Electrical and computer engineering department

Control engineer practical work

4th year AE

v2.0

2ND SEMESTER

Contents

EXPERIMENT 1

Finding paths in a graph : LegoRover

Lego Robot EV3

1.1 The purpose of the experiment

This experiment's purpose is to control a 2-wheel drive Lego robot that has a free wheel for balance, whose command is similar to that of a rover (Figure 1), in order to:

- efficiently follow a black line on the ground to guide its trajectory (low level continuous control: Level 0)
- correctly choose a path to follow according to a given criterion and constraints (high level command such as Artificial Intelligence: Level 2)
- correctly follow the desired path (runtime controller type command: Level 1)

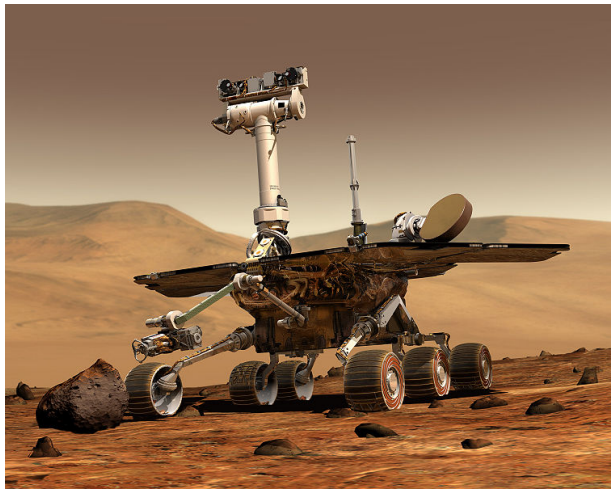


Figure 1: Curiosity rover

These different levels of control correspond to different fields in the field of Automation in general, and are illustrated in the Figure 2.

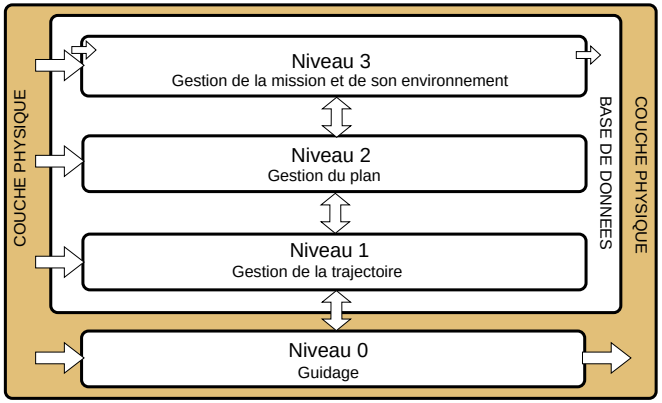


Figure 2: The different levels of control in an autonomous architecture

1.2 Model presentation

We are working with a Lego robot mounted on 2 wheels and a freewheel for balance as shown in the Figure 16.

The sensor used for line tracking is a LineLeader sensor (Figure 4). It is composed of 8 pairs of photodiodes / phototransistors, which, after calibration, returns a bit to 1 when the pair is in front of the color black (Figure 5). A function has been developed to return a sum weighted according to the position of the sensor in relation to a black line (see SensorLine function in PilotRobert.java).

The robot is controlled by an ARM9 microprocessor.

The robot moves on a map made up of black lines and identified points.

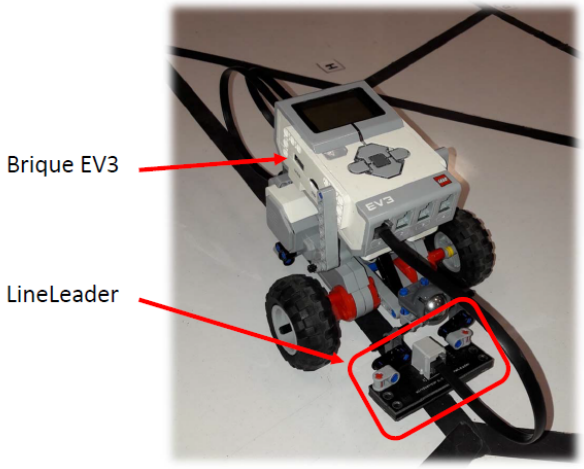


Figure 3: The Lego Robot

1.3 Presentation of the TPTomTom folder

Download the TP TomTom folder from moodle into your directory. Open the file MiseEnPlace-TPTomTom.pdf, follow the implementation of the TPTomTom project until the end. The EV3 project corresponds to the programming files of the Lego Robot. The generated code will be embedded in the brick of the robot. The PC project corresponds to the programming files of the PC that will communicate with the robot.

1.4 EV3 files presentation

The EV3 project contains the following files:

- `Pilote.java` : main function of the Pilot class, mission creation, start-up;
- `PilotRoberto.java` : robot guidance, control;
- `MissionRobot.java` : processes the missions sent by the PC and received by the robot;

Figure 4: The LineLeader sensor

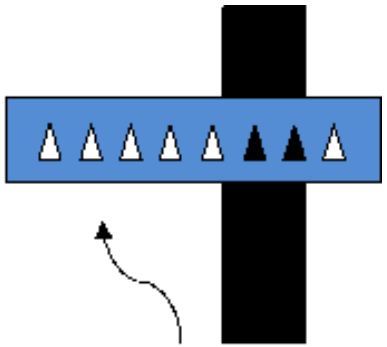


Figure 5: Line tracking principle diagram

- `CommBTRobot.java` : management of the Bluetooth communication with the PC;
- `Message.java` : management of the messages sent by the PC and received by the robot;

1.5 PC files presentation

The PC project contains the following files :

- `ShortestPath.java` : main class of the PC. Setting up the board, launching the graphical interface, launching the Dijkstra, building the mission, launching the mission on the robot;

- `test_BasNiveau.java` : test program for the guidance section. 3-point test to confirm the orientation;
- `Dijkstra.java` : code of the Dijkstra algorithm;
- `CommBTPC.java` : management of the Bluetooth communication with the Lego robot;
- `InterfaceDialogue.java` : graphical interface;
- `Map.java` : map management;
- `Message.java` : building messages between the PC and the robot;
- `MissionPC.java` : mission management; sequencing of the commands sent to the robot;
- `Order.java` : class for the commands to be sent to the robot;
- `Point.java` : point class;

1.6 Implementation of the guiding aspect

First, the objective is to implement the low level command on the robot, i.e. the guiding around a black line, as shown in the Figure 6.

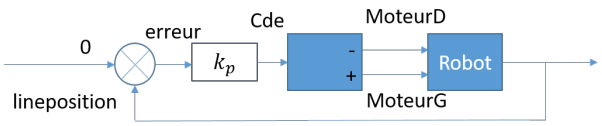


Figure 6: Control diagram

▶ Explain the control diagram. Why are the commands sent to the two motors different?

▶ Fill in the file `PilotRoberto.java` in order to set up the proportional control loop around a speed reference noted `speed` and a central position on the black line, knowing that the error and the control are of type (float). The value of kp is fixed and worth 10.

▶ Test your regulation in real time with the file `test_BasNiveau.java` with a constant speed value set at 300°/s.

▶ Vary the speed between 0 and 500 to determine the acceptable range of values for the speed.

▶ Vary Kp between 5 and 40 for example in order to highlight the characteristics of a proportional regulator.

Once the low level control (Level 0: guidance) has been validated, we will set up the level 1 control: trajectory management.

1.7 Implementation of the trajectory management aspect

This part consists in verifying that the trajectory management is properly carried out on the robot. To do this, we simply check the correct sequencing of a mission written "by hand".

