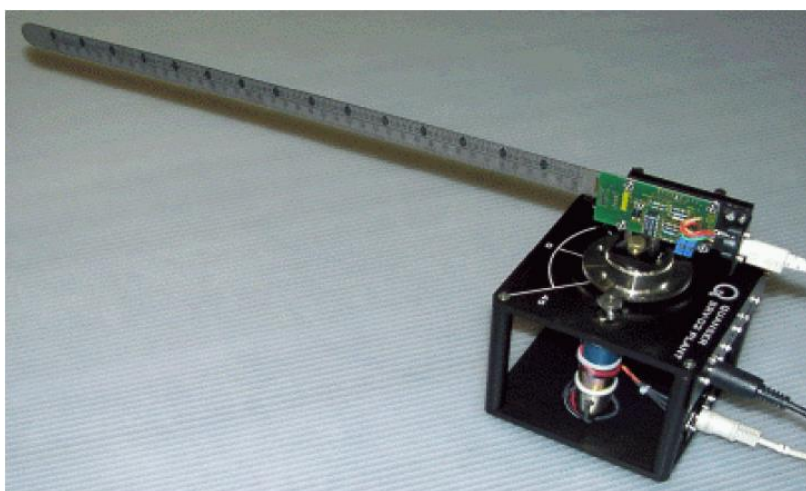


Master Degree in Electrical and Computer Engineering
Master Degree in Aerospace Engineering
2020/2021 – Winter Semester

Computer Control (Controlo por Computador)

LABORATORY WORK

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



Prepared by

João Pedro Gomes 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 must answer the questions marked with ✂. The end of the question is indicated by the symbol 🖐. The symbol 💻 corresponds to tasks to be performed in simulation.

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.

List of symbols and units

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

L [m] → Bar length.

θ [degree] → Angle of rotation of the motor shaft with respect to a “zero” direction.

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

ω [degree/s] → Motor angular speed.

θ_e [V] → Electrical tension yielded by the sensor of θ .

α_e [V] → Electrical tension yielded by the sensor of α .

ω_e [degree/s] → Electrical tension yielded by the sensor of ω .

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

K_p [degree/V] → Motor shaft angular position 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).

Unfortunately, due to limitations associated to the health situation we are now living, the tests with the real plant have been replaced with simulations using a SIMULINK block. Hereafter, whenever we say “plant”, we refer to this “true” model. Therefore, “experiments with the plant” actually means “simulations with the plant model provided”.

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.

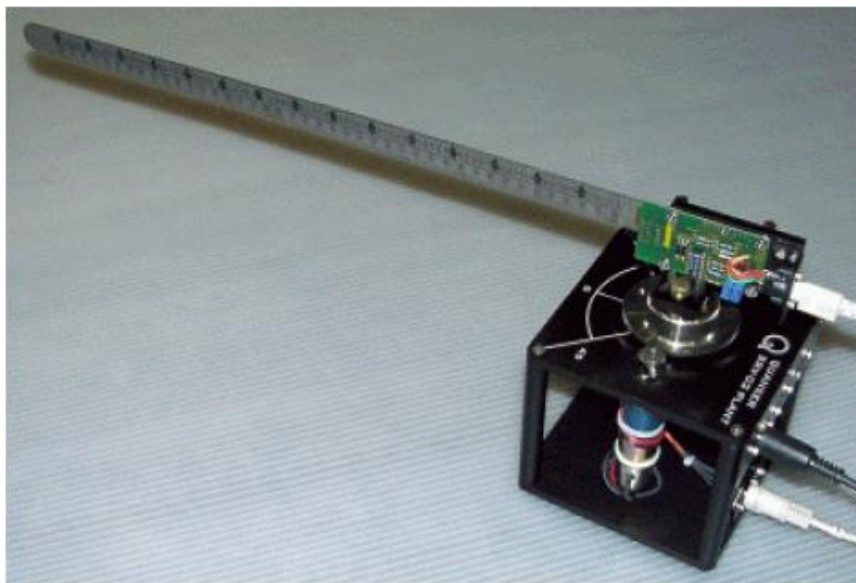


Figure 1. The physical system (plant) to control.

