbf{G} & \mathbf{FG} & \mathbf{F}^2\mathbf{G} & \cdots & \mathbf{F}^{n-1}\mathbf{G} \end{bmatrix}$$

Controllability, which solely relies on $\mathbf{F}$ and $\mathbf{G}$, implies that by manipulating the input $u(t)$, we can affect the behavior of the system's state $\mathbf{x}(t)$.

Categorizing state-space representations based on observability and controllability yields four distinct subsystems:

- *Non-observable and controllable*: in this case, the system's internal states cannot be fully inferred from the output, and not all states can be influenced by the input.
- *Observable and non-controllable*: here, the system's states can be fully inferred from the output, but not all states can be influenced by the input.
- *Observable and controllable*: this scenario indicates that both full observability and controllability are present, meaning all states can be inferred from the output and influenced by the input.
- *Non-observable and non-controllable*: in this situation, neither full observability nor controllability exists. The internal states cannot be fully inferred from the output, and not all states can be influenced by the input.

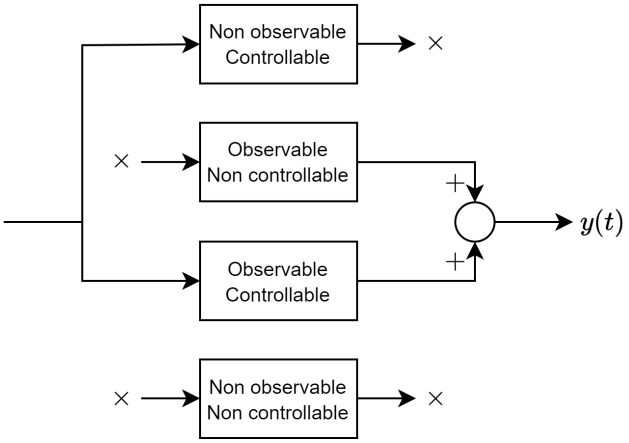


Figure A.1: State space subsystems

When representing these subsystems externally using a transfer function, only the observable and controllable parts of the system can be captured. The other subsystems, which lack either observability or controllability or both, remain hidden from the input and output perspective. Thus, the transfer function representation provides insight only into the observable and controllable aspects of the system.

A.1.1 Hankel matrix

The Hankel matrix of order n is a square matrix constructed from the impulse response representation. It starts from the impulse at time one (excluding zero), and each counter-diagonal contains responses from the same time instant. Mathematically, it's defined as:

$$\mathbf{H}_n = \begin{bmatrix} \omega(1) & \omega(2) & \omega(3) & \cdots & \omega(n) \\ \omega(2) & \omega(3) & \omega(4) & \cdots & \omega(n+1) \\ \omega(3) & \omega(4) & \omega(5) & \cdots & \omega(n+2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \omega(n) & \omega(n+1) & \omega(n+2) & \cdots & \omega(2n-1) \end{bmatrix}$$

Given the transformation from transfer function to impulse response representations:

$$\omega(t) = \mathbf{H}\mathbf{F}^{t-1}\mathbf{G} \qquad \text{if } t \geq 1$$

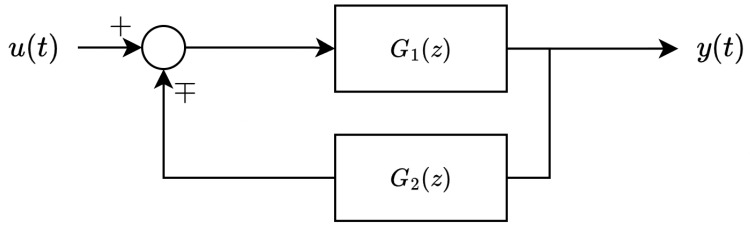
We can rewrite the Hankel matrix as:

$$\mathbf{H}_n = \begin{bmatrix} \mathbf{HG} & \mathbf{HFG} & \mathbf{HF^2G} & \cdots & \mathbf{HF^{n-1}G} \\ \mathbf{HFG} & \mathbf{HF^2G} & \mathbf{HF^3G} & \cdots & \mathbf{HF^nG} \\ \mathbf{HF^2G} & \mathbf{HF^3G} & \mathbf{HF^4G} & \cdots & \mathbf{HF^{n+1}G} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{HF^{n-1}G} & \mathbf{HF^nG} & \mathbf{HF^{n+1}G} & \cdots & \mathbf{HF^{2n+2}G} \end{bmatrix} = \begin{bmatrix} \mathbf{H} \\ \mathbf{HF} \\ \mathbf{HF^2} \\ \vdots \\ \mathbf{HF^{n-1}} \end{bmatrix} . \begin{bmatrix} \mathbf{G} \\ \mathbf{FG} \\ \mathbf{F^2G} \\ \vdots \\ \mathbf{F^{n-1}G} \end{bmatrix}^T$$

Therefore, the Hankel matrix is the outer product of the observability and controllability matrices.

A.2 Feedback systems

The transfer function of a feedback system can be derived from its block diagram as follows:



We can write the equation for the output $y(t)$ in terms of the input $u(t)$ and the transfer functions $G_1(z)$ and $G_2(z)$:

$$\begin{aligned} &y(t) = G_1(z)\left(u(t) \pm G_2(t)y(t)\right) \rightarrow \\ &y(t)\left(1 \pm G_1(z)G_2(z)\right) = G_1(z)u(t) \rightarrow \\ &y(t) = \frac{G_1(z)}{1 \pm G_1(z)G_2(z)}u(t) \end{aligned}$$

This formula represents the general transfer function for any feedback system. It shows that the input-output behavior of the closed-loop system is relatively straightforward, as many dynamics are hidden in the non-observable part of the system by design.

A.2.1 Feedback system stability

To check the stability of a feedback system, we create the loop function $L(z)$ defined as the product of the transfer functions $G_1(z)$ and $G_2(z)$:

$$L(z) = G_1(z) \cdot G_2(z)$$

From this loop function, we can construct the characteristic polynomial $\chi(z)$ by considering its numerator and denominator:

$$\chi(z) = L_N(z) \pm L_D(z)$$

Here, $L_N(z)$ represents the numerator of the closed-loop function, and $L_D(z)$ represents the denominator.

Theorem A.2.1. *The feedback system is asymptotically stable if and only if all the roots of the characteristic polynomial $\chi(z)$ are strictly inside the unit circle.*

In summary, to determine the stability of a feedback system, we analyze the roots of the characteristic polynomial obtained from the loop function $L(z)$. If all the roots lie inside the unit circle in the complex plane, the system is asymptotically stable. Otherwise, if any root lies on or outside the unit circle, the system is unstable.

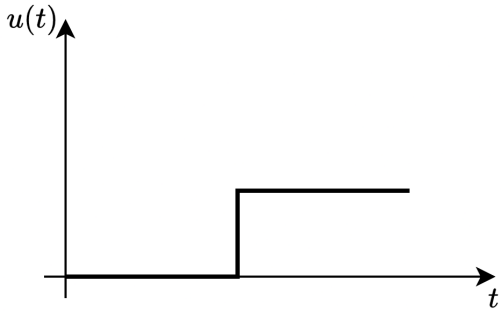
A.3 Minimum phase systems

Definition (*Minimum phase system*)**.** A system is classified as a minimum phase system if all the zeros of its transfer function lie inside the unit circle in the complex plane.