▶ Using the file `test_BasNiveau.java`, create by hand a mission going from point A to point Q on the map, with a speed of 360°/s, using commands such as: `listOfOrders.add(new Order(int angle, int distance, int speed))`; with the angles in degrees, the distance in mm and the speed in degrees per second.

▶ Verify this mission on the robot.

1.8 Implementation of the mission plan management aspect

This part consists of developing a strategy for the development of a mission plan. To do this, we are led to define a graph whose vertices are the points that can be reached and whose arcs are the paths between these points.

- Formalize the optimization problem you are going to try to solve in order to elaborate a path for the robot by establishing an optimization criterion, drawing inspiration from the appendix entitled "Graph Theory - Dijkstra Algorithm".
- Complete the function `SetArc` in the file `Map.java` in order to define the weight of the arcs in the graph according to the chosen criterion.
- Suggest on paper an intuitive path finding method, for example a greedy algorithm that would allow to find a path between a starting point and an end node.
- Open the file `Dijkstra.java`. Study the code in order to find the steps of the pseudo-algorithm given in the appendix.
- Fill in the file `Dijkstra.java` :
  - method `Trouvmin()` which returns the vertex closest to the initial vertex: complete the search loop for the minimum value node not yet explored in `tab_value`.
  - method `MajCcum(int noeud_retenu)` which updates `tab_value` and `tab_noeuds`.
- Test this algorithm using a distance criterion (shortest path). Note: remember to uncomment line 53 in `ShortestPath.java`. (line BuildPath.ComputeDijkstra).
- We are now interested in making the fastest path in terms of travel time. Edit the `SetArc` function in the `Map.java` file to adapt the algorithm.
- Highlight a change following this criterion change by modifying the speeds on the map for instance.

1.9 Possible upgrades

This part aims at extending the algorithm to take into account other criteria or other ways of modeling the problem. For example,

- To suggest a solution allowing the robot to start from an initial node, to arrive at a end node, passing through particular nodes specified and not ordered, in an optimal manner (typical for a letter carrier's delivery route for example).
- Suggest a solution if the robot's resources are limited and it can, for example, only cover a limited distance (limited fuel).
- We can now imagine a rover on Mars, which has to carry out measurement points of more or less importance, with a limited amount of resources. Imagine an algorithm that solves the problem by maximizing the rewards on the measurement points while ensuring that the resources will be sufficient.

EXPERIMENT 2

Analysis of non-linear systems: Phase Plane Method

Study of a relay position servo control

2.1 The purpose of the experiment

The purpose of this experiment is to study the behavior of a system controlled by an all-or-nothing controller, with threshold, and/or hysteresis (Figure 7), by the phase plane method using Matlab.

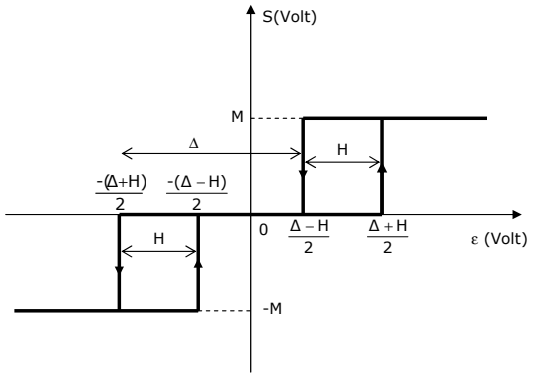


Figure 7: Non-linear element

The main benefit of the on-off controller is its simplicity. A relay or a simple switch can be used to materialize the device. The drawback is often premature wear of the components due to sudden and rapid changes of the command. In order to verify some of the results given in the part of the course on non-linear servo systems that deals with this method, we will simulate such a system with Simulink. The trajectories will be plotted in the phase plane of the system. The amplitude and period of self-oscillations for different parameter values (threshold and hysteresis of the relay, tachometer feedback rate, initial conditions) will also be evaluated if necessary.

2.2 Course reminders

General information on the Phase Plane method

The phase plane method makes it possible to study systems whose model is non-linear by using a state space method (as opposed to the 1st harmonic method which is a frequency method). In practice, this method is limited to second-order systems. Beyond that, graphic representation is impossible.

Let a physical system with one degree of freedom governed by a differential equation of the second order which is written:

$$\ddot{x} = f(x, \dot{x}) \tag{1}$$

if we set: $\dot{x} = y$, the equation (1) becomes equivalent to the system:

$$\begin{cases} \dot{x} = y \\ \\ \dot{y} = f(x, y) \end{cases} \tag{2}$$

The evolution therefore depends on two parameters, the x position and the $\dot{x}$ speed of the system, which can be defined as the two state variables of the system $X = (x_1 = x, x_2 = \dot{x})$.

- The state of the system is characterized in the plane $(x, v = \dot{x})$, called **phase plane**, by the point P whose coordinates are $(x_1, x_2) = (x, \dot{x})$
- The evolution of the system as a function of time for given initial conditions is described by the so-called **phase trajectory** of the P plane in the phase plane.
- The set of phase trajectories corresponding to the various initial conditions $(x_0, \dot{x}_0)$ in P_0 allowed, constitutes the **phase portrait** of the system.

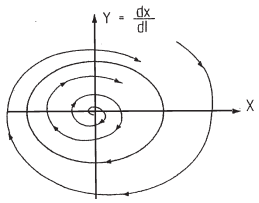


Figure 8: Example of phase trajectory

Notion of singular points

Any point where $\dot{X} = 0$ is called a **fixed point**, **critical point** or **equilibrium point**. A pair (x_1, x_2) such that $\dot{X} = 0$ is an equilibrium point, it can be stable or unstable. These points are singular points for trajectories. For any other point than a singular point, there is only one phase trajectory that passes through this point.

Notion of limit cycle

The limit cycles are the trajectories in the phase plane that correspond to the "limit oscillations" of the system, periodic solutions towards which the trajectories tend for all initial conditions located in a certain region of the phase plane. Let us recall the essentially non-linear nature of such phenomena. Figure 8 shows an example of a stable limit cycle towards which trajectories converge, diverging from an unstable focus and converging from distant regions of the phase plane.

In the case of 2nd order controls with discontinuous nonlinearities (relays, dry friction), the system can be considered as piecewise linear. The integration of the equation (2) is relatively easy to do by analytical methods. We obtain, for the different states of this non-linearity, a non-parametric equation independent of t of the form $f(x, y) = 0$ defining several types of elementary trajectories. The construction of the global trajectory will be made from these elementary trajectories connected at "switching points" corresponding to the change of state of the discontinuity. It is thus shown that in certain cases (relay with hysteresis for example), the trajectory obtained tends towards a stable limit cycle; it is then possible to study the stability of such a cycle, and to calculate its amplitude and period using the Poincaré point transformation method.

Special properties related to the form $\dot{x}_1 = x_2$.

Direction of phase trajectories: Since x increases in the area where $\dot{x} > 0$ and x decreases in the area associated with $\dot{x} < 0$, the phase trajectories are followed in a clockwise direction.

Intersection of the Ox axis with the phase trajectories: When the phase trajectory crosses the x-axis, it is either perpendicular to the Ox axis or it passes through a singular point. Indeed, the tangent to the phase trajectory can be defined as the line making an angle α with the Ox axis such that :

$$tan\alpha = \frac{dv}{dx} = \frac{\frac{dv}{dt}}{\frac{dx}{dt}} = \frac{f(x,v)}{v}$$

because by definition $\frac{dv}{dt} = f(x,v)$. We then have two cases:

- $f(x,v) \neq 0$ we then have a vertical tangent
- $f(x,v) = 0$ and we have a singular point (undetermined form).

Equilibrium position in the phase plane: The equilibrium positions are defined by $v = \dot{x}$ =0 and $\dot{x} = \dot{v}$ =0, and are therefore on the abscissa axis.

