MANEL VELASCO AND PAU MARTÍ

INTRODUCTION TO NETWORKED CONTROL SYSTEMS

ESAII

1
Introduction

If you are reading this preliminary notes on networked control systems i because you have realised that the network is becoming pervasive, everywhere... just like the control systems are pervasive in our lives.

Introducing a network in a control system has its own problems, and its advantages. 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 maintenance
- Large flexibility
- Deployment in harsh environments
- Less wires
- Optimal size in production (inverters)
- ...

It also has its own problems, from a control point of view:

- Varing sampling/transmission interval
- Varing communications delays
- Packet loss
- Communication constrains through shared networks
- Quantization

All these proplems [1] must be addressed from an analytical point of view to grant a correct behaviour of the controller and the plant

[1] And some others not listed here

Before we start we are going to classify the nature of the control systems depending on how the network integrates in the control system. We may consider two possible frameworks for a networked control system:

- Physically distributed
- Logically distributed

In the case of a physically distributed system we face the problem of controlling a system over the network, so in this case the network is not part of the system itself, but the controller gets the information trough the network, and apples the control signals through the network. The plant is located at one physical place and the controller[2] is located in a different place while the network actuates as interface between both systems.

[2] One single controller with all the information

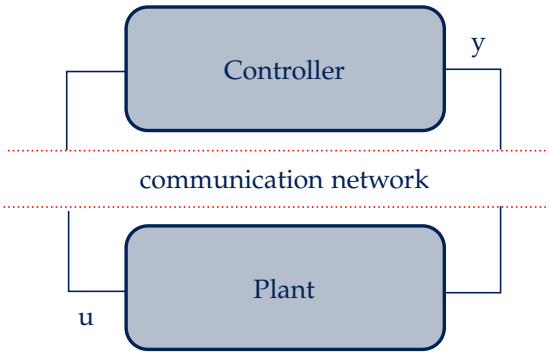


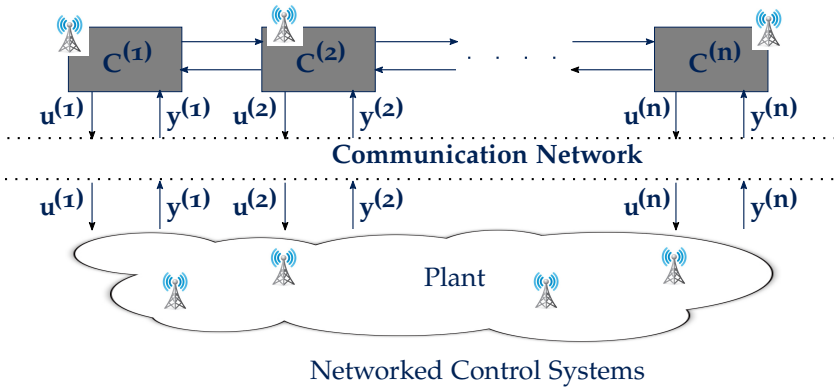
Figure 1.1: Networked control system abstraction

In the case of a logically distributed system we don't have a clear notion of single controller nor single plant. The system becomes a set of controllers, which may or not communicate among them, 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 physically distributed system plus those associated with the logically distributed control system.

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

To start we review the discrete time model associated to a continuous system. This is an important issue that you must take into account, the system is continuous, the controller is discrete, it only makes actions in a finite set of times. We develop a discrete time model from the continuous one only to be able to design the controller. Once the controler is designed the discrete model plays no roll in the control loop, nor in the controller actions. It's just for design purposes that we develop the next[3] discrete model.

[3] and oll the models developed in these notes



Networked Control Systems

1.1 *Basic model development , from continuous to discrete models*

We start with the simple continuous linear time invariant system[4]

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y &= Cx(t) + Du(t) \end{aligned} \tag{1.1}$$

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

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{1.2}$$

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 system and is defined on $\mathbb{R}^{n\times 1}$ and u is the input to de system and is supposed 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 1.2 The solution to the state of equation 1.2 is

$$\begin{aligned} x(t) &= e^{At}x_0 + \int_0^t e^{As}Bu(t)ds \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{1.3}$$

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

If we use equation 1.3 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 1.1.1. Supose you have a system[5] given by the matrices

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

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

[5] We just need the A and B matrics of equation (1.2) to perform the calculations, even more, the system is supossed to be LTI contiunuous time, so we need no more information right now

Solution: To compute the solution of system (1.2) 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
Phi=expm(A*t)
```

Result:

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

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 (1.3) becomes

$$\begin{aligned} x(t) &= e^{At}x_0 + \int_0^t e^{As}Bds\, u \\ y(t) &= Cx(t) + Du \end{aligned} \tag{1.4}$$

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

$$\begin{aligned} x_{k+1} &= e^{Ah}x_k + \int_0^h e^{As}Bds\, u_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{1.5}$$

Be aware, now e^{Ah} and $\int_0^h e^{As}Bdt$ are renamed as $\Phi(h)$ and $\Gamma(h)$.[6]. This leads to our final discrete time model

[6] 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)

$$\begin{aligned} x_{k+1} &= \Phi(h)x_k + \Gamma(h)u_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{1.6}$$

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

1.2 *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[7] as

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

[7] 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[8]. 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.

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

---

Example 1.2.1. Lets use the dobule integrator model

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

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 $h = 0.1$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

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

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

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

Where

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

Example 1.2.2. 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:

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

Matlab code:

Result:

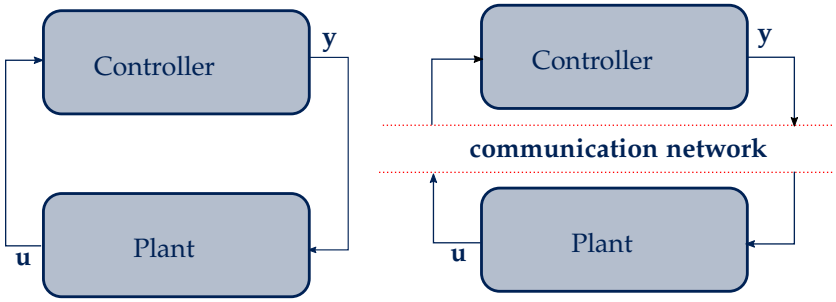
```
k =
[10   3.5]
```

