TÉCNICO LISBOA

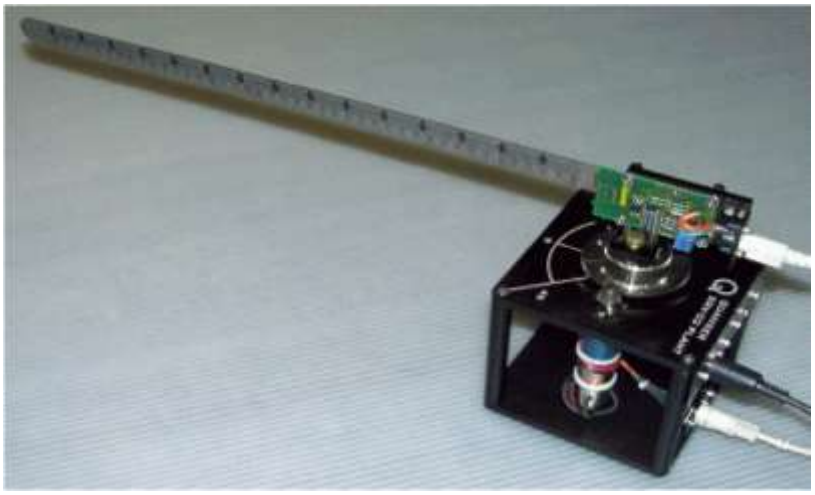
*Master Degree in Electrical and Computer Engineering*

**2016/2017 – Winter Semester**

# Computer Control (Controlo por Computador)

**LABORATORY WORK**

## *Identification and Computer Control of a Flexible Robot Arm Joint*



Prepared by

**João Pedro Gomes, Alexandre Bernardino and João Miranda Lemos**

**Instituto Superior Técnico**

**Department of Electrical and Computer Engineering**

**Scientific Area of Systems, Decision and Control**

## Translation to Portuguese of selected words

Arm – *Braço*

Comb – *pente*

Gear box – *caixa de desmultiplicação*

Motor shaft – *veio do motor*

Plant – *instalação (a controlar)*

Strain gage – *extensómetro*

Tip – *ponta*

## Instruction on the reports and software to develop

The students must submit 2 reports:

- The first report corresponds to the model identification and validation;
- The second report corresponds to control design and testing.

The language of the report may be either Portuguese or English.

In each report the student should answer the questions marked with ✍. The end of the question is indicated by the symbol ✋. The symbol 💻 corresponds to tasks to be performed during the laboratory sessions.

Each of the two reports must be written in a text processor and must indicate:

- The number and name of the students that author the report.
- The part of the work to which it refers (either 1 or 2).
- The number of the questions addressed as indicated below in this guide (Q1, Q2, …).

The answers must be direct, concise and short, but insightful and technically correct.

Both the reports and the software must be sent to the professor in charge of the laboratory within the prescribed time limits.

The reports and the software developed must be original and the data used must be actually obtained by the students that sign the report. All forms of plagiarism or copy detected will be punished without contempt, according to IST and UL regulations, and the Portuguese Law.

## Read this with attention!

### Safety warnings

- To avoid being injured, all the students should be at a safe distance from the flexible bar to avoid being hit if it turns.
- When performing tests, one student must always have a finger close to the power amplifier switch to stop the bar if it starts running.
- In order to avoid malfunctions, never allow the bar to rotate over itself.

## Leia isto com atenção!

### Avisos de segurança

- Para evitar ser feridos, todos os alunos devem estar a uma distância de segurança da barra flexível por forma a não poderem ser atingidos se esta rodar.
- Quando se executam os testes, deve haver sempre um aluno com o dedo sobre o interruptor do amplificador de potência por forma a parar a barra se esta começar a rodar.
- Por forma a evitar avarias, nunca deixe a barra rodar sobre si própria.

**Notice also:**

- You must bring to the lab a flash drive to store the results obtained during the class sessions;
- When using the lab computers make sure that you are using a working folder with write permissions (can be an external flash drive).

## List of symbols and units

$D$ [m] → Distance of deflection of the bar.

$L$ [m] → Bar length.

$\theta$ [degree] → Angle of rotation of the motor shaft with respect to a "zero" direction.

$\alpha$ [rad] or [degree] → Angle of deflection of the flexible bar with respect to the motor shaft angular direction.

$\omega$ [degree/s] → Motor angular speed.

$\theta_e$ [V] → Electrical tension yielded by the sensor of $\theta$.

$\alpha_e$ [V] → Electrical tension yielded by the sensor of $\alpha$.

$\omega_e$ [degree/s] → Electrical tension yielded by the sensor of $\omega$.

$K_b$ [degree/V] → Flexible bar deflection transducer constant.

$K_p$ [degree/V] → Motor shaft angular position transducer constant.

$K_t$ [degree/s/V] → Tachometer transducer constant.

$y$ [rad] → Total angular position of the tip of the flexible bar tip with respect to the "zero" direction. $y = \theta + \alpha$.

## 1.Objective

The objective of this work consists of the identification and computer control design for the position of a flexible joint of a robot arm (a flexible bar). The work proposed here is in the spirit of Control applied to Cyber-Physical Systems: A computer is attached to a physical plant (a flexible robot arm joint) to modify its physical behavior through the interaction with the computational part (the computer algorithm).

The work consists of two parts. In part 1 a plant model that relates the motor that drives the arm with the angular position of its tip is obtained using System Identification methods and data obtained from experiments performed with the plant. In part 2, a controller is designed using the previously obtained model in order to drive position the tip of the robot joint to a desired angle. The controller is then tested on the actual plant.

## 2.Plant description

The plant is a single joint of a flexible robot arm and consists of a DC motor, driven by a power amplifier, whose shaft is solidary with one extremity of a flexible bar (the joint). Figure 1 below shows a picture of the physical system (the "plant") to control.

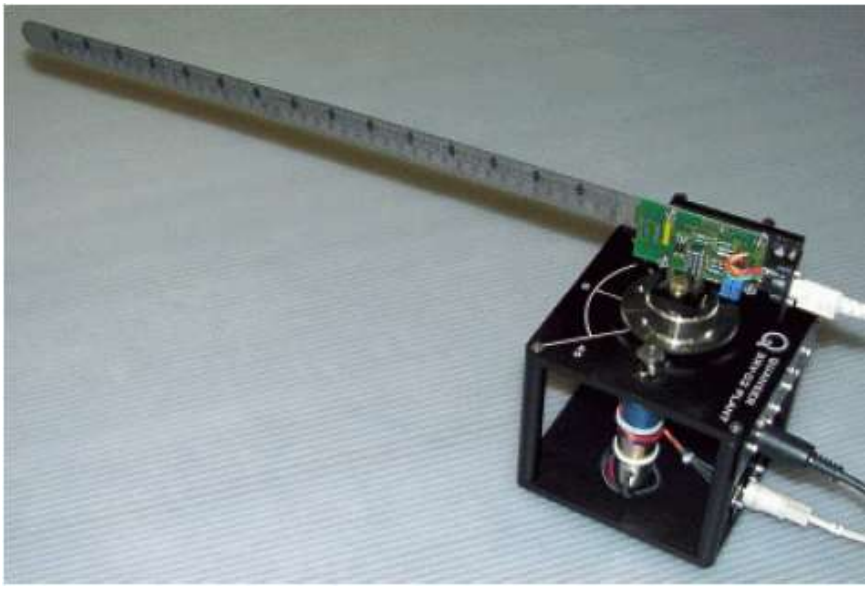


Figure 1. The physical system (plant) to control.