Notion of switching lines

The switching of the non-linear element takes place when the ϵ input is equal to characteristic values. At this point, the output value of the relay switches. This corresponds in the state plane to two shifted parallel lines. For example, in the case of a pure threshold, the relay switches for values $\epsilon = \pm\frac{\Delta}{2}$. The value of ϵ can be expressed as a function of the input and output. This equation defines two switching lines: when the path crosses one of the lines, the relay switches.

2.3 Experiment

The block diagram of the servo to be studied is given by the figure 9.

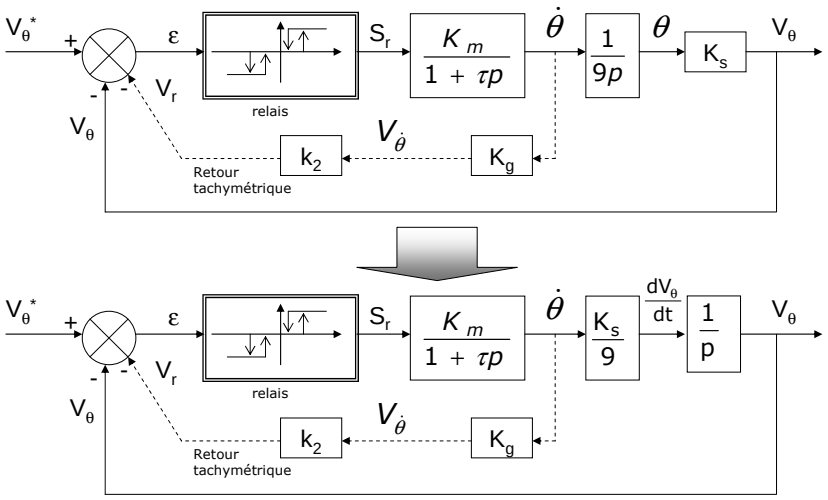


Figure 9: Block diagram of the servo system

- The physical parameters have been identified

$$\begin{array}{ll} K_m = 38.57 rad/V.s & K_s = 1.57 V/rad \\ K_g = 0.23 V.s/rad & \tau = 0.27 s \end{array} \quad (3)$$

- k_2 is an adjustable parameter to set the tachometric correction.
- The non-linear element consists of a relay with threshold and hysteresis whose characteristic $S_r = F(\epsilon)$ is represented by the figure 7. The output voltage level of the relay that supplies the motor is $\pm M = \pm 10V$.

Comment 1 *: In order to simplify the phase plane analysis of this simulated system, angles will be expressed in radians, voltages in volts and time in seconds; Moreover, the system will be considered as an autonomous system (input $V^*_{\dot{\theta}}(t) = 0$), disturbed from its equilibrium position by the initial condition $V_\theta(0)$, with trajectories represented in the plane $(V_\theta(t), dV_\theta/dt)$.*

Study of the system without tachometric correction

Study of the linear system:

- Calculate the expression of the transfer function $V_\theta(p)/V_\theta^*(p)$ without taking into account the relay. What are the characteristics of the closed-loop system (damping, self-pulse, static error, overshoot, rise time, ...)? In Simulink, check these data on the step response of the system with zero initial conditions.
- Give its state representation by setting $X_1 = V_\theta$ and $X_2 = dV_\theta/dt$

Study of the whole system: Add the non-linear element in the Simulink schematic, configurable with H and Δ.

The differential equations that govern the system are:

$$\begin{cases} \frac{d\dot{V_\theta}}{dt} &= -\frac{1}{\tau}\frac{dV_\theta}{dt} + \frac{K_m K_s S_r}{9\tau} \\ \frac{dV_\theta}{dt} &= \dot{V_\theta} \end{cases}$$

- What are the equilibrium conditions for this system?

Study of the pure threshold ($\Delta \neq 0$, $H = 0$)

- We take for example $\Delta = 5$. What is the condition on the initial condition $V_\theta(0)$ for the system to evolve towards an equilibrium point? Show it with Simulink.
- What are the characteristics of this system? (static error, overshoot, frequency...)
- Theoretically justify the shape of the curves: asymptotes, slope, etc.
- The switching lines are the lines on which the non-linear element switches. What are the equations of the switching lines in this case? Show them on your graphs.
- Repeat this study by adjusting the value of Δ. What do we notice about the static error? The overshoot? Is there a link with the switching lines? Conclude.

Study of the pure hysteresis ($\Delta = 0$, $H \neq 0$) A similar study is carried out for pure hysteresis.

- We take for example H =5. Is there a condition on $V_\theta(0)$ for the system to evolve towards an equilibrium point? Show it with Simulink. Will the system stabilize towards a point of equilibrium? Justify.
- Theoretically justify the shape of the curves: asymptotes, slope, etc.
- The switching lines are the lines on which the non-linear element switches. What are the equations of the switching lines in this case? Show them on your graphs.
- Repeat this study by adjusting the value of Δ. What do we notice about the static error? The overshoot? Is there a link with the switching lines? Conclude.

Study with hysteresis + threshold relay ($\Delta \neq 0$, $H \neq 0$)

- Do a similar study in this case. In particular, for $H = 5$, experimentally find the Δ_{lim} between the oscillating and non-oscillating mode. In a dual way, for $\Delta = 5$, find experimentally H_{lim} between the oscillating mode and the non-oscillating mode.
- What are the equations of the switching lines in this case? Show them on your graphs.
- Conclude.

Comment 2 *The choice of the number of iterations and the value of the calculation step should be a good compromise between the accuracy of the plot and the execution time.*

Influence of the tachometric correction on the transient state

An internal feedback loop is now implemented on the system (k_2 non-zero).

- Update your model accordingly and check that it is working properly.
- For $H = 5$, $\Delta = 0$ (pure hysteresis), find the new switching line equations. Identify them on your graphs.
- Identify two values of the counter-reaction rate, one corresponding to a sliding mode and the other to a non-sliding mode.
- Experimentally find the limit feedback rate and that of the optimal regime (origin reached after a single switching), for example using dichotomy.

2.4 Réalisation sur site réel

- Pratiquez les résultats obtenus en 2.3 sur le site réel.
- Observez la commande en régime glissant et en régime optimum.

2.5 Bibliographical references

★ http://moodle.insa-toulouse.fr/course/view.php?id=66 - Cours d'ASNL sous Moodle

★ C. MIRA : *Cours de systèmes asservis non linéaires.* DUNOD UNIVERSITE), 1969.

★ K . OGATA : *Modem control engineering* PRENTICE HALL, 1970.

★ J.C. GILLES, M. PELEGRIN : *Systèmes asservis non linéaires tomes 1,2 et 3.* DUNOD AUTOMATIQUE, 1975

EXPERIMENT 3

Nonlinear systems analysis: 1st harmonic method

Study of a relay position control system

3.1 Experiment presentation

The purpose of the experiment

The purpose is to study a position control in which the amplification element is a relay (non-linear element). This study focuses on the experimental finding of the critical location of the relay, based on the knowledge of the transfer location of the linear section, and the characteristics of the pumping oscillation (amplitude and pulsation), measured empirically.

Course reminders : approximation of the first harmonic

When a sinusoidal signal is applied to a non-linear system, a periodic but non-sinusoidal $s(t)$ output is usually obtained.