```
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
```

2

Basic model for a networked control system

Whenever we work with a networked control system we experience some delays induced by the network. The nature of these delays may be random and unpredictable. This fact adds a lot of difficulty to the analysis of this kind of systems, we propose to use a simplified path to analyze them.



- Analyze the effects of the network in a time line
- Fix the delay to be regular and the sampling period to be constant
- Construct a model that contains these simplified effects

  Afterward we will increase the difficulty of the analysis.

2.1 Time delays induced by the network

Figure (2.1) sketches a simplified version of what may happen in a network. In this example the control loop is isolated in the network, meaning that nobody else is going to interrupt nor delay the messages on the network.

Under these conditions the sequence of events in the control loop is described as:

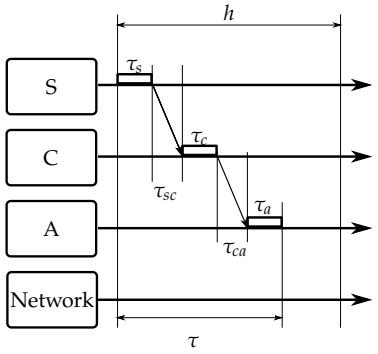


Figure 2.1: Networked system timeline analysis

- A time interrupt happens in the sensor processor, so the sensor starts the measuring, this instant of time may be labeled as kh. Where h is the sampling interval and this is occurring at the k^{th} sample.
- The sensor takes some time to perform the measure, so a small delay gets into de loop, τ_s.
- The sensor starts retransmitting the message that carries the information of the measurement to the controller, this also takes some time, τ_{sc}.
- The controller gets the message at time $kh + \tau_s + \tau_c a$ and starts the computation of the control action, which also takes some time in the processor, τ_c.
- Once the control action is computed the controller starts the transmission of the message that carries the control action information. This transmission takes some time in the network, namely tau_{ca}
- The actuator node get the message at time $kh + \tau_s + \tau_{sc} + \tau_c + \tau_{ca}$, but also takes some small time to apply it to the plant, and thus it induces a small delay τ_a into the control loop.

This sequence shows that there is an offset between the sampling instant and the actuation instant, which brakes the assumptions we did to derive the discrete time model in previous sections.

To overcome this problem we propose a new model.

2.2 *Delayed time model*

The proposed model should take into consideration the effects of the network in the control loop, the detailed timing of the model is shown in figure (**??**)

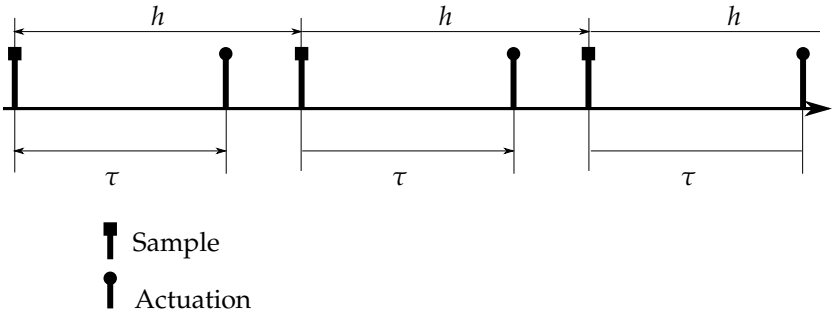


Figure 2.2: Time analysis of the networked control loop

As it can be seen the actuation and the sampling aren't performed in the same time instant, so we have to clarify how the control action

is really aplied to the system. Next figure shows a random control signal as seen from the controller plant point of view.

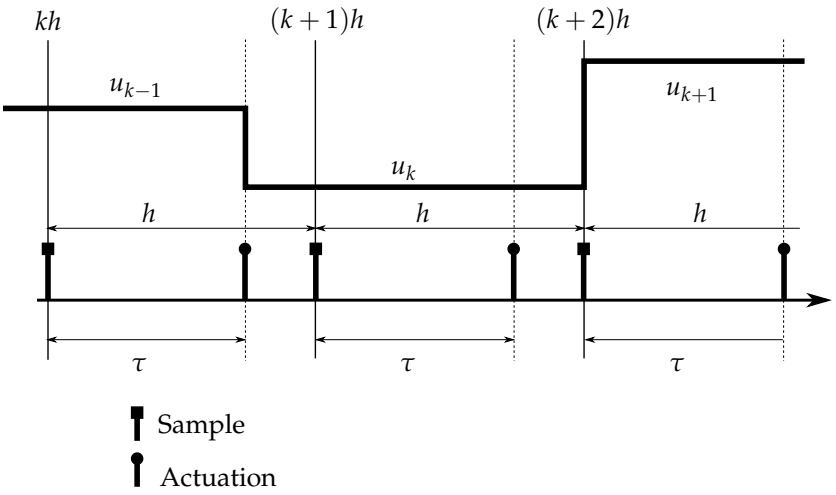


Figure 2.3: Control signal change instants due to the delays induced by the network

It can be seen that the control signal changes in the middle of the period, so the assumpltios made in the deduction of the discrete time model are no longer valid. We see how there is a control signal constant from kh up to $kh + \tau$, and then a new interval from $kh + \tau$ up to $(k + 1)h$ where the control signal is constant again.

Applying the idea of chain of initial condition we can write these equations:

- From kh up to $kh + \tau$ we can write

$$x_{kh+\tau} = e^{A\tau} x_{kh} + \int_0^{\tau} e^{As} B ds \, u_{(k-1)h} \tag{2.1}$$

- From $kh + \tau$ up to $(k + 1)h$ the exact solution is

$$x_{(k+1)h} = e^{A(h-\tau)} x_{kh+\tau} + \int_0^{h-\tau} e^{As} B ds \, u_k \tag{2.2}$$

If we substitute equation (2.1) into equation (2.2) we get:

$$x_{(k+1)h} = e^{Ah} x_{kh} + e^{A(h-\tau)} \int_0^{\tau} e^{As} B ds \, u_{(k-1)h} + \int_0^{h-\tau} e^{As} B ds \, u_k \tag{2.3}$$

And, using a more convenient notation we usually write[1] it in the form

$$x_{k+1} = \Phi(h) x_k + \Phi(h - \tau)\Gamma(\tau) u_{k-1} + \Gamma(h - \tau) u_k \tag{2.4}$$

[1] We use the property $\Phi(h - \tau)\Phi(\tau) = \Phi(h)$. This property mey be only applied if the matrices e^{Ah} and $e^{A(h-\tau)}$ commute. In this case those matrices commute, try to prove that

As it can be seen that is a two step model. If we propose a controller with the form $u_k = Kx_k$ we get the equation

$$x_{k+1} = \Phi(h)x_k + \Phi(h-\tau)\Gamma(\tau)Kx_{k-1} + \Gamma(h-\tau)Kx_k \tag{2.5}$$

Where is impossible to get a closed loop expression, due to the fact that we cannot take common factor x_k. Going back to equation (2.7) we see that it may be rearranged as[2]

$$\begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} = \begin{bmatrix} \Phi(h) & \Phi(h-\tau)\Gamma(\tau) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} \Gamma(h-\tau) \\ 1 \end{bmatrix} u_k \tag{2.6}$$

This lifted[3] system collects the system dynamics and the dynamics of the control signal. It has an extra pole, so some care must be taken when designing the controller. The standard way to design controllers works flawlessly with these lifted systems.

[2] Just perform the matrix multiplications to get two equations, the first one is (2.5) and the second one just states that $u_k = u_k$

[3] A matrix of matrices operates as this example. Suppose R and T to be

$$R = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$
$$T = \begin{bmatrix} e \\ f \end{bmatrix}$$

Then the matrix of matrices

$$S = \begin{bmatrix} R & T \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 0 \end{bmatrix}$$

Example 2.2.1. Lets use the dobule integrator model

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

We want to design a controller that sets the discret eigen-values of the system to $z_{1,2} = \pm 0.5$ with a sampling period of $h = 0.01$s, we know that there is a small delay induced by the network of $\tau = 0.005$s

Solution:

Taking advantage of our knowledge of Matlab we can construct directly the delayed system and construct the controller

---

Matlab code:

```
A=[0 1;0 0];                  %system matrix
B=[0;1];                      %Input matrix
h=0.01;                       %sampling period
tau=0.005
[phi_h,gamma_h]=c2d(A,B,h); %Discrete model
[phi_t,gamma_t]=c2d(A,B,h); %Discrete model
[phi_ht,gamma_ht]=c2d(A,B,h);%Discrete model
Phi_extended=[phi_h phi_ht*gamma_t;
              0  0       0       ];
Gamma_extended=[gamma_ht;1];
```

Result:

```
Phi_extended =
[1.0000    0.0100    0.0002]
[     0    1.0000    0.0100]
[     0         0         0]
Gamma_extended =
[0.0001]
[0.0100]
[1.0000]
k =
1.0e+03 *
[3.7500    0.0813    0.0010]
```

```
z=[0.5, -0.5 , 0];          %Discrete poles
k=acker(Phi_extended,Gamma_extended,z) %Controler
```

2.3 *Induced delay longer than the sampling period*

Now let us consider the case when τ is longer than h. The number of possible cases grows as τ gets longer and longer. To develop the model we are going to consider the simplest case, the one where $h > \tau > 2h$. The associated timing diagram of the sampling and the control actions may be observer in next figure

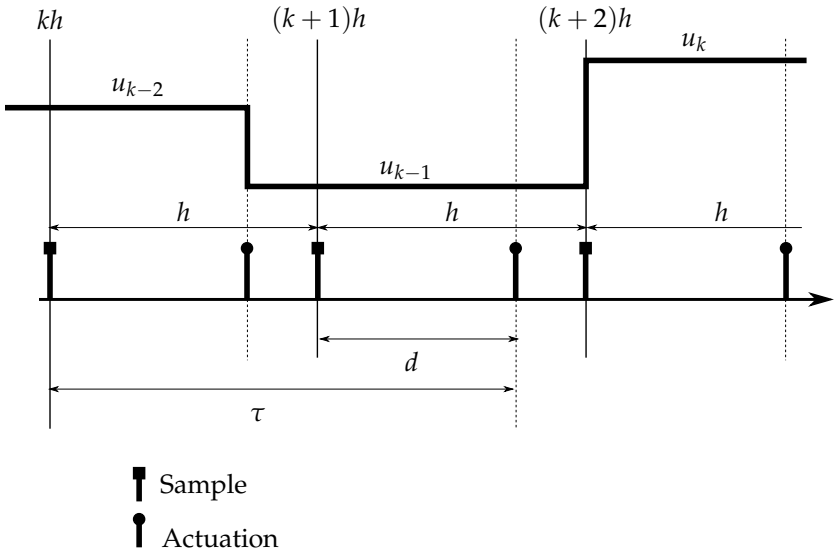


Figure 2.4: Control signal change instants due to the delays induced by the network. In this case the delay is larger than the sampling period, and thus, $h > \tau > 2h$

We define[4] $d = \tau - h$. As it can be seen in the diagram of figure (**??**), at sampling instant kh the control law acting on the system is u_{k-2}, taking thisinto account and following the philosophy of chain of initial conditions we can state[5] that

$$x_{k+1} = \Phi(h)x_k + \Phi(h-d)\Gamma(d)u_{k-2} + \Gamma(h-d)u_{k-1} \tag{2.7}$$

Once again the control signal replaced by its expression will result into a strange pattern which can be solved using a lifted system. In this case will be

[4] For longer delays this is defined as $d = \tau - nh$, and n is the number of full sampling steps that the delay covers

[5] Try to prove this yourself

$$\begin{bmatrix} x_{k+1} \\ u_{k-1} \\ u_k \end{bmatrix} = \begin{bmatrix} \Phi(h) & \Phi(h-d)\Gamma(d) & \Gamma(h-d) \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-2} \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_k \quad (2.8)$$

So new dimensions appear as the delay grows. This can be easily extended to longer delays.

Example 2.3.1. Lets use the dobule integrator model

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

We want to design a controller that sets the discret eigen-values of the system to $z_{1,2} = \pm 0.5$ with a sampling period of $h = 0.01$s, we know that there is a small delay induced by the network of $\tau = 0.015$s

Solution:

Taking advantage of our knowledge of Matlab we can construct directly the delayed system and construct the controller, in thid case $n = 1$ and $d = 0.005$

Matlab code:

