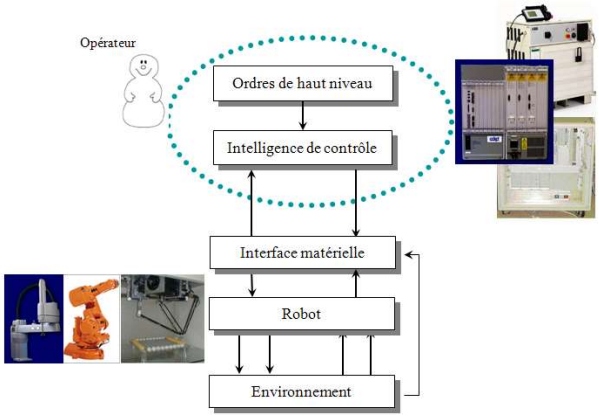
Mohamed Bouri, 2022

PART IV
Control of robots

Closed loop control,

Motion trajectories,

and Hardware



ABSTRACT

This chapter introduces the tools required for the understanding, analysis and implementation of a numerical control of industrial robots. The control laws of the PID family as well as the nonlinear laws based on dynamic models are presented. Trajectory generation and interpolation are also covered.

Mohamed Bouri
Dec, 2022
Beta version

Contents

4.1 Introduction

Robot control, theory and applications, is an evolving discipline since the existence of the concept of “robots”. This science of engineering enabled controlling robots and pilot them so that we can use them for automated and autonomous tasks as much diverse as complex.

The term "**Robot control**" is very generic because it implies everything related to the control of the robot. This term can just as well mean the robot control cabinet, the closed loop control algorithm of the robot, the robot control software or all these definitions together.

Thus, “Robot control” concerns the algorithmic, software and hardware parts involved in carrying out the tasks that the robot must perform.

4.2 General structure of the robot and its controller

4.2.1 Definitions

The figure below shows the example of a robot and its control cabinet linked by a connector.

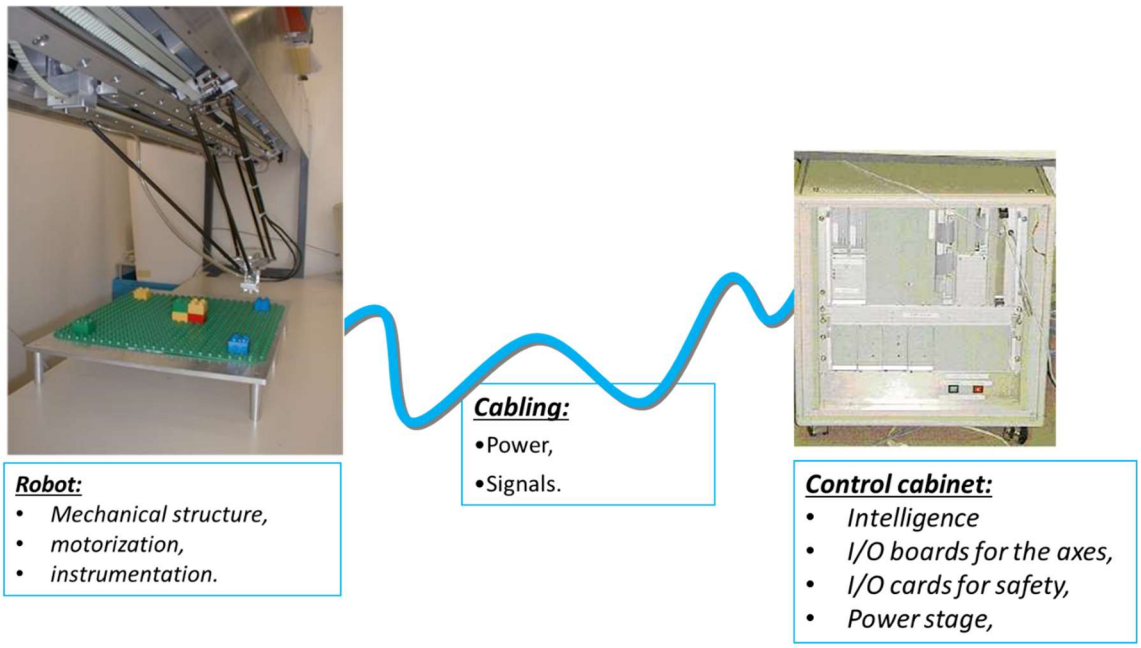


Figure 4.1- The robot “Linear Delta” and its control cabinet

The robot is defined as being the poly-articulated mechanical structure by associating the motorization and the instrumentation. The control cabinet includes the necessary electronic boards to control the robot (processor cards, analog and digital input-output cards, security cards, etc.), amplifiers, power supplies, fans, etc. To connect the robot to its cabinet There are two types of connectors: power connectors and signal connectors (mainly related to sensors).

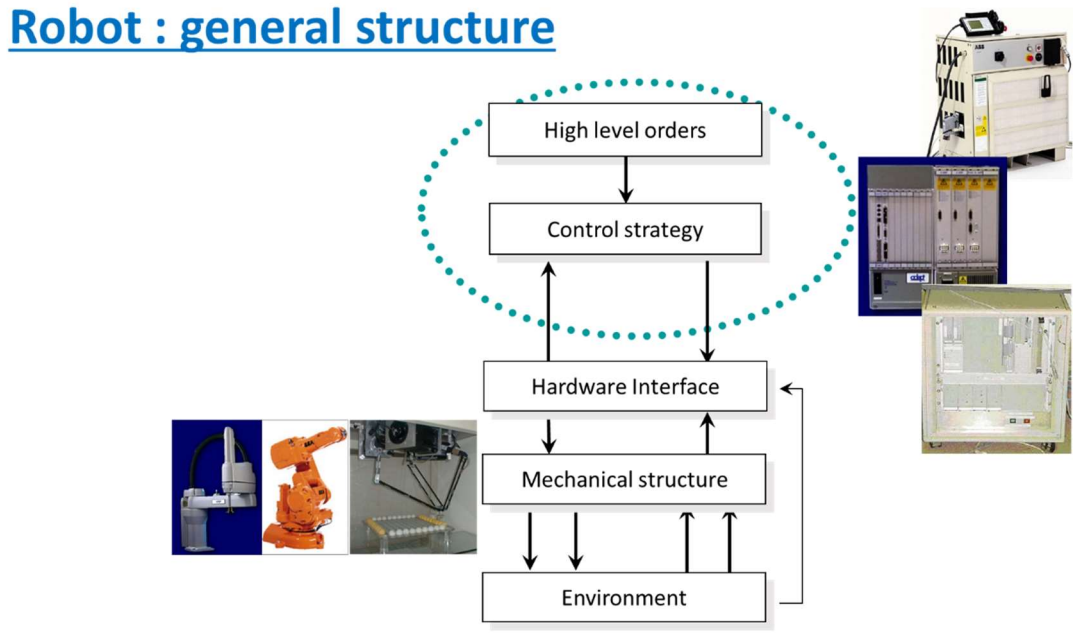


Figure 4.2- General structure of a robot associated with its numerical control

As shown in figure 4.2, the robot is at the center of the system (Robot, control, environment and the operator). The operator expresses his requests to make the robot move to a defined position, outputs to be piloted or tests to be carried out. These orders are processed in software. The hardware interface, as the name (interface) suggests, is the intermediate element between this software part and all the hardware components connected to the robot (mainly sensors and amplifiers). Finally, the robot interacts with the environment through actions (open/close grippers, on-off pneumatic actuators, etc.) and sensors (presence/absence of a part, closing, opening of a door, force sensor, motion detector, etc.). The environment is very much related to the application the robot should be part of; it can be simple positioning of objects, collaborative work around a part, welding, polishing, deburring, or even machining of parts.

This chapter will thus be divided into three sub-chapters which will each introduce and detail the three important parts of robot control, namely:

- Control algorithms.
- Hardware.
- Software.

4.3 Mechanical components of a robot

4.3.1 The axis

The basic element of a robot is the axis. An axis is a motorized mechanical element followed by a transmission. If there is no transmission we will speak of a direct actuation (Direct drive in English).

There are two types of axes:

- A linear axis.
- A rotary axis.

The linear axis is either a simple direct drive type linear motor or an axis obtained with a rotary motor and rotary-linear transmissions[1].

Two examples of motors (ref. Etel)

Linear motors with magnetic rod (ref. LinMot)

Figure 4.3, Examples of linear motors

Concerning transmissions, we mainly distinguish the following types:

- Screw-nut,
- Rack-pinions,
- Belt
- Chains are very little used because they are heavy and not synchronous (polygonal effect).

Linear axis with screw-Nut

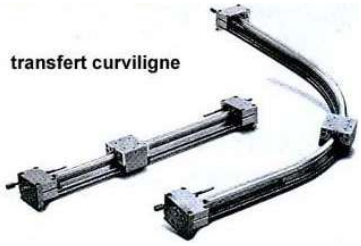
Linear axis with rack pinion mechanism

Linear axis with belt

Figure 4.4, Examples of rotation-linear transmissions

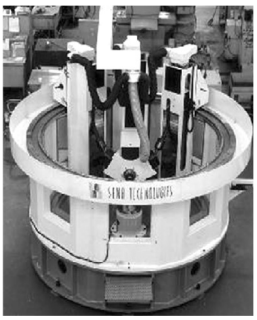
And what else!

Here is the example of some slightly exotic translational systems dedicated to special applications. These systems are not linear but curved:



Pneumatic curvilinear transfer (ref. SMC)

Curvilinear guidance (ref. THK)



Example of a parallel robot with curvilinear guidance (ref. Eclipse)

Figure 4.5, Examples of curvilinear translations

Rotary axes: Rotary axes are often composed of a motor and a transmission. This transmission mainly includes elements such as a belt or gear reducers[2]; sometimes an angle transmission.

[1] For more details see the chapter "actuators"

[2] Gear reducers (gear stages, planetary, cycloidal, harmonic drives,…) are covered in the lecture "Applied and Industrial robotics".

Whether for linear or rotational transmissions, backlash compensation is an operation that is always considered. It often involves additional prestressing, the effect of which on positioning accuracy will be discussed later in this lesson chapter.

4.3.2 The robot

We defined the robot as a poly-articulated, motorized and instrumented mechanical structure of which the axis is the basic component. The combination of the movements of the axes while respecting a given kinematics gives rise to the robot.

Taking for example three separate linear axes, we can obtain the following different structures:

- three-axis robot **Cartesian**,
- a parallel robot like the **Linear Delta** or the **Orion**,
- any **combination** kinematic
- or even **three completely independent axes**.

Cartesian machine 3 axes XYZ

Parallel robot Linear Delta (ref. Unimec).

Figure 4.6, Examples of robots

The construction of the structure, and therefore the choice of the kinematics, defines the type of control required. We refer to **axis control**, **multi-axis control**, **numerical control** or even **robot control**. All these terms are used; sometimes you get lost but what you have to remember is that **they are all valid**.

So, Robots or not Robots.

A pithy turn of phrasing, any machine involving motorized and automated operations is commonly referred to as a "robot". For instance, it may be **loom robots** such as weaving, cutting, folding robots, etc. These robots are also well controlled with axis commands to control their motors in position or in velocity according to the required mechanical operations.

Weaving robot Weaving

robot

Robot with press brake

Figure 4.7, Examples of professional robots

4.4 Closed loop control algorithms

The algorithmic part of robot control concerns the mathematical tools necessary for formalizing the behavior of the robot with its control. This formalization is linked to the following two aspects:

- The adjustment algorithms,
- The generation of robot trajectories, i.e., the geometry of the generated trajectories and the generation of the associated temporal profiles.

4.4.1 Regulation and servo-control aspects

The objective of a closed-loop control of a physical system is to reach a desired setpoint and to remain as close as possible to it. The classic diagram of such a closed-loop control is as follows[3]:

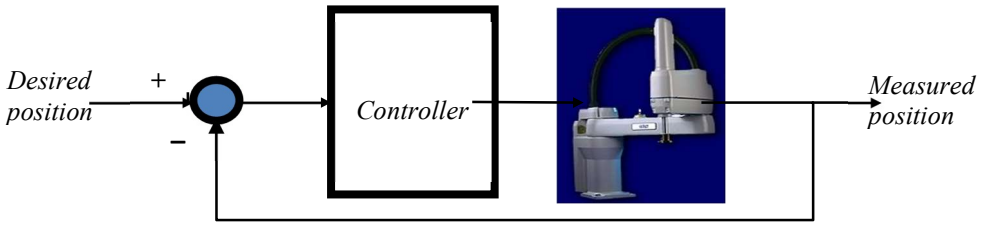


Figure 4.8, Block diagram of the closed-loop control of a robot

The controller is the algorithm that uses the information of measured and desired positions to ***guarantee the position to follow the desired setpoint (respectively of the speed).***

Other terms for the controller:

- control, command, regulation, …
- controller, control unit, regulator,…

To control what?

- the position of an axis,
- the speed of an axis.
- the force (at the joints or at tool output).

Parenthesis

Robot motors are often direct current motors (**DC** for Direct current) brushed (brushed) or brushless (brushless) motors. DC motors are controlled by voltage or current and we respectively refer **to speed** or **torque** control modes, because the speed is proportional to the motor input voltage; respectively the torque is proportional to the input current of the motor (via the speed constant, respectively via the torque constant).

4.4.2 Classic P, PD, PID controllers

The controllers **P**, **PD**, **PI** and **PID** are the most classic controllers used for controlling a robot. These controllers are composed of the proportional, derivative and integral contributions of the error on the position[4]. The proportional contribution is proportional to the error. The derivative contribution is proportional to the derivative of the error. The integral contribution is proportional to the integral of the error.

[3] Pay attention, this diagram does not include the power stage of the robot motors (power amplifiers).

[4] In what follows we will exclusively interest in the generalized position variables. The reasoning with respect to velocities is similar.

Here are the expressions of the different contributions:

Proportional contribution: $u_p = K_p \cdot e$

Derivative contribution: $u_d = K_d \cdot \frac{de}{dt}$

Integral contribution: $u_i = K_i \cdot \int_0^t e(\tau)d\tau$

The P controller includes only a proportional contribution,

The PD controller includes the proportional and derivative contributions and is written as follows:

$$u_{PD} = u_p + u_d = K_p \cdot e + K_d \cdot \frac{de}{dt}$$

Other writing $u_{PD} = K_p \cdot \left(e + T_d \cdot \frac{de}{dt}\right)$

The PI controller includes the proportional and integral contributions and is written as follows:

$$u_{PI} = u_p + u_i = K_p \cdot e + K_i \cdot \int_0^t e(\tau)d\tau$$

Another writing $u_{PI} = K_p \cdot \left(e + \frac{1}{T_i} \cdot \int_0^t e(\tau)d\tau\right)$

In this case we refer to the integration time constant T_i. Pay attention that the integrator effect is more important as the observation of time T_i is small.

The PID controller includes the three contributions, the proportional, the derivative and the integral contributions and is written as follows:

$$u_{PID} = u_p + u_d + u_i = K_p \cdot e + K_d \cdot \frac{de}{dt} + K_i \cdot \int_0^t e(\tau)d\tau$$

Another expression $u_{PID} = K_p \cdot \left(e + T_d \cdot \frac{of}{dt} + \frac{1}{T_i} \cdot \int_0^t e(\tau)d\tau\right)$

To understand the contributions of the proportional, derivative and integral terms, consider the torque control of a DC motor. In this case, the calculated command is a torque (or a force for a translational system). Here are the questions which will help to clarify our purpose:

Question: To which mechanical element corresponds the proportional action?

