Networked Control Systems, Introduction

Manel Velasco and Pau Martí

October 2, 2016

1 *Subject Introduction*

The main reason of this course is to learn about the way to control networked systems. The network may be considered as a tool or as a part of the system itself, but it has clear advantages in front of a non networked system, namely:

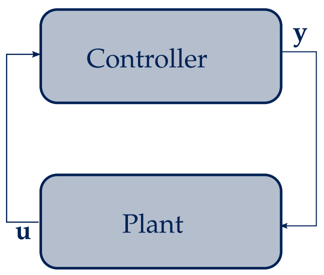
- Cost reduction
- Ease installation and mantinance
- Large flexibility
- Deployment in harsh environments
- Less wires
- Optimal size in production (inverters)
- ...

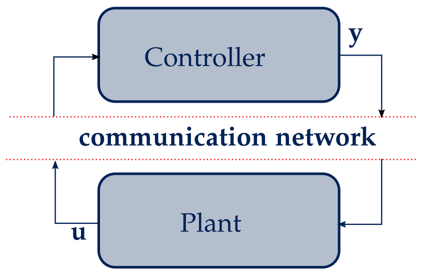
We may consider two possible situations for a networked control system:

- Phisically dostributed
- Logicaly distributed

In the case of a phisically distributed system we face the problem of controlling a system over the network, o the plant is located at one phisical place and the controller[1] is located in a different place while the network actuates as interface between both systems.

[1] One single controller with all the information





This configuration produces a set of problems that must be addressed froma a control point of view, among some of them:

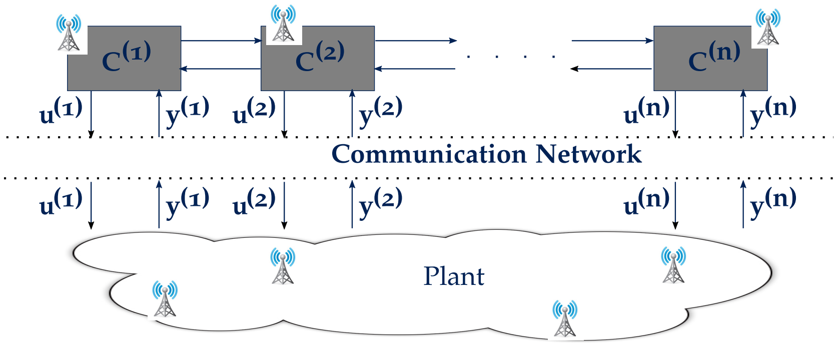
- Varing sampling/transmission interval

- Varing communications delays
- Packet loss
- Communication constrains through shared networks
- Quantization

All these proplems [2] must be addressed from an analytical point of view to garant a correct behairbour of the controller and the plant

[2] And some others not listed here

In the case of a logically distributed system we don't have a clear notion of single controller nor single plant. The sistem becomes a set of controllers, which may or not comunicate among thema, and a set of plant which may or not interact. This is the most hard problem to solve due to the fact that is suffers the problems of the phisically distributed system plus those associated whith the logically distributed control system.



Networked Control Systems

The course is finite and the among of theory is huge, we will face the problems of a phisically distributed system and then we will try to face those associated with the logical distributed systems.

2 *Simple Network models, from continuous to discrete models*

We start with the simple continuous linear time invariant system[3]

[3] You should be able to understand tthese equations, if you are not familiar with them try to read "Modern Control Enginyering" from Katsuhiko Ogata

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y &= Cx(t) + Du(t) \end{aligned} \tag{1}$$

Usually we drop the dependency on t from x and u because we know that these are the time dependent variables and we simply write

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2}$$

Where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$,$C \in \mathbb{R}^{1\times n}$ and $D \in \mathbb{R}^{1\times 1}$. x is the state of the sistem and is defined on $\mathbb{R}^{n\times 1}$ and u is the input to de system and is suposed to be sacalr along all the time.

