Control Laboratory: **a.a. 2015-16**

Lezione 18 — April, 20

Instructor: Luca Schenato *Scribes: M. Scattolin, D. Marino, R. Campagnolo*

18.1 Discretization of continuous systems

One way to use a discrete time controller to control a continuous time system is to design the continuous time controller $C(s)$ and to discretize it into the discrete time one $C(z)$. The scheme in figure 18.1 shows the correspondence between the output signal $u(t)$ of the continuous time controller $C(s)$ and the output signal $\tilde{u}(t)$ of the discrete time controller $C(z)$, after having passed it through a zero order hold H_0.

The signals $u(t)$ and $\tilde{u}(t)$ are never the same.

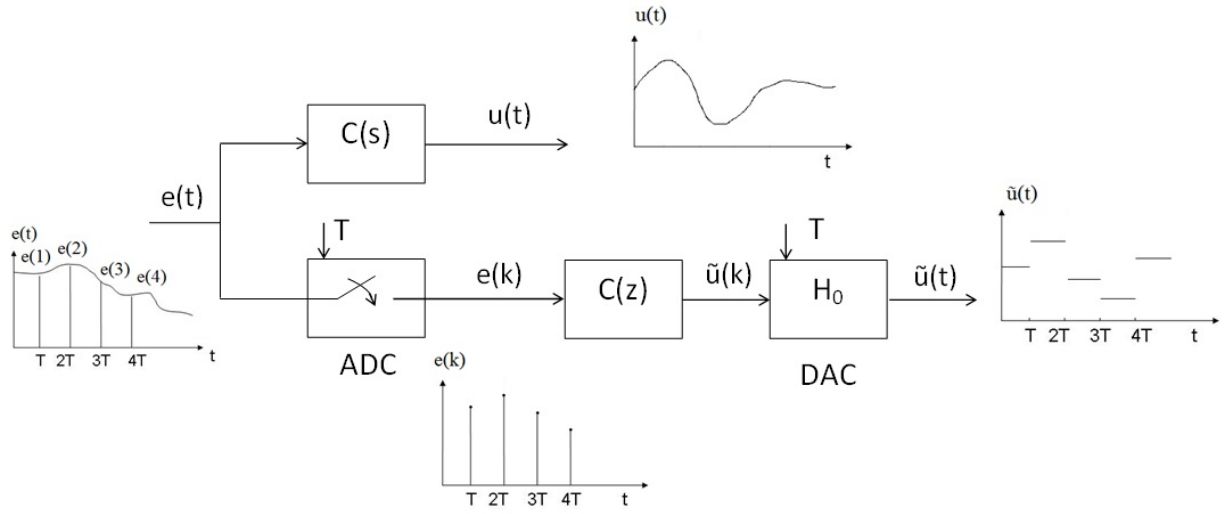

Figure 18.1. Scheme of a continuous time controller and the correspondant discrete time controller.

For SISO system, the controller is identified, as showed in figure 18.2, by the transfer function:

$$C(s) = \frac{b_0 s^{n-1} + ... + b_{n-1}}{s^n + a_0 s^{n-1} + ... + a_{n-1}}$$

This form can be also translated into the matrices A, B, C and D for the state space representation.