> **Answer**: It is an element whose applied force is proportional to the elongation around a position of equilibrium. So, it is a spring. This to say that **the proportional action produces the same behavior as a spring and the proportional gain Kp contributes to the stiffness of the control.**
> The natural frequency of the closed loop system is also directly related to this proportional action.
>
> **Notes**:
>
> - The higher the gain Kp, the stiffer the system (conversely, the lower Kp, the more compliant or softer the system).
> - The higher the gain Kp, the higher the natural frequency of the system and the system responds quickly (the system time constant is lower).
> - The higher the gain Kp, the lower the control sampling period must be (due to the low time constant).
> - The higher the gain Kp, the more the closed loop system is likely to oscillate because of the saturation of the control variable which implies that the system operates in open loop during the saturation.

Question: To which mechanical element corresponds the derivative action?

Answer: It is an element whose applied force is proportional to the derivative of the elongation around a position of equilibrium. It is therefore a viscosity or a viscous friction. This to say that **the derivative action produces the same effect as a damper whose damping coefficient is Kd.**

Notes:

- The higher the Kd gain, the more the system is braked and damped.
- The higher the Kd gain, the slower the system becomes.
- A compromise must be found in order to ensure the best possible dynamic behavior (a damping coefficient equal to 1 corresponds to the critical regime: see figure).
- The bypass operation amplifies the noise. Particular attention must be given to the static behavior when the setpoint is constant and the quantization of the speed becomes predominant (see paragraph 3.1.4).
- A high Kd gain may cause chatter in behavior due to velocity quantization.
- The derivative action is an anticipation on the dynamic behavior (because of the information of increase and decrease of the error). This information is additional to the simple error information, which modulates the value of the command as proportional to the velocity to improve the dynamic behavior.

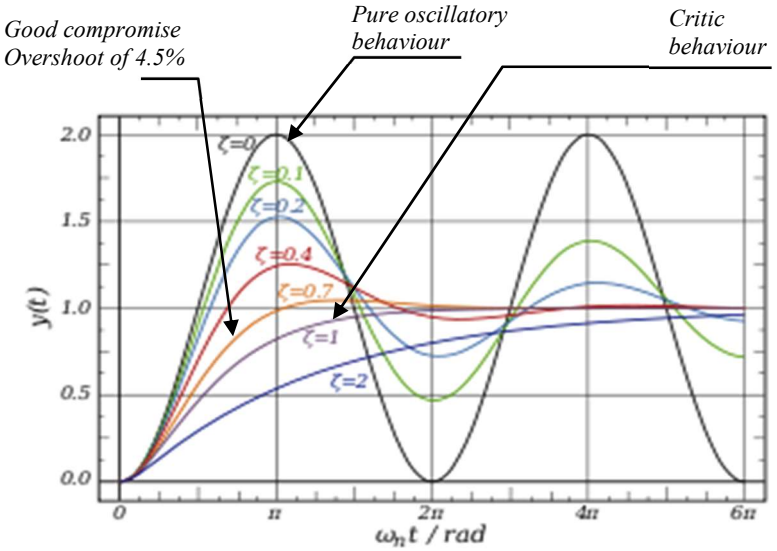


Figure 4.9, Responses of a typical second-order system

Question: To which element corresponds the integral action?

Answer: The integration operation is an accumulation operation, i.e. analog to that of a capacitor which charges when it is put under direct voltage. Integral action therefore amounts to accumulating the command quantity as long as the error is not canceled out. **When there is a static difference**, the proportional action has no effect (because the difference is static and the proportional contribution therefore remains constant). The derivative action is zero (within a quantification error). There is only the integral action which can have effect to cancel this static variation. The integrator is a perturbation is perturbation estimator that dynamically compensates for the estimated perturbation, till converging in the steady state to the right value which is at the origin of the steady state error. (ref, additional document P / PD and PID control on Moodle).

Notes:

- The higher the Ki gain, the faster the static error is reduced.
- The higher the Ki gain, the greater the integral action and the faster the integrator is charged.
- If the integrator loads quickly, you will have to pay attention to changes in direction that require the integrator to unload. A limitation of the maximum load of the integrator is necessary: “Anti-Reset Windup” (ARW) operation.

Exercise 4.1. Current control of a direct current motor (DC, Direct Current)

Consider a DC motor with torque constant k_t and inertia J_m. This motor is current controlled to rotate an inertial load J_L through a gear reducer **n**.

- Establish the dynamic model of the motor with and without viscous friction of coefficient f_v (current-position relationship)
- Prove that in the absence of viscous friction, that it is impossible to control the position of the motor with simple proportional feedback?
- Loop the motor with a **PID** and determine the closed loop transfer function.

Exercise 4.2. Effect of Dry Friction

Consider a DC motor with torque constant $\mathbf{k_t}$ and inertia $\mathbf{J_m}$. This motor is current-controlled to rotate an inertial load $\mathbf{J_L}$. This system presents a dry friction of torque Γ_{dry}. The figure below shows how dry friction acts (ω is the speed of the motor)

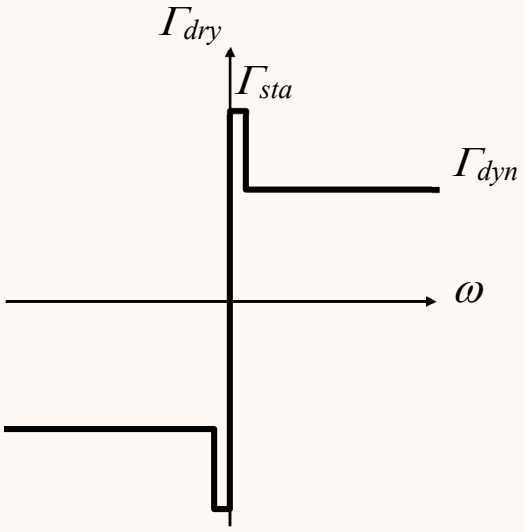


Figure 4.10, Coulomb model of dry friction

- Show that in the presence of dry friction, the command in position of the ***PD*** automatically induces a static deviation on the position.
- What solution do you propose? Show the effectiveness of the proposed solution.
- Is it the same in the case of any disturbing torque?

4.4.3 Cascaded PID Controller

Another technique that is also widely used is the technique of cascaded position control with velocity control.

The following diagram represents a PI speed loop of a motor;

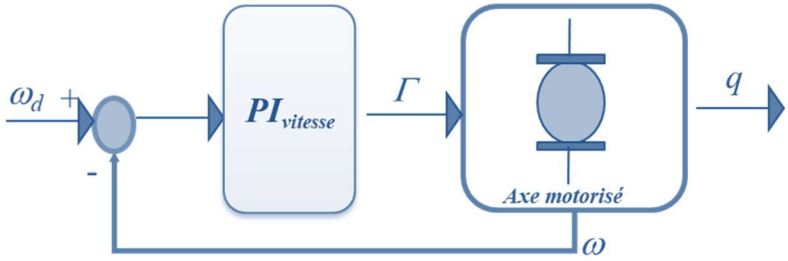


Figure 4.11, PI speed loop of a DC motor

This technique is widely used because of the availability of speed loops on all motor drives currently available on the market. It then suffices only to implement a additional cascaded position control loop. Fr instance, the fast speed loop can be on the motor drive and the position loop on the computer, or both on the motor drive. The following block diagram illustrates a cascaded position / velocity control loops.

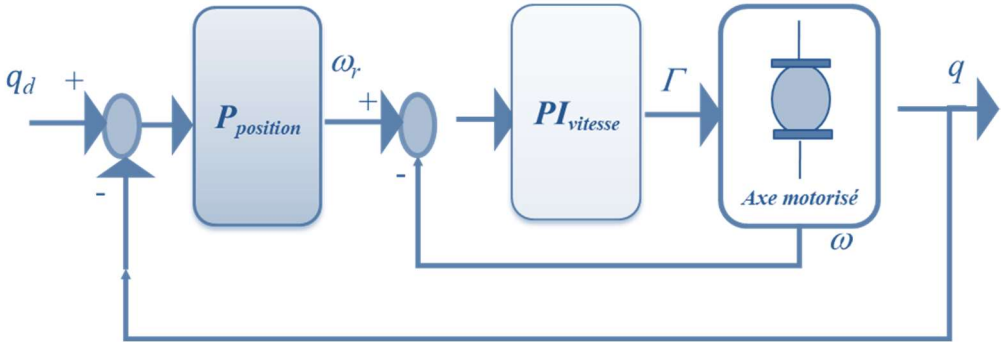


Figure 4.12, PI speed loop cascaded with a P loop in the position of a DC motor

To improve this loop two a priori contributions can be added: the speed a prior and the torque a priori. This gives the following control diagram:

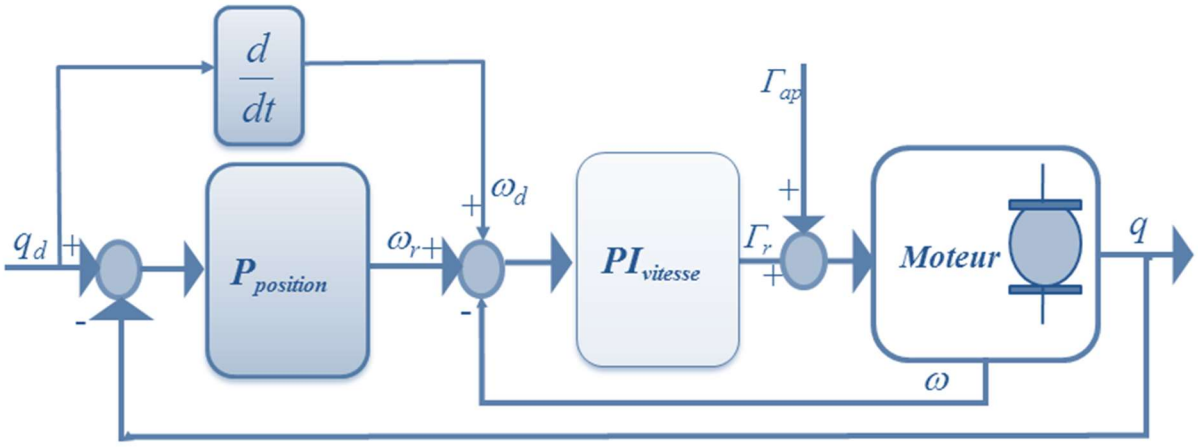


Figure 4.13, Position-speed cascaded loop with a speed a priori and a torque a priori

The expression of the cascaded PID control law with a speed and torque prior is given as follows:

$$\Gamma = \Gamma_r + \Gamma_{ap} = K_{pv} \cdot e_\omega + K_i \cdot \int e_\omega(\tau)d\tau + \Gamma_{ap}$$

Where: $e_\omega = \omega_r + \omega_d - \omega$

ω_r is the contribution of the position regulator $P_{position}$, $\omega_r = K_{pp}(q_d - q)$

ω_d is the target speed of the robot trajectories.

ω is the measured joint velocity.

Note:
Depending on how the mechanical system is implemented (motorisation, transmission, load and disturbance), one can imagine that the position and speed loops are both PIDs. It is therefore necessary to be very careful with the errors of quantifiation of the speed (paragraph below).

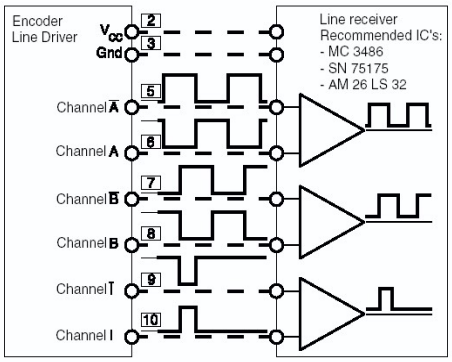
4.4.4 Control, measurement and quantification

The sensors used for measurement, their resolutions and their technologies are important aspects for the positioning accuracy.

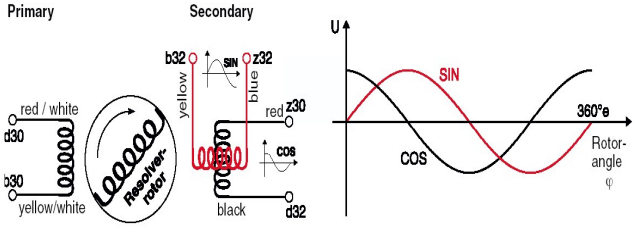
Parenthesis
The sensor resolution is the smallest value that can be measured.
Accuracy or the reachable precision **concerns the entire system**, ie the mechanics, the electronics, the sensors and the used control algorithms. The accuracy is at best equal to twice the resolution of the sensors.

A few reminders about position sensors

To measure the position of robots, incremental encoders or resolvers are most often used. In most cases, the signals are transmitted from the sensors to the numerical controller in a differential way to reduce noise. Moreover, the incremental encoders are transmitted by digital pulses which makes them quite robust to classic analog noise.



Signals from an incremental sensor



SIN/COS sensor

Figure 4.14, Position sensors

Analog sensors of the potentiometric type are not recommended for servo operations because they are very sensitive to noise and their resolution is limited.

Exercise 4.3

Consider a 10-turn multi-turn potentiometer powered between +/- 10 volts. The acquisition is carried out using a 12-bit analog-to-digital converter.

- What is the position resolution of this sensor?
- Is there a possibility to improve this resolution? Discuss your proposal.

A few reminders on velocity sensors

The velocity used for robotic servo-control is most often obtained by digital derivation of the position (the analog tachometric signal being too noisy).

Let us consider

- T_s be the control sampling period.
- R(θ): the position resolution.
- R(ω): the velocity resolution.

Obtaining the velocity by numerical derivation of the position at time (kT_e) is as follows:

$$\omega(kT_e) = \frac{\theta(kT_s) - \theta((k-1)T_s)}{T_s}$$

Which, gives the resolution of the following velocity: $R(\omega) = \frac{R(\theta)}{T_s}$

In steady state, the control value should be constant. This assumption is not always true because the control value will fluctuate according to the position and velocity quantization errors (these quantization values correspond to the resolutions of the position and velocity measurements). The amplification effect of the digital derivation, because of the division by T_s multiplied by the derivative action, increases the noise of the derivative action: $K_d \cdot \frac{de}{dt} = K_d \frac{e(kT_s) - e((k-1)T_s)}{T_s}$.

Let u_{dq}, be the control value corresponding to the effect of the derivation quantization noise. u_{dq} is expressed as follows:

$$u_{dq} = K_p \cdot T_d \cdot R(\omega) = K_p \cdot T_d \cdot \frac{R(\theta)}{T_e}$$

Therefore,

- the more we want to dampen our system (high Td), the more this quantization control noise u_{dq} increases.
- The more we want the system to be dynamic or rigid (high Kp), the more this quantization control noise u_{dq} increases.

