POLYPUBLIE
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Commande adaptative d'une articulation robotique avec la méthode du modèle de référence
Title:

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

Date: 1988

Type: Rapport / Report

Référence: Quintal, M., & De Santis, R. M. (1988). Commande adaptative d'une articulation robotique avec la méthode du modèle de référence. (Rapport technique n° EPM-RT-88-15). https://publications.polymtl.ca/9580/
Citation:

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: https://publications.polymtl.ca/9580/
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: EPM-RT-88-15
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

2 9 AVR. 1988

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

Gratuit

CA2PQ
UP 5
R88-15

CONSUL

Tous droits réservés. On ne peut reproduire ni diffuser aucune partie du présent ouvrage, sous quelque forme que ce soit, sans avoir obtenu au préalable l'autorisation écrite de l'auteur.

Dépôt légal, 2e trimestre 1984
Bibliothèque nationale du Québec
Bibliothèque nationale du Canada

Pour ce procurer une copie de ce document, s'adresser au:

Editions de l'Ecole Polytechnique de Montréal
Ecole Polytechnique de Montréal
Case postale 6079, Succursale A
Montréal (Québec) H3C 3A7
(514) 340-4000

Compter 0,10$ par page (arrondir au dollar le plus près) et ajouter 3,00$ (Canada) pour la couverture, les frais de poste et la manutention. Régler en dollars canadiens par chèque ou mandat-poste au nom de l'Ecole Polytechnique de Montréal. Nous n'honorerons que les commandes accompagnées d'un paiement, sauf s'il y a eu entente préalable dans le cas d'établissements d'enseignement, de sociétés ou d'organismes canandiens.

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;

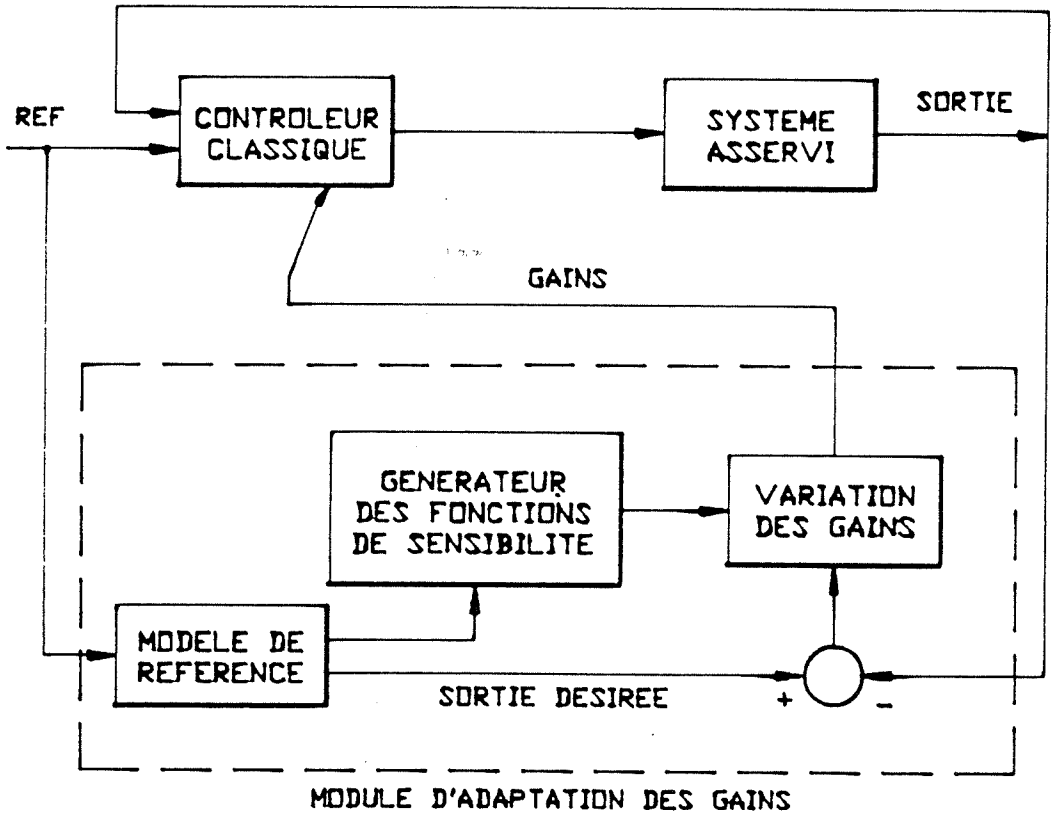

Figure 2.1: Schéma d'une commande adaptative avec modèle de référence.

