Automatic Control

Edoardo Demichelis

March 2025

Contents

Chapter 1

LTI systems

1.1 Representation and solution of LTI systems

1.2 Stability of LTI systems

Stability in a system describes its ability to maintain bounded behavior over time in response to initial conditions or external inputs.

1.2.1 Definition of stability

An LTI system is said to be **internally stable** if the zero input state response $x_{zi}(t)$ is bounded for any initial state x_0.

$$\forall x_0, \quad \exists M \in \mathcal{R} : |x_{zi}(t)| < M \quad \forall t \geq 0$$

A stronger condition can be imposed on the zero input state response: an LTI system is said to be **asymptotically stable** if the zero input state response $x_{zi}(t)$ tends to zero for any initial state x_0.

$$\forall x_0, \quad \lim_{t\to\infty} |x_{zi}(t)| = 0$$

Proposition 1.2.1. *If a system is asymptotically stable then it is internally stable* □

Of course, a system can be proved to be **unstable** if at least one initial condition causes the zero state response to diverge.

$$\exists x_0 : \lim_{t\to\infty} |x_{zi}(t)| = \infty$$

1.2.2 Natural modes of a system

The direct application of either definition of stability is too complex to be a standard procedure, but it can lead to characterizations that are easier to prove. This passage will not be covered in depth but it consists in solving a general LTI system in the Laplace domain by partial fraction expansion. The final $X_{zi}(s)$ will be a linear combination of terms of the kind $\frac{1}{(s-\lambda_i)^k}$. The inverse Laplace transform of such terms will make the steady state response in time domain a linear combination of terms of the kind

$$t^{k-1}e^{\lambda_i t}, \quad t^{k-1}e^{Re(\lambda_i t)}\cos(\dots)$$

These terms are functions of time known as **natural modes** of the system. The direct application of the definitions of stability on the natural mode leads us to the following:

Theorem 1.2.1 (Characterization of internal stability)**.** *A linear time invariant system of system matrix A is internally stable if and only if all eigenvalues of*

A have real part lesser or equal than zero and all purely imaginary eigenvalues have minimal multiplicity equal to one. That is

$$\text{LTI of matrix } A \text{ is stable} \iff$$

$$Re[\lambda_i(A)] \leq 0 \wedge \mu'(\lambda_i(A)) = 1 \quad \forall \lambda_i : Re[\lambda_i(A)] = 0$$

□

1.3 Control systems design via static state feedback

Let's consider an LTI continuous time system described by the following steady state representation: $\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases}$, $x \in \mathcal{R}^n, u, y \in \mathcal{R}^p$

The dynamics of the LTI system depend on the eigenvalues of matrix A, so we need to find an input that can affect the structure of A. If we consider the input :

$$u(t) = -Kx(t) + Nr(t), K \in \mathcal{R}^{p,n}, r(t) \in \mathcal{R}^p, N \in \mathcal{R}^{p,p}$$

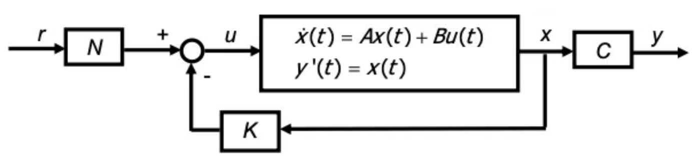
$$\implies \dot{x}(t) = Ax(t) + Bu(t) = (A - BK)x(t) + BNr(t)$$

and the dynamical properties of the controlled system depend on the eigenvalues of the matrix $A - BK$.
The matrix K is defined as the **state gain**. A suitable choice of K allows us to modify the system eigenvalues and improve the dynamic properties of the system, thus stabilizing it.

Theorem 1.3.1 (Eigenvalue Assignment)**.** *Given n arbitrary, real or complex-conjugate eigenvalues for $A - BK$, the matrix $K \in \mathcal{R}^{p,n}$ exists if and only if $\rho(M_R) = \rho([B\ AB\ \dots\ A^{n-1}B]) = n$.* □

The matrix $M_R = [BAB...A^{n-1}B]$ is the **reachability matrix**. If $\rho(M_R) = n$ the dynamic system $\dot{x}(t) = Ax(t) + Bu(t)$ is defined as **reachable**.

Implementing the Eigenvalues theorem in Matlab The input $u = -Kx + Nr$ represents a static state feedback control law



- Typically, K is computed by placing the eigenvalues (poles) of the controlled system to obtain, other than asymptotic stability, also a good damping coefficient and fast transient response.

```
lambda_des = [-1 -2];
K = place(A, B, lambda_des);

% OR
lambda_des = [-1 -1];
K = acker(A, B, lambda_des);
```

definition of K

- N can be choosen to fix the dc-gain of the system to an arbitrary value, usually unitary, guaranteeing zero steady state tracking error in the presence of a constant reference signal $r(t) = \bar{r}\epsilon(t)$.

```
    sys_cont_c = ss(A-BK, B, C, 0);
    N = 1 / dcgain(sys_cont_c)
```

definition of N

1.3.1 Requirement analysis

Requirement analysis is carried out by means of two important values: s and α.

