

LECTURE NOTES ON SYSTEM IDENTIFICATION

Dr. Prabir Barooah

August 25, 2008

Disclaimer: This document probably contains typos. For comments on the content and typos, please contact the author at pbarooah@ufl.edu

© Prabir Barooah. Please do not distribute this document without the author's consent.

Contents

1	Parameter identification through least squares	3
1.1	An example	3
1.2	Least squares solution	4
1.2.1	Basic least squares:	4
1.2.2	Geometric interpretation of least squares	6
1.3	ARX model identification by least squares	8
2	Mathematical models of physical systems	10
2.1	ODE models	10
2.1.1	Numerical integration	12
2.1.2	Modeling exercise:	13
2.1.3	Equilibrium point and linearization around it	13
3	Continuous time SISO LTI systems	15
3.1	Laplace transform	15
3.2	Transfer function	18
3.2.1	Impulse response and transfer function	20
3.3	Steady state response to sinusoidal inputs	20
4	Discrete time SISO LTI systems	22
4.1	Z-transforms	23
4.1.1	Properties of Z-transform	24
4.2	Discrete time transfer function	25
4.2.1	Unit pulse response and transfer function	26
4.3	Steady-state response to sinusoidal inputs	26
4.4	Approximating $H_c(s)$ with $H_d(z)$	27
4.4.1	Forward difference approximation	27
4.4.2	Trapezoidal approximation	28
4.4.3	Properties	28
4.5	Difference equation from transfer function	29
4.6	Parametric system identification revisited	30

Chapter 1

Parameter identification through least squares

1.1 An example

: Suppose the sampled outputs $y(k), k = 0, 1, \dots, n, \dots$ of a system are related to the inputs $u[k], k = 0, 1, \dots, n, \dots$ according to the following difference equation:

$$y[k + 1] = ay[k] + bu[k],$$

where the scalar parameters a and b are unknown. We can conduct an experiment and record a sequence of inputs and the resulting outputs. The task is to identify the parameters from N samples of the input $\{u_k\}, k = 1, \dots, N$ and the measured output $\{y_k\}, k = 1, \dots, N$. Suppose y_0 is the measured output at time 0, then:

$$\begin{aligned} y[1] &= ay[0] + bu[0] \\ y[2] &= ay[1] + bu[1] \\ &\dots = \dots \\ y[N] &= ay[N - 1] + bu[N - 1] \end{aligned}$$

stacking them together, we get

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} y_0 & u_0 \\ y_1 & u_1 \\ \dots & \dots \\ y_{N-1} & u_{N-1} \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\boldsymbol{\theta}} \quad (1.1)$$

Note that in this case, \mathbf{y} is an N -vector of data, Φ is a $N \times 2$ matrix, and $\boldsymbol{\theta}$ is a 2-vector of unknown parameters. We will write it as $\mathbf{y} \in \mathbb{R}^N$, $\Phi \in \mathbb{R}^{N \times 2}$ and $\boldsymbol{\theta} \in \mathbb{R}^2$. \mathbb{R}^n is called the *n-dimensional real coordinate space*, since it is the set of all n -vectors whose entries are real numbers. $\mathbb{R}^{m \times n}$ is the set of all $m \times n$ matrices whose entries are real numbers.

At this point, it might be tempting to say that to identify the parameters in the vector $\boldsymbol{\theta}$, all we need to do is construct the vector \mathbf{y} and the matrix Φ from the measured data, and then solve the equation above to determine $\boldsymbol{\theta}$. However, in practice, the measured outputs are not the y_i 's, but \tilde{y}_i 's corrupted by measurement noise, which we denote by \tilde{y}_i :

$$\tilde{y}_i := y_i + \epsilon_i, \quad i = 1, \dots, N,$$

We follow the convention of writing a vector as a column, be default. If x is a m -vector, y is an n -vector, and A is an $n \times m$ matrix, the expressions Ax and $y^T A$ are well defined but not xA and yA . If x and y are two n -vectors, their *scalar product* $x^T y := \sum_{i=1}^n x_i y_i$ is sometimes referred to as $x \cdot y$.

where ϵ_i is an unknown measurement noise. As a result, what we can construct from measured data are not \mathbf{y} and Φ but their noisy counterparts:

$$\tilde{\mathbf{y}} = \begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \dots \\ \tilde{y}_{N-1} \end{bmatrix} \quad \tilde{\Phi} = \begin{bmatrix} \tilde{y}_0 & u_0 \\ \tilde{y}_1 & u_1 \\ \dots & \dots \\ \tilde{y}_{N-1} & u_{N-1} \end{bmatrix}$$

Because of noise, there may not be a $\boldsymbol{\theta}$ that will satisfy

$$\tilde{\mathbf{y}} = \tilde{\Phi}\hat{\boldsymbol{\theta}}.$$

Instead, it is more likely that for every $\hat{\boldsymbol{\theta}}$ we try, there will be a discrepancy between the right hand and left sides of the equation above, which we denote by \mathbf{e} :

$$\mathbf{e} := \tilde{\mathbf{y}} - \tilde{\Phi}\hat{\boldsymbol{\theta}}.$$

In this case, we try to find the so-called *least squares solution* $\hat{\boldsymbol{\theta}}_{LS}$ that minimizes the squared error

$$\|\mathbf{e}\|^2 = \sum_{i=1}^N (\tilde{y}_i - \tilde{\Phi}\hat{\boldsymbol{\theta}})_i^2 = \|\tilde{\mathbf{y}} - \tilde{\Phi}\hat{\boldsymbol{\theta}}\|^2$$

The “norm of an n -vector” $x = [x_1, x_2, \dots, x_n]^T$ usually means its 2-norm, which is defined as $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$.

1.2 Least squares solution

1.2.1 Basic least squares:

The problem above can be thought of as finding the best-fit straight line to a set of data points. Suppose two scalar variables x and y are related by a linear equation of the form

$$y = ax + b \tag{1.2}$$

Suppose we conduct N experiments and get N pairs of data points (x_i, y_i) , for $i = 1, 2, \dots, N$. Due to measurement noise etc., it is unlikely that $y_i = ax_i + b$ is exactly satisfied by the i -th data point. Define

$$e_i := (ax_i + b) - y_i \tag{1.3}$$

as the error between the data y_i and the “prediction” $\hat{a}x_i + \hat{b}$, for every set of values (\hat{a}, \hat{b}) we try. Least squares identification attempts to find values of a and b so that the the sum of squares errors, $\sum_i e_i^2$, is as small as possible.

Problem 1. Find the values for a and b that minimizes the Sum-of-Squares Error (SSE):

$$SSE = \sum_{i=1}^N (ax_i + b - y_i)^2. \quad \square$$

Solution to 1. Let us use calculus and solve

$$\begin{aligned}\frac{\partial}{\partial a} SSE &= \sum_{i=1}^N 2x_i(ax_i + b - y_i) = 0 \\ \frac{\partial}{\partial b} SSE &= \sum_{i=1}^N 2(ax_i + b - y_i) = 0\end{aligned}$$

for the unknown parameters a and b . This amounts to solving a system of two linear equations for a and b :

$$\begin{aligned}\left(\sum_{i=1}^N x_i^2\right)a + \left(\sum_{i=1}^N x_i\right)b &= \sum_{i=1}^N x_i y_i \\ \left(\sum_{i=1}^N x_i\right) + Nb &= \sum_{i=1}^N y_i\end{aligned} \Leftrightarrow \underbrace{\begin{bmatrix} \sum_{i=1}^N x_i^2 & \sum_{i=1}^N x_i \\ \sum_{i=1}^N x_i & N \end{bmatrix}}_R \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N x_i y_i \\ \sum_{i=1}^N y_i \end{bmatrix}$$

From which we compute the least squares solutions of a and b as

$$\begin{aligned}\hat{a}_{LS} &= \frac{N(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{N(\sum_i x_i^2) - (\sum_i x_i)^2} \\ \hat{b}_{LS} &= \frac{(\sum_i x_i^2)(\sum_i y_i) - (\sum_i x_i y_i)(\sum_i x_i)}{N(\sum_i x_i^2) - (\sum_i x_i)^2}\end{aligned}$$

Caution: The solution for $\hat{a}_{LS}, \hat{b}_{LS}$ described above is valid only if the denominator $N(\sum_i x_i^2) - (\sum_i x_i)^2$ is not zero. This equivalent to asking $\det(R) \neq 0$.

The method described above can be extended to the case when there are more than two parameters to be estimated.

Problem 2. Given an $n \times m$ matrix A and an n -vector y , find the m -vector \hat{x}_{LS} so that it minimizes the SSE among all n -vectors x :

$$SSE := \|y - Ax\|^2 = \sum_{i=1}^n (y_i - (Ax)_i)^2$$

where $(Ax)_i$ denotes the i -th entry of the vector Ax . □

The vector \hat{x}_{LS} is called the *the solution to the problem $Ax = y$ in the least squares sense*, or simply *the least squares solution to the problem $Ax = y$* . This might be confusing; remember that the “the solution to the problem $Ax = y$ ” means “the vector x so that, given A and y , x satisfies $Ax = y$ ”. When no such x exists, we settle for a vector that minimizes the sum of squared error between Ax and y , which is the least squares solution.

Solution to 2. We will solve this problem in a manner similar to Problem 1, by determining the scalars x_i that satisfies

$$\frac{\partial}{\partial \hat{x}_1} SSE = \frac{\partial}{\partial \hat{x}_2} SSE = \dots = \frac{\partial}{\partial \hat{x}_m} SSE = 0.$$

This equation can be re-written as

$$\frac{d}{dx} SSE := \left[\frac{\partial}{\partial x_1} SSE \quad \frac{\partial}{\partial x_2} SSE \quad \dots \quad \frac{\partial}{\partial x_m} SSE \right] = 0.$$

When x is a vector and $f(x)$ is a scalar function of x , the Taylor series expansion of f around x is

$$\begin{aligned}f(x_o + \delta x) &= f(x_o) + \frac{df}{dx} \Big|_{x_o} \delta x \\ &\quad + \frac{1}{2} \delta x^T \frac{d^2 f}{dx^2} \Big|_{x_o} \delta x + \dots\end{aligned}$$

Since each term must be a scalar, the derivative df/dx is a row vector and the second derivative $d^2 f/dx^2$ is a matrix. The second derivative $d^2 f/dx^2$ is called a *Hessian* of f .

