POLYPUBLIE
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Contrôle de systèmes robotiques
Title:

Auteurs: Romano M. De Santis, Marc Quintal, & Caroline Serfass
Authors:

Date: 1989

Type: Rapport / Report

Référence: De Santis, R. M., Quintal, M., & Serfass, C. (1989). Contrôle de systèmes robotiques. (Rapport technique n° EP-R-88-26). https://publications.polymtl.ca/9597/
Citation:

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: https://publications.polymtl.ca/9597/
PolyPublie URL:

Version: Version officielle de l'éditeur / Published version

Conditions d'utilisation: Tous droits réservés / All rights reserved
Terms of Use:

Document publié chez l'éditeur officiel
Document issued by the official publisher

Institution: École Polytechnique de Montréal

Numéro de rapport: EP-R-88-26
Report number:

URL officiel:
Official URL:

Mention légale:
Legal notice:

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

28 FEV. 1989

Cours 3.678

CONTRÔLE DE SYSTÈMES ROBOTIQUES

L34508

Romano M. DeSantis
avec la collaboration de
Marc Quintal
Caroline Serfass

Département de génie électrique

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Janvier 1989

CA2P9
UP 201
1989
C57

consul

Contrôle de systèmes robotiques (1989)

R. DeSantis (département de génie électrique)
(avec la collaboration de M. Quintal et C. Serfass)

1589

1 2 3 4 5 EP 93 92 91 90 89

EPM/RT-88/26

ON THE CONTROL OF A ROBOTIC MANIPULATOR

By

Romano M. DeSantis, Professor

Département de Génie Electrique

Ecole Polytechnique de Montréal

Septembre 1988

PREFACE

The following notes are a partial documentation in support of the eight week course on "Identificazione e Controllo dei Manipolatori Robotici" offered at the Politecnico di Milano, Dipartimento di Elettronica, during the spring 1988. This course is itself a part of "Teoria della Regolazione" a full year course under the responsibility of Professor A. Locatelli. A debt of gratitude on the part of the author is due to Professor A. Locatelli for the opportunity to collaborate with him in this project, and to professor S. Bittanti for making my coming to Milano possible. Many thanks are also due to them for technical interactions, fine hospitality and warm friendship; these thanks must be extended to the other members of the group of Automatica, among them: professors G. Guardabassi, N. Schiavoni, C. Maffezzoni, P. Bolzern, P. Colaneri, and R. Scattolini; and to the various students with whom I have had the pleasure to interact.

Our objective is to introduce the main ideas and techniques associated with the control of a robotic manipulator. The interdisciplinary nature of the subject, the width and depth of its territory, its partly well established and partly rapidly evolving character make this not an easy thing to do. We will therefore opt for a pragmatic approach: to simply aim at opening a window, creating a point of departure, providing the motivation and the capability for

the reader to proceed by himself as far down the road as he may wish to go.

We will assume familiarity with a certain number of items such as basic kinematics, dynamics and automatic control theory. Our presentation strategy will be to recall the basic idea behind a technique and to illustrate its application to the case in point; care will be taken to pinpoint connections with the specialized literature; a special attention will be given to provide the student with the background required to benefit from a number of proposed hands-on simulation experiments.

These were the intentions... As it turns out, we do realize that what we end up with is only the beginning of the document we would have liked to get. We are nevertheless confident that the serious student complementing what we have with the references listed in the bibliography (particularly recommended are [Wr.1] and [Fu.1], plus the educational reports [RMDS. 3-5]), should have no difficulty into attaining the intended objective.

ON THE CONTROL OF A ROBOTIC MANIPULATOR

PREFACE ii

1. INTRODUCTION 1

1.1 Introduction

1.2 Dynamic model of a robotic manipulator

1.3 The control problem

2. MODEL BASED CONTROLLERS 13

2.1 Introduction

2.2 Computed torque controller

2.3 Acceleration resolved motion

3. PID CONTROLLERS 19

3.1 Introduction

3.2 Position servo and trajectory tracking PID controllers

3.3 Observations

4. ADAPTIVE CONTROLLERS 26

4.1 Introduction

4.2 Model reference adaptive controllers

4.3 Self-tuning adaptive controllers

4.4 Multivariable nonlinear adaptive controllers

5. SLIDING MODE CONTROLLERS 39

5.1 Introduction

5.2 Some selected results

5.3 PID/Sliding mode controllers

5.4 Observations

5.5 Application to real time parameter identification

6. LEARNING CONTROL 54

6.1 Introduction

6.2 Implementation in a computed torque scheme

BIBLIOGRAPHY 58

EPM TECHNICAL REPORTS 59

APPENDICES

A. Laboratory Simulation Experiments 60

*1: Application of a classical PID controller to a robotic link

*2: Application of model reference and self-tuning adaptive controllers to a robotic link

*3: Sliding mode control of a robotic link

B. Laboratory simulations 63

POSSPEED
SIMNOG
MRAC
SELFTUNE
ROBEL

LIST OF FIGURES

1.1.1 A robotic manipulator

.2 Physical components of a robotic system

.3 Conceptual structure of a robotic system

.2.1 Dynamic model of a link

.2 Simplified model of a link

.3.1 Structure of the overall controller

.2 Structure of a link feedback controller

.3 Components of a link feedback controller

2.2.1 Structure of a computed torque controller

.3.1 Structure of an acceleration resolved motion controller

3.2.1 PID Position servo controller

.2.2 PID Trajectory tracking controller

4.2.1 Structure of a model reference adaptive controller

.3.1 Structure of a self-tuning adaptive controller

.4.1 Multivariable nonlinear adaptive control of a robotic manipulator

5.3.1 PID/Sliding mode servo controller

.2 PID/Sliding mode trajectory tracking controller

6.1.1 Learning control scenario

.2.1 Learning control as applied to a computed torque controller

B.1.1 Bloc diagram of the system considered in POSSPEED

.2 The nonlinear element in POSSPEED

.3 Gain adjustment in POSSPEED

.2.1 The robotic system considered in SIMNOG

.2 The dynamical model in SIMNOG

.3.1 Controller/plant ensemble considered in MRAC

.2 The reference model in MRAC

.3 Gain adaptation module in MRAC

.4.1 Correspondence of notations in the estimator of SELFTUNE (a speed measurement is available)

.2 Correspondence of notations in the estimator of SELFTUNE (a speed measurement is not available)

.5.1 Elastic link considered in ROBEL

.2 Linear state regulator used in ROBEL

.3 Variable structure controller used in ROBEL

1. INTRODUCTION

1.1 Introduction

The physical components of the guidance unit of a robotic system, a robotic manipulator (fig. 1.1) complemented with whatever peripheral equipment may be required to perform a given task (fig. 1.2), are similar to those which are usually found in the supervision unit of an industrial control process: transducers, detectors, actuators, power generators, computers, communication, memory, visualization, interface, software, timing etc.

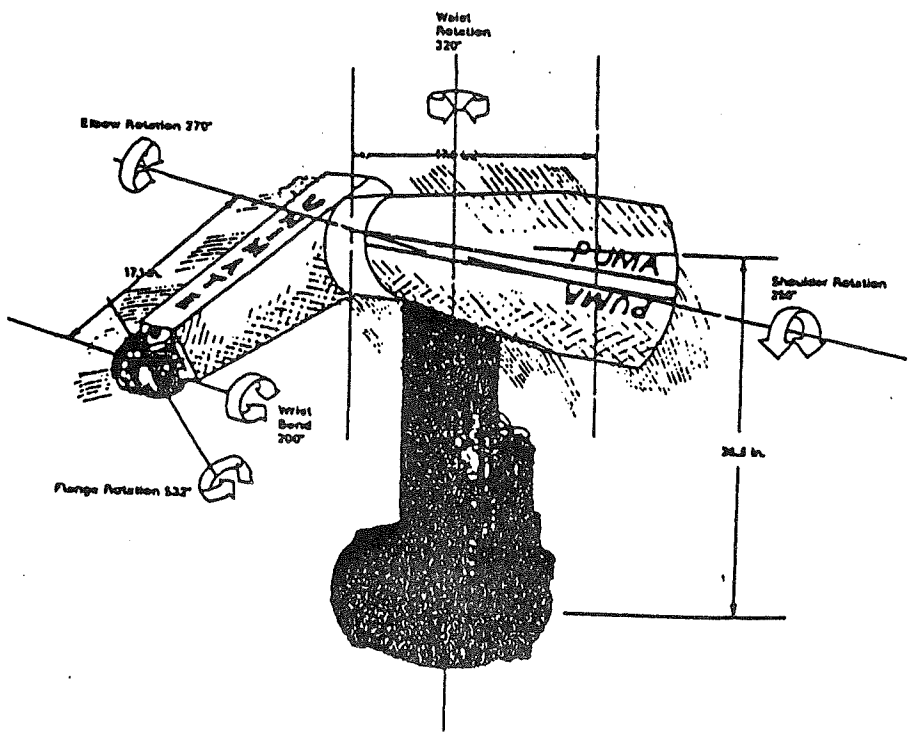


Figure 1.1: A robotic manipulator

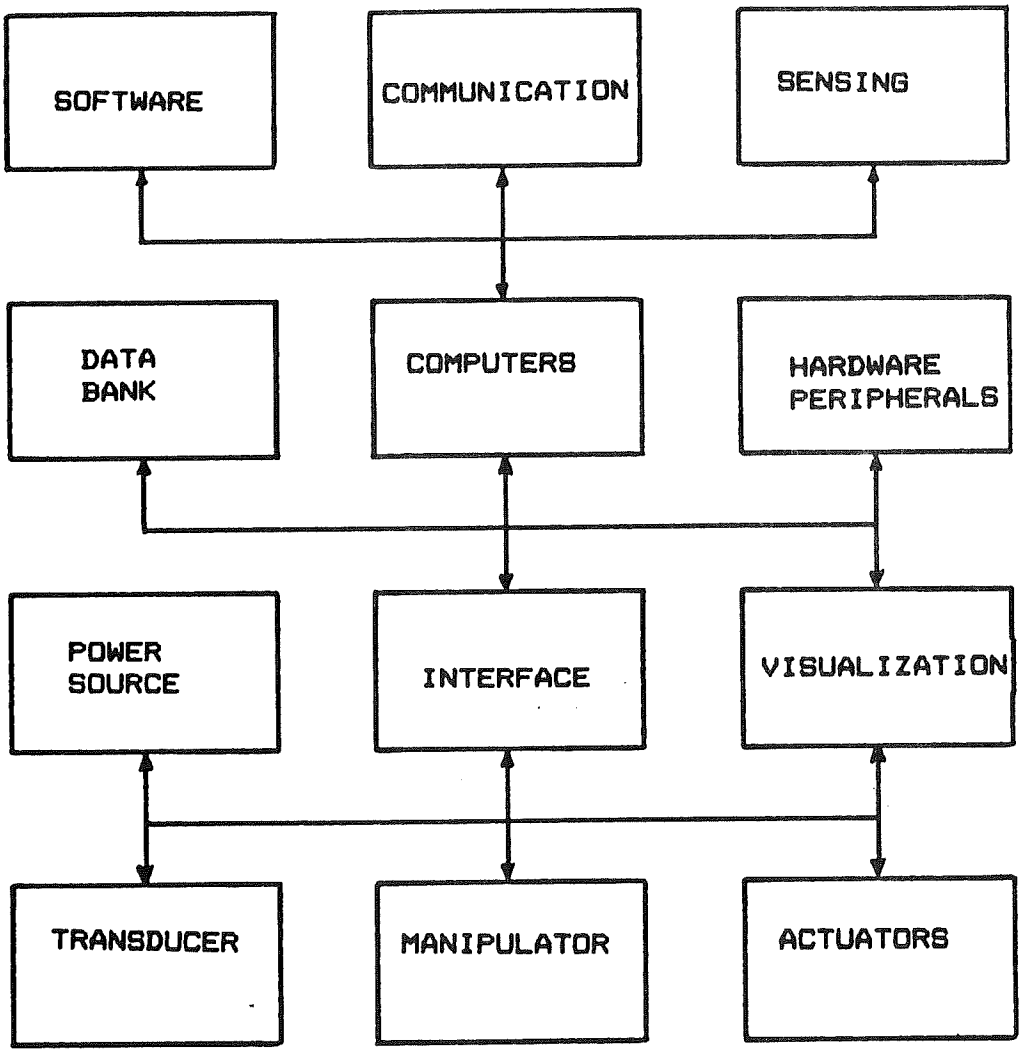


Figure 1.2: Physical components of a robotic system

With reference to fig. 1.3, the conceptual operation of such a system can be described as follows:

i. an operator communicates through a man-machine interface the task to be accomplished;

ii. an artificial intelligence module determines the action strategy needed to accomplish such a task; this strategy is usually a function of the task itself and of the information about the work scene which is provided by the sensing elements;

iii. a trajectory generator module computes the manipulator links position and velocity required to implement such an

action;

iv. a controller computes the inputs to the amplifiers driving the actuators so that position and velocity measured by the links transducer coincide as rapidly, precisely and reliably as possible with the required values;

v. this sequence is repeated until the task has been accomplished.

Our purpose in what follows is to focus attention on ideas and techniques at the basis of the design of the controller module. Preliminary requirements to the intended subject are: an exposure to the principles governing the kinematics and the dynamics of articulated chains [Cr.1] ; a general knowledge of the various families of robots available in the industry together with the variety of applications [En.1] ; a good familiarization with the working and the modalities of operation of at least one specific example of a robotic guidance system (a useful educational example can be found in [RMDS.1]); a good familiarization with standard control techniques.

The main questions of interest are: How are standard control techniques applied to robotic manipulators? With what success? With what modifications? What new techniques are considered? What is the interplay between kinematics, dynamics and control?

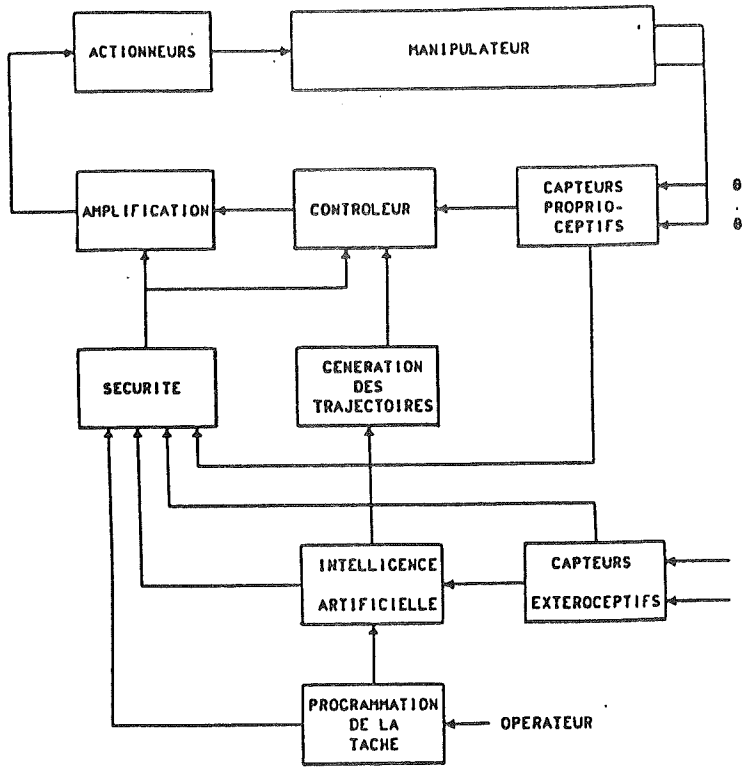


Figure 1.3: The conceptual structure of a robotic system.

1.2 Dynamic model of a robotic manipulator

The dynamic model of a robotic manipulator allows one to determine acceleration speed and position of the robot which are produced by the application of a given set of forces (direct dynamics); or, inversely, to determine the set of forces required to produce a desired acceleration speed and position trajectory (inverse dynamics). Such a model is usually obtained by using either the Newton-Euler approach, particularly efficient for computational purposes, or the Euler-Lagrange approach, particularly convenient for

an analytical closed form representation of the physical elements involved.

A systematic application of the Euler-Lagrange approach [Pa.1,chp 4] leads to the following model (fig.2.1):

$$\Gamma_i = \sum_{j=1}^{n} D_{ij}\,\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk}\,\dot{q}_j\dot{q}_k + D_i,\quad i=1\ldots n$$

where:

n : is the number of joints (degrees of freedom, links);

Γ_i: is the resultant of the forces, (moments), applied to link i;

$q_i, \dot{q}_i, \ddot{q}_i$: represent position, speed and acceleration of link i with respect to link i-1;

D_{ij}: represents the effective inertia of the manipulator

$$D_{ij} = \sum_{p=\max i,j}^{n} \mathrm{Trace}\,(U_{pj}\,J_p\,U'_{pj})$$

J_p: pseudo inertia matrix of the link;

U_{pj}: $= \dfrac{\partial}{\partial q_j}\,T\,[p,o]$;

$T[p,o]$: transformation matrix associated to the frames of link p and of the base of the manipulator;

D_{ijk} : centripetal and Coriolis force coefficients;

$$D_{ijk} = \sum_{p=\max i,j,k}^{n} \mathrm{Trace}\,(U_{pjk}\,J_p\,U'_{pi})$$

$$U_{pjk} = \frac{\partial}{\partial q_i} \left[\frac{\partial}{\partial q_k} \; T\left[p,o\right] \right]$$

The complexity of such a model may be illustrated by simply noticing that for n=6, matrices D_{ij} and D_{ijk} have dimensions respectively equal to 6x6=36 and 6x6x6=216; the number of multiplications required to compute the model is roughly equal to

$$16x6^4=86x\ \frac{5}{12}\ x\ 6^3+\ \frac{171}{4}\ x\ 6^2+\ \frac{53}{3}\ x\ 6-128=66.271;$$

the number of additions is

$$25x6^4+22x6^3+\ \frac{129}{2}\ x\ 6^2+14x6-96=51.548.$$

Using standard state and control notations

$$x_1 \triangleq \begin{bmatrix} q_i \\ \vdots \\ q_n \end{bmatrix} \quad ; \quad x_2 \triangleq \dot{x}_1 ; \qquad x \triangleq \begin{bmatrix} x_1 \\ \hline x_n \end{bmatrix} \quad ; \quad u \triangleq \begin{bmatrix} \Gamma_1 \\ \vdots \\ \Gamma_n \end{bmatrix}$$

one obtains a state model of the form

$$\dot{x} = f(x) + B(x)u + \beta$$

where

$$f(x) \triangleq \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ -\left[D_{ij}\right]^{-1} \left[D_{ijk}\right] \end{bmatrix} x_1 \bullet x_2 + \left[D_{ij}\right]^{-1} \left[D_i\right]$$

$$B(x) \triangleq \begin{bmatrix} 0 \\ [D_{ij}]^{-1} \end{bmatrix}$$

and β is a vector representing perturbation forces (friction, stiction, external perturbations, ...).

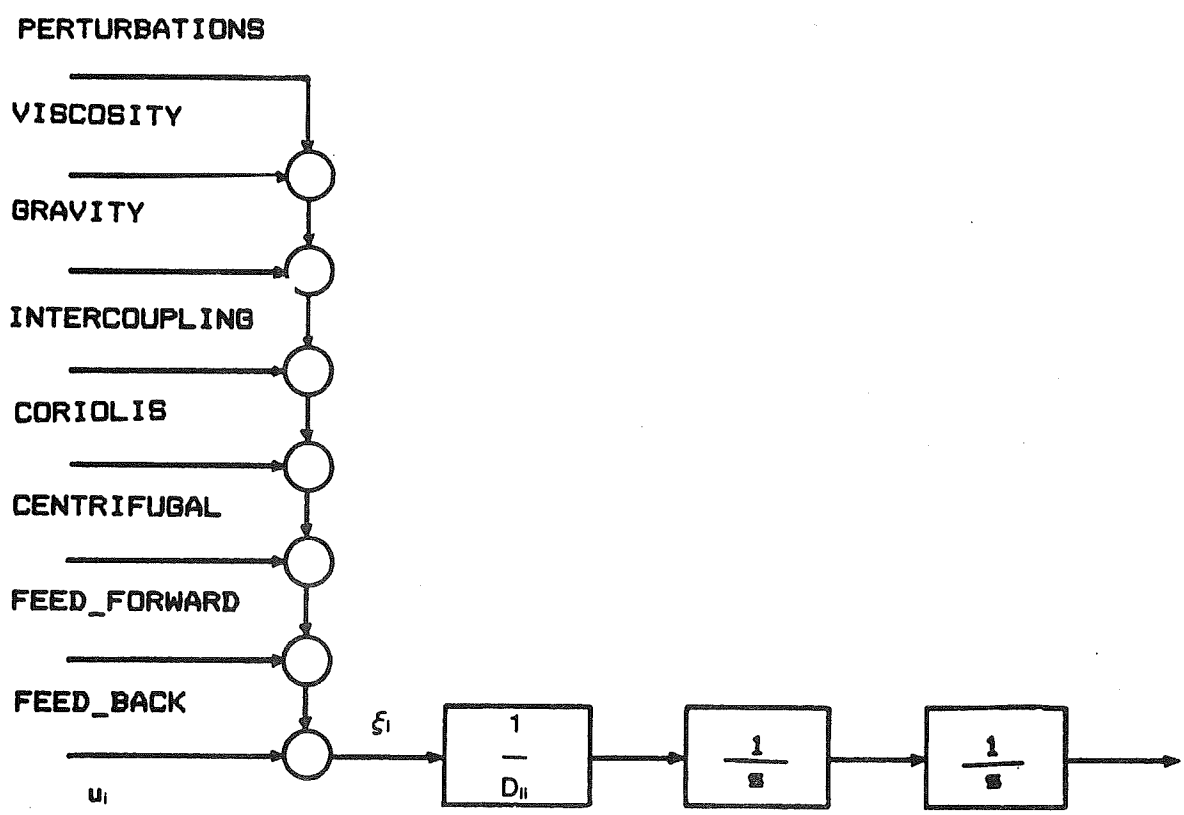


Figure 2.1: Dynamic model of a link.

Usually, the control force applied to a link is made of a centralized feedforward component, u_{FFi}, directed at neutralizing dynamic and perturbation forces (intercoupling, gravity, centrifugal, Coulomb, ...); plus a decentralized feedback component, u_{FBi}, directed at reducing the distance between actual and desired link position and velocity.

Indicating with x_{1i} and x_{2i} such position and velocity, one can then represent the dynamics of a link in terms of the following simplified model (fig. 2.2):

$$\dot{x}_{1i} = x_{2i}$$

$$\dot{x}_{2i} = -\frac{x_{2i}}{\tau_i} + \frac{K_{mi}}{\tau_i}\left[u_{FBi} + \xi_i\right]$$

where:

$K_{mi} \triangleq \frac{1}{ii} \triangleq$ gain of the link, $\alpha_{ii} \triangleq$ a viscosity coefficient;

$\tau_i \triangleq \frac{\alpha_{ii}}{D_{ii}} \triangleq$ time constant of the link;

$\xi_i \triangleq$ the resultant of u_{FFi} and the dynamic and perturbation forces it is supposed to neutralize.

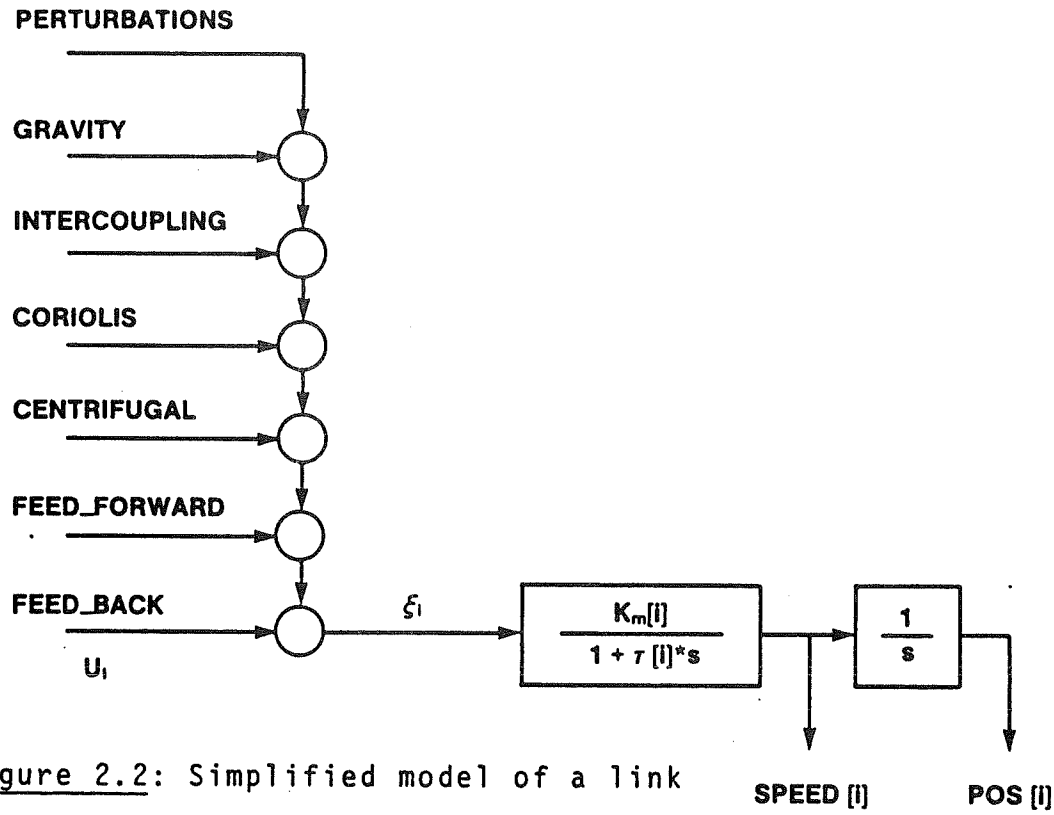


Figure 2.2: Simplified model of a link

1.3 The control problem

The structure of the controller of a robotic manipulator is represented in fig. 3.1: each link is servoed by means of a decentralized feedback loop controller; these controllers are complemented with a centralized coordinator which, on the basis of current and desired configurations of the manipulator, generates a convenient feedforward action; this coordinator may also tune the gains of the feedback loop controllers. The physical components of the feedback controller are described by fig. 3.2 and 3.3.

Representing the manipulator with the model

$$\dot{x} = f(x) + B(x)u + \beta$$

the controller design problem is: to develop a control $u(\cdot)$ capable of forcing $x(\cdot)$ to follow a desired $x_D(\cdot)$ as

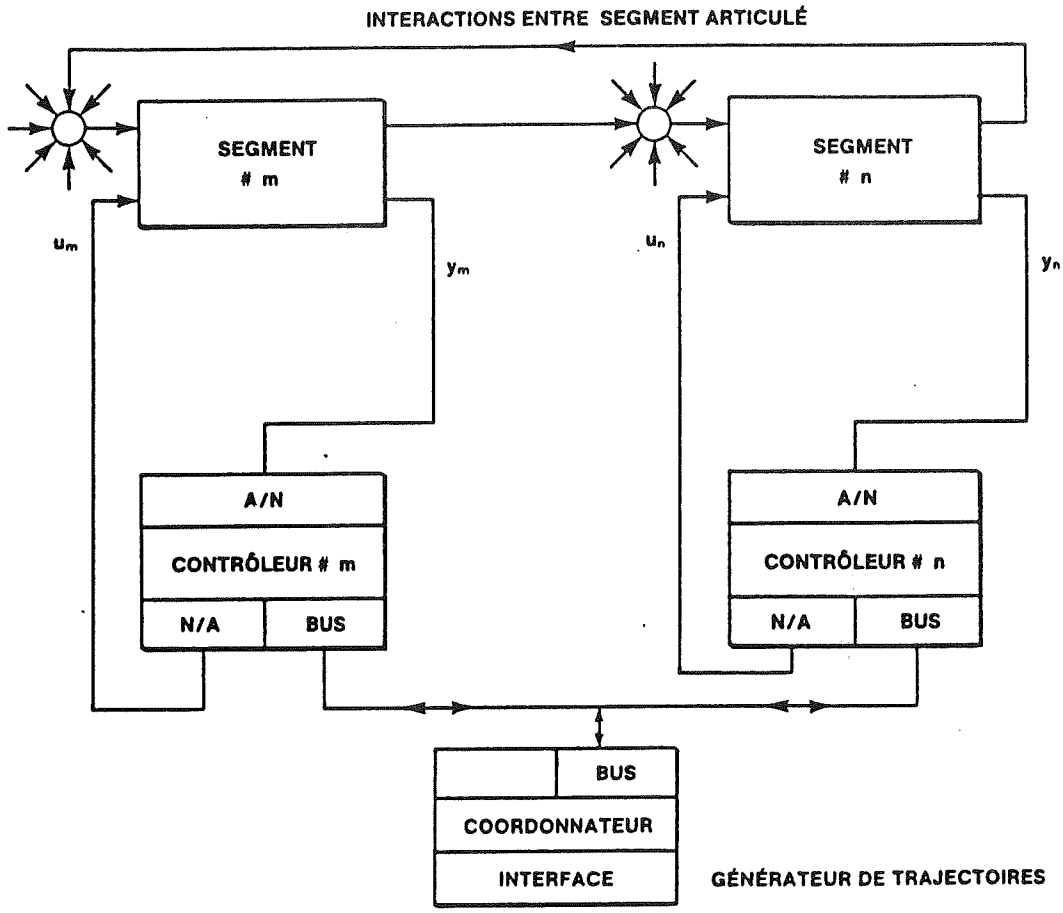


Figure 3.1: Structure of the overall controller

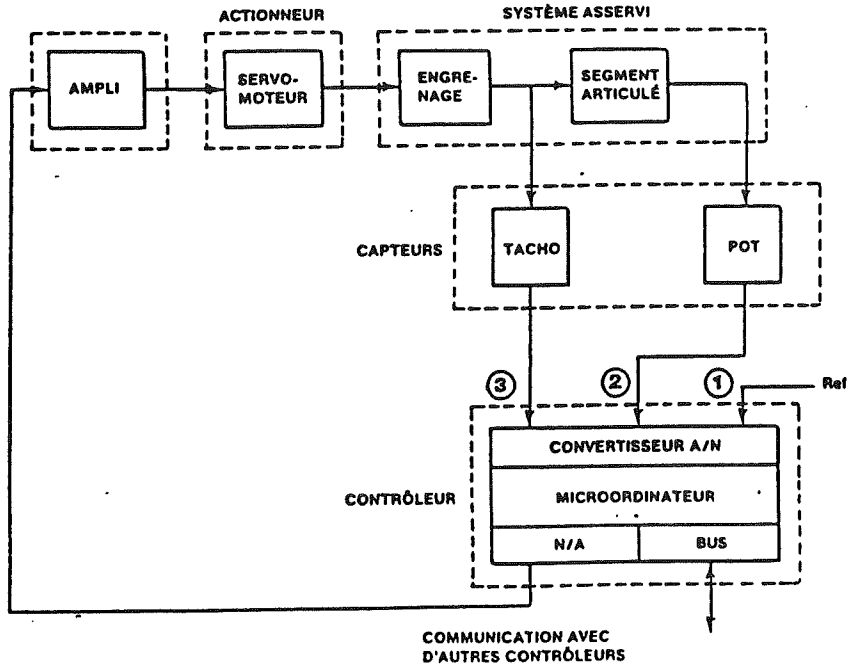


Figure 3.2: Structure of a link feedback controller

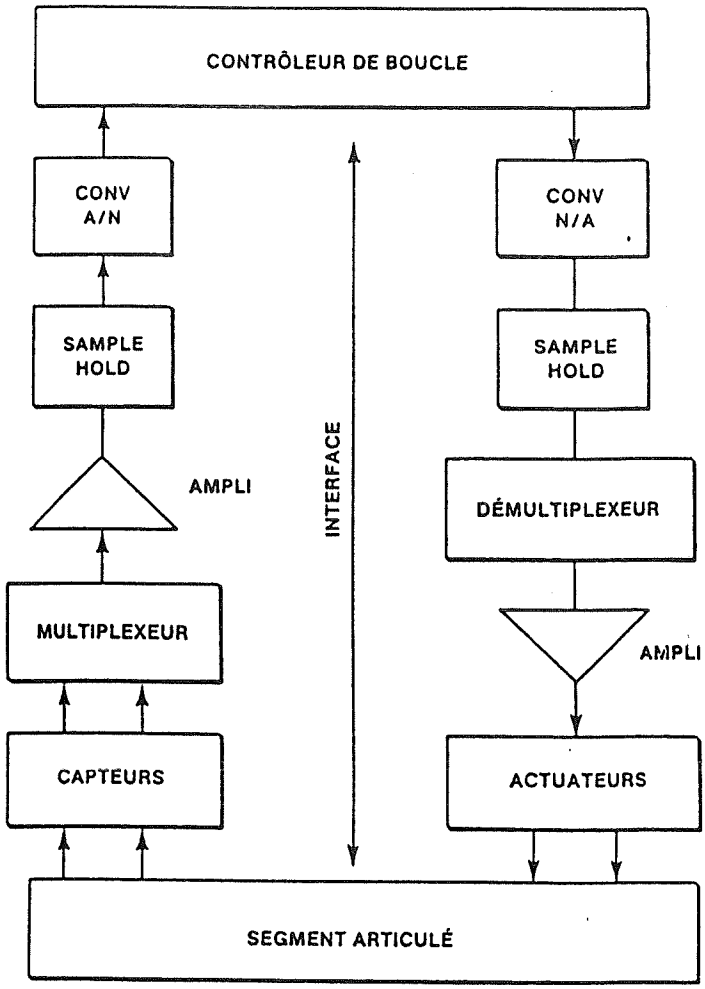


Figure 3.3: Components of a link feedback controller

rapidly, precisely and reliably as possible in spite of the presence of the usually unknown β. Such a problem has been the focus of much attention on the part of control theoreticians and most of the available standard control techniques turn out to be relevant to its solution.

One can mention in particular:

- linearization and classical theory (frequency, transfer function and static space techniques);
- Lyapunov and hyperstability theory;
- optimal control (LQG, maximum principle, Hamilton-Jacobi, etc.);
- adaptive control, identification;
- variable structure systems, etc.

In what follows we will discuss how these techniques are indeed applied to the control of robotic manipulators. In particular: the concept of computed torque and acceleration resolved controllers; the design of currently employed PID controllers; sliding mode controllers; adaptive and learning controllers.

2. MODEL BASED CONTROLLERS

2.1 Introduction

Model based controllers rely on the assumption that the dynamical model of the manipulator is perfectly known together with its parameters values and external perturbations [Le.1, p. 188-195]. Their potential adoption usually also presumes the availability of high performance actuators, transducers and computational units.

In spite of these demanding assumptions, these controllers are interesting in that they offer an ideal vehicle to put into perspective the physical, structural and computational complexity of the problem; they represent an efficient term of comparison against which to discuss the technical features of other controllers; they suggest many of the ideas and techniques which subsequently find at least a partial application in a variety of other more realistic control schemes.

2.2 Computed torque controller

Considering the manipulator dynamics as described in terms of the Lagrange-Euler model,

$$u_i = \sum_{j=1}^{n} D_{ij}\,\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk}\,\dot{q}_j\dot{q}_k + D_i,$$

the idea of a computed torque controller is to provide a control given by

$$u_i = u_{FFi} + u_{FBi}$$

with

$$u_{FFi} = \sum_{i=1}^{n} D_{ijc}\, \ddot{q}_{jD} + \sum_{j=1}^{n} \sum_{k=1}^{n} D_{ijkc}\, \dot{q}_{jm}\, \dot{q}_{km} + D_{ic}$$

and

$$u_{FBi} = \sum_{i=1}^{n} D_{ijc} \left(K_{1i} \left[q_{iD} - q_{im} \right] - K_{2i} \left[\dot{q}_{iD} - \dot{q}_{im} \right] \right),$$

where the subscripts m,c and D are used to denote measured, computed and desired values.

Under the assumption that measured, computed and actual values all coincide, that is

$$q_i = q_{im} \qquad \dot{q}_i = \dot{q}_{im}$$

and

$$D_{ij} = D_{ijc} \qquad D_{ijk} = D_{ijkc} \qquad D_i = D_{ic},$$

usage of such a control leads to

$$\sum_{j=1}^{n} D_{ij} \left[(\ddot{q}_{iD} - \ddot{q}_i) + K_{1i}\, (q_{iD} - q_i) + K_{2i}\, (\dot{q}_{iD} - \dot{q}_i) \right] = 0$$

and, from the nonsingularity of $[D_{ij}]$,

$$\ddot{q}_{iD} - \ddot{q}_i + K_{1i}(q_{iD} - q_i) + K_{2i}(\dot{q}_{iD} - \dot{q}_i) = 0$$

With a convenient choice of gain values K_{1i}, K_{2i}, one can then come up with a perfectly decoupled, arbitrarily poles assignable, second order link dynamics. Introducing the notation

$$h(q,\dot{q}) \triangleq \left[h_i(q,\dot{q}) \right] \triangleq \left[\sum_{j=1}^{n} \sum_{k=1}^{n} D_{ijk}\, \dot{q}_j \dot{q}_k + D_i \right],$$

one can represent the structure of a computed torque controller using the diagram in fig. 2.1. The numerical algorithm employed can be interpreted as an inverse dynamic problem, i.e.: compute the forces required to have an acceleration given by

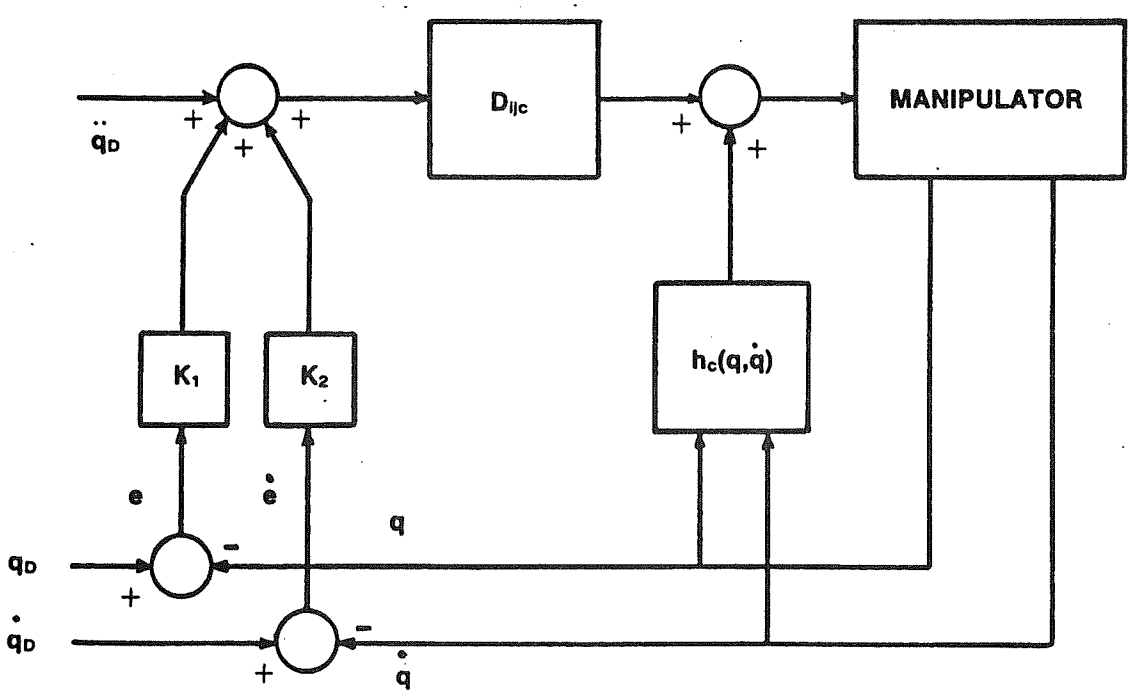


Figure 2.1: Structure of a computed torque controller

$$\ddot{q}^{*}=\ddot{q}_D+K_1(q_D-q)+K_2(\dot{q}_D-\dot{q})$$

in correspondence to speed and position $\dot{q}$, and q.
An effective way to implement such an algorithm is provided by the Lu-Walker-Paul approach [Fu.1, p. 117].

2.3 Acceleration resolved motion

An acceleration resolved motion controller is similar to a computed torque controller in that it strives at applying a force capable of generating an acceleration equal to the desired acceleration plus a PD component of the position error. The distinction is that these elements are now directly expressed in work space coordinates. Accordingly, one assumes to have available desired workspace position and orientation trajectories ($p_D(t)$, $\Omega_D(t)$) of the manipulator end effector, together with their linear and angular speed and acceleration denoted, respectively, with the symbols $v_D(t)$, $\omega_D(t)$, and $\dot{v}_D(t)$, $\dot{\omega}_D(t)$. Usually the vectors p_D, v_D and $\dot{v}_D$ refer to desired position speed and acceleration of the origin of the end effector frame with respect to the work space frame; Ω_D, ω_D and $\dot{\omega}_D$ refer to desired actual first and second derivative values of the angles (often Euler or RPY angles) characterizing the orientation of the end effector frame.

Control forces are then applied so as to generate end effector linear and angular accelerations given by the expression

$$\begin{bmatrix} \dot{v}^* \\ \dot{\omega}^* \end{bmatrix} = \begin{bmatrix} \dot{v}_D \\ \dot{\omega}_D \end{bmatrix} + K_1 \begin{bmatrix} p_D - p \\ \Omega_D - \Omega \end{bmatrix} + K_2 \begin{bmatrix} v_D - v \\ \omega_D - \omega \end{bmatrix}$$

where p, Ω, v and ω characterize actual position and velocity values. Under ideal conditions this leads to an error dynamics governed by the equation

$$\begin{bmatrix} \dot{v}_D - \dot{v} \\ \dot{\omega}_D - \dot{\omega} \end{bmatrix} + K_1 \begin{bmatrix} p_D - p \\ \Omega_D - \Omega \end{bmatrix} + K_2 \begin{bmatrix} v_D - v \\ \omega_D - \omega \end{bmatrix} = 0$$

that is, (upon a choice of diagonal K_1 and K_2): a perfectly decoupled in work space coordinates, arbitrarily poles assignable dynamics.

The control forces can be computed by applying the Luh-Walker-Paul inverse dynamic algorithm [Fu.1, p.117]. For this, one needs first to have available the manipulator links actual position and velocity, plus the desired acceleration. Position and velocity q and $\dot{q}$ are usually obtained from the link transducers. Denoting with J the manipulator Jacobian matrix, the links acceleration vector $\ddot{q}$ is computed by exploiting the following incremental kinematic equations

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = J\dot{q}$$

$$\begin{bmatrix} \dot{v}(t) \\ \\ \dot{\omega}(t) \end{bmatrix} = J\ddot{q} + \dot{J}\dot{q}$$

$$\ddot{q} = J^{-1} \begin{bmatrix} \dot{v}(t) \\ \\ \dot{\omega}(t) \end{bmatrix} - J^{-1}\dot{J}\dot{q} \quad .$$

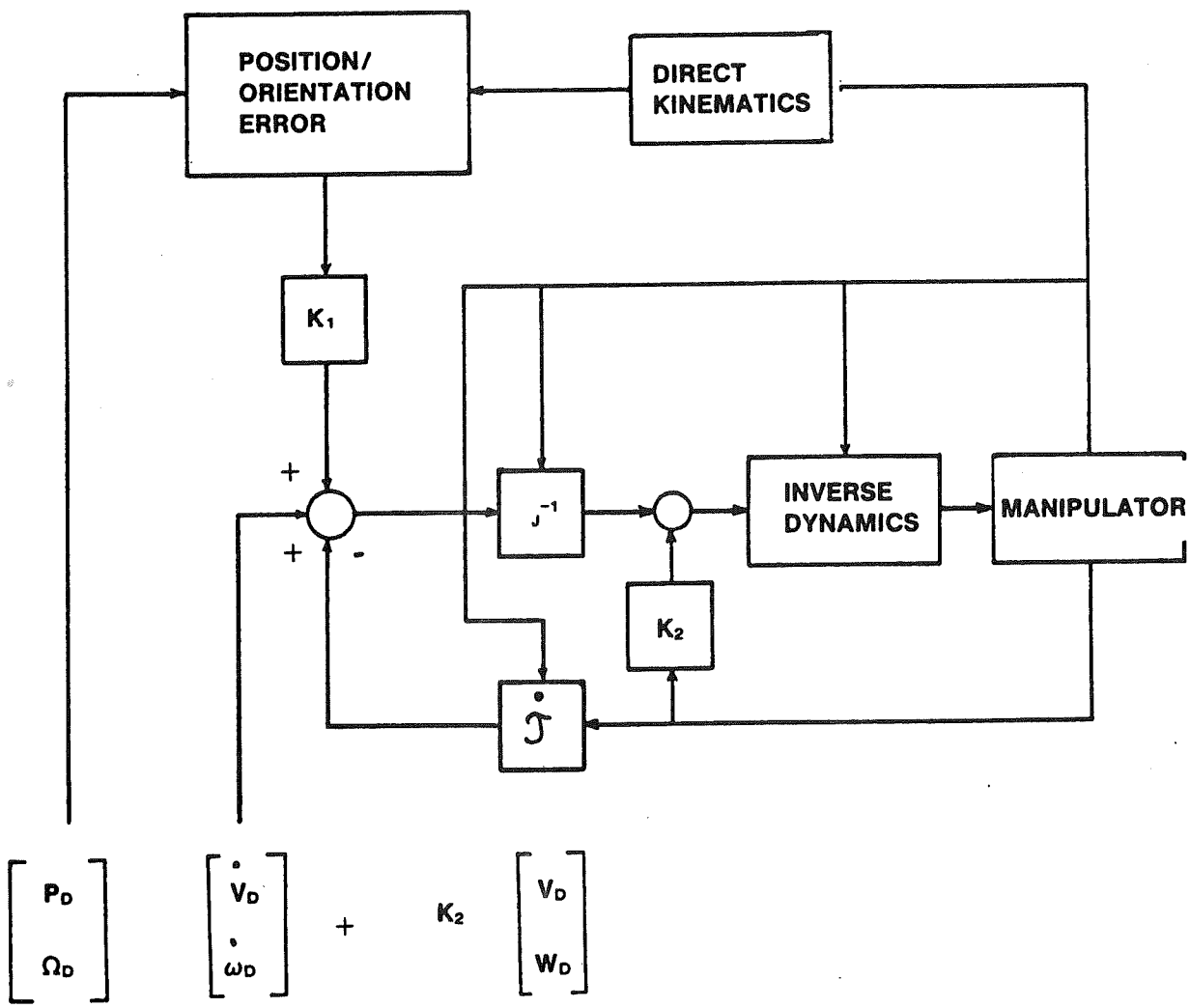


Figure 3.1: Structure of an acceleration resolved motion controller

3. PID CONTROLLERS

3.1 Introduction

In current industrial practice the control force applied by the controller to the i-th link is given by

$$u_i = u_{FFi} + u_{FBi}$$

where u_{FFi} is a centralized feed-forward component aimed at compensating for gravity; u_{FBi} is a decentralized feed-back component with the function of reducing the difference $x_{iD}-x_i$, where x_{iD} and x_i represent, respectively, required and actual values of link position and velocity.