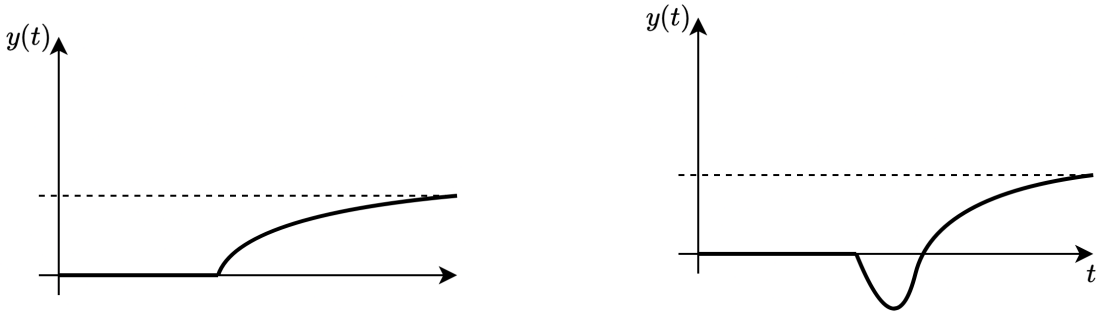
When considering a step input signal at a certain time instant, the response of a minimum phase system typically exhibits a behavior where the output initially moves towards its final value. This behavior is due to the absence of zeros outside the unit circle, which results in a stable and predictable response.

Conversely, in the case of a non-minimum phase system, some zeros of the transfer function lie outside the unit circle. As a consequence, the initial response of the system may move in the opposite direction compared to its final value. This behavior arises due to the presence of unstable or oscillatory modes introduced by the zeros outside the unit circle.

Consider a step on the input signal at a certain time instant:



Visually, this difference in behavior between minimum and non-minimum phase systems can be illustrated by observing the direction of the initial response relative to the final value in the step response plot. In minimum phase systems, the initial response moves towards the final value, while in non-minimum phase systems, it may initially move away from the final value before eventually converging to it.



(a) Minimum phase system (b) Non-minimum phase system

Figure A.2: System comparison

A.4 Open-loop system

In situations where a system is open-loop unstable, conducting experiments directly on the open-loop system can be challenging or even impossible due to the inherent instability. However, one approach to analyze and identify the system's characteristics is to perform a closed-loop experiment.

In a closed-loop experiment, the system is stabilized by feedback, allowing for safe and controlled data collection. The feedback loop ensures that the output of the system is regulated and can be measured accurately. This setup enables the collection of data for system identification and analysis.

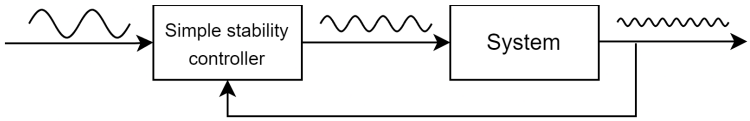


Figure A.3: Closed-loop system identification

By performing experiments in a closed-loop configuration, it is possible to study and understand the behavior of unstable systems while maintaining stability and safety. This approach is commonly used in control system analysis and design, particularly when dealing with unstable or uncertain systems.

APPENDIX B

Matrix decomposition

B.1 Unitary matrix

Definition (*Unitary matrix*)**.** A matrix is termed unitary if its determinant is one and the inverse is equal to its transpose:

$$\det(\mathbf{A}) \qquad \mathbf{A}^T = \mathbf{A}^{-1}$$

This means that multiplying a unitary matrix by its transpose should yield the identity matrix.

B.2 Singular Value Decomposition

Singular Value Decomposition is a fundamental matrix factorization technique in linear algebra. It decomposes a matrix into three separate matrices, revealing the inherent structure and properties of the original matrix. Given a matrix $\mathbf{A}$ of size $m \times n$, the Singular Value Decomposition of $\mathbf{A}$ is represented as:

$$\text{SVD}(\mathbf{A}) = \mathbf{USV}^T$$

Here:

- $\mathbf{U}$ is an $m \times m$ orthogonal matrix, where the columns are the left singular vectors of $\mathbf{A}$.
- $\mathbf{S}$ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal, known as the singular values of $\mathbf{A}$. The diagonal entries are typically arranged in descending order.
- $\mathbf{V}$ is an $n \times n$ orthogonal matrix, where the columns are the right singular vectors of $\mathbf{A}$.