This noise is unfavorable because it may excite the natural frequencies of our mechanics, cause limit cycles or even harm our mechanical components. To reduce this noise, suggested solutions are as follows:

- Reduce the resolution of the sensor; which has a direct impact on the complete system and its cost.
- Reduce dynamics or damping; which has a direct impact on dynamic performance.
- Increase the sampling period which also implies reducing the dynamics by the increasing the response time constraint. A possible solution would be to operate with two sampling periods: this approach is called oversampling. **A faster sampling** rate is used to calculate the proportional and integral actions and **a slower sampling rate** to calculate the derivative action.
- Use derivative filters or derivative estimators, different from the Euler filter. A good compromise would be to carry out a derivative over 2 sampling periods as follows:

$$\omega(kT_e) = \frac{\theta(kT_e) - \theta\big((k-2)T_e\big)}{2 \cdot T_e}$$

Which, gives the following resolution of the velocity: $R(\omega) = \frac{R(\theta)}{2 \cdot T_e}$

Exercise 4.4. Quantization noise

Consider a motor with a -velocity gear reducer n = 32, and a 500-line incremental encoder. This motor is controlled with a PD controller at a sampling rate of 1 kHz.

- What is the positional resolution of this axis?

The velocity is obtained by numerical derivation.

- What are the velocity resolutions when the derivation are respectively implemented over 1 or 2 sample periods?
- The gear backlash is about 0.5 degrees. Please comment?
- What is the quantization noise reported on the current?

4.4.5 Coupled model-based robot controls

4.4.5.1 MIMO system

In this section we consider robot controllers based on coupled dynamic models. These models are multi-input and multi-output models (MIMO, Multi Inputs Multi Outputs).

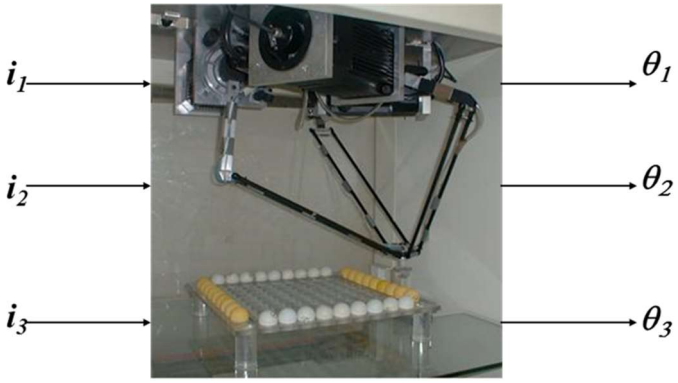


Figure 4.15, Delta MIMO robot with three inputs and three outputs (i_1, i_2 and i_3 are the motor currents and θ_1, θ_2 and θ_3 are the motor positions)

4.4.5.2 Notion of coupling:

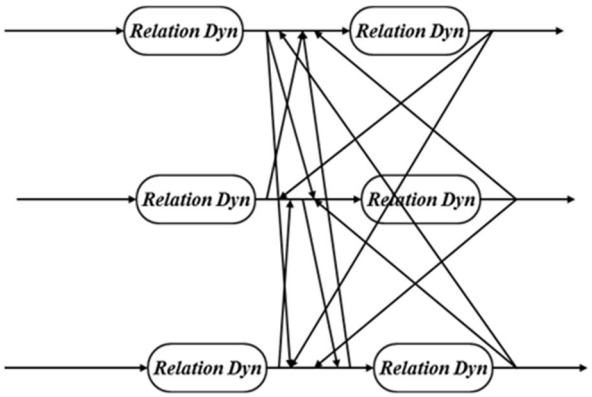


Figure 16, Representation of dynamic couplings

Dynamic couplings concern the influence of the movement of one axis on another. These couplings are function of the robot positions and velocities. The dynamic model of a robot can be represented as follows:

$$\Gamma = B(q).\ddot{q} + G(q) + C(q,\dot{q}) + F(q,\dot{q}) + K(q)$$

Generalized torque

Matrix of Inertia

Gravity

Centripetal and Coriolis

Friction

Stiffnes

4.4.5.3 Robot models and control model

We put the above model in the following form: $\Gamma = B(q).\ddot{q} + H(q,\dot{q})$

This writing gives rise to different other model writings according to the knowledge of the system parameters and as required.

$\ddot{q} = (B(q))^{-1}(\Gamma - H(q,\dot{q}))$	Is the second-order differential dynamic model of the robot
$\Rightarrow \quad \dot{x} = f(x) + g(x).u$	Is the state dynamic model of the robot
$\ddot{q} = \left((B^{mod})^{-1}\left(\Gamma - H^{mod}()\right)\right)$	Dynamic model of the robot obtained during the first modeling
$\ddot{q} = (B^*(q))^{-1}(\Gamma - H^*(q,\dot{q}))$	Simplified dynamic model
$\dot{x} = f^*(x) + g^*(x).u$	State model for simulation
$\dot{x} = f(x) + g(x).u$	Simplified state model for control
$\ddot{q} = (B(q))^{-1}(\Gamma - H(q,\dot{q}))$	Robot model of the second order for the control

Example

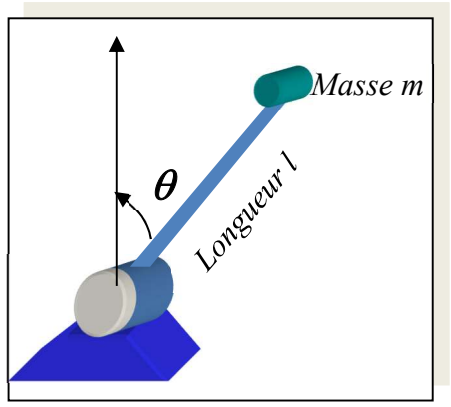


Figure 4.17, inverted pendulum

Representation model:

$$\ddot{\theta} = \frac{k_c}{J}i - \frac{mgl}{J}sin(\theta) - \frac{\alpha_{vis}(\omega,\theta)}{J}\omega - \frac{\Gamma_{dry}}{J} - \frac{\Gamma_{pert}}{J}$$

Γ_{pert} is a perturbation torque function of the play, temp, wear, etc.

Simplified model

$$\ddot{\theta} = \frac{k_c}{J}i - \frac{mgl}{J}sin(\theta) - \frac{\alpha^*_{vis}}{J}\omega - \frac{\Gamma_{dry}}{J}$$

State model for simulation

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{mgl}{J}sin(x_1) - \frac{\alpha^*_{vis}}{J}x_2 - \frac{\Gamma_{dry}}{J} + \frac{k_c}{J}i$$

$$\underline{such\ that.\quad (x_1,x_2) = (\theta,\omega)\ et\ u = i}$$

State model for control

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{mgl}{J}sin(x_1) - \frac{\alpha^*_{vis}}{J}x_2 + \frac{k_c}{J}u$$

4.4.5.4 Methods for synthesizing control laws for robots

Several control techniques exist for controlling robots. We have cited in the previous paragraphs the classic techniques which do not take into account the couplings between the axes of the robot (PD, PI, PID and cascaded PID loops).

Other techniques are based on coupled models of robots and sometimes differ according to the chosen representation model. Three types of dynamic representations exist:

- Second-order robot models (ref, previous paragraph)
- Nonlinear model, first-order state space model.
- Linear state space model, obtained from a tangent linearization. In this case, the linear control techniques are applicable (stabilizing state feedback, linear decoupling controls, RST, adaptive controls, etc.).

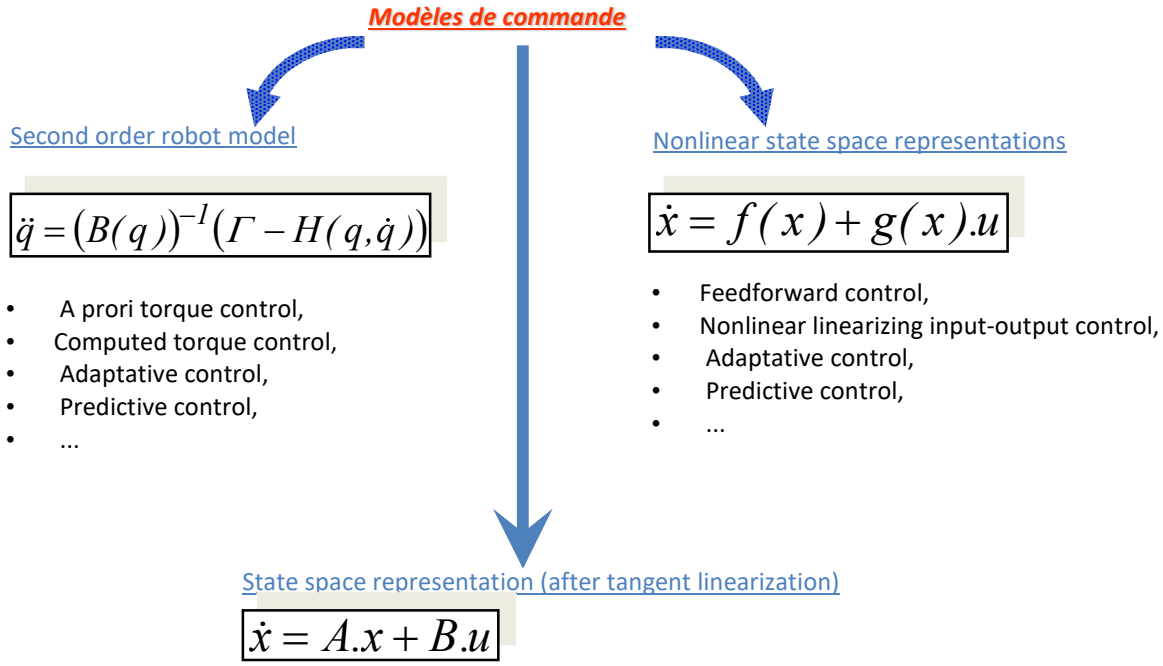


Figure 4.18, Summary of some robot control techniques

4.4.5.5 method by a priori torque compensation

Principle,

The a priori torque is the torque calculated using the inverse dynamic torque model according to the desired trajectories (positions, velocities and accelerations) . If the model is perfect (ie there is no modeling error), controlling the robot with this a priori will involve finding exactly the desired trajectories (ie q ***= q_d***).

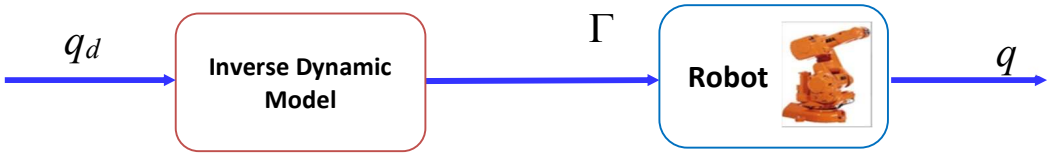


Figure 4.19, robot controlled in open-loop with its inverse dynamic model

It is unfortunately not possible to drive the robot in an open loop in this way because the models are never perfect and the slightest external disturbance will lead to unknown and undesirable behaviors due to the lack of feedback sensors. This a priori torque is helpful but not sufficient. The a priori is a tool which is used to improve the dynamic performance of positioning because it takes into account the couplings according to the desired positions, velocities and accelerations.

The model of the robot does not correspond to reality!

The following control diagram highlights the use of the inverse dynamic model as an a priori to compensate (in a priori) the known non-linearities and couplings. The rest of the imperfections are compensated by the linear controller added to this a priori.

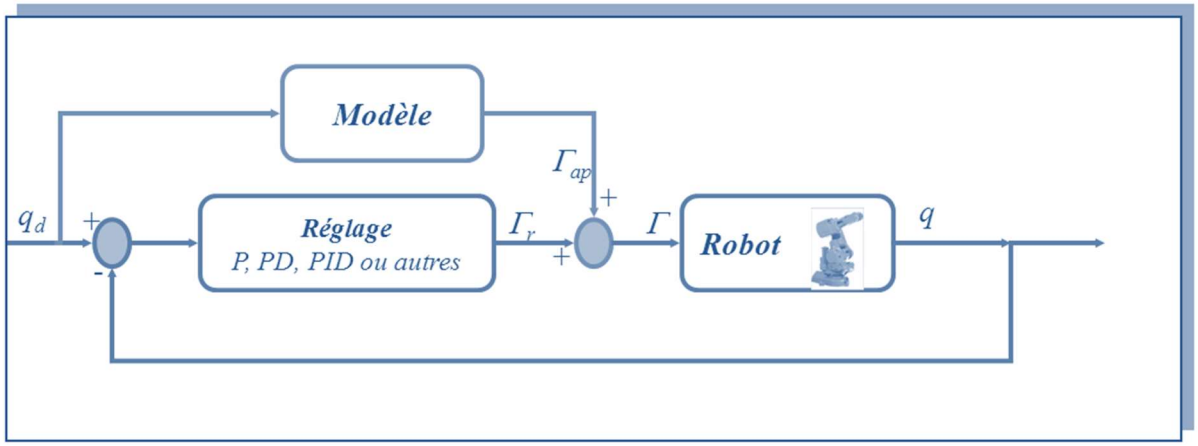


Figure 4.20, A priori torque control diagram.

4.4.5.6 *Computed torque controller, known as nonlinear linearizing input output controller*

This control technique is also called the linearizing input-output control technique. The robot is a physical system with a non-linear input-output dynamic with motor torques as Input vector and motor positions as Output vector (The only special case is that of Cartesian robots which are decoupled linear systems). The input-output linearization control technique amounts to compensating for the non-linearities by looping and making the behavior linear with respect to a new input. More precisely, the dynamics with respect to this new input is equivalent to a double integrator for each joint.

From the following nonlinear model:

$$\ddot{q} = (B(q))^{-1}(\Gamma - H(q, \dot{q}))$$

Let the double integrator of the position variable be written as follows:

$$\ddot{q} = v$$

v is a new input and the system (input-output) which is (v, q) is a linear system with a dual integrator.

How to achieve this linearization?

By the following loop: $\Gamma = B(q).v + H(q,\dot{q})$

Which leads to the following closed loop expression:

$$\ddot{q} = (B(q))^{-1}(B(q).v + H(q,\dot{q}) - H(q,\dot{q})) = v$$

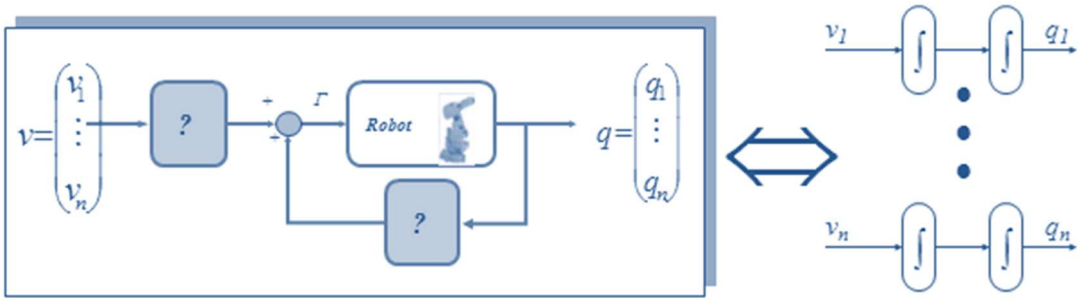


Figure 4.21, Linearizing loop (left) to transform the system into a cascade of double integrators (right)

It is now necessary to stabilize the double integrators by a linear loop of the type, PD, PID, or any other advanced controller (adaptive, sliding mode, ...)