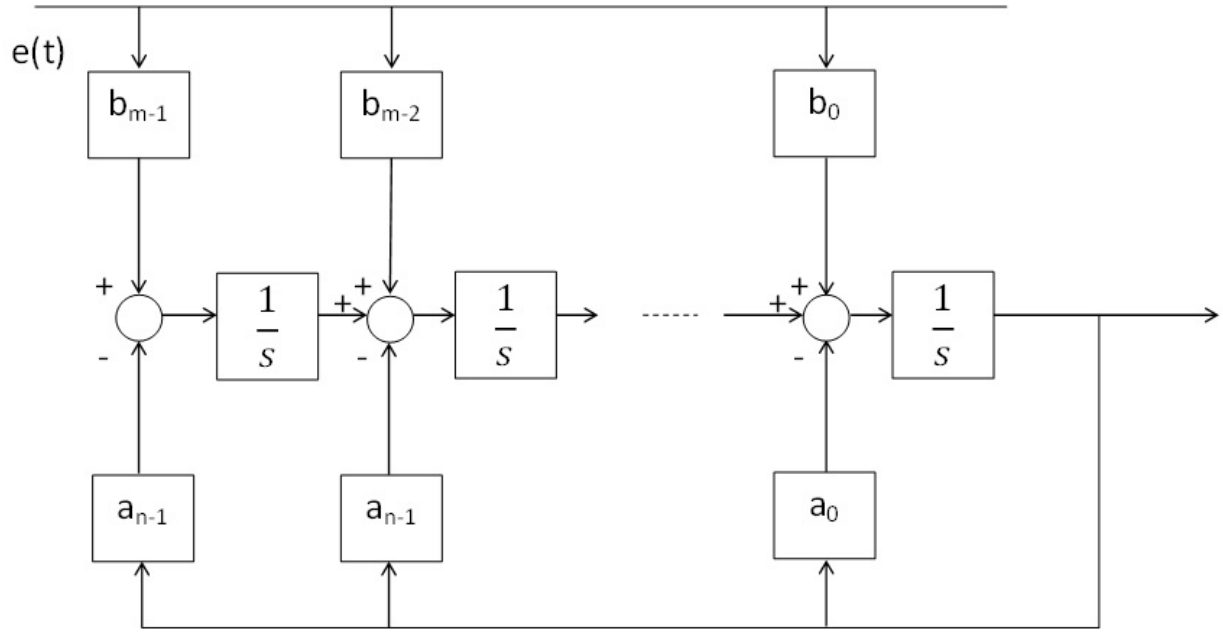

Figure 18.2. Scheme of the transfer function of a continuous controller $C(s)$

We want to substitute this structure with something else such that the output $\tilde{u}(t)$ at the points kT, $k \in \mathbb{N}$, is the *as close as possible* to the continuous output $u(t)$.
So we want to replace the integrators, which take a continuous input $u(t)$ and give a continuous output $y(t) = \int_0^t u(\tau)d\tau$, with blocks which return a discrete output $\tilde{y}(k)$, such that $\tilde{y}(k) \simeq y(kT) = y(k)$.

To do that, different methods can be used: three kind of approximations to pass from continuous to discrete time are described below.

18.1.1 Forward Euler discretization

We sample the signal $y(t)$ and we compute the area between two consecutive instants kT and $(k+1)T$.

We have an integral which can be split in two, in this way:

$$y((k+1)T) = \int_0^{(k+1)T} u(\tau)d\tau = \int_0^{kT} u(\tau)d\tau + \int_{kT}^{(k+1)T} u(\tau)d\tau \tag{18.1}$$

where the first integral of the two is the signal $y(kT) = y(k)$, that is to say the continuous signal $y(t)$ sampled with period T (figure 18.4).

