الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

12/89

وزارة التعليم العالي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ÉCOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT Génie Eléctrique

PROJET DE FIN D'ETUDES

SUJET

LOGICIEL D'AIDE A LA CONCEPTION DE MODELES LINEAIRES EN AUTOMATIQUE

Proposé par :

Etudié par :

Dirigé par :

PROMOTION : JUIN 1989

A nos familles

REMERCIEMENTS

Nous remercions Mr GACI pour ses précieux conseils , Mr SARI pour son aide trés précieuse ,ainsi que tout le personnel de l'annexe du centre de calcul et tous ceux qui ont contribué à la réalisation de ce travail .

SOMMAIRE

INTRODUCTION

La contribution du logiciel au développement de l'informatique occupe une part de plus en plus importante. Loin de se spécialiser, les composants de toute nature tendent à acquerir leur specifité par le recours au logiciel. De plus en plus des microprocesseurs et des mémoires, des micro-ordinateurs et des mini-ordinateurs sont utilisés pour assurer des fonctions antérieurement réalisées par des dispositifs materiels spécialisés. De plus, l'imbrication étroite de l'informatique, et des télécommunications, ainsi que la recherche d'une meilleure utilisation des élements nombreux constituant un système informatique, conduisent à la réalisation de logiciels d'exploitation extrêmement complexes et volumineux, qui seuls peuvent apporter aux utilisateurs non techniciens une relative simplicité d'emploi et une adaptabilité indispensable à leurs besoins.

Pour réaliser ces logiciels, pouvoir les maintenir, les modifier et en conserver la pérennité pendant un terme suffisant pour en amortir les coûts de développement, toute une gamme d'outils logiciels et de méthodes d'emploi de ces outils sont développés.

Ainsi prend naissance une véritable ingenierie du logiciel appliquée par les constructeurs d'ordinateurs, les sociétés de service et de conseil en informatique (SSCI) et les grands utilisateurs.

Parallèlement à l'élaboration de logiciels destinés à une utilisation particulière et unique, il existe des logiciels de base ou d'application qui sont conçus ou réalisés en vue d'une utilisation répétée chez de nombreux utilisateurs.

Ces logiciels sont désignés sous le terme de produits programmes ou progiciels. leurs commercialisation constitue un des volets de l'activité des industriels de l'informatique, construction et société de service.

Notre projet consiste donc à réaliser un logiciel qui permet à l'utilisateur de determiner les lois de commande sachant que la période actuelle se caractérise par un retour au domaine temporel avec le développement de la théorie des variables d'état. Celle-ci est à l'origine de la théorie de la commande optimale par rapport à un critère de performance. Elle donne une description trés générale des systèmes multidimensionnels, tout particulièrement des systèmes linéaires à fonctionnement permanent ou intermittant.

Nous avons divisé le travail en quatre chapitres dont le premièr présente la conception assistée par ordinateur en automatique et quelques systèmes de CAO.

Le second chapitre est consacré à la présentation du logiciel en donnant les fonctions des commandes qui constituent ses menus.

Le troisième chpitre donnera l'étude automatique où l'utilisateur pourra faire la simulation, l'analyse et calculer la commande optimale.

Et le dernier apportera quelques exemples d'application.

CHAPITRE I

CONCEPTION ASSISTEE EN AUTOMATIQUE

ETAT DE L'ART

I - 1 Principes généraux sur la CAO [1]

a . Definition

La conception assistée par ordinateur est une technique dans laquelle l'homme et l'ordinateur sont rassemblés pour la solution des problèmes techniques. Le travail est organisé en équipe de telle sorte que chaque élement lui soit associée une tâche bien définie.

Une affectation a été proposée à la CAO, est que l'ordinateur a trois fonctions principales:

- servir le besoin de mémorisation du concepteur,
- amplifier le pouvoir d'analyse et de logique du concepteur,
- organiser l'information pendant et à la fin du processus de conception.

b . Objectifs de la CAO

Les objectifs de la CAO sont nombreux, les plus importants sont:

-augmenter la créativité,
-améliorer la qualité des produits,
-réduire les délais et les coûts de conception,
-vaincre la complexité,
-pallier le manque de main d'oeuvre qulifiée,
-faciliter l'archivage et le manque de circulation de l'information.

c . Mise en oeuvre d'un système de CAO

La structure d'un système de CAO est formée de trois parties:

-un ensemble modulaire d'equipement (unité centrale, unité de disques, processeurs, terminaux, ...),
-un système de base de donnée permettant le stockage et la gestion de l'information,
-une banque d'algorithmes comprenant des bibliothèques de programmes.

d . L'écriture des programmes d'un système de CAO

La mise en oeuvre des programmes d'un système de CAO doit traiter les questions suivantes:

-la définition des interfaces du dialogue homme machine,
-la définition des types de structure de donnée manipulable par l'ensemble des programmes de traitement et utilisateurs,
-la communication des données entre les programmes,
-la définition de la structure de données graphiques,
-la question d'optimisation des programmes.

e . Choix d'un système de CAO

Le choix d'un système de CAO depend des problèmes de conception à résoudre et des moyens disponibles.

f . La CAO en automatique

D'une manière générale, en automatique, on a pas d'objet physique à concevoir, mais schema de commande d'un processus physique. C'est ainsi qu'on parle de conception de commande assistée en automatique.

Le travail d'un automaticien se déroule en quatre phases pour tout schema de commade d'un processus:

1- Obtention de la représentation du systéme (le modèle) souvent complexe et non linéaire. Cette étape constitue la phase modélisation.

2- Obtention des propriétés du modèle par étude de simulation et des propriétés structurelles. C'est la phase analyse du modèle.

3- A partir des résultats obtenus des deux premières étapes, on détermine la structure et les lois de commande adéquates aux processus. C'est la phase synthèse du modèle.

4- La dernière phase est la mise en oeuvre du schema de commande, c'est à dire, l'implémentation dans le processus physique.

Pour réaliser toutes ces phases, l'automaticien a besoin d'un outil informatique qui:

- le libère des tâches répétitives,
- lui permette d'utiliser les algorithmes complexes sans avoir à les programmer.

Donc l'élaboration de systèmes de CAO d'automatique nécessite des compétences en automatique, en informatique et en analyse numérique.

I - 2 . Systèmes de CAO

Nous allons présenter quelques systèmes de CAO qui traitent des problèmes variés d'automatique.

A- SIGA+ Conception assistée par ordinateur dans la commande des systèmes

SIGA+ est un système interactif pour la CAO dans l'analyse la simulation et le contrôle technologique.

Il intègre beaucoup de programmes pour les calculs spécifiques associés aux grandes étapes de la méthodologie de commande des systèmes et utilise un langage de haut niveau spécifique à l'automatique.

Ainsi nous présentons la version améliorée du SIGA+ qui traite particulièrement la commande des systèmes linéaires décrits dans le domaine fréquentiel ou le domaine temporel.

La figure 1 décrit les grandes étapes de la méthodologie de conception du modèle assistée en automatique.