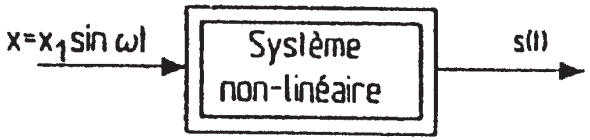


Figure 10: Non-linear element

For such a system, one cannot define a transfer function as one does for a system described by a linear differential equation with constant coefficients. The output $s(t)$ can be decomposed into Fourier series. By only considering the 1st harmonic of $s(t)$: $W_l sin(\omega t + (\phi))$, we define an equivalent or generalized transfer function:

$$module = W_l/X_l \qquad argument = \phi$$

This transfer function depends on the pulse ω of the input amplitude X_l.

$$W_l/X_l = B(X_l, \omega) \qquad \phi = \phi(X_l, \omega)$$
$$N(X_l, \omega) = B(X_l, \omega)e^{(j\phi(X_l,\omega))}$$

A very important case in practice is the one where N does not depend on the frequency, and depends only on the amplitude of the input signal X_l. This is the case of non-linear elements such as: threshold, saturation, all or nothing, ... In this case, the generalized transfer function $N(X_l)$ is called the equivalent complex gain.

Study of the stability of a looped non-linear servo system

Let's consider the system of the figure 11.

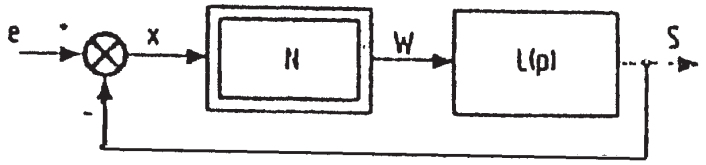


Figure 11: Non-linear loop

The generalized open-loop transfer function of this system is:

$$\frac{s}{x} = N(X_l).L(j\omega)$$

Suppose X_l constant; $N(X_l)$ is then a fixed number (real or complex). Applying the reversal criterion in the Nyquist plane, we can say that the control is stable for the amplitude X_l of the error if the

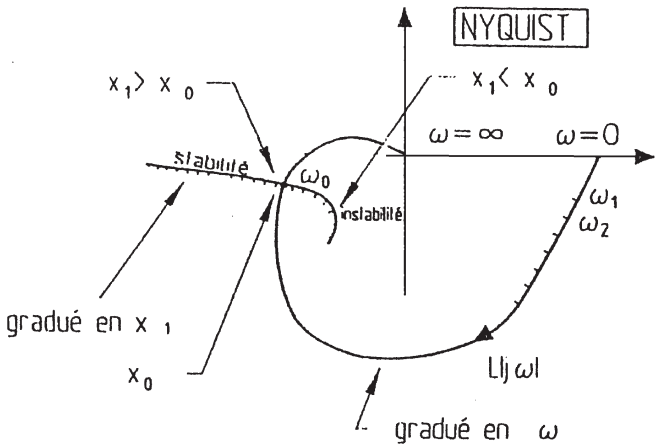


Figure 12: Geometrical criterion for determining a limit cycle

transfer locus $N(X_l).L(j\omega)$ traversed in the direction of increasing ω leaves the critical point - I on the left.

We can follow the same reasoning by considering the position of the locus $L(j\omega)$ with respect to the point $-\frac{1}{N(X_l)}$.

More generally, for each amplitude X_l of the error, we can define a critical point $-\frac{1}{N(X_l)}$. All of these points constitute the critical locus of the non-linear element.

On this critical locus, regions of stability and regions of instability can be identified (see figure 12).

- For $X_l < X_o$ the system is unstable. The amplitude of the error will therefore increase and we move on the critical locus towards X_0.
- For $X_l > X_o$ the system is stable, X_l decreases to X_0.
- At the limit the system oscillates with an amplitude X_0 of the error, at a pulse ω. This oscillation is called **pumping**.

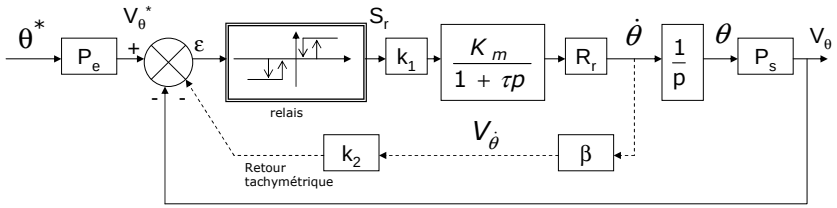


Figure 13: Assembly of the different elements

3.2 Equipment used

The experiment comprises a DC motor studied in 3rd year, and an adjustable relay.

Comment 3 *: We will use a transfer function $G(p)$ for the engine similar to $\frac{K_m}{p(1+\tau p)}$ with $K_m = 42.8 rad/s.V$ and $\tau = 0.214 s$.*

3.3 Experiment

Study of self-oscillations. Plotting of the critical locus

The idea is to experimentally construct the critical locus corresponding to a relay with certain characteristics, and to compare the results obtained with the critical locus constructed from the mathematical formulas studied in class.

Assembly

- Carry out the position controller without the tachymetric feedback represented by the following block diagram (Fig. 13) for which:
  1. P_e, P_s are respectively the conversion gains (physical data, voltage) of the control and the output. P_s is worth $1.58 V/rad$;
  2. R_r is a 1/9 ratio reducer;

3. k_1 is an attenuator potentiometer (10 turns) ($0 < k_1 < 1$) that allows to vary the open-loop gain of the linear section;
4. The tachometric generator placed at the end of the motor shaft delivers a voltage with a gain $\beta = 0.1V/rad.s^{-1}$.

Modus operandi

It is a question of experimentally constructing the critical locus for the relay with hysteresis $H = 1V$.

- First of all, we want to check the desired relay characteristic. We will take the input of the relay from X and the output of the relay from Y. Set the oscilloscope to XY mode by pressing **Main Delayed>XY**, and using the persistent mode of the oscilloscope **Display>** ∞**Persist**. Observe the graph $S_r = f(\epsilon)$ after having set the threshold (dead band) to zero, then the hysteresis H to the correct value of 1 V on the simulated relay module. The output S_r is worth $\pm 7.25V$.
- **Theoretical study:** We choose $k_1 = 0.5$. Draw under Matlab in the Nyquist plane the transfer locus of the linear portion $k_1.L(j\omega)$. The intersection of the critical locus $-1/N(X_l)$ with the linear locus $k_1.L(j\omega)$ gives rise to a self-oscillation whose amplitude is $X_l = X_0$, and the pumping pulse is ω. Note these theoretical values.
- To return the oscilloscope to normal mode, do **Main Delayed>Roll**. Set k_1 to 0.5. Raise the peak-to-peak voltage ($2X_0$), as well as the frequency for these self-oscillations. Compare to the theoretical values of amplitude and pumping pulse.
- By varying the calibrated attenuator, the gain k_1 is varied by $k_1 L(j\omega)$); a new point of intersection with the critical location is determined, and so on. Compare the theoretical and experimental values of amplitude and pumping pulse for the following values of k_1: 0.7, 0.6, 0.5, 0.4, 0.3. Comments?
- Based on the actual data obtained and the known process, plot the critical location of the relay.
- What can we conclude from this?

Improvement of the control performance

Scanning linearization Relay characteristic $H = 1V$; attenuator $k_1 = 0.5$.

- We now take the low frequency generator (GBF) which we set beforehand to obtain a sinusoidal output with a frequency of 50 HZ and an amplitude of 0.2V.
- With the system in self-oscillation, add to the error signal, on another input of the relay, the voltage delivered by the low frequency generator. Gradually increase the amplitude. What do you observe?
- Find this result with Matlab.
- Conclusion

Correction by tachometric secondary loop The voltage delivered by the tachometric dynamo, previously attenuated by a potentiometer, replaces the previous sinusoidal voltage: this is called a tachometric feedback.

- For $H = 1V$; set $k_1 = 0.5$. What is the amplitude of self-oscillation?
- Experimentally determine the value of the tachometric feedback rate allowing to halve the amplitude of self-oscillations.
- Find this result under Matlab using the critical locus drawn previously.