```
A=[0 1;0 0];                    %system matrix
B=[0;1];                        %Input matrix
h=0.01;                         %sampling period
tau=0.005
[phi_h,gamma_h]=c2d(A,B,h); %Discrete model
[phi_t,gamma_t]=c2d(A,B,h); %Discrete model
[phi_ht,gamma_ht]=c2d(A,B,h);%Discrete model
Phi_extended=[phi_h phi_ht*gamma_t gamma_ht;
              0  0       0           1
              0  0       0           0    ];
Gamma_extended=[0 ; 0 ; 0 ; 1];
z=[0.5, -0.5 , 0 , 0 ];          %Discrete poles
k=acker(Phi_extended,Gamma_extended,z) %Controler
```

Result:

```
Phi_extended =
[1.0000    0.0100    0.0002    0.0001]
[     0    1.0000    0.0100    0.0100]
[     0         0         0    1.0000]
[     0         0         0         0]
Gamma_extended =
[0]
[0]
[0]
[1]
k =
1.0e+03 *
[3.7500    0.1188    0.0014    0.0020]
```

3
Lyapunov stability

Lyapunov stability is a way to characterize the qualitative idea of stability in models of physical systems. A stable solution to a differential equation is one which does not change much when the initial conditions are changed slightly. We shall look at two methods of checking the stability of solutions to ordinary differential equations. The first works by characterizing the solutions of linear systems to study their stability and then looking at nonlinear systems which are close to the linear case. The second uses an auxilliary function to find stability without having to characterize the solutions. Finally we look at some of the limitations of the Lyapunov notion of stability.

3.1 Introduction

We are going to look at general ordinary differential equations of the following type:

$$x'(t) = f(t, x(t)) \tag{3.1}$$

where $x : \Re \to \Re^n$ and $f : \Re \times \Re^n \to \Re^n$ are functions and t, which usually represents time, is the only independent variable. The derivative $x'(t)$ is simply the derivatives of each of the component functions, so if $x(t) = (x_1(t), ..., x_n(t))$ where $x_i(t)$ are real-valued functions on $\Re$ then

$$x'(t) = (x_1'(t), ..., x_n'(t)).$$

In some cases we will be considering the autonomous case

$$x'(t) = f(x(t)) \tag{3.2}$$

where $f : \Re^n \to \Re^n$. We are going to assume the existence of continuous solutions and uniqueness of a solution with the initial condition $x(t_0) = x_0$ where t_0 is a constant and x_0 is a constant n-vector. In the simplest case, a solution will tend to a constant as t goes to infinity. In the autonomous case, we can think of the solution as a continuous

curve through n-space (called phase space), and when the solution tends to a constant, the curve will converge to a point in phase space. If a constant function c satisfies the differential equation, then the the derivative $x'(t) = f = 0$ for $x_0 = c$ and c is called an equilibrium point. Periodic solutions satisfy $x(t) = x(t + n\tau)$ for some τ and any integer n. Other, more complicated solutions also exist which may or may not be stable. Nonautonomous systems have analogous types of solutions.

We are going to be studying the stability of (3.1). Stability is a qualitative notion that if we perturb the initial condition slightly, say $x(t_0) = x_0 + \varepsilon$, then the new solution will be close to the original system. Stable differential equations are more useful in practice since x_0 is often a measured value and does not have infinite precision. A stable differential equation would insure that our model would come up with an answer reasonably close to the actual answer an exact initial condition.

First, we will define our notion of stability. What we shall refer to as *stable* is more correctly termed *Lyapunov stable* since there are many different mathematical notions of stability.

Definition 3.1.1. A solution $x(t)$ is *stable* if for every $\varepsilon > 0$ there exists $\delta > 0$ such that if $\|x_0 - \bar{x}_0\| < \delta$ then $\|x(t) - \bar{x}(t)\| < \varepsilon$ for every $t \geq t_0$, where $\bar{x}(t)$ is the solution to

$$\bar{x}'(t) = f(t, \bar{x}(t))$$

with $x(t_0) = x_0$ and $\bar{x}(t_0) = \bar{x}_0$.

Sometimes our model works so well that if we change the initial condition slightly, the solution will still converge to the exact solution under the original initial conditions. This means that when we use a measured value, our approximate model (with an inexact initial condition) not only stays close to the actual solution, but gets closer and closer to the actual solution as t increases. This notion is called *asymptotic stability*.

Definition 3.1.2. A solution $x(t)$ is *asymptotically stable* if it is stable and there is a $\rho > 0$ such that if $\|x_0 - \bar{x}_0\| < \rho$ then

$$\lim_{t \to \infty} \|x(t) - \bar{x}(t)\| = 0.$$

In general, asymptotic stability is more desirable, since we know that no matter where we place our initial condition, we will converge to the actual answer if we are close enough to the actual initial condition. Unfortunately, asymptotic stability is not always possible, as we shall see in the fourth section.

We will primarily use these two notions of stability. There are two main methods used to study Lyapunov stability, known as Lyapunov's First and Second (or Direct) Methods.

3.2 *Lyapunov's First Method*

Lyapunov's First Method works by characterizing solutions to differential equations of the type (3.1) and using those characterizations to learn what we can about stability. Thus our approach will begin with simple linear systems and move on to certain kinds of nonlinear systems which exhibit similar behavior to the linear systems.

We will first look at linear differential equations

$$x'(t) = A(t)x(t) \tag{3.3}$$

where $A : \Re^n \to \Re^n$ is linear and $x : \Re \to \Re^n$.

When working with linear systems of the form (3.3) we will use the following theorem, without proof, to characterize solutions. The theorem is presented as a matrix equation where the derivative $C'(t)$ of a matrix function C is simply defined as the derivative of each of the entries of the matrix.

Theorem 3.2.1. If $\Phi'(t) = A(t)\Phi(t)$ with Φ an invertible $n \times n$ matrix then all solutions of (3.3) are linear combinations of the columns of Φ and, in particular, we can choose some $\Phi(t)$ such that $\Phi(t_0) = I$ where I is the identity matrix, so that $x(t) = \Phi(t)x_0$ is a solution to (3.3).

Proof. Omitted.

Definition 3.2.1. Φ as described in the last theorem is called the *fundamental matrix* of the differential equation.

The fundamental matrix depends only on the differential equation and thus only on the matrix function $A(t)$.

We also will need to define the operator norm for a matrix A:

Definition 3.2.2. If A is an $n \times n$ real matrix then

$$\|A\| = \sum_{i,j=1}^{n} |a_{ij}|$$

Note that $\|Ax\| \leq \|A\| \, \|x\|$ where $\|A\|$ is the operator norm and the other two norms are the usual Euclidean vector norm in $\Re^n$.

We are now ready to prove an important theorem. It turns out that in the linear case the property of solutions being stable is equivalent to the property of solutions being bounded.

Theorem 3.2.2. All solutions of (3.3) are stable if and only if they are bounded.

Proof. All solutions are of the form $\Phi(t)x_0$ where Φ is the fundamental matrix such that $\Phi(t_0) = I$. Let $x(t)$ and $\bar{x}(t)$ be solutions satisfying $x(t_0) = x_0$ and $\bar{x}(t_0) = \bar{x}_0$ respectively.

Assume that all solutions to (3.3) are bounded. Thus we can find an M such that $\|\Phi(t)\| < M$ for all $t > 0$. Given some $\varepsilon > 0$, if $\|x_0 - \bar{x}_0\| < \frac{\varepsilon}{M}$ then

$$\begin{aligned} \|x(t) - \bar{x}(t)\| &= \|\Phi(t)x_0 - \Phi(t)\bar{x}_0\| \\ &\leq \|\Phi(t)\| \, \|x_0 - \bar{x}_0\| \\ &< \varepsilon. \end{aligned}$$

Conversely, assume that all solutions are stable. Thus the particular solution $x(t) \equiv 0$ is stable, so for every $\varepsilon > 0$ there exists $\delta > 0$ such that if $\|\bar{x}_0\| < \delta$ then

$$\|\bar{x}(t)\| = \|\Phi(t)\bar{x}_0\| < \varepsilon.$$

In particular, let

$$\bar{x}_0^{(i)} = \frac{\delta}{2} e_i$$

where e_i is the ith standard basis vector of $\Re^n$. Then

$$\|\Phi(t)\bar{x}_0^{(i)}\| = \|\phi_i(t)\| \frac{\delta}{2} < \varepsilon$$

where ϕ_i is the ith column of Φ. We can sum these over all i and get:

$$\|\Phi(t)\| \leq \sum_{i=1}^{n} \|\phi_i(t)\| \leq \frac{2n\varepsilon}{\delta} = k$$

so

$$\|x(t)\| = \|\Phi(t)x_0\| \leq k\|x_0\|$$

and $x(t)$ is bounded. □

