Networked Control Systems, Stability of non repeating sequences of matrices

Manel Velasco and Pau Martí

February 6, 2017

What is a mtatrix inequality?

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

$$A \prec B \tag{1}$$

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

$$x^T Ax < x^T Bx \tag{2}$$

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 (2) you simply get

$$\lambda x^T Ax\lambda < \lambda x^T Bx\lambda \tag{3}$$

$$x^T Ax < x^T Bx \tag{4}$$

which is again equation (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 (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

$$x = \begin{bmatrix} \cos(2\pi t) \\ \sin(2\pi t) \end{bmatrix} \tag{5}$$

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

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{6}$$

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

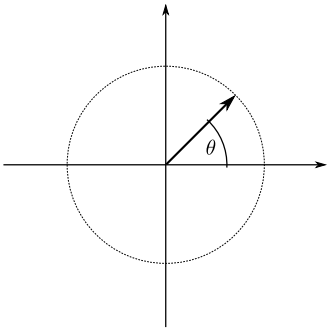

Figure 1: Unit circle vector

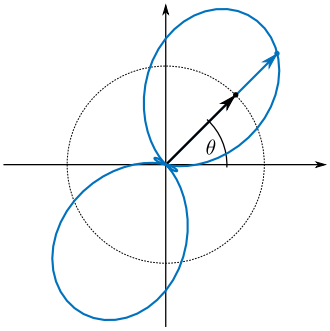

Figure 2: Reulting curve after computing $x^T Ax$ for all possible values of x in the unit circle. The blue line has been scaled to fit into the margin

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 (2) shows the results of these operations[2].

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

$$A \succ 0 \tag{7}$$

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].

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 (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.

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 $eig(A) \leq 0$

Linear Matrix inequalities

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

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i \succ \mathbf{0} \tag{8}$$

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

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

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

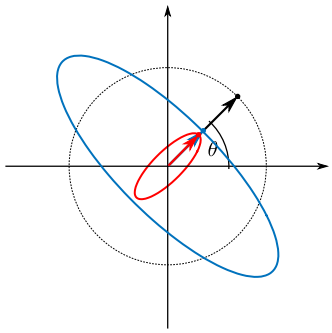

Figure 3: Example of $A \succ B$

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:

$$F(x) = F((1-\lambda)x_1 + \lambda x_2) = (1-\lambda)F(x_1) + \lambda F(x_2) \succ \mathbf{0} \qquad (9)$$

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 2.1. Most LMIs are not formulated in the standard form (8) but they can be rewritten as is shown in this example. Let us consider the following Lyapunov problem:

$$\begin{aligned} P = P^T &\succ \mathbf{0} \\ A^T P + PA &\prec \mathbf{0} \end{aligned} \qquad (10)$$

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:

$$P([x_1, x_2, x_3]) = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix} = x_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + x_3 \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = x_1 P_1 + x_2 P_2 + x_3 P_3 \qquad (11)$$

In order to include both equations of (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:

$$F(x) = x_1 \begin{bmatrix} P_1 & \mathbf{0} \\ \mathbf{0} & -A^T P_1 - P_1 A \end{bmatrix} + x_2 \begin{bmatrix} P_2 & \mathbf{0} \\ \mathbf{0} & -A^T P_2 - P_2 A \end{bmatrix} + x_3 \begin{bmatrix} P_3 & \mathbf{0} \\ \mathbf{0} & -A^T P_3 - P_3 A \end{bmatrix} \qquad (12)$$

which is in the form as in (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 $n \times n$.

Solving LMIs

To solve a LMI you have two chances, the first one is to write it explicitily as seen in example (2.1), 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 3.1. 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

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} x \tag{13}$$

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[4].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

$$A^T P + PA \prec 0 \tag{14}$$

We decide to create our matrix as

$$P = \begin{bmatrix} x & y \\ y & z \end{bmatrix} \tag{15}$$

So equation (**??**) becomes

$$\begin{bmatrix} 0 & -1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x & y \\ y & z \end{bmatrix} + \begin{bmatrix} x & y \\ y & z \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \prec \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{16}$$

which can be written explicitly as

$$\begin{bmatrix} -2y & x-2y-z \\ x-2y-z & 2y-4z \end{bmatrix} \prec \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{17}$$