- 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 contrôleur~~s~~ 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:

$\frac{\delta \underline{e}}{\delta \underline{\alpha}}$: 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:

$\frac{\delta \underline{y}}{\delta \underline{a}}$: 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:

$$\dot{\underline{\alpha}} = - \Gamma^{(-1)} \frac{\delta \underline{y}^T}{\delta \underline{\alpha}} \; Q \; \underline{e}(t) \qquad (2.8)$$

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).

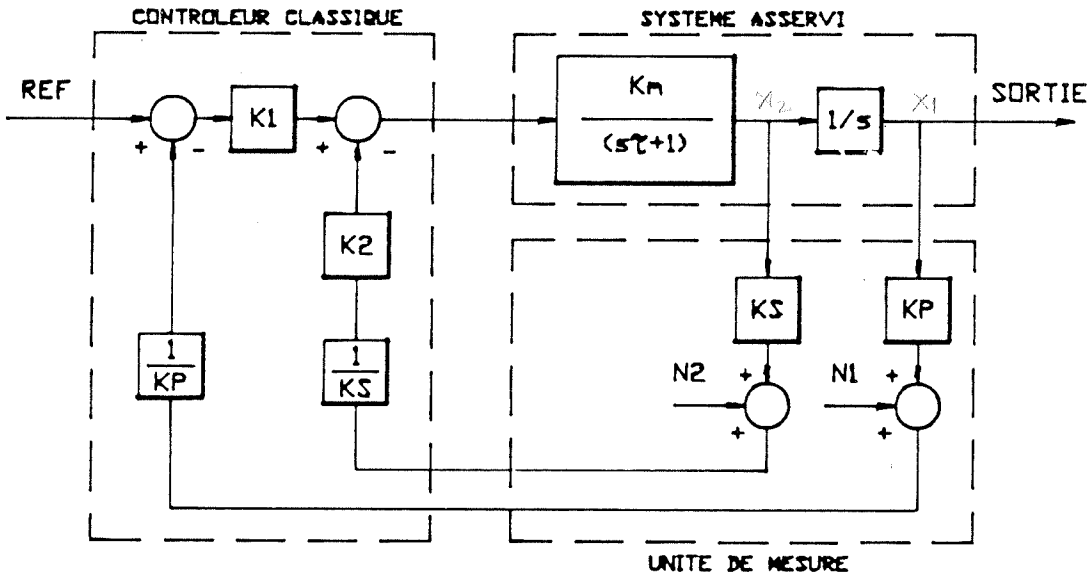

Figure 3.1: Le système en étude.

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

$$\dot{x}_1 = x_2 \tag{3.1}$$

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

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

$$\alpha_1 = \frac{Km\ K1}{\tau} \tag{3.2}$$

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

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.

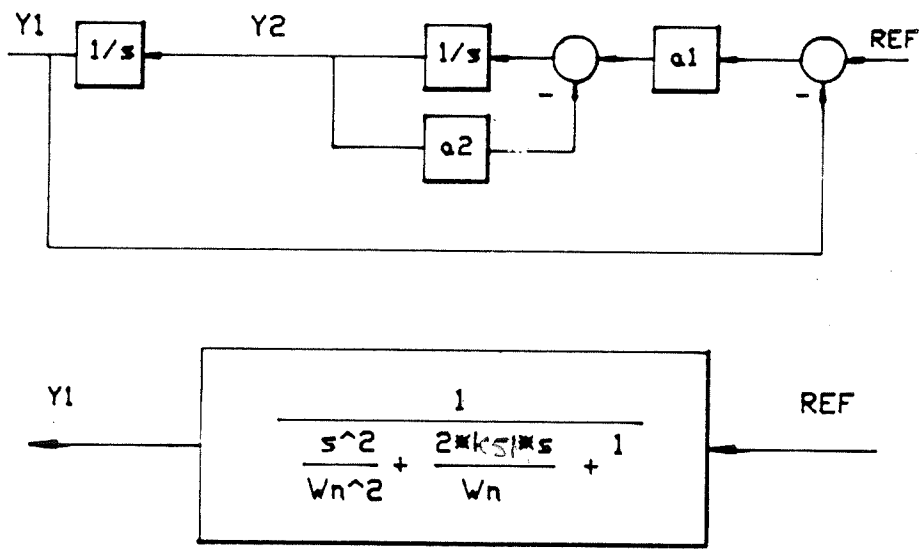

Figure 3.2: Le modèle de référence.

