Literature Review
Embedded Control Systems

student: Dirk Vos
University: Delft University of Technology
Company: AVL List GmbH
version 1.0(censured)

February 2, 2018
Abstract

This literature review will focus on two important topics: designing the actual control system and getting an accurate process model so the control system will be able to predict the process behaviour as closely as possible.

First topic is the design of a linear control system, which is used to automatise the process of calibrating gas analysers to a variety of gas pressures and gas concentrations, on an embedded system using C++ programming language. This will include discussing possible design methods, robustness, optimality and a small piece on dealing with non linearities.

Second topic will be identification. How to acquire the 4 matrices (A,B,C and D) needed by modern linear control systems or how to get the transfer function parameters in order to create an as accurate model of the process as possible? Identification methods try to fit a model to a bunch of input and output data acquired through carefully chosen experiments of the process. Having an accurate model of the process is critical to designing a good linear control system, which eventually result in a fast and accurate calibration of the gas analysers.
Contents

0.1 Designing a Linear Control System .......................... 2
  0.1.1 Introduction ........................................ 2
  0.1.2 Existing Control Systems ............................. 4
  0.1.3 Conclusion ......................................... 32

0.2 Identification of the Process Model ......................... 33
  0.2.1 Introduction ........................................ 33
  0.2.2 Existing Identification methods ....................... 36
  0.2.3 Conclusion ......................................... 43
0.1 Designing a Linear Control System

0.1.1 Introduction

For a long time mathematical models of dynamic systems have been a set of ordinary differential equations (ODE) like the oldest example in the book: a mass spring damper system, see equation 1.

\[ m\ddot{q} + c(\dot{q}) + kq = 0 \] (1)

The variable \( q \) represents the position of the mass \( m \) with respect to its rest position. We use the notation \( \dot{q} \) to denote the derivative of \( q \) with respect to time (i.e., the velocity of the mass) and \( \ddot{q} \) to represent the second derivative (acceleration). The spring is assumed to satisfy Hookes law, which says that the force is proportional to the displacement. The friction element (damper) is taken as a non linear function \( c(\dot{q}) \), which can model effects such as stiction and viscous drag. The position \( q \) and velocity \( \dot{q} \) represent the instantaneous state of the system. We say that this system is a second-order system since the dynamics depend on the first two derivatives of \( q \).

To be able to model external disturbances and controlled forces as well, the ODE’s were expanded with an input term \( u \), see equation 2. This equation is called a controlled differential equation, where \( u \) represents the effect of external inputs. Adding the input makes the model richer and allows new questions to be posed. For example, we can examine what influence external disturbances have on the trajectories of a system. Or, in the case where the input variable is something that can be modulated in a controlled way, we can analyse whether it is possible to steer the system from one point in the state space to another through proper choice of the input.

\[ m\ddot{q} + c(\dot{q}) + kq = u \] (2)

In the late 1950’s control theory shifted towards a more state space approach, which was inspired by the use of state space perspective by mechanical engineering. Before the 1950’s, control theory was largely influenced by the input/output view used in electrical engineering (the use of transfer functions to describe dynamics, this topic will be addressed later in this chapter). The state space perspective is called modern control theory and the older transfer function perspective (input/output view) is called classical control theory.

The development of state space models involved modifying the models from mechanics to include external actuators and sensors and utilizing more general forms of equations. In control, the model given by equation 2 was replaced by equation 3

\[ \dot{x} = \frac{dx}{dt} = f(x,u), y = h(x,u) \] (3)

where \( x \) is a vector of state variables, \( u \) is a vector of control signals and \( y \) is a vector of measurements. The term \( \frac{dx}{dt} \) represents the derivative of \( x \) with respect to time, now considered a vector, and \( f \) and \( h \) are (possibly nonlinear) mappings of their arguments to vectors of the appropriate dimension.
Note that in the control formulation we model dynamics as first-order differential equations, but we will see that this can capture the dynamics of higher-order differential equations by appropriate definition of the state and the maps $f$ and $h$. Adding inputs and outputs has increased the richness of the classical problems and led to many new concepts. For example, it is natural to ask if possible states $x$ can be reached with the proper choice of $u$ (reachability) and if the measurement $y$ contains enough information to reconstruct the state (observability). More on these topics will follow soon.

A final development in building the control point of view was the emergence of disturbances and model uncertainty as critical elements in the theory. The simple way of modeling disturbances as deterministic signals like steps and sinusoids has the drawback that such signals cannot be predicted precisely. A more realistic approach is to model disturbances as random signals. This viewpoint gives a natural connection between prediction and control. The dual views of input/output representations and state space representations are particularly useful when modeling uncertainty since state models are convenient to describe a nominal model but uncertainties are easier to describe using input/output models (often via a frequency response description). There is a special notation for systems that are linear, in that case equation 3 can be rewritten to a linear time invariant system (LTI), see equation 4.

\[
\dot{x} = \frac{dx}{dt} = Ax + Bu, \quad y = Cx + Du
\]  

where $A$, $B$, $C$ and $D$ are constant matrices. Such a system is said to be linear and time-invariant, or LTI for short. The matrix $A$ is called the dynamics matrix, the matrix $B$ is called the control matrix, the matrix $C$ is called the sensor matrix and the matrix $D$ is called the direct term. Frequently systems will not have a direct term, indicating that the control signal does not influence the output directly. A different form of linear differential equations, generalizing the second-order dynamics from mechanics, is an equation with the form of equation 5.

\[
\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_n y = u
\]  

where $t$ is the independent (time) variable, $y(t)$ is the dependent (output) variable and $u(t)$ is the input. The notation $d^k y / dt^k$ is used to denote the $k$th derivative of $y$ with respect to $t$, sometimes also written as $y(k)$. The controlled differential equation 5 is said to be an $n$th-order system. This system can be converted into state space form by defining the state vector $x$ as:

\[
x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} \frac{d^{n-1} y}{dt^{n-1}} \\ \frac{d^{n-2} y}{dt^{n-2}} \\ \cdots \\ \frac{dy}{dt} \\ y \end{pmatrix}
\]

The state space equation will now become:
\[ \begin{align*}
\frac{d}{dt} \begin{pmatrix}
x_1 \\
x_2 \\
\vdots \\
x_{n-1} \\
x_n
\end{pmatrix} &= \begin{pmatrix}
-a_1 x_1 - \cdots - a_n x_n \\
x_1 \\
\vdots \\
x_{n-2} \\
x_{n-1}
\end{pmatrix} + \begin{pmatrix}
u \\
0 \\
\vdots \\
0 \\
0
\end{pmatrix}, \\
y &= x_n
\end{align*} \]

The system can be made even more general by making the output equal to a combination of system states:

\[ y = (c_1 \ c_2 \ \cdots \ c_n) \ x + du \]

In this case the system matrices become:

\[ \frac{d}{dt} \begin{pmatrix}
x_1 \\
x_2 \\
\vdots \\
x_{n-1} \\
x_n
\end{pmatrix} = \begin{pmatrix}
-a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\
0 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0 \\
0 & 0 & \cdots & 0 & 1
\end{pmatrix} \begin{pmatrix}
x \\
0 \\
\vdots \\
0 \\
0
\end{pmatrix} + \begin{pmatrix}
1 \\
0 \\
\vdots \\
0 \\
0
\end{pmatrix} u
\]

\[ y = (c_1 \ c_2 \ \cdots \ c_n) \ x + (d) u \]

This particular form of a linear state space system is called reachable canonical form and a SISO form (Single Input Single Output), more on this will be addressed later on.

### 0.1.2 Existing Control Systems

Control engineering has its own categorization depending on the different methodologies\(^1\) used, which are as follows.

- **Classical Control Engineering:**
  The systems are usually represented by using ordinary differential equations. In classical control engineering, these equations are often transformed and analysed in transformed domain. Laplace transform and z transform are examples. This method is commonly used in Single Input Single Output systems.

- **Modern Control Engineering:**
  In modern control engineering higher order differential equations are converted to first order differential equations. These equations are solved very similar to vector method. By doing so, many complications dealt in solving higher order differential equations are solved. These are applied in Multiple Input Multiple Output systems where analysis in frequency domain is not possible. Nonlinearities with multiple variables are solved by modern methodology. State space vectors, Eigenvalues and Eigenvectors longs to this category. State Variables describe the input, output and system variables.

- **Robust Control Engineering:**
  In robust control methodology, the changes in performance of system with change in parameters are measured for optimization. This aids in widening the stability and performance, also in finding alternate solutions. Hence in robust control the environment, internal in accuracies, noises and disturbances are considered to reduce the fault in system.

\(^1\)https://www.electrical4u.com/control-engineering-historical-review- and-types-of-control-engineering
• **Optimal Control Engineering:**
  In optimal control engineering, the problem is formulated as a mathematical model of the process, with physical constraints and performance constraints, to minimize the cost function. Thus, optimal control engineering is the most feasible solution for designing a system with minimum cost.

• **Adaptive Control Engineering:**
  In adaptive control engineering, the controllers employed are adaptive controllers in which parameters are made adaptive by some mechanism. The block diagram given below shows an adaptive control system. Adaptive control system. In this kind of controllers an additional loop for parameter adjustment is present in addition to the normal feedback of process.

• **Non linear Control Engineering:**
  Non linear control engineering focuses on the non linearity which cannot be represented by using linear ordinary differential equations. This system will exhibit multiple isolated equilibrium points, limit cycles, bifurcations with finite escape time. The main limitation is that it requires laborious mathematical analysis. In this analysis the system is divided into linear part and non linear part.

• **Game Theory:**
  In game theory, each system will have to reduce its cost function against the disturbances/noises. Hence it is a study of conflict and cooperation. The disturbances will try to maximize the cost function. This theory is related to robust and optimal control engineering.

Not every one of these control systems will be covered by the following sections, just the important ones are selected: classical control, modern control, robust control, optimal control, non linear control. Game theory and adaptive control are considered to be out of the scope of this document.

**Classical Control Engineering**

Figure 1 is a block diagram for a typical control system, consisting of a process to be controlled and a controller that combines feedback and feedforward. Such a system can be analyzed in two ways: state space description of all the blocks or by describing the input/output characteristics of the system using transfer functions.

Since it is the inputs and outputs that are used to connect the systems, one could expect that this point of view would allow an understanding of the overall behaviour of the system.

A transfer function describes the output relation to the input of the system. The power of transfer functions is that they provide a particularly convenient representation in manipulating and analysing complex linear feedback systems. There are many graphical representations of transfer functions that capture interesting properties of the underlying dynamics.

Transfer functions also make it possible to express the changes in a system because of modelling error, which is essential when considering sensitivity to process variations in order to check robustness. More specifically, using transfer functions, it is possible to analyze what happens when dynamic models are
Figure 1: A block diagram for a feedback control system. The reference signal \( r \) is fed through a reference shaping block, which produces the signal that will be tracked. The error between this signal and the output is fed to a controller, which produces the input to the process. Disturbances and noise are included as external signals at the input and output of the process dynamics.

approximated by static models or when high-order models are approximated by low-order models. One consequence is that we can introduce concepts that express the degree of stability of a system.

The input/output dynamics of a linear system has two components: the initial condition response and the forced response. In addition, we can speak of the transient properties of the system and its steady state response to an input. The transfer function focuses on the steady-state forced response to a given input and provides a mapping between inputs and their corresponding outputs. To get an idea of how a transfer function from a linear system looks like, we will derive a transfer function in terms of the exponential response of a linear system in the following section.