3.4 Bibliographical references

- ★ http://moodle.insa-toulouse.fr/course/view.php?id=66 - Cours sur la méthode du 1er harmonique sous Moodle
- ★ C. MIRA *Cours de systèmes asservis non linéaires.* DUNOD UNIVERSITE), 1969.
- ★ J.C. GILLES, M. PELEGRIN *Systèmes asservis non linéaires tomes 1,2 et 3.* DUNOD AUTOMATIQUE, 1975

EXPERIMENT 4

Digital control: dead-beat response & PID control

Computer control of an electric motor

4.1 Purpose of the experiment and presentation of the process

We have an electric motor, which we would like to control. The input of the system is a voltage $u(t)$ between +10 and -10V (a saturator can be added to avoid exceeding the limit values). The output is the rotation speed $\omega(t)$[1].

The motor is modeled as a first order system:

$$G(p) = \frac{\Omega(s)}{U(s)} = \frac{K_m}{1+\tau s}.$$

The purpose of this experiment is:

- to study the influence of the sampling period as well as the loop gain on the performance of a digital control;
- to implement an exact response;
- to implement a numeric PID control.

Matlab is used for the command. The numerical implementation of the real-time control is done in Matlab with the toolbox "Simulink Desktop RealTime" (see help under moodle).

4.2 Theoretical study (preparation)

Digital control with proportional controller

A sample regulation is set up according to the diagram in figure 14.

[1]The process actually has two outputs $\omega(t)$ and $\omega_2(t)$ which represent the rotation speed of the motor filtered and unfiltered. We will use the measurement filtered by a Butterworth filter.

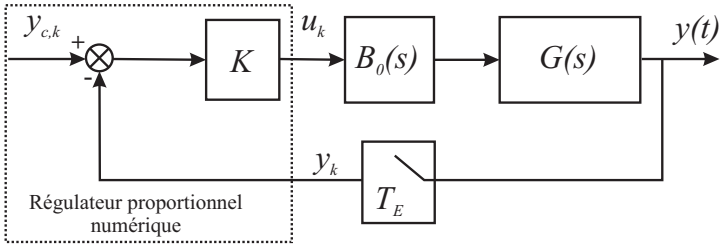


Figure 14: Discrete looping principle

- Calculate the sampled transfer function of the motor when a forward discretization is used:

$$p = \frac{z-1}{T_e}.$$

- Calculate the closed-loop transfer function, for a K proportional controller, as a function of K, K_m, T_E and τ.
- Calculate the static error of the sampled regulation as a function of K, K_m, T_E and τ.
- By searching for the boundary stability conditions, calculate the critical stability area:

$$K = f(T_E, \tau, K_m), \tag{4}$$

where K represents the gain of the proportional controller and T_E, τ and K_m the sampling period, time constant and system gain respectively.

Digital control: dead-beat response

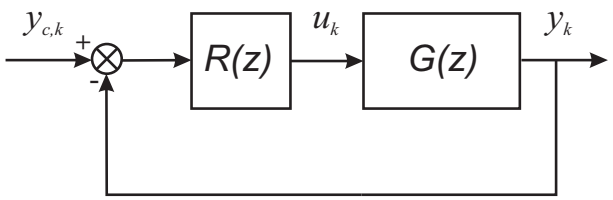


Figure 15: Discrete corrector

We wish to carry out a loop control of a sampled transfer function process $G(z)$ with a discrete corrector $R(z)$ according to the diagram of figure 15.

Give the expression of a $R(z)$ corrector to obtain a transfer function model for the looped system:

$$\frac{Y(z)}{Y_c(z)} = \frac{k_{damp}}{z^{(d+1)}} \tag{5}$$

where the d parameter represents the pure process delay.

Control by PID

When using proportional control, you observed a corrected system with a static error. Now, the goal is to use a digital PID controller to eliminate this static error.

- What type of digital PID controller (P, PI, PD, or PID) would meet this specification?
- Provide the block diagram representation of this control system.
- Propose a method for tuning the various parameters.

4.3 Experimental study

Process identification

We have an electric motor that we want to control using the 'Simulink Desktop RealTime' toolbox from Matlab. Please refer to the 'Simulink Desktop RealTime Connection Procedure' in Moodle. The signal from the sensors corresponds to a voltage ranging from -10.0 to +10.0 volts. Similarly, the output values that drive the DACs (Digital-to-Analog Converters) must also fall within this same range of values. Therefore, a saturator can be added to prevent exceeding the allowed values.

The motor is modeled by a first order system:

$$G(p) = \frac{K_m}{1+\tau p}.$$

The first step is to identify the parameters of the engine by studying its step response. To do this, under Matlab,

- Create an .xls that will handle the command. Set up a 4V **step** input connected to the DAC block.
- Once the relevant parameters have been chosen for identification (for example, 0.01 can be used as the calculation step), run the command. Display the motor speed on the oscilloscope or via Matlab. Press the **stop** button on the oscilloscope once the motor has reached its steady state.
- Using this response, identify the motor's parameters.

Auto-oscillations

- For the values of K_m and τ identified on the real system, plot the graph
$$K = f(T_E, \tau, K_m), \tag{6}$$
- For the sampling period $T = 0.35$s, the gain K_m and response time τ identified, give the value of the gain K_{lim} corresponding

to the stability limit. Verify this result in simulation with your "Simulink"application and then experimentally verify this result.

- For the sampling period $T = 0.35$s, the gain K_m and response time τ identified, calculate the static error with $K = 1$, under "Simulink"and verify it in practice.

Dead-beat response

Apply your theoretical results to the case of the motor speed control, for $d = 0$.

- Verify by simulation the expected result and then compare with experimental measurements.
- Conclude

Implementation of a Digital PID Controller (bonus)

Apply your theoretical results to the case of speed regulation of the motor.

- Specify the chosen requirements for tuning a controller to eliminate static error while maintaining acceptable performance.
- Verify the expected result through simulation and compare it with experimental measurements.
- Draw conclusions.

EXPERIMENT 5

Optimal control: the inverted pendulum

Lego robot EV3

5.1 Purpose of the experiment

The purpose of this experiment is to operate the Lego EV3 robot to keep it upright on its two wheels ("inverted pendulum" type experiment). The system is naturally unstable. For that, we will:

- move from a software design to a hardware implementation, including a simulation verification step. The design chain will be achieved via Matlab and its toolboxes;
- implement a full state feedback / pole placement control;
- implement an optimal lqr control, and compare different Q and R values.

5.2 Model presentation

We are using a Lego EV3 robot. It's a robot with two mobile wheels. The inputs and outputs of the system are shown in Figure 16.

At each wheel, an angular position sensor gives the angle of the wheel in degrees. An ultrasonic sensor allows the distance to an obstacle to be measured with a high degree of accuracy. It allows the robot to make an avoidance decision. The measured distances are in centimeters. A gyroscopic sensor measures the rotation of the robot thanks to a single-axis gyroscope placed on a quartz resonator. Concretely, this gyroscopic sensor measures the number of degrees per second as well as the direction of the rotation.

The robot operations are managed by an ARM9 processor under a Linux OS.

For a more detailed presentation of the robot and its model, please refer to the appendix.