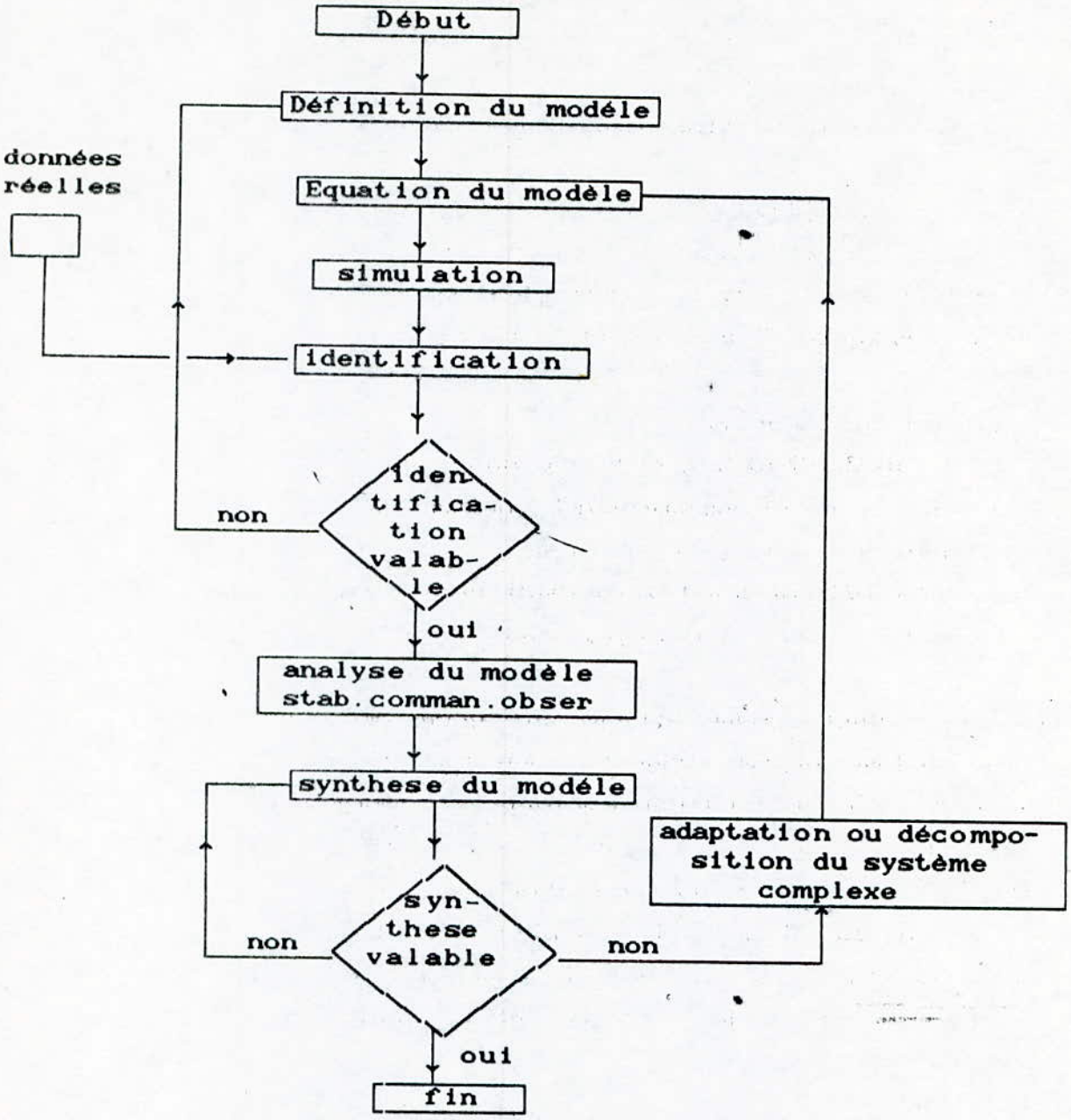

Début
Définition du modèle
données réelles
Equation du modèle
simulation
identification
identification valable
non
oui
analyse du modèle stab.comman.obser
synthese du modèle
synthese valable
non
non
oui
adaptation ou décomposition du système complexe
fin

figure 1

A - 1 . Elements fondamentaux

SIGA+ est un système de CAO conçu pour le developpement de la commande automatique.Il a essentiellement trois objectifs:

1- faciliter le dialogue utilisateur-système,

2- introduire des programmes généraux pour l'analyse, la simulation et la commande des systèmes,

3- présenter les facilités de l'utilisation graphique.

En général SIGA+ peut être défini comme un système d'ordinateur qui automatise les étapes de la méthodologie de conception.

L'implantation de ce système nécessite la résolution des quatre problèmes suivants:

P1- Interpretation du langage de haut niveau.
P2- Implementation et standardisation des programmes.
P3- Sortie graphique.
P4- Coordination du système global.

Les élements qui constituent SIGA+ sont:

Interpreteur : execute l'analyse lexique, syntaxique et semantique de commande et expressions du langage de haut niveau.

Librairie : contient les routines standard du traitement numérique, fonctions externes, matrices et opérations polynômiales.

Structure de donnée : assure les representations internes et externes des modèles.

Software graphique : il est développé dans le but d'utiliser l'ecran graphique et le traceur pour la sortie des résultats.

Module de programmes : contient les programmes interactifs associés aux étapes de commande de processus. Nous retenons les huit modules ayant les fonctions suivantes:

SIMPAC : simulation package, permet d'obtenir les réponses fréquentielles et/ou temporelles des systèmes continus et/ou discrets.

ANAPAC : system analysis package , traite la stabilité, sensibilité, gouvernabilité et observabilité des systèmes.

TRNPAC : system transformtion package(module de transformation) la transformation continu → discret, temporel ← → freqentiel.

IDNPAC : identification package, structure d'idenditification , parametres d'estimation.

SYFPAC : synthesis and design package, module de synthese et de conception dans le domaine frequentiel.

SYFPAC : module de synthese et de conception dans le domaine temporel (commande optimale, observateur de Luenberger...).

POLPAC : pole placement package, placement de pôle pour les systèmes monovariables et multivariables.

AGRPAC : agregation et décomposition des modules de systèmes complexes.

Editeur : assiste l'utilisateur dans l'ensemble des directives et permet de modifier , de sauvegarder, d'executer et d'effacer des programmes écrits en en SIGA+.

A - 2 . Interface du dialogue homme-machine .

Cette interface est composée d'une description spécifique du langage pour la commande automatique. Ce langage permet l'assimilation des notions automatiques telles que les modèles.

La structure modulaire du SIGA+ est donnée par le schema de la figure 2.