Note that

$$SSE = (y - Ax)^T(y - Ax) = y^T y - 2y^T Ax + x^T A^T Ax \quad (1.4)$$

therefore, the solution to $\frac{d}{dx}SSE = 0$ is

$$\frac{\partial SSE}{\partial \hat{x}} = 2A^T Ax - 2A^T y = 0 \Leftrightarrow \hat{x} = (A^T A)^{-1} A^T y,$$

which yields the least squares estimate for x :

$$\hat{x}_{LS} = (A^T A)^{-1} A^T y. \quad (1.5)$$

Matrix inversion is a computationally expensive operation that is sensitive to small changes in the entries of the matrix. Instead, one can compute the least squares solution to $A(\cdot) = y$ efficiently in MATLAB by using `x = A \ y`.

The *quality of the fit* can be judged from the SSE achieved by the estimate \hat{x}_{LS} . Replacing (1.5) in (1.4), we get

$$\frac{SSE}{\|y\|^2} = \frac{\|y - A\hat{x}_{LS}\|^2}{\|y\|^2} = \frac{(y - A\hat{x}_{LS})^T (y - A\hat{x}_{LS})}{y^T y} = 1 - \frac{y^T A\hat{x}_{LS}}{y^T y}$$

1.2.2 Geometric interpretation of least squares

Consider a general linear system of equations:

$$Ax = y, \quad \text{where } A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, y \in \mathbb{R}^m \quad (1.6)$$

Question: When does a solution to the problem $A(\cdot) = y$ exist?

Example: Consider the following set of equations:

$$\begin{aligned} 5x_1 + 2x_2 &= 3 \\ 10x_1 + 4x_2 &= 7 \end{aligned} \Rightarrow \underbrace{\begin{bmatrix} 5 & 2 \\ 10 & 4 \end{bmatrix}}_A \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_x = \underbrace{\begin{bmatrix} 3 \\ 7 \end{bmatrix}}_y \quad (1.7)$$

It is clear that the second equation is inconsistent with the first, so that there is no $x = [x_1, x_2]^T$ that satisfies both the equations. We say that the equation (1.7) is *inconsistent*.

Another way of understanding the inconsistency comes from viewing matrices as input-output machines. An $n \times m$ matrix A taken an m -vector as an “input” and produces a n -vector as the “output”. For an arbitrary m -vector x as the input, the output Ax is a *weighted linear combination of the columns of A , where the weights are the entries of x* . To see that is true, consider a following matrix A and vector x :

$$A := \begin{bmatrix} 1 & 2 & 4 \\ 10 & 3 & 2.1 \\ 5 & -2 & 0 \\ 2 & 1 & 23 \end{bmatrix}, \quad x := \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (1.8)$$

$$\Rightarrow Ax = \begin{bmatrix} x_1 + 2x_2 + 4x_3 \\ 10x_1 + 3x_2 + 2.1x_3 \\ 5x_1 - 2x_2 + 0x_3 \\ 2x_1 + x_2 + 23x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 10 \\ 5 \\ 2 \end{bmatrix} x_1 + \begin{bmatrix} 2 \\ 3 \\ -2 \\ 1 \end{bmatrix} x_2 + \begin{bmatrix} 4 \\ 2.1 \\ 0 \\ 23 \end{bmatrix} x_3 \quad (1.9)$$

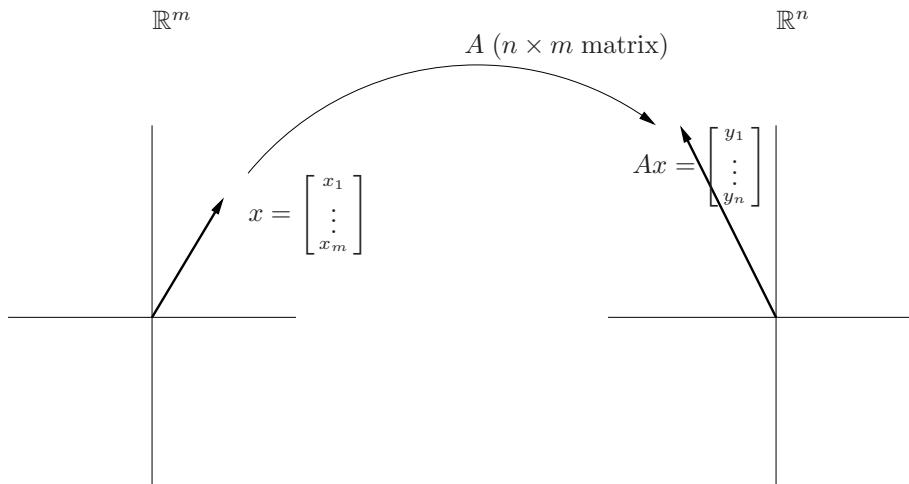


Figure 1.1: A $m \times n$ matrix A is a linear transformation from \mathbb{R}^m to \mathbb{R}^n .

Very important
 er that a matrix
 transformation.
 $z = x + y$, then
 $(x + y) = Ax + Ay$.

You can easily verify that the statement above is true for an arbitrary matrix A and vector x , as long as their dimensions are compatible so that the product Ax is well-defined.

The set of output vectors that a matrix A is capable of producing, when the input x takes all possible values, is called the *range space* of the matrix A , and is denoted by $\mathcal{R}(A)$.

$$\mathcal{R}(A) = \{y \in \mathbb{R}^n \mid \exists x \in \mathbb{R}^m \text{ such that } y = Ax\} \tag{1.10}$$

It is the set of all vectors that A can produce as “output” when you try all possible input vectors.

Now going back to our example (1.7): given the vector y and the matrix A , we want to find a vector x such that “output” Ax for the “input” x is equal to y . To answer the first question of whether such a vector x exists (before we spend time trying to find it), use what you learned about possible outputs of a matrix. Clearly, the given vector y must lie in the range space of the given matrix A for any Ax to equal y . In other words, such an x exists if and only if y is a linear combination of the columns of A . Now let’s check if that is the case:

$$\alpha \begin{bmatrix} 5 \\ 10 \end{bmatrix} + \beta \begin{bmatrix} 2 \\ 4 \end{bmatrix} = 5\alpha \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 2\beta \begin{bmatrix} 1 \\ 2 \end{bmatrix} = (5\alpha + 2\beta) \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

So all linear combinations of the columns of A lie on one vector: $[1, 2]^T$. Since $[3, 7]^T$ does not lie along this vector, we now know that there is no solution to (1.7).

To summarize:

Given $A \in \mathbb{R}^{m \times n}$ and $y \in \mathbb{R}^n$, there is a vector x that satisfies $Ax = y$ if and only if $y \in \mathcal{R}(A)$.

When a solution does not exist, we have a situation that is depicted in Figure 1.2. The least squares solution \hat{x}_{LS} is the vector in the input space of the matrix A so that the corresponding output $\hat{y}^* := A\hat{x}_{LS}$ is closest to y . Therefore, y^* is the vector in $\mathcal{R}(A)$ that is closest to y , and can be determined by dropping a perpendicular to $\mathcal{R}(A)$ from y . Note that the idea of “dropping a perpendicular” works not only in \mathbb{R}^2 and \mathbb{R}^3 , but can be extended to \mathbb{R}^m for arbitrary m .

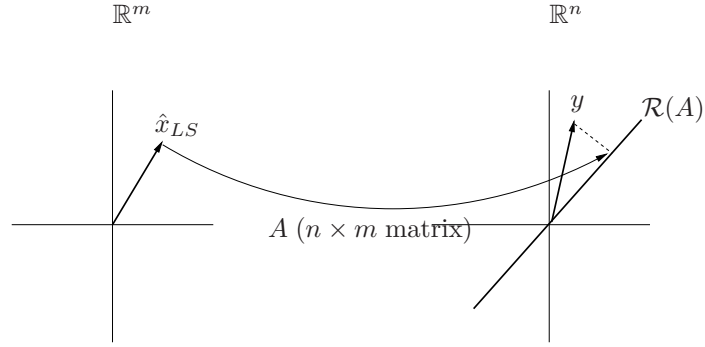


Figure 1.2: No solution to $A(\cdot) = y$. The least squares solution \hat{x}_{LS} is characterized by the fact that the perpendicular from y onto $\mathcal{R}(A)$ is precisely the point $A\hat{x}_{LS}$.

1.3 ARX model identification by least squares

Consider a system whose outputs and inputs are described by a difference equation:

$$\begin{aligned} y[k+n] + \beta_{n-1}y[k+n-1] + \beta_{n-2}y[k+n-2] + \cdots + \beta_1y[k+1] + \beta_0y[k] \\ = \alpha_mu[k+m] + \alpha_{m-1}u[k+m-1] + \cdots + \alpha_1u[k+1] + \alpha_0u[k]. \end{aligned} \quad (1.11)$$

We re-write (1.11) to see that the output $y[\cdot]$ can be computed recursively from the inputs and past outputs:

$$\begin{aligned} y[k+n] = -\beta_{n-1}y[k+n-1] - \beta_{n-2}y[k+n-2] - \cdots - \beta_1y[k+1] - \beta_0y[k] \\ + \alpha_mu[k+m] + \alpha_{m-1}u[k+m-1] + \cdots + \alpha_1u[k+1] + \alpha_0u[k] \end{aligned} \quad (1.12)$$

or, equivalently,

$$\begin{aligned} y[k] = -\beta_{n-1}y[k-1] - \beta_{n-2}y[k-2] - \cdots - \beta_1y[k-(n-1)] - \beta_0y[k-n] \\ + \alpha_mu[k-(n-m)] + \alpha_{m-1}u[k-(n-m+1)] + \cdots + \alpha_1u[k-(n-1)] + \alpha_0u[k-n] \end{aligned}$$

Note that the output $y[k]$ at time k depends only on the past outputs $y[k-1], y[k-2], \dots, y[k-n+1], y[k-n]$ and inputs $u[k-n+m], u[k-n+m-1], \dots, u[k-n+1], u[k-n]$.