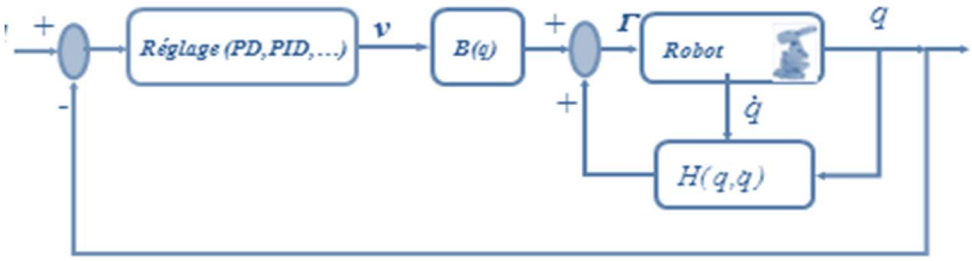


Figure 4.22, Linearizing looping and stabilizing adjustment

4.4.5.7 Exercise 4, Control of a robot axis by a priori and by the linearizing control

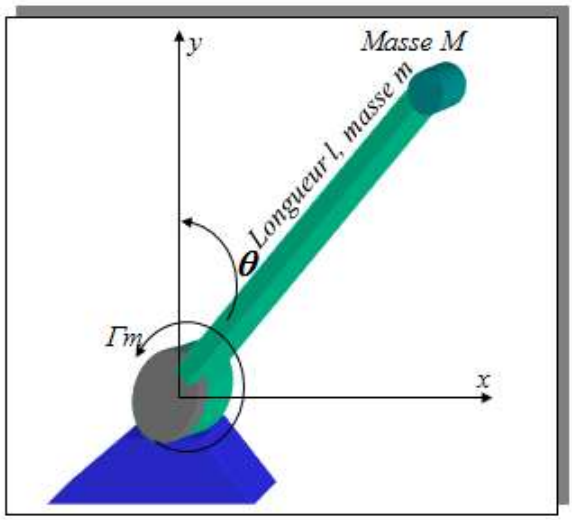


Figure 4.23, inverted pendulum

Consider a DC motor actuated robot axis- kt and Jm are respectively the torque constant torque and moment of inertia of the motor. We consider the current control mode.

- Establish the dynamic model of the motor with and without viscous friction (current-position relationship). What is the transfer function?
- Loop the motor with a PD and an a priori torque command.
- Loop the motor with a PD and a command with the linearizing command.

4.4.5.8 *Principle of an adaptive control with input-output linearization:*

The linearizing loop studied previously is expressed as follows: $\Gamma = B(q).v + H(q,\dot{q})$.

Adaptive control techniques address the unknown terms of the model and come out to using estimators which identify these unknown terms by iterative algorithms. For the models of robots, one of the techniques peculiarly focus in the identification of the terms of the inertial matrix B(q) and the term $H(q,\dot{q})$.

Let $\tilde{B}(q)$ and $\tilde{H}(q,\dot{q})$ these terms to be identified. The model to be estimated is then expressed as:

$\Gamma = \tilde{B}(q).v + \tilde{H}(q,\dot{q})$.

The basic diagram of this control technique is given by the following figure:

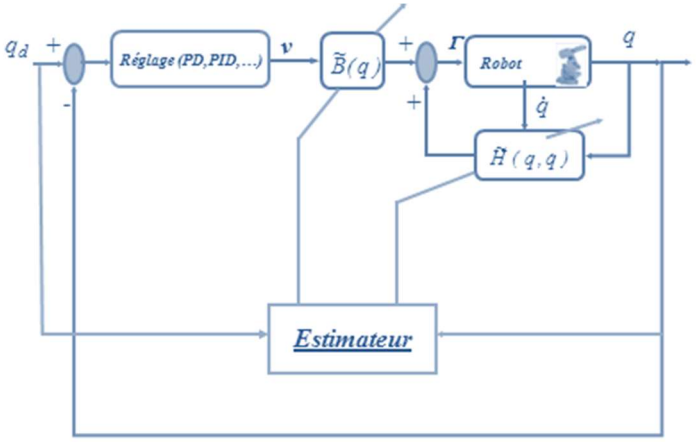


Figure 4.24, Self-adaptive linearizing looping and stabilizing adjustment

4.4.6 Conditions for couplings to be considered as disturbances

Whatever the robot considered, the dynamic equation for each joint i can be written as follows:

$$\ddot{q}_i = k_{ci}.i_i - f_{vi}.\dot{q}_i - \Gamma_{couplings}(q_i, q_j, \dot{q}_i, \dot{q}_j)$$

- k_{ci} is the torque constant of each joint actuator i,
- f_{vi} is the coefficient of viscosity at each joint i,
- i_i is the motor current at joint i,
- $\Gamma_{couplings}(q_i, q_j, \dot{q}_i, \dot{q}_j)$ is the effect of the other joints on the considered joint i.

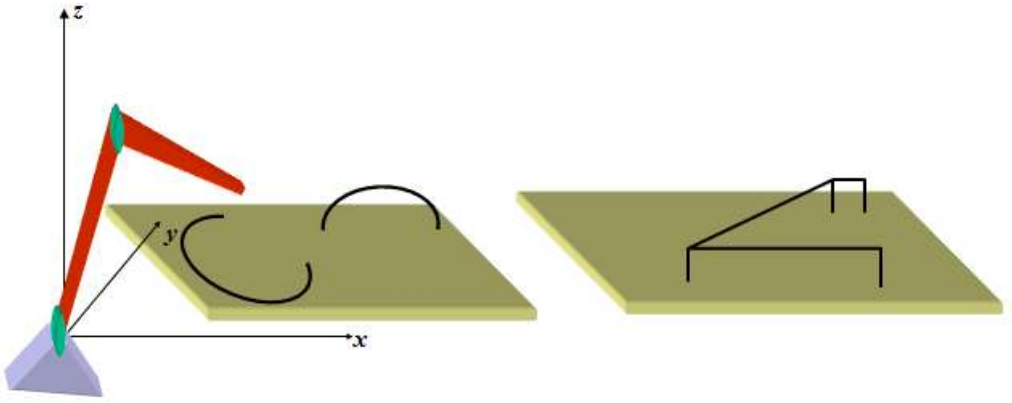
In some cases, the couplings $\Gamma_{couplings}(q_i, q_j, \dot{q}_i, \dot{q}_j)$ can be considered as constant or quasi-constant disturbances and thus be compensated either by an integral action (term I of the PID) or a disturbance estimator implemented thanks to an observer or a parametric identification. *This is possible if the following conditions are verified:*

- the dynamic couplings are weak. This can take place for example if the reduction ratios of the motors are so high that the inertia brought back to the actuators are very low (therefore the inertial torques),
- The couplings are very slightly variable (thanks to a low dynamic) so that their effects can be eliminated by an integral action

4.4.7 The Be careful of the control tuning

- Always specify the type of adjustment and the conditions of implementation:
    - PI, PID, linear or not, with dynamic model or not,
    - Uses of certain artefacts such as dead zone, filtering,
- Always specify the sample period used. Sometimes, several sampling periods are used as in the case of a cascaded velocity/position loops-
- When making comparisons, always compare the comparable by specifying:
    - the scales,
    - the working areas,
    - the values of the amplitudes,
    - the desired time constants.
- Specify the difficulty of the choice of the parameters of the adjustment and the way in which they were sized.

4.5 Trajectory generation for manipulation robots



Robot trajectory generation addresses trajectory patterns which connect two points of the robot workspace, as well as the way in which the velocity and the acceleration evolve over time.Trajectory generation is therefore associated with the following two aspects:

- Geometric aspect, which considers the description of the set of intermediate points from the starting to the ending point.
- Temporal aspect, which considers the description over time of the evolution over time of each generalized coordinate and its derivatives (velocity and acceleration). These evolutions as a function of time will later be called respectively acceleration, velocity and position profiles.

The geometric and temporal aspects of a trajectory are definitely dependent, because any spatial (geometric) error induces a temporal error and vice versa. The shapes of the paths are very important and the time profiles must therefore be well chosen so as not to accelerate or decelerate anywhere on these paths. A good example to that is a circular path implemented at a constant velocity. The constant velocity along this circular path will induce a non-null acceleration, which is the centripetal acceleration V^2/R, R is the ray of the implemented circle.

4.5.1 Reminder on the transformation of coordinates

The transformation of robot coordinates, also called geometric model, is very important in trajectory generation. The transformation of the coordinates of the robot is the relation which links the joint space and the tool apace of the robot. Regardless particular cases, the path of the robot is defined in its tool space **where the operational task is defined**, and the control of the motors is implemented in its **joint**.

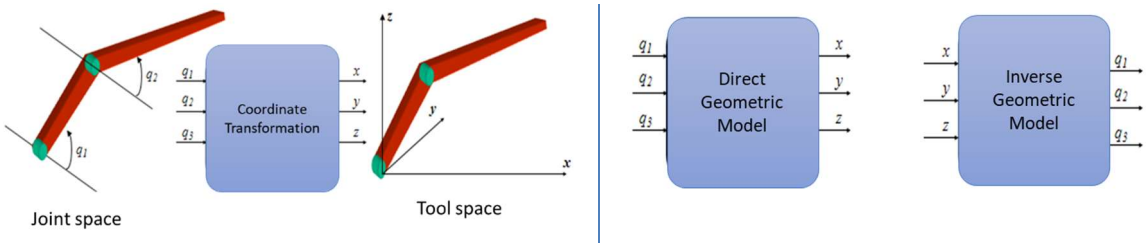


Figure 4.25, Joint and tool spaces (left) and direct and inverse coordinate transformations (right)

Summary of the uses of each model:

	Inverse Geometric Model	Direct Geometric Model
Use	Control of a robot, by calculating the joint coordinates from defined trajectories in tool coordinates.	• Initialization of the robot by the deduction of the operational starting point. • Learning points by measuring operational coordinates. • Control in operational space by measuring operational coordinates.

4.5.2 Admissible trajectories

A trajectory is admissible if:

- it is geometrically feasible; it does never leave the robot workspace,
- it is temporally feasible; the velocities and accelerations can be achieved by the motors.

To that, it is necessary that:

- The power electronics is appropriate to perform the required motor voltage and current.
- The hardware is appropriate for data acquisition and computing the required control in the convenient sampling period.
- The control algorithm is appropriate for the required performance and its parameters are well adjusted.
- The motorization is adequate for the required dynamic performance (velocities and torques).

4.5.3 Geometric aspects of robot trajectories

The basic patterns used to define a robot's trajectory are the straight line and the arc of a circle. The term **interpolation** is a very common term to describe a robot or machine tool path.

What is path interpolation in the CNC world? In the case of trajectory generation, we refer to interpolated axes when the movements of these axes are **geometrically dependent**.

The basic known interpolations in the world of multi-axes machines and robots are:

- **linear** interpolation,
- **circular** interpolation (clockwise and counterclockwise).

Other interpolations also exist:

- **spiral** interpolation,
- **helical** interpolation,

Here, are some typical trajectories for picking up and depositing objects *(pick and place)*:

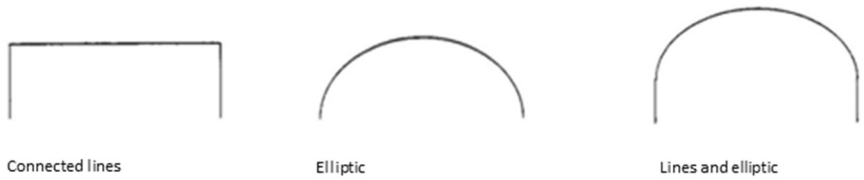


Figure 4.26, Typical pick-and-place trajectories

4.5.4 Temporal aspects of robot trajectories

To characterize the trajectory of the robot over time, we refer to **profiles.** The profile defines the time curve of the desired position, the desired velocity, or the desired acceleration. We generally refer to ***velocity profiles*** and **acceleration profiles,** the position profile is automatically deduced from that of the velocity by integration.

The simplest profile, to define the desired position, is the step profile. It is a **0-order profile** that is discontinuous in position, which means that we require the robot to instantaneously move from one defined point to another point. The position step profile, commonly used in the theory of control to deduce transfer functions, is too aggressive and generates a lot of oscillations and is not appropriate for tracking.

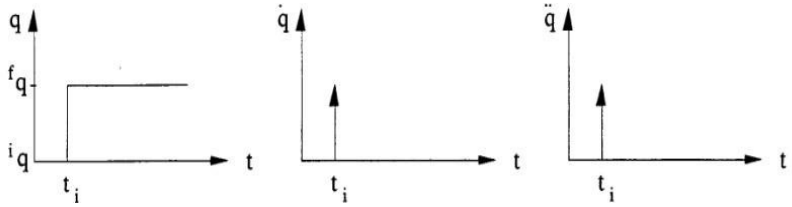
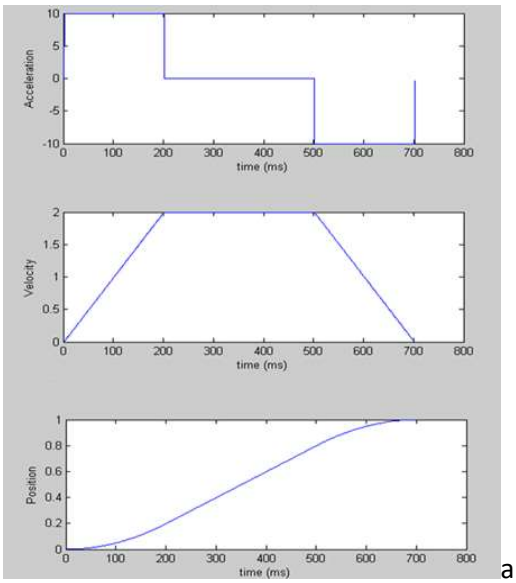


Figure 4.27, Step position to position trajectory, interpolation of order 0.

The most known and used profile is the trapezoidal velocity profile (also called Bang-Bang in acceleration and parabolic in position).



a

Figure 28, Trapezoidal profile in velocity (Bang Bang in acceleration)

Unfortunately, the trapezoidal velocity profile also implies a too brutal solicitation of the mechanics because of the instantaneous transition in the acceleration, thus of the motor torque. To avoid this, we simply smooth the acceleration rise by adding a jerk phase (the jerk is the derivative of the acceleration). This new profile corresponds to a **trapezoidal acceleration profile**, which is smoother for the mechanics and is also called parabolic in velocity profile (ref. side figure).

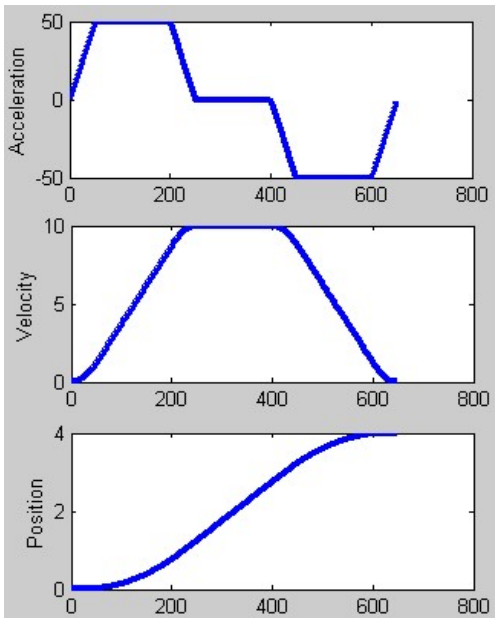


Figure 4.29, Trapezoidal profile in acceleration, or parabolic in velocity

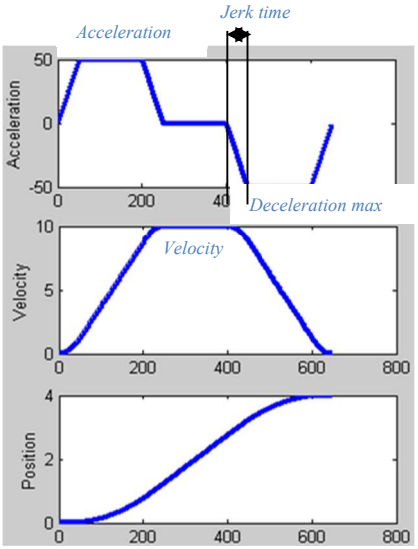


Figure 4.30, Parameters of the trapezoidal profile in acceleration

4.5.5 Continuous profile generation

Let us consider the points A, B, C, D, E and F that define a path from point A to point D. The common path of a robot from point A to point F amounts to traversing the intermediate positions with a stop at each point (Figure 4.31).