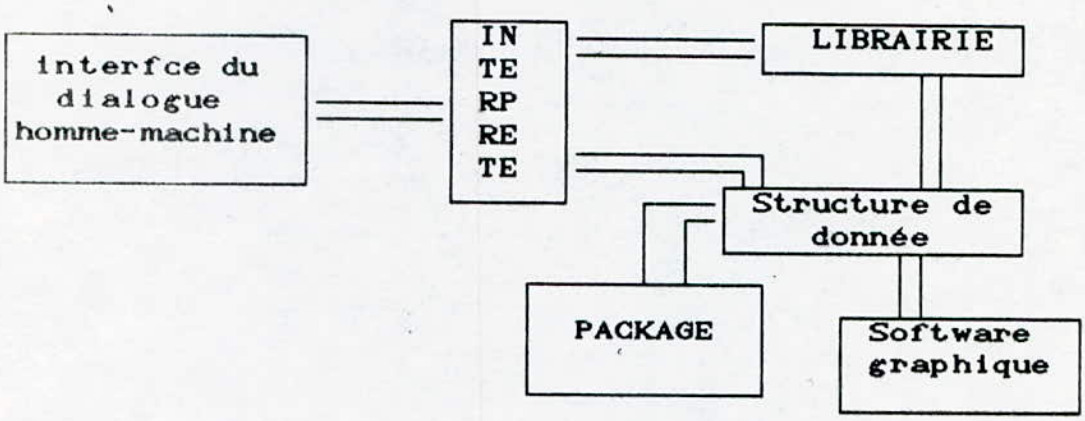


figure 2

B- Le système interactif **Blaise** pour le contrôle engineering [2]

Le système Blaise a été conçu pour faire face aux contraintes des systèmes interactifs et pour répondre aux éxigences de la transportabilité et le traitement numérique. Il introduit un langage de programmation complet travaillant sur la base de donnée adaptée aux problèmes de la commande automatique.

Blaise incorpore la bibliothèque standard MATLAB manipulée à l'aide d'un langage de commande. IL a été développé de telle sorte qu'il satisfasse les objectifs suivants:

- une base de donnée large
- paramétrisation des macros
- méthodes numériques robustes.

Ces choix ont été faits pour traiter efficacement la variété des problèmes de la commande automatique.

B - 1 . Macros

La notion de macro a été développée pour utiliser le plus grand nombre, d'algorithmes dans le contrôle engineering. Ces macros admettent plusieurs types de variables comme paramétres d'entrée/sortie et peuvent se referer aux variables définies au préalable.

Pour l'utilisateur, il n'y a pas de difference entre un macro et une fonction fondamentale du système, ils sont totalement reccursifs.

La large utilisation des macros a insité à écrire un compila-teur. Les macros compilés sont deux fois plus rapides que les macros interprétés. Ainsi, l'efficacité du système est basée sur le choix entre les fonctions fondamentales (écrites en fortran) et les macros (écrits en Blaise).

B - 2 . problèmes non linéaires

Généralement la description informatique des problèmes non linéaires est possible seulement par un programme.

Depuis que les macros sont des éléments de données fondamentales, Blaise peut manipuler ces problèmes, car les fonctions non linéaires peuvent être définies comme macros.

B - 3 . Interface avec macsyma

A un stade de développement, le besoin de calcul non numérique est apparu, par exemple le calcul du jacobien, le calcul des équations adjointes dans les problèmes de commande, l'analyse et les matrices de trasfert.

On a développé quelques programmes spécifiques et de petites interfaces qui permettront de transmettre les variables des données fondamentales de Blaise à Macsyma. On a généré des macros implantant les formules analytiques, dérivées des calculs symboliques de Macsyma

Le système Blaise est assez bien adapté aux solutions numériques des problèmes de commande, il contient environ 12000 lignes fortran et la bibliothèque contient environ 100000 lignes.

C- SIRENA+ Système interactif souple pour la simulation, l'identification et la conception

Sirena+ est un système interactif écrit en fortran 77, et a été installé dans une grande variété de calculateurs, sa strucrure modulaire (700 routines) assure sa maintenabilité et son evolution. C'est un système qui travaille en temps réel et utilise le concept de langage de commande comme moyen de communication avec l'utilisateur.

Le module simulation de Sirena+ permet de prendre en considération les modèles dynamiques décrits de différentes manières.

La simulation suit les étapes suivantes:

- description structurée et formelle des systèmes à simuler
- description des entrées (signaux:impulsion, rampe, sinusoide...)
- définition d'un intervalle de temps ou de fréquence

La méthode de simulation depend du type de système (s'il est continu il sera transformé en discret).

soit le système suivant:

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

c'est la représentation d'état du système continu, son équivalent en discret est:

$$\underline{x}_{k+1} = \phi\,\underline{x}_k + \Gamma\,\underline{u}_k$$
$$\underline{y}_k = C\,\underline{x}_k$$

Dans le cas de non linéarités ou quand la méthode ne s'applique pas, les équations sont intégrées par les algorithmes de Runge Kutta ou Gear. Donc la simulation des systèmes discrets combine toutes les techniques numériques.

C - 2 . Identification

Sirena+ intègre deux méthodes d'identification qui consistent à miniser l'erreur de sortie ou l'erreur de l'équation

de sortie.

Pour définir un système on peut utiliser trois modèles :

$$A(Z^{-1})\,Y(k) = \sum_{i=1}^{r} B_i(Z^{-1})\,u_i(k) \qquad (1)$$

$$y(k) = \sum_{i=1}^{r} \frac{B_i(Z^{-1})}{A_i(Z^{-1})}\,u_i(k) \qquad (2)$$

$$\$\$@\ \frac{y(s)}{u(s)} = \frac{B(s)}{A(s)} = H(s) \qquad (3)$$

La minimisation de l'erreur de sortie qui est une méthode trés générale peut être utilisé aux trois types de modèle.

Pour les modèles (1) et (2) la méthode procède par la réponse échantillonnée du système, le critère est:

$$\theta_{OE} = \arg(\min\Sigma(y_m(k)-y(k))^2)$$

où $y_m(k)$ est la sortie.

Pour le modèle (3) la réponse est fréquentielle, le critère est alors:

$$\theta_{OE} = \arg\,(\min\,\Sigma|H(jw_k)-(V_k+j\,X_k)|^2)$$

où V_k et X_k représentent les parties réelle et imaginaire du point experimental de Nyquist à la pulsation w_k.

La minimisation de l'erreur de l'équation de sortie est appliquée au modèle (1), on utilise trois méthodes:moindres carrés, moindres carés généralisés et maximum de vraisemblance. L'identification suivra les étapes indiqués sur l'organigramme suivant:

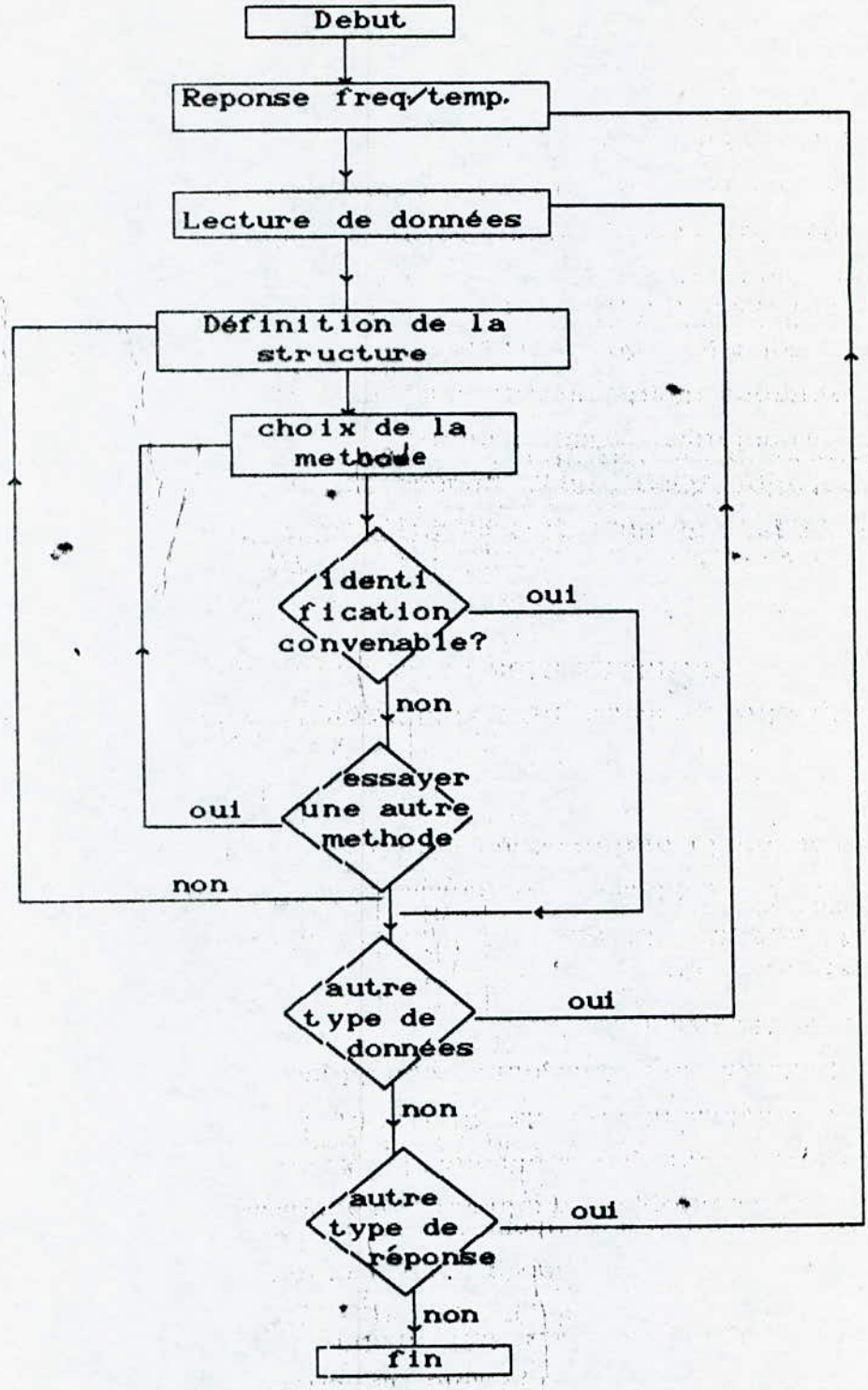

Debut
Reponse freq/temp.
Lecture de données
Définition de la structure
choix de la methode
identification convenable?
oui
non
essayer une autre methode
oui
non
autre type de données
oui
non
autre type de réponse
oui
non
fin

figure 3

C - 3 . Outils de commande

Quelques facilités ont été introduites dans Sirena+ afin d'executer la synthèse par la représentation d'état.

Soit le système suivant:

$$\dot{x} = A\,x + B\,u$$
$$y = C\,x + D\,u \qquad (1)$$

Evaluation de pôle/zéro

Pour le modèle (1), les pôles du système sont obtenus en calculant les valeurs propres de la matrice A.

Placement de pôle

Sirena+ introduit la méthode de Targa (1981) qui transforme essentiellement la paire (A,B) à la forme de Schur par une transformation orthogonale et calcule le gain dans le cas où le système est stable puis retourne aux bases initiales.

Algorithme de commande linéaire quadratique

Pour le calcul des élements de la structure de commande, il est necessaire de distinguer les systèmes continus des systèmes discrets. Ce calcul nécessite l'utilisation des méthodes de Laub et de Barraud pour les deux cas (continu, discret).

C - 3 . Conclusion

Sirena+ est complètement écrit en fortran. Il est actuellement transportable sur de nombreuses machines, il garantit sa maintenance et son évolution. Sirena+ est actuellement commercialisé.

D-CYPROS Outils pratiques de CAO pour le modèle mathémtique

D - 1 . Modèle mathématique

Le modèle mathématique peut être conçu en utilisant l'approche physique ou l'approche de la boite noire. L'ingénieur applique les équations d'équilibre et les lois physiques fondamentales pour générer certaines équations differentielles et algébriques.

La procédure de modélisation est rigoureusement simplifiée si le programme interactif de simulation est disponible. Il est alors facile de vérifier les differentes structures du modèle et voir comment les hypothèses et les valeurs des paramètres agissent sur le comportement du modèle.

D - 2 . Utilisation des programmes interactifs de CAO

L'utilisation de programmes interctifs de simulation, d'identification et d'analyse réduit le nombre de problèmes de construction du modèle. CYPROS (Cybernetic program package) est un systèmes de programmes interactifs conçu pour étudier non seulement les problèmes mais aussi l'analyse et la conception de la commande des systèmes.

La figure (4) illustre la procédure de modélisation et les capcités du système Cypros.