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?).

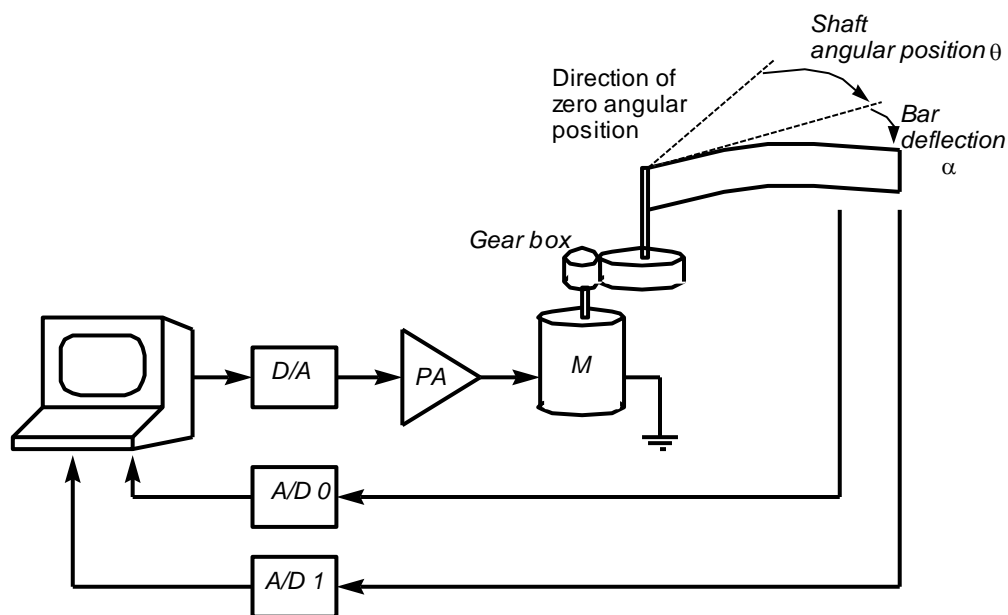


Figure 2. Schematic view of the hardware of the plant interconnected with the computer control system. The motor M (including a gear box) is driven by the output of a power amplifier PA and moves the flexible bar. The computer is interconnected with the plant using AD and DA converters.

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.

In this work, the model of the plant provided, has **only one input that corresponds to the total angle of the tip of the bar**, given by the sum of the shaft angular position, θ , and the bar deflection, α .

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. In this case, the physical plant is replaced by a Simulink model. This Simulink model allows to:

- Perform open loop experiments with the plant, that is to say, simulations with the model, in order to obtain data to identify a model that will be used for controller design
- Test the controller designed for the model obtained from plant data.

It is assumed that the student has a basic knowledge of MATLAB and Simulink, that can be easily acquired either in a previous course (such as the one on Modelling and Simulation – *Modelação e Simulação*) or by following one of the many tutorials available in the Web. 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.