To investigate how a linear system responds to an exponential input \( u(t) = e^{st} \), we consider the state space system:

\[
\frac{dx}{dt} = Ax + Bu, \quad y = Cx + Du
\]

With the input signal being \( u(t) = e^{st} \) and assume that \( s \neq \lambda_j(A), \ j = 1, ..., n \). Where \( \lambda_j(A) \) is the jth eigenvalue of A. The state is then given by:

\[
x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Be^{st}d\tau = e^{At}x(0) + cAt \int_0^t e^{(sI-A)\tau}Bd\tau
\]

When evaluating the integral from equation 6, the following equation arises:

\[
x(t) = e^{At}x(0) + e^{At}(sI-A)^{-1}(e^{(sI-A)t} - I)B = e^{At}(x(0) - (sI-A)^{-1}B) + (sI-A)^{-1}Be^{st}
\]

When substituting this equation in the output equation, the relation between input and output is given:

\[
y(t) = Cu(t) + Du(t)
\]

\[
y(t) = Ce^{At}(x(0) - (sI-A)^{-1}B) + (C(sI-A)^{-1}B + D)e^{st}
\]
Figure 2: Examples of exponential signals. The top row corresponds to exponential signals with a real exponent, and the bottom row corresponds to those with complex exponents. The dashed line in the last two cases denotes the bounding envelope for the oscillatory signals. In each case, if the real part of the exponent is negative then the signal decays, while if the real part is positive then it grows.

Now before equation 7 is further investigated, knowledge of the matrix exponential $e^{At}$ is required.

The matrix exponential holds all the system modes (all the eigenvalues of the system) and thus will tell how the states respond. Considering the following homogeneous equation:

$$\frac{dx}{dt} = Ax$$

For a scalar differential of such kind $\frac{dx}{dt} = ax$, the solution is given by the exponential: $x(t) = e^{at}x(0)$. By looking at the Taylor series definition of the scalar exponential:

$$e^t = I + t + \frac{1}{2!}t^2 + \frac{1}{3!}t^3 + \sum_{k=0}^{\infty} \frac{1}{k!}t^k$$

Then the matrix exponential of this system is will be defined as (substituting $t$ for $At$):

$$e^{At} = I + At + \frac{1}{2}A^2t^2 + \frac{1}{3!}A^3t^3 + \sum_{k=0}^{\infty} \frac{1}{k!}A^kt^k$$

So then the scalar solution can also be extended to the matrices, giving:

$$x(t) = e^{At}x(0)$$

Now, calculating the matrix exponential can be quite computational heavy so linear algebra is used to simplify these computations by making coordinate changes. More specifically: diagonalising matrices, because the exponential
of diagonal matrices is easier to calculate. As an example of how the matrix exponential looks like when \( A \) is in a diagonal form:

\[
A = \begin{pmatrix}
\lambda_1 & 0 & \cdots & 0 \\
0 & \lambda_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda_n
\end{pmatrix}, \quad (At)^k = \begin{pmatrix}
\lambda_1^k & 0 & \cdots & 0 \\
0 & \lambda_2^k & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda_n^k
\end{pmatrix}
\]

\[
e^{At} = \begin{pmatrix}
e^{\lambda_1 t} & 0 & \cdots & 0 \\
0 & e^{\lambda_2 t} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & e^{\lambda_n t}
\end{pmatrix}
\]

Unfortunately the \( A \) matrix is not always diagonalisable, for example when an eigenvalue has an algebraic multiplicity higher than 1. In this case the Jordan form can be used.

The Jordan form is a diagonal matrix with so called Jordan blocks on the diagonal. When every eigenvalue has a algebraic multiplicity of 1, then the Jordan blocks are all 1 x 1 and it will be a normal diagonal matrix. If an eigenvalue has an algebraic multiplicity of 2 or more, then ones will appear on the diagonal line above the eigenvalues:

\[
J = \begin{pmatrix}
J_1 & 0 & \cdots & 0 \\
0 & J_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & J_n
\end{pmatrix}, \quad J_i = \begin{pmatrix}
\lambda_i & 1 & 0 & \cdots & 0 \\
0 & \lambda_i & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
0 & 0 & 0 & \cdots & \lambda_i
\end{pmatrix}
\]

The dimensions of \( J_i \) depends on the algebraic multiplicity of the eigenvalue \( \lambda_i \), so translating this property to the matrix exponential will result in:

\[
e^J = \begin{pmatrix}
e^{J_1} & 0 & \cdots & 0 \\
0 & e^{J_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & e^{J_n}
\end{pmatrix}, \quad e^{J,t} = \begin{pmatrix}
1 & \frac{t}{\eta} & \cdots & \frac{t^{n-1}}{(n-1)!} \\
0 & 1 & \cdots & \frac{t^{n-2}}{(n-2)!} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & t \\
0 & 0 & 0 & \cdots & 1
\end{pmatrix} e^{\lambda_i t}
\]

Also here, \( e^{J,t} \) will be the dimension of the algebraic multiplicity of \( \lambda_i \).

So to get back to the equation 7, this is a linear combination of the exponential functions \( e^{st} \) and \( e^{At} \). The first term in equation 7 is the transient response of the system. The matrix exponential \( e^{At} \) can be written in terms of the eigenvalues of \( A \) (using the Jordan form in the case of repeated eigenvalues), and hence the transient response is a linear combination of terms of the form \( e^{\lambda_j t} \), where \( \lambda_j \) are eigenvalues of \( A \). If the system is stable, then \( e^{At} \to 0 \) as \( t \to \infty \) and this term dies away. The second term of equation 7 is proportional to the input \( u(t) = e^{st} \). This term is called the pure exponential response.

If the initial state is chosen as \( x(0) = (sI - A)^{-1}B \) then the output consists of only the pure exponential response and both the state and the output are proportional to the input:

\[
x(t) = (sI - A)^{-1}Be^{st} = (sI - A)^{-1}Bu(t) \\
y(t) = (C(sI - A)^{-1}B + D)e^{st} = (C(sI - A)^{-1}B + D)u(t)
\]

8
This is also the output we see in steady state, when the transients represented by the first term in equation 9 have died out. The map from the input to the output is:

\[ G_{yu}(s) = C(sI - A)^{-1}B + D \]  

Equation 9 is the transfer function from \( u \) to \( y \) for the system 1, and we can write \( y(t) = G_{yu}(s)u(t) \) for the case that \( u(t) = e^{st} \). An important point in the derivation of the transfer function is the fact that we have restricted \( s \) so that \( s \neq \lambda_j(A) \), the eigenvalues of \( A \). At those values of \( s \), we see that the response of the system is singular (since \( sI - A \) will fail to be invertible). If \( s = \lambda_j(A) \), the response of the system to the exponential input \( u(t) = e^{\lambda_j t} \) is \( y = p(t)e^{\lambda_j t} \), where \( p(t) \) is a polynomial of degree less than or equal to the multiplicity of the eigenvalue \( \lambda_j \)

To illustrate how transfer functions could be used, a small example of a damped linear oscillator will follow. First the transfer function is evaluated and then the transfer function is used to calculate the output response to a step input. The state dynamics of the system is:

\[
\frac{dx}{dt} = \left( \begin{array}{cc}
0 & \omega_0 \\
-\omega_0 & -2\zeta\omega_0
\end{array} \right) x + \left( \begin{array}{c}
0 \\
\kappa\omega_0
\end{array} \right) u, \quad y = \left( \begin{array}{c}
1 \\
0
\end{array} \right) x
\]

So the system matrices are in this case: \( A = \left( \begin{array}{cc}
0 & \omega_0 \\
-\omega_0 & -2\zeta\omega_0
\end{array} \right) \), \( B = \left( \begin{array}{c}
0 \\
\kappa\omega_0
\end{array} \right) \), \( C = \left( \begin{array}{c}
1 \\
0
\end{array} \right), D = 0 \)

This system is stable if \( \zeta > 0 \), and so we can look at the steady-state response to an input \( u = e^{st} \),

\[
G_{yu}(s) = C(sI - A)^{-1}B + D
\]

\[
= (1 \ 0) \left( \begin{array}{cc}
0 & \omega_0 \\
-\omega_0 & -2\zeta\omega_0
\end{array} \right)^{-1} \left( \begin{array}{c}
0 \\
\kappa\omega_0
\end{array} \right)
\]

\[
= (1 \ 0) \left( \frac{1}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \left( \begin{array}{cc}
0 & \omega_0 \\
-\omega_0 & -2\zeta\omega_0
\end{array} \right) \right) \left( \begin{array}{c}
0 \\
\kappa\omega_0
\end{array} \right)
\]

To compute the steady-state response to a step function, we set \( s = 0 \) and we see that

\[
u = 1 \rightarrow y = G_{yu}(0)u = k.
\]

Instead of using the state dynamics of a system to get the transfer function, we can also get it by analysing the controlled difference equations, like equation 10.

\[
\frac{d^ny}{dt^n} + a_1 \frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_n y = b_0 \frac{du}{dt^n} + b_1 \frac{d^{n-1}u}{dt^{n-1}} + \cdots + b_m u
\]

where \( u \) is the input and \( y \) is the output. This type of description arises in many applications, bicycle dynamics and AFM modeling are two specific examples. Note that here we have generalized our previous system description to allow both the input and its derivatives to appear. To determine the transfer function of the system expressed with equation 10, we will make the input \( u(t) = e^{st} \). Since the system is linear, there is an output of the system that
is also an exponential function \( y(t) = y_0e^{st} \). Inserting the signals into equation 10, we find

\[
(s^n + a_1s^{n-1} + \cdots + a_n)y_0e^{st} = (b_0s^m + b_1s^{m-1} + \cdots + b_m)e^{st}
\]

the response of the system can be completely described by two polynomials:

\[
a(s) = s^n + a_1s^{n-1} + \cdots + a_n, b(s) = b_0s^m + b_1s^{m-1} + \cdots + b_m
\]

The polynomial \( a(s) \) is the characteristic polynomial of the ordinary differential equation. If \( a(s) \neq 0 \), it follows that

\[
y(t) = y_0e^{st} = \frac{b(s)}{a(s)}e^{st}
\]

The transfer function of the system expressed with equation 10 is thus the rational function:

\[
G(s) = \frac{b(s)}{a(s)}
\]

where the polynomials \( a(s) \) and \( b(s) \) are given earlier. Notice that the transfer function can be obtained by inspection, since the coefficients of \( a(s) \) and \( b(s) \) are precisely the coefficients of the derivatives of \( u \) and \( y \). The order of the transfer function is defined as the order of the denominator polynomial.

The transfer function has many useful interpretations and the features of a transfer function are often associated with important system properties. Three of the most important features are the gain and the locations of the poles and zeros.

The zero frequency gain of a system is given by the magnitude of the transfer function at \( s = 0 \). It represents the ratio of the steady-state value of the output with respect to a step input (which can be represented as \( u = e^{st} \) with \( s = 0 \)). For a state space system, we computed the zero frequency gain using the definition of the convolution integral:

\[
y(t) = Ce^{At}x(0) + \int_0^t Ce^{A(t-\tau)}Bu(\tau)d\tau + Du(t)
\]

When setting the initial condition to zero \( x(0) = 0 \) and solve the integral, then the equation looks like this:

\[
y(t) = C \int_0^t e^{A\sigma} B d\sigma + D
\]

\[
= CA^{-1}e^{At}B - CA^{-1}B + D
\]

where the first part \( CA^{-1}e^{At}B \) identifies as the transient behaviour of the system and the second part \( D - CA^{-1}B \) identifies as the steady state behaviour of the system.

The first term, the transient response, decays to zero as \( t \to \infty \). The second term is the steady-state response and represents the value of the output for large time.