3.3 La matrice des fonctions de sensibilité

La matrice de sensibilité est représentée par:

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

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).

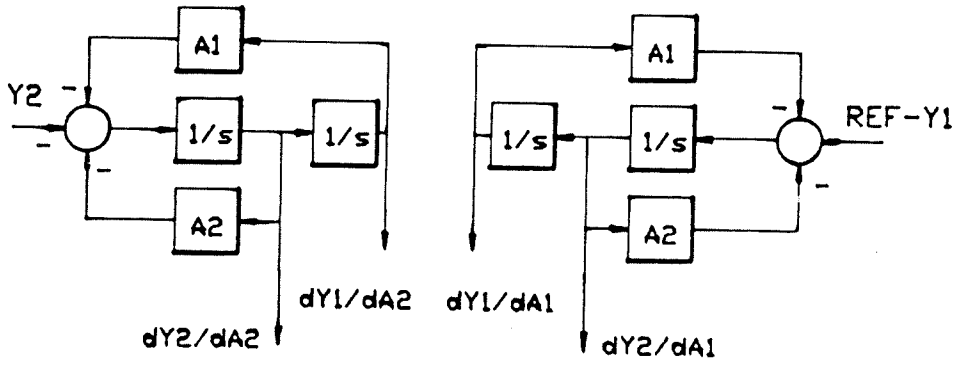

Figure 3.3: Le générateur des fonctions de sensibilité.

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.

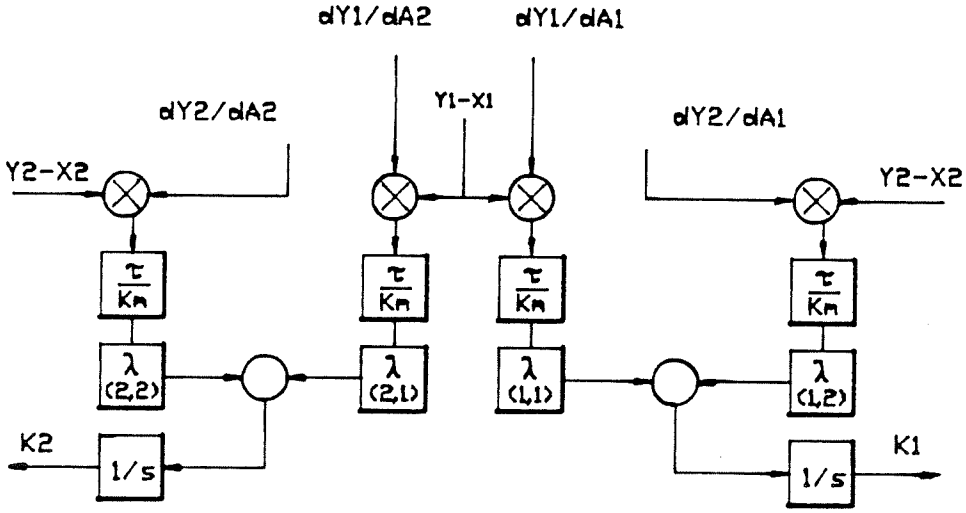

Figure 3.4: Loi de variation des gains.