- $s = \frac{Y_{max} - Y_{ss}}{Y_{ss}}$ represents the **normalized overshoot**, indicating the relative peak deviation of the system's response from its steady-state value

- α defines the settling bound belt, ensuring that the system's response remains within α of the steady-state value for all times $t > t_s$, where t_s is the settling time

It is possible to directly choose the eigenvalues based on the system's performance requirements by converting them into damping ratio ζ and natural frequency ω_n:

$$s = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}$$

$$t_s = \frac{4}{\zeta\omega_n}$$

Once ζ and ω_n are determined, the corresponding pair of complex conjugate eigenvalues can be calculated as:

$$\lambda_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$$

Alternatively, to avoid oscillations in the output step response, a prototype second-order model with coincident real poles can be used to determine the eigenvalues. The control law is defined as:

$$u = -Kx + Nr$$

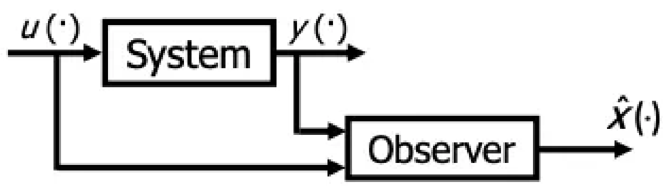
where K is designed to assign $n = \dim(x)$ eigenvalues. For $n > 2$, additional eigenvalues are selected with a faster time constant τ_{add} compared to the prototype poles, ensuring minimal influence on the transient response. Note that it's important to have $\tau_{add} \ll \frac{1}{\zeta\omega_n}$. A good rule of thumb is $\tau_{add} = (0.1 \,\text{to}\, 0.2) \cdot \frac{1}{\zeta\omega_n}$

1.4 State observers

Let's consider an LTI continuous time system described by the state space representation: $\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases}$

A state feedback control law of the form $u = -Kx + Nr$ can be evaluated only when the system state can measured. However in general, only the measurement of the system output is available. Under suitable assumptions, an estimate of the system state can be obtained based on the knowledge of the provided input and the measurement of the output signal.

A **state observer** is a device that provides an estimate $\hat{x}$ of the sistem state x exploiting the knowledge of the system input u and the measurement of the output y.



For a continuous time dynamic system, we define the state estimation error $e(t) = \hat{x}(t) - x(t)$. An observer such that $\lim_{t\to\infty} |e(t)| = \lim_{t\to\infty} |\hat{x}(t) - x(t)| = 0$ is referred to as **asymptotic state observer**.

To obtain an asymptotic observer, a term depending on the output estimation error $\hat{y} - y$ is included:

$$\dot{e}(t) = (A - LC)e(t)$$

The newly introduced term adds an additional degree of freedom and a suitable choice of L allows us to modify the observer eigenvalues and improve its dynamic properties.

Theorem 1.4.1. *Given an LTI system in its state space equation, there exists an L such that the n eigenvalues of $A - LC$ can be arbitrarily assigned if and only if* $\rho(M_O) = \rho(\begin{pmatrix} C \\ CA \\ \dots \\ CA^{n-1} \end{pmatrix}) = n$ $\square$

The matrix $\begin{pmatrix} C \\ CA \\ \dots \\ CA^{n-1} \end{pmatrix}$ is the **observability matrix**. If $\rho(M_O) = n$, the dynamic system is said to be **observable**.

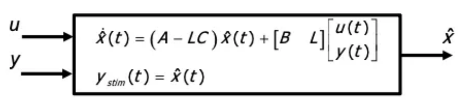
```
Mo = obsv(A, C);
rho_Mo = rank(Mo)
```

Definition of observability matrix and its rank in Matlab

Now let's put all the pieces together: the state equation of the state observer is

$$\dot{\hat{x}} = (A - LC)\hat{x}(t) + Bu(t) + Ly(t)$$

.
The matrix L is referred to as the **observer gain** and has dimension $\dim(L) = n \times q$. The observer is thus a dynamical system with inputs $u(t)$ and $y(t)$ and output $\hat{x}(t)$:



L can be computed through the Matlab place and acker statements:

```
lambda_obsv_des = [-1 -2];
L = place(A', C', lambda_obsv_des)'

% OR

lambda_obsv_des = [-1 -1];
L = acker(A', C', lambda_obsv_des)'
```

computation of L