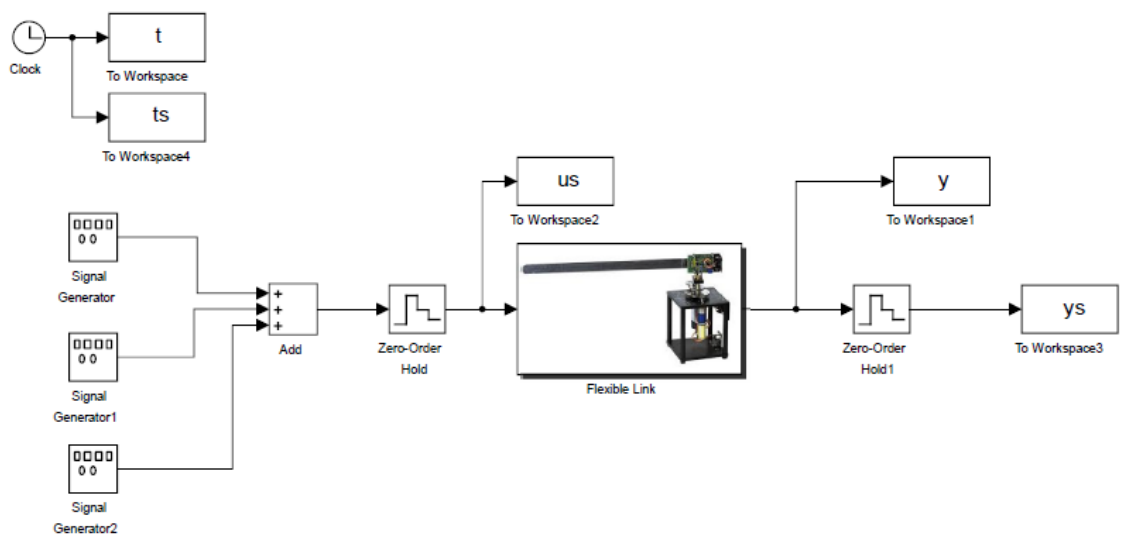


Figure 3. Simulink block diagram *Barra1.slx*. This diagram is used (in replacement of the physical model) to generate data for identification and, with the controller aided, to test the performance of the controlled plant.

Figure 3 shows the Simulink block diagram **Barra1.slx** (available in Fénix). This block diagram is used (in replacement of the physical model) to generate data for

identification and, when the blocks that define the controller are aided, to test the performance of the controlled plant.

The bar is simulated in continuous time by the block named *Flexible Link* (the one with the picture of the flexible bar setup on it). The input and out put are continuous signals that represent, respectively, the tension applied to the motor and the total deflection angle of the bar tip. The parameters of the “true” bar model are in a MATLAB data file named **barrassmodel.mat**. This file must be loaded to the MATLAB workspace before running the diagram in figure 3.

Before the bar input, and after the bar output, there are two sample and hold blocks that model, respectively, the D/A and the A/D converters. The sampling time is the variable **Ts**. To run the diagram in figure 3, you must also define the value of Ts in second. This parameter is an important one that you must optimize. As a starting point, we suggest 0.01 s.

The **output variables** of the block diagram are

- t – continuous time (s)
- ts – vector of sampling instants (s)
- us – vector of sampled manipulated variables (tension applied to the motor, expressed in volt)
- ys – vector of sampled plant output variables (total deflection angle)

Figure 3 refers to an open-loop experiment in which the motor is excited by the sum of 3 square waves with different frequencies.

Tasks to be performed

Objectives: *At the end of this task the following elements should be available:*



1. *SIMULINK block diagram to impose commands to the motor and to read signals from the plant;*
2. *Numerical values of the input (tension applied to the motor) and the output (deflection angle) in the form of vectors of numbers that correspond to the value of each of these variables at a given sampling instant (time series);*
3. *Plots of these time series.*

Preparation of this task

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 task 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.

A – Generation, viewing and recording signals.

1. **Open the SIMULINK.** Open the MATLAB and run the command <<simulink>> There is also a button in the MATLAB ruler that you may press, with the same effect. 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 blank 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. **Adjusting the duration of the simulation.** You may adjust the duration of the simulation either by writing it in the numerical field in the Simulink command ruler identified as "Simulation stop-time" or by clicking in "simulation > Model configuration parameters" and then filling the field "Stop-time". Stop-time may be either a number or a variable which numerical value is defined in the workspace.
7. **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)
```

8. **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 – Testing the system in open-loop.

9. **Use barra1.slx to generate open loop signals.** Write a MATLAB script (.m file) to simulate the response of the bar to an input signal (motor excitation) defined by you, and to record in a file on the computer hard-disk the vectors *ts*, *us*, *ys*. You may use the following set of instructions:

% File simubarra1.m

```
% Flexible robot link
%-----

% Loads the true flexible link model

load('barrassmodel.mat')

% Defines experiment parameters

Ts=0.01;      % Sampling interval

tfinal=200;   %

% Simulates the true model to generate data for identification

sim('barral');

% Plots the continuous and the discrete time outputs

figure(1)

gg=plot(t,y);
set(gg,'LineWidth',1.5);
gg=xlabel('t (s)');
set(gg,'FontSize',14);
gg=ylabel('y (volt)');
set(gg,'FontSize',14);