In physical systems where the present output does not depend on future inputs, $m \leq n$. Such systems are called *causal* systems. If $m < n$, the present output $y[k]$ is affected only by the past inputs, whereas if $m = n$, the present output is affected also by the present input. Eq. (1.11) is said to be of *order* n .

The equation (1.11) can be compactly written as

$$y[k] = \varphi[k]^T \boldsymbol{\theta} \quad (1.13)$$

where $\boldsymbol{\theta}$ is a vector containing the parameters $\alpha_i, i = 0, \dots, m$ and $\beta_j, j = 0, \dots, n-1$, and

$\varphi[k]$ contains past inputs and outputs:

$$\boldsymbol{\theta} := \begin{bmatrix} -\beta_{n-1} \\ \vdots \\ -\beta_1 \\ -\beta_0 \\ \alpha_m \\ \alpha_{m-1} \\ \vdots \\ \alpha_1 \\ \alpha_0 \end{bmatrix} \quad \varphi[k] := \begin{bmatrix} y[k-1] \\ \vdots \\ y[k-n] \\ u[k-n+m] \\ \vdots \\ u[k-n+1] \\ u[k-n] \end{bmatrix}$$

The vector $\varphi[k]$ is called the *regressor vector* and the equation (1.13) is called an ARX model (Auto-Regression model with eXogeneous inputs).

Eq. (1.13) is a linear equation of the form $y = Ax$, but there are $n + m + 1$ unknowns and only one equation. By collecting the values of the input and the output signals for N time steps, where $N \geq n + m + 1$, we can construct the following system of linear equations with at least as many equations as there are unknowns:

$$\mathbf{y} = \Phi \boldsymbol{\theta} \quad (1.14)$$

where

$$\Phi := \begin{bmatrix} \varphi^T[1] \\ \varphi^T[2] \\ \vdots \\ \varphi^T[N] \end{bmatrix} = \begin{bmatrix} y[0] & \dots & y[1-n] & u[1-n+m] & \dots & u[1-n] \\ y[1] & \dots & y[2-n] & u[2-n+m] & \dots & u[2-n] \\ \vdots & & \vdots & & & \vdots \\ y[N-1] & \dots & y[N-n] & u[N-n+m] & \dots & u[N-n] \end{bmatrix}, \quad \mathbf{y} := \begin{bmatrix} y[1] \\ y[2] \\ \vdots \\ y[N] \end{bmatrix} \quad (1.15)$$

In practice, when we construct \mathbf{y} and Φ from noisy data, there may not be a solution to (1.14). In that case, we compute the least squares solution to $\Phi \boldsymbol{\theta} = \mathbf{y}$:

$$\hat{\boldsymbol{\theta}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}, \quad (1.16)$$

which is the *least squares estimate* of $\boldsymbol{\theta}$.

To summarize, least squares identification of the parameters of an ARX model consists of the following steps:

1. Choose an $N > n + m + 1$.
2. Apply a probe input signal $u[k]$, $k \in \{1, 2, \dots, N\}$ to the system. The probe signal should be sufficiently “rich”.
3. Measure the corresponding output $y[k]$, $k \in \{1, 2, \dots, N\}$.
4. Construct the vector \mathbf{y} and matrix Φ described in (1.15).
5. Determine the value of the parameter vector $\boldsymbol{\theta}$ by computing the least squares solution to the equation $\Phi \boldsymbol{\theta} = \mathbf{y}$.

The MATLAB command `arx` from the identification toolbox can be used to perform least squares identification of ARX models.

Chapter 2

Mathematical models of physical systems

2.1 ODE models

Example: (*pendulum*): Consider the inverted pendulum shown in Figure 2.1. It consists of a mass-less link of length ℓ , with a point mass m at the end. There is an actuator which can apply a torque of T to the link at the base around the z axis (the z axis is normal to the plane of the paper). The drag on the tip mass due to its motion through air is given by $-b\dot{\theta}$. The angle of the link from the horizontal is denoted by θ (positive in the CCW direction). The dynamics of the angle θ is determined by the following Ordinary Differential Equation (ODE):

Derive this ODE from Newton's laws.

$$m\ell^2\ddot{\theta}(t) = -mg\ell \cos\theta(t) - b\dot{\theta}(t) + T(t), \quad (2.1)$$

where T is the CCW torque applied to the base, and g is the acceleration due to gravity. Dividing throughout by $m\ell^2$, we get

$$\ddot{\theta}(t) = -\frac{b}{m\ell^2}\dot{\theta}(t) - \frac{g}{\ell} \cos\theta(t) + u(t), \quad (2.2)$$

where $u(t) := \frac{T(t)}{m\ell^2}$.

Define

$$x_1 := \theta \qquad x_2 := \dot{\theta}. \quad (2.3)$$

The ODE (2.2) can now be re-written as

$$\begin{aligned} \dot{x}_1 &= x_2 & &= f_1(x, u, t) \\ \dot{x}_2 &= -\frac{b}{m\ell^2}x_2 - \frac{g}{\ell} \cos x_1 + u(t) & &= f_2(x, u, t) \end{aligned} \quad (2.4)$$

Which is equivalent to the vector form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} f_1(x, u, t) \\ f_2(x, u, t) \end{bmatrix} \quad (2.5)$$

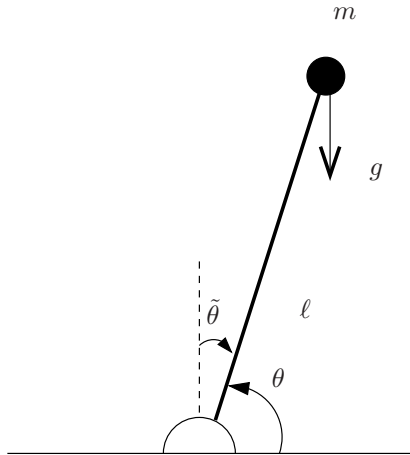


Figure 2.1: Inverted pendulum.

We will deal with dynamical systems that are modeled by a number of coupled first-order differential equations:

$$\begin{aligned}
 \dot{x}_1 &= f_1(x_1, x_2, \dots, x_n, u_1, \dots, u_p, t) \\
 \dot{x}_2 &= f_2(x_1, x_2, \dots, x_n, u_1, \dots, u_p, t) \\
 &\vdots \\
 \dot{x}_n &= f_n(x_1, x_2, \dots, x_n, u_1, \dots, u_p, t)
 \end{aligned} \tag{2.6}$$

Vector notation can be used to represent these equations in compact form. Define

$$x := \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad u := \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \end{bmatrix} \quad f(x, u, t) := \begin{bmatrix} f_1(x, u, t) \\ \vdots \\ f_n(x, u, t) \end{bmatrix}$$

and rewrite (2.6) as

$$\dot{x} = f(x, u, t), \quad x \in \mathbb{R}^n. \tag{2.7}$$

The eq. (2.7) is called the *state equation* and x is referred to as the *state* of the dynamic system. Sometimes, another equation

$$y = h(x, u, t), \quad y \in \mathbb{R}^q \tag{2.8}$$

is associated with (2.7), thereby defining a q -dimensional output vector y . The output y frequently consists of signals that are physically measured.

The behavior of dynamic systems without external inputs is governed by the so-called *unforced* state equation:

$$\dot{x} = f(x, t) \tag{2.9}$$

which is obtained by setting $u \equiv 0$ in (2.7). If an external input u is specified as an explicit function of time $\gamma(t)$, or as a feedback function of the state $\gamma(x)$, or both $\gamma(x, t)$, then substituting this function in (2.7) also leads to (2.9).

If the function f in the right hand side of (2.9) does not depend on t , so that the state equation has the form

$$\dot{x} = f(x) \quad (2.10)$$

then the system is said to be *time-invariant*.

Let $x(t)$ be the solution of (2.7), with an input signal $u(t)$, that starts at some initial state $x_0 := x(t_0)$ at initial time t_0 . The locus of the solution $x(t)$ for all $t \geq t_0$ is called the trajectory of (2.7) from x_0 with input $u(t)$. Except for a very limited class of ODEs for which analytical solutions are possible, one has to resort to numerical integration to determine the trajectories. It should be emphasized that numerical integration of an ODE yields only an approximation to the trajectory.

Some standard notational confusions: $s(t)$ may mean the value of the signal s at a specific time t , or it may mean the entire time history of the signal. This can be avoided by using the cumbersome notation $\{s(t)\}_a^b$ to denote a signal $s(t)$ during the time interval $a \leq t \leq b$. We try to adhere to this latter notation in these notes, but lapse – not too infrequently – into the standard confusing notation.

Systems where the present output does not depend on future inputs are called *causal* systems. In a causal system the output at time t_0 may depend only on inputs applied at times $t \leq t_0$ but not on inputs applied at $t > t_0$.

For *linear systems*, the state model (2.7) takes the special form:

$$\begin{aligned} \dot{x} &= A(t)x + B(t)u \\ y &= C(t)x + D(t)u \end{aligned}$$

Linear time invariant (LTI) systems take the special form:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

where the matrices A, B, C, D do not depend on time.