The *control objective* is to actuate on the motor such that the tip of the bar tracks a specified angle.

**Q1** ✍ Describe reasons that cause this control objective to be nontrivial. (hints: Is it possible to develop a table of electric tensions to apply to the DC motor such that the bar tip moves to a constant position? What happens if a constant electric tension is applied to the motor? Does feedback with a controller made of a single gain amplifying

the tracking error solve the problem? Think about the dynamics of the motor and the bar. What is a plausible rough distribution of the open loop plant poles?).

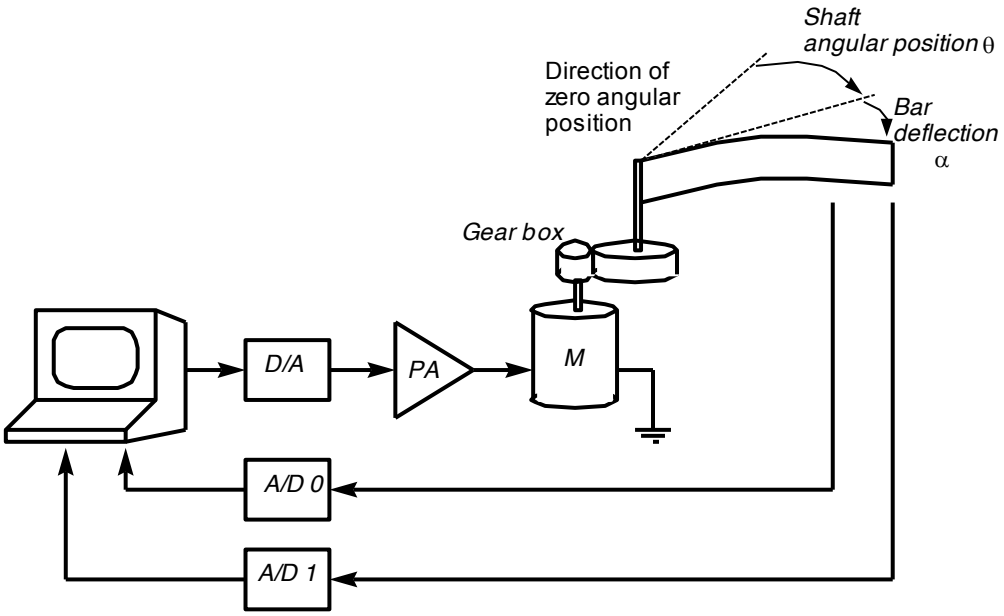



Figure 2. Schematic view of the hardware of the plant interconnected with the computer control system. The motor *M* (including a gear box) is driven by the output of a power amplifier *PA* and moves the flexible bar. The computer is interconnected with the plant using AD and DA converters.

Figure 2 shows a schematic view of the interconnection of the plant (power amplifier *PA*, DC motor with gearboxes *M*, and the flexible bar) with a computer.

The computer receives information about the angular position of the flexible bar tip (given by the sum of the motor shaft angular position and the bar deflection) through the AD converters. At each sampling time, an algorithm embedded in a computer program reads this information and computes the value of the electric tension to apply to the motor. This value is then applied through a D/A converter to the power amplifier PA that drives the motor.

***Measuring the bar deflection***

The flexible bar is instrumented with a strain gage (figure 3) that measures the deflection of the bar. The strain gage consists of an electrical resistance whose value changes when its length varies. When the bar is deflected, one of its faces is slightly stretched, while the other is compressed, the changes being approximately proportional to the bar tip deflection. For further details on strain gage sensors see

http://www.omega.com/literature/transactions/volume3/strain.html

Figure 3 below shows a detail of the flexible bar, indicating the mounting of the strain-gage (red encirclement).


Figure 3. Detail of the flexible bar showing the mounting of the strain-gage (indicated by the red encirclement). The electronic circuit at the fixed part of the bar contains the Wheastone bridge used to measure the electrical resistance of the strain gage.

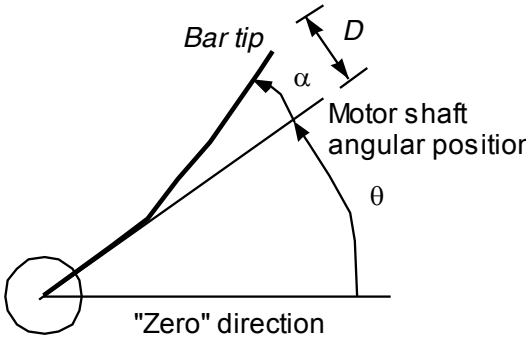



Figure 4. Schematic model of the flexible bar deflection.

With respect to the circular movement of the motor and the tip of the flexible bar, skwtched in figure 4, consider the following quantities:

- $L$ [m], the bar length;
- $D$ [m], the length of the deflection of the bar tip;
- $\alpha$[rad], the angular deviation of the bar tip with respect to the motor shaft direction;
- $\theta$ [rad], the angle of the motor shaft direction with respect to a “zero” direction.

Recall from basic Geometry that the value of an angle in radian (rad) is given by the length of the corresponding arc of a circumference of radius of length to 1 in the length unit used. Therefore, the relation of the angular deflection of the bar tip with its deflection (measured along a straight line) is approximately given by

$$\alpha = \frac{D}{L} \qquad \text{(rad)}.$$

In practice, it is preferable to work in (degree) instead of (rad) units. For that sake, observe that the conversion between the same measure expressed in these units is made by

$$\alpha_{[degree]} = \frac{180}{\pi}\alpha_{[rad]}.$$

Hereafter we will always use the angles expressed in degrees.

> *The issue of using the right units is fundamental. A space probe sent to Mars was lost because part of the design team was using centimeters as unit length, while the other part was using inches…*

The total angle that defines the position of the tip of the flexible bar is

$$y = \theta + \alpha \quad \text{(degree)}.$$

***Flexible bar deflection transducer***

The angle sensors provide an electric tension that is an image of the corresponding angle. In the case of the flexible bar, the sensor provides an electrical tension $\propto_e$ that is related to the actual deflection angle $\alpha$ by

$$\alpha = K_b \alpha_e,$$

where $K_b$ is a constant with units [degree/V].