Note that boundedness is not necessarily true for stable systems in general. For instance, take the differential equation $x'(t) = 1$ whose solutions are all stable but not bounded.

Lyapunov's First Method consists mainly of showing that solutions of linear systems, especially if $A(t)$ is a constant funtion, are bounded and hence stable. It then looks at functions which are nonlinear, but simply small perturbations of linear functions. We shall state without proof several theorems. For a more detailed explanation with proofs, see [2].

Consider the constant linear equation:

$$x'(t) = A\, x(t) \tag{3.4}$$

where A is a constant matrix. We can easily show the following.

Theorem 3.2.3. If all eigenvalues of A have negative real parts then every solution to (3.4) is asymptotically stable.

Proof. Omitted.

We can easily extend this to equations with slight perturbations. So we characterize the following nonlinear equation

$$x'(t) = A\ x(t) + f(t, x(t)) \tag{3.5}$$

with the following theorem.

Theorem 3.2.4. If f satisfies:

1. $f(t,x)$ is continuous for $\|x\| < a,\ 0 \leq t < \infty$ and
2. $\lim_{\|x\|\to 0} \frac{\|f(x,t)\|}{\|x\|} = 0$

then the solution $x(t) \equiv 0$ is asymptotically stable.

It turns out that to characterize the general linear case where A is a function of time, we need a slightly stronger notion of stability called uniform stability. The reader is again referred to [2].

3.3 *Lyapunov's Second Method*

Lyapunov's Second or Direct Method is unique in that it does not require a characterization of the solutions to determine stability. This method often allows us to determine whether a differential equation is stable without knowing anything about what the solutions look like, so it is ideal for dealing with nonlinear systems.

The method uses a supplementary function called a Lyapunov function to determine properties of the asymptotic behavior of solutions to a differential equation of the general form (3.1). We will need the following definitions.

Definition 3.3.1. A function $V(t,x)$ is *positive [negative] definite* if there exists a real-valued function $\varphi(r)$ such that:

1. $\varphi(r)$ is strictly increasing on $0 \leq r \leq a$ and $\varphi(0) = 0$ and
2. $V(t,x) \geq \varphi(\|x\|)$ $[V(t,x) \leq -\varphi(\|x\|)]$ for all $(t,x) \in \{(t,x) : t_0 \leq t < \infty, \|x\| \leq b < a\}$.

Definition 3.3.2.

$$V' = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i(t,x) + \frac{\partial V}{\partial t}$$

Note that $V'(t, x(t))$ is the time derivative of V.

Definition 3.3.3. A real function $V(t,x)$ is said to admit an *infinitesimal upper bound* if there exists $h > 0$ and a continuous, real-valued, strictly increasing function ψ with $\psi(0) = 0$ such that

$$|V(t,x)| \leq \psi(\|x\|) \text{ for } \|x\| < h \text{ and } t \geq t_0.$$

Using these definitions we shall construct a function V. If we can find such a V, then we can find stable solutions to the differential equation. Unfortunately, there is no general way to construct V from the differential equation (3.1). Note that we can define V in terms of t and x (not $x(t)$) and since our definition of V' uses the function f, the existence of V will have implications for the differential equation.

Theorem 3.3.1. If a continuous real-valued function $V(t,x)$ exists such that:

1. $V(t,x)$ is positive definite and
2. $V'(t,x)$ is non-positive

then $x(t) \equiv 0$ is a stable solution of (3.1).

Proof. We know that $V(t,x)$ is positive definite, so there exists a strictly increasing function $\varphi(r)$ such that

$$0 < \varphi(\|x\|) \leq V(t,x)$$

for $0 < \|x\| < b$ and $t > t_0$.

Given $\varepsilon > 0$ let

$$m_\varepsilon = \min_{\|x\|=\varepsilon} \varphi(\|x\|) = \varphi(\varepsilon).$$

Notice that $m_\varepsilon > 0$. Since V is continuous and $V(t,0) = 0$ for all t (by positive definiteness), we can choose a $\delta > 0$ such that

$$V(t_0, x_0) < m_\varepsilon$$

if $\|x_0\| < \delta$. Now

$$V'(t, x(t)) \leq 0$$

for $t_1 \geq t_0$ and $\|x_0\| < \delta$ implies

$$V(t_1, x(t_1)) \leq V(t_0, x(t_0)) = V(t_0, x_0) < m_\varepsilon$$

Now, suppose for some $t_1 \geq t_0$ that $\|x(t_1)\| \geq \varepsilon$ whenever $\|x_0\| < \delta$. Then

$$V(t_1, x(t_1)) \geq \varphi(\|x(t_1)\|) \geq \varphi(\varepsilon) = m_\varepsilon$$

which is a contradiction, so for every $\varepsilon > 0$ there exists $\delta > 0$ such that $\|x(t)\| < \varepsilon$ if $\|x_0\| < \delta$ for $t \geq t_0$, i.e. $x(t) \equiv 0$ is stable. □

We call a function V which satisfies the hypotheses of the theorem a *Lyapunov function*. In physical systems, the Lyapunov function used is often an expression for energy, for instance stability for the differential equation for a spring:

$$\begin{aligned} x'(t) &= y \\ y'(t) &= -kx \end{aligned}$$

can be done with $V(x,y) = \frac{1}{2}y^2 + \frac{1}{2}kx^2$, which is the equation for total energy of the system.

By strengthening the requirements on V we can find conditions for asymptotic stability.

Theorem 3.3.2. If a continuous function $V(t,x)$ exists satisfying:

1. $V(t,x)$ is positive definite,
2. $V(t,x)$ admits an infinitesimal upper bound, and
3. $V'(t,x)$ is negative definite

then the solution $x(t) \equiv 0$ of (3.1) is asymptotically stable.

Proof. By the previous theorem, the solution must be stable. Suppose that $x(t)$ is not asymptotically stable, so for every $\varepsilon > 0$ there exist $\delta > 0$ and $\lambda > 0$ such that any nonzero solution $x(t)$ where $x(t_0) = x_0$ satisfies $\lambda \leq \|x(t)\| < \varepsilon$ if $t \geq t_0$ and $\|x_0\| < \delta$.

Since $V'(t,x)$ is negative definite, there is a strictly increasing function $\varphi(r)$ vanishing at the origin such that $V'(t,x) \leq -\varphi(\|x\|)$. We know that $\|x(t)\| \geq \lambda > 0$ for $t \geq t_0$ so there is a $d > 0$ such that $V'(t,x(t)) \leq -d$. This implies that

$$\begin{aligned} V(t,x(t)) &= V(t_0,x_0) + \int_{t_0}^{t} V'(s,x(s))ds \\ &\leq V(t_0,x_0) - (t-t_0)d \end{aligned}$$

which is less less than zero for large t, contradicting the assumption that $V(t,x)$ is positive definite.

Thus no such λ exists and since $V(t,x(t))$ is positive definite and decreasing with respect to t

$$\lim_{t\to\infty} V(t,x(t)) = 0$$

which implies that

$$\lim_{t\to\infty} \|x(t)\| = 0,$$

since $V(t,x(t)) \leq \psi(\|x(t)\|)$ where $\psi(r)$ is strictly increasing and vanishes at the origin (by the second hypothesis). Thus as $V(t,x(t)) \to 0$, $\psi(\|x(t)\|) \to 0$, and $\|x(t)\| \to 0$ (because ψ is strictly increasing and vanishes at o). Thus $x(t) \equiv 0$ is asymptotically stable. □

We can also use the direct method to determine instability in some cases.

Theorem 3.3.3. If a continuous real-valued function $V(t,x)$ exists on some set $S = \{(t,x) : t \geq t_1 \text{ and } \|x\| < a\}$ satisfying:

1. $V(t,x)$ admits an infinitesimal upper bound,
2. $V'(t,x)$ is positive definite on S, and
3. There exists a $T > t_1$ such that if $t_0 \geq T$ and $h > 0$ then there exists $c \in \Re^n$ such that $\|c\| < h$ and $V(t_0,c) > 0$

then the solution $x(t) \equiv 0$ is not stable.

Proof. Suppose $x(t)$ is a solution not identically zero. Then, by uniqueness, $\|x(t)\| \neq 0$ for all $t \geq t_1$. If $x(t)$ is defined at t_0 then for every $t \geq t_0$ for which $x(t)$ is defined

$$V(t,x(t)) - V(t_0,x(t_0)) = \int_{t_0}^{t} V'(s,x(s))ds > 0$$

since $V'(t,x(t))$ is positive definite.

Let $t_0 \geq T$ and $\varepsilon > 0$. There exists a c such that $\|c\| < \min(a,\varepsilon)$ and $V(t_0,c) > 0$ by the third hypothesis. Since V admits an infinitesimal upper bound and is continuous, there is a $\lambda \in (0,a)$ such that $\|x\| < \lambda$ for $t \geq t_1$. Therefore,

$$|V(t,x)| < V(t_0,c).$$

Let $x(t)$ satisfy the initial condition $x(t_0) = c$. Then $V(t,x(t) > V(t_0,x(t_0)) > 0$ (from the positive definiteness argument), so $\|x(t)\| \geq \lambda$. Also by the second hypothesis we know that there is a strictly increasing function $\varphi(r)$ such that $V'(t,x) \geq \varphi(\|x\|)$. Let

$$\mu = \min_{\|x\| \in [\lambda,a]} \varphi(\|x\|) = \varphi(\lambda).$$

then

$$V'(t,x(t)) \geq \varphi(\|x(t)\|) \geq \mu.$$

Using the integral inequality, we get

$$V(t,x(t)) \geq V(t_0,x(t_0)) + (t - t_0)\mu.$$

Thus we can make $V(t,x(t))$ arbitrarily large. Since V(t,x) admits an infinitesimal upper bound, there exists a strictly increasing function $\psi(r)$ such that $|V(t,x(t))| < \psi(\|x\|)$ and $\psi(0) = 0$. So since $V(t,x(t))$ can be arbitrarily large, $\psi(\|x(t)\|)$ can also be arbitrarily large, and thus $\|x(t)\|$ gets arbitrarily large and the solution is not stable. □