Usage Singular Value Decomposition has been a significant milestone in machine learning due to its ability to efficiently address several key problems:

1. *Compression of information*: Singular Value Decomposition allows for the compression of information by representing data in a lower-dimensional space while retaining essential features. This is achieved by truncating the singular values and corresponding singular vectors, effectively reducing the size of the data representation.

2. *Separation of signal and noise*: Singular Value Decomposition can be used to separate signal from noise in data. By decomposing the data matrix into its singular values and vectors, it becomes possible to isolate dominant patterns from random fluctuations.

3. *Solution to order-reduction problems*: Singular Value Decomposition provides a solution to order-reduction problems by capturing the dominant modes or patterns in the data while discarding less significant components. This is particularly useful for reducing the complexity of models or data representations while preserving essential information.

Rank reduction Singular Value Decomposition enables an optimal rank reduction, which is crucial for minimizing the loss of information during dimensionality reduction. This optimal rank reduction aims to minimize the residual matrix, ensuring that the discarded components contain the least amount of information possible.

APPENDIX C

Discretization of analog dynamical systems

C.1 Introduction

Let's examine the configuration of a contemporary control system, as illustrated below:

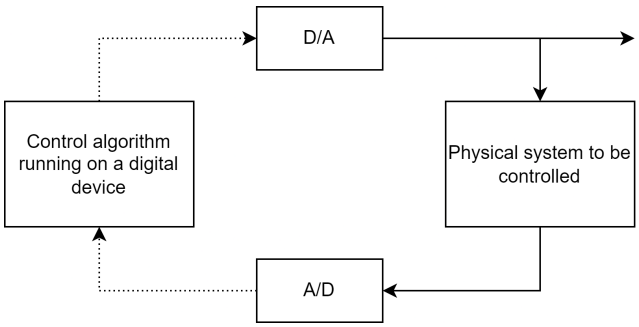


Figure C.1: Physical system

Analog to digital converter The analog to digital converter (ADC) facilitates the conversion of an analog signal into a discrete form, operating on both time and amplitude domains.

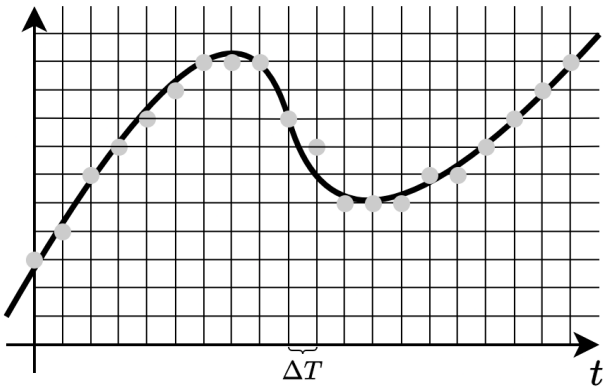


Figure C.2: Analog to digital conversion

The sampling frequency f_S is the reciprocal of the sampling time ΔT:

$$f_S = \frac{1}{\Delta T}$$

Here, ΔT represents the time discretization, while the amplitude discretization denotes the number of discrete levels available for conversion. The effectiveness and cost of an ADC hinge on ΔT (lower values being preferable) and the number of levels (higher values being desirable).

Digital to analog converter The digital to analog converter (DAC) performs the conversion of digital signals into their analog counterparts, operating in the following manner:

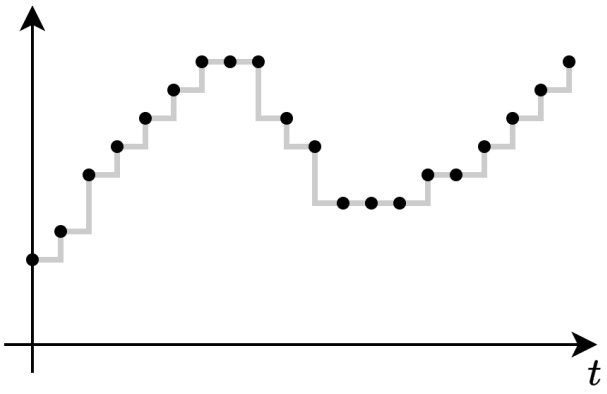


Figure C.3: Digital to analog conversion

The output signal from the DAC typically exhibits a step-like characteristic, known as zero-order-hold. While higher-order hold techniques exist, they are less commonly employed in practical applications.