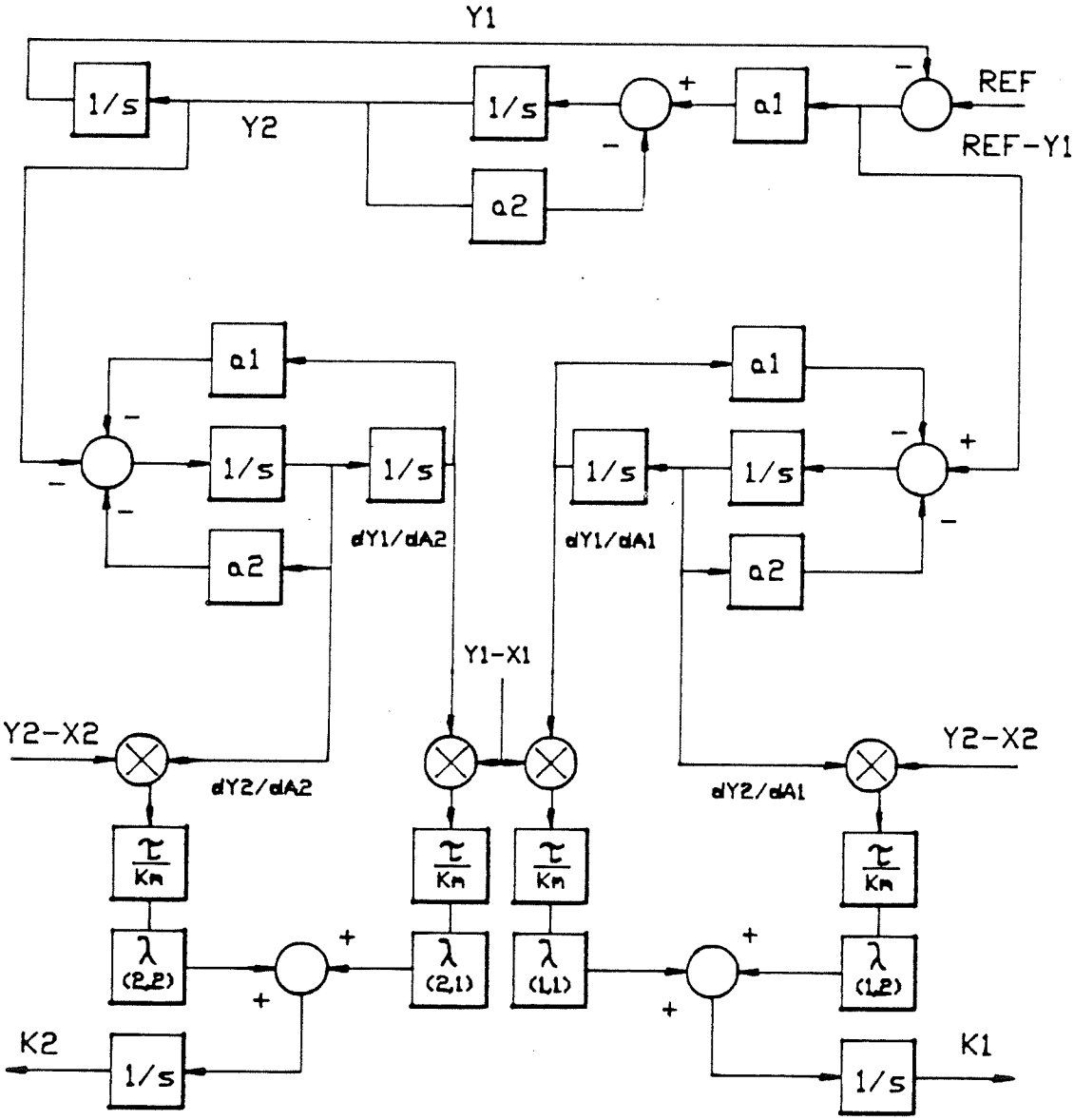

Figure 3.5: Le module d'adaptation des gains.

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.

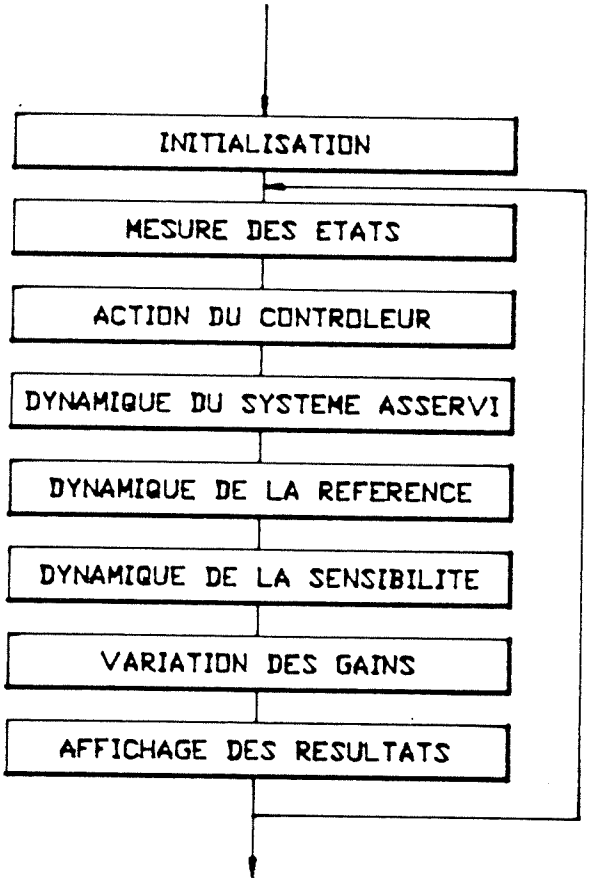

Figure 4.1: La structure du progiciel MRAC.