3.4 *Limitations of Lyapunov Stability*

The Lyapunov notion of stability has some problems, some of which we shall look into here. Asymptotic stability as we have defined it does not always work well for periodic systems, so often other notions such as orbital stability are used when studying differential equations which have periodic solutions. In addition, certain classes of systems such as Hamiltonian Systems do not react well to a Lyapunov analysis. Finally, there is the question of whether Lyapunov stability actually reflects our qualitative idea of stability.

We will first look at how Lyapunov stability works with periodic solutions to the autonomous system (3.2). It turns out that our notion of asymptotic stability is incompatible with such solutions, since the points will not get closer together, even thought two solutions may get closer and closer to the same final state (a periodic cycle). To prove this we will first need a Lemma that solutions to autonomous systems are invariant under translation.

Lemma 3.4.1. If $x(t)$ is a solution to (3.2) then $x(t+h)$ is also a solution for any real number h.

Proof. Let $\bar{x}(t) = x(t+h)$ and $s = t+h$. Then

$$\begin{aligned} \bar{x}'(t) &= \frac{dx}{ds}\frac{ds}{dt} \\ &= x'(s) \\ &= f(x(s)) \\ &= f(x(t+h)) \\ &= f(\bar{x}(t)) \end{aligned}$$

so $\bar{x}(t) = x(t+h)$ is a solution to (3.2). □

Now we can use the fact that translations of solutions are also solutions to show that nontrivial periodic solutions cannot be asymptotically stable. By nontrivial periodic we mean that the solutions do not converge to a constant vector, i.e., $x(t) \neq x(t+\varepsilon)$ for some small $\varepsilon > 0$.

Theorem 3.4.1. If $x(t)$ is a nontrivial periodic solution of an autonomous system (3.2) then $x(t)$ is not asymptotically stable.

Proof. Suppose $x(t)$ is a nontrivial periodic solution. Thus there exists t_0 such that $f(x(t_0)) \neq 0$ because if not, $x'(t) = 0$ for all t and the solution is constant (trivially periodic). Suppose also that $x(t)$ is asymptotically stable.

The stability assumption says that given $\varepsilon > 0$ there exists $\hat{\delta} > 0$ such that $\|x(t) - \bar{x}(t)\| < \varepsilon$ if $\|x_0 - \bar{x}_0\| < \hat{\delta}$ where $x(t)$ satisfies

$x(t_0) = x_0$ and $\bar{x}(t)$ is a solution to the same equation satisfying the initial condition $\bar{x}(t_0) = \bar{x}_0$.

By the asymptotic stability assumption we can find a $\rho > 0$ such that

$$\lim_{t\to\infty} \|x(t) - \bar{x}(t)\| = 0$$

if $\|x_0 - \bar{x}_0\| < \rho$.

Let $\delta = \min(\hat{\delta}, \rho)$. By the continuity of $x(t)$ we can fix some small $\bar{t}$ so that

$$\|x(t_0) - x(t_0 + \bar{t})\| < \delta.$$

By the Lemma, $x(t + \bar{t})$ is a solution to the differential equation, so by our asymptotic stability assumption

$$\lim_{t\to\infty} \|x(t) - x(t + \bar{t})\| = 0.$$

We know that $x'(t_0) \neq 0$ so we can find $r > 0$ so that $\|x(t_0) - x(t_0 + \bar{t})\| > r$. Also $x(t)$ is periodic, so assume it has period τ. Then

$$\|x(t_0 + n\tau) - x(t_0 + \bar{t} + n\tau)\| > r > 0$$

for all n, so the limit cannot go to zero, which contradicts our assumption of asymptotic stability. Thus $x(t)$ cannot be asymptotically stable. □

Another class of problems which for which asymptotic stability does not work is Time-Independent Hamiltonian systems. They are defined as follows.

Definition 3.4.1. A *time-independent Hamiltontian system (TIHS)* is the 2n dimensional system

$$\begin{aligned} x'(t) &= H_y(x(t), y(t)) \\ y'(t) &= -H_x(x(t), y(t)) \end{aligned}$$

where $x(t)$ and $y(t)$ are both functions from $\Re$ to $\Re^n$ and $H(x, y)$ is a real valued function with continuous partial derivatives.

Hamiltonian Systems are used to model a large class of physical systems and have many applications. It turns out that, like nontrivial periodic solutions to autonomous systems, no solutions to a TIHS can be asymptotically stable.

Theorem 3.4.2. If $(x(t), y(t))$ is a solution to a TIHS then it is not asymptotically stable.

Proof. Suppose $(x(t), y(t))$ is an asymptotically stable solution. Then we can find t_0 and $\delta > 0$ such that if $(\bar{x}(t), \bar{y}(t))$ is a solution and if $\|(x(t_0), y(t_0)) - (\bar{x}(t_0), \bar{y}(t_0))\| < \delta$ then

$$\lim_{t\to\infty} \|(x(t), y(t)) - (\bar{x}(t), \bar{y}(t))\| = 0.$$

But also,

$$\frac{d}{dt}H(x,y) = \frac{\partial H}{\partial x}\frac{dx}{dt} + \frac{\partial H}{\partial y}\frac{dy}{dt} = H_x(H_y) + H_y(-H_x) = 0$$

so $H(x,y)$ is constant on any solution $(x(t), y(t))$. By the continuity of H, it must be constant on some δ-neighborhood N of $(x(t_0), y(t_0))$, so every point in N is an equilibrium point. Thus $(x(t), y(t))$ cannot be asymptotically stable (for initial condition $(\bar{x}(t_0), \bar{y}(t_0)) \in$ N, the solution $(\bar{x}(t), \bar{y}(t))$ will not converge to $(x(t), y(t))$. □

It turns out that although Lyapunov stability is extremely useful in mathematically defining our qualitative view of stability, it is neither necessary nor sufficient for something to appear stable on a physical level. Some examples can be found at the end of Chapter 4 of [1].

4
What is a mtatrix inequality?

In this section we are going to explain the meaning of the equation

$$A \prec B \tag{4.1}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times n}$. That curly symbol "less than", $\prec$, means that A is samller than B. Although matrices are not an ordered set, there is a partial ordering, which is represented in this way.

To check if a matrix A is less than a matrix B just take a vector $x \in \mathbb{R}^{n\times 1}$ and check that

$$x^T Ax < x^T Bx \tag{4.2}$$

for any value[1] of x.

One may be overhelmed because the number of possible values of x is infinite, but if yoy take a new vector λx and you check expressión (4.2) you simply get

$$\lambda x^T Ax\lambda < \lambda x^T Bx\lambda \tag{4.3}$$

$$x^T Ax < x^T Bx \tag{4.4}$$

which is again equation (4.2). So you do not nedd to check x for the entire space, you only need to check it in the unit circle.

Even in this case is hard to check all possible values in the unit circle, but expressión (4.2) gives us a nice interpretation. To get a vissual interpretation of this expression let's make a drawing, in this case in two dimenssional space. We are going to take all possible vectors in the unit circle, that is $x \in \mathbb{R}^{2\times 1}$ such that

$$x = \begin{bmatrix} \cos(2\pi t) \\ \sin(2\pi t) \end{bmatrix} \tag{4.5}$$

with $t \in [0..1)$, so for any value aof t we get a unit vector that points in the angle $\theta = 2\pi t$ as shown in figure (4.1) Now compute the

[1] Observe the diference in the symbology, now $x^T Ax$ and $x^T Bx$ are scalars, so the comparation is easy to perform

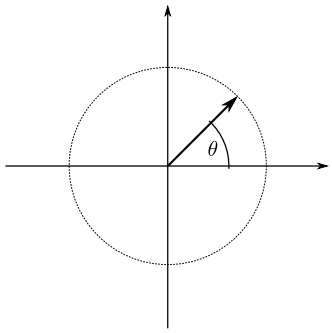


Figure 4.1: Unit circle vector

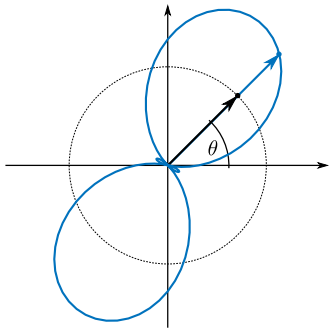


Figure 4.2: Reulting curve after computing $x^T Ax$ for all possible values of x in the unit circle. The blue line has been scaled to

value of $x^T Ax$, which is an scalar, in order to ilustrate this take the A matrix to be

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{4.6}$$

The computation of $x^T Ax$ throws an scalar value which we name r and is given by $r = \sin(2\pi t)(2\cos(2\pi t) + 4\sin(2\pi t)) + \cos(2\pi t)(\cos(2\pi t) + 3\sin(2\pi t))$. Now plot the curve $(r(t), \theta(t))$ in polar coordinates on top og the unit circle. Figure (4.2) shows the results of these operations[2].

[2] Obiously, among all possible vectors there are the eigenvactors of A, such vectors have the nice property $x_i^T Ax_i = \lambda_i \|x\|$

Now we are in conditions to make some interpretations, for instance, when we say

$$A \succ 0 \tag{4.7}$$

we mean that $x^T Ax > 0$ for any x except $x = 0$. Thismeans that the transformation of the unit circle never touches the origin and is always positive, in this case we say that the matrix is ositive definite[3].

[3] A positive definite matrix has all its eigen values positive and real.

Whenever we use the expression $A \succ B$, we are saying that $A - B \succ 0$, which means that the subtraction of the matrices is positive definite. A nice interpretation of this fact can be seen in figure (4.3), the A matrix grater than the B matrix means that the transformaton of the unit circle of the former sorrounds without touching the transformation of the last one.