From a practical control point of view, this model is useless, just because nowadays the control is performed inside a processor, which has a very simple way of actuating:

- Measure the system's output (y) or the state if possible (x)
- Compute the control signal u
- Apply the control signal up to the next sampling

As it can be seen, the way in which the controller works is discrete, so the way to design the controller should take into account this fact. To this end we present an easy way to undertand the discrete time model, it's based on the general solution of the model 2 The solution to the state of equation 2 is

$$\begin{aligned} x(t) &= e^{At}x_0 + \int_0^t e^{As}Bu(t)ds \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{3}$$

If we use equation 3 we are in conditions to compute the state in any time t once the initial condition x_0 and the control signal over time $u(t)$ are given.

For the shake of simplicity we rename the matrices e^{At} and $\int_0^t e^{As}Bds$ as $\Phi(t)$ and $\Gamma(t)$.

Example 2.1. Supose you have a system[4] given by the matrices

[4] We just need the A and B matrics of equation (2) to perform the calculations, even more, the system is supossed to be LTI contiunuous time, so we need no more information right now

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

And a initial condition $x(0) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and a input signal $u(t) = t$

Solution: To compute the solution of system (2) we first compute the expression e^{At} in Matlab

---

Matlab code:

```
close all % to close all plots
clear all % to clean workspace
A=[0 1;0 0]; % Declare A matrix
syms t;      %symbolic variable t
```

Result:

```
Phi =
[ 1, t]
[ 0, 1]
```

```
Phi=expm(A*t)
```

Now we compute the expression $\int_0^t e^{As}Bu(t)ds$

Matlab code:

```
syms s; % symbolic variable s to
        % compute the integral
B=[0;1];
Gamma=int(expm(A*s)*B*s,s,0,t)
```

Result:

```
Gamma =
[t^3/3]
[t^2/2]
```

Now we can say that the system evolves with

$$x(t) = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} \frac{t^3}{3} \\ \frac{t^2}{2} \end{bmatrix} = \begin{bmatrix} 1 + 2t + \frac{t^3}{3} \\ 2 + \frac{t^2}{3} \end{bmatrix}$$

The previous exaple allows us to compute the evolution of the system under any initial condition when $u(t)$ is known, but if we focus in the control algorithm we see that the last step is to aply the control signal up to the next sampling. This means that the control signal is constant over time, adding this fact to our solution equation (3) becomes

$$\begin{aligned} x(t) &= e^{At}x_0 + \int_0^t e^{As}Bds\, u \\ y(t) &= Cx(t) + Du \end{aligned} \tag{4}$$

Now we may set a chain of solutions, in such a way that initial condition for step 2 is the final state for tep 1, initial condition for step 3 is the final state for step 2 and so on. The only requirement is that the sampling is constant every h seconds. This may be written as

$$\begin{aligned} x_{k+1} &= e^{Ah}x_k + \int_0^h e^{As}Bds\, u_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{5}$$

Be aware, now e^{Ah} and $\int_0^h e^{As}Bdt$ are renamed as $\Phi(h)$ and $\Gamma(h)$.[5]. This leads to our final discrete time model

[5] These are matrices defined over the same space of its continuous counterparts. The computation of those is not difficult but is out of the scope of these notes. In matlab you can compute these matrices as [Phi,Gamma]=c2d(A,B,h)

$$\begin{aligned} x_{k+1} &= \Phi(h)x_k + \Gamma(h)u_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{6}$$

which is the basis of the design of discrete time controllers.

3 *Easy controller design*

The easiest way to design a controller is to solve the system of equation that one may get comparing the desired characteristic polynomial with the actual characteristic polynomial in closed loop form.

The closed loop characteristic polynomial is defined[6] as

$$|zI - (\Phi - \Gamma K)|$$

[6] The sign in this expression is completely arbitrary, for historical reasons is defined as minus, that forces you to impose a sign beforehand. Take care in matlab, because is defined in this way

The desired polynomial is

$$p_d(z) = (z - z_1)(z - z_2)\dots(z - z_n)$$

where z_i are the desired poles of the discrete system in closed loop.

Whenever we want the system with feedback K to have those z_i poles we would equal both polynomials and we would solve the generated system of equations where the unknowns are k_i.

