Control Laboratory: **a.a. 2015/2016**

Lecture 2 — March 2, 2016

Instructor: Luca Schenato *Scribes: Daniele Alpago, Alessandro Scarso, Elena Zennaro*

2.1 Introduction

We are considering the system in Figure 2.1, which is described by the transfer function $P(s)$ (composed by plant, sensors and actuators).

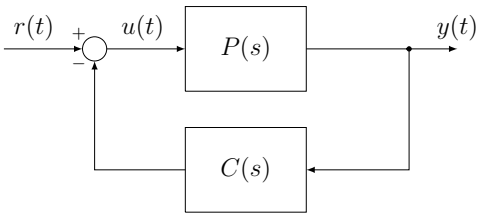


Figure 2.1.

The goal is to design a digital controller $C(s)$, satisfying the following requirements:

- **Stability**
- **Performance** concerning overshoot, rising time, settling time, steady state error
- **Robustness** in terms of noise rejection, model uncertainties, closed loop bandwidth

Starting from a given transfer function $P(s)$, we will design a continous time controller $C(s)$ and, by means of a proper discretization procedure, its discrete counterpart $C(z)$, as shown in Figure 2.2. There are two different approaches to reach this goal: *classic control*, which we have seen in Automatic Control class and *modern control*, presented in System Theory class.

2.1.1 Plant representations

$P(s)$ can be represented in several ways:

- **Transfer function:**

$$P(s) = \frac{b_m s^m + \cdots + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_0} \tag{2.1}$$

  A transfer function describing a system is usually called proper if $m \leq n$ or strictly proper if $m < n$.

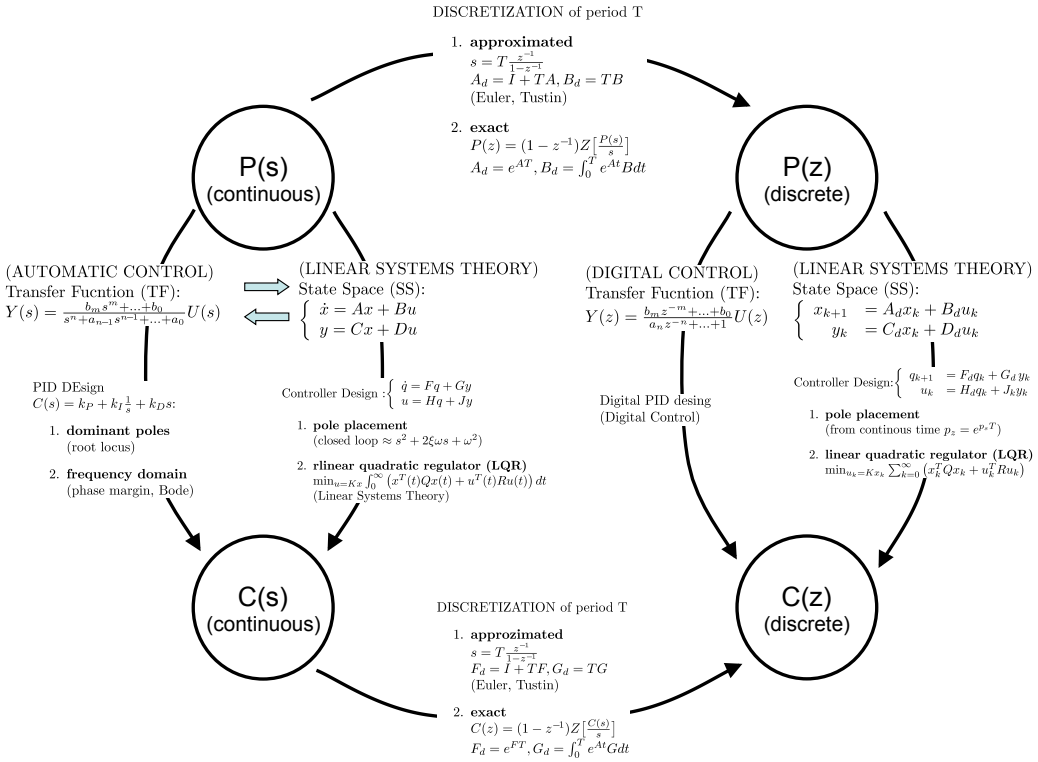


Figure 2.2. Design Procedure starting from a continuous-time plant to a digital controller

- **State-space representation:**

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{2.2}$$

If the system is strictly proper $D = 0$.