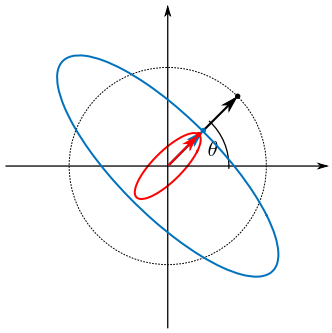


Figure 4.3: Example of $A \succ B$

Whith this intuitions in hand we can define the positiviness of a matrix. Next definitions are formal

- a matrix A is positive-definite iff $A \succ 0$, which means that $x^T Ax > 0 \ \forall x$ and also means that $eig(A) > 0$
- a matrix A is positive-semidefinite iff $A \succeq 0$, which means that $x^T Ax \geq 0 \ \forall x$ and also means that $eig(A) \geq 0$
- a matrix A is negative-definite iff $A \prec 0$, which means that $x^T Ax < 0 \ \forall x$ and also means that $eig(A) < 0$
- a matrix A is positive-definite iff $A \preceq 0$, which means that $x^T Ax \leq 0 \ \forall x$ and also means that $eig(A) \leq 0$

4.1 *Linear Matrix inequalities*

A linear matrix inequality (LMI) is an affine matrix-valued function,

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i \succ \mathbf{0} \tag{4.8}$$

where $x \in \mathbb{R}^m$ are called the decision variables and $F_i = F_i^T \in \mathbb{R}^{n \times n}$ are symmetric matrices.

A very important aspect of the LMI is that it defines a convex set. To see this let x_1 and x_2 be two solutions of a LMI problem, i.e. $F(x) \succ \mathbf{0}$. In this case, as x_1 and x_2 are solutions of this problem we know that $F(x_1) \succ \mathbf{0}$ and $F(x_2) \succ \mathbf{0}$. Then any convex combination $x = (1-\lambda)x_1 + \lambda x_2$ with $\lambda \in [0,1]$ solves the LMI:

$$F(x) = F((1-\lambda)x_1 + \lambda x_2) = (1-\lambda)F(x_1) + \lambda F(x_2) \succ \mathbf{0} \tag{4.9}$$

Efficient numerical methods have been developed to solve these kind of problems. Some of thesemost efficient algorithms for solving LMI problems are based on interior point methods. These methods are iterative and each iteration includes a least squares minimization problem. We never solve these equations by hand, but if you want to give it a try it is possible to do so in any of the examples we present.

The format pressented in this section is the standard one, but we usually do not write it in this way, but, as it can be seen in the next example, transformation from a set of matrix inequalities into a LMI is an easy task.

Example 4.1.1. Most LMIs are not formulated in the standard form (4.8) but they can be rewritten as is shown in this example. Let us consider the following Lyapunov problem:

$$\begin{aligned} P = P^T &\succ \mathbf{0} \\ A^T P + PA &\prec \mathbf{0} \end{aligned} \tag{4.10}$$

Note that P enters linearly in both inequalities. To show how to rewrite this into the standard LMI form, we assume that $P \in \mathbb{R}^{2\times 2}$. Parametrize P as a linear function in x:

$$P([x_1, x_2, x_3]) = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix} = x_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + x_3 \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = x_1 P_1 + x_2 P_2 + x_3 P_3 \tag{4.11}$$

In order to include both equations of (4.10) we have to make a matrix of matrices, then $F(x) = \text{diag}\left[P(\mathbf{x}), -A^T P(\mathbf{x}) - P(\mathbf{x})A\right]$ which maps easily to:

$$F(x) = x_1 \begin{bmatrix} P_1 & \mathbf{0} \\ \mathbf{0} & -A^T P_1 - P_1 A \end{bmatrix} + x_2 \begin{bmatrix} P_2 & \mathbf{0} \\ \mathbf{0} & -A^T P_2 - P_2 A \end{bmatrix} + x_3 \begin{bmatrix} P_3 & \mathbf{0} \\ \mathbf{0} & -A^T P_3 - P_3 A \end{bmatrix} \tag{4.12}$$

which is in the form as in (4.8) with $F(\mathbf{0}) = \mathbf{0}$. In this case three decision variables are needed. In general we need $\frac{n(n+1)}{2}$ decision variables for symmetric matrices and n^2 for full square matrices of size $n \times n$.

5
Solving LMIs

To solve a LMI you have two chances, the first one is to write it explicitily as seen in example (4.1.1), and try to sove it by hand, the other solution is to use a numerical solver, which is more swited to our needs. In any case we show the two methods in this section to allow the user to undertand whats going behind the scenes when calling a numerical solver. One must take into account that there is an infinite set of possible solutions in a LMI, so an extra restriction is aded to fix the final solution, the restriction is an optimization. For instance we will require that the trace of a matrix is minimal or that the determinant is maximal or some kind of extra optimization that allows us to pick just one single matrix out of all the possible solutions.

Example 5.0.1. In this example we are going to solve by hand a LMI, we will see that there are infinite solutions an we will add an optimization problem to pick one solution outo af all possible ones.

Asume we have a system described by

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} x \tag{5.1}$$

And we wnat to check its stability; one easy way to do it is to compute the eigen values of the matrix of the system[1].From a LMI point of view, we can check the stability of the system if we are able to find a symetric positive definite matrix $P \in \mathbb{R}^{2\times 2}$, such that

[1] In this case the eigenvalues are at -1

$$A^T P + PA \prec 0 \tag{5.2}$$

We decide to create our matrix as

$$P = \begin{bmatrix} x & y \\ y & z \end{bmatrix} \tag{5.3}$$

So equation (5.2) becomes

$$\begin{bmatrix} 0 & -1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x & y \\ y & z \end{bmatrix} + \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \prec \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.4}$$

which can be written explicitly as

$$\begin{bmatrix} -2y & x-2y-z \\ x-2y-z & 2y-4z \end{bmatrix} \prec \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.5}$$

In order to be able to do the computations[2] we may fix how samll should this matrix be. For instance, we know that

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \prec \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.6}$$

[2] In a matrix inequality, $A \prec B$, the curly "less than" is not refered term y term, is refered as $x^T Ax < x^T Bx$ for $x \neq 0$, so we have to take care, we can not set each of the elements of A "less than" each of the elements of B

So we can turn the inequality into an equality that acomplishes the inequality

$$\begin{bmatrix} -2y & x-2y-z \\ x-2y-z & 2y-4z \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{5.7}$$

which easily yelds to $y = \frac{1}{2}$, $z = \frac{1}{2}$ and $x = \frac{3}{2}$. We just piked out one matrix over all the possible solutions. A more difficult problem to solve by hand [3] wold be to add a restriction in terms of an optimization, for instance

[3] The problem is harder, but it allows us to use numerical optimizers

$$\begin{aligned} &\underset{x,y,z}{\text{minimize}} && \text{Trace}(P) \\ &\text{subject to} && A^T P + PA \prec 0 \end{aligned} \tag{5.8}$$

This becomes realy hard to us to solve, but is specially well swited to solve with numerical solvers. For instance YALMIP in matlab.

5.0.1 Numerical solver, Yalmip in Matlab

We have seen that there are infinite solutions to a set of matrix inequalities, to choose one out of all the possibilities we decide to use a numerical solver, these solvers use interior point methods to find a single solution in the feasible set of solutions. In order to do so, we add an optimization restriction to the set of matrix inequalities in such a way that only one solution will be picked from the feasible set of solutions.

The most easy to use solver is Yalmip[4], an interface to a set of numerical solvers. You must install Yalmip side by side with the recomended numerical solvers, at least SEDUMI[5]. Folow installation

[4] https://yalmip.github.io/

[5] http://sedumi.ie.lehigh.edu/

instructions in the web sites to get a functional copy of both in your Matlab distribution.

Once the solvers are installed we are in conditions to solve a simple problem, for instance the one given in example (5.0.1), equations (5.11)

Example 5.0.2. Recovering the problem... Prove that the system driven by

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} x \tag{5.9}$$

is stable. To do so, we need to find a positive definite matrix P, such that

$$A^T P + PA \prec 0 \tag{5.10}$$

holds. In order to allow the solver to solve the problem we add a restriction, and the problem becomes

$$\begin{array}{lll} \underset{x,y,z}{\text{minimize}} & \text{Trace}(P) & (5.11) \\ \text{subject to} & A^T P + PA \prec 0 & \end{array}$$

First we define the matrix A in matlab