2.1.1 Numerical integration

In MATLAB, we can use the following script to compute the (approximate) trajectory to (2.2) by numerical integration:

```
clear all;
global g
input_sine_params = [];
input_sine_params.amp = 0.000; %Nm (put 0 to simulate free response)
input_sine_params.freq = 10; %Hz
%model parameters
ModelParams = [];
ModelParams.LinkLen=0.20; %m
ModelParams.DampingCoeff=0.0005; %Ns/m
ModelParams.TipMass = 0.02; %Kg
g = 9.8; % accn due to gravity, m/s
x_init = [pi*0.45; 0];
OPTIONS=odeset('RelTol',1e-5,'AbsTol',1e-8);%these control the accuracy
[times,x_hist] = ode45(@f_InvPendulumModel,[0 10],x_init,OPTIONS,...
    ModelParams, input_sine_params);
plot(t,x(:,1));
```

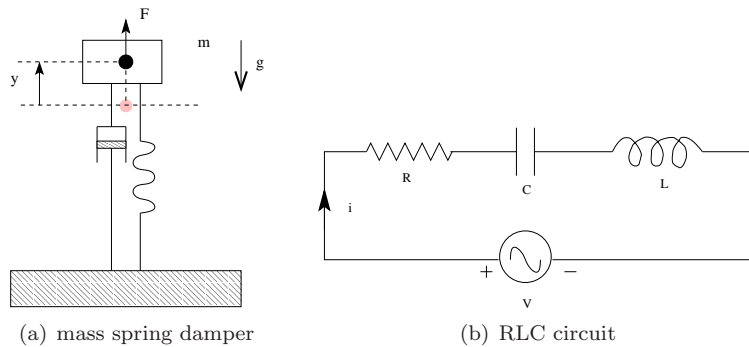


Figure 2.2: A mechanical and an electrical system that are governed by similar ODEs.

where the function `f_InvPendulumModel` is defined in a file named “`f_InvPendulumModel.m`”:

```
function [x_dot] = f_InvPendulumModel(t,x,ModelParams,...
    input_sine_params)

global g
x1 = x(1);
x2 = x(2);
amp = input_sine_params.amp;
freq = input_sine_params.freq;
%model parameters
LinkLen = ModelParams.LinkLen; %m
DampingCoeff = ModelParams.DampingCoeff; %Ns/m
TipMass = ModelParams.TipMass; %Kg

T = amp*sin(2*pi*freq*t);
x1_dot = x2;
x2_dot = - DampingCoeff/(TipMass*LinkLen^2)*x2 - g/LinkLen*cos(x1) ...
    + T/(TipMass*LinkLen^2);

x_dot = [x1_dot; x2_dot];
```

2.1.2 Modeling exercise:

Determine the mathematical model of the following systems:

1. Mass-spring-damper: consider the mass-spring-damper systems shown in Figure 2.2(a). The vertical displacement of the center of the mass, from its equilibrium position, is denoted by $y(t)$. Determine the differential equation that relates y to the external force F . The spring coefficient is k and damping coefficient is b .
2. RLC circuit: consider the RLC circuit shown in Figure 2.2(b). Determine the differential equation that relates the current $i(t)$ to the externally supplied voltage $V(t)$.

2.1.3 Equilibrium point and linearization around it

A point x^* in the state space is said to be an equilibrium point of (2.9) if $f(x^*, t) = 0$ for every t . That is, whenever the starts at x^* , it stays at x^* for all time. If the state starts

close to an equilibrium point, the dynamic evolution of the deviation from the equilibrium $x(t) - x^*$ is sometimes described by a linear system.

In the pendulum model (2.4), $x = [\pi/2, 0]^T$ is an equilibrium point. Suppose $\theta(t_0) \approx \frac{\pi}{2}$ for some t_0 . Define the deviation of θ from $\frac{\pi}{2}$ as

$$\tilde{\theta} := \frac{\pi}{2} - \theta.$$

and then determine the dynamics of $\tilde{\theta}$ from (2.2), which leads to the following equation

$$\begin{aligned} \ddot{\tilde{\theta}} &= -\frac{b}{m\ell^2}\dot{\tilde{\theta}} + \frac{g}{\ell}\sin(\tilde{\theta}) - u \\ &\approx -\frac{b}{m\ell^2}\dot{\tilde{\theta}} + \frac{g}{\ell}\tilde{\theta} - u \end{aligned} \quad (2.11)$$

Thus, if the position of the pendulum is close to the upright position, the dynamics of the deviation-from-the-upright can be *approximated* by

$$\ddot{\tilde{\theta}} = -\frac{b}{m\ell^2}\dot{\tilde{\theta}} + \frac{g}{\ell}\tilde{\theta} - u \quad (2.12)$$

With the states x_1 and x_2 as defined in (2.3), eq. (2.12) can be represented in the state space form as

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{g}{\ell} & -\frac{b}{m\ell^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} u.$$

Chapter 3

Continuous time SISO LTI systems

We will deal extensively with causal linear time invariant (LTI) systems with a single input and a single output (SISO). Dynamics of SISO LTI systems can be expressed as a linear, constant coefficient differential equation of the following form:

Exercise: express (3.1) as a state model.

$$y^{(n)} + \beta_{n-1}y^{(n-1)} + \dots + \beta_2\ddot{y} + \beta_1\dot{y} + \beta_0y = \alpha_m u^{(m)} + \alpha_{m-1}u^{(m-1)} + \dots + \alpha_2\ddot{u} + \alpha_1\dot{u} + \alpha_0u \quad (3.1)$$

where $y^{(n)}$ is the n -th derivative of y , and α_i, β_j 's are parameters that do not change with time.

3.1 Laplace transform

The Laplace transform of a signal $\{y(t)\}$ is defined by

$$Y(s) = \mathcal{L}(y(t)) := \int_{-\infty}^{\infty} y(\tau)e^{-s\tau} d\tau, \quad s \in \mathbb{C} \quad (3.2)$$

The integral exists (has a well defined, finite value) only if the signal $y(t)$ grows with t at a rate slower than the exponential e^{-st} decays, or vice versa. The rate of decay depends on the complex number s . Therefore, for a given signal $y(t)$, for the Laplace transform to exist, the value of s must be such that the integral above converges. The *region of convergence* of a Laplace transform $Y(s)$ is the region in the complex plane \mathbb{C} that s can take values in so that $Y(s)$ is finite and well defined.

Example: The unit step signal $1(t)$ defined below will be useful in the subsequent discussion:

$$1(t) := \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases} \quad (3.3)$$

Let us now evaluate the Laplace transform of

$$\begin{aligned} x(t) &= e^{(a+bj)t} \mathbf{1}(t) := \begin{cases} e^{a+bj}t & t \geq 0 \\ 0 & t < 0 \end{cases} \\ \Rightarrow \mathcal{L}(x(t)) = X(s) &= \int_0^\infty x(t)e^{-st} dt = \int_0^\infty e^{(a+bj-s)t} dt \\ &= \frac{1}{a+bj-s} \left(\lim_{\tau \rightarrow +\infty} e^{(a-Re(s)+(b-Im(s))j)\tau} - 1 \right) \\ &= \begin{cases} \frac{1}{s-(a+bj)} & \text{if } Re(s) > a \\ \text{undefined} & \text{otherwise} \end{cases} \end{aligned}$$

Hence, we say that the Laplace transform of the signal $e^{(a+bj)t} \mathbf{1}(t)$ is $\frac{1}{s-(a+bj)}$ with a region of convergence $Re(s) > a$.

The Laplace transform $Y(s)$ of a signal $\{y(t)\}$ is a complex number, whose values depend on the complex argument s . Therefore Laplace transform can be thought of as a complex function.

Exercise: plot the values of $Y(s)$ for a few complex numbers s .

The Laplace transform is linear:

$$\mathcal{L}(ax(t) + by(t)) = a\mathcal{L}(x(t)) + b\mathcal{L}(y(t)),$$

where $\{x(t)\}, \{y(t)\}$ are signals and a, b are complex scalars.

Why does the ROC matter? Two quite different signals may have the same Laplace transform (but distinct ROCs). So, if the Laplace transform is specified without the ROC, there is ambiguity as to which signal it is the Laplace transform of. Example:

$$y_1(t) = e^{-t} \mathbf{1}(t) \quad Y_1(s) = \frac{1}{1+s}, \quad \text{ROC: } s > -1 \quad (3.4)$$

$$y_2(t) = -e^t \mathbf{1}(-t) \quad Y_2(s) = \frac{1}{1+s}, \quad \text{ROC: } s < -1 \quad (3.5)$$

Inverse Laplace transform: The inverse Laplace transform is defined as

$$y(t) = \mathcal{L}^{-1}(Y(s)) := \int_{\sigma-j\infty}^{\sigma+j\infty} Y(s)e^{st} ds, \quad (3.6)$$

Does the ROC play a role in evaluating the inverse Laplace transform? which recovers the signal $y(t)$ from the Laplace transform $Y(s)$.

The inverse transform is also linear. That is, if $X(s)$ and $Y(s)$ are Laplace transforms of two signals $x(t)$ and $y(t)$ with associated ROCs A and B such that $A \cap B$ is not empty, then the inverse Laplace transform of $X(s) + Y(s)$ is given by $\mathcal{L}^{-1}(X(s) + Y(s)) = x(t) + y(t)$.

Evaluation of the inverse transform involves complex integration, but usually the signal can be recovered from $Y(s)$ by partial fraction expansion and using knowledge of Laplace transform of complex exponential signals. Of course, the ROC of the transform $Y(s)$ must be specified. Example:

$$Y(s) = \frac{7s^3 + 12s^2 + 28s + 18}{s^4 + 2s^3 + 11s^2 + 18s + 18}, \quad \text{ROC: } Re(s) > 0.$$

What is $\mathcal{L}^{-1}(Y(s))$? From partial fraction expansion, we get

$$Y(s) = \frac{1}{s+(1+j)} + \frac{1}{s+(1-j)} + \frac{5s}{s^2+9}$$

$$\Rightarrow \mathcal{L}^{-1}(Y(s)) = \mathcal{L}^{-1}\left(\frac{1}{s+(1+j)}\right) + \mathcal{L}^{-1}\left(\frac{1}{s+(1-j)}\right) + \mathcal{L}^{-1}\left(\frac{5s}{s^2+9}\right)$$

These are the possibilities for each term:

$$\mathcal{L}^{-1}\left(\frac{1}{s+(1+j)}\right) = \begin{cases} e^{-(1+j)t}1(t) & \text{if ROC: } \operatorname{Re}(s) > -1 \\ -e^{-(1+j)t}1(-t) & \text{if ROC: } \operatorname{Re}(s) < -1 \end{cases}$$

$$\mathcal{L}^{-1}\left(\frac{1}{s+(1-j)}\right) = \begin{cases} e^{-(1-j)t}1(t) & \text{if ROC: } \operatorname{Re}(s) > -1 \\ -e^{-(1-j)t}1(-t) & \text{if ROC: } \operatorname{Re}(s) < -1 \end{cases}$$

$$\mathcal{L}^{-1}\left(\frac{s}{s^2+\omega^2}\right) = \begin{cases} 5 \cos(3t) 1(t) & \text{if ROC: } \operatorname{Re}(s) > 0 \\ -5 \cos(3t) 1(-t) & \text{if ROC: } \operatorname{Re}(s) < 0 \end{cases}$$

The ROC specification of the transform $Y(s)$ as $\operatorname{Re}(s) > 0$ implies that the possibilities above that correspond to left-half plane ROCs are ruled out. So,

$$\mathcal{L}^{-1}(Y(s)) = e^{-(1+j)t}1(t) + e^{-(1-j)t}1(t) + 5 \cos(3t)1(t)$$

A few useful properties of the Laplace transform are

1. Laplace transform of the derivative of a signal: if $Y(s)$ is the Laplace transform of a signal $\{y(t)\}$, then the transform of the derivative \dot{y} is

$$\mathcal{L}(\dot{y}(t)) = sY(s).$$

This property follows from differentiating the inverse Laplace transform:

$$y(t) = \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} Y(s)e^{st} ds$$

which leads to

$$\dot{y}(t) = \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} sY(s)e^{st} ds,$$

which shows that $sY(s)$ is the Laplace transform of $\dot{y}(t)$.