A sample step response is shown in Figure 3. Several terms are used when referring to a step response. The steady-state value \( y_{ss} \) of a step response is the final level of the output, assuming it converges. The rise time \( T_r \) is the amount of time required for the signal to go from 10% of its final value to 90% of its final value. It is possible to define other limits as well, but these are used most often in control engineering. The overshoot \( M_p \) is the percentage of the final value by which the signal initially rises above the final value. This usually assumes
that future values of the signal do not overshoot the final value by more than this initial transient, otherwise the term can be ambiguous. Finally, the settling time $T_s$ is the amount of time required for the signal to stay within 2% of its final value for all future times. The settling time is also sometimes defined as reaching 1% or 5% of the final value. In general these performance measures can depend on the amplitude of the input step, but for linear systems the last three quantities defined above are independent of the size of the step.

So the zero frequency gain of a system is defined by $G(0) = D - CA^{-1}B$. For a system written as a linear differential equation:

$$
\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_n y = b_0 \frac{d^m u}{dt^m} + b_1 \frac{d^{m-1} u}{dt^{m-1}} + \cdots + b_m u
$$

if we assume that the input and output of the system are constants $y_0$ and $u_0$, then we find that $a_n y_0 = b_m u_0$. Hence the zero frequency gain is

$$
G(0) = \frac{y_0}{u_0} = \frac{b_m}{a_n}
$$

Next consider a linear system with the rational transfer function:

$$
G(s) = \frac{b(s)}{a(s)}
$$

The roots of the polynomial $a(s)$ are called the poles of the system, and the roots of $b(s)$ are called the zeros of the system. If $p$ is a pole, it follows that $y(t) = e^{pt}$ is a solution of differential equation (like equation 10) with $u = 0$ (the homogeneous solution). A pole $p$ corresponds to a mode of the system with corresponding modal solution $e^{pt}$. The unforced motion of the system after an arbitrary excitation is a weighted sum of modes.

Zeros have a different interpretation. Since the pure exponential output corresponding to the input $u(t) = e^{st}$ with $a(s) \neq 0$ is $G(s)e^{st}$, it follows that the pure exponential output is zero if $b(s) = 0$. Zeros of the transfer function thus block transmission of the corresponding exponential signals.

For a state space system with transfer function $G(s) = C(sI - A)^{-1}B + D$, the poles of the transfer function are the eigenvalues of the matrix $A$ in the state space model. One easy way to see this is to notice that the value of $G(s)$
is unbounded when s is an eigenvalue of a system since this is precisely the set of points where the characteristic polynomial \( \lambda(s) = \det(sIA) = 0 \) (and hence \( sIA \) is non invertible). It follows that the poles of a state space system depend only on the matrix A, which represents the intrinsic dynamics of the system. We say that a transfer function is stable if all of its poles have negative real part.

To find the zeros of a state space system, we observe that the zeros are complex numbers s such that the input \( u(t) = u_0e^{st} \) gives zero output. Inserting the pure exponential response \( x(t) = x_0e^{st} \) and \( y(t) = 0 \) in equation 4 gives:

\[ se^{st}x_0 = Ax_0e^{st} + Bu_0e^{st}, 0 = Ce^{st}x_0 + De^{st}u_0 \]

which can be written like:

\[ \begin{pmatrix} A - sI & B \\ C & D \end{pmatrix} \begin{pmatrix} x_0 \\ u_0 \end{pmatrix} e^{st} = 0 \]

This equation has a solution with non zero \( x_0 \) and \( u_0 \), only if the matrix on the left does not have full rank. The zeros are thus the values s such that the matrix:

\[ \begin{pmatrix} A - sI & B \\ C & D \end{pmatrix} \]

loses rank.

Since the zeros depend on A, B, C and D, they therefore depend on how the inputs and outputs are coupled to the states. Notice in particular that if the matrix B has full row rank, then the matrix above has n linearly independent rows for all values of s. Similarly there are n linearly independent columns if the matrix C has full column rank. This implies that systems where the matrix B or C is square and full rank do not have zeros. In particular it means that a system has no zeros if it is fully actuated (each state can be controlled independently, fully reachable) or if the full state is measured (fully observable).

A convenient way to view the poles and zeros of a transfer function is through a pole zero diagram, as shown in Figure 4. In this diagram, each pole is marked with a cross, and each zero with a circle. If there are multiple poles or zeros at a fixed location, these are often indicated with overlapping crosses or circles (or other annotations). Poles in the left half-plane correspond to stable modes of the system, and poles in the right half-plane correspond to unstable modes. We thus call a pole in the left-half plane a stable pole and a pole in the right-half plane an unstable pole. A similar terminology is used for zeros, even though the zeros do not directly relate to stability or instability of the system. Notice that the gain must also be given to have a complete description of the transfer function.

Because transfer functions are often polynomials in s, it can sometimes happen that the numerator and denominator have a common factor, which can be cancelled. Sometimes these cancellations are simply algebraic simplifications, but in other situations they can mask potential fragilities in the model. In particular, if a pole/zero cancellation occurs because terms in separate blocks that just happen to coincide, the cancellation may not occur if one of the systems is slightly perturbed. In some situations this can result in severe differences between the expected behavior and the actual behavior.
Figure 4: A pole zero diagram for a transfer function with zeros at 5 and 1 and poles at 3 and 22j. The circles represent the locations of the zeros, and the crosses the locations of the poles. A complete characterization requires that we also specify the gain of the system.

To illustrate how pole/zero cancellation can throw a system of balance, we use the following example:

\[ C(s) = \frac{n_c(s)}{d_c(s)}, \quad P(s) = \frac{n_p(s)}{d_p(s)} \]

Where \( C \) is the controller transfer function and \( P \) is the Process transfer function from figure 1. This leads to the following transfer function from \( r \) to \( e \):

\[ G_{er}(s) = \frac{1}{1 + PC} = \frac{d_c(s)d_p(s)}{d_c(s)d_p(s) + n_c(s)n_p(s)} \]

Now if the controller has a zero at \( s = -a \) and the process has a pole at \( s = -a \), then we will have pole zero cancellation:

\[ G_{er}(s) = \frac{(s+a)d_c(s)d_p(s)}{(s+a)d_c(s)d_p(s) + (s+a)n_c(s)n_p(s)} = \frac{d_c(s)d_p(s)}{d_c(s)d_p(s) + n_c(s)n_p(s)} \]

where \( n_c(s) \) and \( d_p(s) \) represent the relevant polynomials with the term \( (s+a) \) factored out. In the case when \( a < 0 \) (so that the zero or pole is in the right half-plane), we see that there is no impact on the transfer function \( G_{er} \). Suppose instead that we compute the transfer function from \( d \) to \( e \), which represents the effect of a disturbance on the error between the reference and the output. This transfer function is given by

\[ G_{ed}(s) = \frac{-d_c(s)d_p(s)}{(s+a)d_c(s)d_p(s) + (s+a)n_c(s)n_p(s)} \]

Notice that if \( a < 0 \), then the pole is in the right half-plane and the transfer function \( G_{ed} \) is unstable. Hence, even though the transfer function from \( r \) to \( e \) appears to be okay (assuming a perfect pole/zero cancellation), the transfer function from \( d \) to \( e \) can exhibit unbounded behaviour. This unwanted behaviour is typical of an unstable pole/zero cancellation.

It turns out that the cancellation of a pole with a zero can also be understood in terms of the state space representation of the systems. Reachability or observability is lost when there are cancellations of poles and zeros. A
The combination of block diagrams and transfer functions is a powerful way to represent control systems. Transfer functions relating different signals in the system can be derived by purely algebraic manipulations of the transfer functions of the blocks using block diagram algebra. Figure 5 shows some basic combinations of transfer functions. Consider a system that is a cascade combination of systems with the transfer functions $G_1(s)$ and $G_2(s)$, as shown in Figure 5. Let the input of the system be $u = e^{st}$. The pure exponential output of the first block is the exponential signal $G_1u$, which is also the input to the second system. The pure exponential output of the second system is

$$y = G_2(G_1u) = G_2G_1u$$

The transfer function of the series connection is thus $G = G_2G_1$, i.e., the product of the transfer functions.

![Figure 5: Interconnections of linear systems. Series (a), parallel (b) and feedback (c) connections are shown. The transfer functions for the composite systems can be derived by algebraic manipulations assuming exponential functions for all signals.](image)

The order of the individual transfer functions is due to the fact that we place the input signal on the right-hand side of this expression, hence we first multiply by $G_1$ and then by $G_2$. Unfortunately, this has the opposite ordering from the diagrams that we use, where we typically have the signal flow from left to right, so one needs to be careful.

The transfer function(s) of a control scheme can be deduced by analysing the block diagram just like discussed above. Let us analyse the typical control block diagram shown in figure 6.

The system has three blocks representing a process $P$, a feedback controller $C$ and a feedforward controller $F$. Together, $C$ and $F$ define the control law for the system. There are three external signals: the reference (or command signal) $r$, the load disturbance $d$ and the measurement noise $n$. A typical problem is to find out how the error $e$ is related to the signals $r$, $d$ and $n$.

To derive the relevant transfer functions we assume that all signals are exponential signals, drop the arguments of signals and transfer functions and trace the signals around the loop. We begin with the signal in which we are interested, in this case the control error $e$, given by:

$$e = Fr - y$$
The signal $y$ is the sum of $n$ and $\eta$, where $\eta$ is the output of the process:

$$y = n + \eta, \eta = P(d + u), u = Ce$$

Combining these equations gives:

$$e = Fr - y = Fr - (n + \eta) = Fr - (n + P(d + u)) = Fr - (n + P(d + Ce))$$

and thus:

$$e = Fr - n - Pd - PCe$$

Finally, solving this equation for $e$ gives:

$$e = \frac{F}{1+PC}r - \frac{1}{1+PC}n - \frac{P}{1+PC}d = G_{er}r + G_{en}n + G_{ed}d$$

and the error is thus the sum of three terms, depending on the reference $r$, the measurement noise $n$ and the load disturbance $d$. The functions:

$$G_{er} = \frac{F}{1+PC}, G_{en} = -\frac{1}{1+PC}, G_{ed} = -\frac{P}{1+PC}$$

are transfer functions from reference $r$, noise $n$ and disturbance $d$ to the error $e$.

A very popular way to visualize transfer functions (besides a pole/zero plot) is the use of a Bode plot, which shows the frequency response of the linear system in question. The frequency response of a linear system can be computed from its transfer function by setting $s = i\omega$, corresponding to a complex exponential:

$$u(t) = e^{i\omega t} = \cos(\omega t) + isin(\omega t)$$

The resulting output will have the form:

$$y(t) = G(i\omega)e^{i\omega t} = Me^{i(\omega t + \phi)} = M\cos(\omega t + \phi) + iM\sin(\omega t + \phi)$$

where $M$ and $\phi$ are respectively gain and phase of the transfer function $G$:

$$M = |G(i\omega)|, \phi = \arctan\left(\frac{\Im(G(i\omega))}{\Re(G(i\omega))}\right)$$
An example of a bode plot is show in figure 7, where a transfer function of a controller is plotted.

![Bode plot](image)

Figure 7: Bode plot of the transfer function $C(s) = 20 + \frac{10}{s} + 10s$ corresponding to an ideal PID controller. The top plot is the gain curve and the bottom plot is the phase curve. The dashed lines show straight-line approximations of the gain curve and the corresponding phase curve.

The Bode plot gives a quick overview of a system. Since any signal can be decomposed into a sum of sinusoids, it is possible to visualize the behaviour of a system for different frequency ranges. The system can be viewed as a filter that can change the amplitude (and phase) of the input signals according to the frequency response. For example, if there are frequency ranges where the gain curve has constant slope and the phase is close to zero, the action of the system for signals with these frequencies can be interpreted as a pure gain. Similarly, for frequencies where the slope is +1 and the phase close to 90 deg, the action of the system can be interpreted as a differentiator.

Three common types of frequency responses are shown in Figure 8. The system in Figure 8a is called a low-pass filter because the gain is constant for low frequencies and drops for high frequencies. Notice that the phase is zero for low frequencies and 180 for high frequencies. The systems in Figure 8b and 8c are called a band-pass filter and high-pass filter for similar reasons.