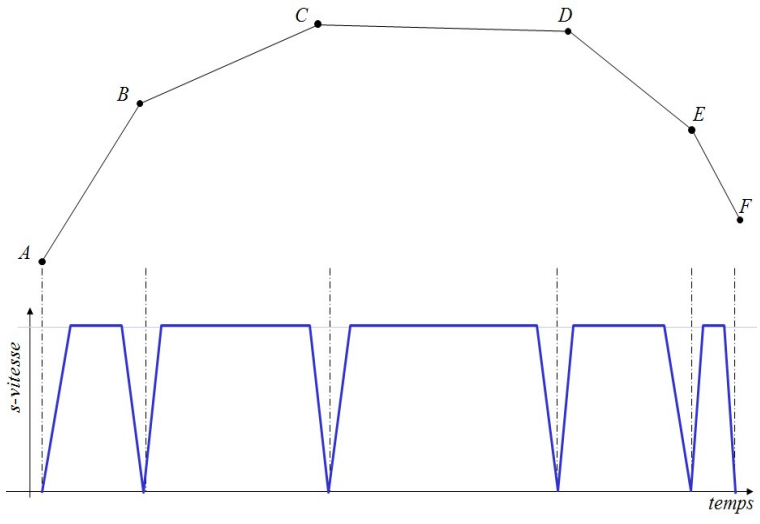


Figure 31, fittings with stop at each intermediate position

In machine tools and in certain robotics applications (such as welding or painting), it is not necessary to stop at intermediate positions. Continuous profiles are thus used with a starting phase, phases at constant velocity and a braking phase (figure 4.32).

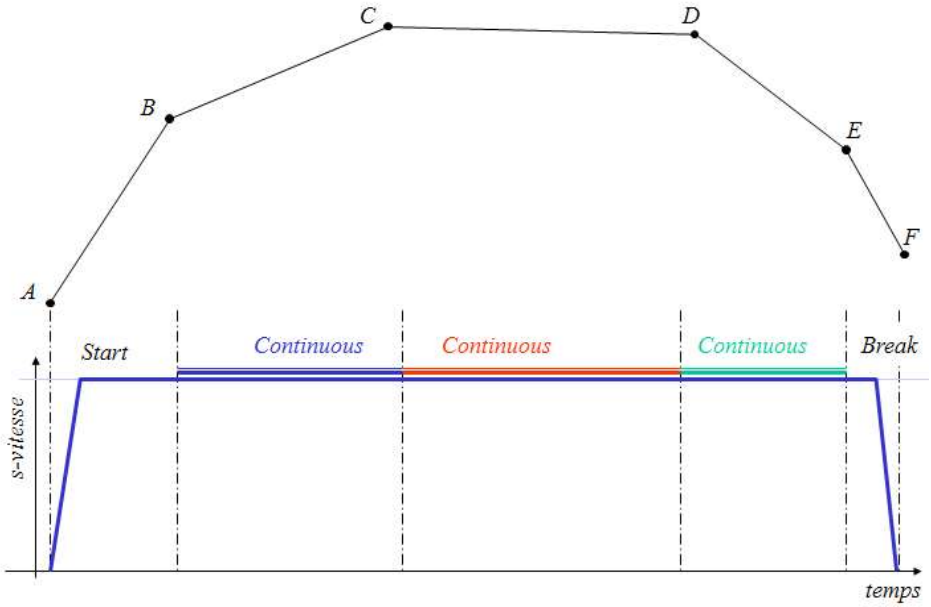


Figure 4.32, Continuous connected velocity profiles without stops at intermediate positions

<u>Continuous profiles (continued): management of connections:</u>

To reduce the effects of high accelerations at the intermediate positions, we may respectively decelerate then accelerate, for example by modulating the acceleration and deceleration depending on the angle at these transition points. This will make the acceleration smoother.

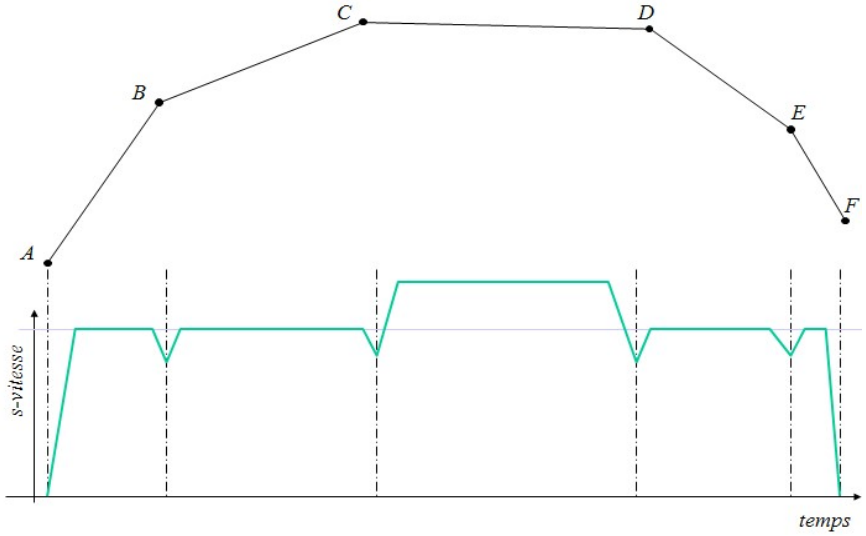


Figure 33, Continuous transitions with management of accelerations and decelerations at intermediate positions

4.1.1 Synchronization of axes

In some multi-axes implementations, geometric dependencies do not exist, by considering for example two rotary axes of a tool changer, two independent linear axes, two conveying axes, etc. In that cases, multi-axes trajectory interpolation does not apply and we have to consider multi-axes synchronization to generate the desired trajectories.

To illustrate this situation, consider two axes x1 and x2, the movement from position 1 defined by the coordinates (p1_x1, p1_x2) to position 2 (p2_x1, p2_x2), can be performed either by:

- Sequentially moving the axes 1 and 2, one after the movement ending of the previous one.
- Independently moving the two axes, ie each axis moves according to its own dynamics defined by the its velocity profile.
- Synchronizing the movement of the two axes, by constraining their movements to the slowest velocity profile.

The following figure illustrates the three operating modes for trajectory generation:

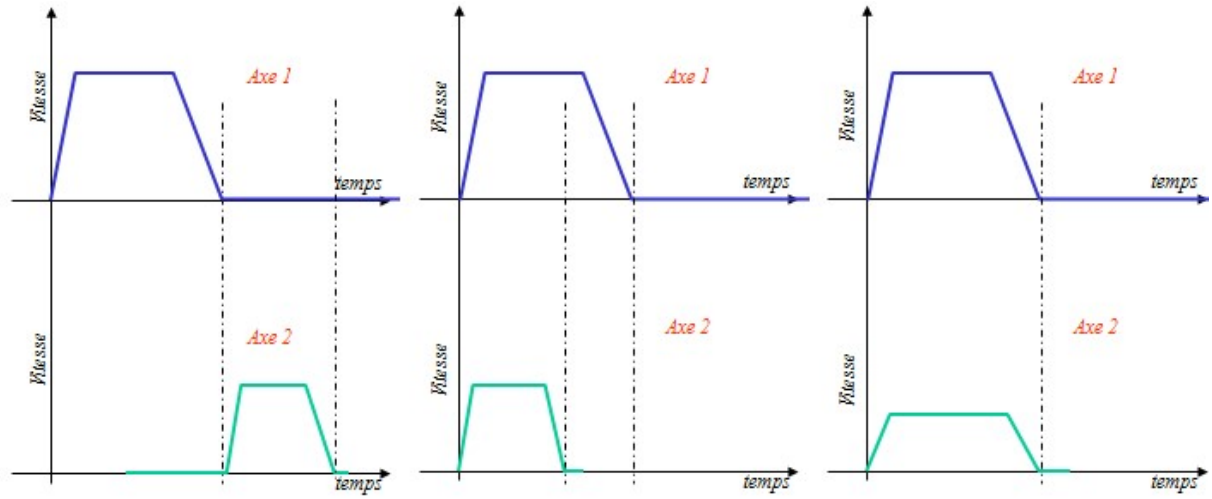


Figure 4.34, Types of axis synchronization (left on the right, sequential, independent, synchronized)

Notes:

- The interpolation of the axes concerns the geometry of the robot trajectories.
- Synchronization of the axes does not define any geometry of the path and it is only related to the management of the velocities and acceleration profiles.

4.2 The hardware part of a robot control

4.2.1 Elements of a control cabinet

The control cabinet is made up of the following elements:

- Processor board, which embeds all the software modules of the controller.
- Electronic boards, to acquire sensor signals, as:
  - Acquisition of digital signals for on/off sensors.
  - Acquisition of analog signals, for example to measure forces, temperature, etc.
  - Acquisition of specific signals such as quadrature counters for incremental encoders.
  - …
- Electronic output boards, to set electric signals, as:
  - Analog signals, to control the motor drives, etc.
  - Digital signals, to control relays, grippers, etc.
  - Miscellaneous signals, such as pulse generation, PWM, sine, etc.
- Card buses.
- Power supplies.
- Fans.
- Connectors, electrical terminals and cables.

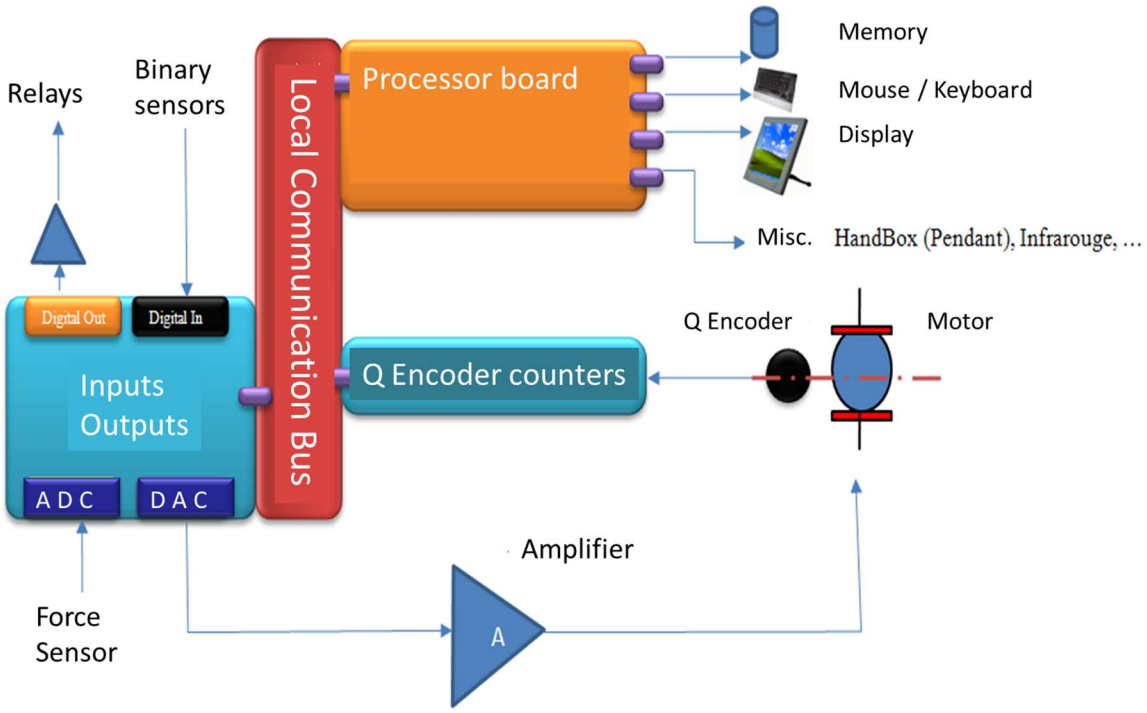


Figure 4.35-a, Robot Controller: Architecture overview

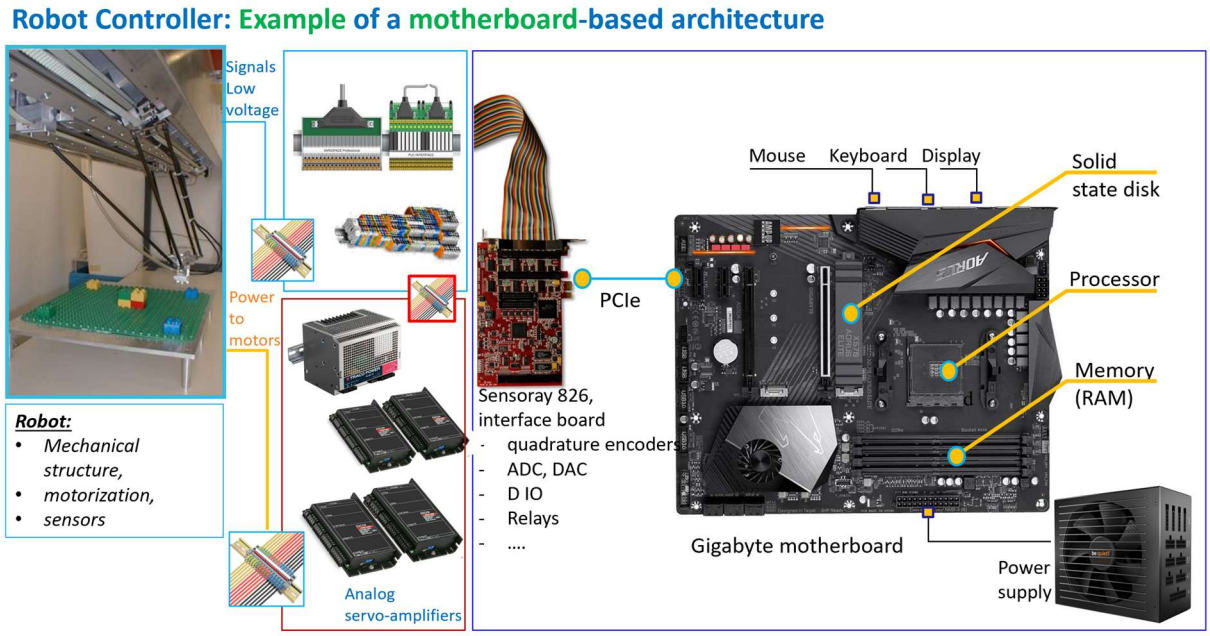


Figure 4.35-b, Robot Controller: Example of a motherboard-based architecture

Figure 4.36, Examples of industrial numerical controller. From Left to right, EPFL robot controller, Adept controller and ABB controller

5.2.1.1 The cabinet

The cabinet is the first hardware component of the controller. It is the casing in which all the other components of the control are fixed.

Figure 4.37, Examples of control cabinets

5.2.1.2 Acquisition boards,

which we also call input-output (IO) boards (Acquisition boards, Input-Output (IO) boards) are the hardware interface between the robot and the processor card. All types of interfaces may be required (analog, digital or other). The minimum structure necessary for controlling a robot requires the following components:

1. Analog outputs for controlling the motors via their analog drives (for torque control or for velocity control).
2. Incremental encoder inputs which consist of quadrature counters.
3. Digital inputs and outputs for controlling relays and reading digital sensors (presence/absence of parts, door closing, end, etc.)