In order to be able to do the computations[5] we may fix how samll should this matrix be. For instance, we know that

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \prec \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{18}$$

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

[5] 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

$$\begin{bmatrix} -2y & x-2y-z \\ x-2y-z & 2y-4z \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{19}$$

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 [6] wold be to add a restriction in terms of an optimization, for instance

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

$$\begin{aligned} \underset{x,y,z}{\text{minimize}} \quad & \text{Trace}(P) \\ \text{subject to} \quad & A^T P + PA \prec 0 \end{aligned} \tag{20}$$

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

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[7], an interface to a set of numerical solvers. You must install Yalmip side by side with the recomended numerical solvers, at least SEDUMI[8]. Folow installation instructions in the web sites to get a functional copy of both in your Matlab distribution.

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

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

Once the solvers are installed we are in conditions to solve a simple problem, for instance the one given in example (3.1), equations (23)

Example 3.2. Recovering the problem... Prove that the system driven by

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} x \tag{21}$$

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

$$A^T P + PA \prec 0 \tag{22}$$

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

$$\begin{array}{lll} \underset{x,y,z}{\text{minimize}} & \text{Trace}(P) & (23) \\ \text{subject to} & A^T P + PA \prec 0 & \end{array}$$

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
```

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 4 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 5 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) 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[9] that stabilizes both dynamics, it could happen that there appears a destructive sequence (destructive in the sense of unstable).

Example 4.1. 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

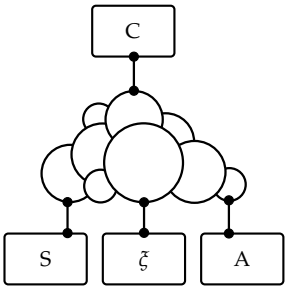

Figure 4: Networked system with an extra node injecting messages into the network

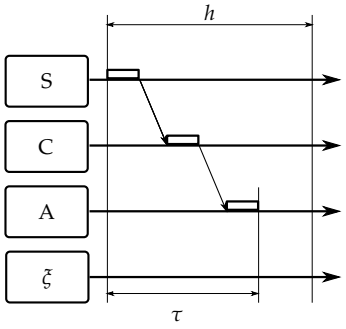

Figure 5: Hypotetic timeline 1, the extra node does not interference into de loop communication

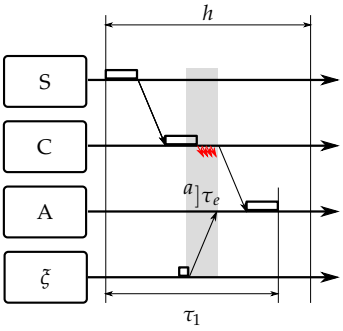

Figure 6: Hypotetic timeline 2, the extra node interference into the loop communications randomly, producing an estra delay τ_e

[9] As explained in the part "Known sequences"

$$x_{k+1} = \begin{bmatrix} 0 & h & h\tau_1 - \frac{\tau_1^2}{2} \\ 0 & 1 & \tau_1 \\ 0 & 0 & 0 \end{bmatrix} x_k + \begin{bmatrix} \frac{(h-\tau_1)^2}{2} \\ h - \tau_1 \\ 1 \end{bmatrix} u_k$$
$$x_{k+1} = \Phi_1 x_k + \Gamma_1 u_k$$

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[10] that may develop an infinite set of sequences. Figure 7 shows its representation.

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

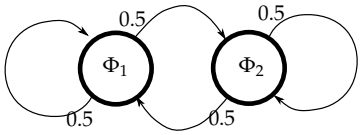

Figure 7: This is the graph associated with the example we are proposing, the probability of having Φ_1 after happening Φ_2 is 0.5, while the reverse case is symmetric. In this situation you cannot predict the sequence of the matrices neither a tendency in the sequences that we get in the execution

---

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.

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.