2. if $Y(s)$ is the Laplace transform of a signal $\{y(t)1(t)\}$, then the transform of the signal $\dot{y}1(t)$ is

$$\mathcal{L}(\dot{y}(t)1(t)) = sY(s) - y(0).$$

If $y(0) \neq 0$, but $y(t) = 0$ for all $t < 0$, the derivative $\dot{y}(t)$ does not exist at $t = 0$. In such cases, we talk about the signal $\dot{y}(t)1(t)$.

This can be shown by integration by parts:

$$\begin{aligned} \mathcal{L}(\dot{y}1(t)) &= \int_{-\infty}^{\infty} \dot{y}(t)1(t)e^{-st} dt = \int_0^{\infty} e^{-st}\dot{y}(t) dt \\ &= \left[e^{-st}y(t) - \int_0^{\infty} (-s)e^{-st}y(t) dt \right]_0^{\infty} = -y(0) + sY(s), \end{aligned}$$

where we have used the fact that, since $e^{-st}y(t)$ is integrable for all values of s in the ROC, $e^{-st}y(t) \rightarrow 0$ as $t \rightarrow \infty$.

3. Time shifting: If $Y(s)$ is the Laplace transform (with $\text{ROC} = R$) of a signal $y(t)$, then the Laplace transform of the time shifted signal $y(t - \tau)$ is

$$\mathcal{L}(y(t - \tau)) = e^{-s\tau}Y(s), \text{ with } \text{ROC} = R$$

4. The convolution $x(t) * y(t)$ of two signals $x(t)$ and $y(t)$ are defined as

$$x(t) * y(t) := \int_{-\infty}^{+\infty} x(\tau)y(t - \tau)d\tau. \quad (3.7)$$

The Laplace transform of the convolution of two signals is given by the product of their Laplace transforms:

$$z(t) = x(t) * y(t) \quad \Leftrightarrow \quad Z(s) = X(s)Y(s). \quad (3.8)$$

Impulse response The impulse response $\{h(t)\}$ of a LTI system is its output when it is driven by an impulse input $\delta(t)$ with all initial conditions set to 0. The function $\delta(t)$ is the so-called *Dirac delta* function that is defined by the property that

$$\int_{-\infty}^{\infty} \delta(t - t_0)f(t)dt = f(t_0) \text{ for every function } f(t).$$

The Dirac delta function $\delta(\cdot)$ can be visualized as a function that is zero everywhere except where the argument is 0, where it takes an infinitely large value, but in such a way that $\int_{-\infty}^{\infty} \delta(t)dt = 1$.

It is impossible to determine the impulse response experimentally, though it is a useful concept for analysis.

3.2 Transfer function

Consider the following linear differential equation that models the dynamics of a mass m that is acted on by an external force u , when the mass has a spring and damper attached to it:

$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = u(t), \quad t \geq 0. \quad (3.9)$$

What does this mean to take Laplace transform of both sides?

Take the Laplace transform of both sides to get

$$m(s^2X(s) - sx(0) - \dot{x}(0)) + b(sX(s) - x(0)) + kX(s) = F(s) \\ \left(s^2 + \frac{b}{m}s + \frac{k}{m}\right)X(s) = F(s) + \frac{s+b}{m}x(0) + \dot{x}(0)$$

which can be rearranged into

$$Y(s) = \underbrace{\frac{1}{s^2 + \frac{b}{m}s + \frac{k}{m}}}_{H_c(s)}U(s) + \underbrace{\frac{as+b}{s^2 + \frac{b}{m}s + \frac{k}{m}}}_{E(s)} \quad (3.10)$$

$$Y(s) = \frac{1}{s^2 + \frac{b}{m}s + \frac{k}{m}}U(s) + \frac{\frac{x(0)}{m}s + (\frac{b}{m}x(0) + \dot{x}(0))}{s^2 + \frac{b}{m}s + \frac{k}{m}} \quad (3.11)$$

For the LTI system (3.1), carrying out the above procedure leads to a similar expression:

$$Y(s) = H_c(s)U(s) + E(s), \quad (3.12)$$

where the complex function

$$H_c(s) := \frac{\alpha_m s^m + \alpha_{m-1} s^{m-1} + \cdots + \alpha_1 s + \alpha_0}{s^n + \beta_{n-1} s^{n-1} + \cdots + \beta_1 s + \beta_0} \quad (3.13)$$

is called the (*continuous time*) *transfer function* from u to y for the system (3.1), and $S(s)$ depends (as it does in (3.10)) on the initial conditions.

Upon taking the inverse s -transform of (3.12), we get

$$y(t) = \mathcal{L}^{-1}(H_c(s)U(s)) + \mathcal{S}^{-1}(E(s))$$

which shows that the output of the system is a superposition of the response due to the input and the response due to initial conditions.

If all the initial conditions are 0, then (3.12) reduces to

$$Y(s) = H_c(s)U(s) \quad (3.14)$$

Since $y(t)$ can be recovered by taking the inverse Laplace transform of the right hand side, it shows that when all initial conditions are zero, the transfer function together with the input completely determines the output.

The roots of the denominator polynomial of $H_c(s)$ are called the *poles* of the system and the roots of its numerator polynomial are called the *zeros* of the system.

A system is called *BIBO stable* if the output is bounded for every bounded input. It has two equivalent characterizations for LTI systems

1. A LTI system is BIBO stable if and only if its impulse response $h(t)$ is absolutely integrable, i.e.,

$$\int_0^{\infty} |h(t)| dt < \infty.$$

2. A LTI system is BIBO stable if and only if each of the poles of its transfer function $H_c(s)$ have negative real parts.

If a system is BIBO stable, it can be shown that

$$\epsilon(t) := \mathcal{L}^{-1}(E(s)) \quad (3.15)$$

is a signal such that $s(t) \rightarrow 0$ as $t \rightarrow \infty$. Therefore, for a BIBO stable system, the output at steady state (as $t \rightarrow \infty$) is given by

$$y_{ss}(t) = \mathcal{L}^{-1}(H_c(s)U(s)) \quad (3.16)$$

Thus the transfer function provides information on steady state behavior of the system.

note that the transfer function is from $U(s)$ to $Y(s)$, not from $u(t)$ to $y(t)$. SO *continuous-time* transfer function is really a misnomer.

In MATLAB, `tf(num, den)` creates a continuous-time transfer function with numerator and denominator specified by `num, den`. Example: `tf(1, [1 2 3])` produces $\frac{1}{s^2+2s+3}$.

In MATLAB, `zpk(z, p, k)` creates a continuous-time transfer function with zeros, poles and gain specified by `z, p` and `k`.

Can you find a bounded input such that if $h(t)$ is not absolutely integrable, the response to this input will be unbounded?

3.2.1 Impulse response and transfer function

Let $H(s)$ be the transfer function of a system from the input u to the output y , and let it now be driven by an impulse input $\delta(t)$. In that case,

$$Y(s) = H(s)\mathcal{L}(\delta(t)) = H(s)$$

The MATLAB command `impz(sys)` can be used to plot the impulse response for the system specified in `sys`.

since the Laplace transform of $\delta(t)$ is 1. The impulse response $h(t)$ is the inverse Laplace transform of $Y(s)$, so that we get

$$h(t) = \mathcal{L}^{-1}(H(s)) \equiv H(s) = \mathcal{L}(h(t))$$

The transfer function of a system is therefore the Laplace transform of its impulse response.

For an arbitrary input $u(t)$, since the output $y(t)$ satisfies $Y(s) = H(s)U(s)$ when initial conditions are zero, from the convolution property of Laplace transforms, we get

$$y(t) = h(t) * u(t).$$

Thus, the response to an arbitrary input can be determined from the knowledge of the impulse response of the system.

3.3 Steady state response to sinusoidal inputs

Stable LTI systems have the unique (and hugely advantageous) property that the output of the system when driven by a sinusoidal input, after transients have died down, is also a sinusoid of the same frequency as the input signal, differing possibly in amplitude and phase. Suppose a system with a transfer function $H(s)$ that is BIBO stable is driven with a sinusoid input signal with frequency ω rad/s:

$$u(t) = A \sin(\omega t + \phi_u).$$

The output of the system is going to be of the form

$$y(t) = \underbrace{g(\omega)A \sin(\omega t + \phi + \Delta\phi)}_{\text{steady state}} + \underbrace{\epsilon(t)}_{\text{transient}}$$

where the *gain* $g(\omega)$ and the phase difference $\Delta\phi(\omega_u)$ are given by

$$\begin{aligned} g(\omega) &= |H(j\omega)| \\ \Delta\phi &= \angle H(j\omega) \end{aligned}$$

Since the system is BIBO stable, $\epsilon(t) \rightarrow 0$ as $t \rightarrow \infty$, so at steady state, when the transients have died down, the output signal just a scaled and phase shifter version of the input sinusoid.