Acquisition cards for controlling robots (often designated by **axis boards**) exist in all bus formats: PCI, Compact PCI and VME.

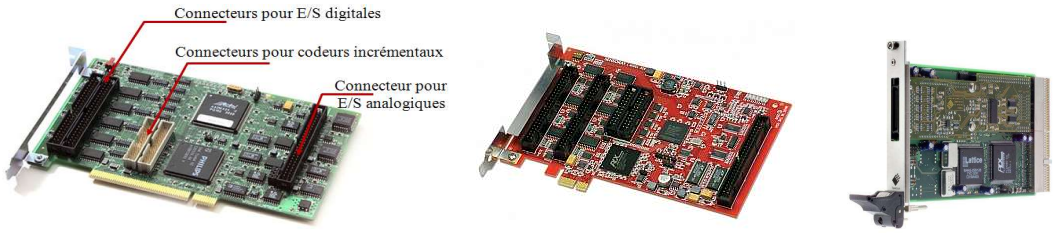


Figure 4.38, Examples of axis boards, from left to right, PCI (sensoray 626), PCIe (Sensoray 826) and CPCI NI

4.2.2 Motor and fieldbus drives

In the case of analog control, the motor drives are controlled by an analog setpoint which imposes the value of the voltage or the current to respectively fix the motor output velocity or torque (Figure 39).

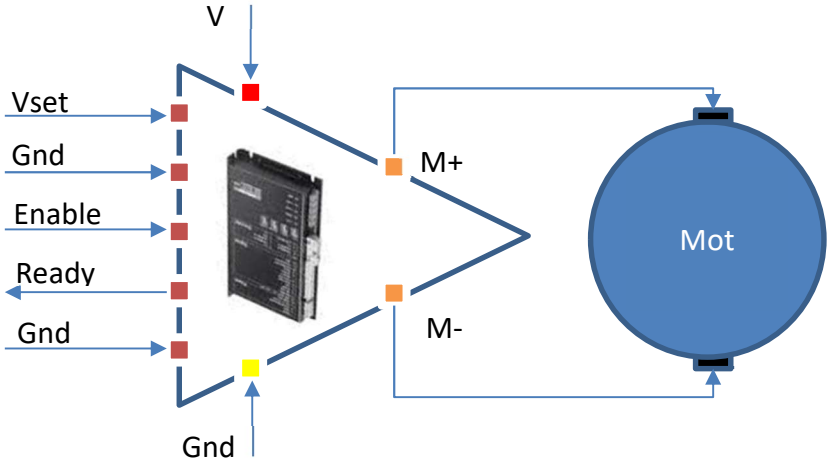


Figure 4.39, Example of wiring a Maxon servo amplifier (description in table below)

Signal	Type	Value	Description
Vset	Analog	-10V to 10V	Torque or velocity reference according to drive configuration
Gnd	Analog	0V	Ground
Enable	Discrete	5V	Amplifier activation (Enable)
Ready	On/Off	5V	Amplifier status (Ok or not)
M+, M-	Analog	--	Brushed motor terminals
Vcc	Analog	50V/5A	DC drive power supply

Table: Outline of the connections in the figure 4.39

Digital communications are appropriate to communicate between a computer board and a remote device, or between 2 computers. The most common digital communication interfaces may be enumerated as follows:

- Serial communication (as RS232, RS422 , and RS485)
- Ethernet
- USB
- Wireless ethernet
- Bluetooth

However, industrial communication needs industrial-compliant buses: **Fieldbuses**. **Fieldbuses** are communication buses for industrial automation. Each Fieldbus is defined by its physical layer, is associated with a bus protocol, and has a topology. The International Electrotechnical Commission (IEC) as IEC 61784/61158 describes the standardization of the fieldbuses.

Fieldbuses are increasingly used. They are a great alternative to analog technology and only use digital communication. Fieldbuses (fieldbuses) are used in the field of industrial automation mainly to improve the interconnection of components. They are all built around serial digital communication of data between a main automation unit and several input/output components. The connection configuration between components is called bus topology. The most frequent topology is that in "Daisy-Chain" with a master and several remote modules which interconnect in chain one after the other.

In this chapter, we will introduce 3 Fieldbuses: CANopen, Profibus, and EtherCAT. They all have a daisy chain topology.

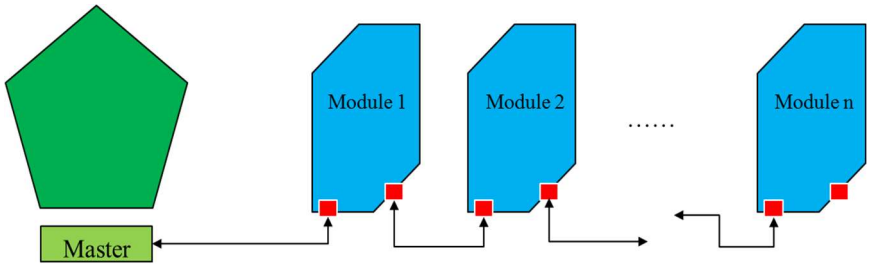


Figure 4.40, Principle of Daisy chain topology

The input-output components affect all the automation functions (analog reading, reading of digital inputs, relays, stepper motor control, motor drive, etc.) and allow not only deport the input-output function from the master component but also to distribute the functions according to the needs of the automation of the robot and its components.

The following paragraphs give a brief overview of the main field buses used for motion control. These field buses all have a protocol dedicated to the control of electric motors which enables to control the position of the axes either in "point mode" or in "profile generation mode". In all cases, the position control loop is located in the motor drive.

- **Profile generation mode**: This mode makes the axis module totally autonomous in its profile generation. The positioning path between the home position and the target position is generated in the drive. It is a simple and ideal mode when the axes are independent or when there is no need for interpolation or synchronization.
- **Point-to-point mode**: In this mode the points of the path are updated from the master at each sampling period. This period often varies between 250us and 2ms depending on the type of bus and the nature of the application. Synchronization and interpolation of the axes is ensured by the control software used at the master level.

4.2.3 The CANOpen (1Mbit/s)

The CAN open bus is a serial bus which reaches a transfer rate of 1Mbit/sec. It is an extension of the CAN bus. Its implementation on a master computer unit requires the use of a CANOpen master card. Compared to other field buses, CANopen is very slow and is not the best choice for operating modes requiring interpolations or synchronization of axes. Its topology is the Daisy chain. The example below shows an implementation with CANopen modules.

EPOS 24/5 and DC motor

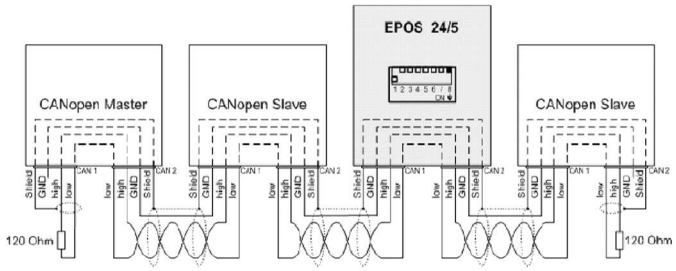


Figure 4.41, Example CAN Open implementation, using the ePOS module from Maxon

4.2.4 The PROFIBUS (9.6 kbit/s to 12 Mbit/s)

This serial bus allows transmission rates faster data transfer (up to 12 Mbit/sec). It was initiated and is promoted by **Siemens**. However, its specifications are open to other manufacturers in order to be able to interconnect components from various suppliers. Its topology is the Daisy chain. The example below shows an implementation with a 5-axis parallel robot. In this case, the position loop was implemented on the PC, which requires the use of a Profibus master card.

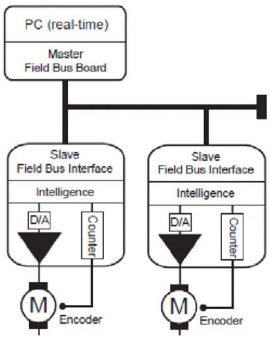


Figure 42, Using the Profibus bus – Example of an implementation with a 5-axis parallel robot

4.2.5 Ethercat

Ethercat is currently the most widely used Fieldbus for both automation and motion control. It was developed by the company Beckhoff and it allows transmission rates of up to[5] 100Mbits/sec. Its attractiveness is linked to two important elements:

- Ethercat uses the Ethernet component on standard computers and processor boards and therefore does not need an additional master card. *Only change the native Ethernet component driver to an Ethercat driver and it works*.
- The high transmission rate of the bus enables to have sampling refresh rates reaching 250µs.

Some numbers (2018);

- 256 digital I/O in 11 µs
- 200 analog I/O (16 bit) in 50 µs, 20 kHz Sampling Rate
- **100 Servo-Axis** (each 8 Byte IN+OUT) **in 100 µs = 0.1 ms**

[5] The new **EtherCAT G** moves the maximum data rate **up to 1 Gbit/s and 10 Gbit/s** (ref. Beckhoff)

Figure 4.43, Using the Ethercat bus – Example of an implementation with a Delta robot

4.2.6 Cascaded control using fieldbus motor drives

Position and velocity control loops can be setup either on the computer or on the motor drive. However, with the clearly improved capacities of the drives, the most interesting configuration is the following:

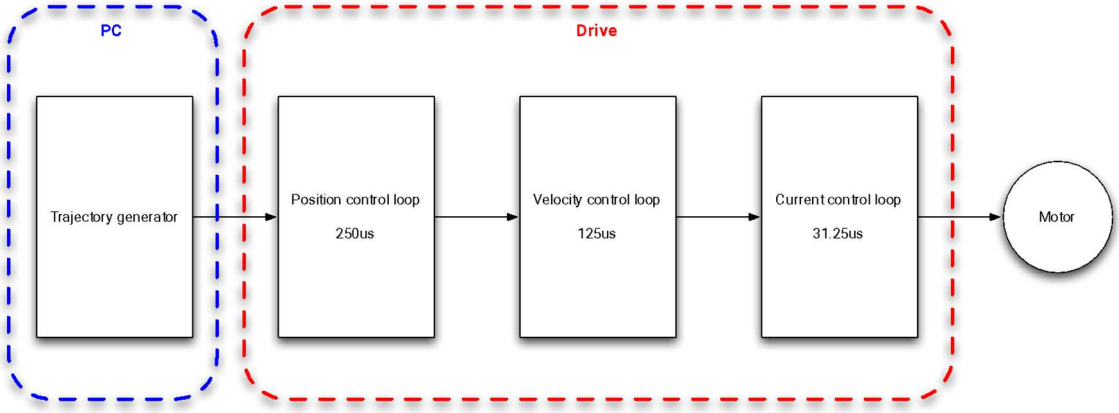


Figure 4.44, Example of distribution of servo loops between the drive and the central

Furthermore, if the refresh period of the position setpoints is not sufficient, many drives are able to interpolate the position setpoints between two consecutive refresh times. This allows to reduce position setpoint discontinuities; e.g. by replacing the setpoint steps with ramps.

4.2.7 Advantages of fieldbuses

- Reduction of wiring.
- Distribute input-output functions according to automation needs.
- Dissociate the control from the trajectory generation, thus dissociate the businesses and the solution providers.
- Increase the quantity of information to send to the motor drives. It is indeed possible to update the parameters of the control loops at each refresh period (fig 4.45), or even calculate the a priori dynamic model and send it through the bus, to consider a priori coupling compensations (fig 4.46).

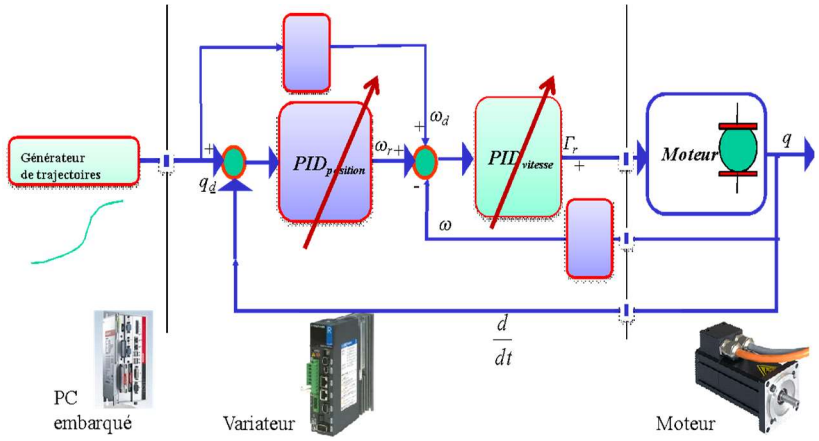


Figure 4.45, Fieldbus – Control loops with variable parameters according to the position of the robot

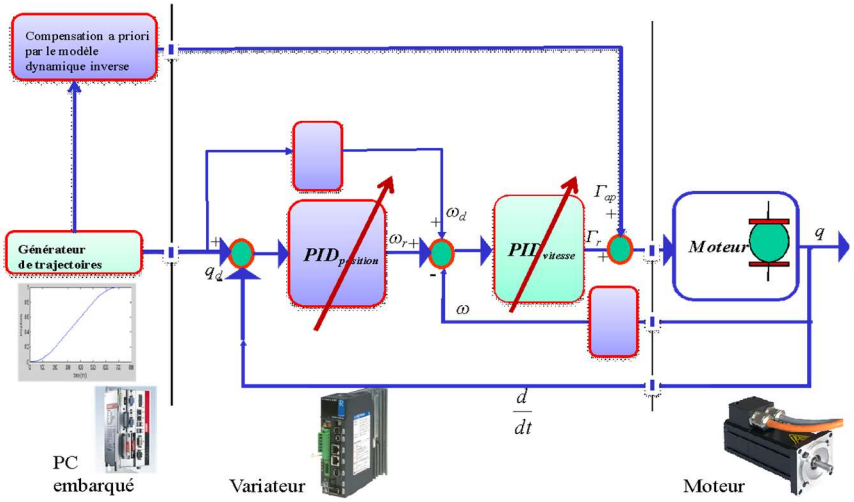
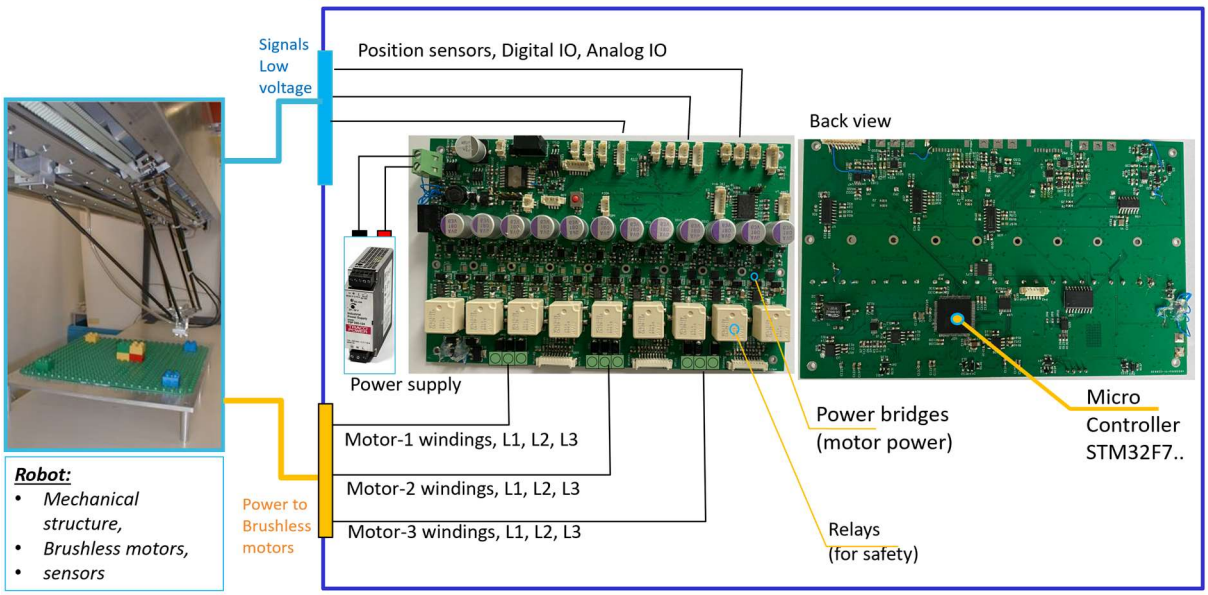


Figure 4.46, Fieldbus – Control loops with implementation of a prior torque

---

Robot Controller: Example of an all-in-one micro-controller robot controller

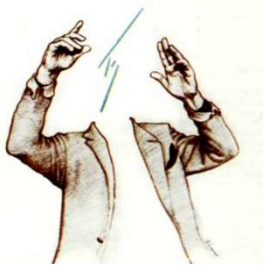


4.3 Software part of the control of a robot

In this part of the course, we consider the software components necessary to build (develop) a robot control.