Figure 5 shows a block diagram that relates the physical variable $\alpha$ (angle) with its electrical tension image $\propto_e$.

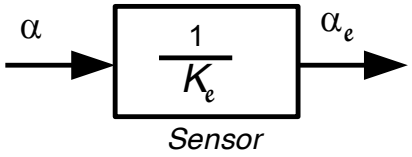



Figure 5. Relation between one angle and its measure given by the sensor.

In order to estimate the value of the constant $K_b$, a sequence of experiments is done, each experiment consisting of deflecting the bar by a known quantity and reading the corresponding electrical tension.

In order to perform these experiments, the bar is rigidly attached to a support stand as shown in figure 6. In the end of the stand there is a calibration "comb" with several slots separated by $1/4$ inch (1 inch = 2,54 cm) that, as shown in figure 7, allows to deflected the bar by known quantities.

In the “ideal” situation, the electrical tension yielded by the sensor would be exactly proportional to the deflection angle. In practice, this is not true because:

- The relation is not exactly linear;
- There are small random fluctuations from experiment to experiment.

Figure 6. The flexible bar mounted on its calibration rig.

Figure 7. The calibration “comb” for two different bar deflections: In rest, with no deflection (left) and deflected to the left by one slot (right).

For these reasons one should:

- Characterize the deviations with respect to the linear behaviors;
- Make several experiments.

*It is remarked that the linear dependency of electrical tension on deflection is only expected for relatively small deflections. Large deflections may cause a permanent deformation of the bar and should of course not be done.*

Due to the deviations with respect to the ideal linear behavior, the value of the constant $K_b$ corresponds to an average behavior. As such, this constant is estimated from the set of data obtained in the different experiments using the least squares algorithm.

The way to read the electrical quantities will be explained below in detail.

***Shaft angle transducer***

The shaft angle is measured with a rotation potentiometer whose axis is rigidly connected to it. Figure 8 shows the internal scheme of the potentiometer being used.

A cursor (3), solidary with the potentiometer axis (4) (and therefore also with the motor shaft), slides over a coiled resistive wired (1). When the cursor moves, the resistance varies in a way that is proportional to the angle of the contact. An electric circuit transforms this resistance in an electric tension value. Indeed, the resistive element is powered at the extremes by a voltage of +5V (point 5) and -5V (point 9) and therefore, depending on the cursor position, the output voltage has a value between -5V and +5V.

It is remarked that, since there are no mechanical stops at the extremes of the resistance (end of course), the measured voltage repeats itself every whole rotation of the cursor.

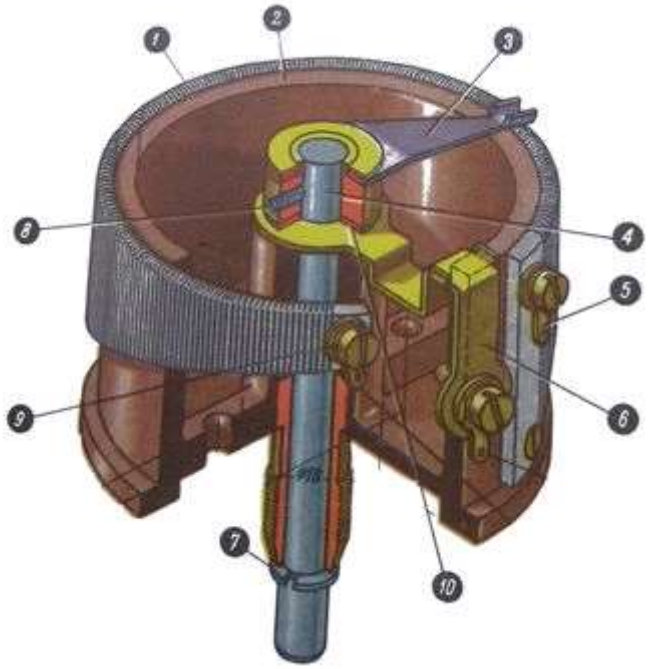



Figure 8. The internal scheme of the potentiometer used to measure the shat angle.

Therefore, the shaft angle $\theta$ is approximately given as a function of the sensor output $\theta_e$ by

$$\theta = K_p \theta_e,$$

where $K_p$ is a constant with units [degree/V].

It is also remarked that, due the coiled structure of the resistance, there is a quantization effect that causes the resistance to vary in a staircase form when the angle varies.

Before proceeding to the estimation of the transducer constants from plant data, the use of the data acquisition and control software is explained.

## 3.Software for data acquisition and control

To build a plant model for controller design we need to perform experiments on the plant and record data from them. Model parameters are then estimated from these data.

To carry out data acquisition (reading data from the plant sensors connected to the A/D converters) and process control (sending commands to the actuator, connected to the D/A converter) the software "MATLAB" + "Simulink" with toolboxes "Real-time Workshop" + "Real-time Windows Target" will be used.

With "Simulink" it is possible to design data acquisition and control systems through a simple graphical interface and with plenty of features. The "Real-time Workshop" allows to transcript Simulink diagrams to C code and compile it to a target platform in a way that is transparent to the user. Finally, the "Real-time Windows Target" allows running real-time models in a computer with the Windows operating system and synchronize the “time of the computer” with the “real time” of the plant.

Although not essential to complete this work, the student is advised to consult the manuals of these software tools, made available on the website of the course. In the following lines, a brief description of the software to be used is presented.

It is assumed that the student has a basic knowledge of MATLAB and Simulink, that can be easily acquired either in the previous course on Modelling and Simulation or by following one of the many tutorials available in the Web.

***Simulink***

Simulink is a simulation environment that works in conjunction with Matlab and allows simulations of dynamical systems in non-real time (simulated mode). There are several component libraries (blocksets) that can be interconnected in easy and intuitive ways, similar to a block diagram

In particular, there are components that allow to simulate dynamic systems, signal generators, signal viewers and communicate with various I/O card, including the ones to be used in this work.

However, since Simulink by itself does not allow to carry out simulations with a synchronously real clock, it is necessary to rely on the Real-time Workshop "Tools" and "Real-time Windows Target" kernel when using blocks that interface with A/D and D/A converters. See the user manual of Simulink, available on the website of the discipline.

***Real-Time Workshop***

The Real-Time Workshop tool allows to transcript automatically Simulink models to C code adapted to diverse hardware and software platforms. The code, when compiled, generates an executable program that run the Simulink model in a separate process, without graphical interface.

It is however possible to save and display the signals available in the model, as well as to modify the properties of the blocks, using the Simulink external mode. In this mode, the Simulink can communicate with the process executable to get/send data and reconfigure the parameters of the simulation.