To illustrate how different system behaviours can be read from the Bode plots we consider the band-pass filter in Figure 8b. For frequencies around $\omega = \omega_0$, the signal is passed through with no change in gain. However, for frequencies well below or well above $\omega_0$, the signal is attenuated. The phase of the signal is also affected by the filter, as shown in the phase curve. For frequencies below $\frac{\omega_0}{10}$ there is a phase lead of 90 deg, and for frequencies above $100\omega_0$ there is a phase lag of 90 deg. These actions correspond to differentiation and integration of the signal in these frequency ranges.

One last very useful addition to transfer functions is the use of the Laplace transform. The Laplace transform transforms equations into the Laplace domain which most of the times make computations a lot more simple. An example:
Figure 8: Bode plots for low-pass, band-pass and high-pass filters. The top plots are the gain curves and the bottom plots are the phase curves. Each system passes frequencies in a different range and attenuates frequencies outside of that range.

\[ y(t) = \int_{0}^{\infty} h(t-\tau)u(\tau)\,d\tau \]

The equation above represents a linear time invariant system with zero initial state, where \( y \) is the output and \( u \) is the input and \( h(t) \) being the impulse response of the system. Now if the equation is converted to the Laplace domain, using the following Laplace transformation:

\[ F(s) = \int_{0}^{\infty} e^{-st}f(t)\,dt \]

With the real part of \( s \) bigger than some finite real number. Which converts the convolution integral to:

\[
Y(s) = \int_{0}^{\infty} e^{-st}y(t)\,dt = \int_{0}^{\infty} e^{-st} \int_{0}^{\infty} h(t-\tau)u(\tau)\,d\tau\,dt \\
= \int_{0}^{\infty} \int_{0}^{\infty} e^{-s(t-\tau)}e^{s\tau}h(t-\tau)u(\tau)\,d\tau\,dt \\
= \int_{0}^{\infty} e^{-st}u(\tau)\,d\tau \int_{0}^{\infty} e^{s\tau}h(t)\,dt = H(s)U(s)
\]

Thus, the input/output response is given by \( Y(s) = H(s)U(s) \), where \( H \), \( U \) and \( Y \) are the Laplace transforms of \( h \), \( u \) and \( y \). A mathematical interpretation is that the Laplace transform of a convolution is the product of the transforms of the functions that are convolved. The fact that the formula \( Y(s) = H(s)U(s) \) is much simpler than a convolution is one reason why Laplace transforms have become popular in engineering.

We can also use the Laplace transform to derive the transfer function for a state space system. Consider, for example, a linear state space system described by equation 4:

\[
\frac{dx}{dt} = Ax + Bu, y = Cx + Du.
\]

Taking the Laplace transform from such a system (assuming initial values are all zero) will be:

\[
sX(s) = AX(s) + BU(s), Y(s) = CX(s) + DU(s)
\]

Elimination of \( X(s) \) gives:

\[ Y(s) = (C(sI - A)^{-1}B + D)U(s) \]
This of course should look familiar with the transfer functions previously found in this chapter at equation 9.

**Modern Control Engineering**

As discussed in the Classical Control Part, Modern Control is in the state space perspective which enables us to analyse MIMO (Multiple Input Multiple Output) systems and use the power of linear algebra to reveal system properties such as stability etc.

An important aspect of linear control systems is reachability, this has been mentioned earlier, now we will dive deeper on this topic.

One of the fundamental properties of a control system is what set of points in the state space can be reached through the choice of a control input. It turns out that the property of reachability is also fundamental in understanding the extent to which feedback can be used to design the dynamics of a system.

First, just consider the equation for state evolution and disregard the output equation:

\[
d\mathbf{x}/dt = A\mathbf{x} + B\mathbf{u}
\]

where \( \mathbf{x} \in \mathbb{R}^n \), \( \mathbf{u} \in \mathbb{R} \), \( A \) is an \( n \times n \) matrix and \( B \) a column vector. A fundamental question is whether it is possible to find control signals so that any point in the state space can be reached through some choice of input. To study this, we define the reachable set \( R(x_0, \leq T) \) as the set of all points \( \mathbf{x}_f \) such that there exists an input \( u(t) \), \( 0 \leq t \leq T \) that steers the system from \( x(0) = x_0 \) to \( x(T) = x_f \), as illustrated in Figure 9.

Figure 9: The reachable set for a control system. The set shown in (a) is the set of points reachable from \( x_0 \) in time less than \( T \). The phase portrait in (b) shows the dynamics for a double integrator, with the natural dynamics drawn as horizontal arrows and the control inputs drawn as vertical arrows. The set of achievable equilibrium points is the x axis. By setting the control inputs as a function of the state, it is possible to steer the system to the origin, as shown on the sample path.

The definition of reachability addresses whether it is possible to reach all points in the state space in a transient fashion. In many applications, the set of points that we are most interested in reaching is the set of equilibrium points of the system (since we can remain at those points once we get there). The set of all possible equilibria for constant controls is given by

\[
\varepsilon = \{ x_e : Ax_e + Bu_e = 0 \text{ for some } u_e \in \mathbb{R} \}
\]
Which means the system is at rest at an equilibrium point, no state dynamics since \( \frac{dx}{dt} = Ax_e + Bu_e = 0 \).

So how to test for reachability? When the initial state is zero, the response of the system to an input \( u(t) \) is given by:

\[
x(t) = \int_0^T e^{A(T-\tau)} Bu(t) \delta \tau
\]

As has been shown earlier in equation 8, the matrix exponential can be written as a sum. Applying that to the above equation will result in:

\[
x(t) = \lim_{N \to \infty} \int_0^T \sum_{k=0}^N \frac{1}{k!} [A(T-\tau)]^k Bu(\tau) \delta \tau
\]

When reordering \( x_N \) in the above formula:

\[
x_N = \sum_{k=0}^N A^kB \int_0^T \frac{(T-\tau)^k}{k!} u(\tau) \delta \tau
\]

Hence \( x_N \) is a linear combination of the columns of \( B, AB, \ldots, A^NB \). Since each \( A^kB, A^{n+1}B, \ldots, A^NB \) is a linear combination of the matrices \( B, AB, \ldots, A^{n-1}B \), we conclude that \( x_N \) is a linear combination of the columns of \( B, AB, \ldots, A^{n-1}B \).

This means \( x_N \in R(\langle B AB \ldots A^{n-1}B \rangle) \) and with \( N \to \infty \) that even means:

\[
x(T) \in R(\langle B AB \ldots A^{n-1}B \rangle)
\]

The matrix \( [B AB \ldots A^{n-1}B] \) is called the Kalman matrix or controllability matrix (reachability and controllability mean the same). This means that any state \( x(T) \) that can be reached from zero is contained in the range space of the Kalman matrix.

If a system is fully reachable and all the states \( \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix} \) are measured (one sensor for every state variable for example), then state feedback is a possible control scheme.

Figure 10: A feedback control system with state feedback. The controller uses the system state \( x \) and the reference input \( r \) to command the process through its input \( u \). We model disturbances via the additive input \( d \).

Figure 10 is a diagram of a typical control system using state feedback. The full system consists of the process dynamics, which we take to be linear, the
controller elements $K$ and $k_r$, the reference input (or command signal) $r$ and process disturbances $d$. The goal of the feedback controller is to regulate the output of the system $y$ such that it tracks the reference input in the presence of disturbances and also uncertainty in the process dynamics.

An important element of the control design is the performance specification. The simplest performance specification is that of stability: in the absence of any disturbances, we would like the equilibrium point of the system to be asymptotically stable. More sophisticated performance specifications typically involve giving desired properties of the step or frequency response of the system, such as specifying the desired rise time, overshoot and settling time of the step response. Finally, we are often concerned with the disturbance attenuation properties of the system: to what extent can we experience disturbance inputs $d$ and still hold the output $y$ near the desired value?

Consider a system described by the linear differential equation:

$$\frac{dx}{dt} = Ax + Bu, \quad y = Cx + Du$$

where we have ignored the disturbance signal $d$ for now. Our goal is to drive the output $y$ to a given reference value $r$ and hold it there. Since the state at time $t$ contains all the information necessary to predict the future behaviour of the system, the most general time-invariant control law is a function of the state and the reference input:

$$u = -Kx + k_r r$$

where $r$ is the reference value, assumed for now to be a constant. This control law corresponds to the structure shown in figure 10. The negative sign is a convention to indicate that negative feedback is the normal situation. The closed loop system obtained when the feedback $u = -Kx + k_r r$ is applied to the system $\frac{dx}{dt} = Ax + Bu, \quad y = Cx + Du$ is given by:

$$\frac{dx}{dt} = (A - BK)x + Bk_r r, \quad y = Cx + Du$$

Now can be seen from the equation above that the eigenvalues (or the system modes) can actually be affected by the feedback $K$, since the characteristic polynomial is given by $\text{det}(sI - (A - BF))$.

The system was considered fully controllable, so all system modes / eigenvalues can be freely assigned, allowing full control of system dynamics. This is called eigenvalue assignment or better known under pole placement and besides just stabilising an unstable (but controllable) system, it can also manipulate step response behaviour like rise time, overshoot etc.

Note that $k_r$ does not affect the stability of the system (which is determined by the eigenvalues of $(A - BK)$) but does affect the steady-state solution. In particular, the equilibrium point and steady-state output for the closed loop system are given by:

$$x_e = -(A - BK)^{-1}Bk_r r, \quad y_e = Cx_e + Du_e$$

hence $k_r$ should be chosen such that $y_e = r$ (the desired output value). Since $k_r$ is a scalar, we can easily solve to show that if $D = 0$ (the most common case):

$$k_r = \frac{-1}{C(A - BK)^{-1}B}$$
Notice that $k_r$ is exactly the inverse of the zero frequency gain of the closed loop system, which is the steady state value as shown with the transfer function in the classical control chapter.

Besides the fact that the system can be assigned any desired eigenmode, care must be taken since the further the eigenvalues/eigenmodes are shifted from their original position the higher the gains will be in the feedback $K$. Which can mean mechanical complications in case of a mechanical feedback system or saturation in case of electrical feedback systems (saturation occurs when a higher voltage is requested than the power supply can deliver).

For many situations, it is highly unrealistic to assume that all the states are measured. States however can be estimated by using a mathematical model and a few measurements. It will be shown that computation of the states can be carried out by a dynamical system called an observer.

Again consider a system described by the linear differential equation:

\[
\frac{dx}{dt} = Ax + Bu, y = Cx + Du
\]

Figure 11: Block diagram of the observer. The observer takes the signals $y$ and $u$ as inputs and produces an estimate $\hat{x}$. Notice that the observer contains a copy of the process model that is driven by $y - \hat{y}$ through the observer gain $L$.

We wish to estimate the state of the system from its inputs and outputs, as illustrated in figure 11. In some situations we will assume that there is only one measured signal, i.e., that the signal $y$ is a scalar and that $C$ is a (row) vector. This signal may be corrupted by noise $n$, but for the calculations below the noise is ignored. We write $\hat{x}$ for the state estimate given by the observer.