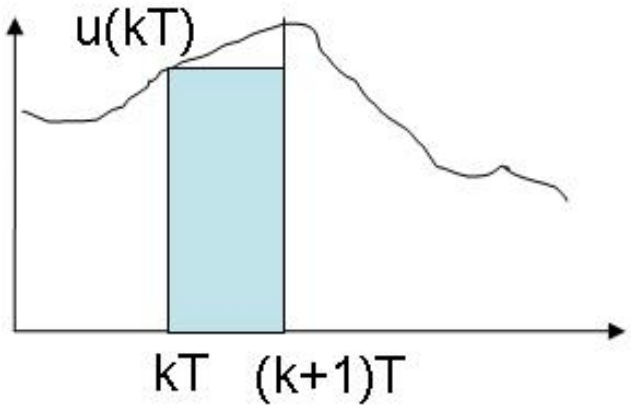

Figure 18.3. Plot of $u(t)$

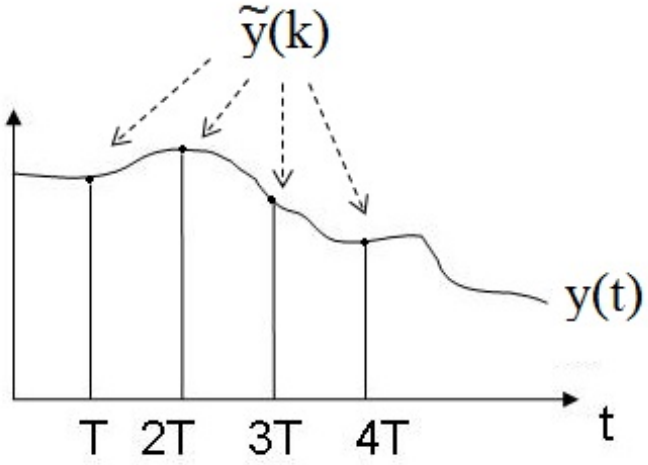

Figure 18.4. Sampled signal $y(t)$.

The second integral instead is the area of the plot of $u(t)$ between $u(kT)$ and $u((k+1)T)$, and it can be approximated with the area of a rectangle with base of amplitude T and height of amplitude $u(kT) = u(k)$, that is the blue area in figure 18.3. Using this approximation, the equation 18.1.1 can be rewritten like this:

$$\tilde{y}(k+1) = \tilde{y}(k) + Tu(k) \tag{18.2}$$

The error in this approximation is:

$$y((k+1)T) - \tilde{y}(k+1) = \int_{kT}^{(k+1)T} u(\tau)d\tau - Tu(k)$$

Now we can apply the $\mathcal{Z}$ - Transform to the equation 18.1.1:

$$z\tilde{Y}(z) = \tilde{Y}(z) + TU(z)$$

from which we can extract $\tilde{Y}(z)$:

$$\tilde{Y}(z) = \frac{T}{z-1}U(z)$$

So we can substitute the integrators, which have transfer function $\frac{1}{s}$, with this expression:

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

which implies this change of variables:

$$s = \frac{1}{T}(z-1)$$

This approximation is called *Forward Euler Discretization* and there is a MATLAB command to apply it.

- *Example*

$$C(s) = \frac{s}{s+1} \tag{18.3}$$

$$C_{FE}(z) = \frac{\frac{1}{T}(z-1)}{\frac{1}{T}(z-1)+1} = \frac{z-1}{z-1+T} = \frac{1-z^{-1}}{1+(T-1)z^{-1}} \tag{18.4}$$

18.1.2 Backward Euler discretization

An another method to approximate the integral is to apply the previous method to the following sample, so:

$$\tilde{y}(k+1) = \tilde{y}(k) + Tu(k+1) \tag{18.5}$$

and applying the $\mathcal{Z}$ - Transform as we did before we obtain:

$$z\tilde{Y}(z) = \tilde{Y}(z) + TzU(z)$$

so:

$$\tilde{Y}(z) = \frac{T}{1-z^{-1}}U(z)$$

As a result we have that the discretized transfer function of the integrators is:

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

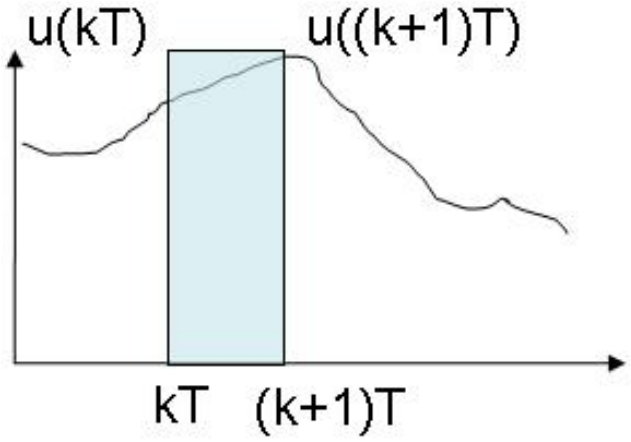

Figure 18.5. Plot of $u(t)$

and finally the change of variables is:

$$s = \frac{1}{T}(1 - z^{-1})$$

This approximation is called *Backward Euler Discretization* and there is a MATLAB command to implement it.