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).

PD 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	

PD 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.

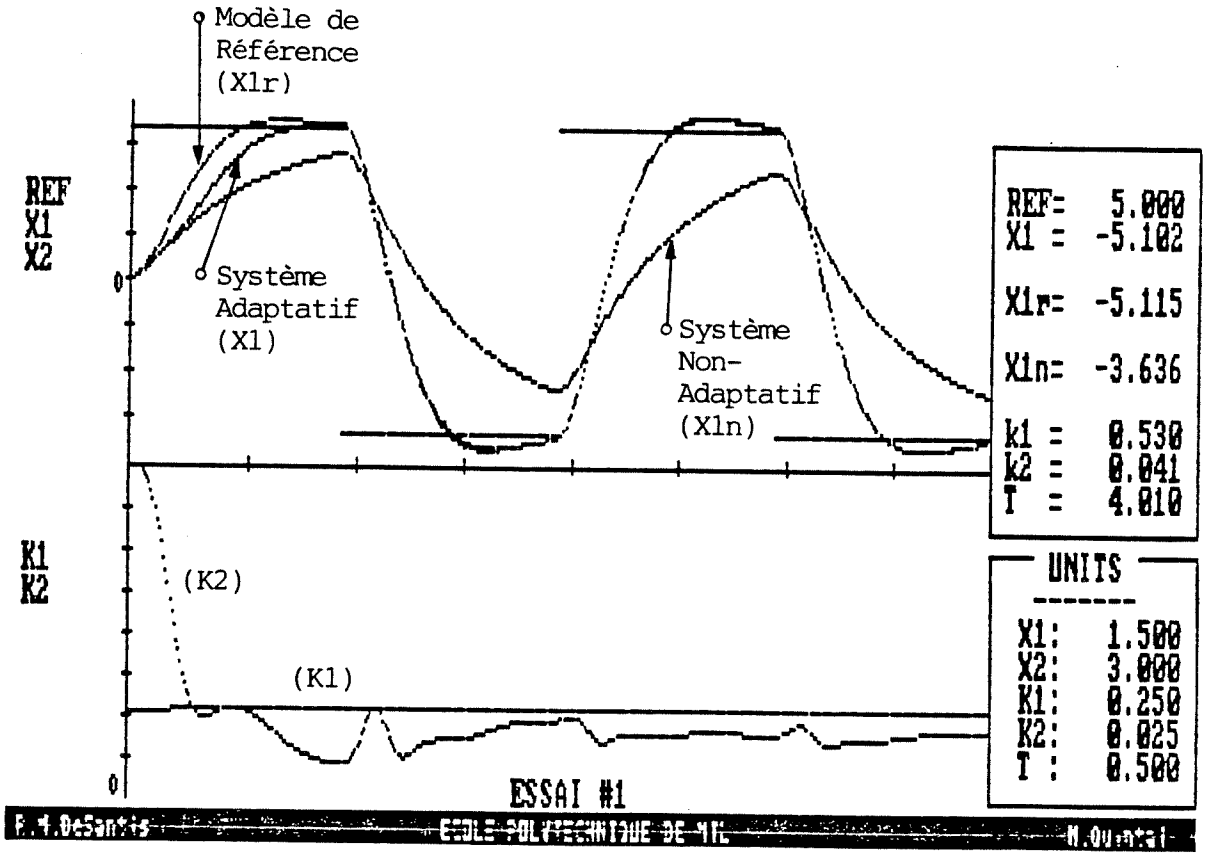

Figure 5.2: Un exemple d'application: Dynamique du système avec et sans adaptation.

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[i]);
        { ================== 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 + deltemps;
      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('    ‡              ROMAND 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);
{‡‡‡‡‡‡‡‡‡‡‡‡‡‡‡‡‡‡‡
```

```
                                    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 ***}
{*** mtre 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];
   n            : 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);
```

```
                Readln(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)

ÉCOLE POLYTECHNIQUE DE MONTRÉAL
3 9334 00289630 4