It's important to note that when the time interval ΔT between successive digital samples is sufficiently small, the step-like behavior becomes imperceptible or negligible. Therefore, the quality of the DAC is primarily determined by the sampling time.

C.2 State space transformation

When transitioning from an analog system model to its discrete-time counterpart, we employ the following approach.

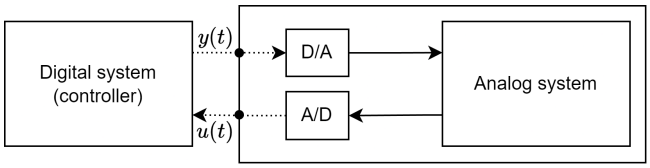


Figure C.4: System

Beginning with a continuous model:

$$\mathcal{S} : \begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{cases}$$

Assuming a sampling time of ΔT:

$$\mathcal{S}: \begin{cases} \mathbf{x}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{cases}$$

The commonly used discretization method involves state space transformation:

$$\mathbf{F} = e^{\mathbf{A}\Delta T} \qquad \mathbf{G} = \int_0^{\delta T} e^{\mathbf{A}\delta}\mathbf{B}\, d\delta \qquad \mathbf{H} = \mathbf{C} \qquad \mathbf{D} = \mathbf{D}$$

Poles discretization The poles of the continuous-time system are discretized into discrete time using the sampling transformation rule $\mathcal{Z} = e^{\mathcal{S}\Delta T}$, and so we have:

$$\lambda_{\mathbf{F}} = e^{\lambda_{\mathbf{A}}\Delta T}$$

Here, λ_F represents the eigenvalues of matrix F, and $\lambda_{\mathbf{A}}$ denotes the eigenvalues of matrix $\mathbf{A}$.

Zeros discretization Beginning with the transfer function in continuous time:

$$G(s) = \frac{\text{polynomial in } s \text{ with } h \text{ zeros}}{\text{polynomial in } s \text{ with } n \text{ poles}}$$

Where $h < n$. n discrete time, we have:

$$G(z) = \frac{\text{polynomial in } z \text{ with } n-1 \text{ zeros}}{\text{polynomial in } z \text{ with } n \text{ poles}}$$

During the transformation of zeros from continuous to discrete time, $n - h - 1$ new zeros are generated, known as hidden zeros. Unfortunately, these hidden zeros do not adhere to the sampling rule and are typically non-minimum phase.

C.3 Discretization of the time derivative

The discretization techniques employed in this context uses a convex combination of Euler forward and Euler backward methods:

$$\dot{\mathbf{x}} \approx \left[\frac{z-1}{\Delta T}\frac{1}{\alpha z + (1-\alpha)}\right]\mathbf{x}(t)$$

where $0 \leq \alpha \leq 1$. A commonly utilized special case is when $\alpha = \frac{1}{2}$ (Tustin discretization method).

C.4 Sampling time choice

The most critical decision in discretizing models lies in selecting the sampling time ΔT.

Definition (*Nyquist frequency*)**.** The Nyquist frequency is defined as:

$$f_N = \frac{1}{2}f_S$$

Definition (*Nyquist angular speed*)**.** The Nyquist angular speed is defined as:

$$\omega_N = \frac{1}{2}\omega_S$$

In general, we aim to choose the smallest possible sampling time. This is because a larger ω_N corresponds to a broader bandwidth over which the discretized model approximates the original one well.