Defining the set of matrices

The set of matrices is given[11] by the possible combinations of h and τ

$$\Phi_{cl_i} = \Phi(h_i, \tau_i) + \Gamma(h_i, \tau_i) K$$
$$i \in 1, 2, \ldots, n$$
$$(h, \tau)_i \in (h, \tau)_1, (h, \tau)_2, \ldots, (h, \tau)_n \tag{24}$$

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

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 $V(x) > 0 \ \forall \ x - \{0\}$ [12] such that

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

Then the system is stable.

Among all possible functions $V(x)$ we decide to use the quadratic ones,$V(x) = xPx$ [13], 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.

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

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

Optimal Control in Networked control systems

We start from the optimal cost function, usually defined as

$$J = \int_0^\infty x^T(t)Qx(t) + x^T(t)Nu(t) + u^T(t)Ru(t)dt \tag{25}$$

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

$$u(t) = u(kh) = u_k \ \forall \ kh \leq t < (k+1)h \tag{26}$$

where h is the sampling period. The discrete verssion of this equation can be easyly[14] found to be

$$J = \sum_{k=1}^{\infty} x^T(k)Q_d x(k) + x^T(k)N_d u(k) + u^T(k)R_d u(k) \tag{27}$$

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

where Q_d,N_d and R_d are computed as[15]

$$\begin{bmatrix} Q_d & N_d \\ N_d^T & R_d \end{bmatrix} = \int_0^h \begin{bmatrix} \Phi^T(\tau) & 0 \\ \Gamma^T(\tau) & I \end{bmatrix} \begin{bmatrix} Q & N \\ N^T & R \end{bmatrix} \begin{bmatrix} \Phi^T(\tau) & \Gamma(\tau) \\ 0 & I \end{bmatrix} d\tau \tag{28}$$

[15] 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 (27) 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 (8).

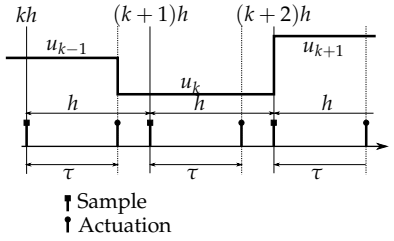

Figure 8: Sampling model of a delayed system

In our case the delayed model was expressed as

$$\begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} = \begin{bmatrix} \Phi(h) & \Phi(h-\tau)\Gamma(\tau) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} \Gamma(h-\tau) \\ 1 \end{bmatrix} u_k \qquad (29)$$

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

$$J|_{kh}^{kh+\tau} = x^T(kh)Q_d(\tau)x(kh) + x^T(kh)N_d(\tau)u(kh-h) + u^T(kh-h)R_d(\tau)u(kh-h) \qquad (30)$$

and

$$J|_{kh+\tau}^{kh+h} = x^T(kh+\tau)Q_d(h-\tau)x(kh+\tau) + x^T(kh+\tau)N_d(h-\tau)u(kh) + u^T(kh)R_d(h-\tau)u(kh) \qquad (31)$$

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

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

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

$$Q_d^k = \begin{bmatrix} Q(\tau_k) + \Phi^T(\tau_k)Q_d(h_k-\tau_k)\Phi(\tau_k) & N_d(\tau_k) \\ N_d^T(\tau_k) & R_d(\tau_k) + \Gamma^T(\tau_k)Q_d(h_k-\tau_k)\Gamma(\tau_k) \end{bmatrix} \qquad (32)$$

$$N_d^k = \begin{bmatrix} \Phi^T(\tau_k)N_d(h_k-\tau_k) \\ \Gamma^T(\tau_k)N_d(h_k-\tau_k) \end{bmatrix} \qquad (33)$$

$$R_d^k = R_d(h_k-\tau_k) \qquad (34)$$

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

$$J = \sum_{k=1}^{\infty} (x_k^e)^T Q_d^k x_k^e + (x_k^e)^T N_d^k u_k + (u_k^e)^T R_d^k u_k^e \tag{35}$$

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 (35) is given by

$$(\Phi^e)^T S \Phi^e - S - \left[(\Phi^e)^T S \Gamma^e + N_d\right] \left[(\Gamma^e)^T S \Gamma^e + R_d\right]^{-1} \left[(\Gamma^e)^T S \Phi^e + N_d^T\right] + Q_d = 0 \tag{36}$$

And the optimal controller is given by

$$K = ((\Gamma^e)^T S \Gamma^e + R)^{-1} ((\Gamma^e)^T S \Phi + N^T) \tag{37}$$

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

$$(\Phi^e + \Gamma^e K)^T S (\Phi^e + \Gamma^e K) - S \tag{38}$$

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