***Real-Time Windows Target***

With the Real-Time Windows Target tool, the C code generated by Real-Time Workshop can be compiled for the Windows platform and run in real time.

For a good understanding of the process of creating code in real time, using all the tools behind described, it is essential to consult the user manual of the Real-time Windows Target, available on the website of the discipline. We recommend in particular reading the chapter "Basic Procedures". Nevertheless, it is remarked that reading these documents is not essential to perform this project.

## 💻 Tasks to be performed at the laboratory

***Number of 3h laboratory sessions needed to complete this task: 1***

***Objectives:*** *At the end of the lab session the following elements should be available:*

1. *SIMULINK block diagram to impose command from the motor and to read signals from the plant;*
2. *Numerical values of the constants that relate the electrical tensions read at the A/D converters with the corresponding physical variables (motor shaft rotation angle and rotation of the tip of the bar).*

***Homework to prepare the lab session***

1. *Read sections 1, 2 and 3 of this lab guide.*
2. *Review your knowledge of basics with MATLAB and SIMULINK. You may find it useful to follow one of the many excellent tutorials found in the Web.*

This introductory session is formatted as a tutorial. As such, few previous knowledge of MATLAB/Simulink will be required. All the necessary steps to configure the simulations are quite detailed.

Since some of the procedures are repeated several times throughout the work, we strongly advise the students to draw up a "check-list" based on the tasks performed, synthesizing the procedures required to configure a real-time simulation, configure the recording of data in the workspace, configure the elements viewers, etc. With these working procedures, work efficiency will be greatly increased.

**A – Generation, viewing and recording signals.**

1. **Open the SIMULINK**. Open the MATLAB (version 8, R2015a) and run the command <<simulink>>. The window "Simulink Library Browser" will be opened.
2. **Create a new model**. In the menu "Simulink Library Browser", choose File-New-Model, or press CTRL + N, or click on the Simulink button in the top bar of the MATLAB window. This opens a graphical blanch window where you will enter and connect the various operational blocks by dragging from a menu as explained next.
3. **Insert and configure a block signal generator**. In the selection tree to the left of the window "Simulink Library Browser" choose "Simulink-Sources". Drag an element "Signal Generator"  for the model window. Open the element (double click) and configure it to generate a square wave 2 volt amplitude and frequency 2 Hz
4. **Insert and configure a block signal Viewer**. In the selection tree choose "Simulink-Sinks" and drag an element "Scope" to the model window. Open the Scope block, and click the toolbar "Parameters" (second from left). In the tab "General" configure "Tick Labels-All", "Sampling-Sample Time" and choose as sampling period 1 millisecond (0.001). In the tab "History" remove "Limit date selection points to last" and select "Save data to workspace". The data presented in this element will be exported to the MATLAB workspace at the end of the simulation. Choose the name of the variable "input" and data format “Structure with time".
5. **Configure and run the simulation**. Connect the "Signal Generator" element to the element "Scope". Start the simulation ("Simulation-Start" menu, or CTRL-T, or press the button  ). Observe the result of the simulation in the viewer element "Scope"
6. **Showing signals in workspace**. Confirm that the workspace of MATLAB has the "input" variable. You can do this with by typing “who” in the MATLAB command line. All the existing variables will be listed. Access the signal with the command "plot" and confirm that this signal has an identical shape to the one observed in the "Scope":

```
>> plot (input.time, input.signals.values)
```

7. **Save data from the workspace**. Save the data file (see how the command "save" operates with the help command: >> help save) in order to generate the graph "offline.

**B – Sending commands to the motor.**

8. **Insert a block D/A to send commands to the motor**. In the selection tree select "Simulink Desktop Real-time" and drag the "Analog Output" element to the window you have created for your project.
9. **Configure the block D/A**. To see which data acquisition card is in use at the laboratory bench, run the utility "Measurement & Automation" that you can find on the Desktop. In the directory tree that appears on the left side of the main window, choose "Devices and Interfaces"/"NI-DAQmx Devices", showing the type of graphics card installed (e.g. NI PCI-6221 or PCI-MIO6040E). Make a note of the type of card. Then go back to Simulink, open the D/A block and select the appropriate board (most probably, the correct one is the only one listed). Set the sampling period (sample time) to 1 millisecond (0.001). Set the output channel 1. Voltage ranges should be configured between -10 and 10V and the data type as "volts". The initial and final values must be set to 0. *For safety reasons, it is quite important that you set the final value to 0 (zero). This is the value that remains at the output of the D/A board after the experiment is ended. It the final value is not zero, the motor will start running in an endless way in one direction and the equipment may be damaged.* Connect the signal generator to this block.
10. **Configure the simulation to work in real-time**.
    a. In the menu "Simulation" select "Configuration Parameters" or CTRL-e. This opens a dialog window.
    b. In the Solver option-> Type choose “Fixed-step".
    c. In “Code Generation” set System Target File: browse to choose “rtwin.tlc”
    d. In “Hardware Implementation” set:
        i. Device vendor: Intel
        ii. Device type: X86/Pentium
11. **Configure Simulink external mode**. In the menu "Code" choose "External Mode Control Panel". This opens a dialog window. Select "Signal & Triggering". This opens a new window. In "Duration" enter the number of total samples required for the simulation, in this case 10 sec X 1000 samples/sec = 10000 samples. Check that simulation mode is “External” by looking at the Simulink command bar (see figure 9). If not, go back to the model window and, in the menu "Simulation", select "External".


Figure 9. Simulink command bar.

12. **Save the project to disk.** Save the project to a file in the disk (e. g., at the desktop) by using the group “file” in the Simulink task bar. Remember also to record your work to a personal mass storage device, such as a pen that you carry with you. Please clean the files you produced at the end of the lab session.
13. **Make an experiment of sending command signals to the motor**. Ask the laboratory staff to check whether everything is correct, and start the operation by pressing the button  . Use the button ■ to stop. Observe the motor operation and register its behavior.

**C – Reading signals from the sensors.**

14. **Insert A/D block to read the values of the sensors**. In the selection tree select "Simulink desktop real-time" and drag one block "Analog Input" to the project template.
15. **Configuring the A/D block**. Open the block and choose the acquisition card installed in the machine. Set the sampling period (sample time) to 1 millisecond (0.001). Set the input channel 1 (potentiometer) to the block. Voltage ranges should be configured between -10 and 10V and the data type as "volts".
16. **Insert one block "Scope" to show the signal of the sensor**. Configure it as in point 4. Give it the name "tensao_pot".
17. **Run the experiment**. Confirm with the teacher that everything is well before beginning the experiment. Observe and register the graphics of the time evolution of the potentiometer voltage and of the tachometer. Make sure that the vector variables "tensao_pot" and "tensao_taq" are available in the workspace of Matlab and save them to a file.