Caution! The phrase *steady state* is crucial, it comes from the fact that this relationship is true only when transients have died down.

In MATLAB, `bode(sys)` can be used to draw the Bode plot of the system `sys`.

The *Bode plot* of a transfer function $H(s)$ depicts

1. the magnitude $|H(j\omega)|$ in dB ($20 \log_{10} |H(j\omega)|$) as a function of frequency ω (in rad/s or Hz).

2. the phase $\angle H(j\omega)$ as a function of frequency ω (in rad/s).

In view of the preceding discussion, the Bode plot shows how much a sinusoidal input signal with a particular frequency is amplified and phase-shifted at steady state by the system. The information contained in the Bode plot is called the *frequency response* of the system.

The statement about the response to sinusoids can be derived quite easily once we understand the response to complex exponential signals. To that end, consider a complex exponential input signal: $u(t) = e^{j\omega t}$. The output, with zero initial conditions, is:

$$\begin{aligned} y(t) &= h(t) * u(t) \\ &= \int_{-\infty}^{\infty} h(\tau) u(t - \tau) d\tau \\ &= \int_{-\infty}^{\infty} h(\tau) e^{j\omega(t-\tau)} d\tau \\ &= e^{j\omega t} \int_{-\infty}^{\infty} h(\tau) e^{-j\omega\tau} d\tau \\ &= e^{j\omega t} H(j\omega) \end{aligned}$$

Therefore, the response to a complex exponential signal is the same complex exponential signal multiplied by a complex scalar, the scalar being given by the value of the transfer function at $s = j\omega$. For BIBO stable system, even when initial conditions are non-zero, their effect dies to 0 as $t \rightarrow \infty$, so at steady state all that is left is the complex exponential.

To derive the response to a sinusoidal signal, note that

$$\sin(\omega t) = \frac{e^{j\omega t} - e^{-j\omega t}}{2j}$$

From linearity, we get that the output $y(t)$ to the input $u(t) := \sin(\omega t)$, with zero initial conditions, is

$$\begin{aligned} y(t) &= \frac{e^{j\omega t} H(j\omega) - e^{-j\omega t} H(-j\omega)}{2j} \\ &= \frac{(H(j\omega) - H(-j\omega)) \cos(\omega t) + j(H(j\omega) + H(-j\omega)) \sin(\omega t)}{2j} \\ &= \text{Imag}(H(j\omega)) \cos(\omega t) + \text{Real}(H(j\omega)) \sin(\omega t) \\ &= |H(j\omega)| \left(\frac{\text{Real}(H(j\omega))}{|H(j\omega)|} \sin(\omega t) + \frac{\text{Imag}(H(j\omega))}{|H(j\omega)|} \cos(\omega t) \right) \\ &= |H(j\omega)| (\sin(\omega t) \cos(\angle H(j\omega)) + \cos(\omega t) \sin(\angle H(j\omega))) \\ &= |H(j\omega)| \sin(\omega t + \angle H(j\omega)). \end{aligned}$$

Again BIBO stability ensures that the response to initial conditions die out and only the sinusoidal component (albeit with a different amplitude and phase but the *same frequency as the input*) remains at steady state.

Chapter 4

Discrete time SISO LTI systems

A discrete time signal is a sequence

$$u = \{u(0), u(1), \dots, u(k), \dots\} \quad \text{and} \quad y = \{y(0), y(1), \dots, y(k), \dots\}$$

Causal LTI systems can be described by difference equations of the form

$$y[k + n] = -\beta_{n-1}y[k + n - 1] - \beta_{n-2}y[k + n - 2] - \dots + \beta_1y[k + 1] - \beta_0y[k] + \alpha_mu[k + m] + \alpha_{m-1}u[k + m - 1] + \dots + \alpha_1u[k + 1] + \alpha_0u[k]. \quad (4.1)$$

The signal $\{y[n]\}$ can be constructed by iterative computation, given the input signal and initial conditions.

Example: Suppose a causal LTI system can be modeled by the following difference equation:

$$y[k + 2] = -2y[k + 1] - y[k] + u[k]$$

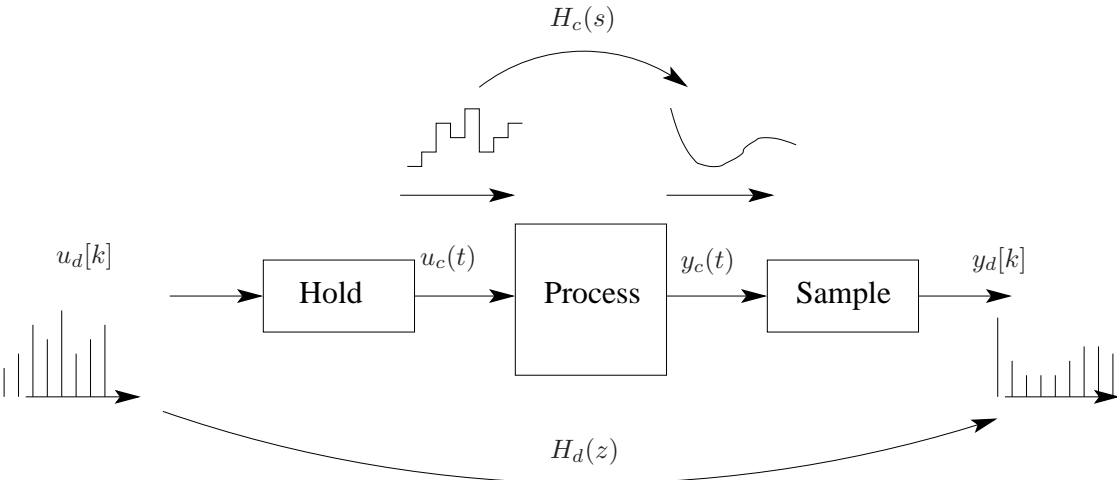


Figure 4.1: A plant with digital control and sampled output.

Suppose we know the initial conditions $y[0]$ and $y[1]$, and the entire input signal $u[k], k \in \{0, 1, \dots\}$ and that $u[k] = 0$ for $k < 0$. Since the system is causal, $y[k] = 0$ for $k < 0$. Now,

$$\begin{aligned} y[2] &= -2y[1] - y[0] + u[0] \\ y[3] &= -2y[2] - y[1] + u[1] \\ &\dots = \dots \end{aligned}$$

and so on, so that the value of $y[n]$ for an arbitrary n can be determined.

Exercise Can you determine $y[n]$ for an arbitrary n if only $y[0]$ is provided along with the inputs? For a n -th order difference equation of the form (4.1), how many initial conditions are needed?

4.1 Z-transforms

The z -transform of a discrete-time signal $\{x[k]\}, k \in \{-\infty, -2, -1, 0, 1, 2, \dots, \infty\}$ is defined as

$$X(z) = \mathcal{Z}(x[k]) := \sum_{k=-\infty}^{\infty} z^{-k} x[k], \quad (4.2)$$

where z is a complex number. The z -transform $X(z)$ may not exist for all complex numbers z . The set of all complex numbers z for which $X(z)$ exists is called the *region of convergence (ROC)* of the z -transform.

Examples:

$$y[k] = \begin{cases} 0 & \text{for } k < 0 \\ a^k & k \geq 0 \text{ (} a \in \mathbb{C} \text{)} \end{cases} \quad (4.3)$$

$$\Rightarrow Y(z) = \sum_{k=0}^{\infty} a^k z^{-k} = \sum_{k=0}^{\infty} \left(\frac{a}{z}\right)^k \quad (4.4)$$

$$= \begin{cases} \frac{1}{1-a/z} & \text{if } |a/z| < 1 \\ \text{undefined} & \text{if } |a/z| \geq 1 \end{cases} \quad (4.5)$$

So the z -transform of the sequence is $z/(z-a)$ with an ROC specified by $|z| > |a|$.

You should verify that the following are true:

$$1. y[k] = 1[k] := \begin{cases} 0 & \text{for } k < 0 \\ 1 & k \geq 0 \end{cases} \Rightarrow Y(z) = \frac{1}{1-z^{-1}}, \quad |z| > 1 \quad (4.6)$$

$$2. y[k] = \delta[k] := \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \Rightarrow Y(z) = 1 \quad \forall z \in \mathbb{C} \quad (4.7)$$

Why does the ROC matter? Two quite different signals may have the same expression for their z -transforms, but distinct ROCs:

$$y_1[k] = a^k 1[k] \Rightarrow Y_1(z) = \frac{1}{1-(a/z)}, \quad \text{ROC: } |z| > 1 \quad (4.8)$$

$$y_2[k] = -a^k 1[-k-1] \Rightarrow Y_2(z) = \frac{1}{1-(a/z)}, \quad \text{ROC: } |z| < 1 \quad (4.9)$$

Given a z -transform $X(z)$ with a ROC, the sequence $x[n]$ can be determined from the inverse z -transform:

$$x[n] = \frac{1}{2\pi j} \oint X(z)z^{n-1}dz,$$

where \oint denotes complex integration around a closed curve in the complex plane that lies inside the ROC. The choice of the curve is immaterial.

4.1.1 Properties of Z -transform

Draw a picture of the signals $x[k-n_0]$ and compare with that of $x[k]$ to see that $x[k-n_0]$ is obtained by *delaying or advancing* the signal $x[n]$ by n_0 steps, depending on whether n_0 is positive or negative.