```
>>A=[0 1; -1 -2];
```

Next we need a symbolic variable that represents P in our problem, in terms of Yalmip this implies to declare a veriable P to find its value, this is done with

```
>> P = sdpvar(2,2);
Linear matrix variable 2x2 (symmetric, real, 3 variables)
```

which means a "semidefinite problem variabe of 2×2"

Having P, we are ready to define the constraints.

```
>>F=[P>0];
++++++++++++++++++++++++++++++++
|   ID|              Constraint|
++++++++++++++++++++++++++++++++
|   #1|   Matrix inequality 2x2|
++++++++++++++++++++++++++++++++
```

P must be positive definite and

```
>>F=[F, A'*P+P*A>0];
++++++++++++++++++++++++++++++++
|   ID|              Constraint|
++++++++++++++++++++++++++++++++
|   #1|   Matrix inequality 2x2|
|   #2|   Matrix inequality 2x2|
++++++++++++++++++++++++++++++++
```

the stability condition. Note that we make an array of restrictions, and we add a new restriction to the old ones with this nomenclature.

In order to solve the inequalities we call the function "solvesdp()", which gets two parameters, the set of inequalities and the minimization function, in our case

```
>>solvesdp(F,trace(P))
SeDuMi 1.3 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, theta = 0.250, beta = 0.500
eqs m = 3, order n = 5, dim = 9, blocks = 3
nnz(A) = 9 + 0, nnz(ADA) = 9, nnz(L) = 6
 it :     b*y       gap    delta  rate   t/tP*  t/tD*   feas cg cg  prec
  0 :            1.69E+01 0.000
  1 :  -1.93E-01 4.38E+00 0.000 0.2592 0.9000 0.9000   1.91  1  1  2.3E+00
  2 :  -8.05E-03 3.99E-01 0.000 0.0912 0.9900 0.9900   1.79  1  1  5.2E-01
  3 :   5.28E-06 2.83E-04 0.000 0.0007 0.9999 0.9999   1.10  1  1  7.3E-05
  4 :   5.24E-13 2.84E-11 0.000 0.0000 1.0000 1.0000   1.00  1  1  7.3E-12

iter seconds digits       c*x               b*y
  4      0.3   2.7  0.0000000000e+00  5.2390377904e-13
|Ax-b| =   6.0e-12, [Ay-c]_+ =   5.2E-13, |x|=  1.1e+00, |y|=  5.2e-13

Detailed timing (sec)
   Pre          IPM          Post
5.248E-01    4.312E-01    1.428E-01
Max-norms: ||b||=1, ||c|| = 0,
Cholesky |add|=0, |skip| = 0, ||L.L|| = 1.
```

To check the value of the solution you may ask yalmip to convert it to a matrix

```
>>P_feasible = double(P)
P_feasible =

   1.0e-12 *

   -0.5145   -0.0576
```

```
    -0.0576   -0.0094
```

Which is really small valued matrix, which does not matter, because the problem was to check the existence of the matrix, not its value. However, sometimes we would like to get the values, and this matrix is not well swited to make computations. We can change the problem such that we will try to make the trace of the P matrix equal to 1.