**D – Sensor Calibration.**

18. **Calibrate the potentiometer sensor model**. Apply a step signal (1 volt) to the motor. Acquire and record the signals obtained by the potentiometer. From the data obtained, calculate the calibration factor $K_p$ (degree/volt), that make it possible to convert sensor measurements in volt for degrees.
19. **Calibrate the bar deflection model**. Similarly, make a set of experiments to estimate the calibration factor $K_b$ (degree/volt) for the bar deflection. Use the procedure described in relation to figures 6 and 7.

**Q2 ✍ Write a report about interfacing the plant with the computer**

The report must include:

- **Comment on the motor operation**. Explain in particular why their angular position never stabilizes when its command signal is constant in the open loop conditions described.
- **Explain the procedure carried out to perform sensor calibration**. Include in the report:
    - The Simulink block diagram used;
    - Plots of the sensor and command signals (don't forget to label the variables used in each axes and the units used!);
    - The auxiliary calculations and/or the graphics used to compute the conversion factors $K_p$ (degree/volt) and $K_b$ (degree/volt).
    - Plot on the same graphic the experimental points obtained for the bar deflection and the linear relation assumed to hold between the bar deflection and the sensor electric tension for the value of $K_b$ that you have estimated.

*Important: When writing your reports, do not forget to display the units in both axes of graphs and to write appropriate captions to identify all the graphs and diagrams, explaining the experimental conditions to which they refer.*

# 4. Plant model identification.

Plant model identification aims at producing a model to be used for control design. For that sake, the following steps are performed:

1. Perform an experiment to collect data;
2. Process the data to remove undesirable components (for instance, to remove an off-set or to filter high-frequency noise);
3. From the processed data identify a model in the form of a difference equation (an ARMAX model). This task comprises:
    a. Select (following a trial and error method) the best model orders (number of poles and zeros);
    b. Estimate the model parameters
4. Convert the ARMAX model to an equivalent state model.

Data collection is performed using the Simulink block diagrams build as explained in the previous lab session (chapter 3 of this document).

A key aspect for the success of your work consists of the experimental conditions under which data is obtained. In this respect the following issues are quite important:

- There is a trade-off in the selection of the amplitude of the signal applied to excite the motor.
    - The amplitude cannot be too low because otherwise the gearbox backlash (*folga da caixa de desmultiplicação*) prevent the bar to move;
    - The amplitude cannot be too high because there may be nonlinearity effects that distort the signal with respect to linear behavior.

    We suggest 1V for the amplitude or values around it.

- The duration of the experiment may not be too long because, since the electrical tension applied to the motor is not perfectly symmetrical, the bar has a tendency to rotate with a drift to one side, dragging the cable connected to the strain-gage, a fact that must never occur. We suggest a few tens (*algumas dezenas*) of seconds for the duration of an experiment. We also suggest that the initial points are discarded, for instance the first 5 or 10 seconds.
- If you excite the motor with a square wave, you should record the data corresponding to several cycles (at least 2 or 3) of this signal. The exciting signal bandwidth must be carefully selected: The exciting signal may not be too fast (the plant filters it and is not able to respond in the frequency range where its dynamics is dominant), and may not as well be too slow (since the transients are not excited). We suggest frequencies for the excitation signal of about 0,4Hz. Consider different frequencies and different types of waveform for the

excitation signal, in addition to the square wave (for instance a PRBS). You can generate a PRBS signal in MATLAB with the command *idinput*

- A major issue is the value of the sampling period. Use 20 ms = 0,020 s, that corresponds to a sampling rate of 50 Hz.

It is suggested that the data of the different experiments performed is placed in files of type .mat. You can organize your software in three different MATLAB .m script files that perform the required tasks as follows:

- **Script for model identification**. Data processing, ARMAX model identification from plant data, and conversion to state model is performed in one file (of type .m). This file starts by reading plant data stored in a .mat file and produces as output the matrices that define the state model that become available in the MATLAB Workspace.
- **Script for controller design.** Another .m script file uses these matrices to design the gains of a state feedback controller.
- **Script for controller testing**. A third script uses the matrices that define the state model and the controller gains to parameterize and call a Simulink diagram connected to the plant to perform the tests in closed loop.

Hereafter we describe a way to encode the first script (for identification). You may use the lines of code suggested in a .m file of your own to identify the model from plant data.

It is assumed that the data (obtained by performing plant experiments and following the data acquisition procedure described in Section 3 of this guide) is stored in a data structure generated by a SIMULINK block of type *scope*. By default, the data is thus stored in a data structure named ScopeData (the name of this variable may be changed in the scope menu, tab “History”; actually, in the first lab session you were told to use the name “input”) and may be accessed by the MATLAB commands

```
t = ScopeData.time;
sigs = ScopeData.signals.values;
```

In the above code, t is a vector that contais as entries the sampling instants in continuous time units (s), and sigs is a matrix with 3 columns, each corresponding to a variable, and a number of rows that is equal to the number of data points:

- The first column contains the samples of the exciting signal applied to the motor;
- The second column contains the signal issue by the potentiometer that yields the motor shaft angle;
- The third column contains the strain gage measurements.

Off course, you may change the column orders. To use variables with more compact and appealing names you may use the code:

```
utrend = sigs(:,1); % Entrada -  Input signal
thetae = sigs(:,2); % Potenciómetro  -  Potentiometer signal
alphae = sigs(:,3); % Extensómetro  - Starin gage signal
```

The position of the bar tip may then be computed using:

```
% Reconstrução ângulo total da barra – Computation of total bar angle
ytrend = thetae*Kp + alphae*Ke;
```