The two representations are related: the mapping from 2.2 to 2.1 is given by:

$$P(s) = C(sI - A)^{-1}B + D \tag{2.3}$$

On the other hand, we can get a state space representation from a transfer function by using some tools called *canonical forms*, *e.g.*, *control based*, *observed based*, *Jordan forms* and *balanced realizations*.

2.2 From P(s) to C(s)

2.2.1 Classic control

The most common continuous time controller in classic control theory is represented by PID. It is obtained by the linear combination of three different actions: proportional (P), integrative (I) and derivative (D).

$$C_{PID}(s) = K_P + \frac{K_I}{s} + sK_D \tag{2.4}$$

A major issue related to the physical feasibility is that C_{PID} is written as a non proper transfer function, and it has thus to be be modified to become physically achievable. For this reason, we can place a remote pole into the derivative term to make it proper. Moreover, the transfer function of C_{PID} can be rewritten by highlighting some temporal constants:

$$C_{PID} = K_P \left(1 + \frac{1}{sT_I} + T_D s\right) \tag{2.5}$$

where $T_I = \frac{K_P}{K_I}$ and $T_D = \frac{K_D}{K_P}$ are the temporal constant of the integral and derivative actions, respectively. Simpler controllers can be implemented, such as **I**, for $K_P = K_D = 0$, **P**, for $K_I = K_D = 0$, **PD**, for $K_I = 0$.

Generally, there are two main approaches to design a PID: root locus and frequency domain design. The former is generally not very effective, as it barely allows to meet predefined performance requirements (it is thus used only for ustable systems). The latter is based on phase margin ϕ_{PM} and crossing frequency ω_c.

2.2.2 Modern control

Since our controller corresponds to a dynamical system, it can be represented by state space model. Therefore, from a transfer function point of view, $C(s)$ can be computed as:

$$C(s) = H(sI - F)^{-1}G + J \tag{2.6}$$

The controller design problem can be turned into the computation of the F, G, H and J matrices. This can be achieved by following two approaches:

- *Pole placement:* we define the position of the close loop poles under some given assumptions.
- *Optimal control:* it consists on choosing a controller minimizing a specific control cost

$$J(u, x_0) = \min_{u(t)} \int_0^m x^T(t)Qx(t) + u^T(t)Ru(t)dt. \tag{2.7}$$

  where Q and R are suitable semidefinite weight matrices.

2.3 From P(z) to C(z)

The relation between $P(z)$ and $C(z)$ can be derived using similar methods to ones just described for continuos time systems. In Digital Control class $P(z)$ was represented by a transfer function like:

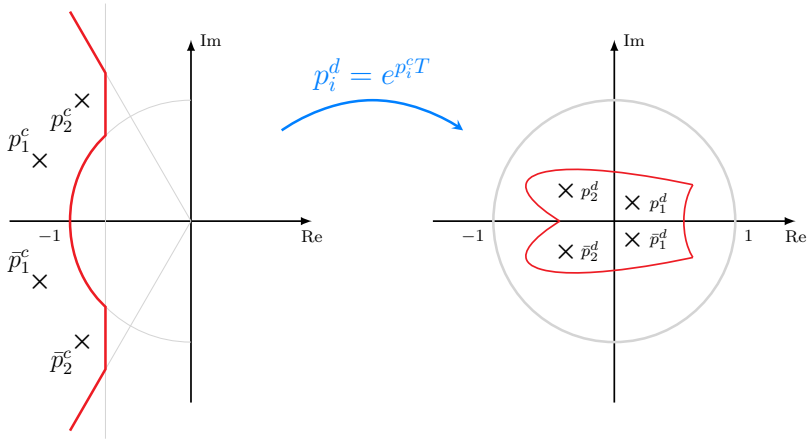
$$P(z) = \frac{b_m z^{-m} + \cdots + b_0}{z^{-n} + a_{n-1} z^{-n+1} + \cdots + a_0} \tag{2.8}$$