The dynamics of the link can then be thought as given by

$$\dot{x}_{1i} = x_{2i}$$

$$\dot{x}_{2i} = -\frac{1}{\tau_i} x_{2i} + \frac{K_{mi}}{\tau_i} (u_{FBi} + \xi_i)$$

where ξ_i, the residual of gravity intercoupling centripetal and perturbation forces which have not been eliminated by the presence of u_{FFi}, is viewed as a perturbation. The control u_{FBi} is usually a nonlinear (relay) or linear (P, PD, PID) function of the difference $x_{iD}-x_i$.

In general, relay controllers are the simplest and the least expensive; however, as they usually produce vibration and imprecision, their use is often confined to

low cost, low performance manipulators. Higher class manipulators usually employ controllers of the PID type (fig. 2.1, 2.2).

3.2 Position Servo and Trajectory Tracking PID Controllers

The action provided by a typical position servo PID controller is given by (see fig. 2.1),

$$u_{FBi}=-K_1(x_1-x_{D1})-K_1x_2-K_3\int_0^t(x_1-x_{D1})dt+K_3K_4\ x_{1D},$$

where K_1, K_2, K_3 and K_4 represent proportional, derivative, integral and feedforward gains respectively. The reason to be of these gains is the classical one:

K_1 is indispensable for stability;

K_2 is required to improve speed of response;

K_3 is required to eliminate the influence of stationary perturbations;

K_4 compensates for the transient reponse deteriorating influence of K_3.

Under the hypothesis that K_{mi} and τ_i are constant, the dynamical behaviour of the link is described by the transfer functions

$$F_1=\frac{x_1}{x_{D1}}(s)=\frac{sK_m(K_1+K_3K_4)+K_3K_m}{\tau s^3+(1+K_mK_2)s^2+K_mK_1s+K_mK_3}$$

$$F_2=\frac{x_1}{\xi}(s)=\frac{sK_m}{\tau s^3+(1+K_mK_2)s^2+K_mK_1s+K_mK_3}$$

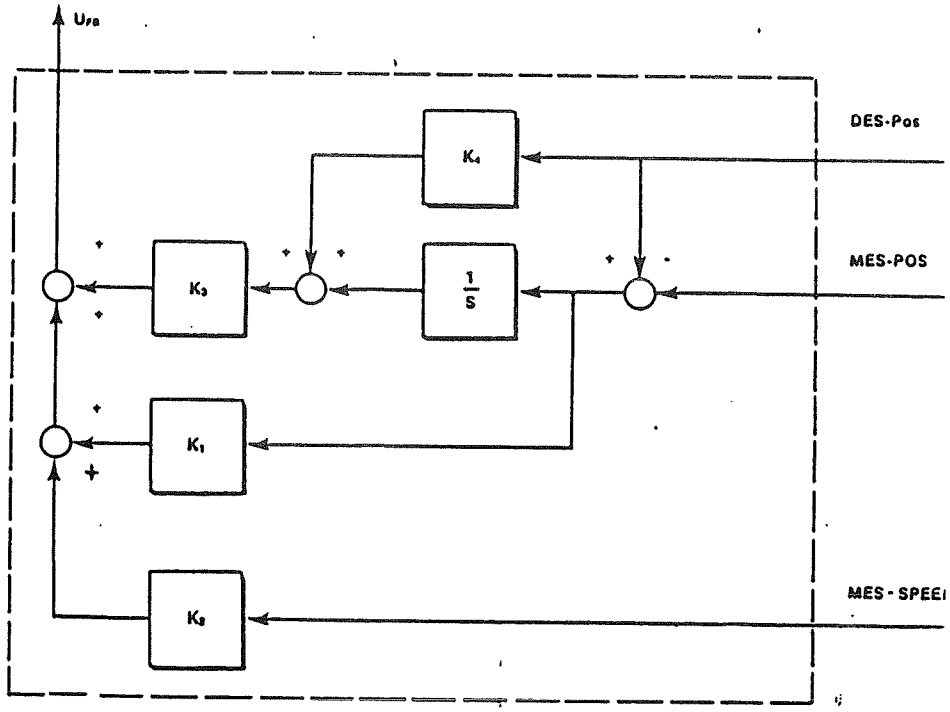


Figure 2.1: PID position servo controller

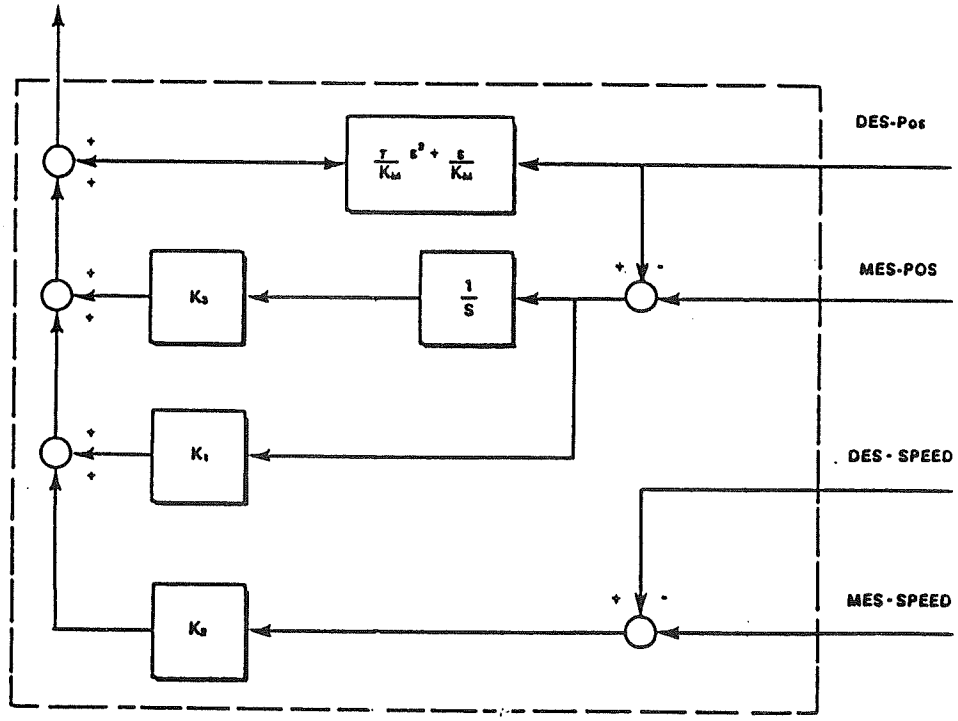


Figure 2.2: PID trajectory tracking controller

For stability one needs

$$K_3 > 0$$
$$K_2 > -1/K_m$$
$$K_1 > \tau\ K_3/(1+K_m K_2).$$

The values of $K_1,\ldots,K_4$ are usually chosen so as to attain some prescribed performance specifications often expressed in terms of response time, settling time, overshoot,control level constraints, sensitivity, robustness and similar. These specifications can usually be translated in terms of poles and zeros.

Indicating with p_1,p_2,p_3 and z the zero-pole configuration correspondent to the required specifications, and using the notations

$$a_1 \triangleq -\left[\frac{1}{p_1} + \frac{1}{p_2}\right] \qquad a_2 \triangleq \frac{1}{p_1 p_2}$$

one has

$$F_1 = \frac{(1-s/z)}{(1+q_1 s + a_2 s^2)\ (1-s/p_3)}$$

$$F_2 = \frac{s}{K_3(1+a_1 s+a_2 s^2)\ (1-s/p_3)}$$

and

$$K_1 = \frac{\tau}{K_m}\ (p_1 p_2 + p_2 p_3 + p_1 p_3)$$

$$K_2 = - \left[(p_1+p_2+p_3)\tau - 1 \right] / K_m$$

$$K_3 = - \frac{\tau}{K_m} p_1 p_2 p_3$$

$$K_4 = (-K_3 - K_1 z) / K_3 z$$

If, in addition to x_{D1}, one also has available $\dot{x}_{D1}$ and $\ddot{x}_{D1}$, the PID position servo controller may be replaced by a trajectory tracking PID controller whose action is described by

$$u_{FBi} = \frac{\tau}{K_m} \ddot{x}_{D1} + \frac{x_2}{K_m} - K_1(x_1 - x_{D1}) - K_2(x_2 - \dot{x}_{D1})$$

$$- K_3 \int_0^t (x_1 - x_{D1}) dt.$$

Using the notations

$$\Delta x_1 \triangleq (x_1 - x_{D1})$$

$$\Delta x_2 \triangleq (x_2 - \dot{x}_{D1})$$

$$\Delta x_3 \triangleq \int \Delta x_1 \, dt$$

one has

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

$$\Delta \dot{x}_2 = \frac{K_m}{\tau} \left[-K_1 \Delta x_1 - K_2 \Delta x_2 - K_3 \Delta x_3 + \xi \right]$$

$$\Delta \dot{x}_3 = \Delta x_1$$

By requiring that the dynamics of the system be once again characterized by the poles p_1, p_2 and p_3 one has

$$F_1 \triangleq \frac{x_1}{x_{D1}} = 1$$

$$F_2 \triangleq \frac{\Delta x_1}{\xi} = \frac{K_m}{\tau} \frac{s}{(s-p_1)\ (s-p_2)\ (s-p_3)}$$

The values and stability requirements of K_1, K_2 and K_3 are identical to the case of the position servo. A more complete discussion of PID controllers may be found in [RMDS.8, Le.1 p. 5-25] . A detailed development concerning a 2 degrees of freedom SCARA manipulator under relay, dual mode, (linear in the proximity of the target, non linear when far away), and PID control may be found in [RMDS.4]. A hands-on analysis of the dynamical behavior of such a system may be developed through the use of the simulators POSSPEED and SIMNOG available at the Computer Center, (Prometeus), (see appendices A,B).

3.3 Observations

The dependence of the link parameters K_{mi} and τ_i on the manipulator configuration may lead to a situation where a given set of gains is satisfactory in one configuration and unsatisfactory in others. This shortcoming represents the main justifica-

tion for the introduction of an adaptive strategy whereby the values of $K_1,\ldots,K_4$ are continuously adjusted to the estimated values of K_{mi} and τ_i (see forthcoming section).

The dynamical intercoupling generated by the simultaneous displacement of more than one link may lead to ξ values which may not be adequately neutralized by the PID controller. This problem may be solved by introducing a dynamical decoupling in the controller feedforward component; an alternative solution may be provided by a PID/sliding mode controller (see relevant section).

Finally, in high accuracy high speed manipulator centripetal and Coriolis effects may further deteriorate performance. This type of manipulators may require full computed torque controllers complemented with nonlinear multivariable adaptive capabilities. The usage of a learning control may also be required (see relevant sections).

4. ADAPTIVE CONTROLLERS

4.1 Introduction

The dynamic performance of a robotic manipulator equipped with a controller provided by either the classical PID or the more advanced model based approach is dependent upon the knowledge of the plant parameter values. While such a knowledge is in general incomplete, it may often be improved during the operation of the system by monitoring the input/output behavior of the plant. It is quite natural then to consider an adaptive scheme whereby plant parameter knowledge is sequentially updated and exploited to improve the controller action.

Early robotic implementations of such a scheme have given rise to the utilization of classical model reference and self-tuning adaptive controllers [Le.1].
While these controllers have been mainly considered with a view of rendering adaptive the decentralized feedback part of the controller, more recently, new advanced controllers capable of extending adaptivity to the feedforward controller component have been proposed [Cr.1].

4.2 Model reference adaptive controllers (MRAC)

The feedback system characterizing an adaptive controller with a reference model is made of a plant, a classical

controller, a reference model and a controller gain adaptation module (fig. 2.1). Its principle of operation is the following: the reference model is chosen so as to generate the desired output trajectory; the controller gains are initially adjusted so that under nominal operating conditions the plant output coincides with the reference model output; when the plant parameters vary, plant and model outputs no longer coincide and a difference is detected; the adaptation module exploits this difference so as to generate a variation in the controller gains having the effect of reducing it.

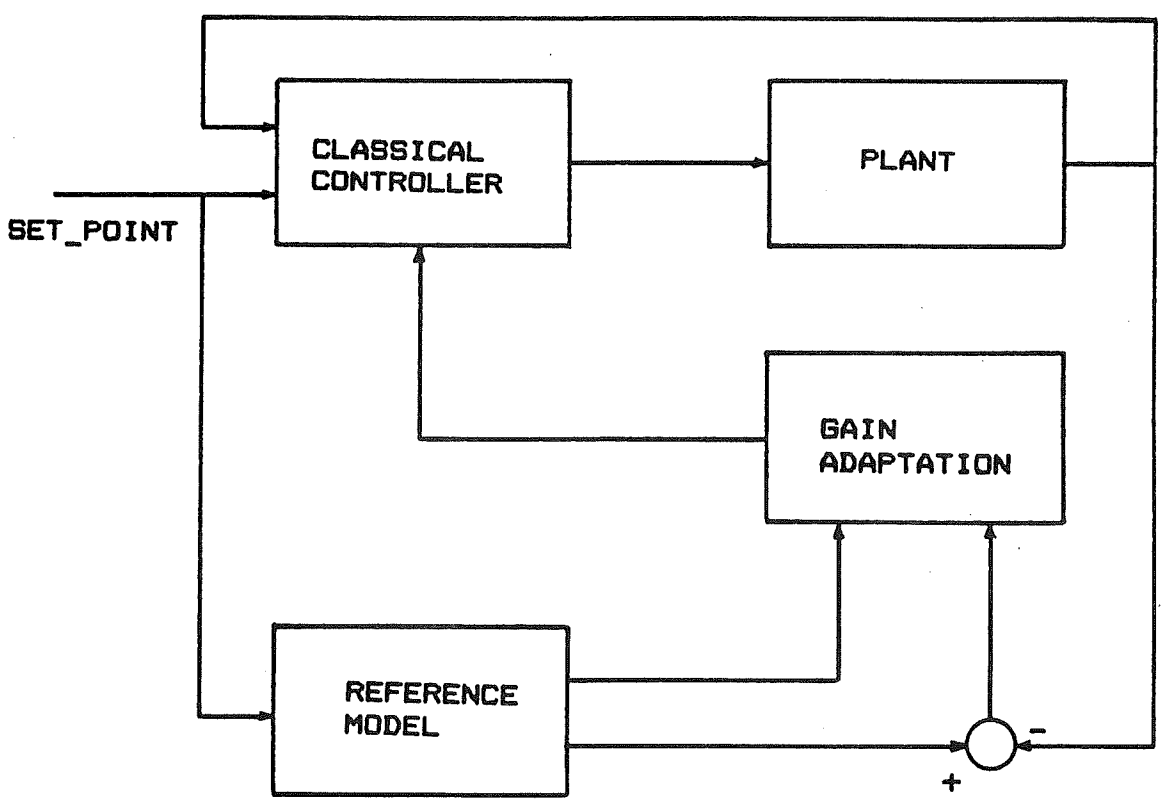


Figure 2.1: Structure of a model reference adaptive controller

To illustrate the implementation of this idea into a computational algorithm, the dynamics of the plant together with the classical controller is modelled by

$$\dot{\underline{x}} = \underline{f}_s(\underline{x}, \underline{\alpha}, \underline{r}, \underline{d})$$

where

$\underline{f}_s$: is a function representing the dynamic behavior of the feedback system;

$\underline{x}$: is the state;

$\underline{\alpha}$: is a variable parameter;

$\underline{r}$: the system entry;

$\underline{d}$: reflects the presence of external disturbances.

The reference model dynamics corresponds to the feedback system desired behavior and is given by

$$\dot{\underline{y}} = \underline{f}_m (\underline{y}, \underline{a}, \underline{r})$$

where:

$\underline{f}_m$: represents the dynamics of the reference model;

$\underline{y}$: is the desired state;

$\underline{a}$: is the nominal value of $\underline{\alpha}$.

The parameter vector α is a function of the controller gains as well as of a certain number of plant parameters subject to unpredictable variations. The idea behind the MRAC is to compensate disturbances and unpredictable variations with adaptively induced gain variations. This is done

by attempting to minimize the output error

$$e(t) = y(t) - x(t)$$

As a measure of this error one considers a quadratic function of the type

$$V(e) = \frac{1}{2} e'Qe$$

where Q is a conveniently chosen positive definite symmetric matrix.

An effective procedure to minimize V(e) may be obtained by applying the gradient method. This approach leads to

$$\dot{\alpha} = -\gamma \frac{\partial V}{\partial \alpha} \cong -\gamma \frac{\partial e'}{\partial \alpha} \quad Qe$$

where $\frac{\partial e}{\partial \alpha}$ represents the plant error sensitivity matrix, while γ is a conveniently chosen acceleration factor matrix. The computation of $\frac{\partial e}{\partial \alpha}$ is difficult to implement because α is unknown. This matrix is then approximated by considering $\partial y/\partial a$, the sensitivity matrix of the reference model. Formally one has

$$\frac{\partial e}{\partial \alpha} = \frac{\partial (y-x)}{\partial \alpha} = - \frac{\partial x}{\partial \alpha} \cong - \frac{\partial y}{\partial a}$$

The matrix $\frac{\partial y}{\partial a}$ is computed by using the sensitivity model

$$\frac{\partial \dot{y}}{\partial a} = \frac{\partial f_m(y,a,r)}{\partial y} \frac{\partial y}{\partial a} + \frac{\partial f_m}{\partial a} \quad (y,a,r).$$

A detailed educational implementation of this algorithm to a robotic link may be found in [RMDS.5] . A simulator MRAC, available at the Computer Center, (Prometeus), allows one to gain a hands-on experience with the working of such a system.

4.3 Self-Tuning adaptive controllers

The conceptual operation of a self-tuning adaptive controller is as follows, (see fig. 3.1): an estimator module gives a recursive plant parameter estimate which is the most compatible with the available plant input/output behavior; an adaptation module updates the controller parameters on the basis of such an estimate; the controller operates as a standard classical controller.

In a typical application of this scheme the plant is modelled in terms of a linear difference equation of the type

$$y(m+1)=a_0\ u(m)+a_1\ u(m-1)+\ldots+a_p\ u(m-p)$$

$$-\ b_0\ y(m)-\ \ldots\ -b_n\ y(m-n)$$

where $u(\cdot)$, $y(\cdot)$, a_i, b_j represent respectively, input, output and unknown parameters of the plant.

The plant parameter estimator is usually of the recursive least square type [As.1]. Using the notation

$$\hat{\theta}(m) \triangleq \begin{bmatrix} a_o \\ \vdots \\ a_p \\ b_o \\ \vdots \\ b_n \end{bmatrix} (m)$$

to indicate the parameter estimate at time m, and with

$$x(m) \triangleq \begin{bmatrix} u(m) \\ \vdots \\ u(m-p) \\ -y(m) \\ \vdots \\ -y(m-n) \end{bmatrix}$$

the input/output data vector relevant to the computation of y(m+1), a typical recursive algorithm computes $\hat{\theta}(m+1)$ from $\hat{\theta}(m)$, y(m+1), x(m) according to the following algorithm:

$$\hat{\theta}(m+1)=\hat{\theta}(m)+K(m)\left[y(m+1)-x'(m)\hat{\theta}(m)\right]$$

$$\gamma(m) = 1/\left[1+x'(m)P(m)x(m)\right]$$

$$K(m) = \gamma(m)\ P(m)\ x(m)$$

$$P(m+1) \triangleq \left[P(m)-K(m)x'(m)P(m)\right] \frac{1}{\lambda}$$

where:

$$P(m) \triangleq \left[X'(m)\ X(m)\right]^{-1}$$

$$X'(m) \triangleq \begin{bmatrix} x'(1) \\ \hline x'(m-1) \end{bmatrix}$$

and λ is a forgetting factor. This algorithm gives the casual parameter estimate $\hat{\theta}(m+1)$ which minimizes the expression

$$J \triangleq \sum_{i=1}^{m} \varepsilon^2(i)\ \lambda^{m-i}$$

$$\varepsilon(i) \triangleq \left[y(i+1) - x'(i)\hat{\theta}(i)\right] .$$

The classical controller employs a strategy which is usually of the pole placement, minimum variance or linear optimal regulator type. Its gains are computed at each instant m by the gain update module. The algorithm adopted by this module presumes the plant parameters to be time invariant and equal to $\hat{\theta}(m)$ (<u>certainty equivalent principle</u>).

A detailed application of this algorithm to a robotic link is described in [RMDS.7]. A simulator, SELF-TUNE, available at the Computer Center, (Sala Prometeus), allows one to gain a hands on experience with the working of such a system.

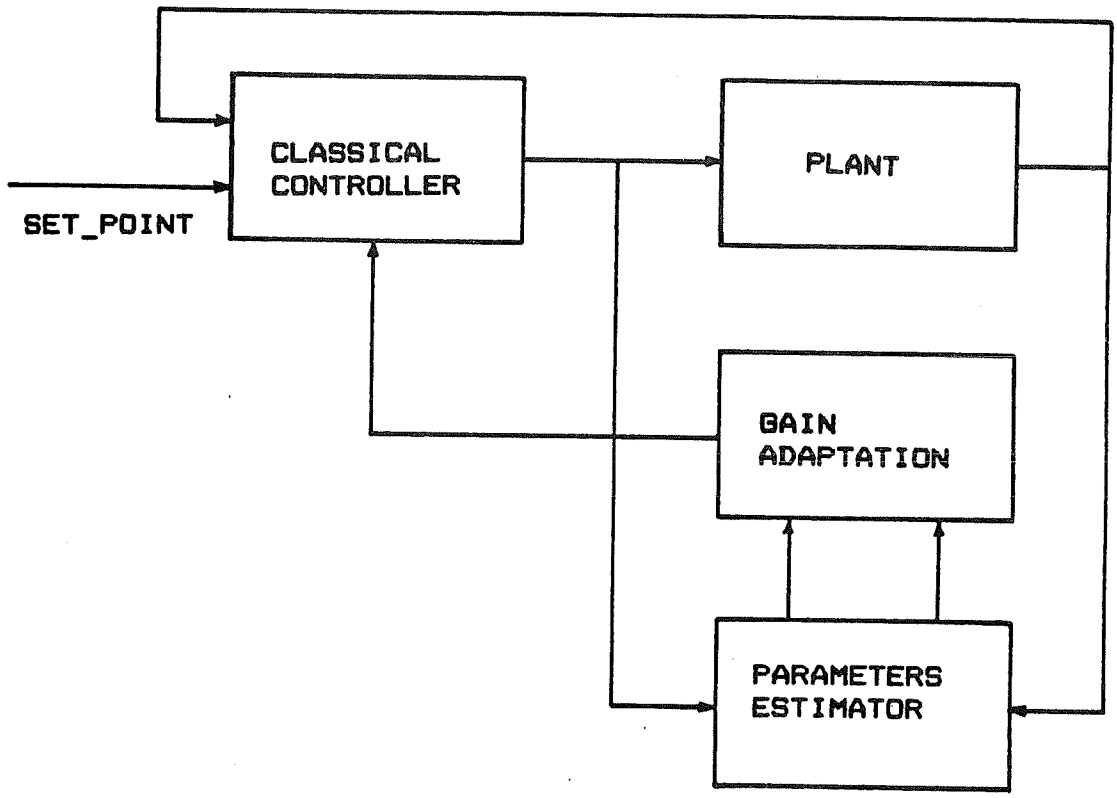


Figure 3.1: Structure of a self-tuning adaptive controller

4.4 Multivariable nonlinear adaptive controllers

The potential of the model reference and self-tuning approaches is somewhat limited by the usual assumption that the plant is linear and made of decoupled scalar systems. More seriously: the adaptivity of these schemes is usually confined to the controller feedback component. Recently,

multivariable non-linear schemes have been developed which take into specific account the particularities of the manipulator dynamics [Cr.2]. A major step forward is in that the adaptivity now concerns both the controller feed-forward and feedback components. Too: identification and control are no longer independent.

The main ideas at the basis of this new development can perhaps be best illustrated by discussing its implementation in a scheme where a precomputed torque controller is used. Such a scheme is represented in fig. 4.1, where $\hat{D}$ and $\hat{h}(q,\dot{q})$ represent respectively the estimated values of the manipulator inertia matrix $[D_{ij}]$ and of the vector

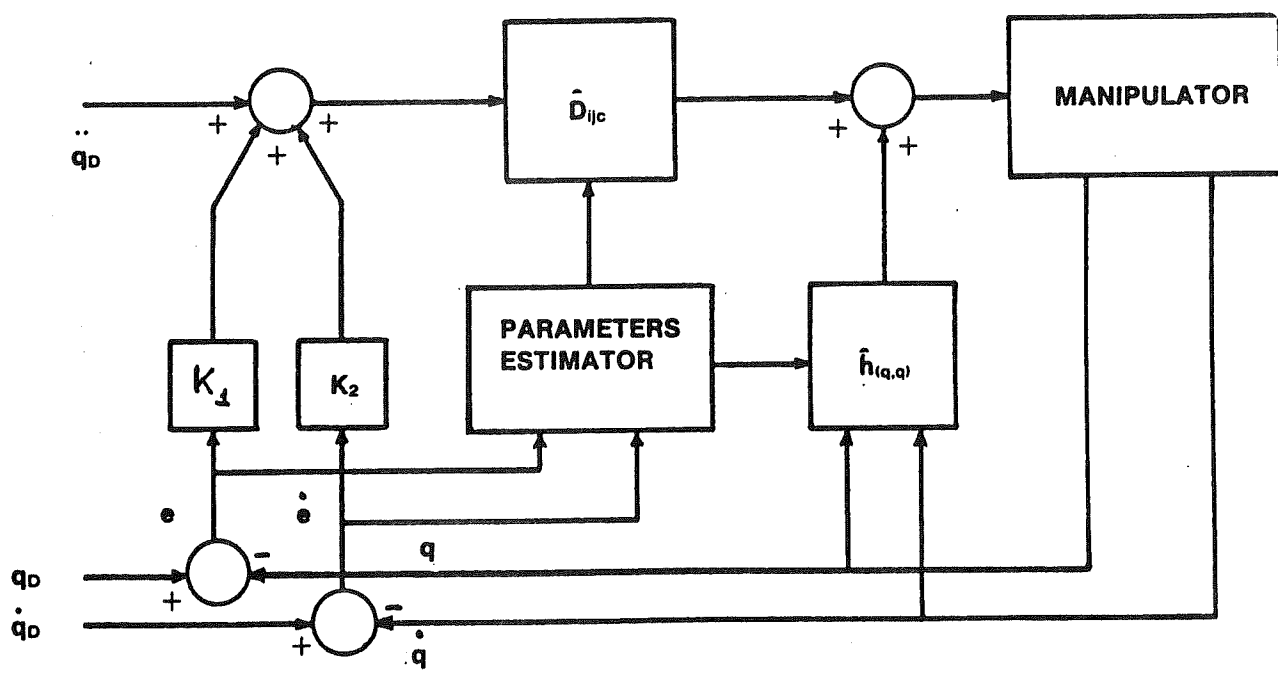


Figure 4.1: Multivariable nonlinear adaptive control of a robotic manipulator

$$h(q,\dot{q}) = \left[h_i(q,\dot{q})\right] \triangleq \left[\sum_{j=1}^{n} \sum_{k=1}^{n} D_{ijk}\, \dot{q}_j \dot{q}_k + D_i\right]$$

The unknown parameters of the manipulator are described in terms of a vector P. This vector is chosen so as to have the representation

$$D\,\ddot{q} = \tilde{D}\,\ddot{q} + W_1\,[q,\ddot{q}]\,P$$

$$h(q,\dot{q}) = \tilde{h}(q,\dot{q}) + W_2\,[q,\dot{q}]\,P$$

where $\tilde{D}$ and $\tilde{h}$ are (known) parts of $D\,\ddot{q}$ and $h(q,\dot{q})$ which do not depend on P; W_1 and W_2 are known functions of $q,\dot{q}$ and $\ddot{q}$. Using the notation $\hat{P}$ to denote the current estimate of P and $W \triangleq [W_1 \,\vdots\, W_2]$, the estimation algorithm is based upon the equation

$$\dot{\hat{P}} = KW'\,\hat{D}^{-1}\,(e+\psi\dot{e})$$

where K and ψ are conveniently chosen real positive diagonal matrices and e denotes the error between the desired and the actual manipulator links position

$$e \triangleq q_D - q.$$

An outline of the justification of such an algorithm goes as follows [Cr.2, p.51] :

i. Usage of the computed torque approach suggests a control action given by

$$\Gamma = \hat{D}(q)\,\ddot{q}^* + \hat{h}(q,\dot{q})$$

where

$$\ddot{q}^* = \ddot{q}_D + K_1 e + K_2 \dot{e}$$

and K_1, K_2 are diagonal matrices.
This, in turn, implies

$$\hat{D}\left[\ddot{q}_D + K_1 e + K_2 \dot{e}\right] = D\ddot{q} + h - \hat{h}$$

$$\hat{D}\left[\ddot{e} + K_1 e + K_2 \dot{e}\right] = \left[D - \hat{D}\right]\ddot{q} + h - \hat{h}$$

$$\ddot{e} + K_1 e + K_2 \dot{e} = \hat{D}^{-1} W(q,\dot{q},\ddot{q})\Phi$$

where

$$\Phi \triangleq P - \hat{P}.$$

Using the notation

$$u \triangleq \hat{D}^{-1} W\Phi$$

it follows

$$\ddot{e} + K_1 e + K_2 \dot{e} = u$$

<u>ii</u>. Considering

$$e_1 \triangleq e + \psi\dot{e},$$

using Laplace notations one has

$$e_1(s) = \left[I + \psi s\right]\left[s^2 I + sK_2 + K_1\right]^{-1} u.$$

The choice of diagonal ψ is made so that the mapping $u \to e_1$ is strictly positive real.
From a well known strictly positive real result, this implies that the mapping $u \to e_1$ is representable in terms of a dynamical system

$$\dot{x} = Ax + Bu$$
$$e_1 = Cx$$

with positive real symmetric matrices Q_1, Q_2 such that

$$A'Q_2 + Q_2A = -Q_1$$
$$Q_2B = C'.$$

<u>iii</u>. One may now consider a Lyapunov function of the type

$$V(x,\Phi) \triangleq xQ_2x + \Phi' K^{-1}\Phi$$

with K a diagonal positive real matrix. Observing that the chosen estimation strategy implies

$$\dot{\hat{P}} = KW' \hat{D}^{-1} e_1$$

hence

$$\dot{\Phi} = -KW' \hat{D}^{-1} e_1$$

one has

$$\begin{bmatrix} \dot{x} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} A & B\hat{D}^{-1}W \\ -KW'\hat{D}^{-1}C & 0 \end{bmatrix} \begin{bmatrix} x \\ \Phi \end{bmatrix}$$

By computing

$$\dot{V} = \frac{\partial V}{\partial x} \dot{x} + \frac{\partial V}{\partial \Phi} \dot{\Phi}$$

one obtains

$$\dot{V} = -x'Q_1x$$

which implies $x \to 0$, hence $e_1 \to 0$ and $e \to 0$.

<u>iv</u>. It is further shown in [Cr.2, p.58-63] that:

- this procedure is robust with respect to external perturbations;
- under the (persistently exciting equivalent) condition

$$\int_{t_o}^{t_o+\rho} W'W \, dt \geq \alpha I$$

  for $\rho > 0$ and some $\alpha > 0$, one also has $|\hat{P}-P| \to 0$;
- if one knows that the parameters values lie in a certain interval

$$l_i \leq p_i \leq h_i$$

  then one may advantageously modify the algorithm with the variant

$$\hat{p}_i(t^+) = l_i \qquad \text{if} \qquad \hat{p}_i(t) \leq l_i - \delta$$

$$\hat{p}_i(t^+) = h_i \qquad \text{if} \qquad \hat{p}_i(t) \geq h_i + \delta$$

  for some $\delta > 0$.

5. SLIDING MODE CONTROLLERS

5.1 Introduction

Even with the utilization of an adaptive controller, it may still be difficult to guarantee a satisfactory performance in high accuracy high speed manipulators where fast parameter variations and high level nonstationary perturbations may be expected. Sliding mode controllers, with a behavior which can potentially be made solely dependent on the controller and not on the plant parameters, present a natural addition to the arsenal of available tools. As these controllers are not usually covered in a classical automatic control course, in what follows we will give a brief introduction of the concept together with the subsequent presentation of some selected results.

To illustrate the concept, consider a robotic link described by [Ut.1, p. 15-17]

$$\dot{x}_1 = x_2 \qquad a \triangleq \frac{1}{\tau}$$

$$\dot{x}_2 = -a\, x_2 + b(u+\xi) \qquad b \triangleq Km/\tau$$

where, as usual,

$x_1 \triangleq$ position of the link;

$x_2 \triangleq \dot{x}_1$;

$u \triangleq$ controller action;

τ, $K_m \triangleq$ time constant and static gain of the link;

$\xi \triangleq$ input equivalent perturbation generated by external unknown forces and parameter variations.

Consider the (<u>sliding mode</u>) function

$$slm \triangleq x_2 + c\, x_1$$

together with the control strategy

$$u = -\psi x_1 + \gamma$$

where

$$\psi \triangleq \alpha \quad \text{if } x_1\, slm > 0,$$

$$\psi \triangleq \beta \quad \text{if } x_1\, slm < 0,$$

and

$$\gamma = -M\ SIGN(slm)$$

One can easily verify that if

$$\alpha > \max \left\{ \frac{1}{b}(a-c)c \ , \ \frac{a^2}{4b} \right\}$$

$$\beta < \frac{1}{b}(a-c)c \quad , \quad M > |\xi|$$

then slm(t) tends to zero. With the (sliding mode) condition

$$slm(t) = 0 \qquad\qquad t \geqq t_0$$

satisfied, one has

$$x_1(t) = e^{-c(t-t_0)} x_1(t_0).$$

This implies that as long as the controller is able to impose the sliding mode condition, the link state trajectory depends on the controller parameter c and is independent from the plant parameter values τ, K_m, ξ.

This example suggests that a robotic controller might be designed by adopting a strategy whereby the desired behavior of the manipulator is first translated in terms of a (sliding mode) condition of the type

$$Cx = 0$$

where x is the state of the manipulator and C is a suitable matrix. A sliding mode control would then be subsequently determined which is capable of satisfying this condition. Expected advantages being in the simplicity of the ensuing controller, a good dynamic and static behavior, a strong robustness to parameter variations and external perturbations. The same type of strategy may be also applied to state estimation and parameter identification; extensions involve adaptive model reference and self-tuning controllers.

5.2 Some Selected Results

Given a dynamical system [Ut.1, p.80]

$$\dot{x} = f(x,t) + B(x,t)u$$

with a controller defined by

$$u_i = u_i^+(x,t) \quad \text{for } s_i(x)>0 \quad , \quad i=1\ldots m$$

$$u_i^-(x,t) \quad \text{for } s_i(x)<0$$

where u_i^+, u_i^- and s_i are certain continuous functions, a <u>sliding mode controller</u> is defined by following two conditions:

<u>i</u>. no trajectory entirely generated by one of the 2^m potential control functions is contained in

$$S(x,t) \triangleq \{x \mid x \in \bigcap_i s_i(x)=0 \} \; ;$$

<u>ii</u>. for any $\varepsilon>0$, there exist $\Delta_0>0$ and $\delta>0$ such that: for any x_0 with a distance from $S(x,t_0)$ smaller than δ, the trajectory of

$$\dot{x}=f(x,t)+B(x,t)\tilde{u}(x,t) \quad , \quad x(t_0)=x_0$$

with $\tilde{u}(x,t)$ such that

$$\tilde{u}_i=u_i \quad \text{if} \quad \|s_i(x,t)\| > \Delta_0$$

$$\min(u_i^+,u_i^-) \leq \tilde{u}_i \leq \max(u_i^+,u_i^-) \text{ otherwise,}$$

has the property that

$$\sum_{i=1}^{n} s_i^2(x,t) < \varepsilon.$$

The control ideas characterizing the sliding mode controller design philosophy are well represented by the following two fundamental results.

<u>Theorem 1.</u> A necessary condition for a sliding mode controller to exist is that: a u_{equ}, (<u>equivalent control</u>), solution of

$$\dot{s} = G\left[f(x,t)+B(x,t)u_{equ}\right] = 0 ,$$

where: G is a matrix whose rows are given by the gradients with respect to x of the s_i components of $s \triangleq \{s_i\}$, exists and satisfies the inequalities

$$\min(u_i^+,u_i^-) \leq u_{equ} \leq \max(u_i^+,u_i^-)$$

<u>Theorem 2</u>. For the dynamical system to be under sliding mode control in the domain S(x,t), it is sufficient that: for all $x \in S(x,t)$ there exist a function v(s,x,t) continuously differentiable with respect to all its arguments such that in a certain region, Ω, of the subspaces $s_1,\ldots,s_m$ containing the origin:

<u>i</u>. v(s,x,t) is positive definite with respect to s;

<u>ii</u>. on the sphere $||s|| \leq R$ for all $x \in \Omega$ and any t:

$\inf v=h_R>0$, $\sup v=H_R>0$

$$\|s\| \leq R \qquad \|s\| \leq R$$

with h_R and H_R only dependent on R;

<u>iii</u>. a total time derivative of v is negative everywhere this function is defined and

$$\sup_{\|s\| < R} \dot{v} = -m_R.$$

with $m_R > 0$ only dependent on R.

The next theorem illustrates the nature of specialized application results which may be obtained from theorem 1 and 2.

<u>Theorem 3</u> [Ut.1, p.148] A necessary and sufficient condition for the system

$$\dot{x} = A(t)x + B(t)u,$$

equipped with the controller

$$u = \sum_{i=1}^{n} \psi_i x_i - \delta \cdot SIGN(slm)$$

with $\delta > 0$, $slm \triangleq cx$

$$\psi_i = \alpha_i \quad \text{if } x_i slm < 0$$

$$\psi_i = \beta_i \quad \text{otherwise,}$$

to be under sliding mode control is that

$$\alpha_i \leq \max_i \quad \langle c, a_i \rangle / \langle c, B \rangle$$

$$\beta \geq \min_i \langle c, a_i \rangle / \langle c, B \rangle$$

where: a_i is the i-th column of A(t).

The following result gives an indication of the limits of a sliding mode controller.

Theorem 4. [Ut.1, p.213] . Let

$$\dot{x} = Ax+Bu+Df$$

be under sliding mode control. A necessary condition for the system dynamics to be invariant with respect to the perturbation f is that the columns of the D matrix be in the range of the B matrix.

A more complete treatment of this subject may be found in the cited references (see also [Fu.1, p.226]). A detailed educational study of a sliding mode controller for a robotic link with an elastic transmission can be found in [RMDS.3]. A hands-on analysis of the dynamical behavior of such a system can be developed using the simulator ROBEL (available at the Computer Center; see Appendix B).

5.3 PID/Sliding mode controllers

A PID/sliding mode controller is obtained by modifying a classical PID with the parallel addition of a nonlinear (almost) on-off switching element (fig. 3.1). The idea is to

complement the well established transient performance and steady state rejection properties of the classical PID with the potential robustness to plant parameter variation and external perturbations of the sliding mode controller.

To illustrate, consider a robotic link described by

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

$$\dot{x}_2 = -\frac{1}{\tau} x_2 + \frac{K_m}{\tau}(u+\xi)$$

where u is the output of a controller with the structure indicated in fig. 3.1. With reference to this figure, observe that if Δu, the control contribution of the switching element, is capable of imposing the <u>sliding mode</u> condition

$$slm(x) \triangleq a_1 x_1 + a_2 x_2 + x_3 = 0$$

where

$$\dot{x}_3 = x_1 - x_{1D},$$

then it would follow

$$slm = 0$$

which implies

$$x_1(s) = \frac{x_{1D}(s)}{1+a_1 s+a_2 s^2} \quad ;$$

that is: a response completely independent of ξ and of the plant parameters.

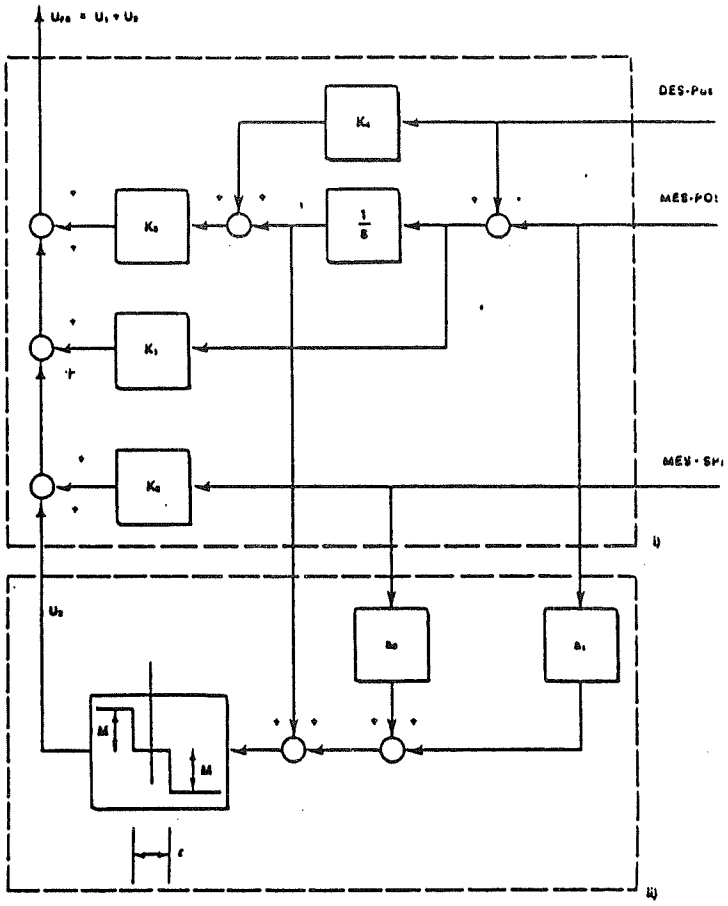


Figure 3.1: PID/Sliding mode servocontroller

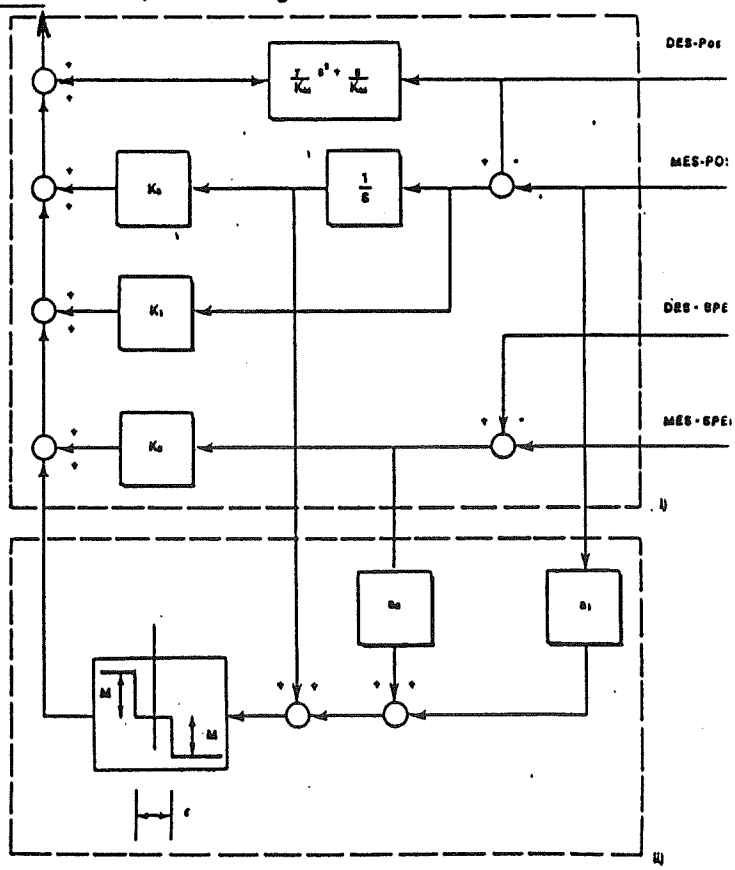


Figure 3.2: PID/Sliding mode trajectory tracking controller

To determine the requirements for a Δu with such a capability, let $K_1 \ldots K_4$ be such that the PID linear component, in the absence of ξ and Δu, generates an input/output transfer function with poles p_1, p_2, p_3 and zero z. Let $a_1 \triangleq -\frac{1}{p_1} - \frac{1}{p_2}$, $a_2 \triangleq \frac{1}{p_1 p_2}$, $z = p_3$.