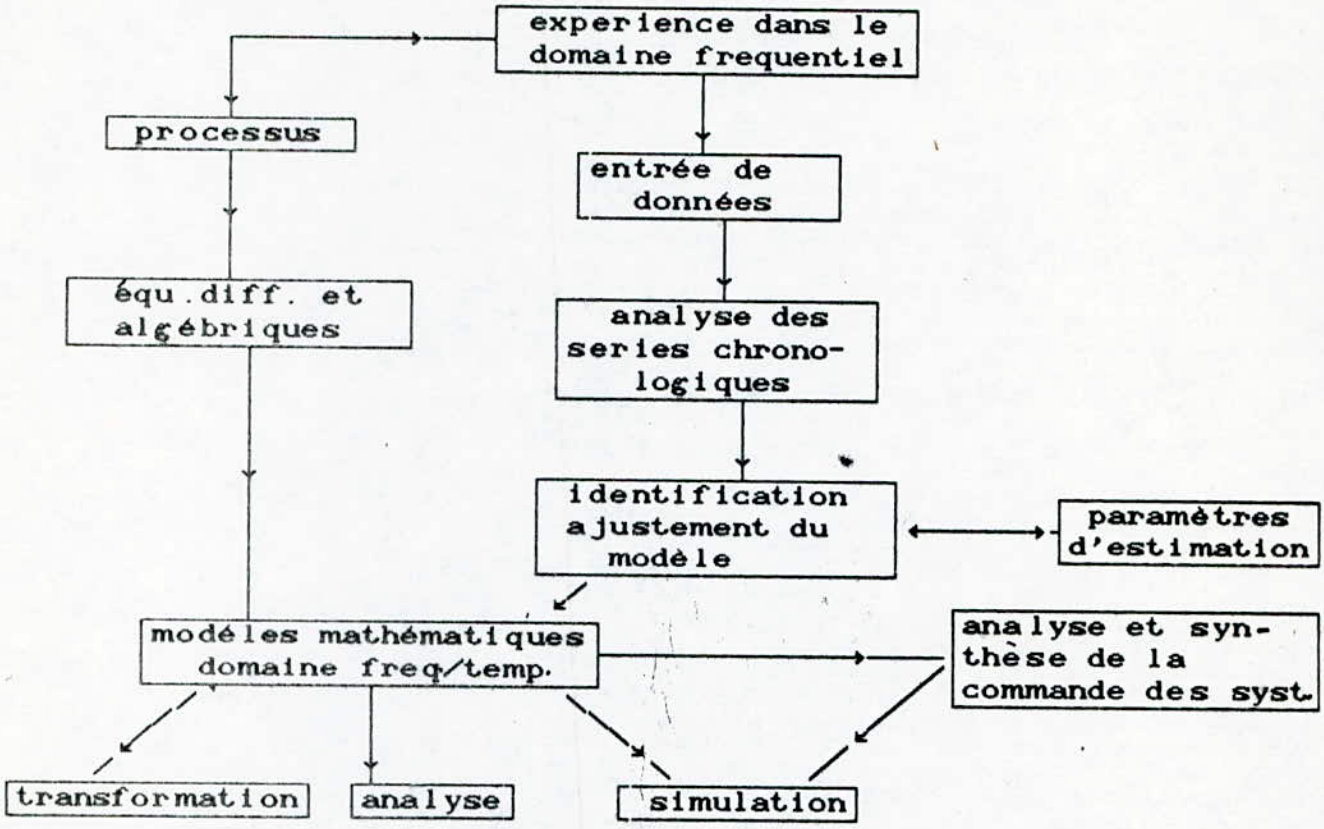


figure 4

Il existe plusieurs méthodes pour résoudre les équations differentielles et les équtions aux differences. Les méthodes les plus couramment utilisées sont celles d'Euler et de Kutta Merson. La méthode de résolution des équations differentielles stiff est aussi introduite.

Le système Cypros peut être implemanté sur un mini aussi bien que sur un micro ordinateur.

E- CACSD. Conception assistée par ordinateur dans la commande des systèmes. Une approche intégrée

La figure ci-aprés récapitule les differentes fonctions assurées par CACSD.

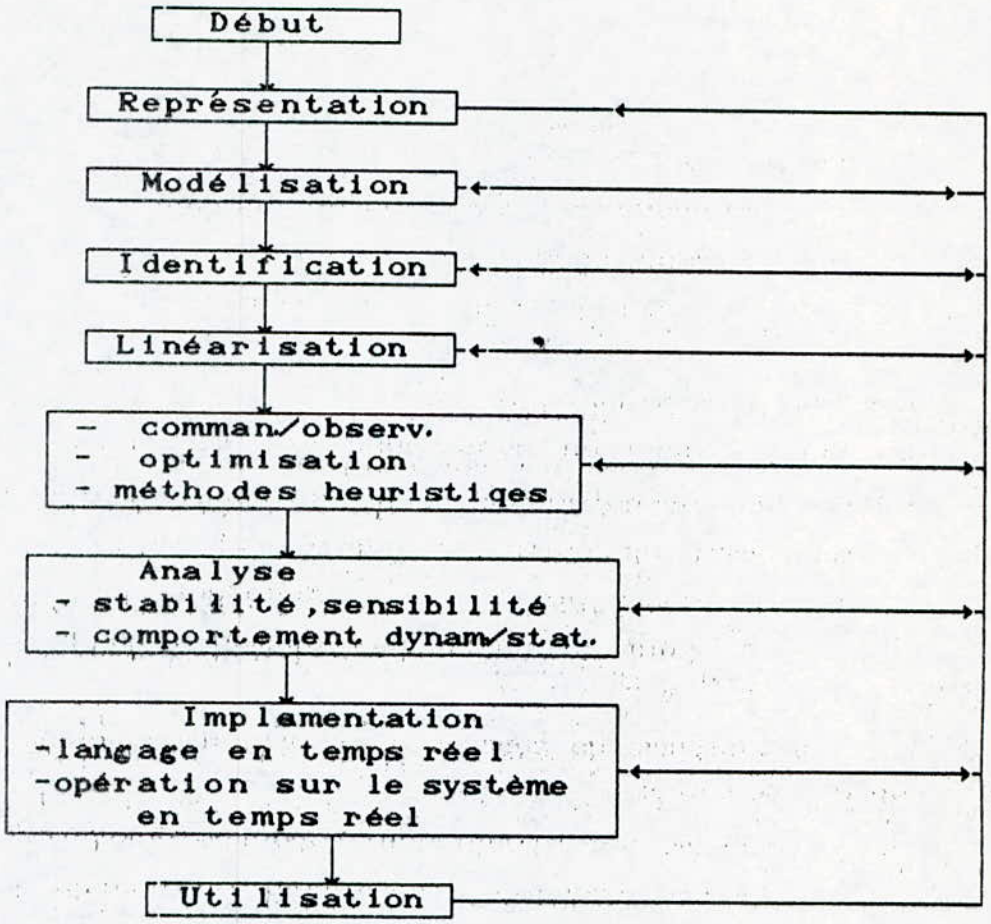


figure 5

Le CACSD peut être un outil d'aide à l'enseignement. Il permét de traiter les problèmes dans des exercices ou des cours de contrôle. En supportant des techniques de conception linéaire aussi bien que non linéaire, il permet aux étudiants de pouvoir observer le comportement du système régulé linéarisé.

La figure ci-dessous montre les utilisations de CACSD dans l'introduction des cours.

Sujet de commande	Utilisation des outils de CACSD dans les exercices
systèmes linéaires représentation d'état et fonction de transfert	stipuler la translation numériquement et algebriquement
formes normales	générer et comparer les formes normales
gouvernabilité et observabilité	générer des tests pour comm/obser. pour des exemples non triviaux qui assistent une interprétation physique
stabilité	tests automatiques de stabilité
commande linéaire des systèmes comportement stat/dynam.	utiliser les caracteristiques graphiques, simulation comparaison avec pôle dominant
systèmes linéaires discrets introduction aux syst. discrets	comparaison entre les comportements des systèmes continus et discrets transformation entre les représentations continues et discretes
système non linéaire	modélisation , linéarisation

figure 6

I - 3 . Conclusion

A travers les systèmes que nous venons de presenter nous pouvons résumer les caracteristiques essentielles que prédéfinissent les systèmes de CAO en automatique.

système	langage	structure	transportable	travaille en temps réel
SIGA+	Langage de haut niveau	modulaire	oui	
SIRENA+	Fortran 77	modulaire	oui	oui
BLAISE	Fortran	programme complet	oui	
CYPROS	Fortran 77	programme complet		temps on line et off line
CACSD	Fortran	programme	oui	

CHAPITRE II

DESCRIPTION DU LOGICIEL

II - 1. Structure du Logiciel [3]

Dans ce chapitre nous allons présenter l'architecture interne du logiciel . Nous rappelons que ce logiciel a été réalisé dans l'objetif d'assister l'utilisateur automaticien ou non à l'analyse , la simulation et la conception de modèles d'automatiques.

Le logiciel est développé sous l'environnement de programmation Turbo-Basic de Borlan INC utilisant le système d'exploitation MS DOS .

Le logiciel tourne actuellement sur la machine Vectra de Hewlett Packard et standardisé de telle sorte à fonctionner sur la norme compatible IBM PC .

Nous avons retenu une structure modulaire dans l'architectùre générale du logiciel . L'ensemble est supervervisé par un menu principal .

Dans ce qui suit , en premiére étape , nous mettons à l'évidence l'organigramme qui représnte la structure globale du logiciel et en deuxiéme étape nous allons décrires les fonctionnalités de chacun de ses modules.

II - 1 - 1. Menu principal

Le menu principal présente cinq (5) options:

- Gestion de fichiers.
- Manipulation de données.
- Simulation.
- Analyse.
- Synthèse.