We are going to focus mostly on the state space representation since the approach of starting by a certain transfer function actually appears less effective in discrete time. Therefore, *modern control* starts from a state space representation, like this:

$$\begin{cases} x_{k+1} &= A_d x_k + B_d u_k \\ y_k &= C_d x_k + D_d u_k \end{cases} \tag{2.9}$$

Also for discrete time systems, we can use two ways to obtain $C(z)$ given $P(z)$:

- *Pole placement:* it refers to the same situation we have previously met in continuos time: tools that are used are exactly the same. We might wonder about the region the pole should be placed in. This can be easily obtained by mapping the correct region in continuous time to its discrete counterpart, using the exponential map $p_i^d = e^{p_i^c T}$.



- *LQ Optimal Control:* also in discrete time, the aim is to minimize a control cost

$$J(u, x_0) = \min_{u_k} \sum_{h=0}^{\infty} x_h^T Q_d x_h + u_h^T R_d u_h. \tag{2.10}$$

The optimal control is given by the control law $u_k = K_{LQ} x_k$, where K_{LQ} is the matrix obtained from the solution of the Riccati difference equation.

2.4 Discretization

Making reference at Figure 2.2, the following step is how to operate the discretization, *i.e.* how to obtain a discrete time controller starting from a continuos one. There are several approaches to reach this purpose and we will highlight the most interesting ones.

2.4.1 Exact approximation

Exact approximation tries to find the equivalent discrete time system that would have exactly the same output of the continuos time system at the sampling interval (assuming that the input is piecewise constant with the same period). Using the modern control approach, we can start from a continuos time system like in (2.11):

$$\begin{cases} \dot{z}(t) &= Fz(t) + Gy(t) \\ u(t) &= Hz(t) + Jy(t) \end{cases} \tag{2.11}$$

and we can obtain the discrete time system in (2.12) by sampling with period T:

$$\begin{cases} z_{k+1} &= F_d z_k + G_d y_k \\ u_k &= H_d z_k + J_d y_k \end{cases} \tag{2.12}$$

The relations between the matrices in continuos time and in discrete time are given by:

$$F_d = e^{fT}, \qquad G_d = \int_0^T e^{fT} G dt \tag{2.13}$$

$$H_d = H, \qquad J_d = J. \tag{2.14}$$

Conversely, in the automatic control approach, $C(s)$ is a polynomial rational function of variable s^{-1} and $C(z)$ is obtained by "sampling and hold" with sampling period equal to T:

$$C(z) = (1 - z^{-1}) \mathcal{Z} \left[S_T \left[\mathcal{L}^{-1} \left[\frac{C(s)}{s} \right] \right] \right] \tag{2.15}$$

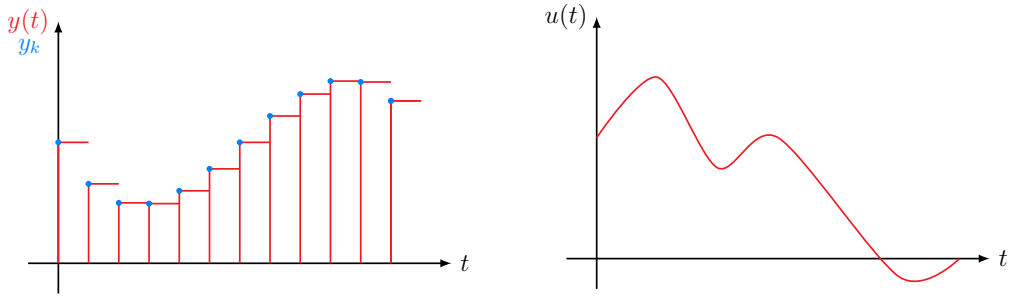
2.4.2 Approximate discretization

In general, in the state space approach, the exponential function can be approximated using the first order Taylor expansion in 0 which returns the following expressions:

$$F_d = I + FT, \qquad G_d = GT \tag{2.16}$$

$$H_d = H, \qquad J_d = J \tag{2.17}$$

On the other hand, in automatic control we resort to a number of possible approximations, such as:



(a) Sampling and hold operation (b) An input signal

- *Forward Euler approximation:*

$$\frac{dx(t)}{dt} \simeq \frac{x(t+T) - x(t)}{T} \tag{2.18}$$

with T sampling period. In the $\mathcal{Z}$-transform domain it can be approximated with:

$$s \simeq \frac{z-1}{T} \tag{2.19}$$

corresponding to the approximation $z = e^{sT} \simeq 1 + sT$.

- *Backwards Euler approximation:*

$$\frac{dx(t)}{dt} \simeq \frac{x(t) - x(t-T)}{T} \tag{2.20}$$

and in the $\mathcal{Z}$-transform domain it becomes

$$s \simeq \frac{1 - z^{-1}}{T} \tag{2.21}$$

corresponding to the approximation $z = e^{sT} \simeq \frac{1}{1-sT}$.

- *Tustin approximation:* with this kind of method $z = e^{sT}$ can be approximated by

$$z = e^{sT} \simeq \frac{1 + \frac{sT}{2}}{1 - \frac{sT}{2}} \tag{2.22}$$

giving $s = \frac{2}{T}\frac{z-1}{z+1}$. It is interesting to notice that Tustin approximation works better than the others and most importantly it preserves stability of the discretized system for any value of the sampling period T.

Generally, approximated discretizations are *mechanical ways* assigning to the complex variable s an equivalent variable the in discrete time domain. Approximations work better when the sampling period T sufficiently smaller than the system's dynamics.

2.5 Basic notions about linear dynamical systems

Systems considered in this class are *Linear Dynamical Systems*. They are a particular set of systems satisfying some properties and which can be represented in different ways. There

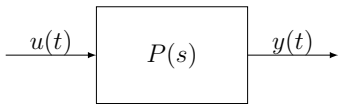


Figure 2.3.

are four important representations:

1. **ODES (Ordinary Differential Equations)**

$$y^{(n)}(t) + a_{n-1}y^{(n-1)}(t) + \cdots + a_0 y(t) = b_m u^{(m)}(t) + b_{m-1}u^{(m-1)}(t) + \cdots + b_0 u(t) \quad (2.23)$$

2. **Transfer function** (in particular we speak about rational transfer function)

$$P(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_0}{s^n + a_{n-1}s^{n-1} + ... + a_0}, \qquad m \leq n. \quad (2.24)$$

3. **Impulse response**

$$p(t) = \mathcal{L}^{-1}[P(s)] \quad (2.25)$$

   the resulting impulse responce will be a linear combination of system's modes.

4. **State-space representation**

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \quad (2.26)$$

It is also important to remember how it is possible to derive a representation from another one. The ones we want to remember are:

- **From 1. to 2.** By $\mathcal{L}$-transform properties and the so called *known transform*, $\mathcal{L}\left(\frac{df(t)}{dt}\right) = s\mathcal{L}((f(t) - f(0^-))$, we have:

$$\mathcal{L}\left(\frac{d^i y(t)}{dt^i}\right) = s^i Y(s) - \sum_{k=0}^{i-1} y^{(k)}(0^-)s^{i-1-k}, \qquad i = 1, 2, 3, \ldots \quad (2.27)$$

$$\mathcal{L}\left(\frac{d^i u(t)}{dt^i}\right) = s^i U(s) \quad (2.28)$$

Applying (2.27) and (2.28) to the ODE, we obtain:

$$(a_n s^n + a_{n-1}s^{n-1} + \cdots + a_0)Y(s) - a_n y(0^-)s^{n-1} - (a_{n-1}y(0^-) + a_n y^{(1)}(0^-))s^{n-2} + \ldots - \left(\sum_{k=0}^{i-1} a_{k+1}y^{(k)}(0^-)\right) = (b_m s^m + b_{m-1}s^{m-1} + \cdots + b_0)U(s) \tag{2.29}$$