Problems There are several issues associated with selecting a very small sampling time:

- Higher cost of sampling devices.
- Increased computational burden: updating algorithms at higher frequencies is more demanding.
- Memory cost: more memory is required for data logging or time buffering with smaller sampling times.
- Numerical precision cost: with small sampling times, poles are mapped very close to the edge of the unit circle in the z-plane, necessitating high numerical precision for discrete-time algorithm management.

For dynamical systems in control applications, a practical guideline for f_S is approximately twenty times the control system's bandwidth. In practice, given the desired design bandwidth of the closed-loop control system denoted as ω_C, we can determine the Nyquist frequency and sampling frequency as follows:

$$\omega_N = 10 \times \omega_C \qquad \omega_S = 20 \times \omega_C$$

C.5 Aliasing

Aliasing poses a significant and critical issue in the vicinity of the analog-to-digital converter.

Theorem C.5.1 (Shannon)**.** *The maximum frequency content of a continuous-time signal to be sampled must be smaller than or equal to the Nyquist frequency:*

$$f \leq f_N$$

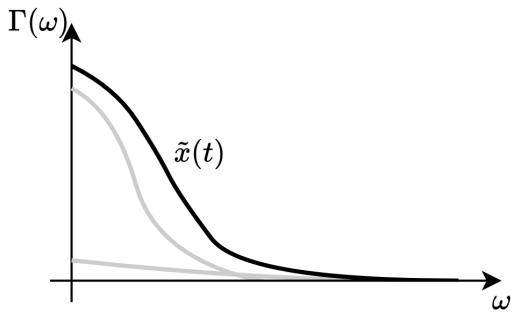


Figure C.5: Sampled signal

The sampled signal is a combination of the true signal and noise:

$$\tilde{\mathbf{x}}(t) = \mathbf{x}(t) + \varepsilon$$

It's worth noting that every sampled signal has a maximum frequency content defined by the point at which the spectrum of $\tilde{\mathbf{x}}(t)$ becomes null.

C.5.1 Anti-aliasing

Consider a sampled signal $\tilde{\mathbf{x}}(t)$ with a null spectrum at a frequency of 2000 Hz. The dilemma arises when we desire $f_N = 500\,Hz$ ($f_S = 1000\,Hz$), which is significantly lower than 2000 Hz. Direct sampling at $f_S = 1000\,Hz$ violates Shannon's theorem, leading to potential aliasing problems.

Analog filter The conventional approach to addressing this issue involves employing an analog pre-filter (anti-aliasing pre-filter).

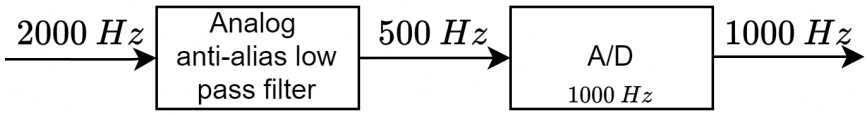


Figure C.6: Analog filter

The output yields a discrete-time signal at $f_S = 1000\,Hz$ with a maximum frequency content of 500 Hz, meeting the minimum requirement of Shannon's theorem.

Digital filter Alternatively, a fully digital solution can also be utilized.

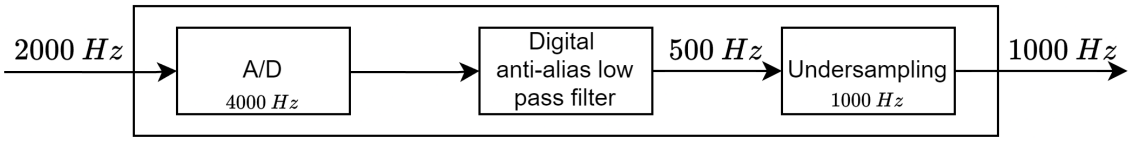


Figure C.7: Digital filter

This approach involves undersampling, taking one sample out of every four. The resulting output is a discrete-time signal at $f_S = 1000\,Hz$ with a maximum frequency content of $500\,Hz$, adhering to Shannon's theorem. The key distinction is the initial oversampling to satisfy Shannon's theorem, eliminating the necessity for an analog pre-filter.