Observe that, using Laplace notations,

$$slm(s) = \frac{K_m}{\tau} \frac{\Delta u + \xi}{(s - p_3)}$$

and

$$\dot{slm}(s) = \frac{K_m}{\tau} (\Delta u + \xi) + p_3 \frac{K_m}{\tau} \left(\frac{(\Delta u)}{(s - p_3)}\right.$$

hence

$$\frac{1}{2} \frac{d}{dt} (slm(t)^2) = p_3 \; slm^2(t) + \frac{K_m}{\tau} (\Delta u + \xi) slm(t)$$

To obtain $slm(t) \to 0$ it is then sufficient to impose $\frac{d}{dt} (slm^2(t)) < 0$. Since $p_3 < 0$, for this it is sufficient to simply choose

$$\Delta u = -M \; SIGN(slm)$$

where $M > \sup |\xi|$.

5.4 Observations

The $M > \sup |\xi|$ inequality suggests that while one may not need to know the value of the perturbation acting on the link, an evaluation of its upper value is required. Since the perturbation reflects the dynamical behavior of other links of the manipulator which may in turn be itself dependent on the value of M, a convenient procedure to obtain such an evaluation is inspired by the "hierarchy of control method" [Ut.1, p.101] [Le.1,p.210-220]. The controller for link i is designed sequentially by: starting from the last link; assuming the dynamics of all the preceding links to be in a sliding mode; continuing the procedure down to the first link.

The sliding mode action may in practice be approximated by replacing the switching control

$$\Delta u = - M \text{ SIGN } (slm)$$

with

$$\begin{aligned} \Delta u &= - M \text{ SIGN } (slm) && \text{for} \quad |slm| > \epsilon \\ &= 0 && \text{for} \quad |slm| < \epsilon \end{aligned}$$

The ϵ value is chosen sufficiently small as to ret ain the desired robustness property of the sliding mode action,

and large enough as to reduce the high frequency chattering which characterizes the operation of such an action.

The sliding mode controller potential of neutralizing not only the influence of external perturbations but also that of parameter variations follows from the fact that this latter influence may be represented in terms of an input-equivalent perturbation

$$\xi = \frac{\tau}{K_m}\left[\left[\frac{1}{\tau_{nom}} - \frac{1}{\tau}\right] x_2 + \left[\frac{K_m}{\tau} - \frac{K_{m\ nom}}{K_m\ \tau_{nom}}\right] u_o\right]$$

where the suffix "nom" denotes a nominal value and u_o is the control contribution fo the PID/Sliding Mode linear component.

A similar procedure may be applied to modify a PID trajectory tracking controller into a PID/Sliding mode trajectory tracking controller (figure 3.2).

5.5 Application to real time parameter identification

Sliding mode techniques may be conveniently applied to real time parameter identification. To illustrate let a system be described by

$$\dot{x}_1 = x_2 \qquad a \triangleq \frac{1}{\tau}$$

$$\dot{x}_2 = ax_2 + bu \qquad b \triangleq \frac{K_m}{\tau}$$

with a and b unknown. Consider the problem of estimating a and b at time t from the knowledge of $u(s)$, $x_2(s)$, $t \in (0,\infty)$,

$s \in (-\infty, t]$

To solve this problem we first use sliding mode to get an estimate of $\dot{x}_2$. For this we consider the system

$$\dot{\hat{x}}_1 = \hat{x}_2$$

$$\dot{\hat{x}}_2 = \hat{a}(t)x_2(t)+\hat{b}(t)u(t)$$

$$+ \left[c|x_2(t)|+d|u(t)|\right] u^*(t)$$

where:

$\hat{a}(t)$ is the estimate of a; $\hat{b}(t)$ of b;

$u^* \triangleq -SIGN(slm)$; $slm \triangleq \hat{x}_2(t)-x_2(t)$;

$c>0$, $d>0$ are to be chosen so that $slm(t) \to 0$.

For $slm(t) \to 0$ we observe that

$$\begin{aligned}
\frac{d}{dt}\, slm(t)^2 &\triangleq \dot{slm}(t)\, slm(t) \\
&= (\dot{\hat{x}}_2-\dot{x}_2)slm(t) \\
&= (\hat{a}(t)-a)x_2\, slm + (\hat{b}(t)-b)u\, slm \\
&\quad - c|x_2||slm| - d|u||slm| \\
&< 0 \quad \text{if}
\end{aligned}$$

$$c > \max|\hat{a}-a| \quad , \; d > \max|\hat{b}-b| \; .$$

With $slm = 0$ one has

$$\hat{x}_1 = x_1 \qquad \dot{\hat{x}}_2 = \dot{x}_2 \quad \text{for almost every } t.$$

We can therefore estimate $\hat{a}(t)$, $\hat{b}(t)$ from $\dot{x}_2(\cdot)$, $x_2(\cdot)$, $u(\cdot)$.

For this we set up the following algorithm

$$\varepsilon_1(t) \stackrel{\Delta}{=} \dot{x}_2(t) - \hat{a}(t)x_2(t) - \hat{b}(t)u(t)$$

$$\varepsilon_2(t) \stackrel{\Delta}{=} \dot{x}_2(t-D) - \hat{a}(t)x_2(t-D) - \hat{b}(t)u(t-D)$$

$$\dot{\hat{a}}(t) = \varepsilon_1(t)x_2(t) + \varepsilon_2(t)x_2(t-D)$$

$$\dot{\hat{b}}(t) = \varepsilon_1(t)u(t) + \varepsilon_2(t)u(t-D)$$

for some $D > 0$. To convince ourselves that the algorithm converges, we consider the Lyapunov function

$$V = (a-\hat{a}(t))^2 + (b-\hat{b}(t))^2$$

and observe that

$$\begin{aligned}
\dot{V} &= -(a-\hat{a}(t))\dot{\hat{a}} - (b-\hat{b}(t))\dot{\hat{b}} \\
&= -(a-\hat{a}(t))\left[\varepsilon_1(t)x_2(t)+\varepsilon_2(t)x_2(t-D)\right] \\
&= (b-\hat{b}(t))\left[\varepsilon_1(t)u(t)+\varepsilon_2(t)u(t-D)\right] \\
&= -\left[(a-\hat{a}(t))\ x_2(t) + (b-\hat{b}(t))\ u(t-D)\right]^2 \\
&\quad - \left[(a-\hat{a}(t)x_2(t-D) + (b-\hat{b}(t))u(t-D)\right]^2
\end{aligned}$$

$$< 0 \quad \text{if} \quad \det \begin{bmatrix} x_2(t) & u(t) \\ x_2(t-D) & u(t-D) \end{bmatrix} \neq 0$$

The technique illustrated by this example may be readily generalized to the problem of identifying the A parameter matrix of

$$\dot{x} = A\ \varphi(x,t)$$

from the knowledge of $x(\cdot)$ and $\varphi(\cdot,\cdot)$. One first estimates $\dot{\hat{x}}$ by considering

$$\dot{\hat{x}}_i = \hat{A}_i\ \varphi(x,t) + \sum_{j=1}^{n} B_{ij}\ |\varphi_j(x,t)|u^*_j$$

with $u^*_j \triangleq -\mathrm{SIGN}(slm_j)$

$$slm_i \triangleq \hat{x}_i - x_i$$

and B_{ij} such that $slm_i \to 0$.
One then identifies $\hat{A}$ from $\dot{\hat{x}}$, x, and $\varphi(x,t)$. More on this in [Ut.1, p. 236-240] .

6. LEARNING CONTROL

6.1 Introduction

To introduce the concept of a learning control it is useful to consider a scenario whereby a robotic manipulator has to repetitively execute a given task; an optimal execution requires the manipulator links position to follow a predetermined trajectory; the presence of an unknown perturbation allows this trajectory to be attained only within a certain error. The problem is to exploit the knowledge of this error in correspondence with a given execution to complement the action of the available controller so as to come up with a smaller error at a subsequent execution (fig. 1.1).

6.2 Implementation in a computed torque scheme

To be more specific, let us introduce the following notations:

$q_D(\cdot)$, $q_k(\cdot)$: manipulator links desired and actual k-th execution position trajectory;

$e_k(\cdot)$: $= q_D(\cdot)-q_k(\cdot)$, trajectory error at the k-th execution;

$f(\cdot)$: the unknown input equivalent perturbation causing the error;

$d_k(\cdot)$: the acceleration equivalent action provided by the learning module in response to e_{k-1}.

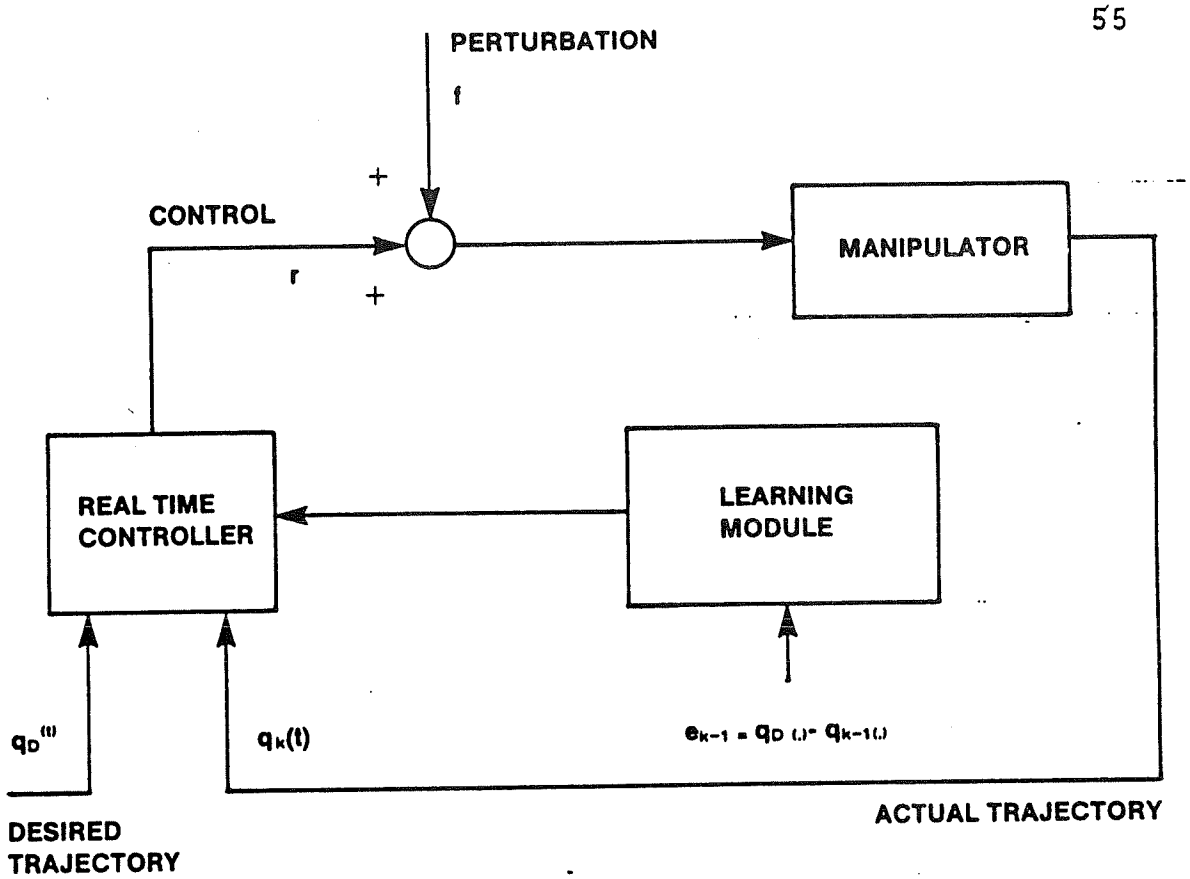


Figure 1.1: Learning control scenario.

Represent these functions as elements of $L_2\,[0,\infty)^n$.

Assuming the utilization of a precomputed torque controller, the manipulator dynamics at the k-th execution is represented by [Cr.2]

$$D_c\left[\ddot{q}_D + K_1 e_k + K_2 \dot{e}_k + d_k\right] = D\ddot{q} + f$$

where D_c and D represent computed and actual inertia matrices. Further, assuming $D_c = D$, it follows:

$$K_1 e_k + K_2 \dot{e}_k + \ddot{e}_k + d_k \stackrel{\Delta}{=} D^{-1} f \stackrel{\Delta}{=} d$$

hence

$$e_k(s) = H(s)\left[(d-d_k)(s)\right]$$

with

$$H(s) = \frac{1}{K_1+K_2 s+s^2}$$

Representing the learning module as described by

$$d_{k+1} = d_k + Qe_k$$

with $Q: L_2\left[0,\infty\right)^n \to L_2\left[0,\infty\right)^n$ a (not necessarily casual) operator to be determined, one has

$$d_{k+1} = d_k + QH\left[d-d_k\right] .$$

For the intended objective to be attained it is then sufficient to require $|d-d_k| \to 0$, hence to choose Q so that $|I-QH| < 1$. A potential (non casual) choice of such a Q is for example represented by $Q=\alpha H^*$ with H^* the adjoint of H and constant α such that $|H| < 1/\alpha$.

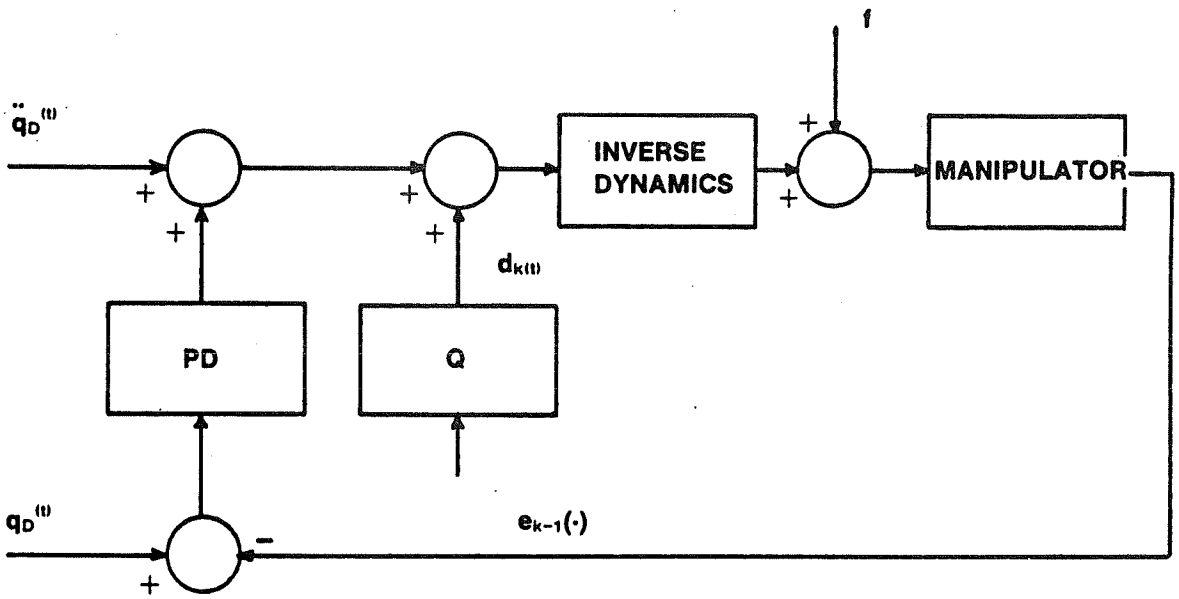


Figure 2.1: Learning control applied to a computed torque controller.

BIBLIOGRAPHY

Cr. 1 Craig, J.J., Introductions to Robotics, Addison Wesley 1986.

As. 1 Asada, H., Slotine, J-J E., Robot Analysis and Control, Wiley 1986.

Fu. 1 Fu, K.S., Gonzales, R.C., Lee, CS.G., Robotics: Control, Sensing, Vision, and Artificial Intelligence, McGraw Hill Book Inc. 1987.

Br. 1 Brady, M., & Co., Robot Motion Planning and Control, MIT Press 1984.

Ut. 1 Utkin, V.I.,Sliding Modes and their Application in Variable Structure Systems, Mir Publisher, Moscow, 1978.

It. 1 Itkins,I.,Control Systems of Variable Structure, John Wiley & Sons Inc., New York, 1976.

La. 1 Landau,Yoan D., Adaptative control, University of Southern California, Dekker 1979.

As. 1 Astrom, K.J. & Co., Theory and Application of Self-Tuning Regulators, Automatica, 13,pp. 457-476, 1977.

En. 1 Engelberger, J.F., Robotics in Practice, AMA-COM, 1980.

Pa. 1 Paul, R.P., Robot Manipulators: Mathematics, Programming and Control, The MIT Press 1981.

Cr. 2 Craig, J.J., Adaptive Control of Mechanical Manipulators, Addison-Wesley 1988.

Le. 1 Lee, C.S.G., Gonzales, R.C., Fu, K.S., Tutorial on Robotics, IEEE Computer Society Press, 1985.

Ecole Polytechnique technical reports

RMDS.1 De Santis, R.M., Théorie des systènes de commande linéaires, EPM Notes de cours, 1986.

RMDS.2 De Santis, R.M., Eléments fondamentaux de la robotique: cinématique, dynamique et commande, EPM Notes de cours, 1986.

RMDS.3 Serfass, C., De Santis, R.M., Contrôleur à structure variable pour une articulation robotique élastique, EPM/RT-87/19, Mai 1987.

RMDS.4 Nguyen, A.T., De Santis, R.M., Etude de Sensibilité du comportement dynamique d'un manipulateur robotique, EPM/RT-87/2, janvier 1987.

RMDS.5 Quintal, M., De Santis, R.M., Commande adaptative d'une articulation robotique avec la méthode du modèle de référence, EPM/RT-88/15, Mai 1988.

RMDS.6 Souphandavong, P., De Santis, R.M., Exécution robotisée du jeu de la Tour d'Hanoi, EMP/RT-87/15, Mai 1987.

RMDS.7 De Santis, R.M., Commande adaptative d'une articulation robotique avec la méthode du Self-Tuning, EPM/RT-88/XX (en préparation).

RMDS.8 De Santis, R.M., PID/Sliding Mode Control of a robotic link, EMP/RT-88/XX (en préparation).

APPENDIX A: LABORATORY SIMULATION EXPERIMENTS

Among the variety of techniques applicable to the control of a robotic link we have seen classical PID, model reference and self-tuning adaptive control, sliding mode control. Our exposition, complemented with the cited bibliography should help the student to familiarize himself with adavantages and shortcomings of each one of these approaches, efficient computational and numerical design procedures, application modalities, and so on.

All this notwithstanding, a number of factors still remain that make these approaches difficult to put into perspective: the complexity of the often nonlinear dynamics, the variety of design criteria, the heuristic nature characterizing some of the design parameters choices, the sometime hard to establish connection between theoretical and physical implications of the available results.

In what follows we suggest a number of questions which may be advantageously clarified via hands-on simulation experiments on a personal computer. These experiments may be carried out by using the simulators described in appendix B.

Simulation Experiment *1: APPLICATION OF A CLASSICAL PID CONTROLLER TO A ROBOTIC LINK

The simulators used in this experiment are POSSPEED and SIMNOG. The questions under investigation are the following:

How well does a PID perform? How to characterize performance? What is the function of each of its components? How to tune the gains? How do gain variations influence performance? What is the influence of other controller parameters? (sampling period, quantization levels, measurement noise, transducer gains, maximum control levels, delay in the control, ...).
What is the influence of the various kinematic and mass manipulator parameters (masses of the links, inertias, lengths, working configuration, viscosity, ...)?
How effective is a PID in neutralizing intercouplings, centripetal and Coriolis forces?
What are a PID most advantageous/disadvantageous features?

Simulation experiment *2: APPLICATION OF MODEL REFERENCE AND SELF-TUNING ADAPTIVE CONTROLLERS TO A ROBOTIC LINK

This experiment uses the simulators MRAC and SELFTUNE. The objective is to analyze via simulation the following type of questions:
Why, how, and with what success an adaptive controller of the model reference or self-tuning type?
What is the comparative difference between the two?
What is the influence of the gain adaptive module parameters?
What are the control strategies?
What "ad hoc" modifications of the off-the-shelf algorithms might be required?

What is the sensitivity to external perturbations, the response time, required control level, robustness, quality of performance?

Simulation experiment *3: SLIDING MODE CONTROL OF A ROBOTIC LINK

The simulator used in this experiment is ROBEL; the objective is to analyze the following type of questions:
Why, how and with what success a sliding mode controller?
Importance of the various parameters of the controller?
Basic structure and required algorithms for the controller?
Comparisons with PID and linear state regulator?
Sensitivity to external perturbations, response time, control level requirements, robustness, reliability, stability, quality of performance.

APPENDIX B: LABORATORY SIMULATORS

In what follows we give a brief description of the simulators used in the laboratory experiments mentioned in Appendix A.

POSSPEED (POSition SPEED control simulator)

POSSPEED simulates a link angular position controller (fig. 1.3.2); the mathematical model is in fig. 1.1; the non-linear element can be anyone of the types in fig. 1.2. The gain values may be computed automatically by POSSPEED so as to position the poles as a function of the damping factor ξ and of the A,B,C acceleration factors indicated in fig. 1.3. Other simulation possibilities include: various PID indutrial configurations (threshold, saturation on the integral component), influence of a delay in the control, external perturbation influence, speed rather than position control.

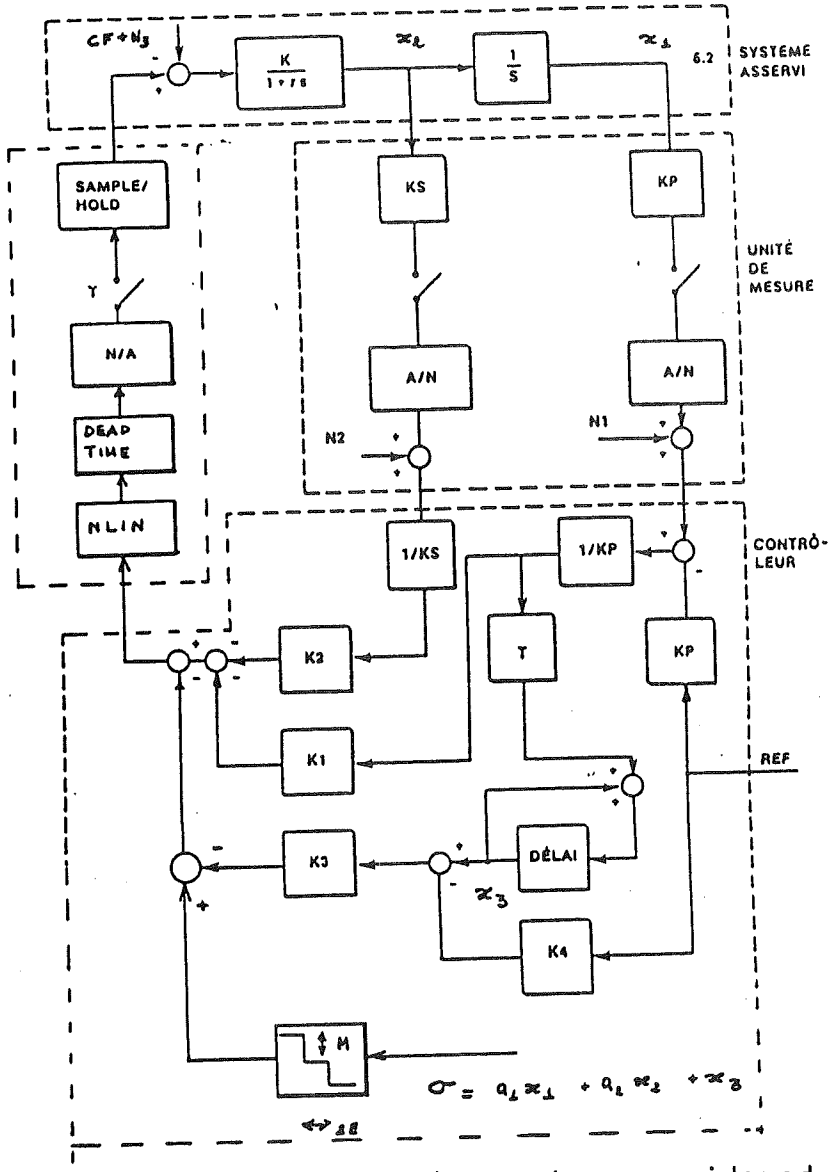


Figure 1.1: Bloc diagram of the system considered in POSSPEED

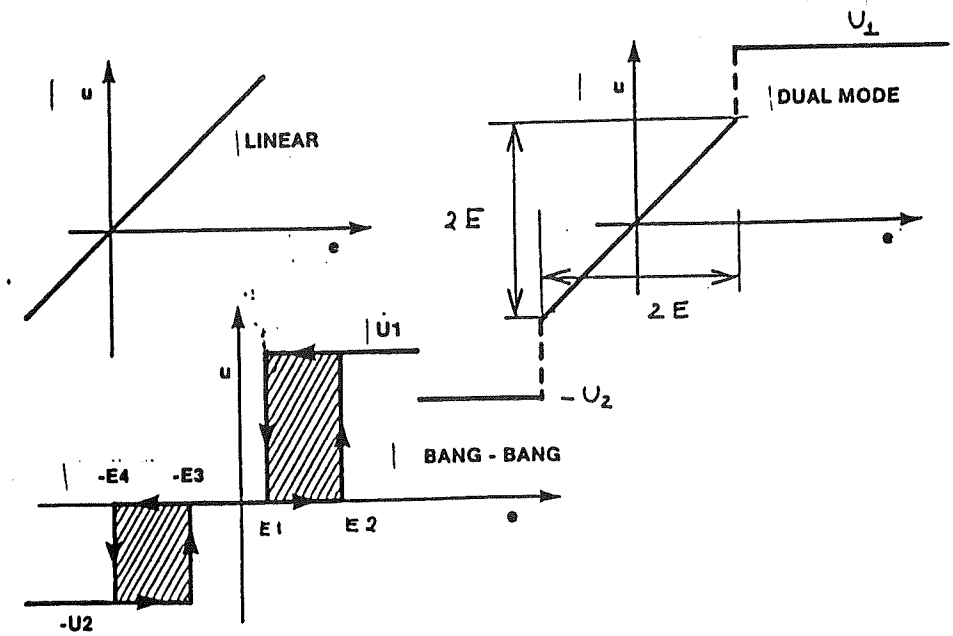
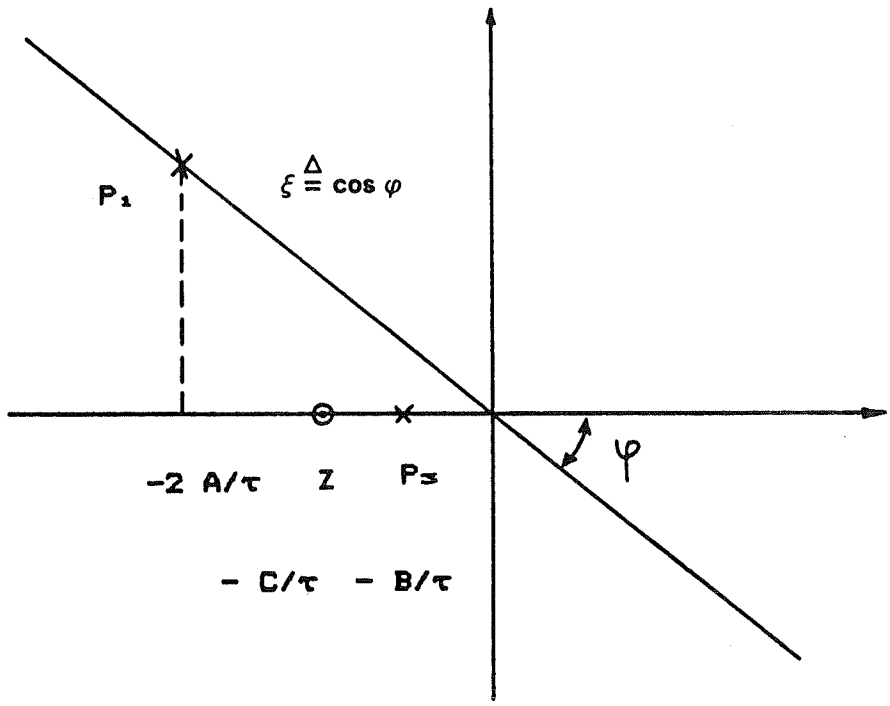


Figure 1.2: The nonlinear element in POSSSPEED



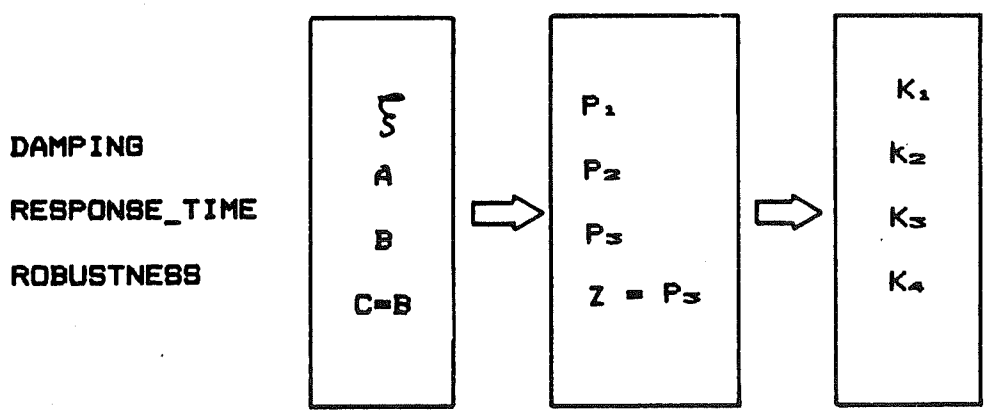


Figure 1.3: Gain adjustment in POSSPEED

SIMNOG (SIMulation with NO Gravity)

SIMNOG simulates the control system in fig. 2.1; the manipulator dynamical model is as in fig. 2.2; the controller is made of two decentralized PID components (fig. 3.2.1), plus a feedforward component compensating for gravity; the nonlinear element is identical to that considered in POSSPEED. More details can be found in [RMDS.4].

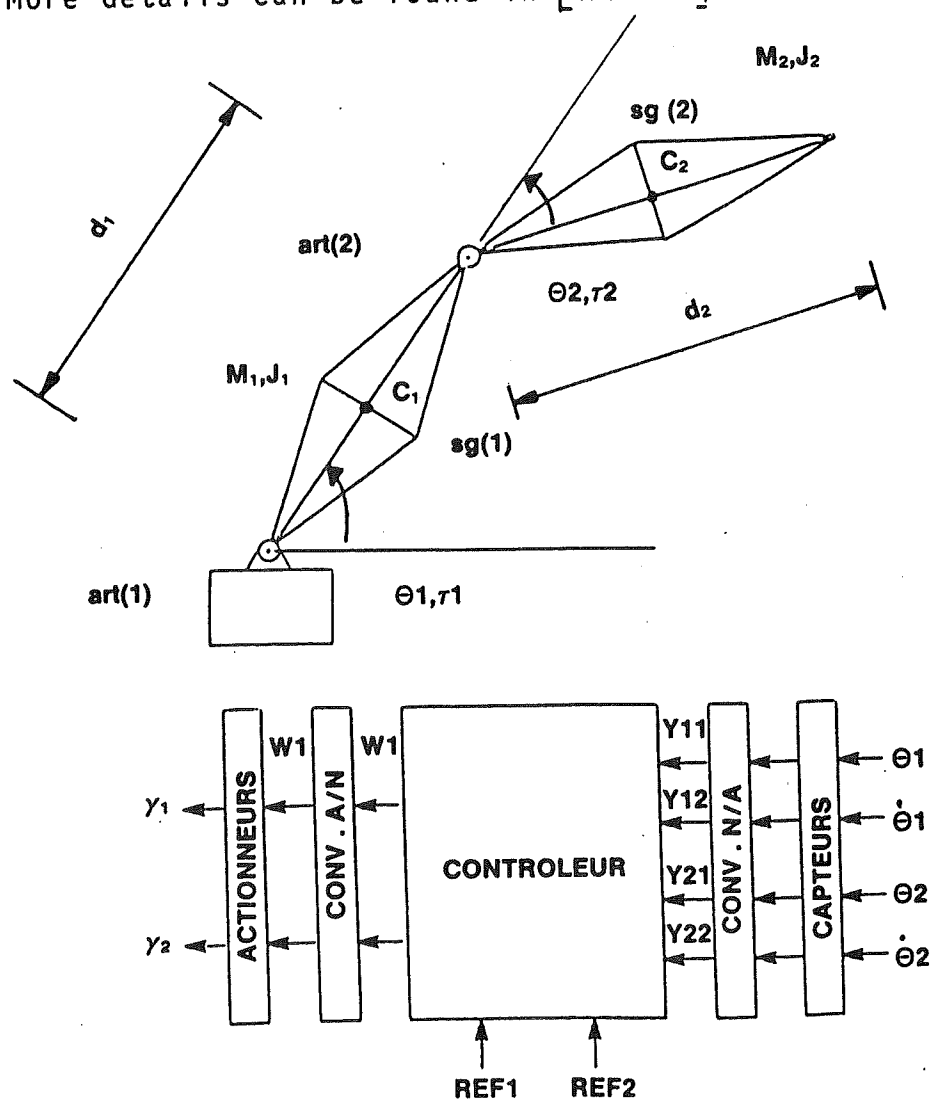


Figure 2.1: The robotic system considered in SIMNOG

$$x11 = \theta 1$$

$$x12 = \dot{\theta}1$$

$$x13 = \int (\theta 1 - REF1)$$

$$x21 = \theta 2$$

$$x22 = \dot{\theta}2$$

$$x23 = \int (\theta 2 - REF2)$$

$$D11 = (1/4)m1\ d1^2 + m2[d1^2 + (1/4)d2^2 + d1\ d2\ \cos x21] + I1 + I2$$

$$D12 = m2(d2^2/4 + d1\ d2\ \cos x21/2) + I2$$

$$D21 = m2(d2^2/4 + d1\ d2\ \cos x21/2) + I2$$

$$D22 = (1/4)m2\ d2^2 + I2$$

$$D112 = -\ m2\ d1\ d2\ \sin x21$$

$$D122 = -\ (1/2)m2\ d1\ d2\ \sin x21$$

$$D211 = +\ (1/2)m2\ d1\ d2\ \sin x21$$

$$a = 1\ /\ (D11\ D22 - D12\ D21)$$

$$T1' = T1 + T1g + T1v$$

$$T2' = T2 + T2g + T2v$$

$$T1 = -K11(x11-REF1)-K12\ x12-K13\ x13+K13\ K14\ REF1$$

$$T1g = -g[(m2 + (1/2)m1)d1\ \cos x11 + (1/2)m2\ d2\ \cos(x11 + x21)]$$

$$T1v = -\ f1\ x12 + f2\ x22$$

$$T2 = -K21(x21-REF2)-K22\ x22-K23\ x23+K23\ K24\ REF2$$

$$T2g = -g[(1/2)m2\ d2\ \cos(x11 + x21)]$$

$$T2v = -\ f2\ x22$$

$$\dot{x}11 = x12$$

$$\dot{x}12 = a\ D22(T1' - D112\ x12\ x22 - D122\ x22^2) - a\ D12(T2' - D211\ x12^2)$$

$$\dot{x}13 = x11 - REF1$$

$$\dot{x}21 = x22$$

$$\dot{x}22 = -\ a\ D21(T1' - D112\ x12\ x22 - D122\ x22^2) + a\ D11(T2' - D211\ x12^2)$$

$$\dot{x}23 = x21 - REF2$$

Figure 2.2: The manipulator dynamical model considered by SIMNOG

MRAC (Model Reference Adaptive Control)

MRAC simulates the dynamic behavior of an adaptive control system of the Model Reference type (fig. 4.2.1).
The controller/plant part is made of a PD position servo (fig. B.3.1); the reference model is as in fig. B.3.2; the gain adaptation module responds to the scheme in fig. B.3.3.
The simulation allows one to analyze the system behavior with and without adaptation as a function of the following parameters: plant time constant and static gain, initial conditions, reference model dynamic parameters, error weighting coefficients, adaptation module gains. More details can be found in [RMDS.5] .

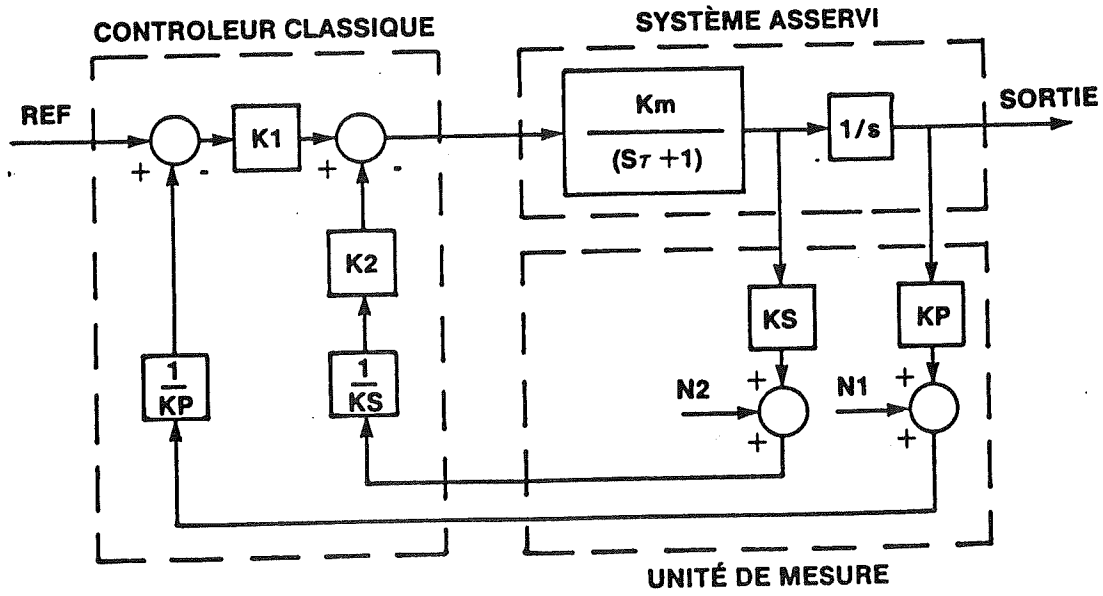


Figure 3.1: Controller/plant ensemble considered in MRAC

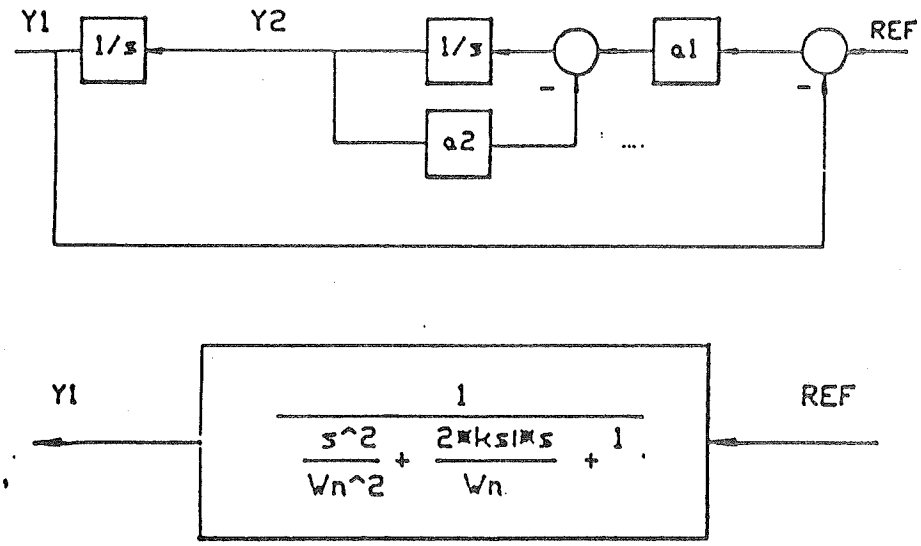


Figure 3.2: The reference model in MRAC

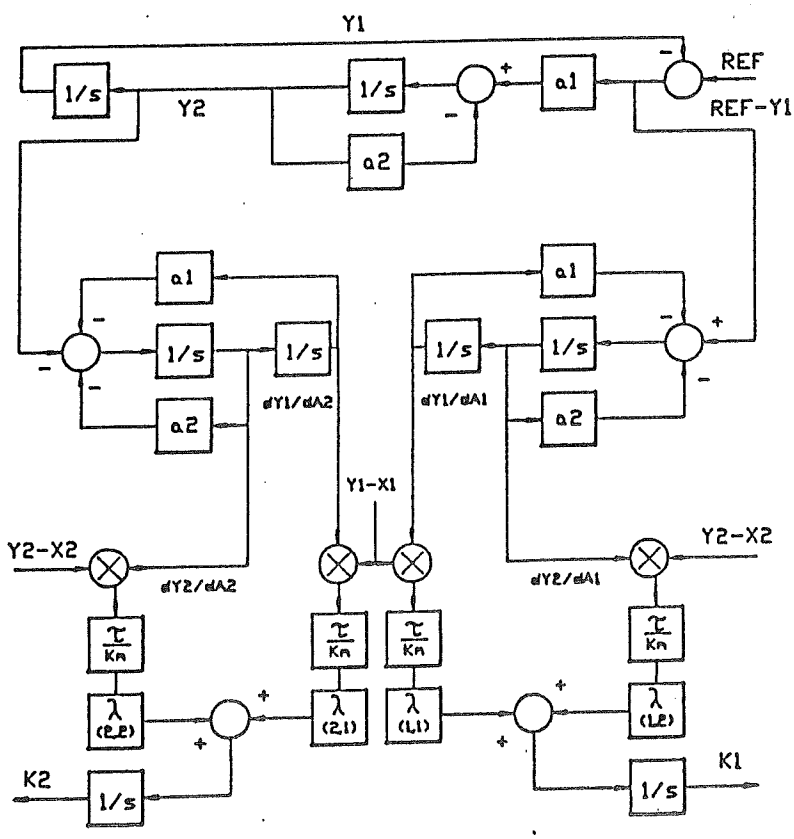
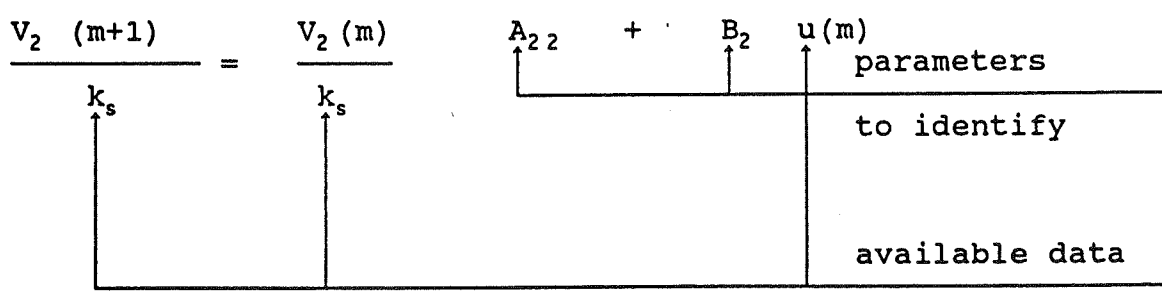


Figure 3.3: Gain adaptation module in MRAC

SELFTUNE (SELFTUNing Controller)

SELFTUNE simulates the dynamic behavior of an adaptive control system of the self-tuning type (see fig. 4.3.1). The controller-plant part is identical to that considered in fig. 1.1. The parameter estimator is of the recursive least square type; it offers the choice to include or not a speed measurement; it uses the formulas given in section 4.3 with the symbol notation illustrated in fig. 4.1 or 4.2. More details are to be found in [RMDS.7] .



$$y(m+1) \overset{\Delta}{=} \frac{V_2\ (m+1)}{K_s} \qquad x(m) \overset{\Delta}{=} \begin{bmatrix} y(m) \\ u(m) \end{bmatrix}$$

$$\hat{\Theta}\ (m) \overset{\Delta}{=} \begin{bmatrix} A_{22} \\ B_2 \end{bmatrix}$$

Figure 4.1: Correspondence of notations in the estimator of SELFTUNE (a speed measurement is available)

$$x_1(i+1) = x_1(i) + A_{12}\, x_2(i) + B_1\, u(i)$$

$$x_2(i-1) = \frac{1}{A_{12}}\left[x_1(i) - x_1(i-1)\right] - \frac{B_1}{A_{12}}\, u(i-1)$$

$$x_2(i-1) = A_{22}\, x_2(i-2) + B_2\, u(i-2)$$

$$\frac{1}{A_{12}}\left[x_1(i) - x_1(i-1)\right] - \frac{B_1}{A_{12}}\, u(i-1) = \frac{A_{22}}{A_{12}}\left[x_1(i-1) - x_1(i-2)\right]$$

$$- \frac{A_{22}}{A_{12}}\, B_1\, u(i-2) = B_2\, u(i-2)$$

$$\frac{V_1(i) - V_1(i-1)}{K_p} = A_{22}\left[\frac{V_1(i-1) - V_1(i-2)}{K_p}\right] + B_1\, u(i-1) + (B_2 A_{12} - B_1 A_{22})\, u(i-2)$$

parameters to identify

available data

$$y(m+1) \overset{\Delta}{=} \left[V_1(m+1) - V_1(m)\right] / K_p$$

$$x(m) \overset{\Delta}{=} \begin{bmatrix} y(m) \\ u(m) \\ u(m-1) \end{bmatrix} \qquad \hat{\Theta} \overset{\Delta}{=} \begin{bmatrix} A_{22} \\ B_1 \\ B_2\, A_{22} - B_{22} \end{bmatrix}$$

Figure 4.2: Correspondence of notations in the estimator of SELFTUNE (a speed measurement is not available)

ROBEL (Control of a ROBotic link with an ELastic Transmission)

ROBEL simulates the dynamic behavior of a robotic link position controller characterized by an elastic transmission (fig. 5.1). The controller configuration can be chosen as one of the following:

i. classical PID (fig. 3.2.1)
ii. PID/sliding mode (fig. 4.3.1)
iii) classical linear state regulator (fig. 5.2)
iv) nonlinear sliding mode state regulator (fig. 5.3).