Now, if we define:

$$d(s) := a_n s^n + a_{n-1}s^{n-1} + \cdots + a_0 \tag{2.30}$$

$$p(s) := a_n y(0^-)s^{n-1} + (a_{n-1}y(0^-) + a_n y^{(1)}(0^-))s^{n-2} + \cdots + \left(\sum_{k=0}^{i-1} a_{k+1}y^{(k)}(0^-)\right) \tag{2.31}$$

$$n(s) := b_m s^m + b_{m-1}s^{m-1} + \cdots + b_0 \tag{2.32}$$

Finally we can rewrite (2.29) as:

$$d(s)Y(s) - p(s) = n(s)U(s) \tag{2.33}$$

and we can obtain the transfer function as the ratio between $Y(s)$ and $U(s)$:

$$P(s) = \frac{n(s)}{d(s)} = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_0}{a_n s^n + a_{n-1}s^{n-1} + \cdots + a_0} \tag{2.34}$$

- **From 3. to 2.** $p(t) \xrightarrow{\mathcal{L}} P(s) = \int_0^{+\infty} p(t)e^{-st}\,dt$
- **From 4. to 2.** Given the matrices describing the system in a state space model, A, B, C, D, it is possible to obtain the transfer function by:

$$P(s) = C(sI - A)^{-1}B + D \tag{2.35}$$

In this class we will consider only two of the aforementioned representations: transfer function and state space representations.

2.5.1 Transfer function representation

A transfer function describing a dynamic linear system consists of the ratio between two polynomials. If we consider $P(s)$ as the transfer function of a system, we have:

$$P(s) = \frac{n(s)}{d(s)}, \qquad n(s), d(s) \text{ polynomials.} \tag{2.36}$$

A transfer function which is physically achievable must be proper, then $\deg[n(s)] \leq \deg[d(s)]$, or strictly proper, so $\deg[n(s)] < \deg[d(s)]$. As regards the structure of the denominator of the transfer function. We can define $d(s)$ as:

$$d(s) := s^n + a_{n-1}s^{n-1} + \cdots + a_0. \tag{2.37}$$

Since $d(s)$ is a polynomial of degree n, it has n roots $\mathbb{C}$. If we consider $d(s)$ as a product of n factors, we can represent this polynomial as:

$$d(s) = \prod_{i=1}^{n}(s - p_i) \tag{2.38}$$

where we identify p_i as the i-th pole of the system. System poles have a very important role in system dynamic because, they determine the trend of the system response. Indeed, the system modes are associated to the roots of d(s) and they can be divided principally into two categories:

- $e^{p_i t}$ **if** $p_i \in \mathbb{R}$
- $e^{\sigma t}\cos(\omega t)$, $e^{\sigma t}\sin(\omega t)$ **if** p_i, $\bar{p_i} \in \mathbb{C}$, $p_i = \sigma + j\omega$

In detail, the system modes associated to a root of d(s) with degree k are:

$$e^{p_i t},\ t\,e^{p_i t}, \ldots,\ \frac{t^{k-1}}{(k-1)!}\,e^{p_i t}.$$

2.5.2 Modes classification

System modes can be easily classified according to their behavior. If we define $m(t) = \frac{t^k}{k!}e^{p_i t}$ as the generic system mode, it can be:

- **convergent(to zero)** if $\lim_{t\to+\infty} m(t) = 0$;
- **limited** if $\exists\ M > 0 :\ |m(t)| < M\ \forall t \geq 0$;
- **divergent** if $m(t)$ is not limited.

In continuos time systems, if convergent modes are related to modes whose poles p_i: $\mathrm{Re}[p_i] < 0$. Therefore, the concept of convergent modes is related to the *asymptotic stability* notion.