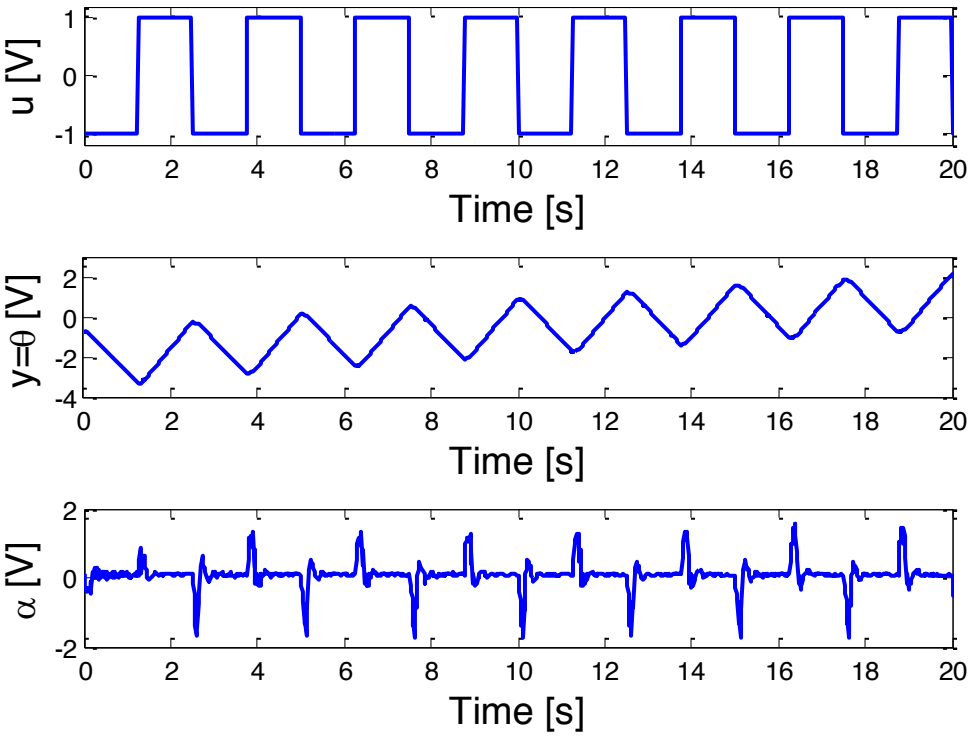



Figura 10. Example of the signals registered in one trial..

An example of a record of the above signals may be seen on figure 10. It is remarked that the signal that corresponds to the motor shaft angle steadilt grows in time. This is due to the fact that the input tension signal has a non-zero average value, and there is an integrator in the motor, given by the relation between the angular speed and the angular position of the motor shaft. In order to get good identification results, this integral effect must be removed before applying the identification methods described hereafter.

Therefore, to build a model for the motor shaft angle dynamics, one must:

1. Remove the integral effect from the motor shaft angle data;
2. Identify the model that relates motor electrical excitation with the motor shaft angle;
3. Add an integrator to the model (pole at 1).

Step 1 above is performed by differentiating the motor shaft angle data. Since differentiation increases high frequency noise (Why? Think in terms of the transfer function of the derivative; for simplicity, consider the continuous time case) a low pass filter must be added to attenuate high frequency noise.

Both operations (differentiation and filtering) may be done with the following MATLAB code:

```
af = 0.8;
Afilt = [1 -af];
Bfilt = (1-af)*[1 -1];
% Filtragem seguida de eliminação de tendências
% Filtering and detrending
yf = filter(Bfilt,Afilt,ytrend);
```

It is a good exercise to write the mathematical expression of the filter and to see from its poles and zeros what are the mathematical operations that this filter performs on data.

In addition to the above code, one should remove the tref (average value) of the input signal. It is also useful to remove the trend from the output signal. This can be done with the function **dtrend**.

```
u = detrend(utrend);
```

After performing the above operations, you have a pair of input/output signals that can be used to identify a model for the signal (motor plus bas). To perform identification you may use the function **armax** of Control Systems Toolbox (MATLAB), according to the code:

```
z = [yf u];
na = 3; % AR part
nb = 2; % X part
nc = na; % MA part
nk = 1; % Atraso puro – pure delay
nn = [na nb nc nk];
th = armax(z,nn)  % th is a structure in identification toolbox format
```

If you simply write **th** at the MATLAB command line, the content of this structure is eched in the form of the model. However, for the sake of programming, you must extract from th the vectors of the coefficients of the numerator and denominator polynomials of the model. This can be done with the command

```
[den1,num1] = polydata(th);
```

Before adding the integrator and build the ste model, you can validate the model you have obtained by comparing the output response of the model to the input signal with the differenced and filtered data of the real system. You may simulate the model with the function **filter**:

```
yfsim = filter(num1,den1,u); % Equivalent to idsim(u,th);
```

Figure 11 shows this comparison (this example does not necessarily uses the orders used in the code above). Try with different orders in order to obtain a model with an order that is not to big, but that leads to a good fitting. Observe that the answer to the question *What is the best order?* Is given in ultimate terms by the performance of the controller you will design with the model in a later stage of this work!

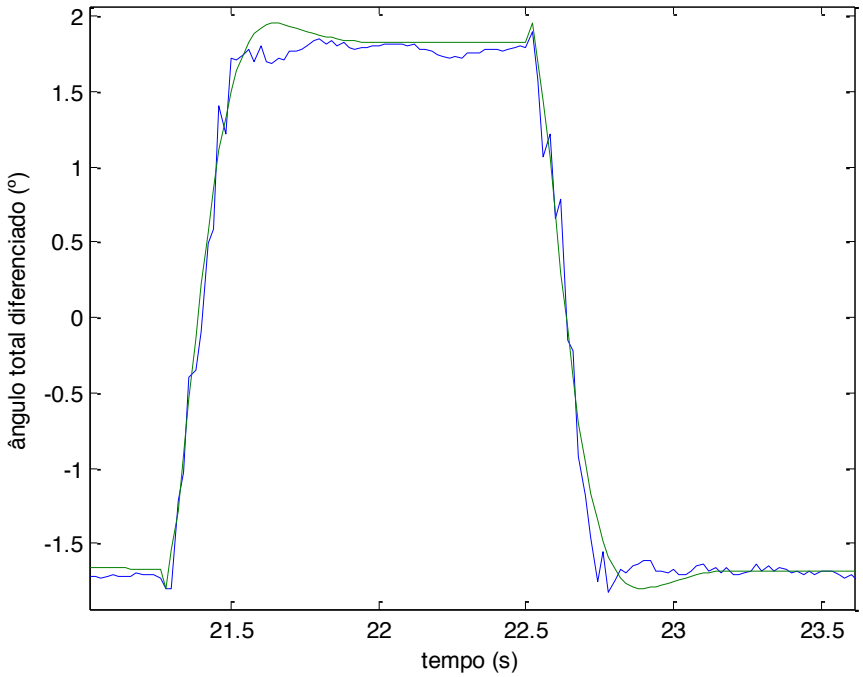



Figure 11. Comparison of the data for the differentiated output with the model output.

You should now add thge integrator that was removed from data. This may be done with the function conv (test this function separately to multiply to 1st order polynomials and check the result):

```
[num,den] = eqtflength(num1,conv(den1,[1 -1]));
```

It is remarked that model indentification was performed with the System Identification Toolbox, that assumes that the polynomials are written in the delay operator. However, conversion to the state model and model design are performed with the Control Systems Toolbox, that assumes that the polynomials are written in the forward shift operator. In order to convert one polynomial representation into another we used above the function **eqtflength**.