1. Linearity: $\mathcal{Z}(\alpha x[k] + \beta y[k]) = \alpha X(z) + \beta Y(z)$, where $\alpha, \beta \in \mathbb{C}$ and $X(z), Y(z)$ are the z -transforms of the signals $x[k], y[k]$, respectively.
2. Time shift: if $X(z)$ is the z -transform of $x[k]$, then the z -transform of the time-shifted signal $y[k] := x[k-n_0]$, where n_0 is a positive or negative integer, is

$$\mathcal{Z}(y[k]) = z^{-n_0} X(z). \quad (4.10)$$

Let's prove this statement:

$$\begin{aligned} \mathcal{Z}(y[k]) &= \sum_{k=-\infty}^{\infty} x[k-n_0]z^{-k} \\ &= \sum_{k=-\infty}^{\infty} x[k-n_0]z^{-(k-n_0)}z^{-n_0} = z^{-n_0} \sum_{\ell=-\infty}^{\infty} z^{-\ell}x[\ell] = z^{-n_0}X(z) \end{aligned}$$

One has to be careful in expressing a delayed one sided signal. Consider this: let $x[k] = a^k$ for $k \geq 0$ and 0 for all $k < 0$. Then $x[k-n_0] = a^{k-n_0} = a^{-n_0}a^k$ is simply a scaled version of the original signal! What is the problem here?

3. If $x[k] = \bar{x}[k]1[k]$ and $X(z)$ is the z -transform of $x[k]$, then the z -transform of the signal $y[k] := \bar{x}[k-n]1[k]$, where n is a positive integer, is

$$\mathcal{Z}(y[k]) = z^{-n}X(z) + \bar{x}[-1]z^{-n+1} + \bar{x}[-2]z^{-n+2} + \dots + \bar{x}[-(n-1)]z^{-1} + \bar{x}[-n]$$

To prove this is so, we note

$$\begin{aligned} \mathcal{Z}(y[k]) &= \sum_{k=-\infty}^{\infty} \bar{x}[k-n]1[k]z^{-k} = z^{-n} \sum_{k=0}^{\infty} z^{-(k-n)}\bar{x}[k-n] \\ &= z^{-n} \left[\bar{x}[-n]z^{-(-n)} + \bar{x}[1-n]z^{-(1-n)} + \dots + \bar{x}[-1]z + \bar{x}[0] + \bar{x}[1]z^{-1} + \bar{x}[2]z^{-2} + \dots \right] \\ &= z^{-n} \left[\bar{x}[-n]z^{-(-n)} + \bar{x}[1-n]z^{-(1-n)} + \dots + \bar{x}[-1]z \right] + z^{-n} \sum_{i=0}^{\infty} \bar{x}[i]z^{-i} \\ &= \bar{x}[-1]z^{-n+1} + \bar{x}[-2]z^{-n+2} + \dots + \bar{x}[-m]z^{-n+m} + \dots + \bar{x}[-(n-1)]z^{-1} \\ &\quad + \bar{x}[-n] + z^{-n}X(z), \end{aligned}$$

which is the desired result.

Note that $y[k]$ is not a delayed version of $x[k]$.

4. If $x[k] = \bar{x}[k]1[k]$ and $X(z)$ is the z -transform of $x[k]$, then the z -transform of the signal $y[k] := \bar{x}[k+n]1[k]$, where n is a positive integer, is

$$\mathcal{Z}(y[k]) = z^n X(z) - (\bar{x}[0]z^n + \bar{x}[1]z^{n-1} + \dots + \bar{x}[n-2]z^2 + \bar{x}[n-1]z)$$

which can be proved in a manner similar to that for the previous statement.

4.2 Discrete time transfer function

Suppose the input $\{u[k]\}$ and the output $\{y[k]\}$ of a causal system is governed by the following difference equation

$$y[k+2] + \beta_1 y[k+1] + \beta_0 y[k] = \alpha_0 u[k], \quad k \geq 0 \quad (4.11)$$

with the *initial conditions* $y[0] = y_0$, $y[1] = y_1$. The input signal $u[k]$ is zero for all $k < 0$. Furthermore, assume that the system is causal, so that $y[k]$ is 0 for all $k < 0$.

Take the z -transform of both sides of (4.11) to get

$$Y(z) = -\beta_1 (z^{-1}Y(z) - y[0]) - \beta_0 (z^{-2}Y(z) - y[0] - z^{-1}y[1]) + \alpha_0 U(z)$$

which can be rearranged into

$$Y(z) = \underbrace{\frac{\alpha_0}{1 + \beta_1 z^{-1} + \beta_0 z^{-2}}}_{H_d(z)} U(z) + \underbrace{\frac{(\beta_1 + \beta_0)y[0] + \beta_0 y[1]z^{-1}}{1 + \beta_1 z^{-1} + \beta_0 z^{-2}}}_{E(z)} \quad (4.12)$$

For the LTI system (4.1), carrying out the above procedure leads to a similar expression:

$$Y(z) = H_d(z)U(z) + E(z), \quad (4.13)$$

where the complex function

$$H_d(z) := \frac{\alpha_m z^m + \alpha_{m-1} z^{m-1} + \dots + \alpha_1 z + \alpha_0}{z^n + \beta_{n-1} z^{n-1} + \dots + \beta_1 z + \beta_0}. \quad (4.14)$$

is called the (*discrete time*) *transfer function* from u to y for the system (4.1), and $E(z)$ depends (as it does in (4.12)) on the initial conditions.

Upon taking the inverse z -transform of (4.13), we get

$$y[k] = \mathcal{Z}^{-1}(H_d(z)U(z)) + \mathcal{Z}^{-1}(E(z))$$

which shows that the output of the system is a superposition of the response due to the input and the response due to initial conditions.

If all the initial conditions are 0, then (4.13) reduces to

$$Y(z) = H_d(z)U(z) \quad (4.15)$$

Since $y[k]$ can be recovered by taking the inverse z -transform of the right hand side, it shows that when all initial conditions are zero, the transfer function together with the input completely determines the output.

The roots of the denominator polynomial of $H_d(z)$ are called the *poles* of the system and the roots of its numerator polynomial are called the *zeros* of the system.

Recall that a system is called BIBO stable if the output is bounded for every bounded input. The system is BIBO stable if and only if all of its poles have magnitude strictly less than unity. In that case, it can be shown that

$$\epsilon[k] := \mathcal{Z}^{-1}(E(z)) \quad (4.16)$$

is a signal such that $\epsilon[k] \rightarrow 0$ as $k \rightarrow \infty$. Therefore, for a BIBO stable system, the output at steady state (as $k \rightarrow \infty$) is given by

$$y_{ss}[k] = \mathcal{Z}^{-1}(H_d(z)U(z)) \quad (4.17)$$

Thus the transfer function provides information on steady state behavior of the system.

What does it mean to take z -transform of an equation?

In MATLAB, `tf(num, den, -1)` can be used to construct a discrete-time transfer function with numerator and denominator specified by `num` and `den`.

4.2.1 Unit pulse response and transfer function

The unit pulse input $\delta[k]$, called the *Kronecker delta function*, is defined as

$$\delta[k] = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \quad (4.18)$$

The output of a discrete-time system (with all initial conditions set to 0) when driven by an unit pulse input $\delta[k]$ is called the *unit pulse response* of the system, and is denoted by $h[k]$.

Compare the unit pulse response of a discrete time system with the impulse response of a continuous time system. The MATLAB command `impz` can also be used to compute the unit pulse response of a discrete time system.

It follows from (4.17) that the z -transform of the unit pulse response of a discrete time system with transfer function $H_d(z)$ is $H_d(z)\mathcal{Z}(\delta[k]) = H_d(z)$. The unit pulse response $h[k]$ is given by $h[k] = \mathcal{Z}^{-1}(H_d(z))$. Hence, the transfer function of a system is the z -transform of its unit pulse response.

A discrete time system is BIBO stable only if and only if its unit pulse response $h[k]$ is *absolutely summable*, meaning

$$\sum_{k=-\infty}^{\infty} |h[k]| < \infty. \quad (4.19)$$

An equivalent characterization of BIBO stability for a discrete time system is that each of its poles have magnitude strictly smaller than one.

4.3 Steady-state response to sinusoidal inputs

Suppose we apply a sinusoidal input with amplitude A and discrete-time angular frequency $\Omega \in [0, \pi]$, given by

$$u[k] = A \sin(\Omega k + \phi), \quad \forall k \in \{0, 1, 2, \dots\},$$

to the system (4.1) with transfer function (4.17). Assuming that the system is BIBO stable, the corresponding output is of the form

$$y[k] = \underbrace{g(\Omega)A \sin(\Omega k + \phi(\Omega))}_{\text{steady state}} + \underbrace{\epsilon[k]}_{\text{transient}} \quad \forall k \in \{0, 1, 2, \dots\}$$

where the gain g and phase ϕ are given by

$$g(\Omega) := |H_d(e^{j\Omega})| \quad \phi(\Omega) := \angle H_d(e^{j\Omega}), \quad (4.20)$$

and $\epsilon[k]$ is a *transient signal* that decays to 0 as $k \rightarrow \infty$.

The Bode plot of a discrete time transfer function shows

1. the magnitude $|H_d(e^{j\Omega})|$ in dB ($20 \log_{10} |H_d(e^{j\Omega})|$), and
2. the phase $\angle H_d(e^{j\Omega})$

as a function of frequency Ω . The Bode plot shows how much a sinusoidal signal of a particular frequency is amplified (in amplitude) and shifted (in phase) by the system.

Exercise A continuous time sinusoidal signal $u_c(t)$ with frequency 20 Hz is sampled at 10 kHz. Write the expression for the discrete time sampled signal $u_d[k]$.

4.4 Approximating $H_c(s)$ with $H_d(z)$

Suppose we model a system using the governing physical laws, which leads us to a continuous time ordinary differential equation. Upon linearization, we get a linear output-output model in terms of a linear differential equation, which gives us a transfer function $H_c(s)$ in the Laplace domain. When the system is used with a computer control system, we are interested in determining the corresponding transfer function $H_d(z)$ between the discrete input and output signals. (refer to Figure 4.1).

When the sampling time T_s is small, one can obtain $H_d(z)$ from $H_c(s)$ in a straightforward manner. We start by noting that a transfer function $H_c(s)$ can be realized by integrators, gains, and summation blocks (See Figure 4.2 for an example). Therefore, if we can determine the discrete time approximation of an integrator, we can replace the integrators ($1/s$ blocks in Figure 4.2, for example) with the discrete time approximation of integrators, and thus get a discrete time approximation of the transfer function.