%axis([0 tfinal -4 8]);

hold on

gg=plot(ts,ys,'r');
set(gg,'LineWidth',1.5);

hold off
```

```
% Saves data for identification

save('iodata1.mat','us','ys','ts')

% us and ys contain i/o data with a sampling interval of Ts

%-----
% End of file
```

Q2 ✎ Write a (very short) report about open-loop experiments and data collection

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.
- **Show the step response.** Include in the report graphics documenting the step response.



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 diagram build as explained 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*) prevents 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.
- 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). 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 $T_s = 0,010$ s, as a starting point to your design.

It is suggested that the data of the different experiments performed is placed in a file 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 is in the vectors (explained above) `ts`, `us`, `ys`. Compute:

```
Ts = mean(diff(ts)); % Intervalo de amostragem (s)
utrend=us;
ytrend=ys;
```

It is remarked that the output angle steadily 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, one must:

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

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 echoed 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 operation 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);
```

Try with different orders in order to obtain a model with an order that is not too 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!

You should now add the 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 identification 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

$$x(k+1) = Ax(k) + Bu(k) \quad (4-1)$$

$$y(k) = Cx(k) + Du(k) \quad (4-2)$$

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.

Simulations to perform

Number of 3h laboratory sessions: 2

Objectives: *At the end of this task, the following elements should be available:*

1. *A model of the plant identified from input/output data and the record of a set of identification experiments that show that the model you selected is the best choice.*

Suggestions

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.
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.
 - d. The same for the pole-zero plot of the model identified.
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

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

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

$$u(k) = -Kx(k), \quad (5-2)$$

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

$$K = (B^T S B + R)^{-1} B^T S A. \quad (5-3)$$

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

$$A^T S A - A^T S B (B^T S B + R)^{-1} B^T S A + Q = 0. \quad (5-4)$$

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 $D = 0$)

$$Q = C^T C. \quad (5-5)$$

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

$$x(k+1) = Ax(k) + Bu(k) + w(k), \quad (5-6)$$

$$y(k) = Cx(k) + Du(k) + v(k). \quad (5-7)$$

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

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

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

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.

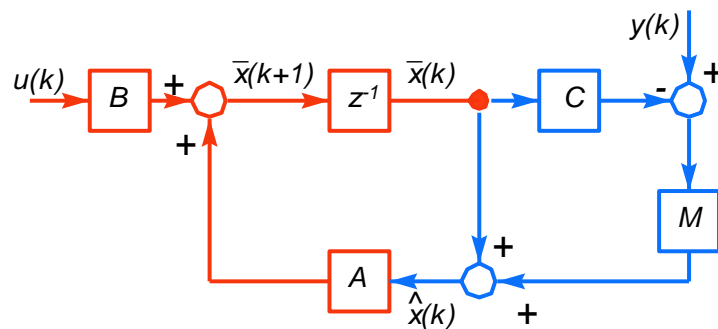


Figure 12. Block diagram of the state estimator. The part in red corresponds to the predictive observer and the part in blue to the correction to obtain the current estimate $\hat{x}(k)$.

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

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

This controller has the structure shown in figure 10.

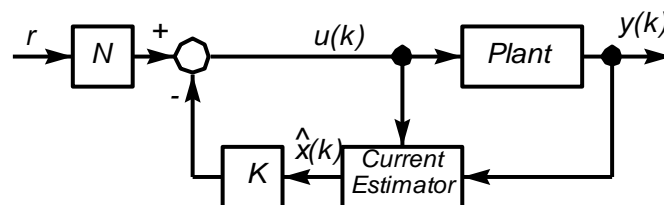


Figure 13. Block diagram of the controlled system.

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.

🖥 Simulations to perform

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

1. A MATLAB script file that calls a SIMULINK block file to perform experiments on the controlled plant;
2. 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.

Suggestions: 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.

Work to perform

Design a LQG-LTR controller according to the procedure explained and test it. 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 $R = 10$.

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* –