More details are to be found in [RMDS.3] .

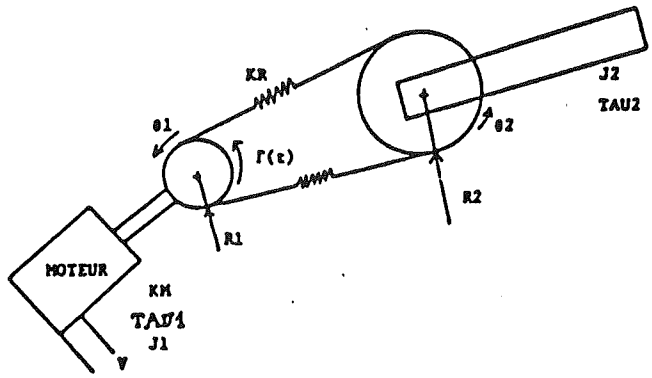
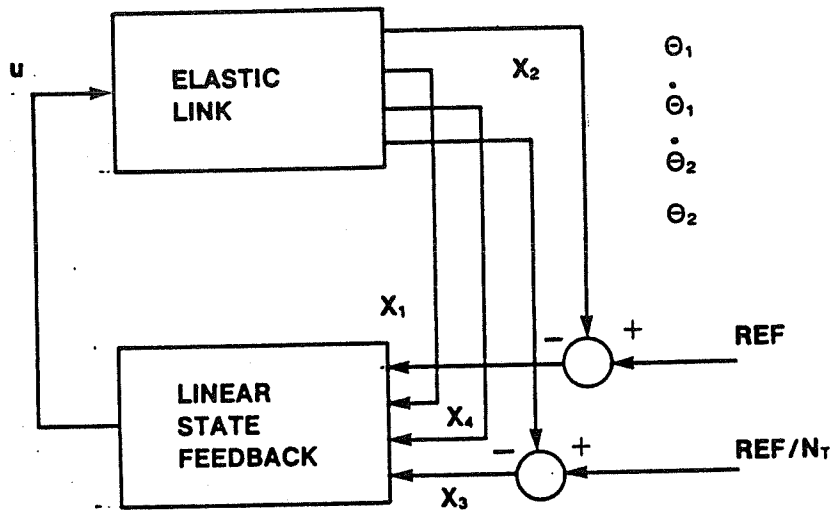


Figure 5.1: Elastic link considered in ROBEL



$$u = \begin{bmatrix} K_1 & K_2 & K_3 & K_4 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

Figure 5.2: Linear state regulator used in ROBEL

$$u = \sum_{i=1}^{n} \psi_i * X_i$$

$$\psi_i = \begin{cases} \alpha_i & \text{si} \quad x_i * \sigma > 0 \\ \beta_i & \text{si} \quad x_i * \sigma < 0 \end{cases} \qquad i = 1..n$$

$$\sigma \triangleq C' * X$$

$$C \triangleq [c1 \quad \ldots\ldots \quad cn]$$

$$\alpha_i < l_i$$

$$\beta_i > l_i$$

$$l_i \triangleq - \frac{A_i^T * C}{B^T * C}$$

```
c1 = -(p1+p2+p3)-(1/τ2)

c2 = 1

c3 = -(J2/(KR*R2²))*(p1*p2*p3) - c1

c4 = (J2/(KR*R2²))*(p3*p1+p3*p2+p1*p2)
                -1-(c1*J2)/(c2*KR*R2²)


l1 = (τ1/Km)*((KR*R1²/J1)-(c4*KR*R2²/J2))

l2 = (τ1/Km)*((1/τ1)-c1)

l3 = (τ1/Km)*(c4*(KR*R2²/J2)-(KR*R1²/J1))

l4 = (τ1/Km)*((c4/τ2)-c3)
```

Figure 5.3: Variable structure controller used in ROBEL

EPM/RT-88/15

COMMANDE ADAPTATIVE D'UNE ARTICULATION ROBOTIQUE AVEC LA METHODE DU MODELE DE REFERENCE

Par

Marc Quintal, Ing.

Projet dirigé par

Dr. Romano M. De Santis

Département de Génie Electrique

Ecole Polytechnique de Montréal

Avril 1988

SOMMAIRE

Ce rapport présente MRAC, un progiciel de simulation infographique qui permet d'analyser le comportement dynamique d'une articulation robotique avec commande adaptative. La commande adaptative considérée est axée sur l'approche qui va sous le nom de "Modèle de Référence", (Model Reference Adaptative Control, MRAC).

MRAC est écrit en Turbo Pascal; ses modules utilisent les librairies "Turbo-Numerical Methods", "Turbo-Graphix" et "Turbo-Toolbox"; il est installé sur un micro ordinateur du type IBM-PC classique.

REMERCIEMENTS

Ce projet a été développé dans le cadre du projet FCAR-CRP-378-76, "Analyse et Commande des Processus Industriels par Ordinateur: Application au Réseaux Electrique", financé par le Ministère de l'Education du Québec.

L'auteur tient à exprimer vivement sa reconnaissance envers son directeur de recherche, Dr. Romano M. De Santis, professeur du département de Génie Electrique, section Automatique, pour sa collaboration et ses précieux conseils dans la réalisation de ce travail.

Il lui est agréable également de mentionner la compréhension et encouragements des membres de la famille et des ami(es) tout au long de ce travail.

TABLE DES MATIERES

SOMMAIRE iv

REMERCIEMENT v

TABLE DES MATIERES vi

LISTE DES FIGURES vii

1. Introduction 1

2. L'idée de base 3

3. L'application à une articulation robotique 8

3.1 Le système asservi 8

3.2 Le modèle de référence 10

3.3 La matrice des fonctions de sensibilité 11

3.4 Loi de variation des gains 13

4. Le progiciel MRAC 16

5. Un exemple d'application 18

CONCLUSION 20

BIBLIOGRAPHIE 21

APPENDICE: Listage de MRAC 22

LISTE DES FIGURES

2.1 Schéma d'une commande adaptative avec modèle de référence 3

3.1 Le système en étude 8

3.2 Le modèle de référence 10

3.3 Le générateur des fonctions de sensibilité 12

3.4 Loi de variation des gains 14

3.5 Le module d'adaptation des gains 15

4.1 La structure du progiciel MRAC 16

5.1 Un exemple d'application: Données à l'entrée 18

5.2 Un exemple d'application: Dynamique du système avec et sans adaptation 19

1. Introduction

Dans un grand nombre de systèmes de contrôle il est important d'atteindre une opération qui soit autant que possible insensible aux variations des paramètres. Avec les contrôleurs actuellement disponibles, cette exigence est souvent difficile à satisfaire lorsque le système asservi subit de larges variations de paramètres. De telles variations sont assez fréquentes en robotique par exemple, où le moment d'inertie effectif d'une articulation robotique peut varier d'une façon importante en fonction de la configuration du robot. Une approche susceptible d'améliorer cette situation concerne l'utilisation de contrôleurs adaptatifs [2, 5]: les effets engendrés par une variation des paramètres du système asservi sont réduits par une adaptation des gains du contrôleurs.

Les références citées illustrent bien les avantages d'une telle approche; elles présentent de nombreuses procédures conceptuelles et des algorithmes de calcul efficaces; elles donnent plusieurs exemples d'application. Malgré ces éléments, plusieurs facteurs font que la conception d'un contrôleur adaptatif reste loin d'être routinière. Il suffit de penser à la complexité des phénomèmes en jeu, au manque de critères de conception précis, à la nécessité de compromis dans le choix de nombreux paramètres, etc... De plus, il est difficile d'obtenir une appréciation réaliste des difficultés rencontrées dans la conception d'un tel contrôleur à partir de résultats théoriques.

Dans ce qui suit nous présentons un progiciel de simulation dynamique qui permet de surmonter au moins en partie ces difficultés. En ce faisant nous poursuivons deux objectifs.

D'une part, l'illustration, par voie d'expérience de simulation, d'un grand nombre des aspects saillants de la commande adaptative: le rôle des fonctions de sensibilités, le choix des facteurs de pondération dans la détermination du critère de performance, l'influence des gains du module d'adaptation, etc... D'une autre part, la détermination, par voie d'un exemple de réalisation, des propriétés, des techniques et des modalités d'opération qui devraient caractériser la conception d'un progiciel de simulation d'un tel système de contrôle: rapidité d'exécution, flexibilité, modularité, graphisme interactif, interface amicale, etc...

Nous nous limitons à considérer l'application de la théorie de la comande adaptative avec l'approche "Modèle de Référence" à la commande d'une articulation robotique [1]. Le progiciel de simulation dynamique qui en résulte, MRAC, partage beaucoup des éléments qui caractérisent des progiciels du même type développés au sein de notre équipe de recherche [6-10]. Il est exécutable sur un micro-ordinateur compatible avec l'IBM-PC. Il est écrit en langage Turbo Pascal; sa structure est modulaire et axée sur une utilisation des librairies commerciales "Turbo-Numerical Methods" et "Turbo-Graphix Tools" [3, 4].

2. L'idée de base

Tel que suggéré par le schéma de la figure 2.1, l'opération d'un système de contrôle adaptatif avec modèle de référence repose sur les principes suivants [2]:

- le système est constitué de trois blocs: un système asservi, un contrôleur classique, un module d'adaptation des gains;
- lorsque les paramètres du système asservi et les gains du contrôleur ont les valeurs nominales, alors le comportement de la sortie du système correspond à celui de la sortie désirée qui est procurée par un modèle de référence;

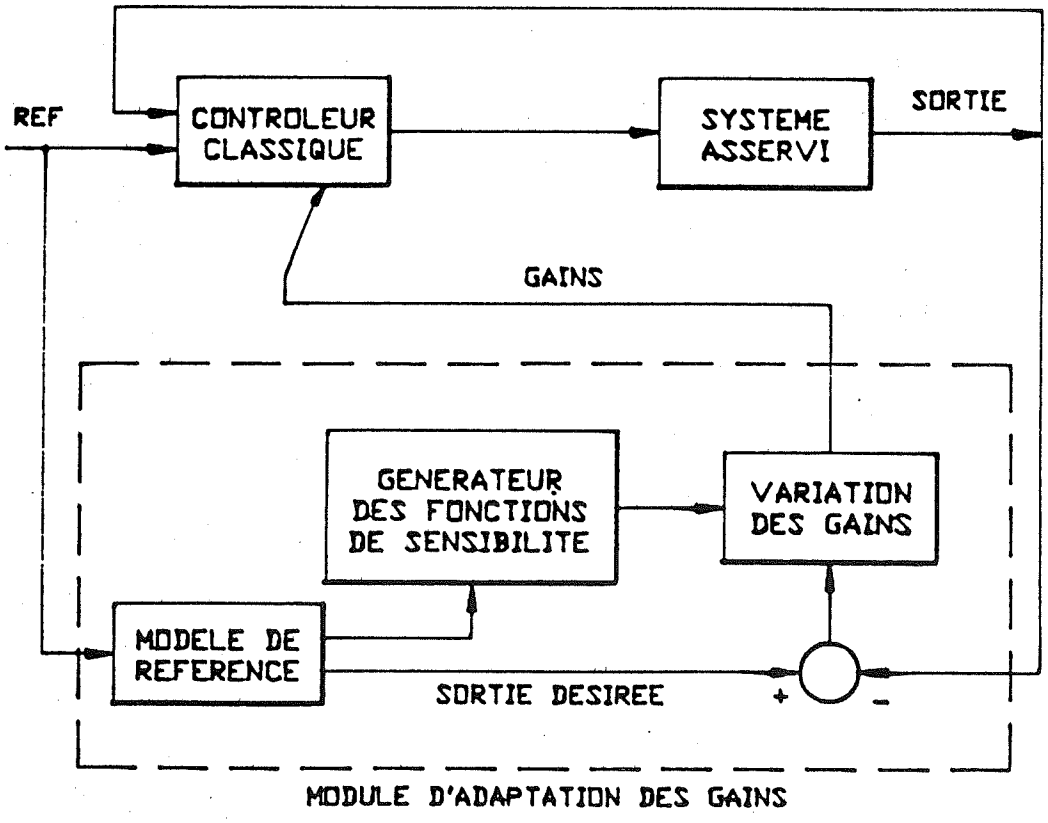


Figure 2.1: Schéma d'une commande adaptative avec modèle de référence.

- lorsque les paramètres du système asservi s'écartent de leurs valeurs nominales, le comportement de la sortie du système asservi s'écarte de la sortie désirée; le module d'adaptation varie alors les gains du contrôleur de façon à réduire autant que possible cet écart.

Afin de représenter les idées à la base des algorithmes utilisés par un tel système, il est utile de représenter sa dynamique globale avec une équation du type suivant (2.1):

$$\dot{\underline{x}} = f_s(\underline{x},\underline{\alpha},\underline{r},\underline{d}) \qquad (2.1)$$

où:

f_s : fonction du système bouclé,

$\underline{x}$: états du système,

$\underline{\alpha}$: paramètres variables,

$\underline{r}$: entrée du système,

$\underline{d}$: perturbation du système.

La dynamique du modèle de référence est représentée avec une équation du type suivant (2.2):

$$\dot{\underline{y}} = f_m(\underline{y},\underline{a},\underline{r}) \qquad (2.2)$$

où:

f_m : fonction du modèle de référence,

$\underline{y}$: états du modèle,

$\underline{a}$: valeurs nominales de $\underline{\alpha}$,

$\underline{r}$: entrée du système.

Les paramètres $\underline{\alpha}$ sont une fonction à la fois des valeurs des gains du contrôleurs et d'un certain nombre de paramètres du système asservi qui peuvent varier d'une façon inprévisible. Le principe à la base du MRAC est de compenser ces variations imprévisibles avec des variations appropriées des valeurs des gains. Pour ce faire, l'on essaie de minimiser l'erreur:

$$e(t) \equiv y(t) - x(t) \tag{2.3}$$

Afin de pouvoir mieux quantifier l'erreur, nous considérons la fonction:

$$V(e) \equiv \frac{1}{2} \underline{e}^T Q \underline{e} \tag{2.4}$$

où Q est une matrice symétrique qui permet de pondérer les différentes composantes de l'erreur. Puisque V(e) est d'autant plus grande que l'erreur est grande, sa minimisation nous permettra d'obtenir la variation des valeurs de $\underline{\alpha}$ qui fera que la sortie du système asservi converge autant que possible vers la sortie du modèle de référence. A partir de cette variation, nous pouvons ensuite calculer la variation requise pour les gains du contrôleurs.

La minimisation de V(e) est effectuée en utilisant la méthode du Gradient [2]. Cette approche nous conduit à considérer l'équation:

$$\dot{\underline{\alpha}} = - \Gamma \frac{\delta V}{\delta \alpha} = - \Gamma \frac{\delta \underline{e}^T}{\delta \underline{\alpha}} Q \underline{e}(t) \tag{2.5}$$

où:

$\dfrac{\delta \underline{e}}{\delta \underline{\alpha}}$: matrice de sensibilité,

Q : matrice des coefficients de pondération,

Γ : matrice des facteurs d'accélération.

Cependant, la dérivée partielle $\delta \underline{e}/\delta \underline{\alpha}$ ne peut en général être calculée car elle demande de connaître la variation des états du système en fonction de ses paramètres variables. Nous devons donc supposer que cette variation sera convenablement exprimée par la dérivée partielle des états de notre modèle de référence en fonction de ses paramètres nominaux. Cette dernière approximation se traduit par l'équation (2.6):

$$\frac{\delta \underline{e}}{\delta \underline{\alpha}} = \frac{\delta(\underline{y} - \underline{x})}{\delta \underline{\alpha}} = - \frac{\delta \underline{x}}{\delta \underline{\alpha}} \approx - \frac{\delta \underline{y}}{\delta \underline{a}} \qquad (2.6)$$

où:

$\dfrac{\delta \underline{y}}{\delta \underline{a}}$: matrice de sensibilité donnée de la solution de la dérivéé partielle de la solution de l'équation (2.2) par rapport aux paramètres $\underline{a}$.

En dérivant l'équation (2.2) par rapport à $\underline{a}$, nous obtenons:

$$\frac{\delta \dot{\underline{y}}}{\delta \underline{a}} = \frac{\delta fm(\underline{y}, \underline{a}, \underline{r})}{\delta \underline{y}} \cdot \frac{\delta \underline{y}}{\delta \underline{a}} + \frac{\delta fm(\underline{y}, \underline{a}, \underline{r})}{\delta \underline{a}} \qquad (2.7)$$

Nous pouvons maintenant réécrire la loi d'adaptation (2.5) sous sa forme finale:

$$\dot{\underline{\alpha}} = - \Gamma \frac{\delta \underline{y}^T}{\delta \underline{\alpha}} Q \; \underline{e}(t) \qquad (2.8)$$

3. L'application à une articulation robotique

Dans ce chapître nous appliquons les idées vues à la section précédente à la commande d'une articulation robotique comportant un contrôleur de type PD.

3.1. Système asservi

Le schéma du système en étude est indiqué à la figure (3.1).

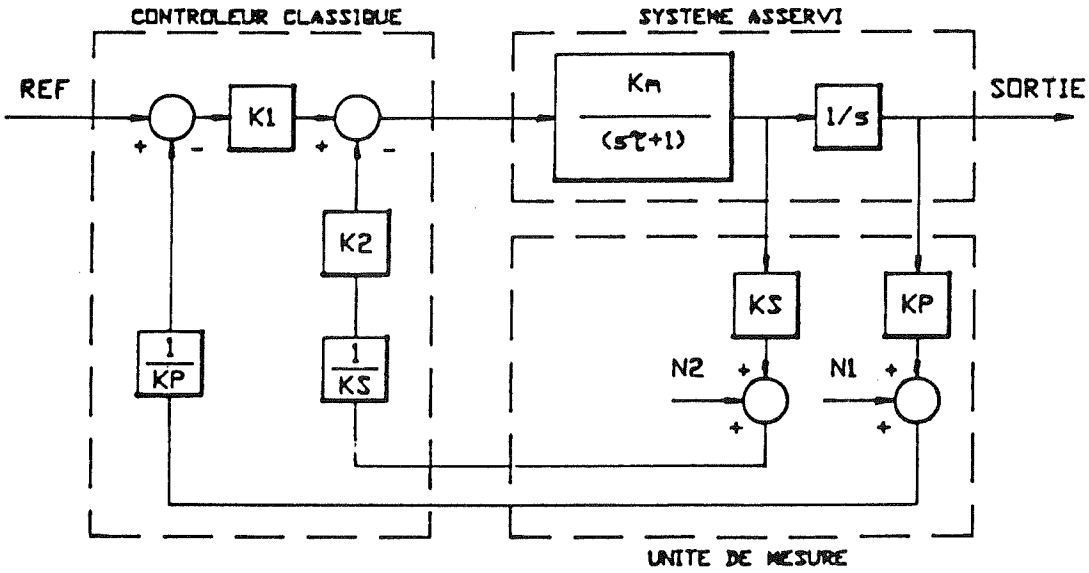


Figure 3.1: Le système en étude.

Pour un tel système, l'équation (2.1) devient:

$$\dot{x}_1 = x_2 \qquad (3.1)$$

$$\dot{x}_2 = -\frac{Km\ K1}{\tau}[x_1 - REF] - \left[\frac{Km\ K2}{\tau} + \frac{1}{\tau}\right] x_2 + \frac{Km\ K1}{KP\ \tau} N_1 + \frac{Km\ K2}{KS\ \tau} N_2$$

avec le vecteur $\underline{\alpha}$ décrit par:

$$\alpha_1 = \frac{Km\ K1}{\tau} \qquad (3.2)$$

$$\alpha_2 = \frac{1 + Km\ K2}{\tau}$$

3.2 Modèle de référence

En considérant le modèle de référence à la figure (3.2), l'équation (2.1) devient:

$$\dot{y}_1 = y_2 \qquad (3.3)$$

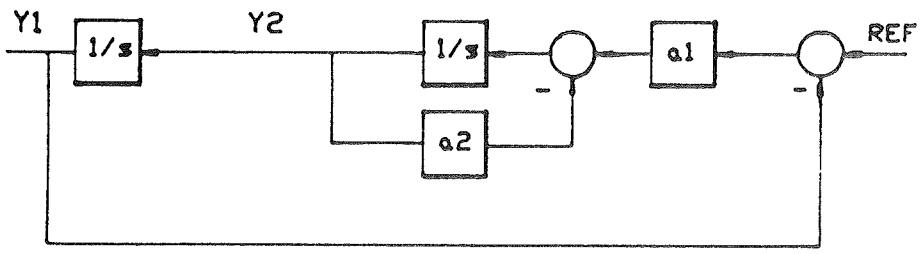
$$\dot{y}_2 = - a_1 \, [y1-REF] - a_2 \, y2$$

avec:

$$a_1 = Wn^2$$

$$a_2 = 2 * ksi * Wn$$

où Wn représente la fréquence naturelle et Ksi le facteur d'amortissement.



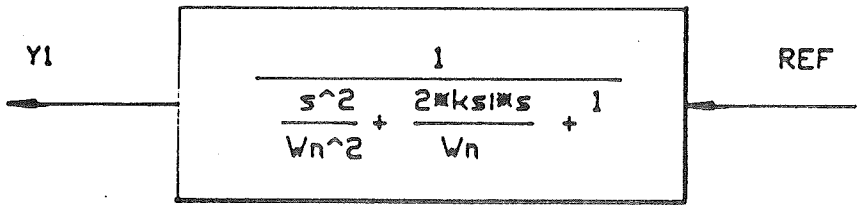


Figure 3.2: Le modèle de référence.

3.3 La matrice des fonctions de sensibilité

La matrice de sensibilité est représentée par:

$$\frac{\delta \underline{e}}{\delta \underline{a}} \approx \frac{\delta \underline{y}}{\delta \underline{a}} = \begin{bmatrix} \dfrac{\delta y_1}{\delta a_1} & \dfrac{\delta y_1}{\delta a_2} \\ \\ \dfrac{\delta y_2}{\delta a_1} & \dfrac{\delta y_2}{\delta a_2} \end{bmatrix} \qquad (3.4)$$

Elle peut être calculée en utilisant le modèle dynamique suivant:

$$\dot{\frac{\delta y_1}{\delta a_1}} = \frac{\delta y_2}{\delta a_1} \qquad (3.5)$$

$$\dot{\frac{\delta y_2}{\delta a_1}} = - a_1 \frac{\delta y_1}{\delta a_1} - a_2 \frac{\delta y_2}{\delta a_1} - [\, y_1 - REF \,]$$

$$\dot{\frac{\delta y_1}{\delta a_2}} = \frac{\delta y_2}{\delta a_2}$$

$$\dot{\frac{\delta y_2}{\delta a_2}} = - a_1 \frac{\delta y_1}{\delta a_2} - a_2 \frac{\delta y_2}{\delta a_2} - [\, y_2 \,]$$

Le schéma bloc de ce modèle dynamique peut être retrouvé à la figure (3.3).

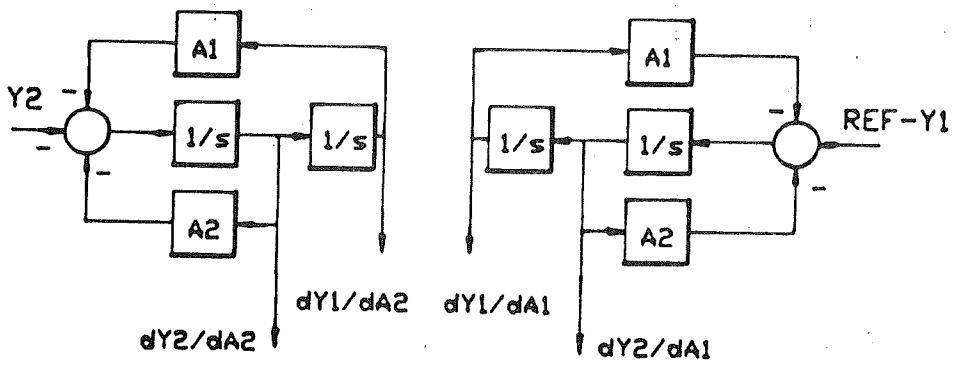


Figure 3.3: Le générateur des fonctions de sensibilité.

3.4 Loi de variation des gains

En utilisant (3.4,3.5), l'équation (2.8) devient:

$$\begin{bmatrix} \dot{\alpha}_1 \\ \dot{\alpha}_2 \end{bmatrix} = -\begin{bmatrix} \Gamma_{11} & \Gamma_{12} \\ \Gamma_{21} & \Gamma_{22} \end{bmatrix} (-1) \begin{bmatrix} \dfrac{\delta y_1}{\delta a_1} & \dfrac{\delta y_1}{\delta a_2} \\ \dfrac{\delta y_2}{\delta a_1} & \dfrac{\delta y_2}{\delta a_2} \end{bmatrix}^T \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

d'où:

$$\dot{\alpha}_1 = \Gamma_{11} \left[\frac{\delta y_1}{\delta a_1} (q_{11} e_1 + q_{12} e_2) + \frac{\delta y_2}{\delta a_1} (q_{21} e_1 + q_{22} e_2) \right] + \tag{3.6}$$

$$\Gamma_{12} \left[\frac{\delta y_1}{\delta a_2} (q_{11} e_1 + q_{12} e_2) + \frac{\delta y_2}{\delta a_2} (q_{21} e_1 + q_{22} e_2) \right]$$

$$\dot{\alpha}_2 = \Gamma_{21} \left[\frac{\delta y_1}{\delta a_1} (q_{11} e_1 + q_{12} e_2) + \frac{\delta y_2}{\delta a_1} (q_{21} e_1 + q_{22} e_2) \right] +$$

$$\Gamma_{22} \left[\frac{\delta y_1}{\delta a_2} (q_{11} e_1 + q_{12} e_2) + \frac{\delta y_2}{\delta a_2} (q_{21} e_1 + q_{22} e_2) \right]$$

En utilisant l'équation (3.2) et, supposant Km et τ constants par rapport au temps, cela nous amène à la loi de variation des gains suivante:

$$\dot{K1} = \frac{\tau \, \dot{\alpha}1}{Km} \qquad \dot{K2} = \frac{\tau \, \dot{\alpha}2}{Km} \qquad (3.7)$$

Cette loi est implantée en considérant le schéma à la figure (3.4). La figure (3.5) illustre le schéma du module d'adaptation des gains qui en résulte.

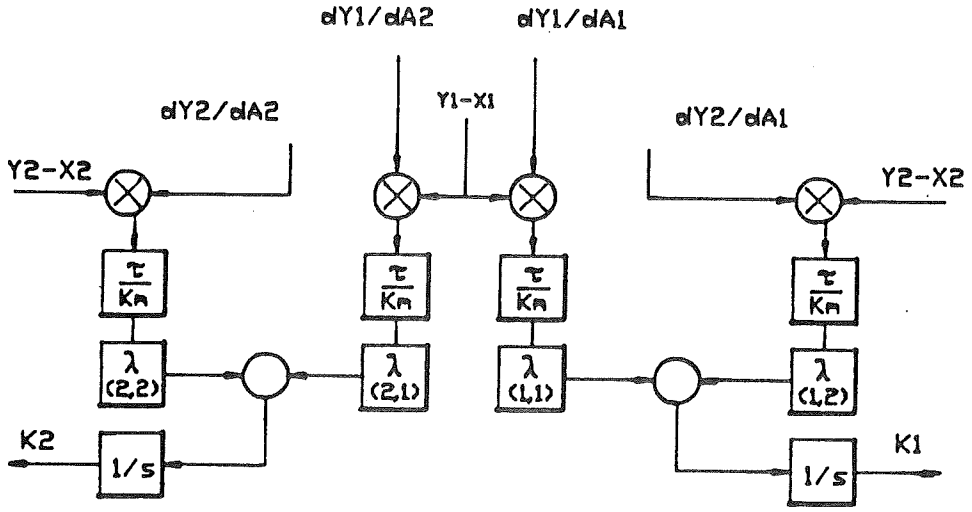


Figure 3.4: Loi de variation des gains.

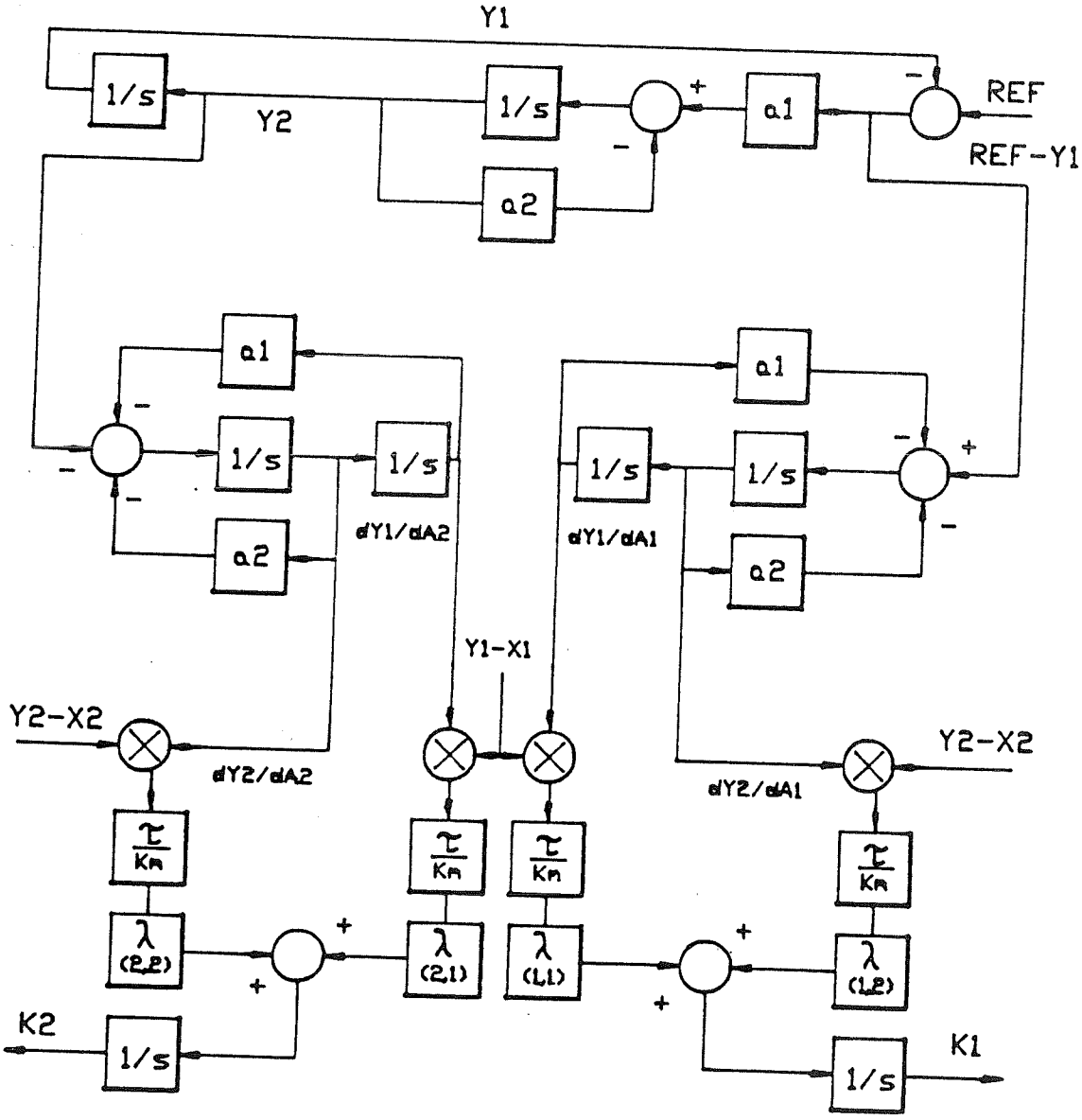


Figure 3.5: Le module d'adaptation des gains.

4. Le progiciel MRAC

MRAC est un progiciel interactif qui permet de simuler graphiquement le comportement dynamique du système représenté à la figure (2.1) avec les divers modules décrits par les figures (3.1) et (3.5). Sa structure interne est représenté par le schéma à la figure (4.1). Son code source écrit en "Turbo-Pascal" et est listé en appendice.

MRAC reçoit comme entrées la valeur des différents paramètres régissant les conditions de simulation ainsi que ceux déterminant le système à simuler.

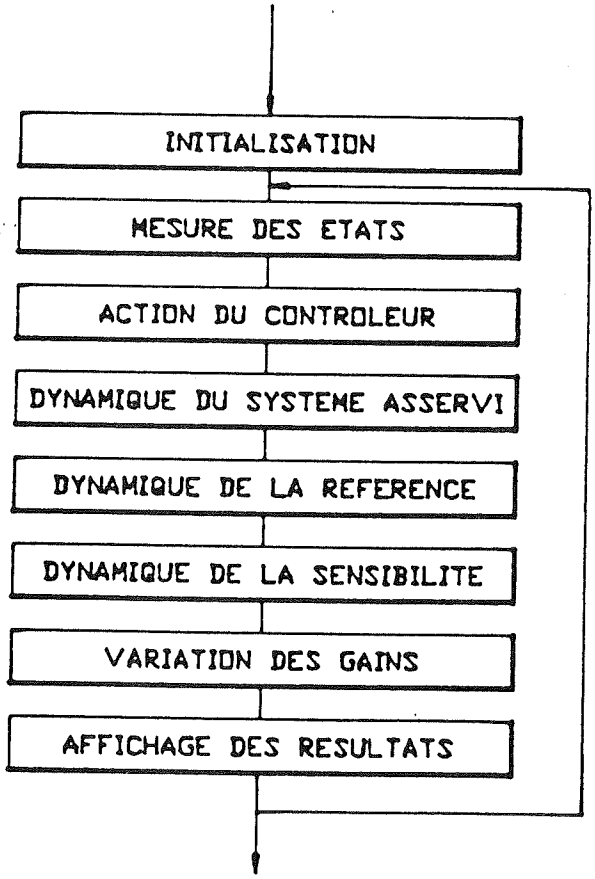


Figure 4.1: La structure du progiciel MRAC.

Les résultats offerts par MRAC sont représentés par: les graphiques de l'évolution de la position et de la vitesse angulaire des systèmes suivants:

- joint robotique asservi non-adaptativement,
- modèle de référence,
- joint robotique asservi adaptativement, et,
- Evolution, dans le temps, des gains du contrôleur PD.

Le potentiel d'expérimentation est facilité par la possibilité de garder, dans un fichier, la valeur actuel des paramètres pour continuer la simulation ultérieurement; la possibilité de conserver, dans un fichier d'un format compatible au logiciel MATLAB, les paramètres et les résultats d'une expérience.

5. Un exemple d'application

Afin d'illustrer l'opération du progiciel, nous considérons une simulation caractérisée par les données représentées à la figure (5.1).

FD ADAPTATIF SELON MODELE DE REEFERENCE
CONDITIONS DE SIMULATION

TEMPS: DEBUT ET FIN DE SIMULATION	(s)	=	0.0000	4.0000
REFERENCE ET SA PERIODE, 0 pour cont.	(rd, s)	=	5.0000	2.0000
MREF : POSITION, VITESSE INITIALE	(rd,rd/s)	=	0.0000	0.0000
SYST : POSITION, VITESSE INITIALE	(rd,rd/s)	=	0.0000	0.0000
PAS D'INTEGRATION ET D'OBSERVATION	(s)	=	0.0050	0.0100
TEMPS MIN ET MAX D'OBSERVATION	(s)	=	0.0000	4.0000
POSITION ET VITESSE MAXIMUMS	(rd,rd/s)	=	6.0000	12.0000
GAINS : K1 MIN et MAX		=	0.0000	2.0000
GAINS : K2 MIN et MAX		=	0.0000	0.2000
ESSAI no.		=	1.0000	

FD ADAPTATIF SELON MODELE DE REEFERENCE
PARAMETRES DU SYSTEME

MREF : FREQ.NATURELLE et AMORTISSEMENT	Wn,Tzeta	=	6.2832	0.7070
CNTL : GAINS PROPORTIONNEL ET DERIVE	K1,K2	=	0.5000	0.2000
CNTL : GAINS ADAPTATIFS	lambda[11,12]	=	10.0000	1.0000
	lambda[21,22]	=	1.0000	10.0000
PONDERATION	Q[11,12]	=	1.0000	1.0000
	Q[21,22]	=	1.0000	1.0000
SYST : GAIN et CSTE de TEMPS	Km, Tau	=	12.5000	0.1600
CAPTEURS : GAINS (v/rd,v*s/rd)	KP,KS	=	1.0000	1.0000
NIVEAUX DE BRUIT : (volt)	N1,N2	=	0.0000	0.0000

Figure 5.1: Un exemple d'application: Données à l'entrée.

La figure (5.2) illustre la représentation graphique de la simulation qui en résulte. Nous pouvons vérifier, sur cette figure, que la composante adaptative du contrôleur amène la courbe du système asservi adaptativement à délaisser le comportement du système asservi non-adaptativement pour épouser celui du modèle de référence dès le deuxième cycle. Nous assistons donc à une très bonne performance du contrôleur adaptatif utilisant la méthode du modèle de référence.

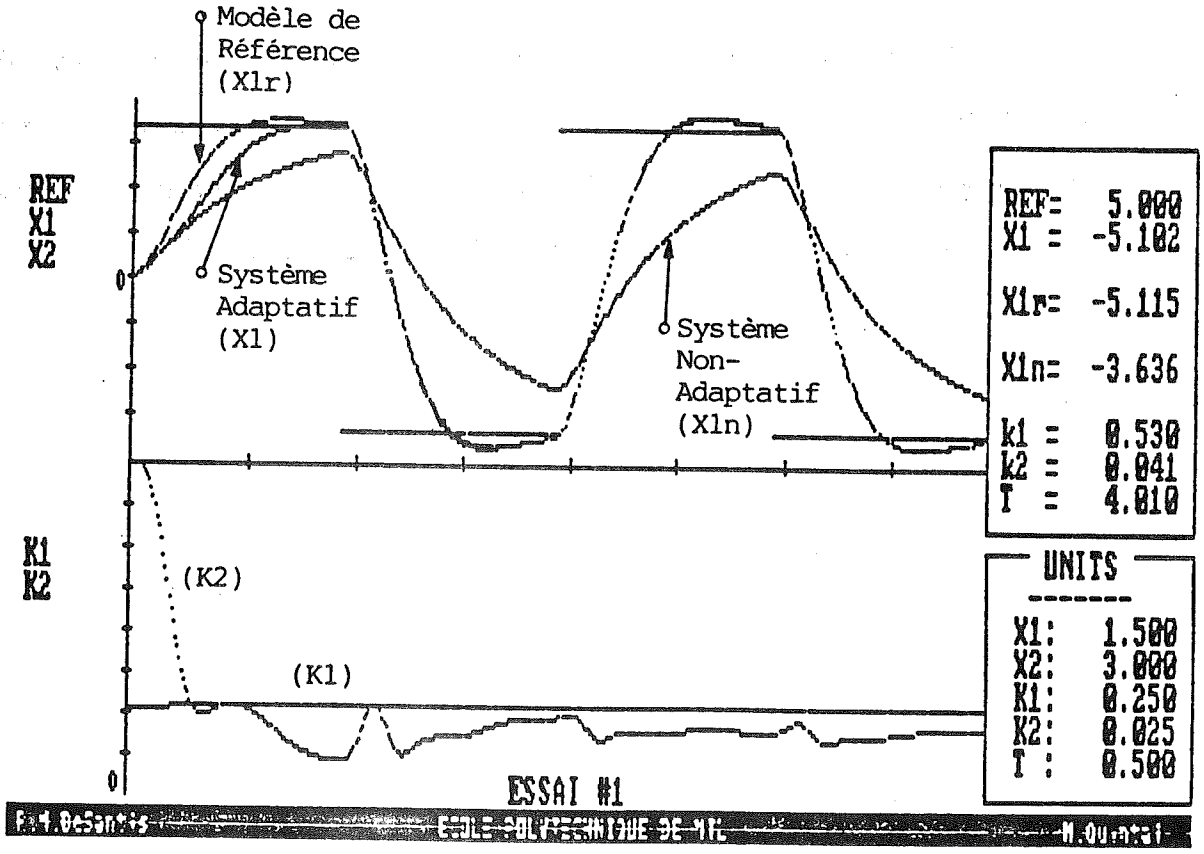


Figure 5.2: Un exemple d'application: Dynamique du système avec et sans adaptation.

CONCLUSION

Le progiciel MRAC semble bien répondre aux exigences fixées. Il permet à l'opérateur de se familiariser avec un grand nombre des aspects saillants de la commande adaptative: la nécessité de fournir un modèle de référence adéquat; la nécessité d'initialiser le système de façon à ce qu'il n'y ait qu'une petite différence entre la sortie du système asservi et celle du modèle de référence; la difficulté de bien choisir des valeur adéquates des coefficients de pondération et de facteurs d'accéleration.

Les expériences de simulation sont exécutées dans un environnement caractérisé par une rapidité d'exécution raisonnable, une bonne flexibilité, un graphisme interactif suffisamment claire et une interface amicale.

BIBLIOGRAPHIE

[1] DUBOWSKY S., KORNBLUH, "On the Developpement of High Performance Adaptative Control Algorithms for Robotic Manipulators", The Robotic Research: The 2nd International Symposium, MIT Pres (1985).

[2] LANDAU YOAN D., "Adaptative control", University of Southern California, Dekker (1979).

[3] BORLAND INTERNATIONAL INC, "Turbo Numerical Toolbox", (1986).

[4] BORLAND INTERNATIONAL INC, "Turbo Graphix Toolbox", (1985).

[5] ASTROM,K.J. & Co., "Theory and Application of Self-Tuning, Regulators, Automatica", 13, pp. 457-476, (1977).

[6] Quintal, M., DeSantis R. M.,"Simulation d'un Système de Contrôle Adaptatif Utilisant la Méthode du Modèle de Référence", Mémoire de projet de fin d'étude, Ecole Polytechnique de Montréal, (Août 1987).

[7] Serfass, C., DeSantis, R.M., "Contrôleur à Structure Variable pour une Articulation Robotique Elastique", EPM/RT-87/19, Rapport Technique de l'Ecole Polytechnique de Montréal, (mai 1987).

[8] Anh-Tu Nguyen, DeSantis, R. M., "Etude de Sensibilité du Comportement Dynamique d'un Manipulateur Robotique", EPM/RT-87/2 Rapport Technique de l'Ecole Polytechnique de Montréal, (Janvier 1987).

[9] Riendau, S., DeSantis, R. M., "Un Système de Mesure des Caractéristiques Electriques d'un Four à l'Arc", EPM/RT-87/43 Rapport Technique de l'Ecole Polytechnique de Montréal, (Novembre 1987).

[10] Terroux, A., DeSantis, R. M., "Asserv.pos: Un programme pour la simulation de la commande d'un joint robotique", Rapport Technique de l'Ecole Polytechnique de Montréal, (Juin 1984).