Consider now the continuous time integrator

$$\dot{y}(t) = u(t) \quad \Rightarrow \quad y(t) = y(t_0) + \int_{t_0}^t u(\tau) d\tau \quad (4.21)$$

with transfer function

$$H_c(s) = \frac{Y(s)}{U(s)} = \frac{1}{s}. \quad (4.22)$$

The output over a single sample period T is given by

$$y(kT + T) = y(kT) + \int_{kT}^{kT+T} u(\tau) d\tau \quad (4.23)$$

Our objective is to find a discrete time approximation $H_d(z)$ to the integrator, such that $Y(z) = H_d(z)U_d(z)$.

4.4.1 Forward difference approximation

Upon taking the *forward difference* approximation of (4.23), we get

$$\frac{y(kT + T) - y(kT)}{T} = u(kT) \quad \Rightarrow \quad \frac{y_d[k + 1] - y_d[k]}{T} = u_d[k] \quad (4.24)$$

Upon taking the z -transform, we get

$$\frac{zY_d(z) - Y_d(z)}{T} = U_d(z) \quad \Leftrightarrow \quad H_d(z) = \frac{1}{\frac{z-1}{T}} \quad (4.25)$$

Therefore, the approximation is $\frac{1}{s} \mapsto \frac{T}{z-1}$. One can now go from a continuous-time transfer function $H_c(s)$ to its discrete-time approximation $H_d(z)$ by doing the following substitution:

$$s \mapsto \frac{z-1}{T} \quad \Leftrightarrow \quad z \mapsto 1 + sT,$$

which is called the *Euler* transformation.

The MATLAB command `c2d(sysc, T, 'zoh')` converts the continuous time system `sysc` to discrete time with Euler transformation, with sampling period specified by `T`.

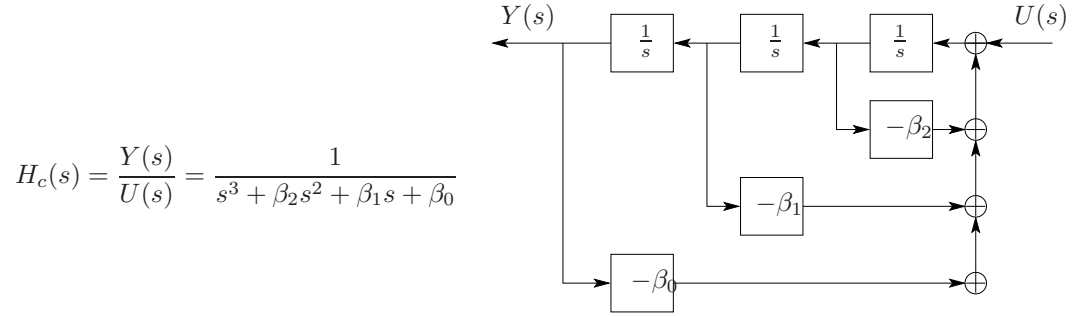


Figure 4.2: Realizing a transfer function $H_c(s)$ with integrators, gains, and summation blocks.

4.4.2 Trapezoidal approximation

A trapezoidal approximation to (4.23) leads to

$$\begin{aligned} y(kT + T) &= y(kT) + Tu(kT) + (u(kT + T) - u(kT)) \frac{T}{2} \\ \Rightarrow y_d[k + 1] &= y_d[k] + Tu_d[k] + \frac{T}{2}(u_d[k + 1] - u_d[k]) \\ \Rightarrow zY_d(z) &= Y_d(z) + TU_d(z) + \frac{T}{2}(zU_d(z) - U_d(z)) \end{aligned}$$

which leads to

$$H_d(z) = \frac{Y_d(z)}{U_d(z)} = \frac{T}{2} \frac{z + 1}{z - 1}.$$

So the approximation is $\frac{1}{s} \mapsto \frac{T}{2} \frac{z+1}{z-1}$. One can go from a continuous time transfer function $H_c(s)$ to its discrete time approximation by using the following transformation:

$$s \mapsto \frac{2}{T} \frac{z - 1}{z + 1} \qquad z \mapsto \frac{2 + sT}{2 - sT}$$

The MATLAB command `c2d(sysc, Ts, 'tustin')` converts the continuous time system `sysc` to discrete time with Tustin transformation.

which is also called the *Tustin* transformation.

4.4.3 Properties

Order: What is the relation between the order of the transfer function $H_c(s)$ and that of $H_d(z)$ obtained by one of these transformations?

If $H_c(s)$ is an n -th order transfer function, then $H_d(z)$ is also an n -th order transfer function with both Euler and Tustin transformations

Stability

If $H_c(s)$ is stable, is $H_d(z)$ stable?

1. The Euler transformation $s \mapsto \frac{z-1}{T_s}$ maps the left half s -plane onto the region shown in Figure 4.3. So, high frequency or lightly damped poles in $H_c(s)$ will produce unstable poles in $H_d(z)$.

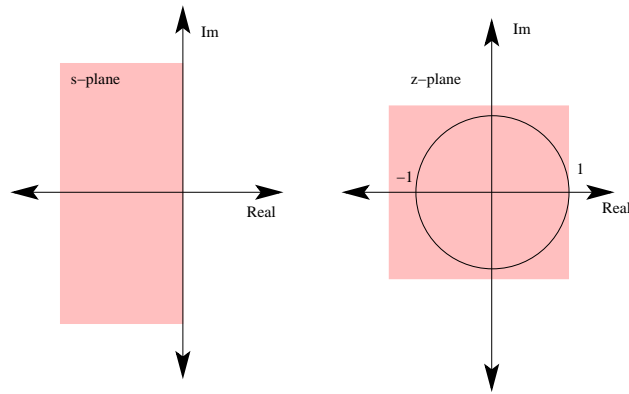


Figure 4.3: The Euler approximation maps the left half s -plane to the box-like area (extended vertically in both directions) in the z -plane that contains the unit circle.

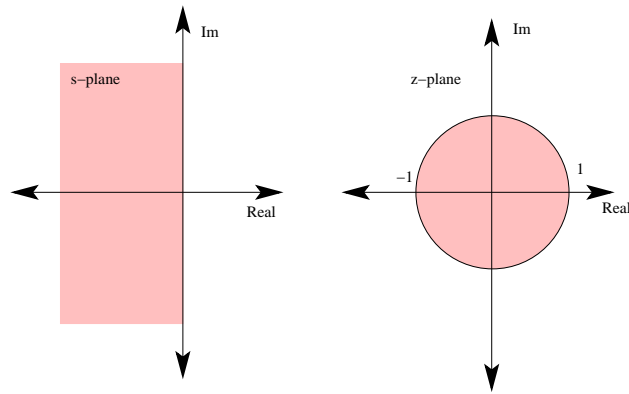


Figure 4.4: The Tustin transformation maps the left half s -plane to the area inside and on the unit circle in the z -plane.

2. The Tustin transformation $s \mapsto 2(z - 1)/(T(z + 1))$ maps the left half s -plane exactly into the unit-disk (see Figure 4.4). So, $H_c(s)$ stable $\Leftrightarrow H_d(z)$ is stable.

The second point above is the reason that the Tustin transformation is more commonly used.

4.5 Difference equation from transfer function

It is possible to come up with a difference equation describing the input-output behavior of a n LTI system from its discrete time transfer function. Let $N(z)$ and $D(z)$ be the numerator and denominator polynomials of a transfer function $H_d(z)$, given by

$$\begin{aligned} N(z) &:= \alpha_m z^m + \alpha_{m-1} z^{m-1} + \cdots + \alpha_1 z + \alpha_0 \\ D(z) &:= z^n + \beta_{n-1} z^{n-1} + \cdots + \beta_1 z + \beta_0 \end{aligned}$$

Then,

$$\begin{aligned}
 Y(z) &= H_d(z)U(z) = \frac{N(z)}{D(z)}U(z) & \Rightarrow & & D(z)Y(z) &= N(z)U(z) \\
 \Rightarrow (z^n + \beta_{n-1}z^{n-1} + \dots + \beta_1z + \beta_0)Y(z) &= (\alpha_mz^m + \alpha_{m-1}z^{m-1} + \dots + \alpha_1z + \alpha_0)U(z), \\
 \Rightarrow z^nY(z) + \beta_{n-1}z^{n-1}Y(z) + \dots + \beta_0Y(z) &= \alpha_mz^mU(z) + \alpha_{m-1}z^{m-1}U(z) + \dots + \alpha_0U(z).
 \end{aligned} \tag{4.26}$$

Noting that if $X(z)$ is the z transform of the signal $x[k]$, then $z^nX(z)$ is the z transform of the signal $x[k+n]$, we can determine the corresponding difference equation by inspection of (4.26):

$$\begin{aligned}
 y[k+n] + \beta_{n-1}y[k+n-1] + \dots + \beta_1y[k+1] + \beta_0y[k] &= \\
 \alpha_mu[k+m] + \alpha_{m-1}u[k+m-1] + \dots + \alpha_1u[k+1] + \alpha_0u[k].
 \end{aligned}$$

4.6 Parametric system identification revisited

To summarize, here's a typical path to parametric identification:

1. Use physics-based modeling – and linearization around an operating point if needed, – to determine a continuous time transfer function $H_c(s)$ from the input $u_c(t)$ to output $y_c(t)$.
2. Derive an approximation of the discrete time transfer function $H_d(s)$. Use of Tustin transformation is preferred. This is the transfer function between the discrete time input $u_d[k]$ and output $y_d[k]$ that are obtained from their continuous time counterparts by sampling, i.e., $u_d[k] = u_c(kT_s)$ and $y_d[k] = y_c(kT_s)$.
3. Derive a difference equation relating the inputs and outputs from $H_d(z)$, and convert it to an ARX model.
4. Apply the least squares parameter identification procedure described in Section 1.3 to identify the parameters of the ARX model, which are the coefficients of the numerator and the denominator polynomial in $H_d(z)$.
5. If desired, convert the identified $H_d(z)$ to $H_c(s)$ by using Tustin transformation.