```
>> F=[F,trace(P)==1]
++++++++++++++++++++++++++++++++++
|   ID|                Constraint|
++++++++++++++++++++++++++++++++++
|   #1|     Matrix inequality 2x2|
|   #2|     Matrix inequality 2x2|
|   #3|   Equality constraint 1x1|
++++++++++++++++++++++++++++++++++
>> solve(F) %without minimization
```

6
Stability Non repeating sequences of matrices

When there is a limited set of matrices in a networked system,but we don't know which will be the sequence that is going to drive the system then the learned techniques no longer hold.

For instance figure 6.1 shows a networked control system with a sensor, a controller, an actuator and an extra node that is going to inject random traffic in the network. Figure 6.2 shows a hypothetic time line of the networked control system. This time line generates an extended matrix corresponds to a free network state when the sensor sends the measurement to de controller and also is free when the controller sends data to the actuator. The actuator applies the control signal to the driven system instantaneously. The decription of the dynamics in this case is given by

$$x_{k+1} = \begin{bmatrix} \Phi(h) & \Phi(h-\tau_1)\Gamma(\tau_1) \\ 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} \Gamma(h-\tau_1) \\ 1 \end{bmatrix} u_k$$

The dynamics proposed may be altered by the aparition of random messages fron the extra node. In (figure 6.3) where the sensor sends data to de controller and finds the network availabe, so there is no problem to use it. When the controller gets the measurement an extra node decides randomly, with a probability of 50%, to use the network, and thus, delaying the message from the controller to the actuator for an extra time τ_e halve of the times the controller wants to send it message. The actuator applies instantaneously the control action to the system. In this case, when the extra node interferences the loop, the equations driving the system are

$$x_{k+1} = \begin{bmatrix} \Phi(h) & \Phi(h-\tau_2)\Gamma(\tau_2) \\ 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} \Gamma(h-\tau_2) \\ 1 \end{bmatrix} u_k$$

So now, and supposing that we have a controller[1] that stabilizes both dynamics, it could happen that there appears a destructive sequence (destructive in the sense of unstable).

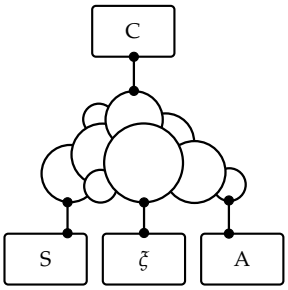


Figure 6.1: Networked system with an extra node injecting messages into the network

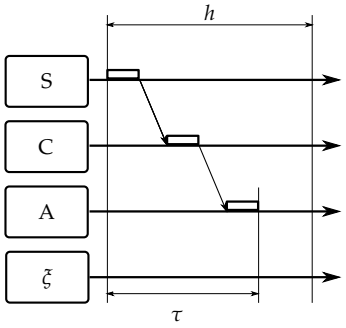


Figure 6.2: Hypotetic timeline 1, the extra node does not interference into de loop communication

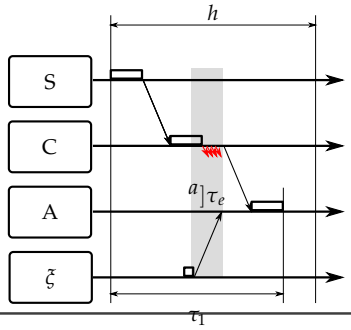


Figure 6.3: Hypotetic timeline 2, the extra node interference into the loop communications randomly, producing an estra delay τ_e

[a] [

Example 6.0.1. If we take the double integrator described by

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

We can describe the extended delayed discrete dynamics of the system for any h and τ as

$$x_{k+1} = \begin{bmatrix} 0 & h \\ 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_k$$

The matrices related to two possible executions over a network as describe previously could be

$$x_{k+1} = \begin{bmatrix} 0 & h & h\tau_1 - \frac{\tau_1^2}{2} \\ 0 & 1 & \tau_1 \\ 0 & 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} \frac{(h-\tau_1)^2}{2} \\ h - \tau_1 \\ 1 \end{bmatrix} u_k$$

$$x_{k+1} = \Phi_1 x_k + \Gamma_1 u_k$$

and

$$x_{k+1} = \begin{bmatrix} 0 & h & h\tau_2 - \frac{\tau_2^2}{2} \\ 0 & 1 & \tau_2 \\ 0 & 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} \frac{(h-\tau_2)^2}{2} \\ h - \tau_2 \\ 1 \end{bmatrix} u_k$$

$$x_{k+1} = \Phi_2 x_k + \Gamma_2 u_k$$

Now imagine you construct a controller that maps the close loop stable for both systems, lets say that the controller is K, so you get

$$x_{k+1} = (\Phi_1 + \Gamma_1 K) x_k = \Phi_{cl_1} x_k$$

$$x_{k+1} = (\Phi_2 + \Gamma_2 K) x_k = \Phi_{cl_2} x_k$$

Now consider the possible sequences under the assumptions we are making, for instance you could get the alternating sequence

$$x_{k+10} = \Phi_{cl_1}\Phi_{cl_2}\Phi_{cl_1}\Phi_{cl_2}\Phi_{cl_1}\Phi_{cl_2}\Phi_{cl_1}\Phi_{cl_2}\Phi_{cl_1}\Phi_{cl_2} x_k$$

Or maybe a sequence given by swapping the last two matrices

$$x_{k+10} = \Phi_{cl_2}\Phi_{cl_1}\Phi_{cl_1}\Phi_{cl_2}\Phi_{cl_1}\Phi_{cl_2}\Phi_{cl_1}\Phi_{cl_2}\Phi_{cl_1}\Phi_{cl_2} x_k$$

The thing is that you fall into a graph[2] that may develop an infinite set of sequences. Figure 6.4 shows its representation.

[2] In fact is not a graph, is a markov chain with given set of probabilities

---

Under the conditions exposed above is not possible to design a controller that maps the closed loop stable, due to the ambiguous sequences, so non of the previously seen techniques will be applicable to this problem.

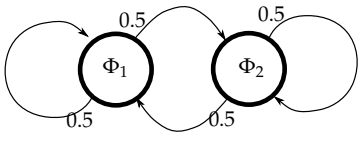


Figure 6.4: This is the graph associated with the example we are proposing, the probability of having Φ_1 after happening Φ_2 is 0.5, while the reverse case is symmetric. In this situation

6.0.1 Problem formulation

The problem formulation is to stabilise the dynamics of a system driven by a sequence of matrices piked randomly from a set of matrices.

6.0.2 Defining the set of matrices

The set of matrices is given[3] by the possible combinations of h and τ

[3] assuming the delay shorter than the sampling period, the extension to delays larger than the sampling period is straightforward

$$\begin{aligned} \Phi_{cl_i} &= \Phi(h_i, \tau_i) + \Gamma(h_i, \tau_i)K \\ i &\in 1, 2, \ldots, n \\ (h, \tau)_i &\in (h, \tau)_1, (h, \tau)_2, \ldots, (h, \tau)_n \end{aligned} \tag{6.1}$$

6.0.3 Stability of the closed loop system

Due to the fact that there is an infinite sequence of matrices we are not allowed to compute the eigen values of anything, just because is not possible to define a closed loop matrix.

In this cases there is no solution except to recall the basic lyapunov stability condition, which states that the system will be stable whenever we are able to find $V(x) > 0 \ \forall \ x - \{0\}$ [4] such that

[4] except in the point **0**, where we allow $V(0) = 0$

$$V(x_{k+1}) - V(x_k) < 0$$

Then the system is stable.

Among all possible functions $V(x)$ we decide to use the quadratic ones,$V(x) = x^\dagger P x$ [5], due to its simplicity, but this decision implies a pessimistic assumption, so not finding a quadratic $V(x)$ does not imply instability of the closed loop system.

[5] P is suposed to be a square symetric matrix, and to cope with the positiviness condition its eigenvalues should be greater than 0

6.1 Optimal Control in Networked control systems

We start from the optimal cost function, usually defined as

$$J = \int_0^\infty x^T(t)Qx(t) + x^T(t)Nu(t) + u^T(t)Ru(t)dt \tag{6.2}$$

The objetive is to minimize the integral using a discrete time controller, such that

$$u(t) = u(kh) = u_k \ \forall \ kh \leq t < (k+1)h \tag{6.3}$$

where h is the sampling period. The discrete verssion of this equation can be easyly[6] found to be

[6] find it in "Computer-Controlled Systems: Theory and Design", written by Karl Johan Astrom and Bjorn Wittenmark

$$J = \sum_{k=1}^{\infty} x^T(k)Q_d x(k) + x^T(k)N_d u(k) + u^T(k)R_d u(k) \tag{6.4}$$

where Q_d,N_d and R_d are computed as[7]

$$\begin{bmatrix} Q_d & N_d \\ N_d^T & R_d \end{bmatrix} = \int_0^h \begin{bmatrix} \Phi^T(\tau) & 0 \\ \Gamma^T(\tau) & I \end{bmatrix} \begin{bmatrix} Q & N \\ N^T & R \end{bmatrix} \begin{bmatrix} \Phi^T(\tau) & \Gamma(\tau) \\ 0 & I \end{bmatrix} d\tau \quad (6.5)$$

[7] In Matlab, you can edit the definition of *lqrd*, where these computations are done, to force the function to return the discrete verssions of the continuous counterparts

For delayed system we developed a state sapce model that was able to takle with the delays of the control signal, we call it the extended system. In this case, equation (6.4) is valid for a system which has no delays, the equivalent verssion of the ecuation for a delayed system is calculated as follows; we fix our attention in a single sampling interval, first we discretize the cost function for the first τ seconds (the delay), and then we discretize it for the remining time up to the next sample. The control signal and the nomenclature can be seen in figure (6.5).

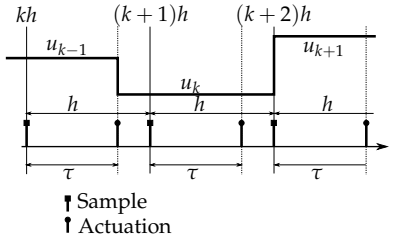


Figure 6.5: Sampling model of a delayed system

In our case the delayed model was expressed as

$$\begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} = \begin{bmatrix} \Phi(h) & \Phi(h-\tau)\Gamma(\tau) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} \Gamma(h-\tau) \\ 1 \end{bmatrix} u_k \quad (6.6)$$

Using the same tenich to discretize the cost function we get to the next expressions

$$J|_{kh}^{kh+\tau} = x^T(kh)Q_d(\tau)x(kh) + x^T(kh)N_d(\tau)u(kh-h) + u^T(kh-h)R_d(\tau)u(kh-h) \quad (6.7)$$

and

$$J|_{kh+\tau}^{kh+h} = x^T(kh+\tau)Q_d(h-\tau)x(kh+\tau) + x^T(kh+\tau)N_d(h-\tau)u(kh) + u^T(kh)R_d(h-\tau)u(kh) \quad (6.8)$$

Adding boths expressions, raplecing $x(kh+\tau) = \Phi(\tau)x(kh) + \Gamma(\tau)u(kh-h)$ and rearranging we find the cost for a sampling interval to be

$$\begin{aligned} J|_{kh}^{kh+h} = {} & x^T(kh)Q_d(\tau)x(kh) + x^T(kh)N_d(\tau)u(kh-h) + u^T(kh-h)R_d(\tau)u(kh-h) \\ & + [\Phi(\tau)x(kh) + \Gamma(\tau)u(kh-h)]^T Q_d(h-\tau)[\Phi(\tau)x(kh) + \Gamma(\tau)u(kh-h)] \\ & + [\Phi(\tau)x(kh) + \Gamma(\tau)u(kh-h)] N_d(h-\tau)u(kh) \\ & + u^T(kh)R_d(h-\tau)u(kh) \end{aligned}$$

all terms have a premultiplicative and a postmultiplicative element, which can be $x(kh)$, $u(kh)$ or $u(kh-h)$, if we group these elements we can rewite te cost in terms of the extended state space vector as

$$\begin{aligned} J|_{kh}^{kh+h} = {} & \begin{bmatrix} x_k^T & u_{k-1}^T \end{bmatrix} \begin{bmatrix} Q(\tau) + \Phi^T(\tau)Q_d(h-\tau)\Phi(\tau) & N_d(\tau) \\ N_d^T(\tau) & R_d(\tau) + \Gamma^T(\tau)Q_d(h-\tau)\Gamma(\tau) \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \\ & + \begin{bmatrix} x_k^T & u_{k-1}^T \end{bmatrix} \begin{bmatrix} \Phi^T(\tau)N_d(h-\tau) \\ \Gamma^T(\tau)N_d(h-\tau) \end{bmatrix} u_k + u_k^T R_d(h-\tau)u_k \end{aligned}$$

which may be simplified to

$$J|_{kh}^{kh+h} = (x_k^e)^T Q_d^k x_k^e + (x_k^e)^T N_d^k u_k + (u_k^e)^T R_d^k u_k^e$$

where

$$Q_d^k = \begin{bmatrix} Q(\tau_k) + \Phi^T(\tau_k) Q_d(h_k - \tau_k)\Phi(\tau_k) & N_d(\tau_k) \\ N_d^T(\tau_k) & R_d(\tau_k) + \Gamma^T(\tau_k) Q_d(h_k - \tau_k)\Gamma(\tau_k) \end{bmatrix} \tag{6.9}$$

$$N_d^k = \begin{bmatrix} \Phi^T(\tau_k) N_d(h_k - \tau_k) \\ \Gamma^T(\tau_k) N_d(h_k - \tau_k) \end{bmatrix} \tag{6.10}$$

$$R_d^k = R_d(h_k - \tau_k) \tag{6.11}$$

And h_k and τ_k are the sampling interval and the delay at each step, assumed to be different at each step.

The aoverall discrete cost function for a networked control system is defined as

$$J = \sum_{k=1}^{\infty} (x_k^e)^T Q_d^k x_k^e + (x_k^e)^T N_d^k u_k + (u_k^e)^T R_d^k u_k^e \tag{6.12}$$

if we fix h_k and τ_k to be the same for all k the we have the standard cost function for a delayed system. In our case, the values of h_k and τ_k are different for each step, so the standad way of computing an optimal controller does not hold. In any case if we assume that there is a fixed period and delay, the associated Riccati equation to the cost function (6.12) is given by

$$(\Phi^e)^T S \Phi^e - S - \left[(\Phi^e)^T S \Gamma^e + N_d\right] \left[(\Gamma^e)^T S \Gamma^e + R_d\right]^{-1} \left[(\Gamma^e)^T S \Phi^e + N_d^T\right] + Q_d = 0 \tag{6.13}$$

And the optimal controller is given by

$$K = ((\Gamma^e)^T S \Gamma^e + R)^{-1} ((\Gamma^e)^T S \Phi + N^T) \tag{6.14}$$

Now, if we use S as a Lyapunov quadratic function, namely $x^T S x$, the rate of variation of the lyapunov function on each step is given by

$$(\Phi^e + \Gamma^e K)^T S (\Phi^e + \Gamma^e K) - S \tag{6.15}$$

Even not beeing a proof of validity, we are going to replace as much terms as possible in this expressión to find the rate of change of this liapunov function in terms of Q_d^e, N_d^e, R_d^e and K, the controller