Ainsi l'utilisateur aura le choix selon ses besoins de travailler sans passer par l'appel des programmes correspondants. Leurs appel et execution se font automatiquement pour n'importe quelle option.

Avant de donner les fonctions de chacune des commandes, nous preférons donner un aperçu sur l'éditeur-analyseur que nous retrouvons dans la majorité des options.

- Editeur Analyseur

- Editeur

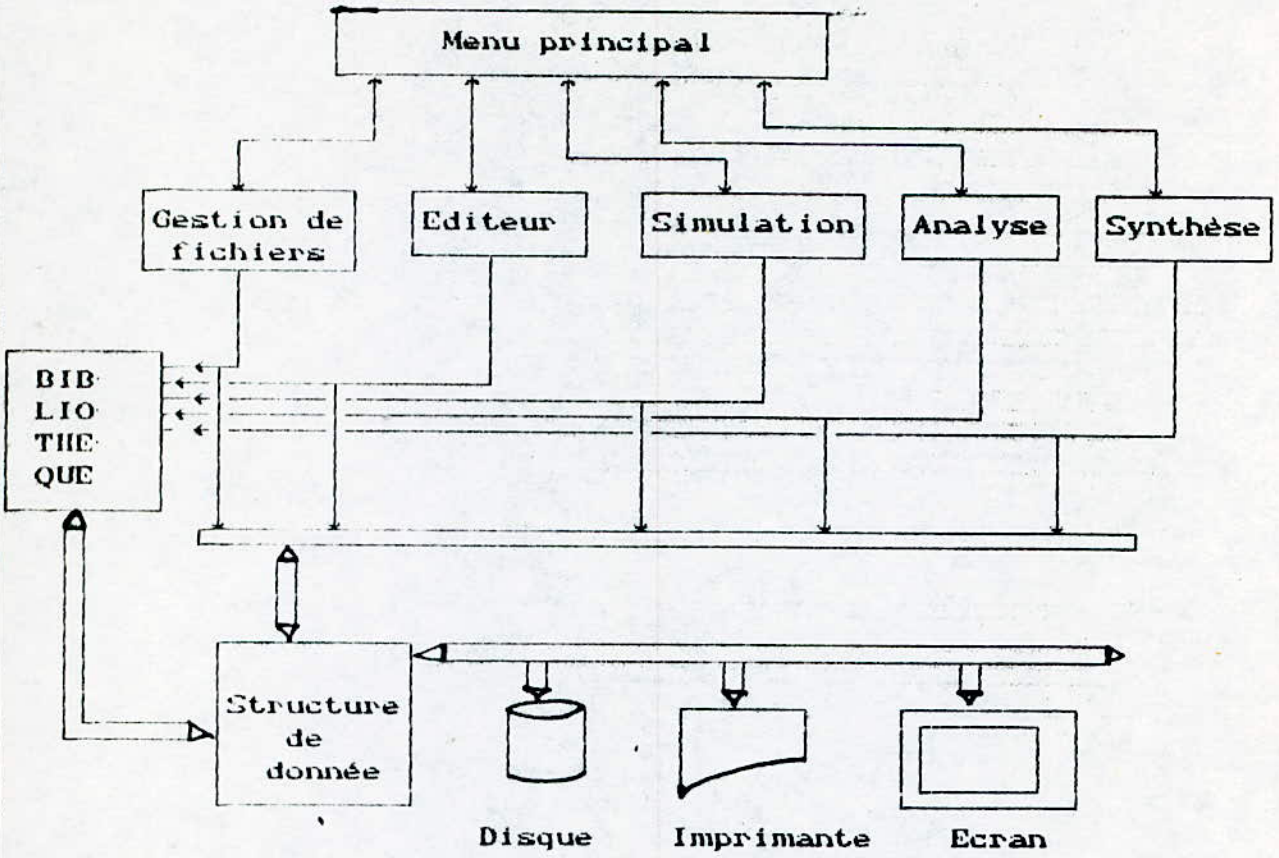
L'éditeur constitue l'interface de communication entre l'utilisateur et le logiciel

En effet les données des modèles sont introduites à partir de l'éditeur .

Il permet de saisir des caractéres numériques ou alphanumériques sur une partie de l'écran ou sur toute la page d'écran .

Il est d'une utilisation trés souple et permet les déplacements du curseur dans les quatre sens, l'effacement des caractères, l'insertion de lignes, la suppression de lignes, etc ...

- Analyseur


Menu principal
Gestion de fichiers
Editeur
Simulation
Analyse
Synthèse
BIB LIO THE QUE
Structure de donnée
Disque
Imprimante
Ecran

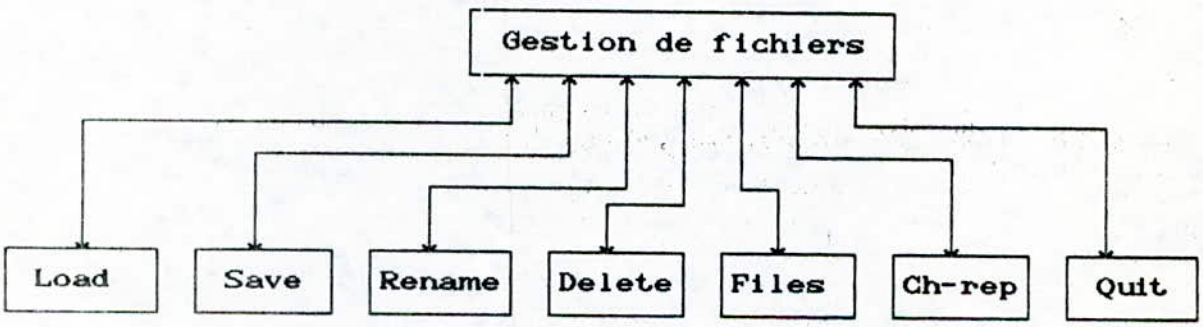
C'est un analyseur syntaxique. Il est utilisé pour détecter les erreurs de syntaxe lors de l'introduction des données et, s'il y a lieu, le type d'erreur.

Il recherche les caractères de 0 à 9, "." , blanc , "+" , et "-". En 1er lieu il recherche soit le point "." soit le moins "-" soit le plus "+" soit le chiffre. Dés qu'il trouve l'un de ces caractères, il va chercher le séparateur indiqué par un blanc ou une virgule. Ensuite il lit la valeur du nombre, et s' il trouve un caractère autre que ceux cités plus haut, il envoie un message d'erreur.

II - 1 - 2. Commandes du menu pricipal

a - Gestion de fichiers

Cette commande permet une communication entre la mémoire de l'ordinateur et l'unité de disques, ou d'operer sur celle-ci.



b - Manipulation de données: Choix de la commande Edit

Il apparait un editeur plein écran qui permet à l'utilisateur d'introduire ses données, de déplacer le curseur, d'effacer et de rectifier en cas d'erreur.

Les noms des données à introduire apparaissent dés l'appel de l'éditeur. On fait entrer:

- les ordres des matrices,
- l'horizon de simulation,
- les conditions initiales,
- les élements des matrices.

L'écriture des élements des matrices peut se faire de manière quelconque, sauf qu'il faut séparer ces élements par des blancs ou des virgules.