The problem of observability is one that has many important applications, even outside feedback systems. If a system is observable, then there are no hidden dynamics inside it; we can understand everything that is going on through observation (over time) of the inputs and outputs. As we shall see, the problem of observability is of significant practical interest because it will determine if a set of sensors is sufficient for controlling a system. Sensors combined with a mathematical model can also be viewed as a virtual sensor that gives information about variables that are not measured directly. The process of reconciling signals from many sensors with mathematical models is also called sensor fusion.
Now consider the equation for state evolution and this time disregard the input equation:

\[
\frac{dx}{dt} = Ax, \quad y = Cx
\]

We wish to understand when it is possible to determine the state from observations of the output. The output itself gives the projection of the state on vectors that are rows of the matrix C. The observability problem can immediately be solved if the matrix C is invertible. If the matrix is not invertible, we can take derivatives of the output to obtain:

\[
\frac{dy}{dt} = C \frac{dx}{dt} = CAx
\]

From the derivative of the output we thus get the projection of the state on vectors that are rows of the matrix CA. Proceeding in this way, we get

\[
\begin{pmatrix}
y \\
y' \\
y'' \\
\vdots \\
y^{n-1}
\end{pmatrix} =
\begin{pmatrix}
C \\
CA \\
CA^2 \\
\vdots \\
CA^{n-1}
\end{pmatrix}x
\]

We thus find that the state can be determined if the matrix:

\[
W =
\begin{pmatrix}
C \\
CA \\
CA^2 \\
\vdots \\
CA^{n-1}
\end{pmatrix}
\]

has n independent rows. This matrix is called the observability matrix. The calculation can easily be extended to systems with inputs. The state is then given by a linear combination of inputs and outputs and their higher derivatives. The observability criterion is the same. In practice, differentiation of the output can give large errors when there is measurement noise, and therefore the method sketched above is not particularly practical. Other methods of getting the observability matrix out of input/output data will be covered in the chapter on Identification.

Having defined the concept of observability, we now return to the question of how to construct an observer for a system. We will look for observers that can be represented as a linear dynamical system that takes the inputs and outputs of the system we are observing and produces an estimate of the systems state. That is, we wish to construct a dynamical system of the form:

\[
\frac{d\hat{x}}{dt} = F\hat{x} + Gu + Hy
\]

Where \(u\) and \(y\) are the input and output of the original system and \(\hat{x}\) is an estimate of the state with the property that \(\hat{x}(t) \rightarrow x(t)\) as \(t \rightarrow \infty\).

Consider the following system (for simplicity, no feed through term or noise is added):

\[
\frac{dx}{dt} = Ax + Bu, \quad y = Cx
\]

We can attempt to determine the state by building the following observer:

\[
\frac{d\hat{x}}{dt} = A\hat{x} + Bu + L(y - \hat{y})
\]
To find the properties of this estimate, introduce the estimation error $\tilde{x} = x - \hat{x}$. Applying this to the above equations will result in:

$$\frac{d\tilde{x}}{dt} = (A - LC)\tilde{x}$$

If the matrix $L$ can be chosen in such a way that the matrix $(A - LC)$ has eigenvalues with negative real parts, the error $\tilde{x}$ will go to zero and the estimated states from the observer converge towards the actual states of the original system. The convergence rate is determined by an appropriate selection of the eigenvalues.

Notice the similarity between the problems of finding a state feedback and finding the observer. State feedback design by eigenvalue assignment is equivalent to finding a matrix $K$ so that $(A - BK)$ has the given eigenvalues. Designing an observer with prescribed eigenvalues is equivalent to finding a matrix $L$ so that $(A - LC)$ has the given eigenvalues. Since the eigenvalues of a matrix and its transpose are the same we can establish the following equivalences:

$$A \leftrightarrow A^T, \quad B \leftrightarrow C^T, \quad K \leftrightarrow L^T$$

This is called duality: The observer design problem is the dual of the state feedback design problem.

Figure 12: Block diagram of an observer-based control system. The observer uses the measured output $y$ and the input $u$ to construct an estimate of the state. This estimate is used by a state feedback controller to generate the corrective input. The controller consists of the observer and the state feedback.

We wish to design a feedback controller for the system where only the output is measured (states are not directly measured). As with the state feedback controller, we will assume that $u$ and $y$ are scalars (SISO system). With an observer that can generate estimates of the state $\hat{x}$ based on inputs and outputs. With the estimated states available from the observer, it is now possible to design
a state feedback system using the estimated states instead of the actual states (which ideally are the same). A possible feedback scheme would be:

\[ u = -K\hat{x} + kr \]

where \( \hat{x} \) is the output vector from the observer containing the estimated states. The observer being:

\[ \frac{d\hat{x}}{dt} = A\hat{x} + Bu + L(y - \hat{y}) \]

making again the substitution \( \tilde{x} = x - \hat{x} \) will make the observer system dynamics:

\[ \frac{d\tilde{x}}{dt} = (A - LC)\tilde{x} \]

substituting the feedback input into the process dynamics will result in:

\[ \frac{dx}{dt} = Ax + Bu = Ax - BK\hat{x} + Bkr = Ax - BK(x - \hat{x}) + Bkr \]

So the closed loop system dynamics are:

\[ \frac{d}{dt} \begin{pmatrix} x \\ \tilde{x} \end{pmatrix} = \begin{pmatrix} A - BK & BK \\ 0 & A - LC \end{pmatrix} \begin{pmatrix} x \\ \tilde{x} \end{pmatrix} + \begin{pmatrix} Bkr \\ 0 \end{pmatrix} r \]

Notice that the state \( \tilde{x} \), representing the observer error, is not affected by the reference signal \( r \). This is desirable since we do not want the reference signal to generate observer errors.

Since the dynamics matrix is block diagonal, we find that the characteristic polynomial of the closed loop system is:

\[ \lambda(s) = \det(sI - A + BK)\det(sI - A + LC) \]

The controller has a strong intuitive appeal: it can be thought of as being composed of two parts, one state feedback and one observer. The dynamics of the controller are generated by the observer. The feedback gain \( K \) can be computed as if all state variables can be measured, and it depends on only \( A \) and \( B \). The observer gain \( L \) depends on only \( A \) and \( C \). The property that the eigenvalue assignment for output feedback can be separated into an eigenvalue assignment for a state feedback and an observer is called the separation principle.

The block diagram 12 shows that the controller contains a dynamical model of the plant. This is called the internal model principle: the controller contains a model of the process being controlled.

Besides applying only feedback as a control scheme, feedforward could be used as well as shown in figure 13. Feedforward combined with feedback is considered to be a more sophisticated way of control is shown by the block diagram in 13, where the controller consists of three parts: an observer that computes estimates of the states based on a model and measured process inputs and outputs, a state feedback, and a trajectory generator that generates the desired behaviour of all states \( x_d \) and a feedforward signal \( u_{ff} \). Under the ideal conditions of no disturbances and no modelling errors the signal \( u_{ff} \) generates the desired behaviour \( x_d \) when applied to the process. The signal \( x_d \) can be generated by a system that gives
Figure 13: Block diagram of a controller based on a structure with two degrees of freedom which combines feedback and feedforward. The controller consists of a trajectory generator, state feedback and an observer. The trajectory generation subsystem computes a feedforward command $u_{ff}$ along with the desired state $x_d$. The state feedback controller uses the estimated state and desired state to compute a corrective input $u_{fb}$.

To generate the signal $u_{ff}$, we must also have a model of the inverse of the process dynamics.

To get some insight into the behaviour of the system, we assume that there are no disturbances and that the system is in equilibrium with a constant reference signal and with the observer state $\hat{x}$ equal to the process state $x$. When the reference signal is changed, the signals $u_{ff}$ and $x_d$ will change. The observer tracks the state perfectly because the initial state was correct. The estimated state $\hat{x}$ is thus equal to the desired state $x_d$, and the feedback signal $u_{fb} = K(x_d\hat{x})$ will also be zero. All action is thus created by the signals from the trajectory generator. If there are some disturbances or some modelling errors, the feedback signal will attempt to correct the situation.

This controller is said to have two degrees of freedom because the responses to command signals and disturbances are decoupled. Disturbance responses are governed by the observer and the state feedback, while the response to command signals is governed by the trajectory generator (feedforward).

**Robust Control Engineering**

When can we show that the stability of a system is robust with respect to process variations? This is an important question since the potential for instability is one of the main drawbacks of feedback. Hence we want to ensure that even if we have small inaccuracies in our model, we can still guarantee stability and performance.

There must be some margins of stability that describe how stable the system is and its robustness to perturbations. There are many ways to express this, but one of the most common is the use of gain and phase margins, inspired by Nyquist's stability criterion. The key idea is that it is easy to plot the loop transfer function $L(s)$.

Nyquist's idea was to investigate conditions under which oscillations can occur in a feedback loop. To study this, we introduce the loop transfer function $L(s) = P(s)C(s)$, which is the transfer function obtained by breaking the feedback loop, as shown in Figure 14. The loop transfer function is simply the
The stability of the feedback system can be determined by tracing signals around the loop. Letting $L = PC$ represent the loop transfer function, we break the loop in (b) and ask whether a signal injected at the point A has the same magnitude and phase when it reaches point B.

An increase in controller gain simply expands the Nyquist plot radially. An increase in the phase of the controller twists the Nyquist plot. Hence from the Nyquist plot we can easily pick off the amount of gain or phase that can be added without causing the system to become unstable. See figure 15.

The gain margin $g_m$ of a system is defined as the smallest amount that the open loop gain can be increased before the closed loop system goes unstable. For a system whose phase decreases monotonically as a function of frequency starting at 0 deg, the gain margin can be computed based on the smallest frequency where the phase of the loop transfer function $L(s)$ is 180 deg. Let $\omega_{pc}$ represent this frequency, called the phase crossover frequency. Then the gain margin for the system is given by:
Similarly, the phase margin is the amount of phase lag required to reach the stability limit. Let $\omega_{gc}$ be the gain crossover frequency, the smallest frequency where the loop transfer function $L(s)$ has unit magnitude. Then for a system with monotonically decreasing gain, the phase margin is given by:

$$\phi_m = \pi + \arg L(i\omega_{gc})$$

These margins have simple geometric interpretations on the Nyquist diagram of the loop transfer function, as shown in Figure 15, where we have plotted the portion of the curve corresponding to $\omega > 0$. The gain margin is given by the inverse of the distance to the nearest point between 1 and 0 where the loop transfer function crosses the negative real axis. The phase margin is given by the smallest angle on the unit circle between 1 and the loop transfer function. When the gain or phase is monotonic, this geometric interpretation agrees with the formulas above.

A drawback with gain and phase margins is that it is necessary to give both of them in order to guarantee that the Nyquist curve is not close to the critical point. An alternative way to express margins is by a single number, the stability margin $s_m$, which is the shortest distance from the Nyquist curve to the critical point. This number is related to disturbance attenuation.

$$s_m = \frac{1}{\sup_{\omega} |1 + L(i\omega)|}$$

Besides just using the loop transfer function to design the behavior of the closed loop system, there is also the possibility to investigate more transfer functions which also describe closed loop behaviour. As is already shown in the chapter on classical control: transfer function can be deceptive when pole zero cancellations occur. A way to avoid this is by using the so called sensitivity functions. Consider the system in figure 16.

Figure 16: Block diagram of a basic feedback loop with two degrees of freedom. The controller has a feedback block C and a feedforward block F. The external signals are the reference signal $r$, the load disturbance $d$ and the measurement noise $n$. The process output is $\eta$, and the control signal is $u$.

The process output $\eta$ is the real variable that we want to control. Control is based on the measured signal $y$, where the measurements are corrupted by measurement noise $n$. The process is influenced by the controller via the control
variable \( u \). The process is thus a system with three inputs - the control variable \( u \), the load disturbance \( d \) and the measurement noise \( n \) - and one output - the measured signal \( y \). The controller is a system with two inputs and one output. The inputs are the measured signal \( y \) and the reference signal \( r \), and the output is the control signal \( u \). Note that the control signal \( u \) is an input to the process and the output of the controller, and that the measured signal \( y \) is the output of the process and an input to the controller.