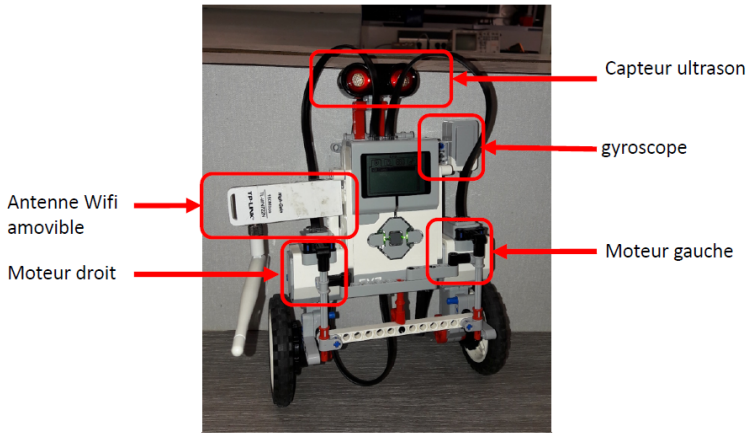


Figure 16: Lego EV3 robot

5.3 Presentation of the RobotPendule folder

Copy the RobotPendule EV3 folder from Moodle and the "Guide pendule Inverse" file to your directory. This file describes the contents of the archive and allows you to start the experiment.

5.4 Experiment

LQ control

Simulation In a first approach, we want to control the robot via state feedback. In order to determine the gain of this feedback, the LQ control principles are used (for more information about this control see the appendix). To do this, you need to perform the following tasks for several values of the weighing matrices Q and R:

- ▶ What are the states of the system, describe the state vector. Since the simulation model distributes the same torque to

both wheel motors, we will consider only one input. What does this imply on the state representation? Give the dimension of Q and R.

- Complete the matlab file lego_selbalance_controller.m. You just have to define the weighing matrices QQ and RR. This file must calculate the return state gain called KK here.
- Modify the simulink "Balance ans Drive Control". On this file, you have several data at your disposal:
  - $x1\_ref$ state model directive
  - $theta\_ref$ directive for the angle of inclination of the wheels
  - $x1$ state model measurement
  - $theta$ measurement for the wheel tilt angle
  - vol control of the wheel motors.

  You must implement the state feedback using the state feedback gain KK and the different data present (not all of them will be used there).
- Simulate the robot's behavior. **We will make a comparative table of the results for different values of QQ and RR according to the answers received**. We will try as much as possible to classify the answers into different categories and give the characteristics of each category. Show which components of QQ and RR are important in the regulation, on which characteristics?
- What values of QQ and RR do you keep? Justify your answer.

Test on the actual robot Once you have determined by simulation an optimal setting for QQ and RR, test your corrector on the robot using simulink's "external" mode. Look at the effects of disturbances, compare the experimental results and the simulation. Conclude.

Integral action

When a state feedback command is performed, the static error is not necessarily zero. This is usually not a problem as long as the objective is a control. However, if the objective is to follow a trajectory, then one can proceed in the same way as for conventional servo systems by inserting an integrator in the direct chain as shown on the figure 19.

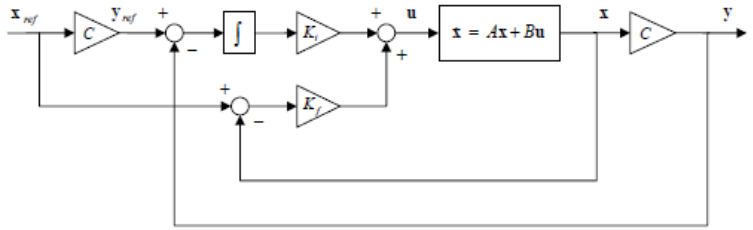


Figure 17: Block diagram of a status feedback control with integral action

In this part, we will include this integrator in the robot control function.

A brief explanation of the theory is given in appendix B.

- Complete the block diagram of the control to implement the integral action.
- Define the augmented system and calculate a value of K_i to reduce the static error using the lqr method.
- Test the robot's behavior in simulation. Compare with previous results. **Make a comparative table of the results.** Find a value of K_i that seems optimal to you.
- Implement this real time control on the robot. Carry out a test. Look at the effects of disturbances.

Pole Placement Control

The LQ control has the advantage and disadvantage of not imposing the choice of the eigenvalues of the corrected control matrix. This is an advantage because the designer does not have to think about the choice of eigenvalues, but it is also a disadvantage because the system dynamics are not chosen.

In this part we propose to refine the parameters found in the LQ section in order to try to improve the dynamics.

- Find the eigenvalues imposed by the LQ command of the first part that seem optimal to you.
- Using Matlab's *place* function, find the value of K_f to impose these system-specific values. A reminder of pole placement is given in Appendix B.
- Test the robot's behavior in simulation. Compare with previous results.
- Vary the imposed eigenvalues to improve the robot's behavior. Test the behavior of the robot in simulation. **Make a comparative table of the results, so as to conclude on the optimal coefficients.**
- Implement the best control on the robot. Run a test. Watch the effects of disturbances.

Pole placement with integral action

In this section, we propose to introduce an integrator into the control function of the robot while keeping the pole placement control.

- Use the block diagram of the control to set up the integral action.
- Define the augmented system and calculate a value of K_i to reduce the static error with Matlab's *place* function.
- Test the robot's behavior in simulation. Compare with previous results. **Make a comparative table of the results.** Find a value of K_i that seems optimal to you.
- Implement this control on the robot. Run a test. Look at the effects of disturbances.

5.5 Conclusion

- Summarize the advantages, disadvantages, and performances of your system according to the corrector used, thanks to the various established tables.
- Finally, test the robot with different wheels. Conclude.

5.6 Bibliographical references

★ Notice NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT, Yorihisa Yamamoto.

★ Réalisation, réduction et commande des systèmes linéaires, A. Rachid, D. Medhi, Technip, Collection Méthodes et techniques de l'ingénieur (Paris)

Appendix

EXPERIMENT A

Appendix: Graph Theory - Dijkstra Algorithm

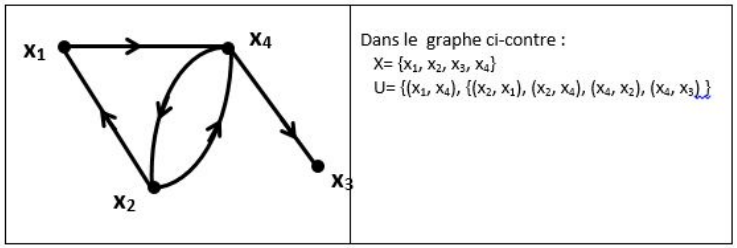
Graph theory is a favoured tool for modelling and solving decision problems in a very large number of fields ranging from fundamental sciences (physics, computer sciences) to the most concrete technological applications (design of telecommunications networks, allocation problems, GPS and shortest path searches, etc.). The basic concepts of graph theory are presented in detail in the Graphs course of the second semester of 4AE. Therefore, only the essential elements for the implementation of the Dijkstra algorithm for finding the shortest path between one vertex and all the others in a graph are introduced here.

A.1 Basic concepts about graphs

Graph, vertices, arcs

A $G(X, U)$ graph consists of

- a set of vertices $X = \{x_1, x_2, \ldots, x_n\}$
- of a set of arcs connecting these vertices $U = \{u_1, u_2, \ldots, u_m\}$. An arc u is defined by a pair of vertices: $u = (x_i, x_j) x_i \in X; x_j \in X$. x_i is the origin of the arc u and x_j its extremity.



Arc length and valued graphs

It is possible to associate a value l_{ij} to each arc (x_i, x_j). This value (which can be positive, negative or null) can, for example, represent a distance, or a speed, or a cost, or any other quantity representative of the problem modeled by the graph. This is called a valued graph.

Paths and circuits

A path is a sequence of arcs such that the end of one arc is the origin of the next arc in that path. For example, $(X_2 - X_1 - X_4)$ is a path between vertices x_1 and x_4.

A circuit is a path whose endpoint is also the origin of the path. So for example, $(X_2 - X_1 - X_4 - X_2)$ is a circuit.

The length of a path (or circuit) is equal to the sum of the lengths of the arcs that make up the path (or circuit).