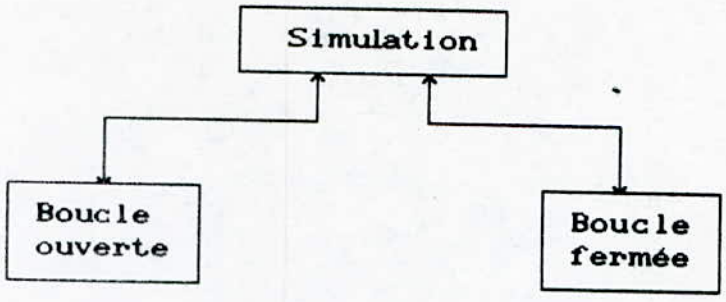
Saisie de données

Les données introduites sur l'éditeur ne sont pas immédiatement mémorisés mais elles peuvent être sauvegardées, de l'éditeur, par la commande "CTRL/A" ou à partir du menu par la commande "Save".

La saisie se fait par ordre , l'analyseur lit d'abord la matrice A ensuite la matrice B et puis la matrice C. Chacune de ces matrices sera mise dans une pile .

Le sous-programme qui fait la sauvegarde s'appelle "SAUV". Son principe est simple.Il ouvre un fichier dans lequel il stocke les données.Ainsi,ces données sont gardées en mémoire.

Choix de la commande simulation



En appelant simulation les commandes suivantes apparaissent:

- Boucle ouverte

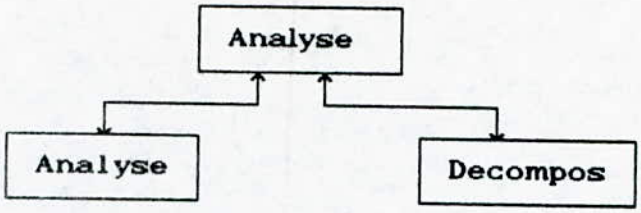
La simulation se fait en boucle ouverte.

- Boucle fermée

La simulation se fait en boucle fermée.

Dans les deux cas, il apparait une petite fenêtre où il est indiqué les types d'entrées à introduire. Lorsque le choix est fait, une autre fenêtre apparaitra dans laquelle il faut introduire soit les coefficients de l'entrée, soit tout le polynôme qui décrit la fonction d'entrée.

d - Choix de la commande Analyse



Les commandes du sous- menu

- Analyse

Détermine la stabilité, la commandabilité et l'observabilité du sytème.

- Décomposition

Permet de décomposer le système en sous systèmes.

e - Choix de la commande synthèse

Détermine la commande optimale du système.

II - 1 - 3.Bibliothèque mathématique

Elle contient tout les sous-programmes de calcul. A l'appe d'une des commandes Analyse, Simulation ou synthèse, l'information et les paramètres sont envoyés vers la bibliothèque qui à son tour les transmet aux sous - programmes de calcul correspondants.

La procédure de cette bibliothèque s'appelle Math, elle détecte la commande activée et appelle les sous-programmes qui font le calcul demandé par l'utilisateur.

II - 1 - 4. Edition des résultats

La sortie des résultats peut se faire soit sur écran, soit sur imprimante, soit sur table traçante.

L'édition peut être numérique ou graphique.

- Résultats numériques:

Les résultas sont mis dans un fichier et envoyés à l'écran ou à l'impriante selon le choix de l'utilisateur.

- Résultats graphiques:

L'édition graphiques des résultats permet de les apprécier sous forme de tracé. Les entrées/sorties (conversation avec l'ordinateur) sont interactifs (choix du mode d'écran, du nombre de courbes à représenter, des courbes à représenter etc ...)

- Gestion de l'écran:

Il n'est pas multicadres, la dispositon des axes et de l'origine est automatique (elle varie selon les valeurs minimums et maximums des résultats).

L'écran a une bonne gestion (multicourbes, disposition des axes, échelles linéaires, multiaxes etc...); il y a aussi possibilité de rappeler les courbes précédentes.

II - 2. Conclusion

Nous pouvons dire que le logiciel ainsi décrit a une structure modulaire. Il contient trois procédures principales: la procédure Menu fait le menu principal et les menus auxiliaires, la procédure Editeur construit l'éditeur, décrit plus haut, ainsi que tous les autres qui apparaissent lorsque certaines commandes sont activées, La procedure Math qui élabore la bibliothèque mathématique.

Le logiciel est indépendant de la configuration du micro - ordinateur: d'autres micro - ordinateurs peuvent le lire.

Il permet un dialogue homme - machine par l'intermédiaire de ses menus.

CHAPITRE III

TRAITEMENT AUTOMATIQUE

III - 1 . Analyse [4]

Toute tentative d'automatisation d'un processus physique nécessite une analyse au préalable des propriétés du modèle le représentant .

En effet ,l'analyse de modèle , a pour objet la mise en évidence des indices de qualités qu'il posséde .

D'une maniére générale , en automatique , l'analyse de modèles, consiste à l'étude de la stabilité , de la gouvernabilité et de l'observalité .

Dans le cadre de ce chapitre , nous allons décrire les algorithmes trés pratiques , implémantés dans notre logiciel et permettant de traiter les questions relatives à l'analyse des modèles linéaires d'automatique .

III -1 -1 . Stabilité :

Un système est dit stable, lorsque ses états atteignent le régime permanent , lors de l'exitation de ses entrées .

Le théorème utilisé pour étudier la stabilité du système est celui des valeurs propres .

Un système est dit stable si les parties réelles de ces valeurs propres sont toutes négatives .

Soit le système décrit par les équations suivantes :

$$\dot{x} = Ax + Bu \qquad (1)$$

$$y = Cx$$

A(n,n), B(n,m), C(r,n)

Les valeurs propres du système s'obtiennent en résolvant l'équation :

$$\det(A - \lambda I) = 0$$

C'est un polynôme de degré n, d'inconnue λ .

$$P(\lambda) = \alpha_n \lambda^n + \alpha_{n-1} \lambda^{n-1} + \ldots + \alpha_0$$

donc si $Re(\lambda i) < 0$ alors le système est stable .

On peut résumer la stabilité dans ce qui suit

- Calcul des valeurs propres λi de la matrice A

- Test des valeurs propres λi { si au moins une valeur propre $\lambda i \geq 0$ alors système instable sinon système stable

III - 1 -2. Commandabilité - Obsevrvabilité :

Les propriétés de commandabilité et d'observabilité représentent une certaine régularité du modèle auquel elles sont appliquées .

- Commandabilité :

La commandabilité traduit l'influence de l'entrée sur l'état et pose le problème de l'existance d'une commande faisant passer le système d'un état initial à un autre état en un temps fini .

Pour tester la commandabilité du système (1), on construit une matrice U constituée des vecteurs colonnes des matrices B, AB, A^2B, ... , $A^{n-1}B$. S'il existe n vecteurs linéairement indépendants, c'est à dire que le rang de la matrice U est n, alors le système est commandable.

$$U = [\, B \quad AB \quad A^2B \quad \ldots \quad A^{n-1}B \,]$$

Si Rang(U) = n alors le système est commandable .

- Observabilité :

L'observabilité traduit l'influence de l'état sur la sortie et pose le problème de la déduction de l'état initial du système à partir de l'observation en un temps fini de la sortie .