APPENDICE: Listage de MRAC.

```
{*****************************************************************}
{* Programme Mrac                                        MRAC.PAS  *}
{*****************************************************************}
{*                                  Ecole Polytechnique de Montréal *}
{*                                            Décemdre 1987         *}
{* Programmeur: Marc Quintal, ing.                                 *}
{* Directeur  : Dr. R.M. DeSantis                                  *}
{*                                                                 *}
{*                                                                 *}
{*                                                                 *}
{*     Ce programme fait la simulation d'un assservissement de position *}
{*  angulaire contol par un controleur adaptatif qui opere selon la *}
{*  mthode du modele de reference.                                 *}
{*                                                                 *}
{*                                                                 *}
{***          Il affiche un menu specifique aux programmes de simulat-***}
{*** tion de systemes dynamiques et execute chacune des parties de ce ***}
{*** menu.                                                          ***}
{*****************************************************************}


{-----------------------------------------------------------------}
{-  Pour compiler ce programme vous avez besoin de:                -}
{-  1   COMMON.INC       TURBO PASCAL NUMERICAL METHODS TOOLBOX    -}
{-  2   RUNGE_1.INC      TURBO PASCAL NUMERICAL METHODS TOOLBOX    -}
{-  3   RUNGE_2.INC      TURBO PASCAL NUMERICAL METHODS TOOLBOX    -}
{-  4   GRAPHIX.SYS      TURBO PASCAL GRAPHIX TOOLBOX              -}
{-  5   TYPEDEF.SYS      TURBO PASCAL GRAPHIX TOOLBOX              -}
{-  6   KERNEL.SYS       TURBO PASCAL GRAPHIX TOOLBOX              -}
{-  7   WINDOWS.SYS      TURBO PASCAL GRAPHIX TOOLBOX              -}
{-----------------------------------------------------------------}

{$C-}
{* {$K-} *}
{$I-}      { Disable I/O error trapping  }
{-----------------------------------------------------------------}
{-          CONSTANTES UTILISEES DANS LE PROGRAMME                -}
{-----------------------------------------------------------------}

CONST
   DefFileName  = 'MRAC';      { Nom par default des fichiers gnrs }
   ExtFileParam = 'PRM';       { Extention par dfault des fichiers de}
   ExtFileRes   = 'M';         { parametre et de rsultats compatibles}
                               { a MATLAB.                          }

   Nbre_Max_Parametres = 10;   { Nombre maximum de parametres par pages }
   Nbre_Max_Pages      = 2;    { Nombre maximum de pages              }
   Nbre_Max_Valeur     = 2;    { Possibilite de valeurs differentes pour}
                               { le meme type de parametre            }
   PremLinParam        = 6;    { Avant-premiere ligne d'affichage des  }
                               { parametres sur l'ecran.              }

                               { Constantes pse rapportant a la procedure}
                               { de simulation}

   IOerr : boolean = false;          { Global variable IOerr  }
```

```
                                        ( Flags an I/O Error     )

   NB_ETATS_ETUDIES  = 2;            ( X1 et X2 )
   ZeroProtect    = 0E-10;
   NbCharLbels    = 78;
   FactZoom       = 1.2;
   NbMaxObs       = 500;        ( Maximum de points calcules par la simulation )
                                ( qui sont gardes en memoire.                  )
   NbCourbes      = 8;          ( Nombre de courbes calculees par simulation : )
                                ( - Etat_sys_ass (vit + pos),                  )
                                ( - Etat_mod_ref (vit + pos),                  )
                                ( - Etat_sys_ass_non_adap (vit+pos),           )
                                ( - EtatK(1&2)                                 )
    ValMax        = 1000;       ( Valeur maximale permise pour affichage       )
                                ( ------------------------- )

{-----------------------------------------------------------------------}
{-              TYPES UTILISES DANS LE PROGRAMME                       -}
{-----------------------------------------------------------------------}

TYPE                                        ( Tableau de parametres )
   Val = Array[1..Nbre_Max_Parametres,1..Nbre_Max_Pages,1..Nbre_Max_Valeur]
         of Real;
                                            ( Tableau du nombre de valeur )
                                            ( de parametre                )
   NVal = Array[1..Nbre_Max_Parametres,1..Nbre_Max_Pages]
         of integer;
                                            ( Tableau des designations des )
                                            (  parametres                  )
   Des = Array[1..Nbre_Max_Parametres,1..Nbre_Max_Pages]
         of String[NbCharLbels];

   VectObs = Array[1..NbMaxObs]          ( Un Resultats d'une Simulations )
             of real;
   MatObs  = Array[1..NbCourbes] of      ( Tous les Resultats d'une Simulations )
             VectObs;

   String15 = String[15];   {Ceci servira pour toutes les demandes de noms}
                            {de fichiers                                  }
   String3 = String[3];     {Extentions des noms de fichiers              }


   STRING79 = string[79];


{-----------------------------------------------------------------------}
{-              TYPE UTILISES DANS LES CALCULS                         -}
{-----------------------------------------------------------------------}
  vector_of_two_integer = array[1..2] of integer;
  vector_of_two_real    = array[1..2] of real;
  MAT2x2                = array[1..2,1..2] of real;
  Vecteur_ETAT                                  ( Etats des systemes )
                        = array[0..NB_ETATS_ETUDIES] of real;
```

```
            {-----------------------------------------------------------------------}
{-              VARIABLES UTILISEES DANS LE PROGRAMME                    -}
{-----------------------------------------------------------------------}
VAR
  essaino,i,j,cycle : integer;
  temps0,                          { Temps de depart de la simulation      }
  temps,                           { Temps actuel durant la simulation     }
  deltemps,                        { Pas des observations de la simulation }
  limit_temps,                     { Temps de fin         de la simulation }
  Kp,Ks              : real;       { Gains des capteurs                    }
  K,
  a,                                 { A1 & A2 param. du Model de Reference }
  NOISE,                             { Bruit lors des lectures de capteurs  }
  dY_da1,dY_da2 : vector_of_two_real;{ Sensibilite du modele de reference   }
  Km , Tau,                          { Param. du systeme asservi            }
  Y1, Y2,                            { Position et vitesse du modele de MREF}
  Ref, Per_Ref       : real;         { Reference appliquee au systeme       }

  {-----------------------------------------------------------------------}
  {-------- PARAMETRE SUIVANT L'EVOLUTION DE LA SIMULATIION       ----------}
  {-----------------------------------------------------------------------}
  Etat_sys_ass,                    { Etat du systeme asservi               }
  Etat_mod_ref,                    { Etat du modele de reference           }
  Etat_dY_da1,Etat_dY_da2,         { Etat de la sensibilite du mod. de ref }
  Etat_K,                          { Etat des gains variables du controleur }
  Etat_k_stable,                   { Etat des gains stables du SANA        }
  Etat_sys_ass_non_adap            { Etat du systeme asservi par le        }
            : Vecteur_ETAT;        { controleur non adaptatif ( Ki stables )}
  lamda,                           { Gains adaptatifs du variateur de Ki   }
  Q          : MAT2x2;             { Facteur de ponderation de la fonction }
                                   { de quantification de l'erreur utilisee }
                                   { dans la methode du gradient           }
{ ===  graphics var ===}

  NbDivY  :       vector_of_two_integer;
  NbDivTime :integer;

  MinT,MaxT, MaxRef, MaxVit,       { Minimums et maximums des resultats qui }
  Mink1, Maxk1, Mink2, Maxk2       { sont montres lors de la simulation     }
                    : real;


                                     { PROCESSUS D'INITIALISATION }
  OkSimul ,
  OvFlow  : boolean;
```

```
{-----------------------------------------------------------------------}
{-          VARIABLES UTILISEES DANS LE PROGRAMME                     -}
{-----------------------------------------------------------------------}

  Fichier_Parametre ,              {Nom du fichier de parametre donn par
                                   {l'utilisateur}
  Fichier_Resultats : String15;    {Nom du fichier de rtesultats donn par
                                   {l'utilisateur}
  TapeP    : File of Val;
  TapeLbel : File of Des; {Fichiers emmagasinant l'information contenue dans}
                   {les tableaux A et P, si desire}

  P     : Val;                     { Tableau de valeurs des parametres  }
  N     : NVal;                    { Tableau donnant le nombre de valeur}
                                   { par parametre                      }
  Lbel : Des;                      { Tableau de descriptions des parame-}
                                   {  tres}
  Plot : array[1..6] of boolean;   { Fanions de visualisation des       }
                                   { graphiques                         }
  Resultats            : MatObs;

  Choixms,
  Choixp,
  Choix,
  ch      : Char;                  {Choix d'un item dans les menus}


                                 { Variables necessaires pour les appels }
                                 { de sousroutines du Numerical Toolbox  }
  NumEquations  : integer;       { The number of first order equations   }
  LowerLimit    : real;          { Lower limit of interval               }
  UpperLimit,
  Pint          : real;          { Upper limit of interval               }
  NumReturn     : integer;       { Number of values to return            }
  NumIntervals  : integer;       { Number of intervals                   }

  Er : byte;                     { Flags if something went wrong  }
  Erreur : byte;                 { Flags if something went wrong  }
  OutFile : text;                { Output file                    }

{=======================================================================}
{-----------------------------------------------------------------------}
{-                      Appels de fichiers                            -}
{-----------------------------------------------------------------------}

{$I D:\TYPEDEF.SYS }
{$I D:\GRAPHIX.SYS }
{$I D:\KERNEL.SYS  }
{$I D:\WINDOWS.SYS }
{$I D:\COMMON.INC}              { Load procedure IOCheck  }
{$I MRAC1.PAS}
```

```
            {----------------------------------------------------------------------------}
{-                       PROCEDURES   ET   FONCTIONS                                -}
{---------------------------------------------------------------------------------}

Procedure Initialise_Description_Parametres;
{*************************************************************************}
{*** Cette procedure initialise le tableau de description des parametres ***}
{*************************************************************************}

Begin    { PAGE 2 }
     Lbel[1,2]:='  MREF : FREQ.NATURELLE et AMORTISSEMENT    Wn,ksi =  ';
     Lbel[2,2]:='  CNTL : GAINS PROPORTIONNEL ET DERIVE       K1,K2 =   ';
     Lbel[3,2]:='  ADAPT: FACTEURS D''ACCELERATION     lambda[11,12] =  ';
     Lbel[4,2]:='                                     lambda[21,22] =  ';
     Lbel[5,2]:='         COEFFICIENTS DE PONDERATION      q[11,12] =  ';
     Lbel[6,2]:='                                          q[21,22] =  ';
     Lbel[7,2]:='  SYST : GAIN et CSTE de TEMPS            Km, Tau =  ';
     Lbel[8,2]:='  CAPTEURS : GAINS (v/rd,v*s/rd)             KP,KS =  ';
     Lbel[9,2]:='  NIVEAUX DE BRUIT  : (volt)                 N1,N2 =  ';

     Lbel[10,2]:='                                                      ';
                                            { ----- graphisme ----- }
          { PAGE 1 }
     Lbel[1,1]:='  TEMPS:  DEBUT ET FIN DE SIMULATION          (s) =  ';
     Lbel[2,1]:='  REFERENCE ET SA PERIODE, 0 pour cont.    (rd, s) =  ';
     Lbel[3,1]:='  MREF :  POSITION, VITESSE INITIALE     (rd,rd/s) =  ';
     Lbel[4,1]:='  SYST :  POSITION, VITESSE INITIALE     (rd,rd/s) =  ';
     Lbel[5,1]:='  PAS D''INTEGRATION ET D''OBSERVATION          (s) =  ';

     Lbel[6,1]:='  TEMPS MIN ET MAX D''OBSERVATION              (s) =  ';
     Lbel[7,1]:='  POSITION ET VITESSE MAXIMUMS           (rd,rd/s) =  ';
     Lbel[8,1]:='  GAINS : K1 MIN et MAX                            =  ';
     Lbel[9,1]:='          K2 MIN et MAX                            =  ';
     Lbel[10,1]:='  ESSAI no.                                        = ';
End;

Procedure Initialise_Valeur_Parametres;
{*************************************************************************}
{*** Initialise les valeurs de default des parametres.                ***}
{*** Cette procedure a pour but d'initialiser les valeurs des parametres ***}
{*** dans le cas ou l'utilisateur n'a pas de fichier de valeurs disponible ***}
{*************************************************************************}

 BEGIN      { PAGE 2 }
  N[1,2]:=2; P[1,2,1]:=2*pi;     P[1,2,2]:=0.707;       {Wn, Tzeta }
  N[2,2]:=2; P[2,2,1]:=0.5;      P[2,2,2]:=0.2;          {K1,K2}
  N[3,2]:=2; P[3,2,1]:=10;       P[3,2,2]:=1;            {Lambda 11,12 }
  N[4,2]:=2; P[4,2,1]:=1;        P[4,2,2]:=10;           {Lambda 21,22 }
  N[5,2]:=2; P[5,2,1]:=1;        P[5,2,2]:=1;            {     Q 11,12 }
  N[6,2]:=2; P[6,2,1]:=1;        P[6,2,2]:=1;            {     Q 21,22 }
  N[7,2]:=2; P[7,2,1]:=12.5;     P[7,2,2]:=0.16;         {Km,Tau}
  N[8,2]:=2; P[8,2,1]:=1.0;      P[8,2,2]:=1.0;          {KP,KS}
  N[9,2]:=2; P[9,2,1]:=0;        P[9,2,2]:=0;            {NOISE(i)}
  N[10,2]:=0;
```

```
                                                    { --------- graphisme --------- }
              { PAGE 1 }

   N[1,1]:=2;   P[1,1,1]:= 0;   P[1,1,2]:= 4;              { TEMPS0,LIMIT_TEMPS }
   N[2,1]:=2;   P[2,1,1]:= 5;   P[2,1,2]:=(P[1,1,2]-P[1,1,1])/2; { REF,PER_REF }

   N[3,1]:=2;   P[3,1,1]:= 0;   P[3,1,2]:= 0;              { MREF:  POS0, VIT0 }
   N[4,1]:=2;   P[4,1,1]:= 0;   P[4,1,2]:= 0;              { SYST:  POS0, VIT0 }
   N[5,1]:=2;   P[5,1,1]:=0.005; P[5,1,2]:=0.01;           { Pint,deltemps=obs }
   N[6,1]:=2;   P[6,1,1]:= P[1,1,1];                       { MinT,             }
                P[6,1,2]:= P[1,1,2];                       {            ,MaxT  }

   N[7,1]:=2;   P[7,1,1]:= FactZoom*P[2,1,1];              { MaxRef,           }
                P[7,1,2]:= 2*FactZoom*P[2,1,1];            {           ,MaxVit }
   N[8,1]:= 2;  P[8,1,1]:= 0;  P[8,1,2]:= 4*P[2,2,1];      { MinK1 ,    Maxk1  }
   N[9,1]:= 2;  P[9,1,1]:= 0;  P[9,1,2]:= 1*P[2,2,2];;     { MinK2 ,    Maxk2  }
   N[10,1]:=1;  P[10,1,1]:=1;                              { ESSAI*            }

    for i:=1 to 3 do Plot[i] := True;        { Initialisation du choix des   }
    for i:=4 to 6 do Plot[i] := False;       { graphiques a tracer           }
    for i:=7 to 8 do Plot[i] := True;

    Fichier_Parametre := DefFileName;        { Initialisation du noms des    }
    Fichier_Resultats := DefFileName;        { fichiers de parametres et de  }
                                             { resultats                     }
    NbDivTime  := 8;                         { Nombre de division des graphs.}
    NbDivY[1]  := 4;   NbDivY[2] := 8;

END;
```

```
PROCEDURE RafraichirParametres;
{*****************************************************************************}
{*** Cette procedure a pour but de rafraichir la valeur des   parametres   ***}
{*** qui peuvent avoir ete changer par l'usager                            ***}
{*****************************************************************************}
 VAR i: integer;
 BEGIN
                                        { PARAMETRES RELATIFS AU SYSTEME }
                                        { ============================ }
    { PAGE 2 }
    a[1]       :=P[1,2,1]*P[1,2,1];                  { a1 = Wn^2            }
    a[2]       :=2*P[1,2,2]*P[1,2,1];                { a2 = 2*Tzeta*Wn      }
    Etat_k[1] :=P[2,2,1];  Etat_k[2] :=P[2,2,2];     { K1, K2               }
    lamda[1,1]:=P[3,2,1];  lamda[1,2]:=P[3,2,2];     { Mat. gains adaptatifs}
    lamda[2,1]:=P[4,2,1];  lamda[2,2]:=P[4,2,2];
    Q[1,1]     :=P[5,2,1];     Q[1,2]:=P[5,2,2];     { Mat. de ponderation Q}
    Q[2,1]     :=P[6,2,1];     Q[2,2]:=P[6,2,2];
    Km         :=P[7,2,1];  Tau       :=P[7,2,2];    { Km,  Tau             }
    KP         :=P[8,2,1];  KS        :=P[8,2,2];    { Kp,  Ks              }
    Noise[1]   :=P[9,2,1];  Noise[2]  :=P[9,2,2];
```

```
                                      { PARAMETRES RELATIFS A LA SIMULATION }
                                    { ================================== }
   { PAGE 1 }
   temps0          := P[1,1,1];     Limit_Temps     := P[1,1,2];
   Ref             := P[2,1,1];     Per_Ref         := P[2,1,2];
   Etat_mod_ref[1] := P[3,1,1];     Etat_mod_ref[2] := P[3,1,2];
   Etat_sys_ass[1] := P[4,1,1];     Etat_sys_ass[2] := P[4,1,2];

   Pint            := P[5,1,1];     deltemps        := P[5,1,2];
   MinT            := P[6,1,1];     MaxT            := P[6,1,2];
   MaxRef          := P[7,1,1];     MaxVit          := P[7,1,2];
   MinK1           := P[8,1,1];     MaxK1           := P[8,1,2];
   MinK2           := P[9,1,1];     MaxK2           := P[9,1,2];
   essaino         := trunc(P[10,1,1]);


                                    { INITIALISATIONS de depart ncessaires}
                                    { a chaque simulation.                }
                                    { =================================== }
   cycle        := 0;               { cycles d'horlorge                   }
   temps := temps0;                 { Temps de depart de la simulation    }
   Etat_k[0]    := 0;
   OvFlow := False;                 { Statut de l'overflow }

   Etat_k_stable :=Etat_k;
   Etat_sys_ass_non_adap   := Etat_sys_ass;
   FillChar( Etat_dY_da1, SizeOf( Etat_dY_da1), 0);
   FillChar( Etat_dY_da2, SizeOf( Etat_dY_da2), 0);

   if Pint=0 then Pint := ZeroProtect; { Protection des divisions par zero}

   if deltemps>Pint then                     { Protection du cas ou l'usager}
      NumIntervals := round(deltemps/Pint)   { pourrait demander un pas d'  }
   else                                      { integration plus grand que le}
      NumIntervals := 1;                     { pas d'observation            }
                                             { ---------------------------- }
END;   { RafraichirParametres; }




Procedure Set_Graphique_a_tracer;
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
{$$$ Cette procedure permet de a l' usager de choisir les graphiques qu'il $$$}
{$$$ desise visualiser.                                                      $$$}
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}

VAR
   I  :integer;           {Cette valeur sert a positionner le curseur}
                          {apres chaque deplacement de ligne         }
   CHR   : Char;          {Variable entree par l'utilisateur pour le }
                          {deplacement du curseur ou la sortie       }
   setting  : boolean;    { decision de tracer ou non un graph }
BEGIN
```

```
TextBackground(7);TextColor(1);
ClrScr;
GotoXY(20,7);   Writeln('CHOIX DE VISUALISATION DES  GRAPHIQUES');
GoToXY(20,10);  Writeln('  Entrez O pour oui, et N pour non.');
Writeln;
GoToXY(5,14);
Writeln('Position du Systme asservi adaptatif.......................');
GoToXY(5,15);
Writeln('Position du Modle de rfrence..............................');
GoToXY(5,16);
Writeln('Position du Systme asservi non adaptatif...................');
GoToXY(5,17);
Writeln('Vitesse  du Systme asservi adaptatif.......................');
GoToXY(5,18);
Writeln('Vitesse  du Modle de rfrence..............................');
GoToXY(5,19);
Writeln('Vitesse  du Systme asservi non adaptatif...................');
GoToXY(5,22);   Writeln('Taper :<C> pour Continuer...');
GoToXY(5,23);   Writeln('       <R> pour Retourner au menu principal. .');

  for i:=1 to 6 do
  begin
     GoToXY(67,13+i);
     if (Plot[i]) then write('O')
     else         write('N');
  end;
  I:=1;CHR:='O';
  setting := true;
  WHILE ((CHR<>'C') and (CHR<>'R')) DO
     BEGIN
       GOTOXY(67,I+13);
       READ(KBD,CHR);CHR:=UPCASE(CHR);
       if (CHR=#27) then
                       begin
                         READ(KBD,CHR);
                         IF (CHR=#72) THEN I:=I-1;
                         IF (CHR=#80) THEN I:=I+1;
                         IF I> 6 THEN I:=1;
                         IF I< 1 THEN I:=6;
                       end;
       if ((CHR='O')or(CHR='N')) then
                       begin
                          GOTOXY(67,I+13);
                          WRITE(CHR);
                          case CHR of
                               'O' : Plot[i] := True;
                               'N' : Plot[i] := False;
                          end;
                        end;
        if CHR='R' then OkSimul := False;
        if CHR='C' then OkSimul := True;
     end;
End;
```

```
function ValOk(x:real) : boolean;
{*************************************************************************}
{*** Cette fonction test si les valeurs calculees sont d'un ordre de   ***}
{*** grandeur permis par ValMax                                        ***}
{*************************************************************************}

Begin
    if ((x<ValMax) and (x>-ValMax)) then
       ValOk := True
    else
    begin
       ValOk := False;
       OvFlow := True;
    end;
End;


Procedure SauvegardeResultats;
{*************************************************************************}
{*** Cette procedure sauvegarde les resultats de la derniere simulation ***}
{*** dans un fichier compatible a MatLab.                              ***}
{*************************************************************************}
VAR
   FichierR : Text;
   FileTemp : String15;
   OK : Boolean;

Begin
   TextBackground(7);TextColor(1);
   ClrScr;
   if cycle=0 then      { if une simulation a te effectue : (cycle<>0 }
   Begin
      GoToXY(20,10); Writeln('IL N''Y A PAS DE RESULTATS EN MEMOIRE,');
      GoToXY(20,11); Writeln('FAITES UNE SIMULATION...');
   End
   Else
   Begin
      GotoXY(1,5);
      Write('Dans quel fichier voulez-vous sauvegarder vos resultats?  ');
      Writeln;
      Writeln('L''extention <.',ExtFileRes,'> est assume si aucune n''est prcise.');

      FileTemp := Fichier_Resultats;
      GoToXY(20,10); Write('===<',FileTemp,'>===> : ');
      Readln(Fichier_Resultats);

      if Fichier_Resultats='' then Fichier_Resultats := FileTemp;

      SetFileName(Fichier_Resultats  ,ExtFileRes);
      Assign(FichierR,Fichier_Resultats);
      {I-} Rewrite(FichierR) {I+};
      OK:=(IOresult=0);
      If Not OK Then
      begin
```

```
                Writeln('Le fichier ',Fichier_Resultats,' existe dja,');
    Writeln('Les rsultats n'' ont pas ts sauvegards');
end;
If OK Then
Begin                    { Bloc de sauvegarde des parametres }
                         { et des resultats dans un fichier 'text'}
                         { et dans un format compatible a Matlab }
   GoToXY(10,18);
      Writeln('La sauvegarde des parametres et rsultats dans');
   GoToXY(10,19);
      Writeln('un fichier compatible a MatLab est en cours...');
   {=============================================}

   Writeln(FichierR,'a = [',     a[1]:4:3,     ' ',  a[2]:4:3, ']};'                  );
   Writeln(FichierR,'Km = ',     Km:4:3,     '; Tau = ',  Tau:4:3, ';');

   Writeln(FichierR,'lamda = [', lamda[1,1]:4:3, ' ',  lamda[1,2]:4:3, ' ;',
                                 lamda[2,1]:4:3, ' ',  lamda[2,2]:4:3, ']};'   );
   Writeln(FichierR,'KP = ',     KP:4:3,   '; KS = ',  KS:4:3, ';'            );

   Writeln(FichierR,'Noise = [', Noise[1]:4:3,   ' ', Noise[2]:4:3,   ' ];'    );

                                    { ----- graphisme ----- }
   Writeln(FichierR,'T0 = ',temps0:4:3,'; DT = ',deltemps:4:3,'; T1 =',
                 Limit_temps:4:3, ';' );
   Writeln(FichierR,'REF = ',Ref:4:3,'; PERIOD = ',  Per_Ref:4:3,   '; '       );


   Writeln(FichierR,'X1=[');
   for i:= 1 to Cycle do  Writeln(FichierR,Resultats[1,i]:4:3);
   Writeln(FichierR,'];');
   Writeln(FichierR,'MRX1=[');
   for i:= 1 to Cycle do  Writeln(FichierR,Resultats[2,i]:4:3);
   Writeln(FichierR,'];');
   Writeln(FichierR,'X1NA=[');
   for i:= 1 to Cycle do  Writeln(FichierR,Resultats[3,i]:4:3);
   Writeln(FichierR,'];');
   Writeln(FichierR,'X2=[');
   for i:= 1 to Cycle do  Writeln(FichierR,Resultats[4,i]:4:3);
   Writeln(FichierR,'];');
   Writeln(FichierR,'MRX2=[');
   for i:= 1 to Cycle do  Writeln(FichierR,Resultats[5,i]:4:3);
   Writeln(FichierR,'];');
   Writeln(FichierR,'X2NA=[');
   for i:= 1 to Cycle do  Writeln(FichierR,Resultats[6,i]:4:3);
   Writeln(FichierR,'];');
   Writeln(FichierR,'K1=[');
   for i:= 1 to Cycle do  Writeln(FichierR,Resultats[7,i]:4:3);
   Writeln(FichierR,'];');
   Writeln(FichierR,'K2=[');
   for i:= 1 to Cycle do  Writeln(FichierR,Resultats[8,i]:4:3);
   Writeln(FichierR,'];');

   {=============================================}

   Close(FichierR);
```

```
                    GoToXY(10,22);   Writeln('...Opration effectue, pressez <ENTER>.');
      End;     ( if ouverture du fichier destination = Ok )
   End;        ( if une simulation a tete effectuee (cycle<>0 )
readln;
End;


PROCEDURE DISPLAY_RESULTS(var REF    : real;
                          var Etat_sys_ass,
                              Etat_mod_ref,
                              Etat_sys_ass_non_adap,
                              Etat_K : Vecteur_Etat);
(*************************************************************************)
(*** Cette procedure dessine les resultats de la simulation en cours    ***)
(***                                                                    ***)
(*************************************************************************)

   BEGIN
    if (ValOk(ref) and  ValOk(temps)) then
    begin
    SetClippingOn;
    SelectWorld(1);
    SelectWindow(1);
    DrawPoint(temps,ref);                        ( REFERENCE )
    GoToXY(68,6);  write('REF=',ref:8:3);

    if (Plot[1]) then
    begin
        if ValOk(Etat_sys_ass[1]) then          ( SYSTEME ASSERVI: POSITION )
        begin
        SelectWorld(1);
        SelectWindow(1);
        DrawPoint(temps,Etat_sys_ass[1]);
        GoToXY(68,7);  write('X1 =',Etat_sys_ass[1]:8:3);
        end;
    end;
    if (Plot[4]) then
    begin
        if ValOk(Etat_sys_ass[2]) then          ( SYSTEME ASSERVI: VITESSE )
        begin
        SelectWorld(2);
        SelectWindow(1);
        DrawPoint(temps,Etat_sys_ass[2]);
        GoToXY(68,8);  write('X2 =',Etat_sys_ass[2]:8:3);
        end
    end;
    if (Plot[2]) then
    begin
        if ValOk(Etat_mod_ref[1]) then          ( MODELE REFERENCE: POSITION )
        begin
```

```
                    SelectWorld(1);
        SelectWindow(1);
        DrawPoint(temps,Etat_mod_ref[1]);
        GoToXY(68,9);  write('X1r=',Etat_mod_ref[1]:8:3);
        end;
    end;
    if (Plot[5]) then
    begin
        if ValOk(Etat_mod_ref[2]) then        { MODELE REFERENCE: VITESSE }
        begin
        SelectWorld(2);
        SelectWindow(1);
        DrawPoint(temps,Etat_mod_ref[2]);
        GoToXY(68,10); write('X2r=',Etat_mod_ref[2]:8:3);
        end;
    end;

    if (Plot[3]) then                       { SYST. NON ADAPTATIF: POSITION }
    begin
        if ValOk(Etat_sys_ass_non_adap[1]) then
        begin
        SelectWorld(1);
        SelectWindow(1);
        DrawPoint(temps,Etat_sys_ass_non_adap[1]);
        GoToXY(68,11); write('X1n=',Etat_sys_ass_non_adap[1]:8:3);
        end;
    end;
    if (Plot[6]) then                       { SYST. NON ADAPTATIF: VITESSE }
    begin
        if ValOk(Etat_sys_ass_non_adap[2]) then
        begin
        SelectWorld(2);
        SelectWindow(1);
        DrawPoint(temps,Etat_sys_ass_non_adap[2]);
        GoToXY(68,12); write('X2n=',Etat_sys_ass_non_adap[2]:8:3);
        end;
    end;
        if ValOk(Etat_K[1]) then             { K1: GAIN PROPORTIONNEL }
        begin
           SelectWorld(3);
           SelectWindow(2);
           DrawPoint(temps,abs(Etat_K[1]));
           GoToXY(68,13); write('k1 =',Etat_k[1]:8:3);
        end;
        if ValOk(Etat_K[2]) then             { K2: GAIN DERIVE       }
        begin
           SelectWorld(4);
           SelectWindow(2);
           DrawPoint(temps,abs(Etat_K[2]));
           GoToXY(68,14); write('k2 =',Etat_k[2]:8:3);
        end;
{$
        { ================== TEST: SENSIBILITE ============== }
          SelectWorld(1);                   { Tracage de la sensibilite  }
          SelectWindow(1);                  { afin de tester le projiciel}
          DrawPoint(temps,Etat_dY_dal[1]);
```

```
                        DrawPoint(temps,Etat_dY_da2[1]);
        { ================== TEST: SENSIBILITE ============== }
*)

         GoToXY(68,15);  write('T  =',temps:8:3);
      end; { ValOk(ref,temp) }
END;  (DISPLAY OUTPUT VALUES)
{==========================================================================}






PROCEDURE INITIALIZE_GRAPHICS;
{****************************************************************************}
{*** Cette procedure initialise le mode graphique avec les facteurs      ***}
{*** echelle voulus et dessine les titres.                               ***}
{****************************************************************************}

  var x1,y1,x2,y2,pixu,deltat : real;       { x1  : world ,   }
      i, sx1,sy1,sx2,sy2,symid,sxpix :integer;      { sx1 : screen,   }
BEGIN                                      { pixu : pix/unit }
  InitGraphic;
  SetBackgroundColor(1);
  SetForegroundColor(15);

  TextBackground(9); TextColor(14);
  GoToXY(5,2);write('SYSTEME CONTROLE ADAPTATIVEMENT SELON LA METHODE DU MODELE DE REFERENCE');
  GoToXY(35,24);write('ESSAI #',essaino);
  SetWindowModeOn;
  SetClippingOff;

  SetColorWhite;

  { PREMIER GRAPH ---------------------------------}
  x1 := MinT;             x2:= MaxT;

  if (ValOk(x1) and ValOk(x2)) then
  begin
     y1 := -abs(MaxRef);      y2:= abs(MaxRef);
     if (ValOk(y1) and ValOk(y2)) then
        DefineWorld(1, x1, y1, x2, y2 );

     y1 := -abs(MaxVit);      y2:= abs(MaxVit);
     if (ValOk(y1) and ValOk(y2)) then
        DefineWorld(2, x1, y1, x2, y2);

     y1 := abs(Mink1);      y2:= abs(Maxk1);
     if (ValOk(y1) and ValOk(y2)) then
        DefineWorld(3, x1, y1, x2, y2);

     y1 := abs(Mink2);      y2:= abs(Maxk2);
     if (ValOk(y1) and ValOk(y2)) then
        DefineWorld(4, x1,y1, x2, y2);
```

```
(*    DefineWorld(5, x1,-0.3, x2, 0.3);            ( TEST : SENSIBILITE )*)

    sx1:= round(0.10*XMaxGlb);  sy1:= round(0.10*YMaxGlb);
    sx2:= round(0.82*XMaxGlb);  sy2:= round(0.55*YMaxGlb);
    symid:= sy1 + round( 0.5*(sy2-sy1) );
    pixu:= (sy2-sy1)/(2*MaxRef);

    DefineWindow(1,sx1,sy1,sx2,sy2);                 ( WINDOW#1: REF/X1/X2 )
    SelectWorld(1);
    SelectWindow(1);

    { AXES TEMPS ET Y1 }
    y1 := -abs(MaxRef);     y2:= abs(MaxRef);
    DrawLine(x1,y1,x2,y1);                                  { axe temps }
    DrawLine(x1,y1,x1,y2);                                  { axe ordonnee }
    { ------------------- }

    { GRADUATION DES Y1 }
      DrawStraight(round(sx1*8)-2,round(sx1*8)+2,symid  );  { grad -0- }
      i:=1;
      while (i<NbDivY[1]) do
      begin
         DrawStraight(round(sx1*8)-2,round(sx1*8)+2,
                      symid - i*round( MaxRef/NbDivY[1]*pixu ));
         DrawStraight(round(sx1*8)-2,round(sx1*8)+2,
                      symid + i*round( MaxRef/NbDivY[1]*pixu ));
         i:=i+1;
      end;
      DrawText(round(sx1*8)-8,symid,1,'0');
      { ------------------- }

      { GRADUATION DU TEMPS }
      deltat := MaxT-MinT;                        { divisions t }
      pixu:= ((sx2-sx1)*8+7)/deltat;
      sxpix:=round(sx1*8)+7;                 { byte to pixel }
                                        { Tracage des divisions du temps }
      SetWindowModeOff;
      i:=1;
      while (i<NbDivTime) do
      begin
         DrawLineClipped(sxpix+round(i*deltat/NbDivTime*pixu),sy2-2,
                         sxpix+round(i*deltat/NbDivTime*pixu),sy2+2);
         i:=i+1;
      end;
      SetWindowModeOn;
      { ------------------- }
      { DEUXIEME GRAPH ---------------------------------}

      sx1:= round(0.10*XMaxGlb);  sy1:= round(0.55*YMaxGlb);
      sx2:= round(0.82*XMaxGlb);  sy2:= round(0.95*YMaxGlb);
      pixu:= (sy2-sy1)/(Maxk1);
      DefineWindow(2,sx1,sy1,sx2,sy2);                 ( WINDOW#2: K1/K2 )


      SelectWorld(4);
```

```
                SelectWindow(2);
      y1 := abs(Mink2);        y2:= abs(Maxk2);
      DrawLine(x1,y1,x1,y2);                          { axe Y2: k1,k2 }

      { GRADUATION DES Y2 }
      i:=1;
      while (i<NbDivY[2]) do
      begin
         DrawStraight(round(sx1*8)-2,round(sx1*8)+2,
                  sy2 - i * round(MaxK1 / NbDivY[2] * pixu));
         i:=i+1;
      end;
      DrawText(round(sx1*8)-8,sy2-4,1,'0');
      { ------------------- }

      { TROISIEME GRAPH ---------------------------------}  { WINDOW#3: DATA BOX }
      DefineWindow(3,round(0.84*XMaxGlb),round(0.15*YMaxGlb),
                 round(1.00*XMaxGlb),round(0.625*YMaxGlb) );
      SelectWindow(3);  DrawBorder;
                                                        { ======== }
                                                   { WINDOW#4: UNIT BOX }
      DefineWindow(4,round(0.84*XMaxGlb),round(0.65*YMaxGlb),
                 round(1.00*XMaxGlb),round(0.95*YMaxGlb) );
      SelectWindow(4);  DrawBorder;
                                                        { ======== }
        GoToXY(2,6);    write('REF');
        GoToXY(2,7);    write('X1');
        GoToXY(2,8);    write('X2');
        GoToXY(2,17);   write('K1');
        GoToXY(2,18);   write('K2');
        GoToXY(70,17);  write(' UNITS ');
        GoToXY(70,18);  write('-------');
        GoToXY(69,19);  write('X1:',  MaxRef/ NbDivY[1] :8:3);
        GoToXY(69,20);  write('X2:',  MaxVit/ NbDivY[1] :8:3);
        GoToXY(69,21);  write('K1:',  MaxK1 / NbDivY[2]  :8:3);
        GoToXY(69,22);  write('K2:',  MaxK2 / NbDivY[2]  :8:3);
        GoToXY(69,23);  write('T :', (MaxT-MinT)/ NbDivTime  :8:3);
                                   {============================}

                                                   {  BORDER OFICIEL }
       for i:=0 to 7 do DrawStraight(1,Trunc(XMaxGlb*8+7),YMaxGlb-i);
      SetColorBlack;
      SetClippingOff;
      DrawText(1,YMaxGlb-4,1,' R.M.DeSantis                          ECOLE POLYTECHNIQUE DE MTL');
      DrawText(XMaxGlb*8-10*6,YMaxGlb-4,1,'M.Quintal');
      SetColorWhite;                                          { ======== }
  { =============== }
  end; { if VakOk( x1,x2 ) }
END;
{=================================================================}
```

```
            PROCEDURE REFERENCE(VAR temps, Per_Ref, REF : real);
{************************************************************************}
{*** Cette procedure calcule la valeur de la reference a chaque pas   ***}
{*** d' observation d'apres le temps actuel et sa periode.             ***}
{************************************************************************}

var sign,i : integer;
Begin
   if (Per_Ref<>0) then
   begin
      sign := (-1);
      for i:=0 to TRUNC( temps/(Per_ref/2) ) do sign := (-1)*sign;
      Ref := sign * abs(Ref);
   end;
End;
{========================================================================}




PROCEDURE SYSTEME_ASSERVI(VAR temps,deltemps,
                              Km, Tau : real;
                          var Etat_K,
                              Etat_sys_ass   : Vecteur_ETAT);
{************************************************************************}
{*** Cette procedure calcule la valeur du systeme asservi a chaque pas ***}
{*** d' observation et avec le pas d'integration desire.               ***}
{************************************************************************}

  const
    TNArraySize = 1;                 { Number of states to return }
  type
    TNvector = array[0..TNArraySize] of real;

  var
{      InitialValue, InitialDeriv : real;} { Initial values at lower limit  }
     TValues : TNvector;                  { Value of T between the limits  }
     XValues : TNvector;                  { Value of X at TValues          }
     XDerivValues : TNvector;             { Derivative of X at TValues     }
     alpha,                               { Parametres variables           }
     lect                                 { Facteur d'erreur dans les      }
                                          { lectures                       }
           :  vector_of_two_real;

   function TNTargetF( T       : real;
                       X       : real;
                       XPrime : real) : real;

    {-----------------------------------------------------------}
    {-        This is the second order differential equation   -}
    {-----------------------------------------------------------}
  begin
    TNTargetF :=
            - alpha[1] * X
```

```
                                - alpha[2] * Xprime
                    + alpha[1] * REF
                    + Etat_k[1]* Km/Tau *random*NOISE[1]/Kp
                    + Etat_k[2]* Km/Tau *random*NOISE[2]/Ks;

    end; { function TNTargetF }

    {$I D:\RUNGE_2.INC}     { Load procedure InitialCond2ndOrder  }

                              { Load procedure GetOutputFile  }
    {-----------------------------------------------------------------}
    procedure Init_sys_ass(var LowerLimit   : real;
                           var UpperLimit   : real;
                           var NumIntervals : integer;
                           var NumReturn    : integer;
                           var Er           : byte;
                           var alpha   : vector_of_two_real);
    {*****************************************************************}
    {*** Cette procedure initialise les parametres d'integration      ***}
    {*** d' observation et avec le pas d'integration desire.          ***}
    {*****************************************************************}

 begin
      LowerLimit := temps;
      UpperLimit := temps + deltemps;
      NumReturn := 1;
      Er := 0;
      alpha[1] := Km * Etat_k[1] /Tau;
      alpha[2] := (1+ Km * Etat_k[2]) /Tau;
    end; { procedure Initialize }
    {-------------------------------------------- -------------}


 BEGIN { SYSTEME ASSERVI }
   Init_sys_ass(LowerLimit, UpperLimit, NumIntervals, NumReturn, Er, alpha);

                                          { InitialValue, InitialDeriv,}
   InitialCond2ndOrder(LowerLimit, UpperLimit,Etat_sys_ass[1],Etat_sys_ass[2],
                       NumReturn, NumIntervals, TValues, XValues, XDerivValues,
                       Er);

     { TValues, XValues et XDerivValues contiennent les valeur de depart  }
     { a la position [0] et les nouvelles valeurs a la position [1]       }
     Etat_sys_ass[0] := TValues[1];
     Etat_sys_ass[1] := XValues[1];
     Etat_sys_ass[2] := XDerivValues[1];

END; { SYSTEME ASSERVI }
{=================================================================}

PROCEDURE MODELE_REFERENCE(var temps,deltemps :real;
                           var a: vector_of_two_real;
                           var Etat_mod_ref :  Vecteur_ETAT);
{*****************************************************************}
{*** Cette procedure calcule la valeur du modele de reference a chaque  ***}
{*** pas d' observation et avec le pas d'integration desire./          ***}
```

```
         {*********************************************************************************}

    const
      TNArraySize = 1;                 { Number of states to return }
    type
      TNvector = array[0..TNArraySize] of real;

    var
{       InitialValue, InitialDeriv : real;} { Initial values at lower limit  }
      TValues : TNvector;               { Value of T between the limits  }
      XValues : TNvector;               { Value of X at TValues  }
      XDerivValues : TNvector;          { Derivative of X at TValues  }

       function TNTargetF(T      : real;
                          X      : real;
                          XPrime : real) : real;

       {--------------------------------------------------------------}
       {-         This is the second order differential equation     -}
       {--------------------------------------------------------------}

    begin
      TNTargetF := -a[1]*X - a[2] * Xprime + a[1]*ref;
    end; { function TNTargetF }

    {$I D:\RUNGE_2.INC}      { Load procedure InitialCond2ndOrder  }
                               { Load procedure GetOutputFile  }
    {------------------------------------------------------------------}
    procedure Init_mod_ref(var LowerLimit   : real;
                           var UpperLimit   : real;
                           var NumIntervals : integer;
                           var NumReturn    : integer;
                           var Er           : byte);
    {**********************************************************************}
    {*** Cette procedure initialise les parametres d'integration       ***}
    {*** d' observation et avec le pas d'integration desire./          ***}
    {**********************************************************************}

    begin
      LowerLimit := temps;
      UpperLimit := temps + deltemps;
      NumReturn := 1;
      Er := 0;

    end; { procedure Initialize }
    {------------------------------------------------------------------}
BEGIN { MODELE REFERENCE }                      { InitialValue, InitialDeriv,}
  Init_mod_ref(LowerLimit, UpperLimit, NumIntervals, NumReturn, Er);

                                                { InitialValue, InitialDeriv,}
  InitialCond2ndOrder(LowerLimit, UpperLimit,Etat_mod_ref[1],Etat_mod_ref[2],
                      NumReturn, NumIntervals, TValues, XValues, XDerivValues,
                      Er);
     { TValues, XValues et XDerivValues contiennent les valeur de depart  }
     { a la position [0] et les nouvelles valeurs a la position [1]       }
     Etat_mod_ref[0] := TValues[1];
```

```
                    Etat_mod_ref[1] := XValues[1];
      Etat_mod_ref[2] := XDerivValues[1];

END;  ( MODELE REFERENCE )
{=========================================================================}


PROCEDURE SENSIBILITE(entree : real;       ( ENTREE QUI DETERMINE )
          var temps,deltemps :real;        ( LA SENS. P/r a[1] ou a[2] )
          var a: vector_of_two_real;
          var Etat_dY_da :  Vecteur_ETAT);    ( temps, d(Y1)_da?, d(Y2)_da? )
{*************************************************************************}
{*** Cette procedure calcule la valeur du modele de sensibilite a chaque **}
{*** pas d' observation et avec le pas d'integration desire./            **}
{*************************************************************************}

   const
     TNArraySize = 1;                  { Number of states to return }
   type
     TNvector = array[0..TNArraySize] of real;

   var
{     InitialValue, InitialDeriv : real;) { Initial values at lower limit  }
     TValues : TNvector;                 { Value of T between the limits  }
     XValues : TNvector;                 { Value of X at TValues  }
     XDerivValues : TNvector;            { Derivative of X at TValues  }

      function TNTargetF(T      : real;
                         X      : real;
                         XPrime : real) : real;
      {-------------------------------------------------------------}
      {-        This is the second order differential equation      -}
      {-------------------------------------------------------------}

   begin
     TNTargetF := -a[1]*X - a[2] * Xprime - entree;
   end; { function TNTargetF }

   {$I D:\RUNGE_2.INC}     { Load procedure InitialCond2ndOrder  }
                              { Load procedure GetOutputFile  }
   {-------------------------------------------------------------}
   procedure Init_sens(var LowerLimit   : real;
                       var UpperLimit   : real;
                       var NumIntervals : integer;
                       var NumReturn    : integer;
                       var Er        : byte);
   {*********************************************************************}
   {*** Cette procedure initialise les parametres d'integration        ***}
   {*** d' observation et avec le pas d'integration desire./           ***}
   {*********************************************************************}

   begin
     LowerLimit := temps;
     UpperLimit := temps + deltemps;
     NumReturn := 1;
     Er := 0;
```

```
    end; { procedure Initialize }
    {----------------------------------------------------------------}
BEGIN  { SENSIBILITE(entree  ,dY?_da?) }

  Init_sens(LowerLimit, UpperLimit, NumIntervals, NumReturn, Er);

                                          { InitialValue, InitialDeriv,}
  InitialCond2ndOrder(LowerLimit, UpperLimit,Etat_dY_da[1],Etat_dY_da[2],
                      NumReturn, NumIntervals, TValues, XValues, XDerivValues,
                      Er);
     { TValues, XValues et XDerivValues contiennent les valeur de depart  }
     { a la position [0] et les nouvelles valeurs a la position [1]       }
     Etat_dY_da[0] := TValues[1];
     Etat_dY_da[1] := XValues[1];
     Etat_dY_da[2] := XDerivValues[1];

END;    { SENSIBILITE(entree  ,dY?_da?) }
{==========================================================================}


PROCEDURE GAINS_Ki(var K_no           : integer;
                       Etat_sys_ass,
                       Etat_mod_ref,
                       Etat_dY_da1 ,
                       Etat_dY_da2      : VECTEUR_ETAT;
                   var GAIN           : real;
                   var Km, Tau        : real;
                   var lamda,Q        : MAT2x2;
                   var temps, deltemps : real);
{************************************************************************}
{*** Cette procedure calcule la valeur du modele des gains Ki  a chaque **}
{*** pas d' observation et avec le pas d'integration desire.           **}
{************************************************************************}

    const
      TNArraySize = 1;               { Number of states to return }
    type
      TNvector = array[0..TNArraySize] of real;

    var
      TValues : TNvector;              { Value of T between the limits  }
      XValues : TNvector;              { Value of X at TValues  }
      e                  : vector_of_two_real;


      function TNTargetF(T      : real;
                         X      : real) : real;
       {-----------------------------------------------------------------}
       {- *******>>> This is the first order differential equation  <<<**-}
       {-----------------------------------------------------------------}
```

```
                begin
      TNTargetF :=
      (  lamda[K_no,1]*(   Etat_dY_da1[1]*( Q[1,1] * e[1]+Q[1,2] * e[2])
                         + Etat_dY_da1[2]*( Q[2,1] * e[1]+Q[2,2] * e[2])
                       )
       + lamda[K_no,2]*(   Etat_dY_da2[1]*( Q[1,1] * e[1]+Q[1,2] * e[2])
                         + Etat_dY_da2[2]*( Q[2,1] * e[1]+Q[2,2] * e[2])
                       )
      )*Tau / Km;

    end; { function TNTargetF }

    {$I D:\RUNGE_1.INC}     { Load procedure                    }
    {-----------------------------------------------------------------}
    procedure Init_GAIN_Ki(var LowerLimit   : real;
                           var UpperLimit   : real;
                           var NumIntervals : integer;
                           var NumReturn    : integer;
                           var Er           : byte;
                           var e            : vector_of_two_real);
    {*****************************************************************}
    {*** Cette procedure initialise les parametres d'integration     ***}
    {*** d' observation et avec le pas d'integration desire./        ***}
    {*****************************************************************}

    begin
      LowerLimit := temps;
      UpperLimit := temps + delteps;
      NumReturn := 1;
      Er := 0;
      e[1] := Etat_mod_ref[1]-Etat_sys_ass[1];
      e[2] := Etat_mod_ref[2]-Etat_sys_ass[2];

    end; { procedure Initialize }
    {-----------------------------------------------------------------}
 BEGIN  { GAIN Ki }

   Init_GAIN_Ki(LowerLimit, UpperLimit, NumIntervals, NumReturn, Er, e);

                                              { InitialValue, InitialDeriv,}
   InitialCond1stOrder(LowerLimit, UpperLimit,GAIN,
                       NumReturn, NumIntervals, TValues, XValues,
                       Er);
    { TValues, XValues et XDerivValues contiennent les valeur de depart  }
    { a la position [0] et les nouvelles valeurs a la position [1]       }

    GAIN := XValues[1];
END;    { GAIN Ki }
{=====================================================================}
```

```
PROCEDURE CALCULE_DES_GAINS(var Etat_sys_ass,
                      Etat_mod_ref,
                      Etat_dY_da1,
                      Etat_dY_da2,
                      Etat_K          : VECTEUR_ETAT;
                 var temps,deltemps : real;
                 var Km, Tau        : real;
                 var lamda,Q        : MAT2x2);
{*******************************************************************************}
{*** Cette procedure appelle la procedure GAINS_ki pour le calcul des gains **}
{*** du controleur a chaque                                                  ***}
{*** pas d' observation et avec le pas d'integration desire.                 ***}
{*******************************************************************************}

VAR K_no : integer;
   GAIN : real;

BEGIN
  K_no := 1;
  GAIN := Etat_K[K_no];
  GAINS_Ki(K_no, Etat_sys_ass, Etat_mod_ref, Etat_dY_da1,Etat_dY_da2,
          GAIN, Km, Tau, lamda, Q, temps, deltemps);
  Etat_K[K_no] := GAIN;

  K_no := 2;
  GAIN := Etat_K[K_no];
  GAINS_Ki(K_no, Etat_sys_ass, Etat_mod_ref, Etat_dY_da1, Etat_dY_da2,
          GAIN, Km, Tau, lamda, Q, temps, deltemps);
  Etat_K[K_no] := GAIN;

  Etat_K[0] := temps + deltemps;
END;

{============================================================================}



PROCEDURE CALCULE_SYSTEME;                { PROCESSUS DE LA SIMULATION }
{****************************************************************************}
{*** Cette procedure fait calculer la valeur de chaque modele utilise  a **}
{*** chaque pas d' observation et avec le pas d'integration desire.      **}
{*** Il fait dessiner de plus la valeur calculee immediatement a chaque ***}
{*** cycle                                                              ***}
{****************************************************************************}

Begin

  DISPLAY_RESULTS(REF, Etat_sys_ass,Etat_mod_ref,
                  Etat_sys_ass_non_adap,Etat_K );

  if (not OvFlow) then
  begin
    Cycle := 0;
    if KeyPressed then read(ch);
    while (temps<limit_temps)  do
```

```
                begin
        if cycle<=NbMaxObs then  cycle := cycle + 1;
        temps := temps + deltemps;

      REFERENCE(temps, Per_Ref, REF);

      SYSTEME_ASSERVI(temps,deltemps, Km, Tau, Etat_K, Etat_sys_ass);

      if (Plot[3] or Plot[6]) then
      begin
          SYSTEME_ASSERVI(temps,deltemps, Km, Tau, Etat_K_STABLE,
                          Etat_sys_ass_non_adap);
      end;

      MODELE_REFERENCE(temps,deltemps,a, Etat_mod_ref);

      Y1 := Etat_mod_ref[1];
      Y2 := Etat_mod_ref[2];



                                  {  SENSIBILITE([Y1-REF],dY_da1); }
      SENSIBILITE(Y1-REF, temps, deltemps, a, Etat_dY_da1);

                                  { SENSIBILITE( Y2     ,dY_da2); }
      SENSIBILITE(Y2, temps, deltemps, a, Etat_dY_da2);

                          { Etat_d(Y?)/da? : temps, d(Y1)/da? d(Y1)/da? }

     CALCULE_DES_GAINS( Etat_sys_ass, Etat_mod_ref, Etat_dY_da1, Etat_dY_da2,
                        Etat_K, temps, deltemps, Km, Tau, lamda, Q);

{*     { afficher resultats }
GoToXY(2,5);     for j:=0 to 2 do  write(' ',Etat_sys_ass[j]:10:6);
GoToXY(2,6);          for j:=0 to 2 do  write(' ',Etat_mod_ref[j]:10:6); writeln;
GoToXY(2,7);          for j:=0 to 2 do  write(' k: ',Etat_K[j]:10:6); writeln;
GoToXY(2,8);          for j:=0 to 2 do  write(' ',Etat_dY_da1[j]:10:6);
GoToXY(2,9);          for j:=0 to 2 do  write(' ',Etat_dY_da2[j]:10:6); writeln;
          writeln;
     for j:=1 to 2 do
GoToXY(2,10);          for i:=1 to 2 do  writeln('  ',lamda[i,j]:10:1);
          writeln;

*}

      Resultats[1,cycle] := Etat_sys_ass[1];    { memoriser les resultats }
      Resultats[4,cycle] := Etat_sys_ass[2];
      Resultats[2,cycle] := Etat_mod_ref[1];
      Resultats[5,cycle] := Etat_mod_ref[2];
      if (Plot[3] or Plot[6]) then
      begin
          Resultats[3,cycle] := Etat_sys_ass_non_adap[1];
          Resultats[6,cycle] := Etat_sys_ass_non_adap[2];
     end;
     Resultats[7,cycle] := Etat_K[1];
     Resultats[8,cycle] := Etat_K[2];                {==================}

     DISPLAY_RESULTS(REF, Etat_sys_ass,Etat_mod_ref,
```

```
                                        Etat_sys_ass_non_adap,Etat_K );
        if OvFlow then
        begin
            GoToXY(5,13);
            Writeln('');
            GoToXY(5,14);
            Writeln('   LA SIMULATION DEMANDE DES VALEURS TROP IMPORTANTES!!!  ');
            GoToXY(5,15);
            Writeln('   VEUILLEZ REVISER VOS PARAMETRES ...                    ');
            GoToXY(5,16);
            Writeln('                                                          ');

            GoToXY(5,17);
            Writeln('   TAPEZ 2$<ENTER> POUR REVENIR AU MENU PRINCIPAL.        ');
            GoToXY(5,18);
            Writeln('');
            temps := Limit_temps;

        end;
        if KeyPressed then temps := limit_temps;
      end; { while }
   end;  { if not overflow }
End; { CALCULE_SYSTEME }
{=========================================================================}

Procedure MainSimulation;
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
{$$$ Cette procedure genere les resultats de la simulation demandee en    $$$}
{$$$ donnant la possibilite de visualiser ladite simulation.              $$$}
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}



Begin { Procedure Simulation }

      ClrScr;
      OkSimul := true;
      Set_Graphique_a_tracer;      { ramene OkSimule true or false }

      if OkSimul then
      BEGIN
          RafraichirParametres;
          INITIALIZE_GRAPHICS;

          CALCULE_SYSTEME;

          readln; readln;
          LeaveGraphic;
      END;

End;  { Procedure MainSimulation }

{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
{$$$$$$$$$                 PROGRAMME PRINCIPAL                    $$$$$$$$$}
{$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$}
```

```
            BEGIN
    TextBackground(11); Clrscr;
    TextBackground(9);TextColor(15);
    Present_title;
    Initialise_Valeur_Parametres;
    Initialise_Description_Parametres;
    RafraichirParametres;

    Repeat

    TextBackground(11); Clrscr;
    TextBackground(9);TextColor(15);
      MenuPrincipal;
      Readln(Choix);

      IF ((Choix <>'9') and (Choix<>'8') and (Choix<>'7'))
      THEN CASE Choix OF                ( Model Reference Adaptative Control )
           '1': AfficheDescriptionDeMRAC;
           '2': begin
                Repeat
                  MenuParametre;
                  Case Choixp of
                  '1': Begin
                         Affiche_Parametres(1);
                         Change_Valeur_Parametres(1);
                       End;
                  '2': Begin
                         Affiche_Parametres(2);
                         Change_Valeur_Parametres(2);
                       End;
                  '3': writeln;      { Graphisme }
                  '4': Writeln;
                  Else Choixp:='C';
                  End;
                  RafraichirParametres;
                Until Choixp='C';
                end;
           '3': MainSimulation;
           '4': Begin
                  MenuSauvegarde;
                  Case Choixms of
                  '1': SauvegardeParametres;
                  '2': SauvegardeResultats;
                  End;
                End;
           '5': Begin
                  ChercheParametres;
                End;
           END;
    Until Choix='6';
END.
```

```
{****************************************************************************}
{***                FICHIER MRAC1.PAS                                     ***}
{****************************************************************************}
{*                                                                          *}
{*                                                                          *}
{* Ce fichier sert de librairie au Programme Mrac                           *}
{****************************************************************************}
{*                                         Ecole Polytechnique de Montral   *}
{*                                                      Dcembre 1987        *}
{* Programmeur: Marc Quintal, ing.                                          *}
{* Directeur  : Dr. R.M. DeSantis                                           *}
{*                                                                          *}
{*                                                                          *}
{*   ( EXTRAIT DE MRAC.PAS )                                                *}
{*      Le programme fait la simulation d'un assservissement de position *}
{*  angulaire contol par un controleur adaptatif qui opere selon la     *}
{*  mthode du modele de reference.                                        *}
{*                                                                          *}
{*                                                                          *}
{***           Il affiche un menu specifique aux programmes de simulat-***}
{*** tion de systemes dynamiques et execute chacune des parties de ce  ***}
{*** menu.                                                              ***}
{****************************************************************************}

{****************************************************************************}


Procedure Present_title;
{****************************************************************************}
{* Cette procedure presente le titre du programme                           *}
{* procedures intermediaires: 0                                             *}
{****************************************************************************}
VAR I:INTEGER;

 BEGIN
  TextBackground(11); Clrscr;
  TextBackground(9);TextColor(15);
  I:=6;
  GOTOXY(14,I);
  WRITELN('                                                          ');
  GOTOXY(14,I+1);
  WRITELN('    **************************************************    ');
  GOTOXY(14,I+2);
  WRITELN('    *                                                *    ');
  GOTOXY(14,I+3);
  WRITELN('    *              ***>   MRAC   <***                *    ');
  GOTOXY(14,I+4);
  WRITELN('    *                                                *    ');
  GOTOXY(14,I+5);
  WRITELN('    *       SIMULATION D''UN SYSTEME ASSERVI          *    ');
  GOTOXY(14,I+6);
  WRITELN('    *        ADAPTATIVEMENT PAR LA METHODE           *    ');
  GOTOXY(14,I+7);
  WRITELN('    *            DU MODELE DE REFERENCE              *    ');
  GOTOXY(14,I+8);
```

```
              WRITELN('    *                                                    *    ');
   GOTOXY(14,I+9);
   WRITELN('    *                                         *    ');
   GOTOXY(14,I+10);
   WRITELN('    *                                         *    ');
   GOTOXY(14,I+11);
   WRITELN('    *     PAR  :    MARC QUINTAL              *    ');
   GOTOXY(14,I+12);
   WRITELN('    *               ROMANO M. DE SANTIS       *    ');
   GOTOXY(14,I+13);
   WRITELN('    *               (Dcembre 1987)            *    ');
   GOTOXY(14,I+14);
   WRITELN('    *                                         *    ');
   GOTOXY(14,I+15);
   WRITELN('    *******************************************    ');
   GOTOXY(14,I+16);
   WRITELN('                                                   ');
   GOTOXY(80,24);    READ;
END;



Procedure Change_Valeur_Parametres(q:integer);
{**********************************************************************}
{*** Avec cette procedure, l'utilisateur se deplace dans la colonne de ***}
{*** valeurs de parametres et peut les changer si desire            ***}
{**********************************************************************}
VAR
   I,                      {Cette valeur sert a positionner le curseur}
                           {apres chaque deplacement de ligne         }
   J,                      {Cette valeur sert a positionner le curseur}
                           {apres chaque deplacement de colonne       }
   RES,                    {Resultat des conversions string to real }
   AUX       : Integer;    {Variable temporaire }
   NewValue  : real;       {Nouvelle valeur du parametre }
   CHR       : Char;       {Variable entree par l'utilisateur pour le  }
                           {deplacement du curseur ou la sortie        }
   VALUE    : STRING[10];  {Variable qui lit une nouvelle valeur de para-}
                           {metre                                      }
  Maxq     : integer;      { Restriction sur la deuxieme page           }
BEGIN

    I:=1;J:=1;CHR:='0';

    if (q=2) then Maxq := Nbre_Max_Parametres-1
    else          Maxq := Nbre_Max_Parametres;
    WHILE (CHR<>'C') DO
       BEGIN
         GOTOXY(46+J*9,I+PremLinParam);
         READ(KBD,CHR);CHR:=UPCASE(CHR);
         IF (CHR=#27) THEN
                          BEGIN
                            READ(KBD,CHR);
                            IF (CHR=#72) THEN I:=I-1;
```

```
                                        IF (CHR=#80) THEN I:=I+1;
                              IF (CHR=#77) THEN J:=J+1;
                              IF (CHR=#75) THEN J:=J-1;
                              IF I>Maxq THEN I:=1;
                              IF I<1  THEN I:=Maxq;
                              IF J>N[I,q] THEN J:=1;
                              IF J<1 THEN J:=N[I,q];
                            END;
          VAL(CHR,AUX,RES);
             IF RES =0 THEN
                           BEGIN
                              TextBackground(11);TextColor(14);
                              GOTOXY(46+J*9,I+PremLinParam);
                              WRITE(CHR);
                              READ(VALUE);
                              VALUE:=CHR+VALUE;
                              VAL(VALUE,NewValue,RES);
                              If RES=0 Then P[I,q,J]:=NewValue;
                              GOTOXY(46+J*9,I+PremLinParam);
                              TextBackground(9);TextColor(15);
                              WRITE(P[I,q,J]:8:4);
                            END;
        END;
END;


Procedure Affiche_Parametres(x:Integer);
{*************************************************************************}
{*** Cette procdure affiche la description et la valeur de chaque para ***}
{*** etre dans deux tableaux distincts                                 ***}
{*************************************************************************}

VAR I,J,L,M:BYTE;
   Maxq    : integer;    { Restriction sur la deuxieme page          }
 BEGIN
    if (x=2) then Maxq := Nbre_Max_Parametres-1
    else          Maxq := Nbre_Max_Parametres;

   CLRSCR;
   GOTOXY(1,1);WRITELN('                PD ADAPTATIF SELON MODELE DE REEFERENCE                    ');

   GOTOXY(1,2);
     If x=2 Then              {x=1 est la premiere page des tableaux}
              Writeln('                          PARAMETRES DU SYSTEME                          ')
     Else
              Writeln('                          CONDITIONS DE SIMULATION                       ');

    GotoXY(1,22);
    Writeln('     => Servez-vous des flches a votre droite pour dplacer le curseur,');
    Writeln('     => Tapez un nouveau nombre quand vous le dsirez ou, ');
    WRITELN('     => Taper <C> pour Continuer...');


   I:=1;
   WHILE I<=Maxq DO
    BEGIN
```

```
                    L:=I+PremLinParam;
        GOTOXY(1,L);write(Lbel[I,x]);
        J:=1;
        WHILE J<= N[I,x] DO
          BEGIN
           M:=46+J*9;
           GOTOXY(M,L);WRITE(P[I,x,J]:8:4);
           J:=J+1;
           END;
       I:=I+1;
       END;
  End;
```

```
Procedure SetFileName(var FileName:string15; ext : string3);
{*************************************************************************}
{*** Cette procedure construit le nom d'un fichier avec l'extention    ***}
{*** fournie par l'utilisateur.    < FileName + Extention >            ***}
{*************************************************************************}

VAR
  position                  { Position du point si un point est detecte }
                            { dans le nom de fichier fourni             }
            : integer;
Begin
   position := Pos('.',FileName);          { trouver le point du FileName }

   if position=0 then
      FileName := FileName + '.' + ext;
End;
```

```
Procedure SauvegardeParametres;
{*************************************************************************}
{*** Cette procedure sauvegarde les valeurs des parametres sur fichier ***}
{*** si desire par l'utilisateur                                       ***}
{*************************************************************************}

VAR
   Fichier : File of Val;
   FileTemp :  String15;
   OK : Boolean;
```

```
            Begin
    TextbackGround(7); TextColor(1);
    ClrScr;
      GotoXY(1,5);
      Write('      Dans quel fichier voulez-vous sauvegarder vos paramtres?  ');
      Writeln;
      Write('      L''extention <.',ExtFileParam,'> est assume si aucune n''est prcise.');

      FileTemp := Fichier_Parametre;
      GoToXY(20,10); Write('===<',FileTemp,'>===> : ');
      Readln(Fichier_Parametre);

      if Fichier_Parametre='' then Fichier_Parametre := FileTemp;

      SetFileName(Fichier_Parametre,ExtFileParam);

      Assign(Fichier,Fichier_Parametre);
      {I-} Rewrite(Fichier) {I+};
      OK:=(IOresult=0);
      If Not OK Then
      begin
          Writeln('Impossibilite d''ouvrir: *',Fichier_Parametre,'*');
          Writeln('Les paramtres n'' ont pas ts sauvegards');
      end;
      If OK Then Begin
         Write(Fichier,P);

         Close(Fichier);
         Writeln;
         Writeln('Opration effectue,pressez <ENTER>.');
      End;

     Repeat Until Keypressed;
     Readln;

End;


Procedure ChercheParametres;
{*****************************************************************************}
{* Cette procedure va chercher dans un fichier deja existant des valeurs de *}
{* parametres deja emmagasinees. Fichierx est un nom fourni par l'usager    *}
{*****************************************************************************}


VAR
  Fichier : File of Val;
  FileTemp :  String15;
  Ok : Boolean;
```

```
  Begin
    ClrScr;
    GotoXY(1,5);
    Writeln('De quel fichier dsirez vous retirer vos paramtres?   ');
    Writeln;
    Writeln('L''extention <.',ExtFileParam,'> est assume si aucune n''est prcise.');

    FileTemp := Fichier_Parametre;
    GoToXY(20,10); Write('===<',FileTemp,'>===> : ');
    Readln(Fichier_Parametre);

    if Fichier_Parametre='' then Fichier_Parametre := FileTemp;

    SetFileName(Fichier_Parametre,ExtFileParam);
    Assign(Fichier,Fichier_Parametre);
    {I-} Reset(Fichier) {I+};
    Ok:=(IOresult=0);
    If Not Ok Then
    begin
      Writeln('Le fichier ',Fichier_Parametre,' n''existe pas, dsol...');
      Writeln('Les rsultats n'' ont pas ts rcuprs');
    end
    Else Begin
          Read(Fichier,P);   {Lecture des valeurs du fichier appelle et
                                        enregistrement dans le tableau P}

          Close(Fichier);
          Writeln;
          Writeln('Operation effectue.');
        End;
  Repeat Until KeyPressed;
  Readln;
End;



Procedure AfficheDescriptionDeMRAC;
{*********************************************************************}
{*** Cette procedure affiche la description du programme   MRAC   par ***}
{*** Sylvio et Romano DeSANTIS                                       ***}
{*********************************************************************}

var
  FDescription : TEXT;
  Line         : String[255];
  a            : Integer;
  Ok : Boolean;
BEGIN
  TextBackground(7);
  TextColor(1);
  ClrScr;
  Assign (FDescription,'MRAC.des');  {MRAC.des est un fichier contenant la}
                                     {description de Mrac                 }
```

```
                Reset(Fdescription);
    Ok:=(IOresult=0);
    if Ok then
    begin
      m:=1;
      Repeat
        Readln(FDescription,Line);
        Writeln(Line);
        m:=m+1;
      Until m=22;                    {Il y a 22 lignes de texte dans Mrac.des}
      GotoXY(4,23);
    end;
    Writeln('Appuyez sur une touche pour continuer.');
    Repeat Until Keypressed;
    Readln;
    Close(FDescription);

    END;



Procedure MenuSauvegarde;
{****************************************************************}
{*** Afichage du menu de sauvegarde et enregistrement d'un choix ***}
{****************************************************************}
var i,c: integer;
Begin
     TextBackground(11); Clrscr;
     TextBackground(9);TextColor(15);
     c:=11;
     i:=5;

     GoToXY(c,i);
     WRITELN ('**************************************************************');
     GoToXY(c,i+1);
     WRITELN ('*                                                            *');
     GoToXY(c,i+2);
     WRITELN ('*              SAUVEGARDE DES DONNEES                        *');
     GoToXY(c,i+3);
     WRITELN ('*                                                            *');
     GoToXY(c,i+4);
     WRITELN ('*                                                            *');
     GoToXY(c,i+5);
     WRITELN ('*        (1) Sauvegarde des paramtres                       *');
     GoToXY(c,i+6);
     WRITELN ('*        (2) Sauvegarde de la simulation pour MatLab         *');
     GoToXY(c,i+7);
     WRITELN ('*                                                            *');
     GoToXY(c,i+8);
     WRITELN ('*        <ENTER> Retour au menu Principal                    *');
     GoToXY(c,i+9);
     WRITELN ('*                                                            *');
     GoToXY(c,i+10);
     WRITELN ('*                                                            *');
     GoToXY(c,i+11);
```

```
                WRITEln ('*          VOTRE CHOIX ===>                                    *');
      GoToXY(c,i+12);
      WRITELN ('*                                                                      *');
      GoToXY(c,i+13);
      WRITELN ('****************************************************************');
      GoToXY(c+29,i+11);

      Readln(Choixms);
      Choixms := UpCase(Choixms);
End;



Procedure MenuParametre;
{*****************************************************************}
{*** Affichage du menu des parametres et enregistrement d'un choix ***}
{*****************************************************************}
var i,c: integer;
Begin
      Choixp:='C';
      TextBackground(11); Clrscr;
      TextBackground(9);TextColor(15);
      c:=11;
      i:=5;

      GoToXY(c,i);
      WRITELN ('****************************************************************');
      GoToXY(c,i+1);
      WRITELN ('*                                                                      *');
      GoToXY(c,i+2);
      WRITELN ('*          ACCES AUX PARAMETRES                                        *');
      GoToXY(c,i+3);
      WRITELN ('*                                                                      *');
      GoToXY(c,i+4);
      WRITELN ('*                                                                      *');
      GoToXY(c,i+5);
      WRITELN ('*      <1> Conditions de simulation,                                   *');
      GoToXY(c,i+6);
      WRITELN ('*      <2> Paramtres du systeme                                        *');
      GoToXY(c,i+7);
      WRITELN ('*                                                                      *');
      GoToXY(c,i+8);
      WRITELN ('*      <ENTER> Retour au menu Principal                                *');
      GoToXY(c,i+9);
      WRITELN ('*                                                                      *');
      GoToXY(c,i+10);
      WRITELN ('*                                                                      *');
      GoToXY(c,i+11);
      WRITEln ('*          VOTRE CHOIX ===>                                            *');
      GoToXY(c,i+12);
      WRITELN ('*                                                                      *');
      GoToXY(c,i+13);
      WRITELN ('****************************************************************');
      GoToXY(c+29,i+11);
```

```
                ReadIn(Choixp);
   Choixp := UpCase(Choixp);
End;
Procedure MenuPrincipal;
{**********************************************************************}
{*** Affichage du menu pricipal des operations possibles.          ***}
{**********************************************************************}
var i,c: integer;
Begin
     TextBackground(11); Clrscr;
     TextBackground(9);TextColor(15);
     c:=11;
     i:=5;
     GoToXY(c,i);
     WRITELN ('************************************************************');
     GoToXY(c,i+1);
     WRITELN ('*                                                          *');
     GoToXY(c,i+2);
     WRITELN ('*           MENU PRINCIPAL:                                *');
     GoToXY(c,i+3);
     WRITELN ('*                                                          *');
     GoToXY(c,i+4);
     WRITELN ('*                                                          *');
     GoToXY(c,i+5);
     WRITELN ('*        <1> Description du progiciel,                     *');
     GoToXY(c,i+6);
     WRITELN ('*        <2> Accs aux paramtres,                         *');
     GoToXY(c,i+7);
     WRITELN ('*        <3> Simulation,                                   *');
     GoToXY(c,i+8);
     WRITELN ('*        <4> Sauvegarde des donnes sur disquette,         *');
     GoToXY(c,i+9);
     WRITELN ('*        <5> Rcuperation des paramtres,                  *');
     GoToXY(c,i+10)    ;WRITELN ('*        <6> Sortie du programme.                          *');
     GoToXY(c,i+11);
     WRITELN ('*                                                          *');
     GoToXY(c,i+12);
     WRITELN ('*                                                          *');
     GoToXY(c,i+13);
     WRITEln ('*           VOTRE CHOIX ===>                               *');
     GoToXY(c,i+14);
     WRITELN ('*                                                          *');
     GoToXY(c,i+15);
     WRITELN ('************************************************************');
     GoToXY(c+30,i+13);
End;
```