4.3.1 The real-time operating system

5.3.1.1 Operating system



An operating system is:

- **An orchestra conductor** who manages all the hardware and software resources of the system in question (processor board and associated boards).
- **A software application** running under a given family of hardware (processors).

The resources to be managed are:

- Physical memory allocation.
- Allocation of processor availability.
- Interrupt management and start of associated routines.
- Management of task execution priorities
- Management of communication events between tasks

The control application **will work thanks to this conductor** who will manage all the tasks that implements our control software.

5.3.1.2 Real Time: Was Ist Das?

At first, to control a robot we **must master time**. To formalize this mastery of time, we say that the operating system must be real time (or have the characteristic of real time). This mastery of time is also called **determinism**.

A real-time operating system is specified as follows:

1. supports multitasking
2. supports synchronization events,
3. **is deterministic** in the execution of tasks,
4. **is deterministic** in the consideration of interrupts,
5. **is deterministic i**n handling events.

When we master time with regard to its orders of magnitude (minutes, seconds, ms, s...), we will say that we operate in real time.

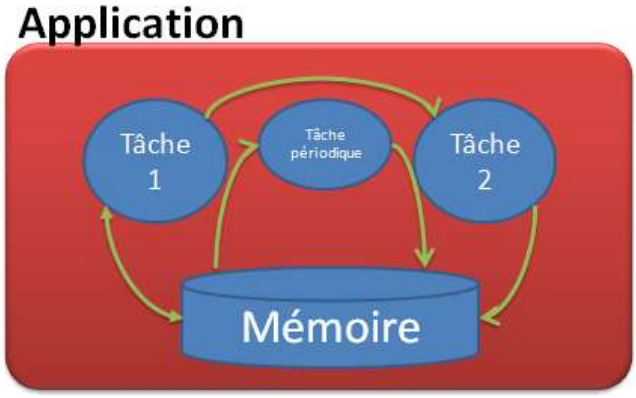


Figure 4.47, Real-time application

Real time is a software specificity of applications when the reaction to the occurring events is realized in a deterministic way. By event we mean:

- an interruption (software or hardware),
- the periodic launching of an operation,
- the access to a memory for reading or writing.

Note: Windows operating systems **(8, 10, and 11) and** Linux operating systems **(Ubuntu, Red Hat,..)** are not real-time operating systems although they are able to ensure appropriate operations about the ms when they are not over loaded. Some use them to develop non-critical real-time applications (ie. which do not obey the above requirements!).

4.3.2 Tools for developing real-time embedded applications

Every real-time OS (Operating System) comes with its own development system.

Development keywords:

- The **host** is the machine on which the developments are carried out
- The **target** is the machine on which the robot control application runs,
- **Cross development**, means to develop on a central unit called **Host** (of type PC for example) for another machine called **target** which contains the control software,
- **Cross compilation**, means to compile an application on a processor for another processor (see also another family of processors)
- **Upload** the application onto the target after cross-compiling it.
- **The Shell** is a remote-control command interface that runs on the Host,
- **The terminal**, is an application to view the exchanged information between the host and the target.

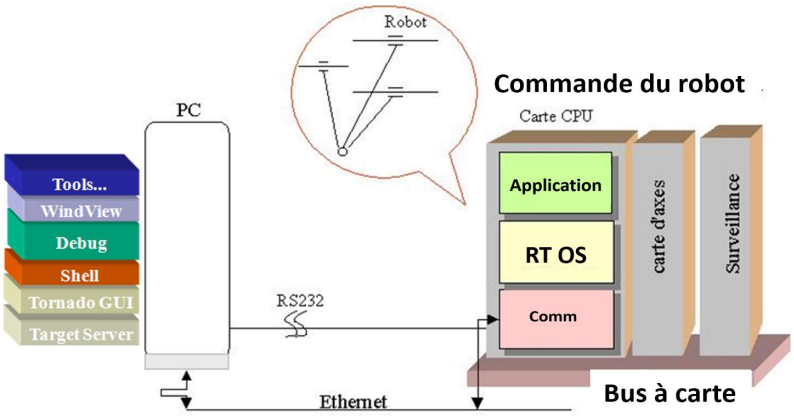


Figure 4.48, Example of a robot control application with its cross-development tools

4.3.3 Real-time communication mechanisms

A real-time OS is a multitasking OS. A real-time application is built around several **tasks** or **processes**. The processes use communication mechanisms (IPC: Inter Process Communication mechanisms) to synchronize and exchange data.

These mechanisms are:

- ***Binary semaphores***: objects that can be exchanged and are consumed or provided.
- ***Counter semaphores***: are initialized to a given number and are consumed or supplied until the stock is exhausted.
- ***Mutexes*** are semaphores that are either available or in use by a resource.
- ***Shared memories*** are memory zones.
- ***Messaging*** to exchange character strings.

4.3.4 List of real-time operating systems

Real time operating systems (RTOS)

- **VxWorks** from WindRiver
- **QNX OS** by QNX Software,
- **LynuxWorks** by LynxOS Inc. for embedded and real-time systems,
- **MS Windows** .net by Microsoft,

Real-time extensions for non-real-time operating systems

- **RTX** from IntervalZero Inc. is a real-time extension for systems running Microsoft OS,
- **RTLinux** real-time extension for Linux,
- Others for Linux (such as **RtaiLib**),
- Others for MS Windows such **Hyperkernel**, **InTime**,….
- **ROS2, includes REAL TIME CAPABILITIES, while the first version** ROS is a "Robot Operating" framework **– ROS2 extends to Linux**
  Kay, J., & Tsouroukdissian, A. R. (2015). **Real-time control in ROS and ROS 2.0.** ROSCon15**.**

Microcontroller Operating Systems

- **FreeRTOS** for Atmel, Microchip, ARM7, ARM9, Cortex-M3, MSP430, MicroBlaze,
- PIC32, 32-bit, RTOS for Microchip Technology's Microcontrollers from Express Logic.
- Linux,
- ARMOS, BeRTOS, Apache MyNewt (ARM, Cortex-M3, ARM ARM7TDMI,)

A complete list can be found in:

- https://en.wikipedia.org/wiki/Comparison_of_real-time_operating_systems

4.3.5 The robot control application

The robot control software contains all the functional blocks previously studied, namely those represented by the figure below:

Figure 49, Components of the control software

High-level orders, concern all functions that manage the man-machine interface, high-level orders but also the management of any type of operator interface. High-level orders can concern sending of the path parameters, the configuration of the dynamics, the control parameters, the activation of the grippers, the verification of the status of a digital input, and the counting of 'high-level operations (such as a number of succeeded or failed object grasping).

Trajectory generation handles trajectory generation with parameters sent from the top-level layer. It is in this module that the management of the paths, the profiles of velocities and accelerations, the interpolation, and the synchronization of the axes are carried out. This module is often called **the interpolator**.

Coordinate transformation, this module implements the direct and inverse geometric models. It is possible to integrate this module into the previous trajectory generation module.

Motion Control, formed mainly by the periodic axis tuning loop.

Board drivers manage the acquisition boards and provide the function services to access to the board input and output registers (analog reading, digital reading, encoder reading, counter reading, activation and deactivation of digital outputs, control of analog outputs, etc.)

Safety concerns the management of the security of the numerical control and of the robot. It can be implemented in the form of a periodic tasks which test the validity of position and velocity errors, the validity of position and velocity desired trajectories, test of sensor redundancy, test of certain differential limits on joint positions, etc.

Note: These axis software modules can be implemented either as a task (equivalent to a thread under Windows), a dynamic or static library, or either as a class.

4.3.6 The minimal controller:

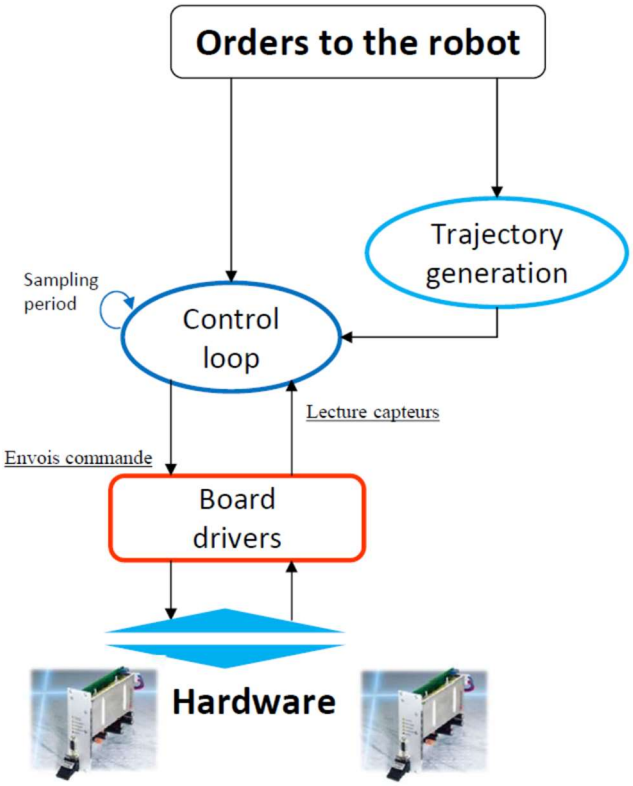


Figure 4.50, Minimal structure of a robot command

The **high level orders may be** implemented as a library of **supervision** to provide simple operator functions as **GoToPoint**(X,Y,Z), **ActivateControl**(), DeActivateControl(), GripperOn(), GripperOff().

The **Control** loop is the most important, it reads sensors, calculates motor torques and sends commands to the amplifiers, etc.

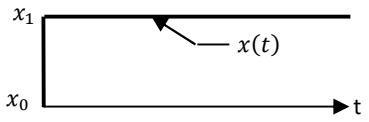
The trajectory generator generates the profiles and performs the axis interpolation.

4.4 Appendix 1: Calculation of trajectory profiles

Consider a linear motor (M). The problem statement of this topic concerns the following question: "***How to generate a position instruction to move from a position x_0 at a position x_1?***". Different time profiles of desired positions, velocities, and accelerations are presented in the next sub-sections. Δx is the path length considered in this paragraph. We can replace it with Δs; the curvilinear path distance if multiple degrees of freedom are to be considered.

4.4.1 Step-like position profile

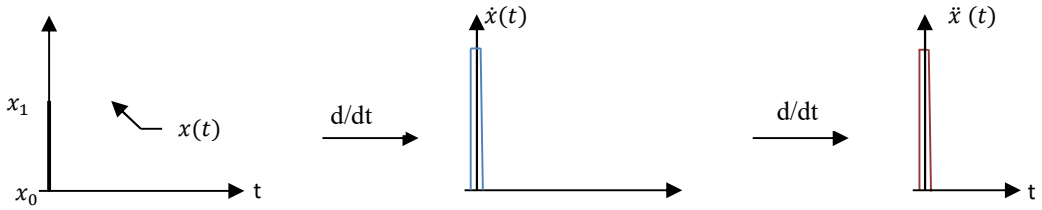
In automatic control theory, we often start by learning that the easiest way to set a desired position is to use a step-like desired position.



x(t) is then defined as follows:

$$x(t) = \{x_0 \quad t \leq 0 \; x_1 \quad t > 0$$

Let us observe the shape of the velocity and the acceleration corresponding to this movement from x_0 to x_1.



The velocity and acceleration associated with this desired position are Diracs and will thus generate a brutal torque which is not good for the mechanics (life cycle of the transmission elements). This jump in acceleration also risks exciting undesired natural frequencies (because not taken into account by the control) and thus causing mechanical resonances. This sudden torque jump, if it is not taken care of by a judicious choice of the control parameters, risks causing overshoots of the controlled position.

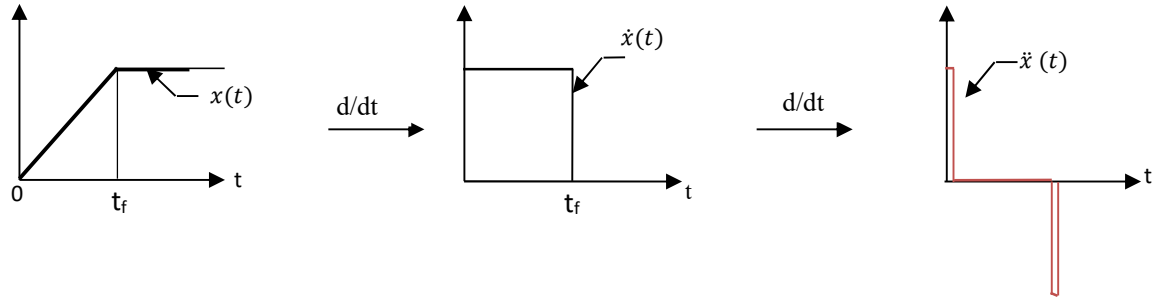
- **In the case of the control of actuators in general and particularly for the control of robots, the step-like trajectories must never be used.**
- In practice, the velocity and acceleration (Dirac peak) are not infinite because the derivatives of the desired position, respectively that of the velocity, are carried out at the rate of the control sampling period Ts.

  The maximum velocity and acceleration values are then given by:

$$V_{max} = \frac{\Delta x}{Ts} \text{ and } Acc_{max} = \frac{V_{max}}{Ts} = \frac{\Delta x}{T_s^2}$$

4.4.2 Ramp-like position profile (Ramp)

Another simple reference trajectory implements the desired position as a ramp-like profile. This trajectory is carried out at constant velocity all along the path.



In this case, we set the end time of the desired path, the velocity is thus limited (constant). The acceleration jump remains nevertheless always brutal but less than in the case of a step. This profile is not appropriate for high velocities.