2.5.3 Natural response

In (2.33) we found the equation that expresses the system output as a sum of two terms: the $\mathcal{L}$-transform of the natural response and the $\mathcal{L}$-transform of the forced response. In particular, we can rewrite (2.33) as:

$$Y(s) = P(s)U(s) + Y_0(s) \tag{2.39}$$

where, relying on the notation introduced in the previous section, $Y_0(s) = \frac{p(s)}{d(s)}$, $\deg[p(s)] < \deg[d(s)]$. Now, if we consider $p(s) = c_{n-1}s^{n-1} + c_{n-2}s^{n-2} + \cdots + c_0$ and apply the inverse $\mathcal{L}$-transform to the term $Y_0(s)$ we obtain the natural response:

$$y_0(t) = \mathcal{L}^{-1}\left[Y_0(s)\right] = \sum_{i=1}^{n} \alpha_i f_i(t) e^{p_i t} \tag{2.40}$$

where α_i (and also the coefficients c_l, $l = 0, 1, \ldots, n-1$) depend on the initial conditions of the system, and $f_i(t) = t^{h_i}$ are monomial whose exponent $h_i \in \mathbb{N}$ depends on the multiplicity of the associated root p_i. For example, if all roots are distinct, then $h_i = 0, \forall i$, i.e. $f_i(t) = 1$.

2.6 Forced response to particular inputs

In the expression 2.33 $U(s) = \mathcal{L}[u(t)]$ where $u(\cdot)$ can be a specific signal. Some typical signals $u(\cdot)$ that we will encounter in pratice are:

$$\text{Dirac's delta:} \quad u(t) = \delta(t) \xrightarrow{\mathcal{L}} U(s) = 1$$

$$\text{Step function:} \quad u(t) = \mathbb{1}(t) \xrightarrow{\mathcal{L}} U(s) = \frac{1}{s}$$

$$\text{Linear ramp:} \quad u(t) = t\,\mathbb{1}(t) \xrightarrow{\mathcal{L}} U(s) = \frac{1}{s^2}$$

In particular, when $u(t) = \delta(t)$ the Laplace transform of the forced response becomes:

$$Y_f(s) = \frac{n(s)}{d(s)} U(s) = \frac{n(s)}{d(s)} \cdot 1 = \frac{n(s)}{d(s)}$$

and the corresponding expression in the time domain is:

$$y_f(t) = d_0\delta(t) + \sum_{i=1}^{r} \sum_{k=0}^{\mu_i - 1} d_{i,k} \frac{t^k}{k!} e^{p_i t} \mathbb{1}(t), \tag{2.41}$$

where p_i are the zeros of the polynomial $d(s)$ and μ_i are the relative moltiplicities, $i = 1, 2, \ldots, n$. $y_f(t)$ is called *impulse response* of the system (for some coefficients d_0, $d_{i,k} \in \mathbb{R}$). Otherwise, if $u(t) = \mathbb{1}(t)$, we have:

$$Y_f(s) = \frac{n(s)}{d(s)} U(s) = \frac{n(s)}{d(s)} \cdot \frac{1}{s}$$

and, by applying the Laplace anti-transform, we obtain the *step response* of the system:

$$y_f(t) = \mathcal{L}^{-1}[Y_f(s)] = \mathcal{L}^{-1}\left[\frac{n(s)}{d(s)} \cdot \frac{1}{s}\right]. \tag{2.42}$$

If p_i is a simple zero of $d(s)$ with $\mathrm{Re}[p_i] < 0$ for all $i = 1, 2, \ldots, n$, we have, more precisely:

$$Y_f(s) = \frac{n(s)}{d(s)} \cdot \frac{1}{s} = \frac{\alpha_0}{s} + \frac{\bar{n}(s)}{d(s)}, \qquad \deg[\bar{n}] < n = \deg[d],$$

and then the step response is:

$$y_f(t) = \mathcal{L}^{-1}[Y_f(s)] = \bar{\alpha}_0 \mathbb{1}(t) + \sum_{i=1}^{n} \bar{\alpha}_i e^{p_i} t.$$