Writing down every possible transfer function in the closed loop system will result in the following matrix:

\[
\begin{pmatrix}
y \\
v \\
u \\
e \\
\end{pmatrix} =
\begin{pmatrix}
\frac{PCF}{1+PC} & \frac{P}{1+PC} & \frac{1}{1+PC} \\
\frac{PCF}{1+PC} & \frac{P}{1+PC} & \frac{1}{1+PC} \\
\frac{CF}{1+PC} & \frac{-C}{1+PC} & \frac{-1}{1+PC} \\
\frac{F}{1+PC} & \frac{-P}{1+PC} & \frac{-1}{1+PC} \\
\end{pmatrix}
\begin{pmatrix}
r \\
r \\
n \\
\end{pmatrix}
\]

When the feedforward block is a gain of 1 (is discarded, like in most cases) then can be observed that several transfer functions are the same and that all relations are given by the following set of 4 transfer functions, which are commonly referred to as the Gang of Four:

\[
T = \frac{PC}{1+PC}, \text{ the complementary sensitivity function} \\
PS = \frac{P}{1+PC}, \text{ the load sensitivity function} \\
CS = \frac{C}{1+PC}, \text{ the noise sensitivity function} \\
S = \frac{1}{1+PC}, \text{ the sensitivity function}
\]

Notice that the previously mentioned equation for the stability margin is actually the maximum value of the sensitivity function. The load sensitivity function is also called the input sensitivity function and the noise sensitivity function is also known as the output sensitivity function.

State space design methods, like many methods developed for control system design, do not explicitly take robustness into account. In such cases it is essential to always investigate the robustness because there are seemingly reasonable designs that give controllers with poor robustness.

The closed loop poles can be assigned to arbitrary locations if the system is observable and reachable, choosing fast stable poles (far into the left half plane) will result in fast controller action. However, if we want to have a robust closed loop system, the poles and zeros of the process impose severe restrictions on the location of the closed loop poles.

As seen before: sensitivity functions are closely related to stability margins. An example will be used to show the effect of slow zero’s and fast poles of the process on the sensitivity functions. Consider the following complementary sensitivity function:

The maximum sensitivities are \( Ms = 13 \) and \( Mt = 12 \) for the original closed loop system (solid line in figure 17), indicating that the system has poor robustness. The peak of the complementary sensitivity function can be avoided by assigning a closed loop pole close to the slow process zero. The controller gives the maximum sensitivities \( Ms = 1.34 \) and \( Mt = 1.41 \), which give much better robustness. Notice that the controller has a pole at \( s = 2 \) that cancels the slow process zero.
Figure 17: Sensitivity functions from a closed loop system with a process zero at \( s = -2 \), a controller zero at \( s = 3.5 \) and in one case two closed loop poles far away from the zero’s (solid line) and in the other case one of the closed loop poles cancelling the stable process zero \( s = -2 \) (dashed line). Complementary sensitivity function on the left and sensitivity function on the right.

Now consider the following sensitivity function:

Figure 18: Gain curves for Bode plots of the sensitivity function \( S \) for designs with 2 closed loop poles and process pole : \( p_1 \) \( p_2 \) \( a \) (two closed loop poles slower than the fast stable process pole, indicated on the left) and \( a \) \( p_1 \) \( p_2 \) (two closed loop poles faster than the fast stable process pole, indicated on the right). The solid lines are the true sensitivities, and the dashed lines are the asymptotes.

In figure 18 clearly can be seen that at least one of the closed loop poles should be close to or cancel the fast stable process pole to keep the peak in the sensitivity function low.

To avoid large values of the complementary sensitivity function we find that the closed loop system should therefore have poles close to or equal to the slow stable zeros. This means that slow stable zeros should be cancelled by controller poles when possible. Since unstable zeros cannot be cancelled (gives unreliable results on small process variations as discussed in the chapter on classical control), the presence of slow unstable zeros means that achievable gain crossover frequency must be smaller than the slowest unstable process zero.

To avoid large peaks in the sensitivity the closed loop system should therefore have poles that match the fast process poles. This means that the controller should cancel the fast process poles by controller zeros. Since unstable modes cannot be cancelled, the presence of a fast unstable pole implies that the gain crossover frequency must be sufficiently large.

To summarize, we obtain the following simple rule for choosing closed loop poles: slow stable process zeros should be matched by slow closed loop poles, and fast stable process poles should be matched by fast closed loop poles. Slow
unstable process zeros and fast unstable process poles impose severe limitations.

There a Robust Control Design based on the sensitivity functions, which is called $H_\infty$. The basic idea is simple, but the details are complicated and will not be included. The $H_\infty$ controller model looks like this:

$$z = \begin{pmatrix} y \\ -u \end{pmatrix} = \begin{pmatrix} \frac{1}{1+PC} & \frac{P}{1+PC} \\ \frac{1}{1+PC} & \frac{P}{1+PC} \end{pmatrix} \begin{pmatrix} n \\ d \end{pmatrix} = H(P,C) \begin{pmatrix} n \\ d \end{pmatrix}$$

$z$ is chosen as the generalized error: $z = (\eta,v)$, where $\eta$ is the process output and $v$ is the part of the load disturbance that is not compensated by the controller. The best controller can then be found by iterating on $\gamma$. This method will guarantee that the sensitivity levels of all four transfer functions will remain under the set variable $\gamma$, hence giving always very good robustness properties. Not meaning that it will be the fastest controller.

**Optimal Control Engineering**

As an alternative to selecting the closed loop eigenvalue locations to accomplish a certain objective, the gains for a state feedback controller can instead be chosen is by attempting to optimize a cost function. This can be particularly useful in helping balance the performance of the system with the magnitude of the inputs required to achieve that level of performance.

The infinite horizon, linear quadratic regulator (LQR) problem is one of the most common optimal control problems. Given a multi-input linear system of the following form:

$$\frac{dx}{dt} = Ax + Bu$$

we attempt to minimize the quadratic cost function:

$$J = \int_0^\infty (x^TQ_xx + u^TQ_uu)dt$$

where $Q_x0$ and $Q_u > 0$ are symmetric, positive (semi-) definite matrices of the appropriate dimensions. This cost function represents a trade-off between the distance of the state from the origin and the cost of the control input. By choosing the matrices $Q_x$ and $Q_u$, we can balance the rate of convergence of the solutions with the cost of the control.

The solution to the LQR problem is given by a linear control law of the form:

$$u = -Q_u^{-1}B^TPx$$

where $P$ is a positive definite, symmetric matrix that satisfies the equation:

$$PA + A^TP - PBQ_u^{-1}B^TP + Q_x = 0$$

(12)

Equation 12 is called the algebraic Riccati equation. One of the key questions in LQR design is how to choose the weights $Q_x$ and $Q_u$. To guarantee that a solution exists, usually the following conditions are chosen: $Q_x > 0$ and $Q_u > 0$.

To choose specific values for the cost function weights $Q_x$ and $Q_u$, we must use our knowledge of the system we are trying to control. A choice often made is to use diagonal weights (only using decoupled weights):
\[ Q_x = \begin{pmatrix} q_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & q_n \end{pmatrix}, \quad Q_u = \begin{pmatrix} p_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p_n \end{pmatrix} \]

For this choice of \( Q_x \) and \( Q_u \), the individual diagonal elements describe how much each state and input (squared) should contribute to the overall cost. Hence, we can take states that should remain small and attach higher weight values to them. Similarly, we can penalize an input versus the states and other inputs through choice of the corresponding input weight \( p \).

**Non linear Control Engineering**

Non linear control falls out of the scope of this document, most of the time linearising around a non linear equilibrium point will still give good overall results without actually using a non linear controller. Considering that this document focuses on embedded controller, it is safe to assume that not a lot of computational power will be available, so solving non linear equations will probably not be very time efficient for a small microcontroller.

What will be covered in this section is feedback linearisation, which uses a non linear feedback mechanism to linearise the non linearities in the process. Block diagram is shown in figure 19.

---

Consider the non linear system:

\[
\begin{align*}
\dot{x}_1 &= a \sin(x_2) \\
\dot{x}_2 &= -x_1^2 + u
\end{align*}
\]

First a change of state is introduced to transform the system equation into the standard form:

\[
\begin{align*}
\dot{z}_1 &= x_1 \\
\dot{z}_2 &= a \sin(x_2)
\end{align*}
\]

Which will transform the system into the form:

\[
\begin{align*}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= a(-x_1^2 + u) \cos(x_2)
\end{align*}
\]

Since the non linearities are now only in the second equation, they can be linearised by choosing the following non linear input:
\[ u = x_1^2 + \frac{1}{a \cos(2x_2)} v \]

Hence, under the condition that \( a \cos(2z_1) \neq 0 \), for \( a \neq 0 \) and \(-\pi/2 < x_2 < \pi/2\), the non linear system reduces to the linear system:

\[
\begin{align*}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= v
\end{align*}
\]

which are easily controlled by the linear control \( v = -k_0 z_1 - k_1 z_2 \) via desired selections of the feedback gains.

### 0.1.3 Conclusion

What is very often seen when control of a process is needed, is the use of PID controllers, which will basically throw away all known process information and treat the process as a black box with three control parameters: \( k_p \) (proportional), \( k_i \) (Integral), \( k_d \) (derivative):

\[ u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{\delta e(t)}{\delta t} \]

This of course is a waste of all the known process information. Best would be to use all information available on the process and build a model out of it which can predict the output response of the process to any input.

With an accurate model, as can be acquired with the techniques explained in the chapter on model identification, the combination of feedback and feedforward would be the best choice. The accurate model will do the feedforward part and any errors like disturbances will be cancelled using a feedback mechanism.

What will also be interesting to explore is if sensors can be left out, specifically the pressure sensor between the proportional valve and the gas pump. With the design of an observer, it might be possible to leave this sensor out. This have to be validated with experiments.

So far we only discussed control systems which rely on the following continuous time equations: \( \frac{dx}{dt} = Ax + Bu, \ y = Cx + Du, \ x(0) = 0 \) where \( x \) represents the system state(s) and \( A, B, C, D \) are matrices which represent the linear relations between input(\( u \)) and output(\( y \)) data. Input data being the sensors (for example: gas flow sensor, gas pressure sensors) and output data being the actuators (for example: flow controller, gas pump, proportional valve, gas divider). An important practicality is that micro controllers do not work in the continuous time domain, but in the discrete time domain. Therefore can only work with difference equations, the \( z \) domain.

As an example:

\[
\begin{align*}
\frac{dx}{dt} &= A\dot{x} + Bu + L(y - \dot{y}) \\
u &= -K\dot{x} + k_r r
\end{align*}
\]

This equation from an output observer can be converted to a difference equation as follows:

\[
\frac{dx}{dt} \approx \frac{\dot{x}(t_{k+1}) - \dot{x}(t_k)}{\Delta t} = A\dot{x}(t_k) + Bu(t_k) + L(y(t_k) - \dot{y}(t_k))
\]

\[ u(k + 1) = -K\dot{x}(t_k) + k_r r \]

where \( t_k \) are the sampling instants and \( \Delta t = t_{k+1} - t_k \) is the sampling period. Rewriting the equation to isolate \( \dot{x}(t_{k+1}) \), we get the difference equation:
Figure 20: Components of a computer-controlled system. The controller consists of analog-to-digital (A/D) and digital-to-analog (D/A) converters, as well as a computer that implements the control algorithm. A system clock controls the operation of the controller, synchronizing the A/D, D/A and computing processes. The operator input is also fed to the computer as an external input.

\[
\dot{x}(t_{k+1}) = \dot{x}(t_k) + \Delta t (A\dot{x}(t_k) + Bu(t_k) + L(y(t_k) - \hat{y}(t_k))), \quad u(k + 1) = -K\dot{x}(t_k) + k_r r
\]

An example block diagram of how a computer controller could look like is shown in figure 20.

0.2 Identification of the Process Model

0.2.1 Introduction

In the chapter in linear control systems, a lot of different designs were discussed. The two most important ones (the two most often used) being: transfer function system and state space system. However an important question remains unanswered: How to get the model parameters? How to acquire the 4 matrices (A,B,C and D) needed by modern linear control systems or how to get the transfer function parameters in order to create an as accurate model of the process as possible?