At this point, vectors num and den have the coefficients of the numerator and denominator polynomials of the transfer function of the model identified for the relation between the electrical excitation of the motor and the bar tip. These

polynomials assume a parametrization in the forward shift operator. We can now use the function

```
[A,B,C,D] = tf2ss(num,den);
```

to make the conversion to the state model parametrized by

$$x(k+1) = Ax(k) + Bu(k) \tag{4-1}$$

$$y(k) = Cx(k) + Du(k) \tag{4-2}$$

where $u$ (scalar) is the motor excitation, $y$ (scalar) is the flexible bar tip angular position, $x$ $(n \times 1)$ is a column vector and $A$ $(n \times n)$, $B$ $(n \times 1)$, $C$ $(1 \times n)$ and $D = 0$ (scalar) are matrices of compatible dimensions, $n$ being the dimension of $x$. The index $k$ denotes discrete time.

These matrices form the input to the next phase of the project, concerned with controller design.

## 💻 Experiments to perform at the laboratory

***Number of 3h laboratory sessions: 2***

***Objectives:*** *At the end of the lab sessions the following elements should be available:*

1. *At the end of the first session you should have a MATLAB script file that calls a SIMULINK block file to perform experiments on the plant and identify its model;*
2. *At the end of the second session you should have a model of the plant and the record of a set of identification experiments that show that the model you selected is the best choice.*

***Before the lab sessions, to prepare them*:** *Read carefully section 3 of this guide. Write a MATLAB script to test, correct and use during the lab sessions, that embeds the procedure and code described above. You will probably need to add some lines of MATLAB code to define some variables (for instance, the sampling interval) or make plots.*

*It is quite important that, before the very first of these 2 sessions you do this preparation.*

**At the lab**

Execute the procedure listed above to obtain a model. Write MATLAB scripts to automate the procedure. Use the suggestions in the previous section. Explore several options in a critical way. Take into consideration the issues you have to answer in the report. In particular, perform the following tasks (see the details above for each one):

1. Select an input signal (for instance a square wave or a PRBS) and perform an experiment in which the plant is excited with it. Register the data. For this sake, use the data acquisition facilities of SIMULINK that you have learned how to use in section 3, and write an appropriate SIMULINK block diagram. Use a sampling interval of 0,02 s (sampling frequency of 50 Hz). Observe that the motor+gear box has a dead zone.
2. The plant output (angular position of the flexible bar tip) data has to be treated in order for the identification algorithm to work properly
    a. Remove constant offsets in data using the MATLAB function *detrend*
    b. Differentiate the plant output to remove the integrator associated to the motor and filter the output to eliminate high frequency dynamics and noise that fall outside the frequency band in which the model is to be valid. For that sake use the commands suggested above.
3. Identify an ARMAX model using the function armax;
4. Add the integrator to the model (use the function *conv*)
5. Convert the model to state-space form using the function *tf2ss*
6. Validate the model in various forms.
    a. Compare data that has not been used for identification with the result obtained by simulating with the model using as input the same input used to excite the system;
    b. Look at the simulated time response and discuss its plausibility;
    c. Look at the frequency response and discuss its plausibility.
7. Review the process, changing the values of some parameters (filter pole, assumed model orders, sampling interval).

**Q3 ✍ Write a report about the plant model**.

The report must address the following issues:

- Explanation on the tests performed on the plant to obtain the data used for identification. Discuss the results obtained with different types of excitation signals and the effect of very small amplitude and very large amplitude excitation signals;
- Discussion of the sampling frequency;
- Effect on identification of filtering the data;
- Explanation on how the pole at the origin has been dwelt with;
- Discussion on how the model orders have been decided. Take into consideration that the plant results from the interconnection of a DC motor and the flexible bar and discuss the models needed for each of them;
- Description of the final ARMAX model;
- Description of the final state-space model.
- Characterization of the plant open loop pole-zero plot, frequency response and time response of the model.

- Model validation;

# 5. Controller design.

In this section, the state-space model identified described on section 4 is used to design a LQG controller.

The controller consists of a state feedback control law whose gains are selected such as to minimize a quadratic cost. The controller gains are designed on the basis of the state model (3-1), (3-2) that has been identified. This is the LQ control law.

Since the state of the plant is not available for direct measurement, it is estimated with a Kalman filter. The Kalman filter receives as input the plant input and output and yields as it output an estimate of the plant state. The Kalman filter has a structure that is equal to the current observer, but its gain is selected in a particular way so as to optimize the signal-to-noise ratio of the estimate in the presence of noise that affects the plant.

The equations of the algorithm that are required to perform this work are described hereafter. Further details can be seen in the book

G. F. Franklin, J. D. Powell and M. Workman. *Digital Control of Dynamic Systems.* 3rd ed., Addison Wesley, 1998, chapter 9 "Multivariable and Optimal Control".

***LQ control law.***

The control law that generates the manipulated variable $u$ for the state model (3-1), (3-2) such as to minimize the steady-state (or infinite horizon) quadratic cost

$$J = \sum_{k=1}^{\infty} x^T(k)Qx(k) + u^T(k)Ru(k), \tag{5-1}$$

where $Q \geqslant 0$ (means $Q$ is positive semi-definite) and $R \succ 0$ (means $R$ is positive denite) is given by the state feedback

$$u(k) = -Kx(k), \tag{5-2}$$

where the vector of feedback gains $K$ (with dimension$1 \times n$) is computed by

$$K = (B^TSB + R)^{-1}B^TSA. \tag{5-3}$$

The matrix $S$ $(n \times n)$ is the only positive definite solution of the algebraic Riccati equation (ARE)

$$A^TSA - A^TSB(B^TSB + R)^{-1}B^TSA + Q = 0. \tag{5-4}$$

If there exists a $K$ such that the closed-loop system that results from coupling (3-1) with (5-2) minimizes the cost (5-1), then it can be found by (5-3), (5-4).

For the purposes of this work, The LQ gain is computed using the MATLAB function *dlqr*. Use the MATLAB help to know how to use this function.

The function also computes the closed-loop eigenvalues, given by the eigenvalues of $A - BK$. This result is an important piece of information. For a well posed problem, the real part of these eigenvalues is always strictly negative.

The main "tuning knob" for the designer when using LQ control are the matrices $Q$ and $R$. For a SISO plant, if the output $y$ is to be regulated, then (due to (4-1) and $D = 0$)

$$Q = C^T C. \tag{5-5}$$

The weight $R$ reduces thus to a scalar. Changing $R$ adjusts the type of response yielded. Indeed, the poles of the closed-loop system are the stable roots of the root square locus. In general, decreasing $R$ increases the bandwidth of the control system. Thus, decreasing $R$ has the advantage of making the response of the close loop faster, but the disadvantage of exciting plant modes that are not modeled.

You may get an overall picture of the dependence of the closed-loop poles on $R$ using a technique known as root-square locus. See the companion document or the reference book.

*Warning*: Do not confuse *dlqr*, that solves the LQ problem in discrete time (the formulation used here) with the function *lqr* that solves a similar problem in continuous time (not considered in this work).