Fichier: MRAC.DES

BIENVENUE AU PROGICIEL MRAC

Mentionnons d'abord que MRAC vient de Model Reference Adaptative Control. Ce progiciel est vou la simulation d'un asservissement de position angulaire control adaptativement par la méthode du modèle de référence.

Pour l'utiliser il suffit de revenir au menu principal et d'y choisir, dans l'ordre voulu, les différentes comandes possibles. Vous pouvez changer les conditions du système simulé et du modèle de référence par le choix <2> alors que le calcul de la simulation et la représentation graphique s'effectue par l'option <3>.

Avec le choix <4>, il est possible de conserver sur disquette soit seulement les paramètres dans un format réutilisable ultérieurement par l'option <5>, soit de sauvegarder la fois les paramètres et les résultats dans un format réutilisable par le logiciel MATLAB.

Référence: Quintal M, DeSantis R. M., Commande Adaptative d'une Articulation Robotique avec la Méthode du Modèle de Référence, Rapport Technique de l'Ecole Polytechnique de Montréal, (Janvier 1988)

EPM/RT-87/19

ASSERVISSEMENT D'UNE ARTICULATION ROBOTIQUE ELASTIQUE AVEC UN CONTROLEUR A STRUCTURE VARIABLE

Par

Caroline Serfass, Ing., M.Sc.A.

Projet dirigé par

Romano M. DeSantis, professeur

Département de Génie Electrique

Ecole Polytechnique de Montréal
Juin 1987

SOMMAIRE

L'application d'un contrôleur à structure variable à l'asservissement de position d'une articulation robotique avec un lien de transmission élastique est utilisée comme vehicule d'exploration, d'illustration et d'évaluation des modalités de conception et de fonctionnement de ce nouveau type de contrôleur. Pour ce faire nous utilisons la théorie générale des systèmes à structure variable (méthode de Utkin), la notion de modèle d'état et la technique du positionnement des pôles; l'étude de simulation est developpée en utilisant un logiciel écrit en Turbo Pascal et opérant sur IBM PC; l'analyse comparative repose sur une étude de la sensibilité des trois contrôleurs aux perturbations et aux variations des paramètres du système.

Les resultats obtenus mettent en evidence un certain nombre d'avantages du contrôleur à structure variable par rapport aux contrôleurs actuellement utilisés. La presence d'oscillations rapides et de hauts niveaux de tension dans la commande, plus une tendance à l'instabilité lors de la presence d'un delai de boucle, suggèrent néanmoins que des améliorations substantielles sont nécessaires avant que ce nouveau type de contrôleur puisse être considerer un candidat serieux au remplacement de ses concurrents.

ABSTRACT

The controller design of a position servo with an elastic link is used as a vehicle to discuss performance sensitivity aspects of a sliding mode controller. The sliding mode controller is designed using the by now classical procedure proposed by Utkin, namely: -the characterization of the desired system dynamic behavior in terms of an assigned sliding mode; -the choice of a controller structure which guarantees the existence and the stability of the sliding mode; -the determination of the controllers parameters which will steer the plant state to the sliding mode. Performance sensitivity is considered in relation to plant parameters variations, the presence of an external disturbance, the introduction of a delay in the control action.

The ensuing results are to the effect that in spite of the advantages offered by the sliding mode controller, the presence of chattering and a tendency to unstable oscillations suggest that further improvements are needed before this type of controller may replace its more standard counterparts.

REMERCIEMENTS

Ce projet a été développé dans le cadre du projet FCAR-CRP-378-76, "Analyse et Commande des Processus Industriels par Ordinateur: Application au Réseaux Electrique", financé par le Ministère de l'Education du Québec. Il a également bénéficié du support financier de l'octroi A-8244, "Sensibilité, Stabilité et Commande Optimale des Systèmes", du Conseil de Recherches en Sciences et en Génie du Canada.

L'auteur tient à exprimer vivement sa reconnaissance envers son directeur de recherche, Dr. Romano M. De Santis, professeur du département de Génie Electrique, pour avoir proposé le thème du projet et pour avoir supervisé son développement.

Un grand remerciement est dû au professeur Rozinoer, directeur du Département d'Automatique de l'Ecole Centrale de Lyon, qui l'a soutenue et aidée dans ses démarches d'inscription à l'Ecole Polytechnique de Montréal. L'auteur tient à remercier également le Dr. R.P. Malhamé, professeur au département de Génie électrique, pour lui avoir facilité la recherche bibliographique.

Il lui tient enfin tout particulièrement à coeur de remercier Lamia Orfali et Raouf Naggar ainsi que leur famille, pour leur présence chaleureuse lors de la phase finale de ce travail.

TABLE DES MATIÈRES

Page

SOMMAIRE iii

ABSTRACT v

REMERCIEMENTS vii

LISTE DES FIGURES xii

LISTE DES TABLEAUX xv

CHAPITRE I - INTRODUCTION 1

1.1 Introduction 1

1.2 Organisation du mémoire..... 4

1.3 Contrôleurs à structure variable..... 5

CHAPITRE II - MODÉLISATION DE L'ARTICULATION ROBOTIQUE ÉLASTIQUE ET DE L'ASSERVISSEMENT..... 7

2.1 Le système en étude..... 7

2.2 Équations physiques du système..... 9

2.3 Choix des valeurs des paramètres..... 10

2.4 Modèle du système avec variables d'état.. 11

2.5 Le système de contrôle..... 13

CHAPITRE III - DESIGN DU CONTRÔLEUR À STRUCTURE VARIABLE..... 15

3.1 La méthode utilisée..... 15

3.2 Procédure de conception..... 15

3.3 Application à l'articulation robotique... 20

CHAPITRE IV - DÉVELOPPEMENT D'UN LOGICIEL DE SIMULATION..... 23

4.1 Introduction 23

4.2 Structure du logiciel vue par..... l'utilisateur..... 25

4.3 Structure interne du logiciel..... 28

CHAPITRE V - COMPORTEMENT DU CONTRÔLEUR À.....
STRUCTURE VARIABLE..... 35

5.1 Positionnement des pôles..... 35
5.2 Choix des gains..... 37
5.3 Résultats..... 39

CHAPITRE VI - ANALYSE COMPARATIVE DES TROIS.....
CONTRÔLEURS..... 42

6.1 Introduction..... 42
6.2 Systèmes de référence..... 43
6.3 Performances statiques et dynamiques..... 44
6.4 Sensibilité à la variation des paramètres 46
6.5 Analyse de la sensibilité à une.....
perturbation..... 53
6.6 Analyse de la sensibilité au délai..... 57

CHAPITRE VII - CONCLUSION..... 63

7.1 Simplicité de conception et fiabilité.... 63
7.2 Résultats obtenus..... 63
7.3 Extensions possibles..... 65

BIBLIOGRAPHIE 67

LISTE DES FIGURES

Figure		Page
2.1	Articulation robotique élastique entraînée par un moteur à courant continu.....	7
2.2	Mise en évidence de la transmission élastique du robot Smart de Comau (en-bas à gauche).....	8
2.3	Représentation du système asservi.....	13
3.1	Représentation fonctionnelle du contrôleur à structure variable.....	16
3.2	Les différentes étapes de la conception du contrôleur à stucture variable.....	18
4.1	Déroulement chronologique du programme vu par l'utilisateur.....	27
4.2	Structure interne du programme.....	33

4.3 Structure interne de la boucle de calcul..... 34

5.1 Essai SSV: valeurs nominales des gains..... 40

6.1 Performance dynamique PID versus structure variable..... 45

6.2 Performance dynamique retour linéaire versus structure variable..... 46

6.3 Variation de Km: retour linéaire d'état versus structure variable..... 48

6.4 Variation de tau1: retour linéaire d'état versus structure variable..... 48

6.5 Variation de j2: retour linéaire d'état versus structure variable..... 49

6.6 Variation de Km: essai nominal versus gains supérieurs..... 51

6.7 Variation de tau1: essai nominal versus gains supérieurs..... 51

6.8 Variation de j2: essai nominal versus gains supérieurs..... 52

6.9 Sensibilité à une perturbation pour le contrôleur PID..... 54

6.10 Sensibilité à une perturbation pour le contrôleur par retour linéaire d'état..... 54

6.11 Sensibilité à une perturbation pour le contrôleur à structure variable..... 55

6.12 Amélioration du contrôleur à structure variable face à une perturbation..... 57

6.13 Effet d'un délai sur le contrôleur PID, à retour linéaire d'état et à structure variable..... 59

6.14 Comportement du système à structure variable avec gains plus élevés face à un délai..... 60

6.15 Effet d'un délai sur un contrôleur à variable peu sensible aux perturbations..... 62

LISTE DES TABLEAUX

Tableau		Page
2.1	Paramètres physiques du système.....	9
4.1	Paramètres du système physique.....	29
4.2	Paramètres de simulation.....	29
4.3	Paramètres des différents contrôleurs.....	30
5.1	Choix des pôles du contrôleur à structure variable.....	36
5.2	Choix des gains du contrôleur à structure variable avec les valeurs nominales des pôles	38
5.3	Paramètres de l'essai nominal.....	39
6.1	Paramètres des systèmes de référence.....	43

CHAPITRE I

INTRODUCTION

1.1 Introduction

Dans beaucoup de systèmes de contrôle il est important que l'opération du système soit insensible aux variations des paramètres et de la charge. Avec les contrôleurs actuellement disponibles, cette exigence peut être difficile à satisfaire lors de la présence de larges variations de paramètres et lorsque ces mêmes variations ne sont pas facilement prévisibles. Ces éventualités sont assez fréquentes en robotique par exemple, où il peut arriver qu'un moment d'inertie d'un robot par rapport à un axe de rotation varie de façon importante suivant la longueur du bras, la charge ou simplement la configuration du robot [6] [7] [8]. Or l'insensibilité aux variations des paramètres constitue une condition essentielle à l'élaboration de robots plus performants.

En vue d'améliorer cette situation, de nombreux articles scientifiques ont récemment attiré l'attention sur les avantages potentiels offerts par les contrôleurs à structure variable [1,2,3,4,5,6,7,8,9,14]. Des études

illustrant la supériorité de leurs performances statiques et dynamiques par rapport aux contrôleurs existants ont été développées [3,5,6,7,8,9]; de nombreuses procédures conceptuelles et des algorithmes de calcul nécessaires pour leur conception ont été proposés [3,5,6,7,8,9,14]; de nombreuses applications dans le domaine de l'électrotechnique [3] [9], ainsi que dans celui de la robotique [6] [7] [8] ont été illustrées.

Malgré tout cet enthousiasme et toute cette richesse de documentation, force est de constater que la réalité industrielle manifeste une forte résistance à remplacer les contrôleurs conventionnels par les contrôleurs à structure variable; en effet, le nouveau type de contrôleur n'existe que sur papier. Ce phénomène s'explique en partie par le fait que des difficultés ont été expérimentées en ce qui concerne la sensibilité des systèmes à structure variable aux retards [8] et une certaine tendance à entrer en oscillations [6] [7] [8]. Par ailleurs, une autre difficulté vient du fait que les contrôleurs à structure variable n'ont pas encore été suffisamment étudiés pour que des conclusions définitives puissent avoir été établies.

Ainsi plusieurs questions restent ouvertes: la méthodologie de conception est-elle véritablement au point pour nous procurer de façon efficace des solutions fiables

et satisfaisantes? La supériorité théorique de leur sensibilité face à la variation des paramètres se concrétise-t-elle en une supériorité sur le banc d'essai? Les niveaux de contrôle requis et les retards de boucle admissibles sont-ils compatibles avec les caractéristiques offertes par les capteurs, les actionneurs et les convertisseurs disponibles sur le marché?

L'objectif de ce projet est de contribuer à une réponse à ces questions. Pour ce faire, nous considérons l'application du contrôleur à structure variable à l'asservissement de position d'une articulation robotique avec un lien de transmission élastique [13]. L'intérêt d'une telle application est double. D'une part, tout en étant d'une complexité modérée, la nature du système asservi nous procure un banc d'essai idéal pour tester d'une façon claire et concluante les traits saillants du contrôleur en étude. D'autre part, l'application envisagée nous permet de contribuer à une meilleure compréhension d'un système d'une grande importance pratique dans le domaine de l'automatisation et de la robotique.

La réalisation du projet prévoit le développement de plusieurs étapes: la modélisation du système asservi, une illustration des caractéristiques générales d'un contrôleur à structure variable, la conception d'un contrôleur pour une étude de cas, une étude de ce

contrôleur par voie de simulation et une analyse comparative des performances du nouveau contrôleur par rapport à celles d'un contrôleur PID et d'un contrôleur linéaire avec rétroaction d'état. La conception du contrôleur à structure variable utilise les théories générales des systèmes à structure variable [1],[2], la notion de modèle d'état [10] et la technique de positionnement des pôles [3] [10]. Pour la simulation, nous avons développé un logiciel spécifique, ROBEL, sur un micro-ordinateur IBM-PC, en langage Turbo-Pascal [16] [17]. L'analyse comparative se propose d'étudier le comportement des trois contrôleurs, d'une part sous des conditions nominales, d'autre part face à une perturbation constante de la commande, à un délai et à des variations dans les paramètres du système.

1.2 Organisation du mémoire

Le mémoire est divisé en 7 chapitres. Le chapitre II présente la modélisation de l'articulation robotique avec un lien de transmission élastique; le chapitre III est consacré au design du contrôleur à structure variable; au chapitre IV, une description synthétique du logiciel ROBEL nous permet de cerner ses possibilités et de comprendre le déroulement d'une simulation; le chapitre V est consacré à

l'étude du comportement du contrôleur à structure variable; le chapitre VI propose une analyse comparative des résultats obtenus avec les trois contrôleurs: PID, par retour linéaire d'état et à structure variable; finalement, les conclusions de ce travail sont tirées au chapitre VII, où nous suggérons également quelques extensions possibles.

Pour le bénéfice du lecteur, nous avons adopté une présentation condensée et un style concis. Des développements de nature complémentaire, qui malgré leur importance technique auraient pu alourdir le texte principal, ont été relégués dans un cahier d'annexes.

1.3 Contrôleurs à structure variable

Les fondements de la théorie des systèmes à structure variable (SSV) ont été développés dans les 25 dernières années presque exclusivement en URSS. Ces systèmes diffèrent des autres systèmes de contrôle par le fait que des changements peuvent se produire dans la structure du système pendant son opération. Le moment où ces changements se produisent ainsi que le type de structure obtenue sont déterminés en fonction de l'état actuel du système [1], [2].

Divers principes de design de systèmes à structure

variable [4] ont été proposés:

- le premier consiste à prendre des "morceaux" de trajectoires d'état des structures existantes et de les raccorder suivant un certain critère, dans le but d'obtenir une trajectoire satisfaisante.

- le second consiste à rechercher des trajectoires d'état de l'un des sous-systèmes, qui possèdent de bonnes performances dynamiques, et d'assurer qu'à partir d'un certain moment, le point de fonctionnement se déplace sur ces trajectoires.

- enfin, le dernier consiste à créer des <u>modes glissants</u>. Ces modes se produisent dans l'espace d'état du système, le long de surfaces où le contrôle est discontinu.

Les espoirs placés dans les deux premiers principes se sont avérés infructueux et on a limité leur application à des systèmes du 2e ordre [4]. Le seul principe de contrôle à structure variable prometteur consiste à introduire délibérément des modes glissants dans le système. Les résultats obtenus à ce jour permettent de considérer le contrôle à structure variable comme l'une des branches d'avenir de la théorie de contrôle [3], [4].

CHAPITRE II

MODÉLISATION DE L'ARTICULATION ROBOTIQUE ÉLASTIQUE ET DE L'ASSERVISSEMENT

2.1 Le système en étude

Le système que nous utiliserons comme banc d'essai pour le développement de notre étude des contrôleurs à structure variable est représenté par une articulation robotique avec un lien de transmission élastique. La figure 2.1 nous montre les différentes composantes de ce système: un moteur électrique à courant continu, deux poulies, une courroie de transmission élastique et la charge.

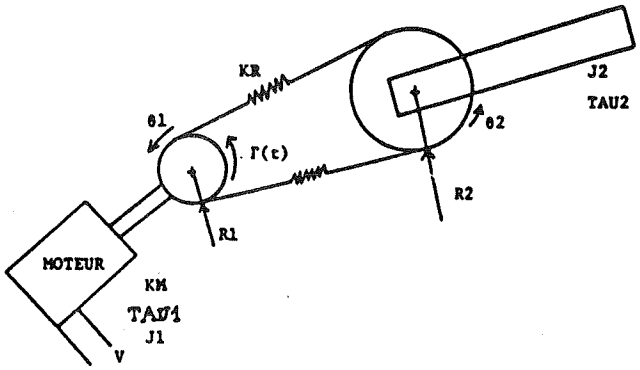


Figure 2.1: Articulation robotique élastique entraînée par un moteur à courant continu.

Des articulations robotiques de ce type sont souvent utilisées dans la construction de robots industriels, par exemple dans le robot italien Smart de Comau, dont nous présentons une photographie à la figure 2.2 [19].

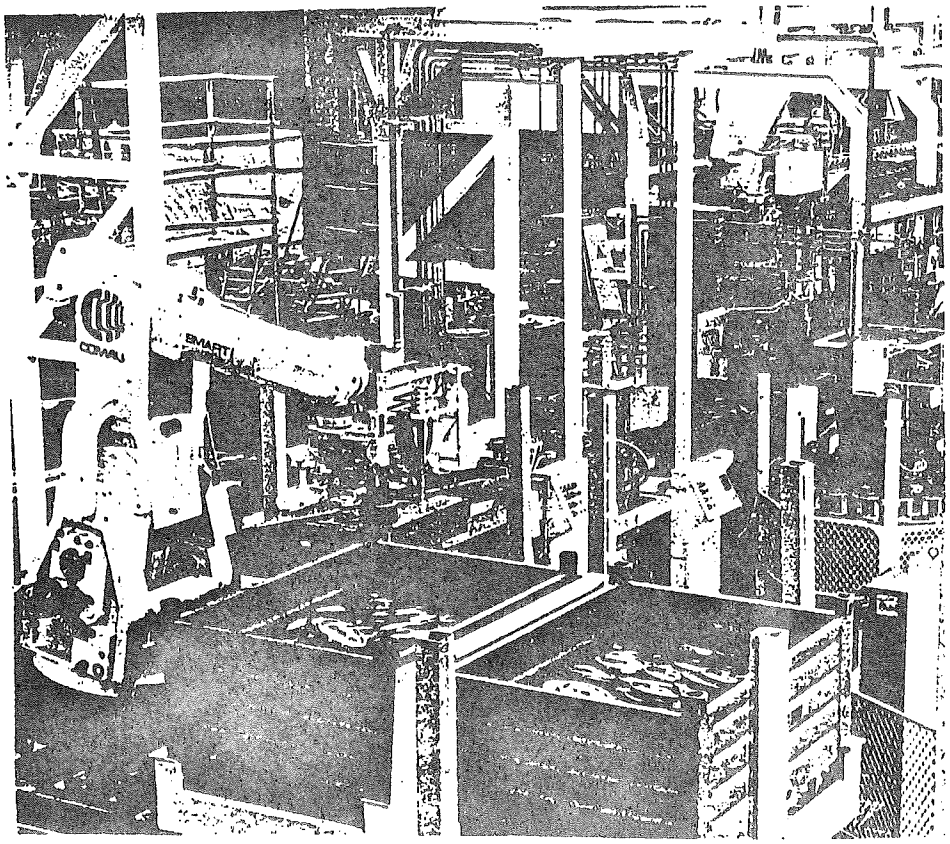


Figure 2.2: Mise en évidence de la transmission élastique du robot Smart de Comau (en-bas à gauche)

2.2 Équations physiques du système

Les notations indiquées à la figure 2.1 sont V la tension appliquée aux bornes de la bobine du rotor, R la résistance et L l'inductance de cette bobine. $\theta 1$ et $\theta 2$ désignent respectivement l'angle de l'axe du moteur et l'angle de l'articulation robotique. De plus, nous utiliserons les notations du tableau 2.1.