You should take into account that usually the desired dynamics are expressed in terms of continuous control[7]. To get the discrete verssion of these continuous poles you have to discretize them using $z_i = e^{s_i h}$, where h is the sampling period.

[7] It may be the continuous poles s_i or the restrictions over the time response of the system, as shown in next examples

Example 3.1. Lets use the dobule integrator model

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x \end{aligned}$$

And we want to design a controller that sets the continuous eigenvalues of the system to $s_{1,2} = -1.8 \pm j3.12$ with a sampling period of $h = 0.1\text{s}$

Solution: The discrete equivalent poles are $z_{1,2} = e^{s_{1,2}h} = 0.8 \pm j0.25$

The polynomial that has as solution these poles is

$$z^2 - 1.6z + 0.7 = 0$$

This will be the one against we compare our closed loop expression. To get this closed loop expression we discretize the system

$$
\begin{aligned}
x_{k+1} &= \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} u_k \\
y_k &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_k
\end{aligned}
$$

thus the closed loop characteristic polynomial is

$$
|z\mathbf{I} - (\Phi - \Gamma K)| = \left| z \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} h^2/2 \\ h \end{bmatrix} \begin{bmatrix} k_1 & k_2 \end{bmatrix} \right| = 0
$$

or

$$
z^2 + (hk_2 + (h^2/2)k_1 - 2)z + (h^2/2)k_1 - hk_2 + 1 = 0
$$

Identifying coefficient we get

$$
\begin{cases} hk_2 + (h^2/2)k_1 - 2 = -1.6 \\ (h^2/2)k_1 - hk_2 + 1 = 0.7 \end{cases}
$$

and solving

$$
K = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} \frac{0.1}{h^2} & \frac{0.35}{h} \end{bmatrix}
$$

Setting $h = 0.1$,

$$
K = \begin{bmatrix} 10 & 3.5 \end{bmatrix}
$$

As it can be seen this is a easy method to place the poles at a given position. There are some formulas to perform this automatically, the Ackermann's formula and the Banach's formula are just few of them. The nice thing about the Ackermann's formula is that it uses explicitly the inverse of the controllability matrix, so that, if the system isn't controllable there is no inverse.

The direct path for controller design, Ackermann's formula

Given the system

$$
\begin{aligned}
x_{k+1} &= \Phi x_k + \Gamma u_k \\
y_k &= C x_k + D u_k
\end{aligned}
$$

And the desired characteristic polynomial

$$
p_d(z) = (z - \lambda_1)(z - \lambda_2) \cdots (z - \lambda_n) = z^n + b_1 z^{n-1} + \cdots + b_{n-1} z + b_n
$$

The Ackermann's formula states that

$$K = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \Gamma & \Phi\Gamma & \Phi^2\Gamma & \cdots & \Phi^{n-1}\Gamma \end{bmatrix}^{-1} p_d\left(\Phi\right)$$

Where

$$p_d\left(\Phi\right) = \Phi^n + b_1\Phi^{n-1} + \cdots + b_{n-1}\Phi + b_n$$

Example 3.2. Taking data from previous example, the desired characteristic polynomial is

$$p_d(z) = (z - 0.8 - j0.25)(z - 0.8 + j0.25) = z^2 - 1.6z + 0.7025$$

And the discrete matrices

$$\Phi = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \text{ and } \Gamma = \begin{bmatrix} 0.005 \\ 0.1 \end{bmatrix}$$

we have:

$$K = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \Gamma & \Phi\Gamma \end{bmatrix}^{-1} p_d\left(\Phi\right) \quad = \left(\begin{bmatrix} 10 & 3.5 \end{bmatrix}\right)$$

Matlab code:

```
A=[0 1;0 0];                   %system matrix
B=[0;1];                       %Input matrix
h=0.1;                         %sampling period
[phi,gamma]=c2d(A,B,h);        %Discrete model
z=[0.8+0.25i, 0.8-0.25i];      %Discrete poles
k=acker(phi,gamma,z)           %Controler
```

Result:

```
k =
[10  3.5]
```