18.1.3 Tustin discretization

A better way to implement the discretization is the Tustin's method, which connects linearly two samples.

In this case we have:

$$\tilde{y}(k+1) = \tilde{y}(k) + T\frac{u(k) + u(k+1)}{2} \tag{18.6}$$

We apply the $\mathcal{Z}$ - Transform to the previous equation obtaining:

$$z\tilde{Y}(z) = \tilde{Y}(z) + T\frac{1+z}{2}U(z)$$

and we can extract $\tilde{Y}(z)$:

$$\tilde{Y}(z) = \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}U(z)$$

So for the integrators we obtain this change of variables:

$$s = \frac{2}{T}\frac{z-1}{z+1}$$

This approximation is called *Tustin Discretization* and there is a MATLAB command to implement it.

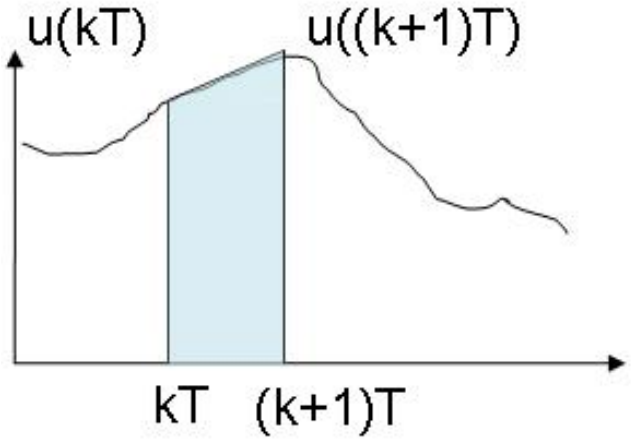

Figure 18.6. Plot of $u(t)$

18.2 Stability preservation

Important considerations have to be done for the stability preservation. First of all, we have to keep in mind that with small T the continuous and the discretized systems are basically equivalent, assuming that T is not small enough so that any numerical problems of the hardware will arise. For larger T, we have poor performance or even instability of the closed loop system. Another problem is that the mapping of the poles does not always preserve stability. Let us recall every method we have discussed so far:

- *Forward Euler*

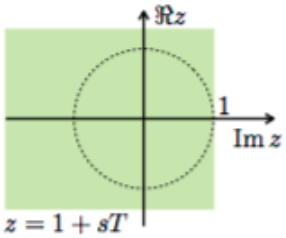

Figure 18.7. Forward Euler method

$$z = 1 + sT$$

So as we can see, this method is the worst because it does not always preserve stability.

- *Backward Euler*

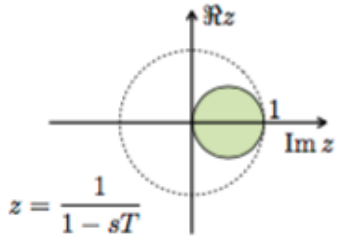

Figure 18.8. Backward Euler method

$$z = \frac{1}{1 - sT}$$

This method maps the poles only into a smaller fraction of the unity circle.

- *Tustin*

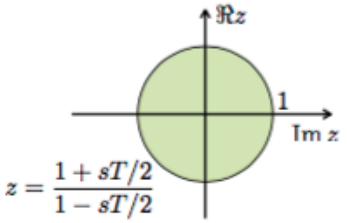

Figure 18.9. Tustin's method

$$z = \frac{1 + s\frac{T}{2}}{1 - s\frac{T}{2}}$$

This is the better method, because it maps the poles into the totality of the unitary circle.