$\tau 1$	constante de temps du moteur
Km	gain statique du moteur
J1	moment d'inertie du moteur
R1	rayon de couplage côté moteur
KR	constante d'élasticité
NT	rapport de transmission
J2	moment d'inertie du robot
$\tau 2$	constante de temps du robot
R2	rayon de couplage côté robot

Tableau 2.1: paramètres physiques du système

L'application des lois de la dynamique au moteur et à l'articulation robotique nous donne respectivement les équations suivantes (plus de détails pourront être trouvés dans [10]):

$$\frac{d^2\theta 1}{dt^2} = \frac{Km*V}{\tau 1} - \frac{1}{\tau 1} * \frac{d\theta 1}{dt} - \frac{KR*R1^2}{J1} * (\theta 1 - NT*\theta 2) \qquad (2.1)$$

$$NT* \frac{d^2\theta 2}{dt^2} = \frac{KR*R2^2}{J2} * (\theta 1 - NT*\theta 2) - \frac{1}{\tau 2} * \frac{d\theta 2}{dt} \qquad (2.2)$$

2.3 Choix des valeurs des paramètres

Pour rendre notre étude plus concrète, nous utiliserons des paramètres d'articulation robotique compatibles avec la géométrie et la distribution des masses qui caractérisent l'articulation verticale du robot Smart présenté à la figure 2.2 [19]. Ceci nous amène à considérer les valeurs des paramètres suivantes:

1. Paramètres relatifs à la charge

- moment d'inertie de la charge: $J2 = 75$ kg.m²
- constante de temps de la charge: $\tau 2 = 10$ s
- rayon de couplage côté charge: $R2 = 0.4$ m

2. Paramètres relatifs à la transmission

- rapport de transmission: $NT = 10$
- constante d'élasticité de la courroie: $KR = 1250000$ N/m

3. Paramètres relatifs au moteur

- moment d'inertie du moteur : $J1 = 0.5$ kg.m²
- gain statique du moteur: $Km = 1.5$ rad/s.V.
- constante de temps du moteur: $\tau 1 = 0.05$ s

2.4 Modèle du système avec variables d'état

En posant

$$x1 \triangleq \theta 1 \qquad x2 \triangleq \frac{d\theta 1}{dt}$$

$$x3 \triangleq NT*\theta 2 \qquad x4 \triangleq NT*\frac{d\theta 2}{dt} \qquad u \triangleq V$$

nous obtenons la représentation du système par le modèle d'état suivant:

$$\begin{cases} \dot{x}1 = x2 \\ \dot{x}2 = -\dfrac{KR*R1^2}{J1} * x1 - \dfrac{1}{\tau 1} * x2 + \dfrac{KR*R1^2}{J1} * x3 + \dfrac{Km}{\tau 1} * u \\ \dot{x}3 = x4 \\ \dot{x}4 = -\dfrac{KR*R2^2}{J2} * x1 - \dfrac{KR*R2^2}{J2} * x3 - \dfrac{1}{\tau 2} * x4 \end{cases} \qquad (2.3)$$

En posant

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\dfrac{KR*R1^2}{J1} & -\dfrac{1}{\tau 1} & \dfrac{KR*R1^2}{J1} & 0 \\ 0 & 0 & 0 & 1 \\ \dfrac{KR*R2^2}{J2} & 0 & -\dfrac{KR*R2^2}{J2} & -\dfrac{1}{\tau 2} \end{bmatrix} \qquad (2.4)$$

$$B = \begin{bmatrix} 0 \\ \dfrac{Km}{\tau 1} \\ 0 \\ 0 \end{bmatrix} \quad (2.5) \qquad \text{et} \qquad X = \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} \quad (2.6)$$

le système (2.3) devient:

$$\dot{X} = A*X + B*u \qquad (2.7)$$

2.5 Le système de contrôle

Le contrôle de l'articulation robotique avec lien de transmission élastique est réalisé en utilisant les éléments suivants: un moteur électrique à courant continu, deux potentiomètres, deux tachomètres, un convertisseur analogique/numérique, un microordinateur, un convertisseur numérique/analogique et un amplificateur de puissance. Les interactions entre ces différents éléments sont illustrées à la figure 2.3.

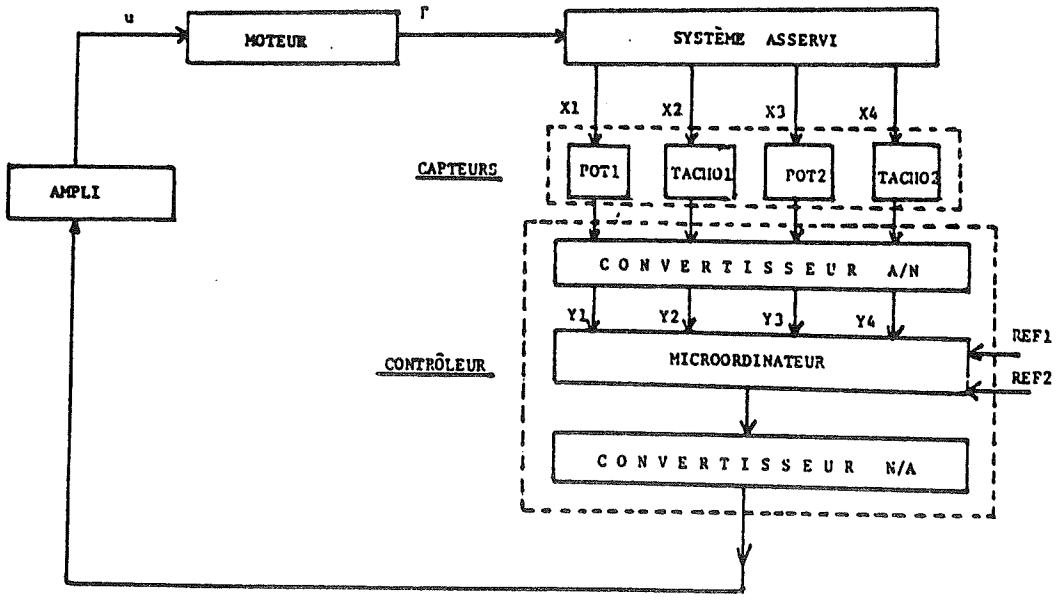


Figure 2.3: Représentation du système asservi

En pratique les notations de la figure 2.3 ont la signification suivante:

-u est la tension appliquée au moteur et $|u_{max}|=125V$

-x1,x2,x3,x4 sont les variables d'état telles qu'elles ont été définies à la section 2.4

-y1,y2,y3,y4 sont ces mêmes variables après avoir été lues par les capteurs et numérisées

-Γ est le couple développé sur l'axe du moteur

-ref1 et ref2 sont les positions désirées respectivement pour l'angle de l'axe du moteur et celui de l'articulation robotique

Finalement le système global sera décrit par l'équation (2.3) $\dot{X} = A * X + B * u$ où $u = f(X)$ représente la loi de contrôle envisagée, qui pourra être PID, par retour linéaire d'état ou à structure variable.

CHAPITRE III

DESIGN DU CONTROLEUR A STRUCTURE VARIABLE

3.1 La méthode utilisée

Le problème de contrôle qui nous concerne peut être énoncé comme suit:

étant donné un système décrit par le modèle d'état

$$\dot{X} = A*X + B*u$$

où A est une matrice de dimension n*n et B une matrice de dimension n*1, X est le vecteur d'état et u=f(X) la loi de commande, définir u de sorte que l'état X atteigne un état de référence assigné, avec un temps de réponse et une précision satisfaisants.

3.2 Procédure de conception

Entre la variété des procédures disponibles pour solutionner ce problème en utilisant la théorie des systèmes à structure variable, nous adopterons celle décrite par A.Y.Sivaramakrishnan dans [3]. Cette procédure, basée sur la technique du positionnement des pôles, utilise un contrôleur dont la structure est illustrée à la figure 3.1.

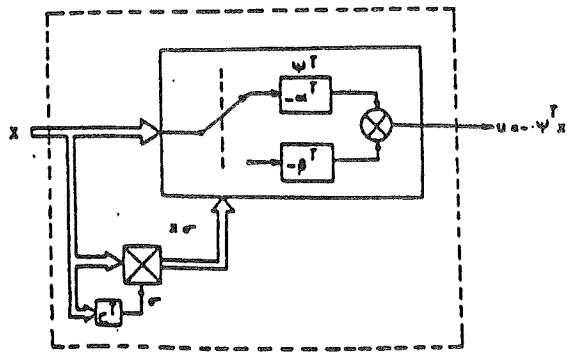


Figure 3.1: Représentation fonctionnelle du contrôleur à structure variable.

La stratégie de commande utilisée par ce contrôleur est donnée par:

$$u = \sum_{i=1}^{n} \psi_i * X_i \qquad (3.1)$$

où les ψ_i représentent des gains qui varient comme suit:

$$\psi_i = \begin{cases} \alpha_i & \text{si} \quad x_i * \sigma > 0 \\ \beta_i & \text{si} \quad x_i * \sigma < 0 \end{cases} \qquad \text{pour} \quad i = 1..n \qquad (3.2)$$

avec

$$\sigma \triangleq C * X \qquad (3.3)$$

$$C \triangleq [c1 \quad ..... \quad cn] \qquad (3.4)$$

n est l'ordre du système. C est appelé vecteur de commutation et la fonction σ prend le nom de <u>fonction du mode glissant</u>. L'équation $\sigma=0$ est celle d'un hyperplan dénommé <u>hyperplan de commutation</u>. C'est en effet en fonction de la position du vecteur d'état X par rapport à cet hyperplan que s'opère le changement de structure.

D'habitude cette stratégie conduit à une situation où l'état du système est contraint à être localisé sur l'hyperplan de commutation (<u>dynamique en mode glissant</u>). Lorsque cette contrainte est satisfaite, le comportement dynamique du système est décrit en termes de n-1 pôles $p_1 \ldots p_{n-1}$. Ces pôles sont une fonction de C, α_i et β_i.

Le design du contrôleur est effectué en passant par les étapes suivantes (figure 3.2):

1. <u>Choix des pôles qui caractérisent la dynamique désirée</u>

Il s'agit de choisir les (n-1) pôles du mode glissant $p_1 \ldots p_{n-1}$. Cette étape est similaire à celle qui est utilisée dans l'approche par rétroaction d'état par la technique du positionnement des pôles [10] (voir en annexe A).

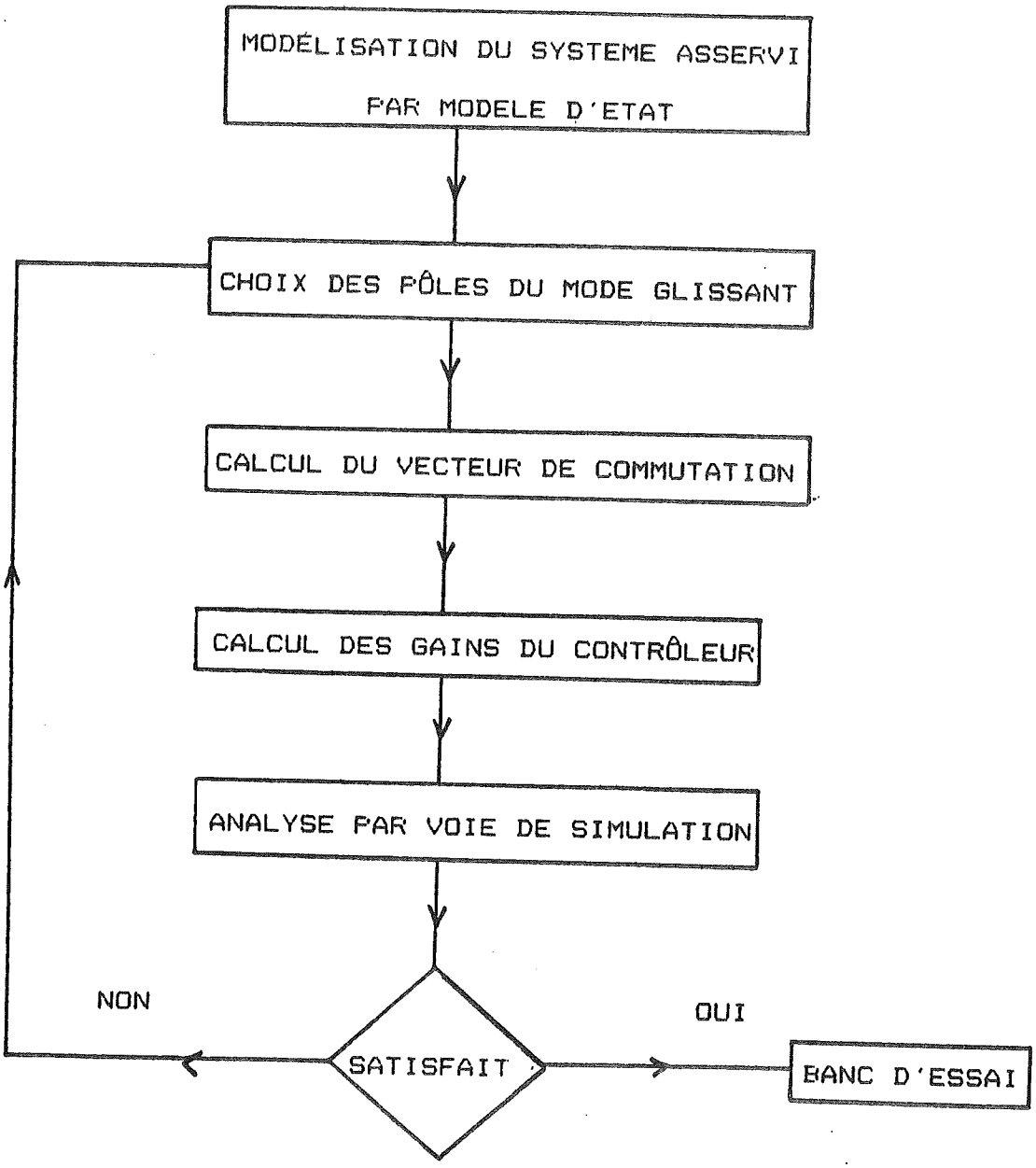


Figure 3.2: Les différentes étapes de la conception du contrôleur à structure variable

2. <u>Détermination de la matrice C correspondant au choix des pôles effectué</u>

On détermine d'abord une matrice M non singulière telle que:

$$M*B = \begin{bmatrix} 0 \\ . \\ 0 \\ b \end{bmatrix}$$ où b est un réel non nul.

En utilisant la transformation définie par cette matrice et avec Y=M*X, on peut écrire:

$$\begin{bmatrix} \dot{Y}1 \\ \dot{y}2 \end{bmatrix} = \begin{bmatrix} A11 & A12 \\ A21 & a22 \end{bmatrix} * \begin{bmatrix} Y1 \\ y2 \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} * u \qquad (3.5)$$

où les sous-matrices A11, A21 et A12 sont de dimensions respectives (n-1)*(n-1), 1*(n-1) et (n-1)*1 et a22 est un scalaire.

On détermine ensuite une matrice C11 telle que:

$$dét(sI-A11+A12*C11)=(s-p_1)*...(s-p_{n-1})$$

Le vecteur de commutation C est donné par:

$$C = [C11 \quad 1] * M \qquad (3.6)$$

3. Détermination des gains α_i et β_i

Les valeurs de α_i et β_i sont déterminées de façon à satisfaire les relations suivantes:

$$\alpha_i < l_i \tag{3.7}$$

$$\beta_i > l_i \tag{3.8}$$

Les symboles l_i marquent les limites où doivent se situer les gains du contrôleur pour garantir le fonctionnement du système en mode glissant [1] [2].

$$l_i \triangleq - \frac{A_i^T * C}{B^T * C} \tag{3.9}$$

où A_i (i=1...n) est le ième vecteur colonne de la matrice d'état A donnée par (2.4).

3.3 Application à l'articulation robotique

L'application de la procédure de conception à l'articulation robotique modélisée au chapitre II (équations (2.4) à (2.6)) se concrétise comme suit:

1. Choix des pôles

Pour choisir les valeurs numériques des pôles du mode glissant, nous nous sommes inspirés des résultats

obtenus avec les contrôleurs PID et par retour linéaire d'état. Ainsi, l'étude préalable du système avec un contrôleur PID nous fournit le point de départ d'une approche systématique (voir résultats en annexe C). En effet, par une technique bien connue [10], exposée en annexe C, nous commençons par ajuster les gains du contrôleur PID en fonction de certaines spécifications (coefficient d'amortissement, temps de réponse, commande maximale), puis, lorsque nous obtenons une réponse satisfaisante, nous calculons les pôles du système en boucle fermée correspondants. Nous nous inspirons de ces résultats pour positionner les pôles du contrôleur par retour linéaire d'état [10], comme exposé en annexe A, et enfin ceux du mode glissant pour le contrôleur à structure variable. On note ces derniers pôles p1, p2, p3.

2. Détermination de la matrice C

Nous choisissons une matrice M 4*4 non singulière, la plus simple possible

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

En appliquant (3.6), nous obtenons en posant

C=[c1 c2 c3 c4]:

$$c1 = -(p1+p2+p3)-(1/\tau 2) \tag{3.10}$$

$$c2 = 1 \tag{3.11}$$

$$c3 = -(J2/(KR*R2^2))*(p1*p2*p3) - c1 \tag{3.12}$$

$$c4 = (J2/(KR*R2^2))*(p3*p1+p3*p2+p1*p2) -1-(c1*J2)/(c2*KR*R2^2) \tag{3.13}$$

3. <u>Détermination des gains α_i et β_i</u>

Nous commençons par calculer les limites l_i et nous proposons de choisir les gains α_i et β_i symétriques par rapport aux l_i. Nous verrons, à la lumière des résultats exposés au chapitre V, que d'un point de vue pratique, le choix des gains du système résulte d'un compromis entre une réponse satisfaisante et une sollicitation raisonnable du moteur.

En appliquant (3.9), nous obtenons:

$$l1 = (\tau 1/Km)*((KR*R1^2/J1)-(c4*KR*R2^2/J2)) \tag{3.14}$$

$$l2 = (\tau 1/Km)*((1/\tau 1)-c1) \tag{3.15}$$

$$l3 = (\tau 1/Km)*(c4*(KR*R2^2/J2)-(KR*R1^2/J1)) \tag{3.16}$$

$$l4 = (\tau 1/Km)*((c4/\tau 2)-c3) \tag{3.17}$$

CHAPITRE IV

DÉVELOPPEMENT D'UN LOGICIEL DE SIMULATION

4.1 Introduction

La fonction du logiciel ROBEL (joint ROBotique ELastique) est de nous permettre de simuler le comportement dynamique d'une articulation robotique élastique dont la position est asservie par un contrôleur PID, par retour linéaire d'état ou à structure variable, en tenant compte éventuellement d'une perturbation constante et d'un retard dans la commande.

Ses caractéristiques d'opération sont les suivantes:

- capacité à simuler le comportement statique et dynamique de l'articulation robotique décrite au chapitre II
- acquisition interactive des paramètres de simulation
- représentation numérique et graphique interactive des résultats
- création et lecture d'une banque de données relative aux expériences de simulation
- execution rapide

Ce logiciel a été écrit en langage Turbo-Pascal [16], [17] sur un micro-ordinateur IBM-PC. Sa structure générale est constituée d'un programme principal et d'un certain nombre de sous-programmes ou modules, dont chacun est voué à une fonction essentielle détaillée dans la section 4.3. Parmi les plus importantes, nous citons le module d'intégration numérique, basé sur la méthode d'intégration numérique Runge-Kutta d'ordre 4 présentée en annexe D.

L'élaboration de ce logiciel résulte de l'interaction des idées et des travaux de plusieurs personnes. Il nous fait plaisir de mentionner ici entre autres: André Terroux et Romano M. De Santis [12] (Asserv. pos: un programme pour la simulation de la commande d'un joint robotique), Pierre Radziszewski, Valérie Tremblay et Romano M. De Santis [13] (Simulation du comportement dynamique de la boucle de commande de l'articulation verticale de Rhino XR1), Anh-Tu Nguyen [15] (Étude de sensibilité du comportement dynamique d'un manipulateur robotique) et Mudit Kumar [14] (A Position Control System with Disturbance Rejection using Sliding Mode).

Dans la section 4.2 nous présentons le logiciel tel que vu par l'utilisateur, alors qu'au 4.3 nous en détaillons la structure interne. Nous espérons ainsi

donner au lecteur une vue d'ensemble qui lui permettra de mieux cerner le problème traîté.

4.2 Structure du logiciel vue par l'utilisateur

La succession chronologique des différentes étapes rencontrées par l'utilisateur lors de l'exécution du programme **ROBEL** est représentée à la figure 4.1. L'opérateur répond aux questions du logiciel pour avancer dans l'exécution. Nous abordons brièvement chaque option dans les paragraphes qui suivent.

4.2.1 Assignation des paramètres

L'utilisateur choisit de fixer les paramètres de la simulation soit par défaut, soit par lecture d'un fichier stocké sur disquette lors d'une précédente exécution.

4.2.2 Modification des paramètres

L'opérateur peut accéder à 3 groupes de paramètres distincts:

- les paramètres du système physique (tableau 4.1)
- les paramètres de simulation (tableau 4.2)
- les paramètres du contrôleur choisi (tableaux 4.3.a,b,c,d).

4.2.3 Simulation

Lors de la simulation, l'utilisateur voit apparaître sur l'écran des informations concernant le comptage des intervalles d'intégration et l'évolution de certaines variables (temps, commande, position).

4.2.4 Tracé graphique

Lorsque les calculs sont terminés, l'opérateur voit 7 courbes se succéder sur l'écran:

- tetapoint1 en fonction de teta1 (plan de phase côté moteur)
- teta1 en fonction du temps
- tetapoint2 en fonction de teta2 (plan de phase côté charge) .
- teta2 en fonction du temps
- commande en fonction du temps
- tetapoint1 en fonction du temps
- tetapoint2 en fonction du temps

Il a alors la possibilité de les imprimer toutes ou seulement les 5 premières.

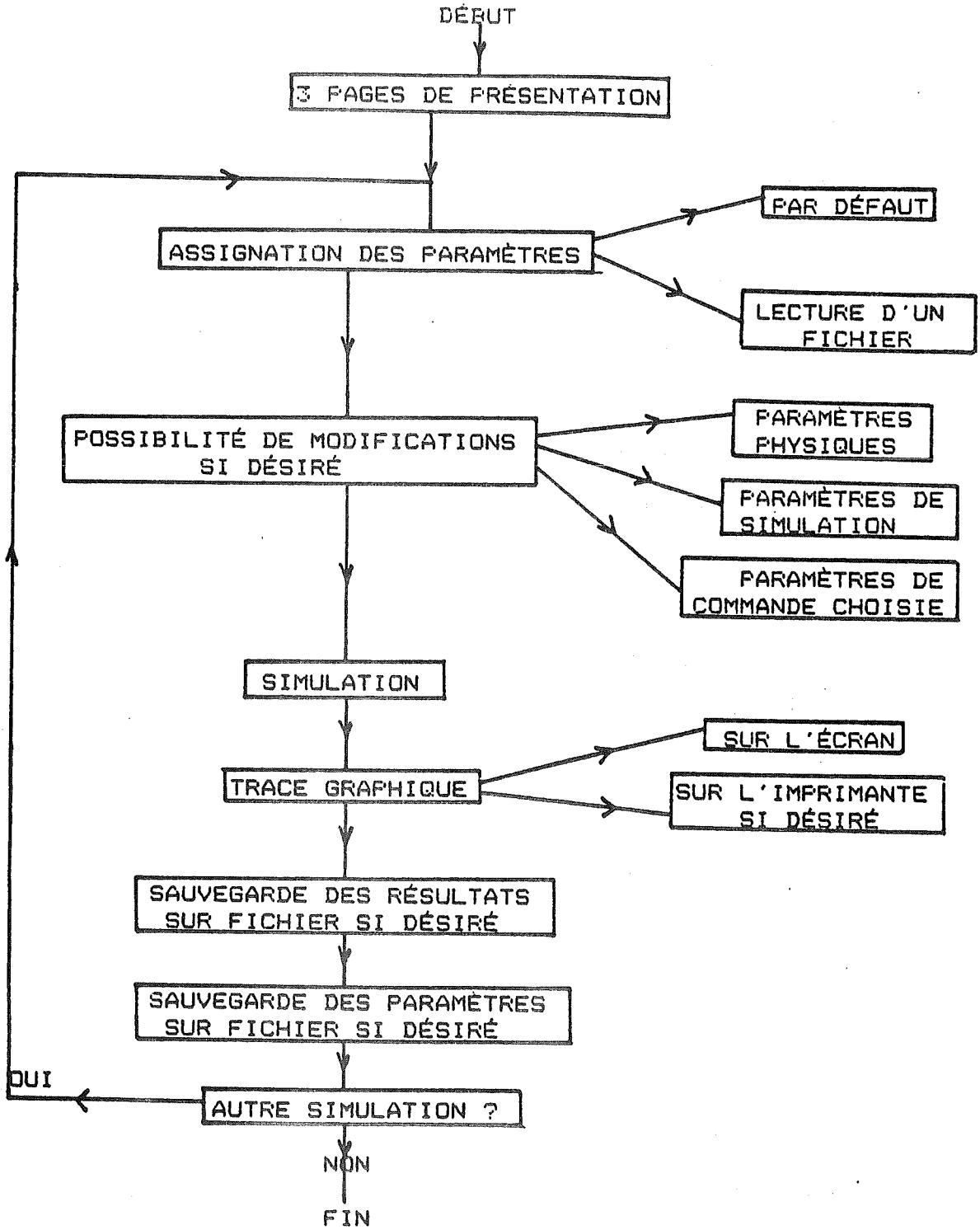


Figure 4.1: Déroulement chronologique du programme vu par l'utilisateur.

4.2.5 Sauvegarde des résultats sur fichier

L'intérêt de cette opération réside dans l'utilisation de certains résultats sous forme vectorielle pour tracer des courbes superposées avec le logiciel Matlab [16] (voir chapitre VI). Cette présentation rend plus faciles et plus clairs à la fois le travail d'analyse et le présent exposé.

4.2.6 Sauvegarde des paramètres sur fichier

Cette option s'avère être très utile dans les cas où l'utilisateur répète un certain nombre d'essais en faisant varier un seul paramètre (par exemple pour l'étude de sensibilité au chapitre VI). Ainsi, il charge les paramètres au début de l'exécution et l'étape de modification des paramètres est très brève.

4.3 Structure interne du logiciel

La structure interne du logiciel est très voisine de la structure fonctionnelle perçue par l'utilisateur. Ainsi la figure 4.2 qui mentionne les différents sous-programmes utilisés a une architecture très voisine de la figure 4.1.

```
***************PARAMETRES DU SYSTEME PHYSIQUE************************

(1)  GAIN STATIQUE DU MOTEUR.................................KM=1.50
(2)  CONSTANTE DE TEMPS DU MOTEUR.........................TAU1=0.050
(3)  MOMENT D'INERTIE MOTEUR.................................J1=0.50
(4)  RAYON DE COUPLAGE MOTEUR...............................R1=0.040
(5)  COEFFICIENT D'ELASTICITE RESSORT.......................KR=1250000.0000
(6)  CONSTANTE DE TEMPS ROBOT............................TAU2=10.000
(7)  MOMENT D'INERTIE ROBOT..................................J2=75.0000
(8)  RAPPORT DE TRANSMISSION................................NT=10
```

Tableau 4.1: Paramètres du système physique.

```
***************PARAMETRES DE SIMULATION******************************

(1)  AMPLI(LINEAIRE/DUALMODE/BANG-BANG):(0/1/2..............AMP=0
(2)  CONTROLE(SSV/PID/RETOUR D'ETAT):(0/1/2)..................R=0
(3)  PERIODE D'ECHANTILLONNAGE..............................DT=0.0010000
(4)  FIN DE SIMULATION......................................TF=1.00
(5)  PERTURBATION SUR L'ENTREE..............................CF=0.00
(6)  ETAT INITIAL...........................T10,T20,T30,T40=0.00 0.00 0.00 0.00
(7)  ETAT INITIAL DE L'INTEGRATEUR..........................I30=0.00
(8)  ETATS DESIRES..................................REF1,REF2=1.00 0.10
(9)  DELAI DE COMMUTATION...................................D=0.000000
(10) ON TRACE UN POINT CALCULE SUR NB.......................NB=10
(11) BRUIT CAPTEURS DE POSITION ET DE VITESSE.............N1,N2=0.000 0.000
(12) GAINS CAPTEURS DE POSITION ET DE VITESSE.............KP,KS=0.020 0.06
(13) NIVEAU DE QUANTISATION CONVERTISSEUR A/N.............Q1,Q2=0.00010 0.00010
(14) NIVEAU DE QUANTISATION CONVERTISSEUR N/A...............Q3=0.00010
(15) EMPLACEMENT DES CAPTEURS:X1,X2(1)/X3,X4(2)/X1,X4(3)......M=1
```

Tableau 4.2: Paramètres de simulation.

```
*****VOUS AVEZ CHOISI UN CONTROLEUR PID*****

    (1)  COMPOSANTE PROPORTIONNELLE.................K1=14.80
    (2)  COMPOSANTE DERIVEE.........................K2=1.30
    (3)  COMPOSANTE INTEGRALE.......................K3=34.90
    (4)  COMPOSANTE ANTICIPATRICE...................K4=-0.20
```

a)

```
*****VOUS AVEZ CHOISI UN CONTROLEUR A RETOUR LINEAIRE D'ETAT*****

    (1)....................KREG1=85.50
    (2)....................KREG2=3.30
    (3)....................KREG3=-32.10
    (4)....................KREG4=0.23
```

b)

```
*****VOUS AVEZ CHOISI UN CONTROLEUR A STRUCTURE VARIABLE*****

LES POLES DU SYSTEME EN MODE GLISSANT SONT PR1,PR2+i*Pi,PR3-i*Pi

    (1)....................PR1=-2.5E+001
    (2)....................PR2=-2.5E+001
    (3)....................PR3=-2.5E+001
    (4).....................PI=7.0E+001
```

c)

```
(1) alfa1<-3.36144<beta1    alfa1=-6.72000 beta1=0.00000

(2) alfa2<-1.83000<beta2    alfa2=-3.66000 beta2=0.00000

(3) alfa3<3.36144<beta3     alfa3= 0.00000 beta3=6.72000

(4) alfa4<0.77523<beta4     alfa4= 0.00000 beta4=1.54000
```

d)

Tableau 4.3: Paramètres des différents contrôleurs

A chaque opération définie à la section 4.2 correspond un sous-programme:

- **PRE**, **DIRECT** et **EQUATIONS** pour les 3 pages de présentation.
- **LECT** pour la lecture d'un fichier de paramètres.
- **DEFAUT** pour fixer la valeur des paramètres par défaut.
- **SYS** pour modifier les paramètres du système physique.
- **SIM** pour modifier les paramètres de simulation.
- **POLES**, **COMMUT**, **PID**, **KREG** pour modifier les paramètres de la commande choisie.
- **TRACE** pour les tracés graphiques (utilise **GRAPH1** et **GRAPH2**).
- **SAUVE** pour sauvegarder des tableaux de résultats sur fichier.
- **STOCK** pour sauvegarder les paramètres de la simulation sur fichier.

La boucle de calcul qui correspond à la partie SIMULATION vue par l'utilisateur (voir figures 4.1 et 4.2) est détaillée à la figure 4.3. A chaque calcul important correspond un sous-programme:

- **COMMANDE** pour le calcul de la commande (en fonction du contrôleur choisi).

- **AMPLI** pour l'amplification de la commande.
- **DELAI** pour tenir compte de l'effet d'un retard dans la commande.
- **RK4** pour effectuer l'intégration numérique Runge-Kutta d'ordre 4 (voir Annexe D).

Pour mieux comprendre les enchaînements de la figure 4.3, il peut être utile de consulter la figure 2.3 page 13 qui explique le fonctionnement du système asservi.

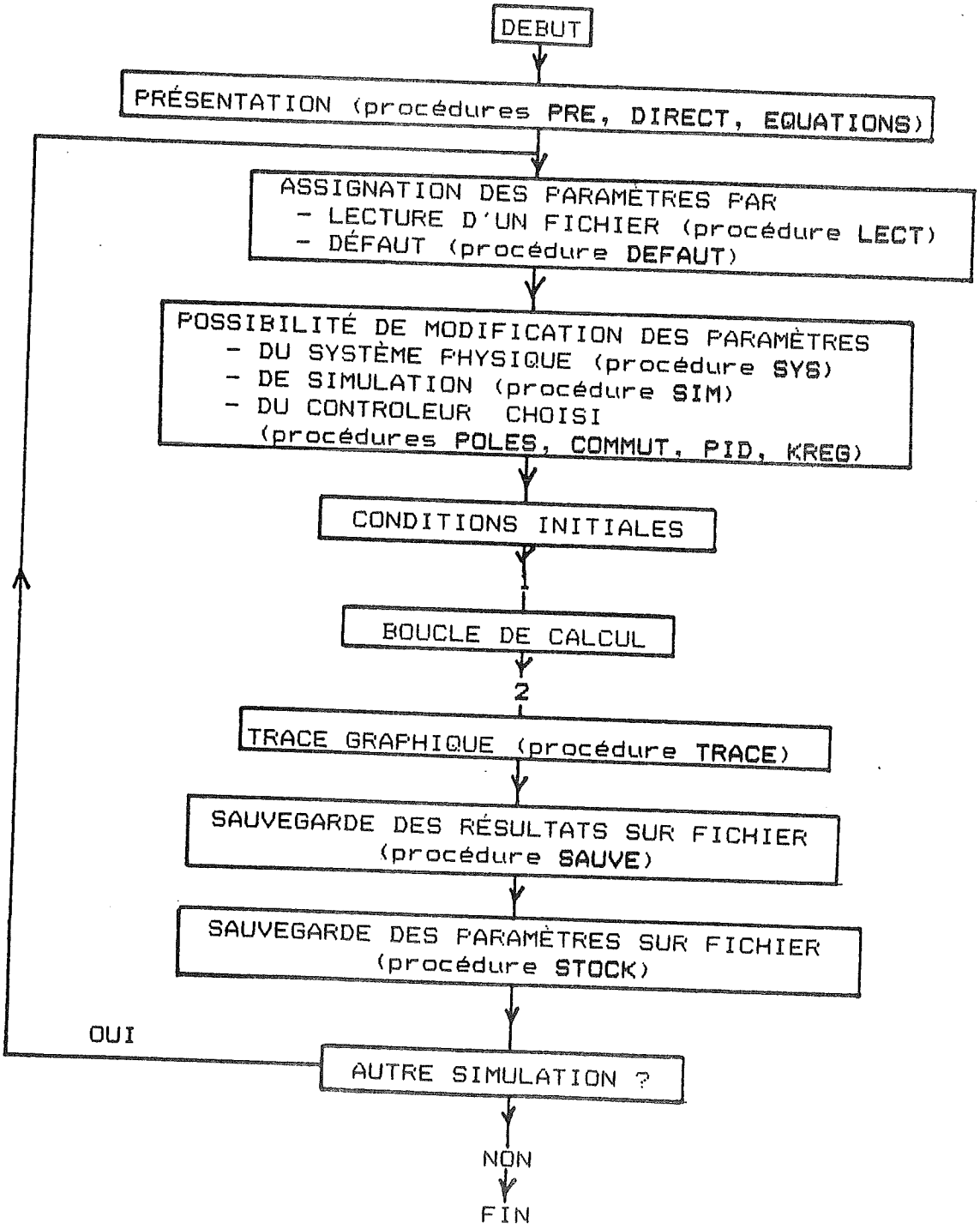


Figure 4.2: Structure interne du programme.

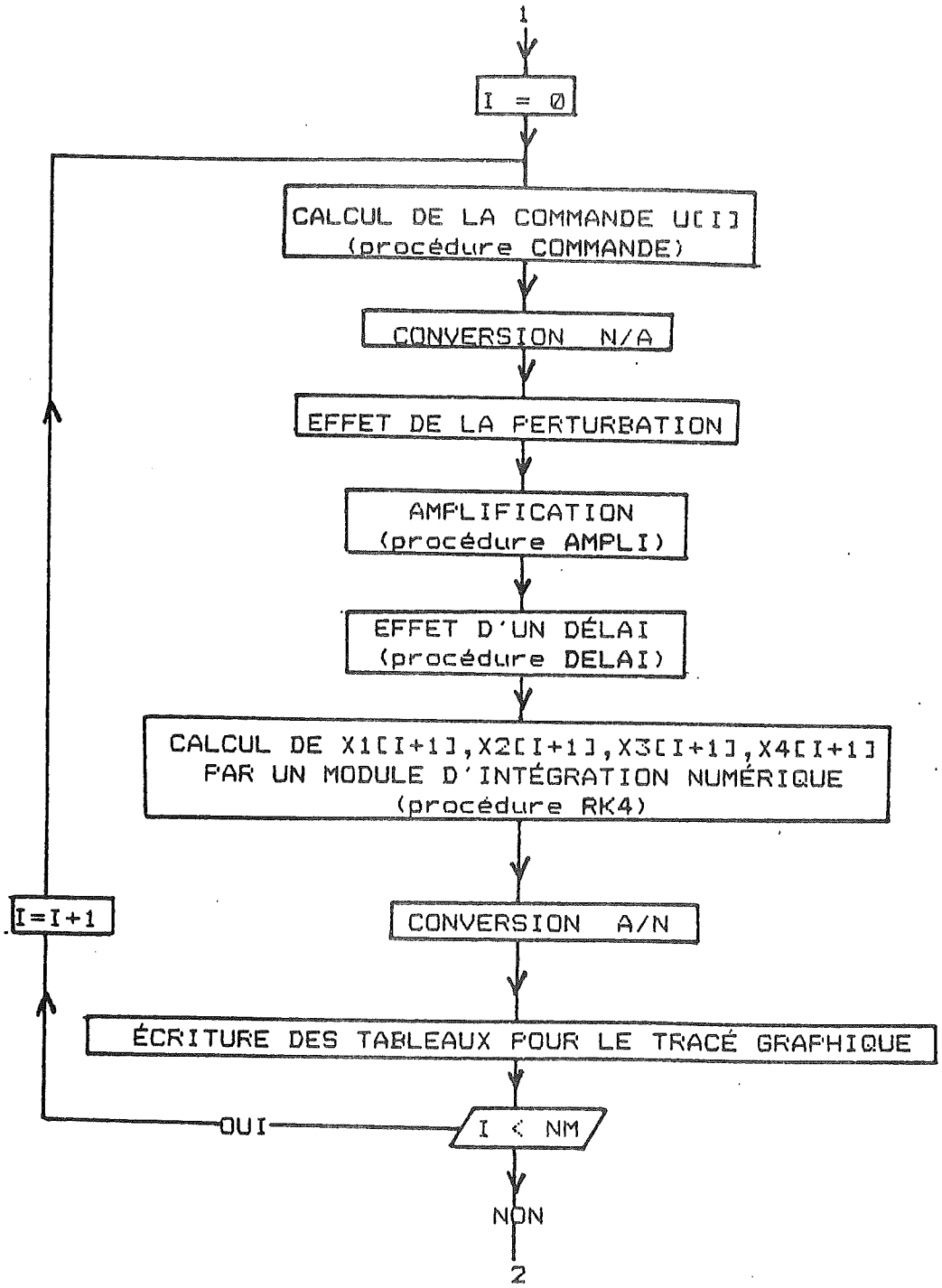


Figure 4.3: Structure interne de la boucle de calcul.

CHAPITRE V

COMPORTEMENT DU CONTROLEUR A STRUCTURE VARIABLE

5.1 Positionnement des pôles

Pour effectuer le positionnement des pôles qui doivent caractériser la dynamique du contrôleur à structure variable, nous nous inspirons des résultats obtenus avec les autres contrôleurs. Après divers essais avec le contrôleur PID (détaillés en annexe C), nous avons calculé les pôles en boucle fermée du système qui nous a paru optimal. Ces pôles sont: $-17\pm76*i$ -12 -5. Ce premier résultat nous a permis de placer facilement les pôles du contrôleur par retour linéaire d'état. Notre choix s'est porté sur des pôles égaux à: $-25\pm70*i$ -25 -25. Ces dernières valeurs ont constitué notre point de départ pour le positionnement des pôles du contrôleur à structure variable. Quelques essais significatifs ont été reportés au tableau 5.1.

pôles choisis	TR5%	extrémas de u
-25 -25 -25	0.25s	300/-350
-17 -17±76*i	0.45s	5/-6
-30 -30±60*i	0.13s	70/-70
-25 -25±70*i	0.20s	20/-30

Tableau 5.1: Choix des pôles du contrôleur à structure variable

Pour chaque essai les valeurs des gains α_i et β_i sont choisies symétriques par rapport aux l_i avec un écart de 100%. Notre choix s'est porté sur les pôles égaux à -25 -25±70*i parce qu'avec des valeurs moyennes des gains ils nous procurent un temps de réponse à 5% convenable, 200ms, et sollicitent le moteur de façon raisonnable.

Les résultats obtenus nous permettent d'affirmer que l'étape de positionnement des pôles dans la conception du contrôleur à structure variable ne présente pas de difficulté particulière.

La différence du contrôleur à structure variable par rapport au contrôleur par retour linéaire d'état réside dans le fait que nous pouvons agir non seulement sur les

pôles , mais aussi sur les gains du contrôleur pour ajuster la réponse désirée.

5.2. Choix des gains

Le tableau 5.2 nous montre les résultats obtenus par le système à structure variable dont les pôles du mode glissant ont été fixés à -25,-25±70*i. Les gains du contrôleur à structure variable sont choisis conformément aux considérations exposées au chapitre III, c'est-à-dire que α_i et β_i ont des valeurs symétriques par rapport aux l_i (donnés par (3.9)). Les facteurs qui interviennent dans notre discussion sont le temps de réponse à 5% et la tension fournie par le moteur. Nous observons que plus les gains du contrôleur sont éloignés des l_i, meilleur est le temps de réponse du système. En effet, dans ce cas, le point de fonctionnement atteint l'hyperplan de commutation $\sigma=0$ plus rapidement et ainsi le système entre plus vite en mode glissant et rejoint la dynamique préassignée par le choix des pôles. Par contre, plus les gains sont élevés, plus on sollicite le moteur, puisque la tension u à fournir est de plus en plus importante. Il s'agit de trouver un compromis entre ces deux aspects.

variation par rapport aux ℓ_1	α_1	β_1	α_2	β_2	α_3	β_3	α_4	β_4	TR5%	Extrêmas de u
10%	-3.70	-3.02	-2.01	-1.65	3.02	3.70	0.69	0.85	>2s	0.6/0
20%	-4.03	-2.69	-2.20	-1.46	2.69	4.03	0.62	0.92	>2s	2/0
30%	-4.37	-2.35	-2.38	-1.28	2.35	4.37	0.54	1.00	1.2s	2/0
40%	-4.70	-2.02	-2.56	-1.10	2.02	4.70	0.46	1.08	0.5s	3/-1
50%	-5.04	-1.68	-2.74	-0.91	1.68	5.04	0.38	1.15	0.35s	4/-9
60%	-5.38	-1.34	-2.93	-0.73	1.34	5.38	0.31	1.23	0.27s	5/-10
70%	-5.71	-1.01	-3.11	-0.55	1.01	5.71	0.23	1.31	0.22s	10/-15
80%	-6.05	-0.67	-3.29	-0.37	0.67	6.05	0.15	1.39	0.20s	15/-20
90%	-6.38	-0.34	-3.48	-0.18	0.34	6.38	0.08	1.46	0.19s	15/-25
100%	-6.72	0	-3.66	0	0	6.72	0	1.54	0.15s	20/-30
150%	-8.4	1.68	-4.57	0.91	-1.68	8.4	-0.38	1.92	0.14s	35/-45
200%	-10.08	3.36	-5.49	1.83	-3.36	10.08	-0.77	2.31	0.13s	55/-80

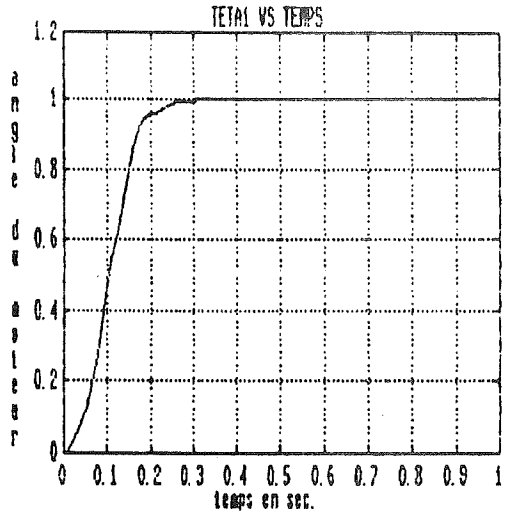
Tableau 5.2: Choix des gains du contrôleur à structure variable avec les valeurs nominales des pôles -25, -25±70*i.

5.3. Résultats

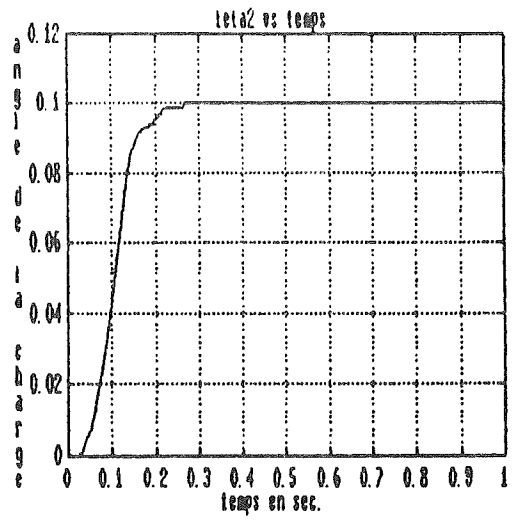
L'essai que nous retenons pour la suite de l'analyse correspond aux pôles en mode glissant déjà cités -25,-25±70*i. Les valeurs nominales des gains choisis α_i et β_i varient de 100% par rapport aux l_i. Ces valeurs sont encadrées sur le tableau 5.2 et vérifient les inégalités (3.7) et (3.8). Les réponses obtenues avec ce système sont reportées à la figure 5.1.

p1	= -25
p2	= -25+70*i
p3	= -25-70*i
α1	= -6.72
β1	= 0
α2	= -3.66
β2	= 0
α3	= 0
β3	= 6.72
α4	= 0
β4	= 1.54

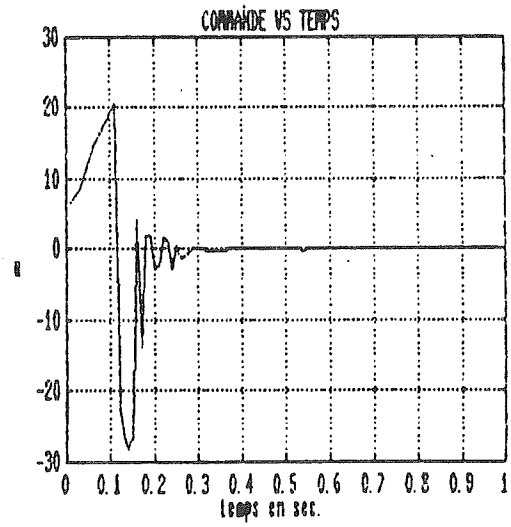
Tableau 5.3: Paramètres de l'essai nominal



(1)



(2)



(3)

Figure 5.1: Essai SSV: valeurs nominales des gains

(1) teta1 versus temps

(2) teta2 versus temps

(3) commande

Tous les essais concernant le contrôleur à structure variable relatés dans le chapitre suivant proviennent, sauf précision contraire, de cet essai nominal.