A.2 Search for shortest paths

A typical problem in a valued graph is to find the shortest path between one vertex of the graph, which we will note s (source vertex) and all the others. Graph theory proposes different algorithms for finding the shortest paths adapted to the properties of the graph. In the particular case where all the lengths of a valued graph are positive, the Dijkstra algorithm can be applied.

This algorithm is iterative. We associate 3 pieces of information to each vertex xi :

- λ_i: length of the shortest path found at the current iteration between s and x_i.
- q_i: last vertex before x_i on this path (allows to reconstruct the shortest path at the end of the algorithm)
- $m_i \in \{0, 1\}$: the vertex x_i is said to be "marked" when $m_i = 1$. It is said "unmarked" when $m_i = 0$. When the vertex is marked, λ_i corresponds to the length of the shortest path between s and x_i (λ_i has its definitive value).

General principle of the algorithm

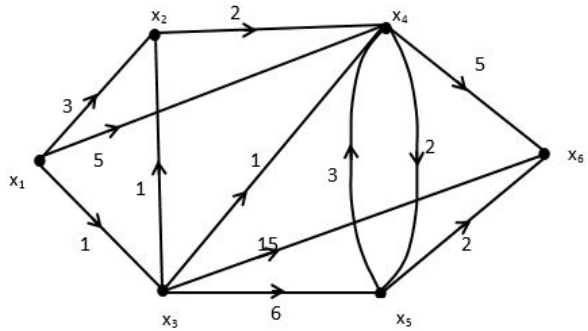
- Initializations: λ_s is initialized to 0 (the shortest path between s and s is of null length) ; the $\lambda_i, \forall i \neq s$ are initialized to ∞ (the shortest paths between s and x_i are not known at the beginning of the algorithm: their length is therefore ∞); these lengths are not definitive $\Rightarrow m_i = 0 \forall i$
- At each iteration:
  - We locate the vertex not yet marked ($m_i = 0$) which has the smallest λ.
  - We mark this vertex ($m_i = 1$) because, given the graph properties, we can be sure that there is no shorter path between s and this vertex.
  - We try to update the λ of the immediate following vertices of the vertex that has just been marked (see detailed algorithm).
  - We start a new iteration until all the vertices are marked ($m_i = 1$).

Detailed Algorithm

-- initialisations
$\lambda_s \leftarrow 0$; $q_s \leftarrow s$; $m_s \leftarrow 0$
pour tout i de 1 à n avec $i \neq s$ **répéter**
  $\lambda_i \leftarrow \infty$; $q_i \leftarrow \phi$; $m_i \leftarrow 0$
fp
-- itérations
tant qu'il existe un sommet non marqué ($m_i = 0$) **répéter**
  Choisir parmi les sommets non marqués celui dont le λ est minimum ; soit x_i ce sommet
  Marquer ce sommet : $m_i \leftarrow 1$
  -- actualiser éventuellement les λ des sommets suivants x_i
  Pour tout sommet $x_j \in S(i)$ et tel que $m_j = 0$
    si $\lambda_i + l_{ij} < \lambda_j$ **alors**
      $\lambda_j \leftarrow \lambda_i + l_{ij}$
      $q_j \leftarrow i$
    fsi
  fin pour
fin tant que
Remarque : on peut alors reconstituer le plus court chemin entre s et x_i en utilisant q_i

Implementation on an example

Consider the valued graph depicted below. We look for the shortest paths between s=x1 and the other vertices of the graph.



The table below details the different iterations of the algorithm. The grayed boxes show the vertices that are marked at the beginning of each iteration (unmarked vertex with the smallest λ). The values updated during the iteration of the λ and m of the vertices following the marked one appear on the same line.

	x_1			x_2			x_3			x_4			x_5			x_6		
	λ	q	m	λ	q	m	λ	q	m	λ	q	m	λ	q	m	λ	q	m
initialisations	0	x_1	0	∞	-	0	∞	-	0	∞	-	0	∞	-	0	∞	-	0
Itération 1	0	x_1	1	**3**	**x_1**	**0**	**1**	**x_1**	**0**	**5**	**x_1**	**0**	∞	-	0	∞	-	0
Itération 2				**2**	**x_3**	**0**	1	x_1	1	**2**	**x_3**	**0**	**7**	**x_3**	**0**	**16**	**x_3**	**0**
Itération 3				2	x_3	1				2	x_3	0	7	x_3	0	16	x_3	0
Itération 4										2	x_3	1	**4**	**x_4**	**0**	**7**	**x_4**	**0**
Itération 5													4	x_5	1	**6**	**x_5**	**0**
Itération 6																6	x_5	1

Reconstruction of the shortest paths :

For example the shortest path between x_1 and x_6 is of length 6 ($\lambda_6 = 6$). The last vertex before x_6 on this path is x_5 (because $q_6 = x_5$). The last vertex before x_5 on this path is x_4 (because $q_5 = x_4$). The last vertex before x_4 on this path is x_3 (because $q_4 = x_3$). The last vertex before x_3 on this path is x_1 (because $q_3 = x_1$). The shortest path is therefore: $x_1 - x_3 - x_4 - x_5 - x_6$.

A.3 Bibliographical references

★ Notice NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT, Yorihisa Yamamoto.

★ Réalisation, réduction et commande des systèmes linéaires, A. Rachid, D. Medhi, Technip, Collection Méthodes et techniques de l'ingénieur (Paris)

EXPERIMENT B

Appendix: State feedback control

B.1 Introduction and working principles

State feedback control is a control technique that consists of multiplying the error between the reference value $x_{ref}(t)$ and the measured state value $x(t)$ by a feedback gain K and using this value as a command. In fact, state feedback control can be seen as a generalization of the Proportional Derivative (PD) control in classical control theory. The figure **??** shows the block diagram of a state feedback control.

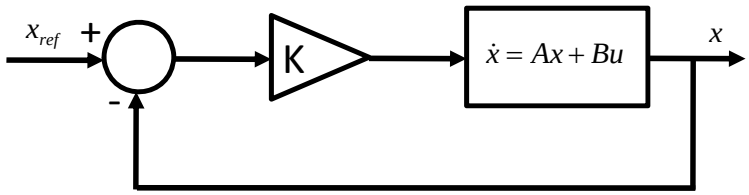


Figure 18: Block diagram of the state feedback control system

The control and the state equation of this system are as follows:

$$u(t) = -K(x(t) - x_{ref}(t)), \quad (7)$$
$$\dot{x}(t) = (A - BK)x(t) + BKx_{ref}(t). \quad (8)$$

It is then possible to stabilize the system by choosing the feedback gain K in order to correctly place the eigenvalues of $A - BK$.

To use a state feedback control, the system must be controllable. A necessary and sufficient condition for the controllability of the system is that the controllability matrix M_c is of full rank.

$$\mathbf{rang}(M_c) = n, \quad (9)$$
$$M_c = [B, AB, \dots A^{n-1}B], \quad (10)$$

with n the dimension of the system. The Matlab "control" toolbox allows to evaluate the controllability matrix.

`Example:` Is the following system controllable?
$A = [0, 1; -2, -3], B = [0; 1] \rightarrow$ controllable.

```
> A = [0, 1; -2, -3]; B = [0; 1];
> Mc = ctrb(A,B);
> rank(Mc)
ans = 2
```

B.2 Obtaining the K-matrix

Many methods exist to find the K gain.However, the *pole placement* and the *linear quadratic control)* are two methods widely used by engineers to find the K state return gain.

Direct Pole Placement

This method consists in calculating the feedback gain K so as to place the poles (the eigenvalues) of the $A - BK$ matrix at a precise value. The chosen values are obviously stable, and are often derived from the desired performance on the corrected system (speed, damping, decoupling...). However, care must be taken to ensure that the value of K is not aberrant (numerically well conditioned, limiting saturation...). This method requires solving a linear system.