These A,B,C,D matrices can be deducted from physic laws, which can be quite challenging since relations between sensors/actuators do not have to be obvious. In this chapter two other techniques will be discussed: AR(MA)X identification and subspace identification. AR(MA)X will try and fit a parametrised transfer function to the input and output data, subspace identification will use linear algebra techniques to fit the A,B,C and D matrices to the input and output data.
Figure 21: The picture above shows the ‘system identification cycle’, which will be repeated until the model reaches an acceptable level of accuracy. Normally 75% of the test data is used to identify the model and the remaining 25% is used to verify the model.

Although parameter identification is the non trivial core part of system identification, many choices need to be made before arriving at an adequate data set and a suitable model parametrisation. The choices at the start of a system-identification task, such as the selection of the digital data-acquisition infrastructure, the sampling rate, and the type of anti-aliasing filters, comprise the experiment design. Designing a system-identification experiment requires a balanced integration of engineering intuition, knowledge about systems and control theory, and domain-specific knowledge of the system to be studied.

The data may need to be polished and pre-filtered to remove deficiencies (outliers, noise, trends) and to accentuate a certain frequency band of interest. This is indicated by the data pre-processing step.

Having acquired a polished data set, the next challenge to be tackled is the actual estimation of the model (parameters). The parameter estimation step has to be preceded by the crucial choice of model structure (choosing the right order of the model). Subspace methods do simply this step by indicating the dominant modes and thus an indication of the model order (amount of states / modes to model).

Next step is to fit a model with the chosen order to the polished dataset. This can be either an AR(MA)X model or a state space model through subspace identification.

Having estimated a model, the quality of the model needs to be addressed. Model validation involves evaluation of the quality of the model and deciding whether it is suitable for the application for which it is intended. This illustrates the basic guidance principle in system identification: to design the experiment and to (pre-)process the data in such a way that these experimental conditions match as closely as possible the real-life circumstances under which the model will be used. For example, when the goal is to design a feedback-control system, it may be recommendable to develop a model based on closed-loop identification experiments. The latter circumstances may stipulate additional requirements on the parameter-estimation methods. However, coping with these additional
requirements often pays off in the actual controller design.

For system identification numerous input signals can be used, such as the elementary inputs: the step input, doublet, short impulse or a harmonic signal. Also used for system identification are signals like: staircase, gaussian white noise, frequency sweep and much more. Choosing an appropriate input signal highly depends on the process itself. For a selection of most used input signals, see figure 22.

![Input Signals](image)

Figure 22: A selection of input signals used to achieve useful data from experiments.

To be able to estimate a model from measured input and output data, the data should contain enough information. In the extreme case that the input sequence $u(k) = 0$ for all $k \in \mathbb{Z}$, no information about the transfer from $u(k)$ to $y(k)$ can be retrieved. Therefore, the input should be different from zero in some sense so that we will be able to identify particular transfer functions. This property of the input in relationship to system identification is generally indicated by the notion of persistency of excitation. When ordering all the input data in a specific way, persistency of excitation can be tested by rank condition:

$$U_{0,n,N} = \begin{pmatrix}
    u(0) & u(1) & \cdots & u(N-1) \\
    u(1) & u(2) & \cdots & u(N) \\
    \vdots & \vdots & \ddots & \vdots \\
    u(n-1) & u(n) & \cdots & u(N+n-2)
\end{pmatrix}$$

The way of ordering data this way is called a Hankel matrix. Notice as well that the formula is in discrete time domain now and not anymore in the continuous time domain. The input data is only persistent of excitation when there exists an integer $N$ for which the matrix $U_{0,n,N}$ has full rank $n$. This can only happen when the data is ‘rich’ enough with information.

Polishing up the data can be done in several ways, for example decimation can be used when the sample frequency is too high: this will typically be the case when a lot of high frequency disturbance shows up in the data, above the frequency band of interest. Detrending the data can be done to remove offsets in the data (to get the rest point / equilibrium point at zero instead of some


offset value), this can be done using these simple formula's:

\[
\hat{y} = \frac{1}{N} \sum_{k=1}^{N} y(k), \quad \hat{u} = \frac{1}{N} \sum_{k=1}^{N} u(k)
\]

Besides decimation or detrending, one can also choose to use a regular low or high filter on the input and output data of course. This is especially helpful when frequency band of the disturbances is known.

### 0.2.2 Existing Identification methods

Two main methods will be discussed in this chapter: transfer function identification for AR(MA)X models and subspace identification for state space models.

#### identification for an AR(MA)X model

The ARMAX model, standing for Auto-Regressive Moving Average with eXogenous input, model structure considers the following specific case of the general input-output description:

\[
y(k) = \frac{b_1 q^{-1} + \cdots + b_n q^{-n}}{1 + a_1 q^{-1} + \cdots + a_n q^{-n}} u(k) + \frac{1 + c_1 q^{-1} + \cdots + c_n q^{-n}}{1 + a_1 q^{-1} + \cdots + a_n q^{-n}} e(k)
\]

where \( e(k) \) is a zero-mean white-noise sequence that is independent from \( u(k) \) and \( a_i, b_i, c_i \) \((i = 1, 2, \cdots, n)\) are real-valued scalars. The \( q \) is the time shift operator, so every time shift corresponds with a sample point in the data.

When \( k_i = c_i - a_i \), then the one step ahead predictor from the ARMAX model is:

\[
\hat{y}(k|k - 1) = \frac{b_1 q^{-1} + \cdots + b_n q^{-n}}{1 + a_1 q^{-1} + \cdots + a_n q^{-n}} u(k) + \frac{k_1 q^{-1} + \cdots + k_n q^{-n}}{1 + a_1 q^{-1} + \cdots + a_n q^{-n}} y(k)
\]

The one step ahead predictor can also be written in state space form and keeping in mind that \( c_i = k_i + a_i \), it will result in:

\[
\dot{x}(k+1) = \begin{pmatrix}
-c_1 & 1 & 0 & \cdots & 0 \\
-c_2 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-c_{n-1} & 0 & 0 & \cdots & 1 \\
-c_n & 0 & 0 & \cdots & 0
\end{pmatrix} \dot{x}(k) + \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix} u(k) + \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_{n-1} \\ k_n \end{pmatrix} y(k)
\]

\[
\hat{y}(k|k - 1) = \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \dot{x}(k|k - 1)
\]

Converting a transfer function to state space is called realisation. This is a topic on its own, but when sticking to the basic parametrisation guidelines it is always possible to convert an ARMAX model like above to state space. State space identification will be covered in the next section.

The Auto-Regressive with eXogeneous input (ARX) model is a special case of the ARMAX model structure constraining the parameters \( c_i = 0 \) for \( i = 1, 2, \cdots, n \). Therefore, the ARX model is given by:

\[
y(k) = \frac{b_1 q^{-1} + \cdots + b_n q^{-n}}{1 + a_1 q^{-1} + \cdots + a_n q^{-n}} u(k) + \frac{1}{1 + a_1 q^{-1} + \cdots + a_n q^{-n}} e(k)
\]

And its one step ahead predictor as:

\[
\hat{y}(k|k - 1) = (b_1 q^{-1} + \cdots + b_n q^{-n}) u(k) + (-a_1 q^{-1} - \cdots - a_n q^{-n}) y(k)
\]
To get the actual parameters $a_0 \cdots a_n, b_0 \cdots b_n, c_0 \cdots c_n$, a minimisation problem needs to be solved. By reorganising the formula’s above it is possible to form a cost function of the modelling error:

$$J_N = \frac{1}{N} \sum_{k=1}^{N} ||y(k) - \hat{y}(k, \theta)||^2$$

With the minimisation problem:

$$\min_{\theta} J_N(\theta)$$

In these formula’s the $\theta$ vector contains all the parameters like $a_0 \cdots a_n, b_0 \cdots b_n$ etc.

There are several ways to solve a minimisation problem, minimisation is a topic on its own, but one algorithm is very often used on linear systems with an overdetermined set of equation of the form $y = Fx$. Consider an ARX model as an example (figure 23), there we can reorganise the equation to make it into the form $y = Fx$:

$$\begin{pmatrix}
y(k) \\
y(k+1) \\
\vdots \\
y(N) 
\end{pmatrix} =
\begin{pmatrix}
a_1 \\
a_n \\
b_1 \\
b_n 
\end{pmatrix}
$$

The equation now are in the linear form $Y_{k,N} = \Phi \theta = [Y_{0,n,N} | U_{0,n,N}] [a_1 \cdots a_n | b_1 \cdots b_n]^T$, which is similar to $y = Fx$ and can be solved using a linear least squares ap-
A few things should be noticed looking at the matrices: The input and output data are both in hankel matrix form and augmented into one matrix named $\Phi$ here. The parameters $a_0 \cdots a_n$ and $b_0 \cdots b_n$ are in one augmented vector $\theta$.

The parameter vector can now be found computing:

$$\hat{\theta} = \Phi^T(\Phi\Phi^T)^{-1}Y_{k,N}$$

The vector $\hat{\theta}$ will contain the estimated parameters which should minimise the cost function $J_N$. Which can be observed from this method is that ARX models suffer from local minima in the cost function as shown in figure 24.

As will be shown, subspace techniques suffer less from this phenomenon. Of course there a lot of minimisation techniques out there like steepest descent etc. which will try to avoid falling into a local minima of the cost function, but this still remains a property that needs attention when designing the model.

**Identification for a state space model**

Subspace identification is based on the fact that, by storing the input and output data in structured block Hankel matrices, it is possible to retrieve certain subspaces that are related to the system matrices of the signal generating state-space model. An examples of such a subspace is the column space of the observability matrix.

Unlike the identification algorithm discussed in previous chapter, in subspace identification there is no need to parametrise the model. Furthermore, the system model is obtained in a non iterative way via the solution of a number of simple linear-algebra problems. The key linear-algebra steps are an RQ
factorization, an SVD (Singular Value Decomposition), and the solution of a linear least-squares problem. This also circumvents having to solve a non-linear optimisation problem.

An important and critical step prior to the design (and use) of subspace identification algorithms is to find an appropriate relationship between the measured data sequences and the matrices that define the model.