Dans tous les essais qui ont été effectués pour ce travail, nous avons adopté une période d'échantillonnage de 1 ms; nous avions vérifié auparavant que les résultats sont identiques si l'on échantillonne avec une période égale à 0.1 ms. Dans les cas où le système diverge, pour nous assurer que la divergence n'est pas une divergence de calculs numériques, nous avons calculé les pôles du système en boucle fermée. Nous avons ainsi pu nous assurer que certains de ces pôles se situent à droite l'origine, ce qui nous conforte dans l'idée qu'il s'agit bien d'une divergence du système. Cette vérification a été réalisée à l'aide du logiciel MATLAB [20] et les détails des calculs se trouvent en annexe C.

CHAPITRE VI

ANALYSE COMPARATIVE DES TROIS CONTROLEURS

6.1 Introduction

Ce chapitre analyse les avantages et les inconvénients de l'utilisation d'un contrôleur à structure variable dans l'asservissement de position d'une articulation robotique avec un lien de transmission élastique. Dans ce but, nous décrivons un certain nombre de simulations effectuées avec le logiciel ROBEL pour évaluer les performances du contrôleur à structure variable en matière de précision, de temps de réponse, de sensibilité à des variations de paramètres, à une perturbation constante et à un délai. Ces performances sont comparées à celles obtenues en utilisant le contrôleur PID et le régulateur par retour linéaire d'état.

De façon générale pour les trois contrôleurs, nous observons que l'angle du moteur $\theta 1$ et l'angle de l'articulation robotique $\theta 2$ ont le même comportement. Ainsi nous nous contenterons d'illustrer cet exposé des courbes $\theta 2$ versus temps. Pour tous les essais, l'état initial de $\theta 2$ a été choisi égal à 0 et la référence ref2

égale à 0.1 rad.

6.2 Systèmes de référence

Les paramètres des contrôleurs PID et par retour linéaire d'état qui constituent notre référence figurent au tableau 6.1.

contrôleur	gains
PID	K1 = 14.80 K2 = 1.30 K3 = 34.90 K4 = -0.20
RETOUR D'ETAT	KREG1 = 65.80 KREG2 = 2.70 KREG3 = -22.61 KREG4 = 0.51

Tableau 6.1: Paramètres des systèmes de référence

Dans la suite les illustrations concernant ces deux contrôleurs proviennent de ces valeurs des paramètres, à l'exception de l'essai avec le contrôleur PID qui figure à la section 6.3.

Pour le contrôleur à structure variable, les paramètres de référence sont donnés au tableau 5.3 et

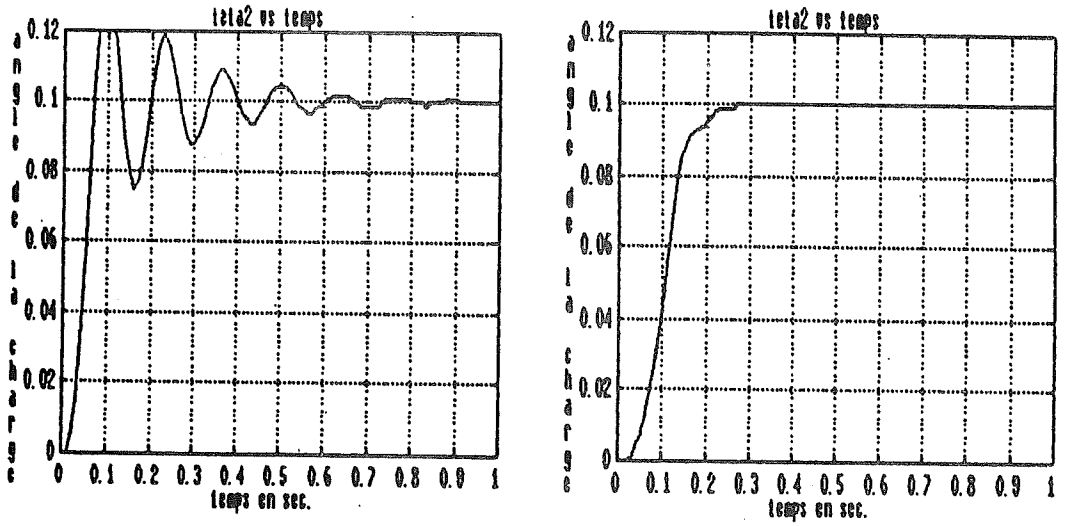
constituent l'essai nominal.

6.3. Performances statiques et dynamiques

La précision des réponses obtenues, que ce soit avec le contrôleur PID, par retour linéaire d'état ou à structure variable, ne prête pas à grande discussion. En effet les réponses du système nominal présentent toujours une erreur à régime quasi nulle.

Il n'en est pas de même du temps de réponse du système. En effet, il ressort des résultats exposés en annexe C que l'on ne peut pas obtenir un temps de réponse aussi petit que l'on veut avec le contrôleur PID. Par contre, comme nous le montrent les résultats du tableau 5.2, avec un contrôleur à structure variable il est possible d'obtenir, en théorie, un temps de réponse aussi court que l'on veut, en augmentant les gains du contrôleur. Par contre, en pratique, on est limité par la tension maximale que peut fournir le moteur (±125V).

Ainsi la figure 6.1(2) nous montre la réponse du système à structure variable avec un temps de réponse de 200 ms environ pour la charge (essai nominal). Si l'on calcule les gains du contrôleur PID pour obtenir ce temps de réponse, on obtient la réponse donnée à la figure 6.1(1).

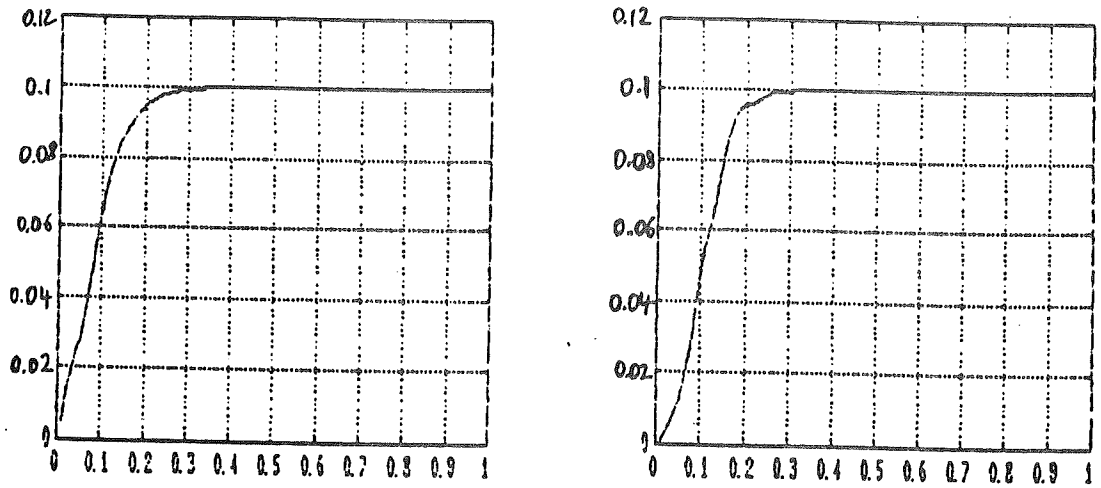


(1) (2)

Figure 6.1: Performance dynamique PID (1) versus structure variable (2)

Cet essai montre bien la supériorité du contrôleur à structure variable face au contrôleur PID en matière de temps de réponse.

Par contre, un système linéaire avec retour d'état peut en général fournir le même temps de réponse qu'un système à structure variable. Cet aspect est illustré par les courbes de la figure 6.2 ci-dessous.



(1) (2)

Figure 6.2: Performance dynamique retour linéaire (1) versus structure variable (2)

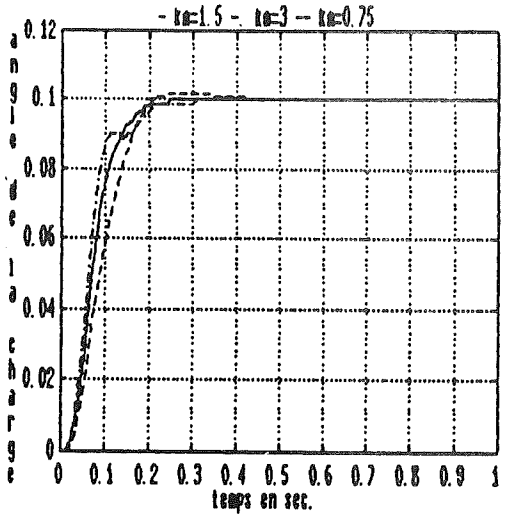
6.4. Sensibilité à la variation des paramètres

Les paramètres que nous avons fait varier dans cette étude sont Km, le gain statique du moteur, tau1, la constante de temps du moteur et j2 le moment d'inertie de la charge. Pour les deux premiers paramètres, il est intéressant d'envisager de telles variations, car on ne peut pas les calculer de façon exacte. Par ailleurs j2, le moment d'inertie du robot par rapport à l'axe de rotation considéré, varie au cours du temps en fonction des mouvements du robot. En termes de variations, nous avons

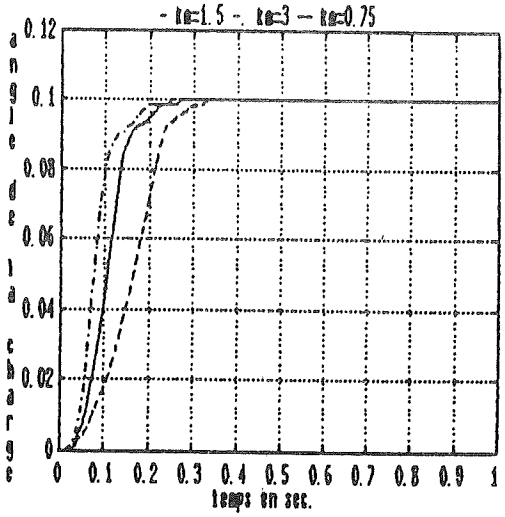
envisagé des valeurs deux fois plus grandes et deux fois plus petites que les valeurs nominales. Soit

Km = 1.5	Km = 3	Km = 0.75
tau1 = 0.05	tau1 = 0.1	tau1 = 0.025
j2 = 75	j2 = 150	j2 = 37.5

Les courbes des figures 6.3, 6.4 et 6.5 nous montrent que la réponse du système à structure variable nominal est moins sensible que celle du système à retour linéaire d'état face à la variation de Km, tau1 et j2.

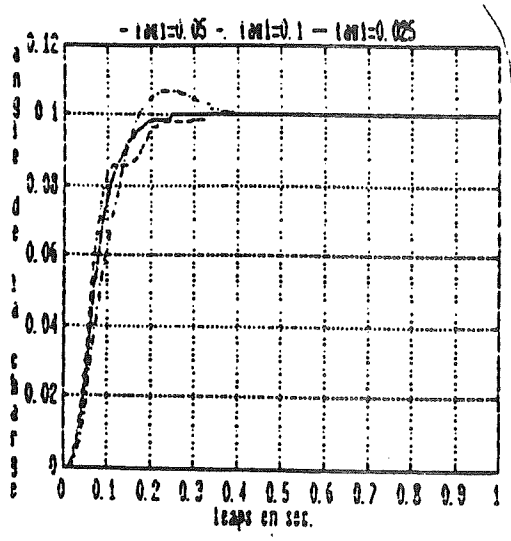


(1)

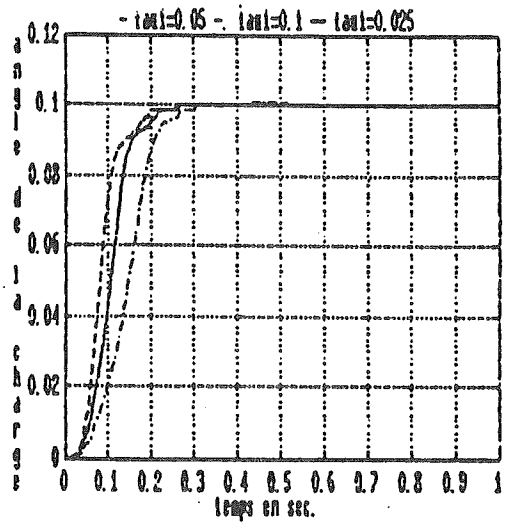


(2)

Figure 6.3: Variation de Km: retour linéaire d'état (1) versus structure variable (2)

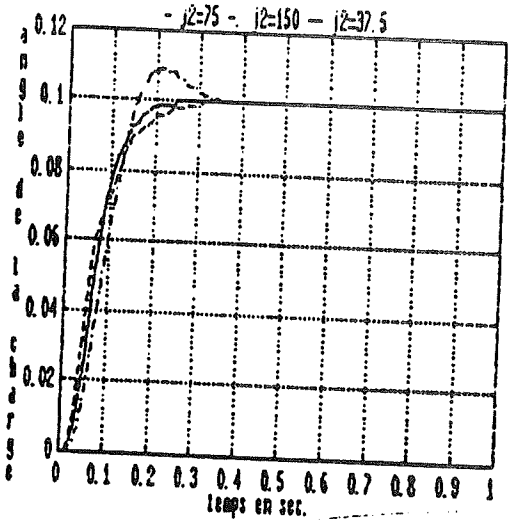


(1)

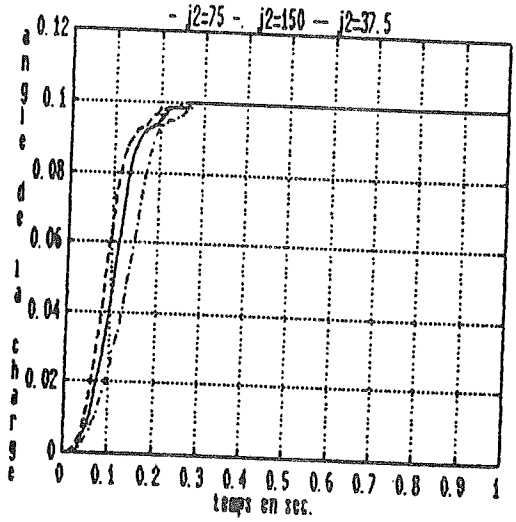


(2)

Figure 6.4: Variation de tau1: retour linéaire d'état (1) versus structure variable (2)



(1)



(2)

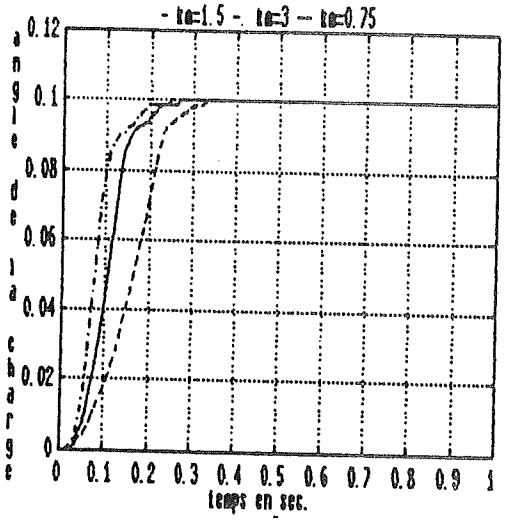
Figure 6.5: Variation de j2: retour linéaire d'état (1) versus structure variable (2)

Nous remarquons que les réponses du système linéaire à retour d'état présentent des dépassements de l'ordre de 5% lorsque les paramètres du système varient. Par contre, ces dépassements n'existent pas lorsque nous faisons varier l'un quelconque des paramètres cités plus haut dans le système à structure variable. Les performances des deux systèmes s'avèrent être identiques d'un point de vue rapidité de rétablissement du régime permanent.

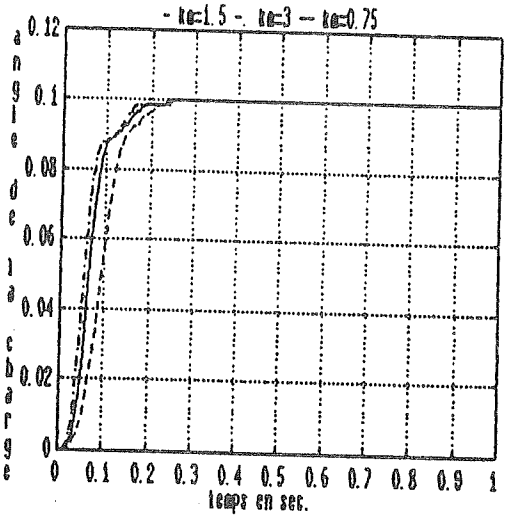
Par contre, lors de la variation d'un paramètre, les différentes réponses du système à structure variable

nominal s'avèrent être décalées dans le temps. Ce résultat provient du fait que le vecteur de commutation C et les valeurs des limites l_i dépendent directement de Km, tau1 et j2 (voir chapitre III équations (3.10) à (3.17)). Ainsi, lorsqu'un de ces paramètres varie, il se trouve que les pôles du système n'ont plus la valeur préassignée et la dynamique résultante est différente. Par ailleurs, comme les l_i ont changé, il s'avère dans notre exemple que l'on n'est plus en mode glissant, même si on en est encore très proche.

Mais la grande force des systèmes à structure variable vient du fait que plus les gains du contrôleur sont élevés, meilleures sont les performances du système face aux variations des paramètres. Cet aspect essentiel est illustré par les schémas (1) et (2) des figures 6.6, 6.7 et 6.8.



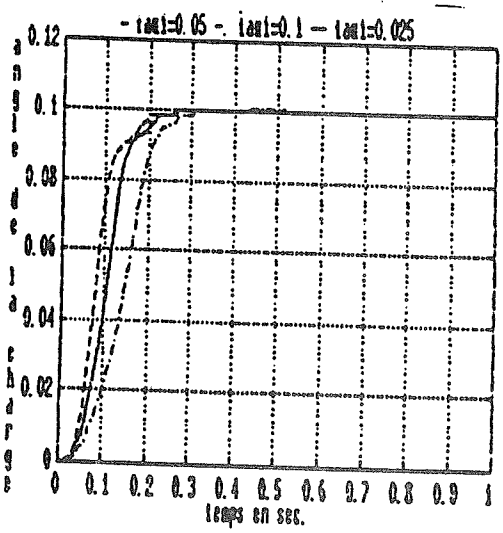
(1)



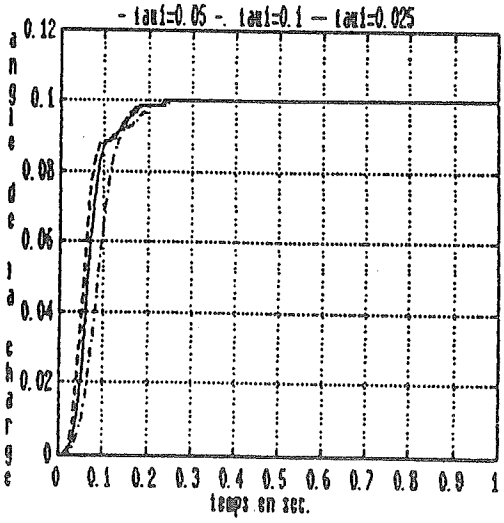
(2)

Figure 6.6: Variation de Km: essai nominal (1)

versus gains supérieurs (2)



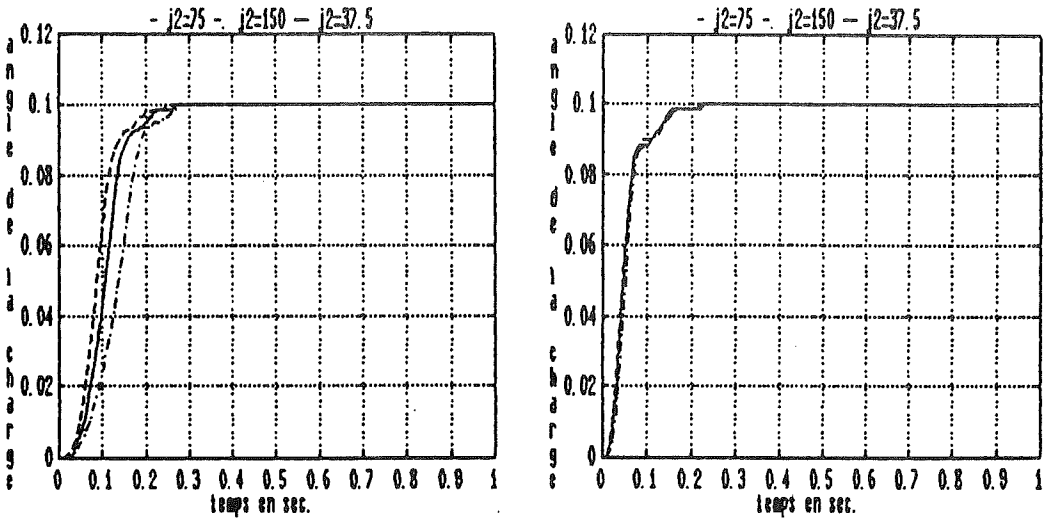
(1)



(2)

Figure 6.7: Variation de tau1: essai nominal (1)

versus gains supérieurs (2)



(1) (2)

Figure 6.8: Variation de j2: essai nominal (1)
versus gains supérieurs (2)

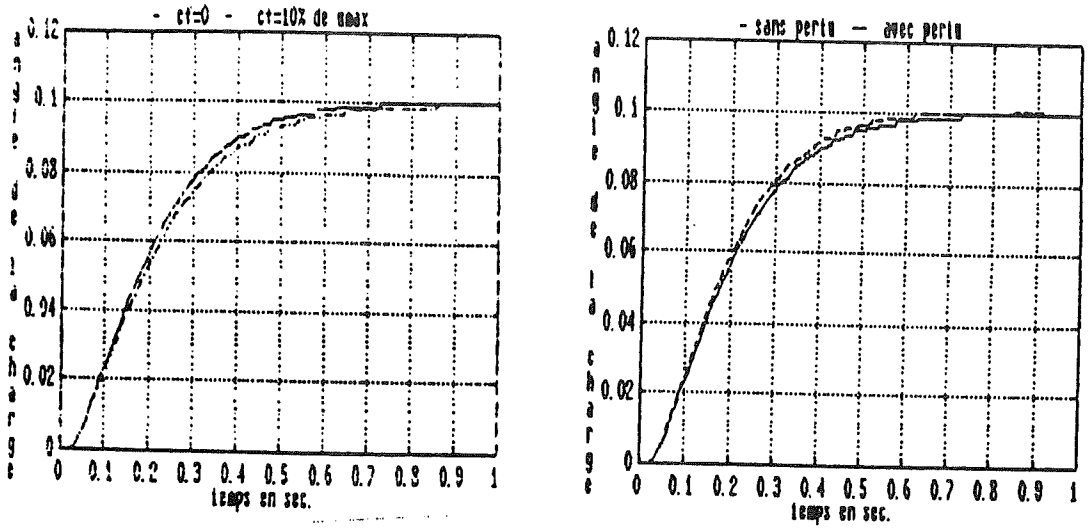
Les schémas (1) des trois figures précédentes constituent les réponses du système nominal avec les gains nominaux respectivement lorsque Km, tau1 et j2 varient; les schémas (2) résultent des variations respectives de Km, tau1 et j2 et constituent les réponses d'un système où les valeurs des gains ont été choisies dans le tableau 5.2, de telle sorte que l'on soit assuré qu'avec n'importe quelle combinaison de paramètres étudiée, on est en mode glissant. Ainsi, pour les variations de Km et tau1, les schémas (2) des figures 6.6 et 6.7 résultent de l'essai constituant la

dernière ligne du tableau 5.2. Pour illustrer la variation de j2 (schéma (2) de la figure 6.8), nous avons choisi des gains encore plus élevés et l'essai reste du domaine de la simulation puisque la commande dépasse les limites prescrites (|umax |=125V).

Nous observons que sur les schémas (2) des trois figures, les réponses du système sont beaucoup plus proches que sur les schémas (1). En fait, la dynamique du système n'est toujours pas celle préassignée par le choix des pôles. Simplement, lorsque les gains sont plus élevés, le système entre en mode glissant et dès lors s'avère insensible à la variation des paramètres.

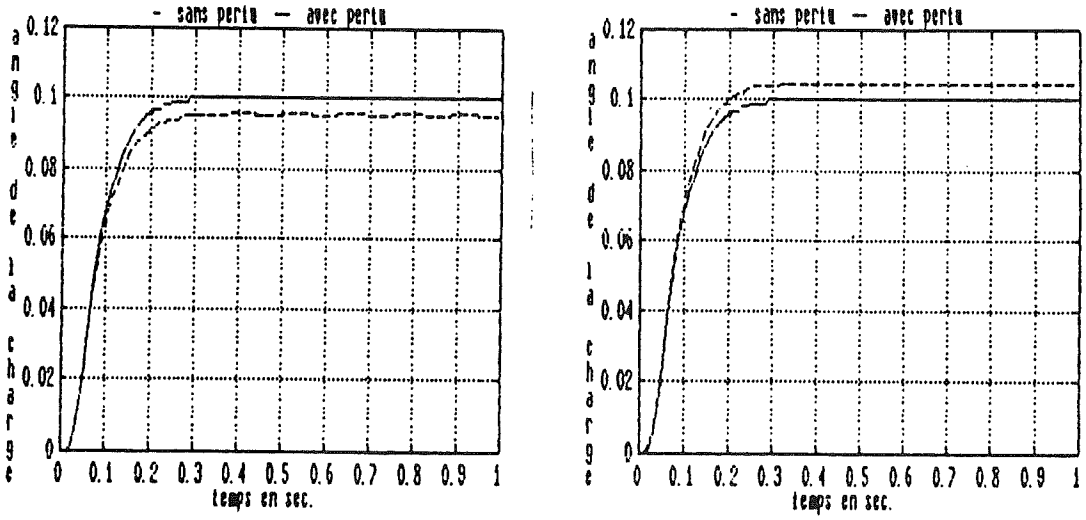
6.5 Analyse de la sensibilité à une perturbation

Nous nous interessons ici à une perturbation constante cf, qui peut être représentée par une charge constante agissant sur l'axe du moteur. Pour chacun des essais effectués, nous appliquons une perturbation égale environ à 10% de la commande maximale fournie lorsqu'il n'y a pas de perturbation. Les courbes des figures 6.9, 6.10 et 6.11 nous montrent que le contrôleur à structure variable est moins robuste face aux perturbations que le contrôleur par retour linéaire d'état, qui l'est lui-même moins que le contrôleur PID.



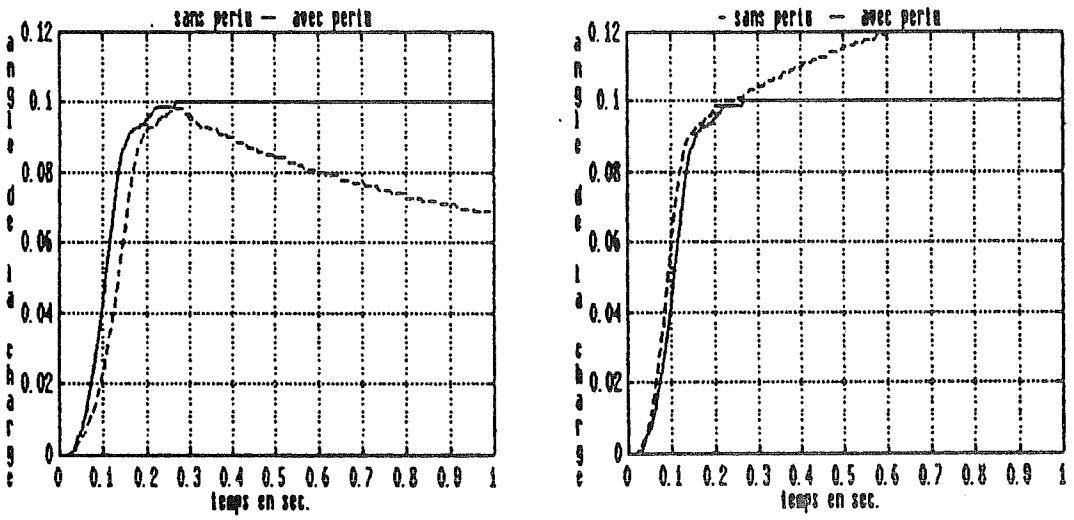
(1) (2)

Figure 6.9: Sensibilité à une perturbation >0 (1) <0 (2) pour le contrôleur PID



(1) (2)

Figure 6.10: Sensibilité à une perturbation >0 (1) <0 (2) pour le contrôleur par retour linéaire d'état



(1) (2)

Figure 6.11: Sensibilité à une perturbation >0 (1) <0 (2) pour le contrôleur à structure variable

En effet, la figure 6.9 nous montre que le système contrôlé par un PID est pratiquement insensible à l'effet d'une perturbation. Ce résultat est la conséquence bien connue de la présence de l'intégrateur. Le contrôleur par retour linéaire d'état est sensible à la perturbation puisqu'il présente une erreur à régime d'environ 5% comme l'indique la figure 6.10. Par contre sur la figure 6.11, on observe une erreur à régime de ±45% dans la réponse du système à structure variable nominal, lorsqu'on applique

respectivement une perturbation positive (1) ou négative (2). Un rapide calcul détaillé en annexe E permet de vérifier ces résultats et de prévoir qu'en augmentant les gains du contrôleur, nous pouvons,en théorie, rendre l'erreur aussi petite que nous le désirons. Mais nous sommes limités, bien-sûr, par les valeurs maximales de la commande ($|u_{max}|=125V$). Ainsi, nous avons obtenu de bons résultats en augmentant $|\alpha 1|$ et $|\alpha 3|$ et en gardant tous les autres gains identiques. La figure 6.12 compare les réponses obtenues avec les valeurs nominales des gains (a), avec les mêmes gains et une perturbation cf=3V (b) et enfin avec des gains $\alpha 1$ et $\alpha 3$ plus élevés ($\alpha 1$ et $\alpha 3 = -150$) et la même perturbation (c). Ainsi avec $\alpha 1 = \alpha 3 = -150$, l'erreur à régime n'est plus que de 1%.

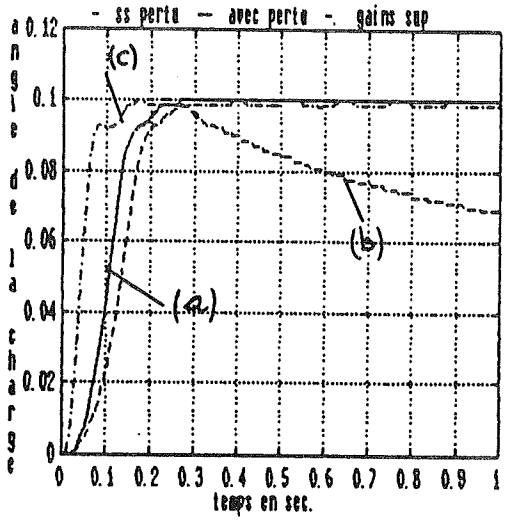


Figure 6.12: Amélioration du contrôleur à structure variable face à une perturbation.

(a) essai nominal

(b) gains nominaux et perturbation

(c) gains supérieurs et perturbation

6.6 Analyse de la sensibilité au délai

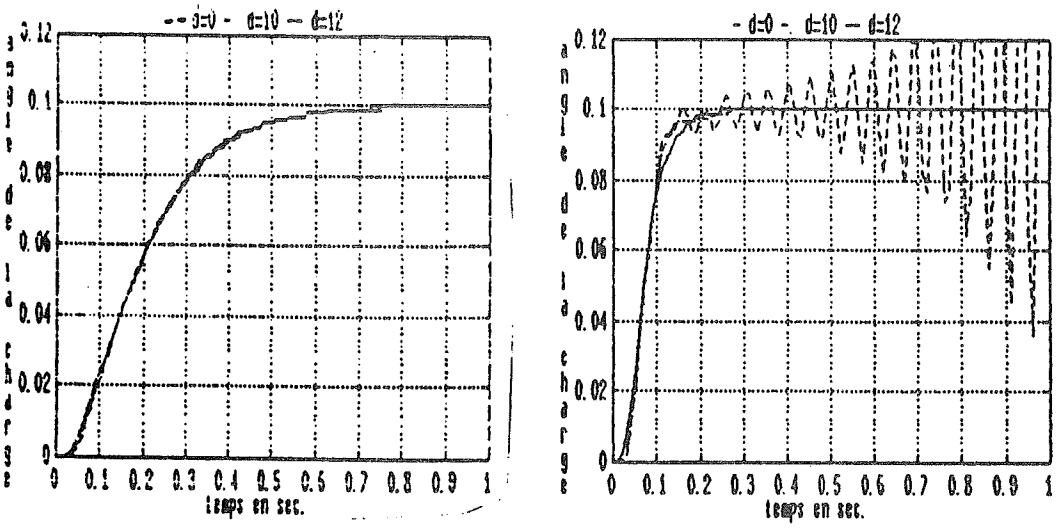
Le programme ROBEL (voir description au chapitre IV), par l'intermédiaire du sous-programme délai, nous offre la possibilité de faire intervenir un délai dans l'application de la commande. Toutes les courbes que nous présentons par la suite figurent la réponse du système avec un délai de 0,10 ou 20 ms.

Comme l'indiquent les schémas (1), (2) et (3) de la figure 6.13, le système à structure variable s'avère très sensible à la présence d'un délai.

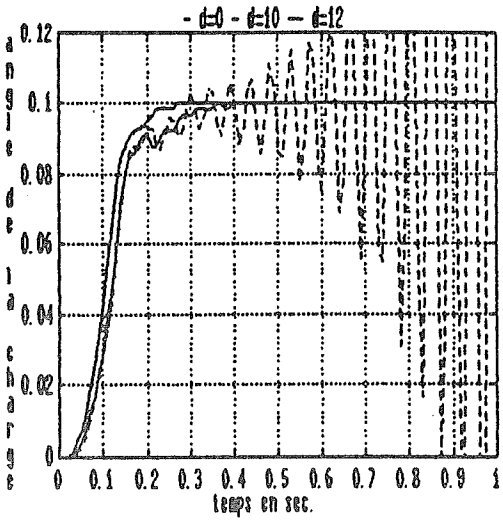
D'après le schéma (1) nous observons que le système avec le contrôleur PID est pratiquement insensible à un délai de l'ordre de 12 ms. De façon intuitive, ce résultat peut s'expliquer par le fait que le système réagit de façon lente et a donc le temps de se réajuster lorsqu'intervient un délai.

Par contre, le schéma (2) nous montre que le système avec retour d'état oscille puis converge avec un délai de 10 ms, alors qu'il diverge avec un délai de 12 ms. La même interprétation intuitive prévaut ici, c'est-à-dire que comme le système est rapide, il va se déstabiliser s'il intervient un délai dans la commande et même diverger si ce délai est trop important.

Le système à structure variable s'avère très sensible au délai (schéma (3)). Il est intrinsèquement encore plus rapide que le système contrôlé par retour linéaire d'état et donc, intuitivement,plus sensible au délai que ce dernier.



(1) (2)



(3)

Figure 6.13: Effet d'un délai sur les contrôleurs PID (1) à retour linéaire d'état (2) et à structure variable (3)

A la lumière des observations faites à la section 6.5, nous pouvons nous demander à présent si une amélioration des performances du système à structure variable face à une perturbation par l'augmentation des gains ne va pas entraîner une dégradation d'un point de vue comportement face à un délai. Ce point est illustré par la figure 6.14.

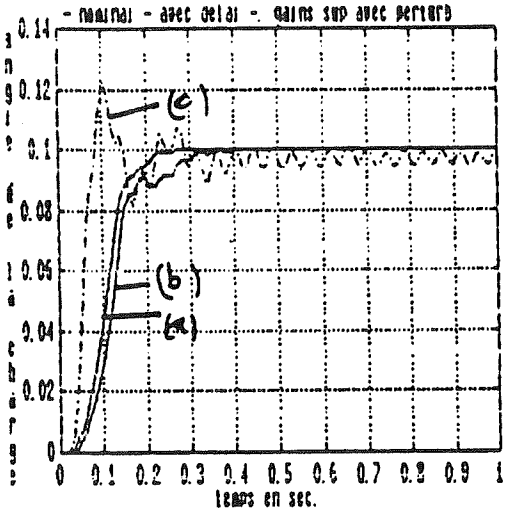


Figure 6.14: Comportement du système à structure variable avec gains plus élevés face à un délai

(a) essai nominal

(b) gains nominaux et délai de 10 ms

(c) gains suprieurs et délai de 10 ms

Les courbes présentées comparent les réponses obtenues avec les valeurs nominales des gains et sans délai (a), avec les mêmes gains et un délai de 10 ms (b), enfin avec des gains α_1 et α_3 (tels que l'erreur à régime soit de 3% en présence d'une perturbation cf=3) et un délai de 10 ms (c). On s'aperçoit qu'avec les gains tels que l'erreur à régime soit de 3% le système converge après un temps très long et d'importantes oscillations. Par contre, si on augmente encore plus les gains pour avoir une erreur à régime de 1%, le système diverge avec un délai de 10 ms. Cet aspect est illustré par la figure 6.15.

La courbe présentée compare les réponses obtenues avec des gains $\alpha_1 = \alpha_3 = -50$ (tels que l'erreur à régime soit de 3% en présence d'une perturbation cf=3) et un délai de 10 ms (a) et avec des gains $\alpha_1 = \alpha_3 = -150$ (tels que l'erreur à régime soit de 1%) et un délai de 10 ms (b).

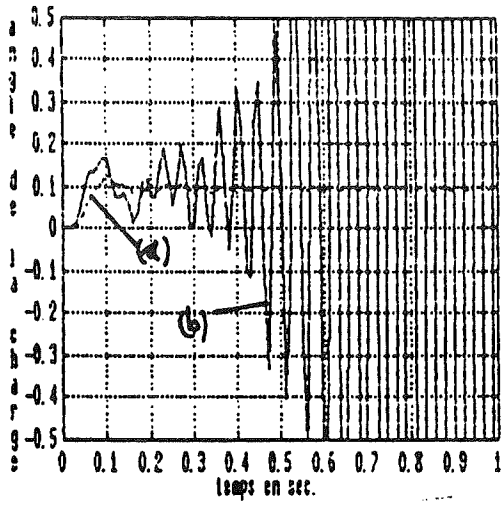


Figure 6.15: Effet d'un délai sur un contrôleur à structure variable peu sensible aux perturbations.

(a) gains moyens (supérieurs aux gains nominaux) et délai de 10 ms

(b) gains supérieurs à (a) et délai de 10 ms

CHAPITRE VII

CONCLUSION

7.1 Simplicité de conception et fiabilité

Sur la base des expériences présentées dans ce mémoire, il nous est permis de constater que malgré leur nouveauté, les systèmes à structure variable bénéficient d'une méthodologie de conception bien définie et efficace. Le choix des paramètres du contrôleur ainsi que les calculs nécessaires se font de façon aisée. Les résultats obtenus correspondent aux prévisions que l'on peut tirer de façon analytique et prouvent la fiabilité du contrôleur à structure variable. Par ailleurs, son implantation sur microordinateur ne présente pas de difficulté particulière.

7.2 Résultats obtenus

L'analyse comparative des performances des contrôleurs PID, par retour linéaire d'état et à structure variable nous permet d'énoncer les conclusions suivantes:

En choisissant des gains suffisamment élevés, le

contrôleur à structure variable s'avère être pratiquement insensible à une perturbation sur la commande et à des variations de paramètres du système. Par contre, il devient alors très sensible à un délai sur la commande et demande au moteur de fournir une tension très élevée et plus oscillante, donc de dépenser plus d'énergie.

De façon plus détaillée, l'analyse comparative des trois contrôleurs nous a permis de constater que:

- Sous des conditions nominales, les performances statiques et dynamiques des contrôleurs à structure variable et par retour linéaire d'état sont équivalentes. Par contre, le contrôleur PID nous donne un système beaucoup plus lent.

- La présence d'une perturbation extérieure a des effets différents sur les trois contrôleurs. Cet effet est pratiquement insignifiant lors de l'application d'un contrôleur PID; il reste très faible lors de l'utilisation d'un régulateur par retour linéaire d'état; par contre il devient très important avec le contrôleur à structure variable. Cette situation peut être améliorée en augmentant les gains.

- Face à un délai de 12 ms sur la commande, le contrôleur PID s'avère être parfaitement insensible, alors que les contrôleurs par retour linéaire d'état et à structure variable divergent tous les deux. Par ailleurs, plus on augmente ses gains, plus le contrôleur à structure variable s'avère être sensible au délai.

- Face à des variations de paramètres du système, les contrôleurs PID et par retour linéaire d'état présentent tous deux des dépassements dans les réponses du système, mais le régime permanent se rétablit beaucoup plus lentement pour le premier. Le contrôleur à structure variable avec les valeurs nominales des gains est supérieur en ce sens que les réponses ne présentent pas de dépassement; par contre, ces réponses s'avèrent être relativement décalées dans le temps lorsqu'on fait varier un paramètre. Cet inconvénient disparaît lorsqu'on augmente les gains.

7.3 Extensions possibles

Pour une première approche globale du contrôle à structure variable, la présente étude a atteint les objectifs fixés. Il serait néanmoins intéressant

d'investiguer les approches suivantes:

- rajouter un intégrateur au contrôleur à structure variable, ce qui pourrait lui conférer une plus grande robustesse face aux perturbations;

- rajouter un observateur d'état, pour prévoir les cas où l'état du système n'est pas accessible par une mesure directe;

- concrétiser cette étude par des tests sur un système physique;

- faire une étude plus poussée de la sensibilité du système aux variations des paramètres.

BIBLIOGRAPHIE

1. V.I. UTKIN, "Sliding Modes and their Application in Variable Structure Systems", Mir Publisher Moscow (1978).

2. U. ITKIS, "Control Systems of Variable Structure", John Wiley & Sons Inc., New York (1976).

3. A.Y. SIVARAMA KRISHNAN, M.V. HARIHARAN, M.C. SRISAILAM, "Design of Variable Structure Load-Frequency Controller Using Pole Assignment Technique", Int. J. Control, Vol. 40, No. 3, 487-498 (1984).

4. V.I. UTKIN, "Variable Structure Systems: Present and Future", Plenum Publishing Corporation (1984).

5. V.I. UTKIN, K.D.YANG, "Methods for constructing discontinuity planes in multidimensional variable structure systems", Automatic Remote Control 39, 1466 (1978)

6. J-J.E. SLOTINE, J.A. COETSEE, "Adaptive sliding controller synthesis for non-linear systems", Int.J. Control, Vol.43 No6, 1631-1651 (1986)

7. J-J.E SLOTINE, "Sliding controller design for non-linear systems", Int.J.Control, Vol.40, No2,421-434 (1984)

8. K.K.D YOUNG, "Controller design for a manipulator using theory of variable structure systems", IEEE Transactions on Systems, Men and Cybernetics SMC-8, pp 101-109 (1978)

9. J.ERSCHLER, F.ROUBELLAT, J.P.VERNHES, "Automation of a Hydroelectric Power Station Using Variable-Structure Control Systems", Automatica, Vol.10, pp31-36(1974)

10. R.M. DESANTIS, "Théorie des systèmes de commande linéaires", École Polytechnique de Montréal (1986).

11. R.M. DESANTIS, "Éléments fondamentaux de la robotique: Cinématique, Dynamique et Commande", École Polytechnique de Montréal (1986).

12. A. TERROUX, R.M. DESANTIS, "Asserv. pos: un programme pour la simulation de la commande d'un joint robotique" (1984).

13. P. RADZISZEWSKI, V. TREMBLAY, R.M. DESANTIS, "Simulation du comportement dynamique de la boucle de commande de l'articulation verticale de Rhino XR1".

14. M. KUMAR, "A position control system with disturbance rejection using sliding mode", Mc Gill University (1986).

15. A.T. NGUYEN, R.M. DESANTIS, "Etude de sensibilité du comportement dynamique d'un manipulateur robotique", Rapport technique EPM/RT-87-2, Ecole Polytechnique de Montréal (1987)

16. ANONYMOUS, "Turbo Pascal Reference Manual", Borland International Inc. (1985).

17. ANONYMOUS, "Turbo Graphix Toolbox", Borland International Inc. (1985).

18. SHAN S. KUO, "Computer Applications for Numerical Methods"

19. COMAU S.p.A, "Smart", Assembly and Handling Systems Str. Borgaretto 22, 10040 Borgaretto, Torino (Italy)

20. C. MOLER, J. LITTLE, S. BANGERT, S. KLEIMAN, "PC-Matlab for MS-DOS Personal Computer", The Mathworks Inc. (1986).

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
3 9334 00294083 9