<u>Calculation of the trajectory</u>: We assume the following for the calculation of the trajectory:

- t_f (time to complete the path) or V_{max} (constant velocity along the path).
- The distance to travel $\Delta x = x_1 - x_0$;

Whether V_{max} is imposed then $t_f = \frac{\Delta x}{V_{max}}$

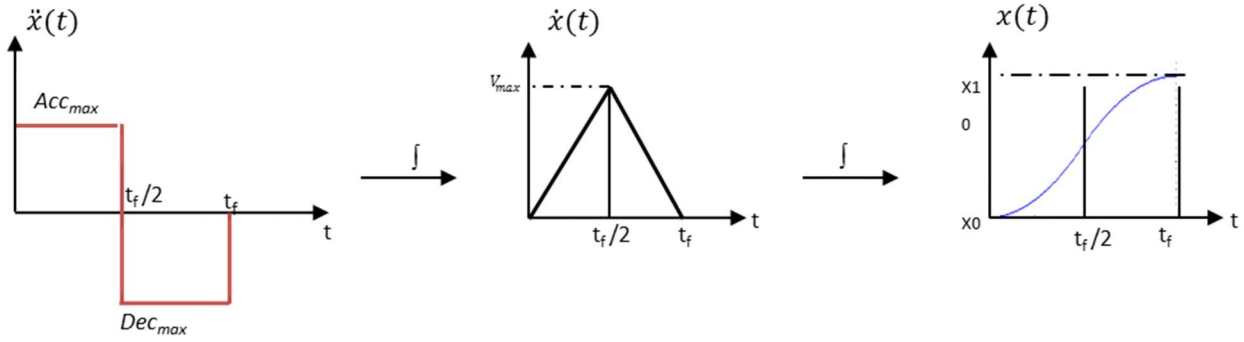
or t_f is imposed then $V_{max} = \frac{\Delta x}{t_f}$

Expression of the trajectory

$$x = \begin{cases} x_0 & if\ t \leq 0 \\ x_0 + V_{max} * t & 0 < t \leq t_f \\ x_1 & t > t_f \end{cases}$$

4.4.3 Triangular Velocity Profile

To avoid acceleration and deceleration diracs, one solution would be to limit the acceleration to a constant Acc_{max} for half of the distant and decelerate with a constant deceleration Dec_{max} on the other half of the distance. The velocity profile is thus triangular and the position profile is parabolic.



Profile calculation assumptions:

- Acceleration Acc_{max}
- Deceleration identical to the acceleration
- The length of the path is Δx

Unknown: t_f

Δx is the area of the triangle describing the velocity profile.

$$\Delta x = V_{max} * \frac{t_f}{2}$$

$$and\ V_{max} = Acc_{max} * \frac{t_f}{2}$$

$$\Rightarrow \Delta x = Acc_{max}\ \frac{t_f^2}{4}$$

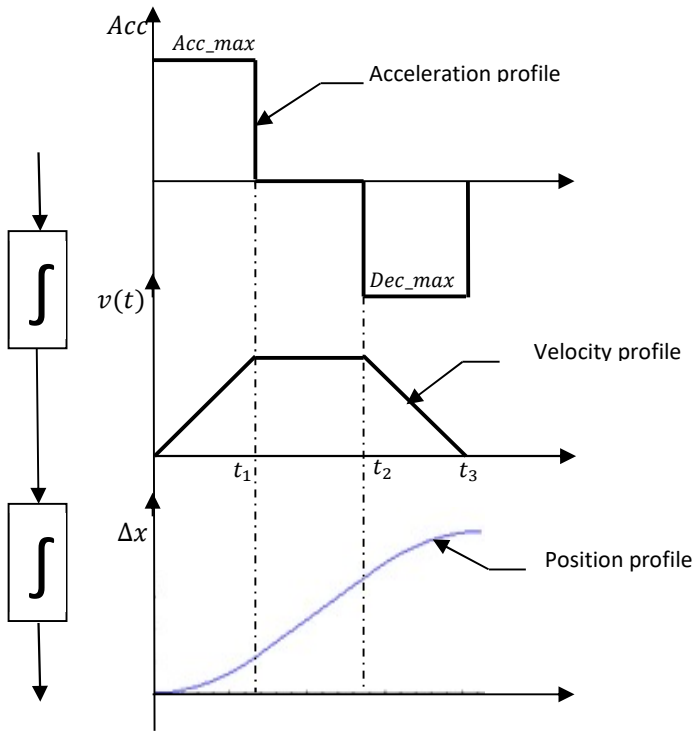
$$t_f = 2\sqrt{\frac{\Delta x}{Acc_{max}}}$$

Trajectory generation:

$$x(t) = \begin{cases} x_0 & t \leq 0 \\ x_0 + Acc_{max}\frac{t^2}{2} & 0 < t < \frac{t_f}{2} \\ \frac{x_1 - x_0}{2} + V_{max}\left(t - \frac{t_f}{2}\right) - \frac{Dec_{max}}{2}(t - \frac{t_f}{2})^2 & \frac{t_f}{2} \leq t < t_f \\ x_1 & t \geq t_f \end{cases}$$

This profile enables to complete the traveled distance in a minimum of time while having a constant acceleration and deceleration. However, the disadvantage is that the maximal velocity can be very high in case the traveled distance is high. It is is then recommended to limit the velocity, as in the trapezoidal velocity profile presented in the next sub-section.

4.4.4 Trapezoidal velocity profile



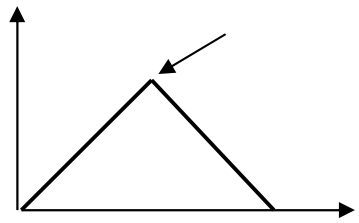
With this profile, we set the maximum velocity to (V_{max}) and we limit the acceleration to (Acc_max), equal to the maximum deceleration Dec_max[6].

Profile calculation assumptions:

- Acceleration Acc_{max}.
- Deceleration identical to acceleration.
- Max velocity V_{max}.
- The length of the path Δx

[6]The acceleration is assumed equal to the deceleration to simplify the problem. In the general case the maximum accelerations Acc_{max} and Dec_{max} are not necessarily identical.

The minimum path ΔX_{min} corresponds to the area obtained for a triangular velocity profile whose peak velocity is *Vmax*. t_{fmin} is the final time corresponding to the completion of this journey ΔX_{min}.



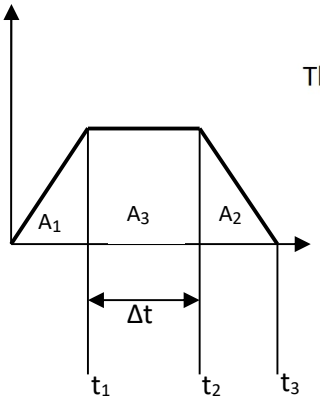
$$\Delta X_{min} = \frac{V_{max}^2}{Acc_{max}} \quad and\ t_1 = \frac{t_{fmin}}{2}$$

$$t_{fmin} = 2\sqrt{\frac{\Delta X_{min}}{Acc_{max}}}$$

1st case:

If $\Delta X \geq \Delta X_{min}$, V_{max} is then reached. It is therefore necessary to add a phase at constant velocity and whose duration is:

$$\Delta t = \frac{\Delta x - \Delta X_{min}}{V_{max}}$$



The areas A_1 , A_2 and A_3 are given as follows:

$$\{A_1 + A_2 = \Delta s_{min}\ A_3 = \Delta s - \Delta s_{min}$$

$$\Longrightarrow \begin{cases} t_1 = \dfrac{tf_{min}}{2} = \sqrt{\dfrac{\Delta X_{min}}{Acc_{max}}} \\ t_2 = t_1 + \Delta t \\ t_3 = t_2 + \dfrac{tf_{min}}{2} = 2\sqrt{\dfrac{\Delta X_{min}}{Acc_{max}}} + \dfrac{\Delta x - \Delta X_{min}}{Vmax} \end{cases}$$

2nd case:

If $\Delta X < \Delta X_{min}$ then $Vmax$ will not be reached and the profile will be purely triangular. The maximum velocity must be adapted to achieve the route to be covered.

$$V_{max\_new} = \sqrt{\Delta x * Acc\_max}$$

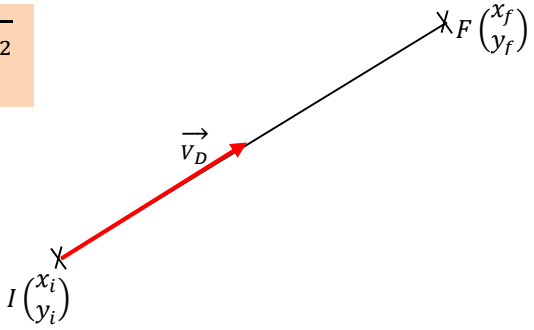
$$t_1 = t_2 = \frac{t_3}{2} = \frac{t_f}{2} \qquad \text{and} \quad t_f = 2\sqrt{\frac{\Delta x}{Acc\_max}}$$

4.5 Appendix 2: Interpolation of trajectories

4.5.1 Linear path interpolation

The total distance to cover is:

$$\Delta s = \sqrt{\Delta x^2 + \Delta y^2} = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$$



v_D is the vector director throughout the pathway :

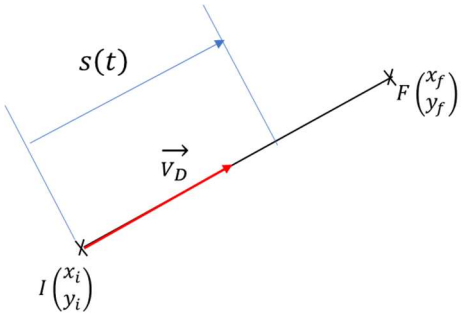
$$v_D = \begin{cases}\dfrac{x_f - x_i}{\Delta s}\\ \dfrac{y_f - y_i}{\Delta s}\end{cases} = \begin{cases}v_{Dx}\\ v_{Dy}\end{cases}$$

How to generate the straight-line path from point I to point F?

This is called linear interpolation and involves calculating the points of the path in question as a function of time. It is performed in two steps:

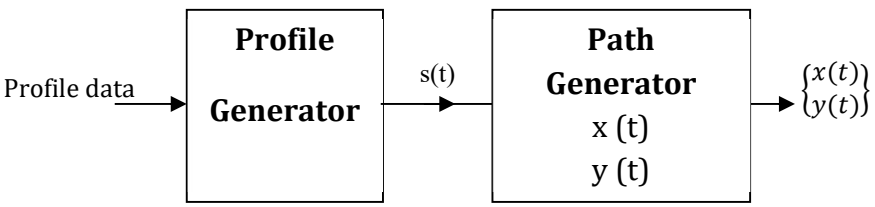
- The first operation is to generate the position profile s(t) of length Δs (ref. profile calculation document).
- The profile s(t) must then be projected onto the shape of the trajectory to obtain the corresponding x(t) and y(t) coordinates. In this way we obtain the equations of the points of the trajectory:

$$x(t) = x_i + v_{Dx} \,.\, s(t)$$
$$y(t) = y_i + v_{Dy} \,.\, s(t)$$



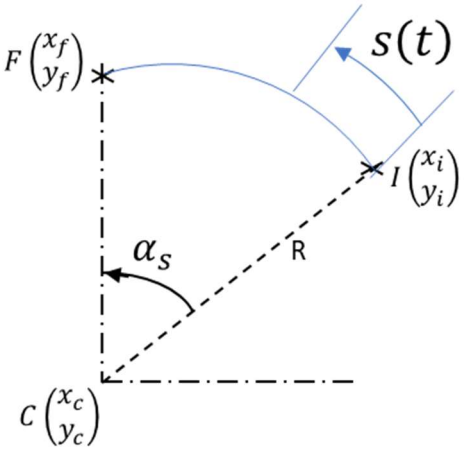
With :

$$x(t=0) = x_i \quad , \quad x(t=t_f) = x_f$$
$$y(t=0) = y_i \quad , \quad y(t=t_f) = y_f$$



4.5.2 Circular interpolation:

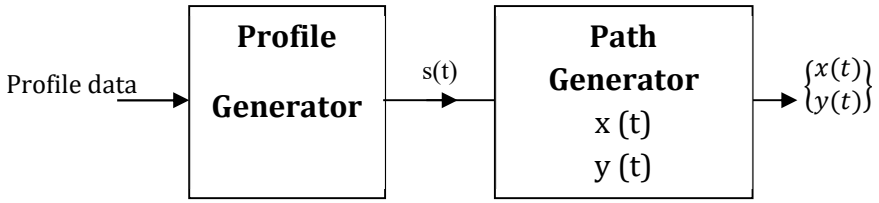
In this case, it is a question of making a circular trajectory (arc of a circle) from the point I to point F. C is the centre of the circle considered, and R is the radius of this circle.
Let α be the angle to cover varying from 0 to α_s.
The first step consists in generating the profile of the angle α(t) thanks to an appropriate choice (trapezoidal in velocity or trapezoidal in acceleration).
The second operation is to derive the coordinate (x(t), y(t)) along the circular path.



The equations of the coordinates x and y are expressed as follows:

$$\boldsymbol{x(t) = x_c + R\sin(\alpha(t)) = x_c + R\sin(s(t)/R)}$$
$$\boldsymbol{y(t) = y_c + R\cos(\alpha(t)) = y_c + R\cos(s(t)/R)}$$

$$x(t=0) = x_i \;\; ; \;\; x(t=t_f) = x_f$$
$$y(t=0) = y_i \;\; ; \;\; y(t=t_f) = y_f$$



4.6 Appendix 3- Additional hardware aspects

4.6.1 Board buses

A bus refers to transport; we are not far from it except that in our case we will speak about transport of data. A hardware bus connects several electronic boards to the microprocessor unit. This bus allows the cards to communicate with each other thanks to the data, address and control signals that constitude the bus. We will call the backplane the electronic board on which all the cards of the bus are inserted. This backplane is passive because it does not include any intelligence (i.e. a processor).

There are many different types of bus. In numerical robot control, one of the first used buses was the VME bus supported mainly by the company Motorola. Other industrial buses are proprietary (i.e. they are supported only by their suppliers). Many have disappeared (such as Gespac from the company Gespac, the MCA bus from IBM ("micro channel architecture", ...). The VME bus has long monopolized the industrial market because of its robustness, durability and simplicity. A consortium led by Intel launched the Compact PCI bus, which mainly uses PCI signals (For more details, refer to http://www.picmg.org/v2internal/specifications.htm)

Form factors and board buses

There are mainly the two following form factors:

1- Blade format cards
The form factor of these boards is like blades and can be inserted into a dedicated cabinet (see figure opposite), which is sized for.
Examples: VME, Compact PCI (CPCI), Micro TCA

Figure 4.51, Examples of blade-like form factors

2- Slot form factors. The typical example of this type of baords is the PCI and PCI express board formats. PCI exists in standard format with motherboards and also with a passive backplane. The motherboard is a processor board that integrates both the processor and a local bus to receive daughter boards. For reasons of durability, industrial hardened motherboard variants exist for demanding applications (automation and control).

Figure 4.52, From Left to right, Industrial motherboard, passive PCI bus, PCI processor board (not motherboard)

4.6.2 Les alimentations :

There are mainly two types of power supplies:

- Bus-dedicated power supplies that have very particular form factors and specific voltage outputs, at at a specific current, according to the buses they are intended to be used for. These power supplies power the electronic acquisition boards and processors.
- Conventional power stages power supplies for power electronic components (motor amplifiers, relays, graspers, heathers, etc.

Figure 53, Examples of power supplies from left to right Euro format PSU for CPCI, ATX motherboard PSU, .common screwable PSU, DIN RAIl PSU