The Matlab function `place` allows to perform a direct pole placement which minimizes the sensitivity.

`Example:` Calculate the state feedback gain for the system $A = [0, 1; -2, -3], B = [0; 1]$ so as to place the poles at -5 and -6.

```
> A = [0, 1; -2, -3]; B = [0; 1];
> poles = [-5, -6];
> K = place(A,B, poles)
K =
28.0000 8.0000
```

Optimal Linear Quadratic (LQ) control

The *pole placement* command aims to synthesize K, a static state feedback corrector (*eg.* such as $u(t) = Kx(t)$) in such a way that the poles of the closed loop are placed at the desired locations in the left complex half-plane.

The general idea: The principle of the *Linear Quadratic (LQ)* control, often - and wrongly - called "optimal control", is to synthesize K, a static state feedback corrector (*e.g.* such as $u(t) = K(x(t) - x_{ref}(t))$, which minimizes a quadratic criterion.

In the case of sampled systems, this quadratic criterion is none other than a weighted (finite or infinite) sum of $x(t)$, the state of the system, and $u(t)$, the command applied to it. In the case of continuous systems, this sum is an integral.

While pole placement can be seen as complex for a user (requiring to choose the poles), the linear-quadratic control is based on an energy criterion, much more meaningful for the engineer. It should be noted that the LQ control has, intrinsically, very good **robustness** properties.

Mathematical formulation (continuous case, infinite horizon): Let the Linear Time Invariant (LTI) system with n_u inputs and n_y outputs defined by its transfer function $H(s) \in \mathbb{C}^{n_y \times n_u}$, complex and analytical on $\mathbb{C}^+$, with an implementation,

$$\dot{x}(t) = Ax(t) + Bu(t),\ y(t) = Cx(t) + Du(t). \tag{11}$$

The objective of the LQ method is to calculate the feedback gain K so as to minimize the following cost function J:

$$J = \int_0^\infty \Big((x(t) - x_{ref}(t))^\top Q (x(t) - x_{ref}(t)) + u(t)^\top R u(t) \Big)\, dt \tag{12}$$

under the constraint of system dynamics (11). Note that the J criterion represents the trade-off between the system energy, $x(t)^T Q x(t)$, and the command energy, $u(t)^T R u(t)$.

It can be shown (see course in the second semester) that the solution to this problem is given by the command

$$u(t) = -K(x(t) - x_{ref}(t)), \tag{13}$$

with

$$K = R^{-1} B^T P, \tag{14}$$

where $P > 0$ (*i.e.* defined positive), is the solution of the Algebraic Riccati equation,

$$A^T P + PA - PBR^{-1}B^T P + Q = 0. \tag{15}$$

Note that the partial cost is $J(x(t), t) = \frac{1}{2} x(t) T P x(t)$.

The parameters of choice are the weights of the symmetric matrices for the state $Q = Q^T$, the command $R = R^T$. The matrices are most often chosen as diagonal. A solution generally chosen to already have an indication of the orders of magnitude of these coefficients is to set $R = I_{n_u}$ and $Q = \rho I_n$, then to vary the scalar ρ (a $\rho \gg 1$ indicates that we will concentrate on minimizing the energy of the system, then $\rho \ll 1$ indicates that we will concentrate on minimizing the command). We then proceed by trial-and-error. The function `lqr` of Matlab allows to obtain the feedback gain with the LQ method.

`Example:` Calculate the state feedback gain for the system $A = [0, 1; -2, -3], B = [0; 1]$ using $Q = [100, 0; 0, 1]$ and $R = 1$.

```
> A = [0, 1; -2, -3]; B = [0; 1];
> Q = [100, 0; 0,1]; R = 1;
> K = lqr(A,B,Q, R)
K =
8.1980 2.1377
```

ou

```
> A = [0, 1; -2, -3]; B = [0; 1];
> Q = [100, 0; 0,1]; R = 1;
> [P,L,G] = care(A,B,Q,R); % Résolution de l'EAR
> K=R\1*B'*P;
K =
8.1980 2.1377
```

About robustness: If we consider a single-input single-output system, looped by a $u(t) = Kx(t)$ command, where K is synthesized by infinite horizon LQ approach, with $R = \rho I$. Therefore, noting the loop transfer $L(s)$

$$L(s) = K(sI_n - A)^{-1}B, \tag{16}$$

and the transfer $F(s)$ (used for sensitivity and robustness analysis) and the sensitivity function $S(s)$ by

$$F(s) = I + L(s) = S(s)^{-1}, \tag{17}$$

it can be proven that

$$\begin{array}{rrcl} & F(-s)^T F(s) & \geq & 1 \\ \Leftrightarrow & |F(j\omega)| & \geq & 1 \\ \Leftrightarrow & |1 + L(j\omega)| & \geq & 1 \\ \Leftrightarrow & |S(j\omega)| & \leq & 1. \end{array} \tag{18}$$

Therefore, if the sensitivity function $|S(j\omega)| \leq 1$, then

- system module margin ≥ 1,
- the gain margin is ∞,
- phase margin ≥ 60deg.

State feedback control with integral action

When a state feedback command is performed, the static error is not necessarily zero. This is usually not a problem as long as the objective is control. Nevertheless, if the objective is to follow a trajectory, one can then proceed in the same way as for conventional servo systems by inserting an integrator in the direct chain, applying it to certain $y(t)$ outputs. So we consider a command of the form[2] (beware, here we have substituted $x(t) - x_{ref}(t)$ by $x_{ref}(t) - x(t)$):

$$u(t) = -K_f(x_{ref}(t) - x(t)) + K_i \int_0^t (y_{ref}(t) - y(t))dt \tag{19}$$

applied to the dynamic system:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{20}$$
$$y(t) = Cx(t) \tag{21}$$

If one introduces a new variable $z(t)$ such as $\dot{z} = y_{ref}(t) - y(t)$, i.e. $\dot{z} = Cx_{ref} - Cx(t)$. The command has an expression of the form

$$u(t) = -\begin{bmatrix} K_f & K_i \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} K_f & K_i \end{bmatrix} \begin{bmatrix} x_{ref} \\ 0 \end{bmatrix} \tag{22}$$

and it can be considered as the state feedback control of the augmented system with $X = \begin{bmatrix} x(t) \\ z(t) \end{bmatrix}$

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ C \end{bmatrix} x_{ref} \tag{23}$$

When we consider the stabilized system, we have $y_\infty = y_{ref}$.
The block diagram of such a command is shown in Figure 19.

[2]The books on command theory usually describe this command with $x_{ref} = 0$. We will keep here the expression of x_{ref} to improve the tracking performance.

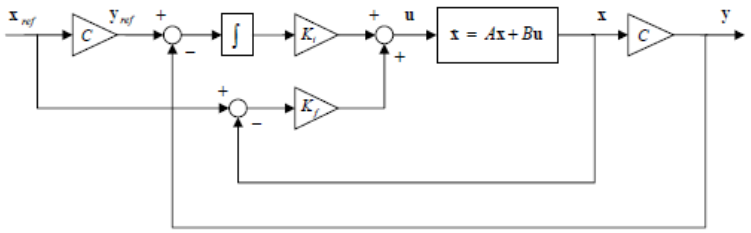


Figure 19: Block diagram of a state feedback control with integral action

B.3 Bibliographical references

- ★ Notice NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT, Yorihisa Yamamoto.
- ★ Réalisation, réduction et commande des systèmes linéaires, A. Rachid, D. Medhi, Technip, Collection Méthodes et techniques de l'ingénieur (Paris)