18.3 Discretization Of State Space Systems

Consider the continuous time system in state space

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{18.7}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{l\times n}$, $D \in \mathbb{R}^{l\times p}$, $x \in \mathbb{R}^{n\times 1}$, $u \in \mathbb{R}^{p\times 1}$ and $y \in \mathbb{R}^{l\times 1}$.
We would like to approximate this system with the discrete time system in state space

$$\begin{cases} x_{k+1} = A_d x_k + B_d u_k \\ \tilde{y}_k = C_d x_k + D_d u_k \end{cases} \tag{18.8}$$

where $A_d \in \mathbb{R}^{n\times n}$, $B_d \in \mathbb{R}^{n\times p}$, $C_d \in \mathbb{R}^{l\times n}$, $D_d \in \mathbb{R}^{l\times p}$, $x_k \in \mathbb{R}^{n\times 1}$, $u_k \in \mathbb{R}^{p\times 1}$ and $y_k \in \mathbb{R}^{l\times 1}$, so that $y_k := y(kT) \approx \tilde{y}_k$. Notice that $\tilde{y}_k$ might not be equal to y_k because, as mentioned above there are infinite input signals $u(t)$ which have the values at the sampling points kT, therefore in general they give rise to different output signals $y(t)$. As so, in is impossible to find an equivalent discrete time system such that $\tilde{y}_k = y(kT)$ for *any* input signal $u(t)$. Therefore, the best we can hope is to find an approximation of the time derivative of $x(t)$. We can approximate $\dot{x}(t) := \lim_{\epsilon\to 0} \frac{x(t+\epsilon)-x(t)}{\epsilon}$ as:

$$\dot{x}(t) \approx \frac{x(t+T) - x(t)}{T} \tag{18.9}$$

for small enough values of T. Since we are interested in the approximation only when $t = kT$, defined $x(kT) := x_k$, we substitute $t = kT$ into the previous expression getting:

$$\begin{aligned} \dot{x}(kT) \approx \left.\frac{x(t+T) - x(t)}{T}\right|_{t=kT} = \frac{x(kT+T) - x(kT)}{T} = \frac{x((k+1)T) - x(kT)}{T} = \\ = \frac{x_{k+1} - x_k}{T} \end{aligned} \tag{18.10}$$

Moreover, defining $u_k := u(kT)$ and $y_k := y(kT)$ and substituting (18.10) into the sampled verision of the continuous time state space system (18.7), we have:

$$\begin{aligned} &\begin{cases} \frac{x_{k+1}-x_k}{T} = Ax_k + Bu_k \\ y_k = Cx_k + Du_k \end{cases} \iff \begin{cases} x_{k+1} - x_k = T(Ax_k + Bu_k) \\ y_k = Cx_k + Du_k \end{cases} \iff \\ &\begin{cases} x_{k+1} = x_k + T(Ax_k + Bu_k) \\ y_k = Cx_k + Du_k \end{cases} \iff \begin{cases} x_{k+1} = (I + TA)x_k + TBu_k \\ y_k = Cx_k + Du_k \end{cases} \end{aligned} \tag{18.11}$$

From this last expression we see that to approximate the continuous time system (18.7) with the discrete time system (18.8) we have to apply in (18.8) the following substitutions:

$$\begin{cases} A_d \leftarrow I + TA \\ B_d \leftarrow TB \\ C_d \leftarrow C \\ D_d \leftarrow D \end{cases} \tag{18.12}$$

The approximation associated with these substitutions is known as *Forward Euler Approximation* for state space models.

Other approximations of $\dot{x}(t)$ are possible and give different expressions for the matrixes A_d, B_d, C_d and D_d. They are known as *Backward Euler Approximation* and *Tustin Approximation*. We will not derive explicitly the substitutions related to these two approximations but will sum them up in a table in the next lecture.

We introduce, instead, another approximation of the continuous time system in state space (18.7) which is exact as long as we assume that the input $u(t)$ of the continuous time system is piecewise constant in every time interval of length T (where T is the sampling time), that is under the additional assumption $u(t) = u_k$ for $kT \leq t \leq (k+1)T, k \in \mathbb{N}$. An illustration of a possible input signal satisfying this condition is illustrated in figure 18.10.