***Kalman filter.***

The Kalman filter aims at estimating the plant state from the observation of the plant input and output. It assumes that the plant model is modified by the inclusion of "disturbance" or "noise" inputs, becoming

$$\mathrm{x(k+1)} = \mathrm{Ax(k)} + \mathrm{Bu(k)} + w(k)\,, \tag{5-6}$$

$$y(k) = Cx(k) + Du(k) + v(k). \tag{5-7}$$

The signal $w$ is the process noise (that models random disturbances) and the signal $v$ is the sensor noise. The Kalman filter has the structure of the current observer, with the gain optimized by taking into account these noise terms.

The equations of the Kalman filter (current observer) are given by

$$\hat{x}(k|k-1) = A\hat{x}(k-1|k-1) + Bu(k-1) \tag{5-8}$$

$$\hat{x}(k|k) = \hat{x}(k|k-1) + M(y(k) - C\hat{x}(k|k-1)). \qquad (5\text{-}9)$$

These equations are interpreted as follows:

- Equation (5-8) computes what you expect the state to be at time $k$ given the previous estimate $\hat{x}(k-1|k-1)$, the input sample applied, $u(k-1)$, and the knowledge about the state dynamics.
- Then, this prediction is corrected by a term that is proportional to what you expect the output to be given the state prediction, $C\hat{x}(k|k-1)$, and what is actually observed for the value of $y(k)$.

The proportionality constant vector $M$ is called the Kalman gain. It is computed so as to optimize a quadratic cost. In this work, the Kalman gain is computed using the MATLAB function *dlqe*. Use the MATLAB help to know how to use this function. Again, do not make a confusion with the function *lqe* that computes the Kalman filter gain for the continuous time case.

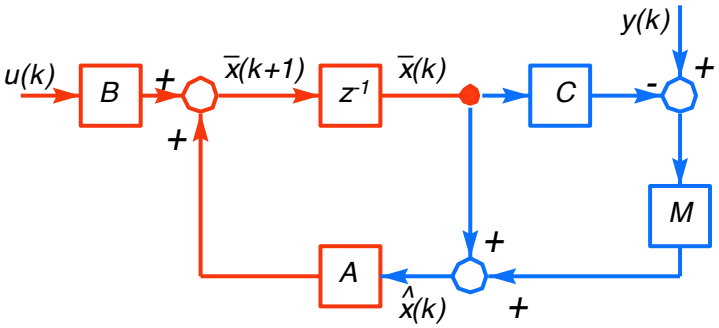



Figure 12. Block diagram of the state estimator. The part in red corresponds to the predictive observer and the part in blue to the correction to obtain the current estimate $\hat{x}(k)$.

Figure 12 shows the block diagram of the current observer / Kalman filter. Its inputs are the output plant observations, $y(k)$, and the samples of the manipulated variable applied to the plant, $u(k)$.

***Loop transfer recovery (LTR).***

The computation of the Kalman gain requires the covariance matrices of the noises entering the model. These are $Q_w = \mathcal{E}(ww^T)$ and $R_v = \mathcal{E}(vv^T)$. In this case, $R_v$ is a scalar. Instead of trying to estimate these noise covariances from plant data, we are going to use them as "tuning knobs". Thus, we make the choices

$$Q_w = 100I \quad \text{and} \quad R_v = 1.$$

Also for the observer, you may get an overall picture of the dependence of the error equation poles of the observer on $R_v$ using a technique known as root-square locus. See the companion document or the reference book.

***LQG controller structure***

The LQG controller is obtained by coupling a LQ controller with a Kalman filter and replacing the state by its estimate, that is to say, replacing (5-2) by

$$u(k) = -K\hat{x}(k), \tag{5-10}$$

This controller has the structure shown in figure 10.

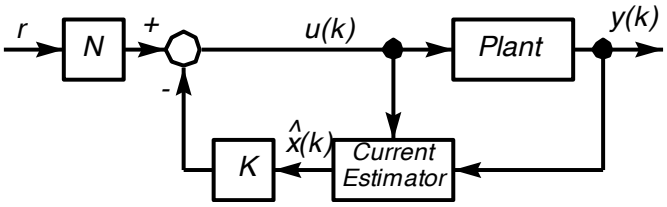



Figure 13. Block diagram of the controlled system.

The constant $N$ is such that, in closed-loop, the gain from the reference $r$ to the output $y$ is equal to 1. For a state of dimension 5 it can be computed by the MATLAB code:

```
N = inv([A-eye(5,5), B; C,0])*[zeros(5,1);1];
Nx = N(1:5,:);
Nu = N(6,:);
Nbar = Nu+K*Nx;
```

See the reference book on page 313 for a justification of this formula.

## 💻 Experiments to perform at the laboratory

***Number of 3h laboratory sessions: 2***

***Objectives:*** *At the end of the lab sessions the following elements should be available:*

1. *At the end of the first session you should have a MATLAB script file that calls a SIMULINK block file to perform experiments on the controlled plant;*
2. *At the end of the second session you should have a LQG controller of the plant and the record of a set of identification experiments that show that the controller you selected is the best choice.*

**Before the lab sessions, to prepare them:** *Read carefully section 4. When doing this, draw (using pencil and paper) a detailed block diagram of the controller by inserting figure 12 on figure 13. There is a companion text to this guide that summarizes the main point of state feedback design. It is suggested that you study it carefully before the lab session. Write a MATLAB script to design and test the controller (see also below, "At the lab) to test, correct and use during the lab sessions, that embeds the procedure and code described above. It is advisable that you test the controller design on the model that has been identified in section 4.*

*It is quite important that, before the very first of these 2 sessions you do this preparation.*

**At the lab**

Design a LQG-LTR controller according to the procedure explained and test it. At home start by testing the controller designed in simulation using the state-space model that has been identified in section 4. Explore several options by changing the weight matrix $R$. Start by making $R = 100$.

When you think you have a satisfactory result, try it on the real system. (Think about what are the features of the time response you look at to consider them "satisfactory"). What happens when you increase $R_v$?

Write MATLAB scripts and SIMULINK block diagrams to automate your design.

Among other things you find convenient to include in your report, document your design with:

- The time response of the closed-loop system
- The frequency response of the closed-loop system
- The loop-gain

**Q4** ✍ Write a report about controller design and testing of the controlled system, including:

- The MATLAB scripts that must include comments;
- The block diagram used to control the system;
- Comment on the choice of the weights on the quadratic cost when using the LQG design approach. Include the root-square locus;
- Effect of the choice of the noise covariance matrices in a LTR framework;
- Resulting closed-loop frequency response and time response;
- Effect of the inclusion of a pre-filter;
- Discuss how do you evaluate the performance of your control system and what are the limits of performance.

✋

***– End of the project –***