The response of an LTI(Linear Time Invariant) discrete system at time instant \( k \) to an initial state \( x(0) \) and an input signal from time 0 to \( k \) can be found from the state equation by recursion:

\[
x(1) = Ax(0) + Bu(0) \\
x(2) = A^2x(1) + ABu(0) + Bu(1) \\
\vdots \\
x(k) = A^kx(1) + A^{k-1}Bu(0) + \cdots + ABu(k-2) + Bu(k-1)
\]

The response from time instant \( k \) to time instant \( k + j \) can also be written as a sum:

\[
x(k + j) = A^kx(j) + \sum_{t=0}^{k-1} A^{k-i-1}Bu(i+j)
\]

This equation should look familiar, only this time it is written as a difference equation. In the chapter on Linear Control Systems it has been introduced in the continuous time domain as the convolution integral:

\[
x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau
\]

The equation for \( y(k+j) \) can be deducted when using the state space \( y = Cx + Du \) and becomes:

\[
x(k + j) = CA^kx(j) + \sum_{t=0}^{k-1} CA^{k-i-1}Bu(i+j) + Du(k+j)
\]

An interesting form appears when separating the system matrices from the input- and state vector:

\[
\begin{pmatrix}
  y(0) \\
  y(1) \\
  \vdots \\
  y(s-1)
\end{pmatrix} =
\begin{pmatrix}
  C \\
  CA \\
  \vdots \\
  CA^{s-1}
\end{pmatrix} x(0) +
\begin{pmatrix}
  D & 0 & 0 & \cdots & 0 \\
  CB & D & 0 & \cdots & 0 \\
  CAB & CB & D & \cdots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  CA^{s-2}B & CA^{s-3}B & CA^{s-4}B & \cdots & D
\end{pmatrix}
\begin{pmatrix}
  u(0) \\
  u(1) \\
  \vdots \\
  u(s-1)
\end{pmatrix}
\]

the matrix \( O_s \) is referred to as the extended observability matrix, the matrix \( T_s \) represents the influence of the input and \( s \) is a finite positive integer which should be bigger than the model order (not known at this point, so \( s \) is often chosen much bigger than the expected model order \( n \)). Again the Hankel matrix form is applied to use all gathered input/output data from experiments:
\[
Y_{0,s,N} = \begin{pmatrix}
y(0) & y(1) & \cdots & y(N-1) \\
y(1) & y(2) & \cdots & y(N) \\
\vdots & \vdots & \ddots & \vdots \\
y(s-1) & y(s) & \cdots & y(N-s+2) 
\end{pmatrix},
\]
\[
X_{0,N} = \begin{pmatrix}
x(0) & x(1) & \cdots & x(N-1) \\
u(0) & u(1) & \cdots & u(N-1) \\
\vdots & \vdots & \ddots & \vdots \\
u(s-1) & u(s) & \cdots & u(N+s-2) 
\end{pmatrix},
\]
\[
U_{0,s,N} = \begin{pmatrix}
y(0) & y(1) & \cdots & y(N-1) \\
y(1) & y(2) & \cdots & y(N) \\
\vdots & \vdots & \ddots & \vdots \\
y(s-1) & y(s) & \cdots & y(N-s+2) 
\end{pmatrix},
\]

\(Y_{0,s,N}\) being the Hankel matrix containing all output samples, \(U_{0,s,N}\) being the Hankel matrix containing all input samples and \(X_{0,N}\) being the Hankel matrix holding all the states. \(N\) represents the amount of samples available, with \(n < s << N\) meaning that \(s\) should be bigger than the expected model order \(n\) and the total amount of samples \(N\) should be way bigger than \(s\). This results in a compact equation:

\[
Y_{0,s,N} = O_s X_{0,N} + T_s U_{0,s,N}
\]

This equation is called the data equation. A very important property of the \(O_s\) matrix is that the rank of this matrix the actual order \(n\) is from the model. This can be shown by looking at the autonomous system (no input influences):

\[
Y_{0,s,N} = O_s X_{0,N},
\]

the state Hankel matrix can also be written as:

\[
X_{0,N} = \begin{pmatrix}
x(0) & Ax(0) & \cdots & A^{N-1}x(0) 
\end{pmatrix}
\]

If the pair \((A, x(0))\) is reachable, then the matrix \(X_{0,N}\) has full row rank \(n\). So the \(O_s\) will therefore also have rank \(n\). This makes subspace techniques a lot more helpful when having to decide on a model order. Getting the rank of the matrix \(O_s\) (or the matrix \(X_{0,N}\) for that matter) is not that easy when there are influences from the input involved.

When reorganising the equation to get the following:

\[
Y_{0,s,N} - T_s U_{0,s,N} = O_s X_{0,N}
\]

Which tells us that the rank of \(Y_{0,s,N} - T_s U_{0,s,N}\) is equal to the rank of \(O_s X_{0,N}\) and thus the rank of \(O_s\) (as seen earlier). Unfortunately \(T_s\) contains the system matrices which are still unknown. It is however possible to get an estimate of \(T_s\) by minimising the following cost function:

\[
\min_{T_s} ||Y_{0,s,N} - T_s U_{0,s,N}||_2^2
\]

\(T_s\) can now be calculated by \(\hat{T}_s = Y_{0,s,N} U_{0,s,N}^T( U_{0,s,N} U_{0,s,N}^T )^{-1}\) and when reorganising this formula a special type of matrix appears:

\[
Y_{0,s,N} - \hat{T}_s U_{0,s,N} = Y_{0,s,N}(I_N - U_{0,s,N}^T U_{0,s,N} U_{0,s,N}^T)^{-1} U_{0,s,N}
\]

\[
\Pi_{-} U_{0,s,N} = I_N - U_{0,s,N}^T U_{0,s,N} U_{0,s,N}^T U_{0,s,N}^T
\]

40
This matrix $\Pi_{\perp U_{0,s,N}}$ is special because it is the orthogonal projection onto the column space of $U_{0,s,N}$. In other words, it cancels out the input influences when multiplied from the left with the input Hankel matrix $U_{0,s,N}\Pi_{\perp U_{0,s,N}} = 0$. This also shows why it is so important for the input matrix to be persistently excited, because otherwise it will not be invertible. Using this property of the $\Pi_{\perp U_{0,s,N}}$ matrix will result in the following simplified equation:

$$Y_{0,s,N}\Pi_{\perp U_{0,s,N}} = O_sX_{0,N}\Pi_{\perp U_{0,s,N}}$$

So it removed the influence of the input leaving only the state response and now the rank can be valued from the $O_s$ matrix by evaluating the rank of $Y_{0,s,N}\Pi_{\perp U_{0,s,N}}$. The proof that the rank of $O_s$ and $Y_{0,s,N}\Pi_{\perp U_{0,s,N}}$ are the same is pretty big, so for more information on the actual proof of this, then check out the book Filtering and Identification [6].

An SVD of $Y_{0,s,N}\Pi_{\perp U_{0,s,N}}$ will reveal the rank of the model:

$$Y_{0,s,N}\Pi_{\perp U_{0,s,N}} = U\Sigma V^T$$

the matrix $\Sigma$ will be a matrix of $s \times s$ (the $s$ chosen earlier on) and it will contain the singular values on the diagonal. Largely depending on experiment and data quality, these singular values will be largely apart or close together, an example is shown in figure 25.

![Figure 25](image)

Figure 25: here an example is shown of a plot of the singular values from $\Sigma$. The absolute values are not important, only the distinction between the values. For example in this graph it clearly shows there are 4 dominant values, so the model order of 4 would be a good choice. Unfortunately the plots are not always this clear, certainly not when there is a lot of noise involved.

A lot of things can actually already be deducted from the singular value plot with a trained eye, like oscillatory modes (pairing of singular values), but model order is the most important one.

Since the rank is the same of matrix $O_s$ and matrix $U$ (achieved from the SVD) there should only be some unknown coordinate transformation between them. This property will be used to get the transformed matrices $C$ and $A$ from the $U$ matrix:
\[ U = O_s T = \begin{pmatrix} C_T \\ C_T A_T \\ \vdots \\ C_T A_T^{s-1} \end{pmatrix} \]

The matrix \( T \) being a coordinate change matrix. So matrix \( C_T \) can be found on the first row of \( U \); the matrix \( A \) can be found by taking \( U \) and starting from row 1 (row 1 to (s-1)) and solving a linear least squares problem with the original \( U \) minus the last row (row 0 to (s-2)):

\[
\begin{pmatrix} C_T \\ C_T A_T \\ \vdots \\ C_T A_T^{s-1} \end{pmatrix} A_T = \begin{pmatrix} C_T \\ C_T A_T \\ \vdots \\ C_T A_T^{s-1} \end{pmatrix} A_T = \min_{A_T} \| U_{0\to(s-2)} - U_{1\to(s-1)} A_T \|_2^2
\]

the solution of this minimisation is already known:

\[ \hat{A}_T = U_{0\to(s-2)} U_{1\to(s-1)}^T (U_{1\to(s-1)} U_{1\to(s-1)}^T)^{-1} \]

The matrices \( B_T \) and \( D_T \), together with the initial state \( x_T(0) = T^1 x(0) \), can be computed by solving a least-squares problem. Given the matrices \( \hat{A}_T \) and \( \hat{C}_T \), the output can now be expressed linearly in the matrices \( B_T \) and \( D_T \):

\[
y(k) = \hat{C}_T \hat{A}_T^k x_T(0) + (\sum_{\tau=0}^{k-1} u(\tau)^T \otimes \hat{C}_T \hat{A}_T^{k-\tau-1}) vec(B_T) + (u(k)^T \otimes I) vec(D_T)
\]

A few mathematical notions are introduced here: the Kronecker product \( \otimes \) is very useful in rewriting matrices. The Kronecker product is defined as:

\[
A \in \mathbb{R}^{m \times n}, \quad B \in \mathbb{R}^{p \times q}, \quad A \otimes B = \begin{pmatrix} a_{11} B & \cdots & a_{1n} B \\ \vdots & \ddots & \vdots \\ a_{m1} B & \cdots & a_{mn} B \end{pmatrix}
\]

making the Matrix \( A \otimes B \) of the size \( mp \times nq \). The operator vec stacks all the columns of a matrix on top of each other to create one big vector:

\[
A \in \mathbb{R}^{m \times n}, \quad vec(A) = \begin{pmatrix} a_{11} \\ \vdots \\ a_{m1} \\ \vdots \\ a_{1n} \\ \vdots \\ a_{mn} \end{pmatrix}
\]

Using the Kronecker product and the vec operator, allows to bring the equation into a least squares minimisation problem:

\[
\Phi(k) = \left( \hat{C}_T \hat{A}_T \sum_{\tau=0}^{k-1} u(\tau)^T \otimes \hat{C}_T \hat{A}_T^{k-\tau-1} \right) u(k)^T \otimes I)
\]
θ = \begin{bmatrix} x_T(0) \\ \text{vec}(B_T) \\ \text{vec}(D_T) \end{bmatrix}

\min_{\theta} ||y(k) - \Phi \theta||^2_2

The solution to this problem has been seen before: \( \hat{\theta} = \Phi^T(\Phi\Phi^T)^{-1}Y_{k,N} \). So subspace methods will derive the system matrices in an unknown coordinate change, but since all the system matrices are in the same coordinate change (all multiplied with some unknown coordinate change matrix T), the results from this system will be the same as the non coordinate changed system. There still very many techniques which can improve efficiency and numeric accuracy of subspace techniques, like RQ factorisation, and so far noise has been ignored.

The model can be improved to account for measurement noise and output noise.

0.2.3 Conclusion

Depending of course on the accuracy needed and the computation power available, one technique might be better than the other. That being said, subspace identification has the nice feature that it has linear properties and will not get stuck in potential local minima.

When using subspace identification to achieve a model of the process which will be very close to the actual process, then it will be much easier for the control system to regulate the process using combined feedforward and feedback construction (two degrees of freedom). So the accurate model acquired with subspace techniques will function as a trajectory generator (feedforward) and the feedback mechanism will cancel out disturbances (if any). This control scheme has been explained at the end of the modern control section, see figure 13.

Figure 26: Distributions of the VAF values for 100 identification experiments on an acoustical duct anti sound process: (left) OE models (like ARMAX), (middle) state-space models obtained from subspace identification, and (right) state-space models obtained from prediction-error optimization. The percentages indicate the mean VAF value.

Figure 26 shows the VAF (Variance Accounted For) values of the different models for repeated identification experiments of the same process. VAF is used often to evaluate the accuracy of the model and is defined as follows:
$$VAF(y(k), \hat{y}(k, \theta)) = \max(0, 1 - \frac{1}{N} \sum_{k=1}^{N} \frac{|y(k) - \hat{y}(k, \theta)|^2}{\sum_{k=1}^{N} |y(k)|^2} \cdot 100\%)$$

The OE models like ARMAX have better VAF values. These models use the estimates of the ARX model as initial estimates. However, the high number of VAF values in the region 10% - 30% indicates that the algorithm gets stuck at local minima. We also see that the combination of subspace identification and prediction-error optimization does yield accurate low-order models. In fact, the models obtained directly by subspace identification are already quite good.

It will be interesting to try both identification algorithms to see if the same differences in result can be found.
Bibliography