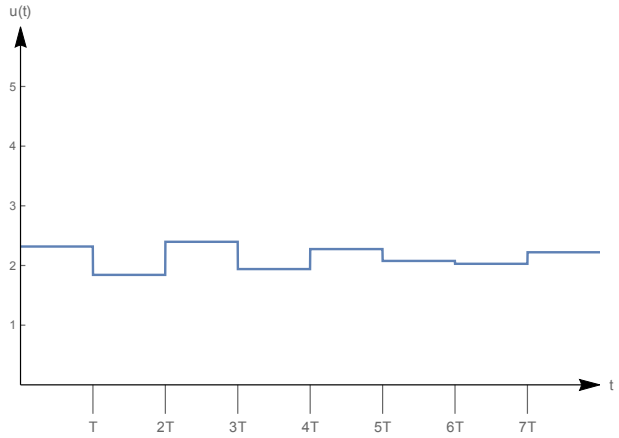

Figure 18.10. Piecewise constant input signal

In such a case we can find four matrixes A_d, B_d, C_d, D_d such that the continuous time system output y is exactly $\tilde{y}_k$ at the sampling instants $t = kT$, that is such that $y(kT) = \tilde{y}_k$. To this end we start from the closed form expression for the state dynamic $x(t)$ given, in $[0, t]$, by

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \tag{18.13}$$

and once defined, as before, $x_k := x(kT)$ and $u_k := u(kT)$, we see that in $[kT, (k+1)T]$

$$\begin{aligned} x_{k+1} = x((k+1)T) &= e^{At}x(kT) + \int_{kT}^{(k+1)T} \left[e^{A((k+1)T-\tau)}Bu(\tau)d\tau\right] = \\ &= e^{At}x(kT) + \int_0^T \left[e^{A\tau}Bu((k+1)T-\tau)d\tau\right] = \\ &= e^{At}x(kT) + \int_0^T \left[e^{A\tau}Bu(kT)d\tau\right] = \\ &= e^{At}x(kT) + \int_0^T \left[e^{A\tau}Bd\tau\right]u(kT) = \\ &= e^{At}x_k + \int_0^T \left[e^{A\tau}Bd\tau\right]u_k = \end{aligned} \tag{18.14}$$

Notice that to derive this expression we have used the fact that $u((k+1)T-\tau)$ is constant and equal to $u(kT)$ for $\tau \in [0, T]$. Moreover we see that the first equation in (18.7) coincides with equation (18.14) if we take $A_d := e^{At}$ and $B_d := \int_0^T e^{A\tau}Bd\tau$ while the sampled version of the second equation in (18.7) is equal to the second equation in (18.8) simply taking $C_d := C$ and $D_d := D$. This kind of approximation is known as exact discretization.

With exact discretization we can guarantee to have exactly the same outputs at the sampling instants (see figure (18.11)) and the substitutions to apply in (18.8) to get this approximation are: $\begin{cases} A_d \leftarrow e^{At} \\ B_d \leftarrow \int_0^T e^{A\tau}Bd\tau \\ C_d \leftarrow C \\ D_d \leftarrow D \end{cases}$

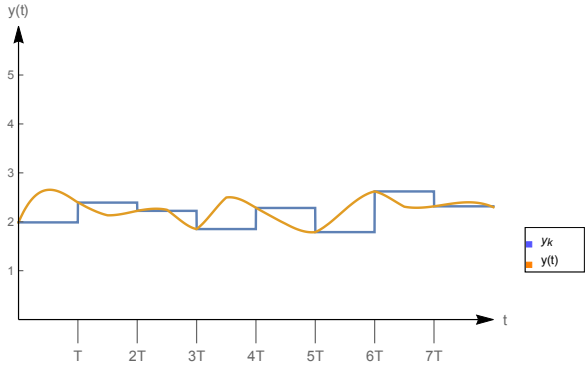

Figure 18.11. Output signal $y(t)$ and y_k