Pour tester l'observabilité, on construit une matrice V constituée, dans ce cas, par les vecteurs colonnes :

$$C^T, A^T C^T, (A^T)^2 C^T, \ldots, (A^T)^{n-1} C^T .$$

L'existance de n vecteurs linéairement indépendants (Rang(V) = n) rend le système observable.

$$V = [\, C^T \; A^T C^T \; (A^T)^2 C^T \; \ldots \; (A^T)^{n-1} C^T \,]$$

Algorithme

- 1 - Lire A , B ou A , C
- 2 - k = 1
- 3 - faire le produit (A , B) ou (A , C) = Q
- 4 - construire U = (B AB ...) ou (C^t $A^t C^t$ ...)
- 5 - si Rg(U) < n alors aller à 7 sinon aller à 10
- 6 - incr k
- 7 - faire produit (A , Q)
- 8 - si k < n alors aller à 9 sinon aller à 11
- 9 - retour à 4
- 10 - système commandable aller à 12
- 11 - système non commandable
- 12 - fin

Nous présenterons ultérieurement l'organigramme du test de dépendence linéaire .

III - 1 - 3 . Décomposition en sous-systèmes :

Nous cherchons à décomposer le système décrit en (1,1), de dimension n, en sous-systèmes de différentes classes .
Ces classes représentent les sous-espaces de controlabilité et ou d'observabilité . .

Pour ce faire, on calcule la matrice modale du système (1). Celle-ci est constituée des n vecteurs propres associés respectivement aux n valeurs propres de la matrice A .

Il s'agit donc de calculer les vecteurs propres du système.

Les valeurs propres s'obtiennet d'après la relation :

$$P(\lambda) = \det (\lambda I - A) = 0$$

On obtient la matrice Λ diagonale formée par les valeurs propres λ_i.
On tire les vecteurs propres de la formule suivante :

$$A\, v_i = v_i \Lambda$$

où v_i sont les vecteurs propres du système i=1,..,n
La matrice modale est alors :

$$M = [\ V1\ \ V2\ \ \dots\ Vn\ ]$$

On forme les matrices $\hat{A}$, $\hat{B}$ et $\hat{C}$:

$$\hat{A} = M^{-1} A\, M$$
$$\hat{B} = M^{-1} B$$
$$\hat{C} = C\, M$$

Le nouveau système sera donc :

$$\left.\begin{array}{l} \dot{Z} = M^{-1} A\, M\, Z + M^{-1} B\, u \\ y = C\, M\, Z \end{array}\right\} \quad \text{ou encore} \quad \left\{\begin{array}{l} \dot{Z} = \hat{A}\, Z + \hat{B} u \\ y = \hat{C}\, Z \end{array}\right.$$

Ainsi la matrice $\hat{A}$ apparait sous forme d'un certain nombre de blocs de Jordan $\hat{A}(i)$ associés aux diverses valeurs propres λ_i selon le schéma ci-dessous :

$$\hat{A} = \begin{bmatrix} \hat{A}(1) & & & & \\ & \hat{A}(2) & & & \\ & & \ddots & & \\ & & & \hat{A}(i) & \\ & & & & \ddots \\ & & & & & \hat{A}(r) \end{bmatrix}$$

Si A1 est le bloc de Jordan associé à λ et $\bar{A}1$ est le complexe conjugué de A1, alors $\bar{A}1$ est le bloc de Jordan de $\bar{\lambda}$.

Introduisons la transformation équivalente $x = \bar{P}\, x$

Alors

$$P = \begin{bmatrix} I & I \\ i\,I & -i\,I \end{bmatrix} \qquad (i^2 = -1)$$

$$P^{-1} = 1/2 \begin{bmatrix} I & -i\,I \\ I & i\,I \end{bmatrix}$$

On peut donc vérifier facilement l'equation dynamique:

$$\begin{bmatrix} \bar{X}1 \\ \bar{X}2 \end{bmatrix} = \begin{bmatrix} \text{Re } A1 & \text{Im } A1 \\ -\text{Im } A1 & \text{Re } A1 \end{bmatrix} \begin{bmatrix} \bar{X}1 \\ \bar{X}2 \end{bmatrix} + \begin{bmatrix} 2\text{Re}\, b_i \\ -2\text{Im}\, b_i \end{bmatrix} U$$

$$Y = \begin{bmatrix} \text{Re } C1 & \text{Im } C1 \end{bmatrix} \begin{bmatrix} \bar{X}2 \\ \bar{X}1 \end{bmatrix}$$

La matrice $\hat{A}$ peut aussi s'écrire de la manière suivante:

$$\hat{A} = \begin{bmatrix} \lambda_1 & 1 & & & & & & \\ & \ddots & & & & & & \\ & & \lambda_1 & & & & & \\ & & & \lambda_2 & 1 & & & \\ & & & & \ddots & & & \\ & & & & & \lambda_2 & & \\ & & & & & & \ddots & \\ & & & & & & & \lambda_r \end{bmatrix} \qquad \hat{B} = \begin{bmatrix} {}_1\hat{B} \\ {}_2\hat{B} \\ \vdots \\ {}_r\hat{B} \end{bmatrix}$$

Ainsi, le système est gouvernable, si la matrice constituée par les dernières lignes de chaque sous-bloc de $\hat{B}$ est de rang égal au nombre de sous-blocs .

Mais un sous-bloc est gouvernable si la dernière ligne du sous- bloc $\hat{B}$ associé à $\hat{A}(i,n_1)$ est non nulle .

Dans le cas où le système n'est pas gouvernable, on choisit le sous-bloc gouvernable qui fera l'objet de calcul des lois de commande .

Cas de dégénéressence

Losrque des blocs de Jordan ont la même valeur propre, on prend une matrice formée des dernières lignes de chaque bloc et on calcule son déterminant. Dans le cas où celui-ci est nul le système n'est pas commandable.

Remarque

La décomposition n'est pas toujours réalisable en raison de la male conception du programme des vecteurs propres.

En effet, les vecteurs propres que nous obtenons sont réels pour n'importe quel système et n'importe quelle application utilisée. Ils sont parfois linéairement dépendants ce qui nous donne une matrice modale singulière et par conséquent nous ne pouvons passer à la synthèse du système.

Algorithme de décomposition :

- calcul des valeurs propres et vecteurs propres .
- Formation de la matrice modale M .
- Calcul de l'inverse de la matrice modale (M^{-1}).
- Calcul de $M^{-1}A\ M$, $M^{-1}B$ et CM .
- Recherche de sous-bloc commandable .

Organigramme du test de dependence linéaire :

Cette soubroutine détecte les dépendences linéaires éventuelles dans une suite de vecteurs ${}^{k}V$ à l'aide de la procédure de Gram-Schmidt et calcule les coefficients de ces dépendences .
Pour ce faire, au fur et mesure de l'introduction des vecteurs ${}^{k}V$, on construit la suite orthonormée ${}^{k}E$ définie par :

$$ {}^{k}E = \frac{V - \sum_{i=1}^{k-1} \langle {}^{k}V\, {}^{i}E \rangle\, {}^{i}E}{\left\| {}^{k}V - \sum_{i=1}^{k-1} \langle {}^{k}V\, {}^{i}E \rangle\, {}^{i}E \right\|} $$

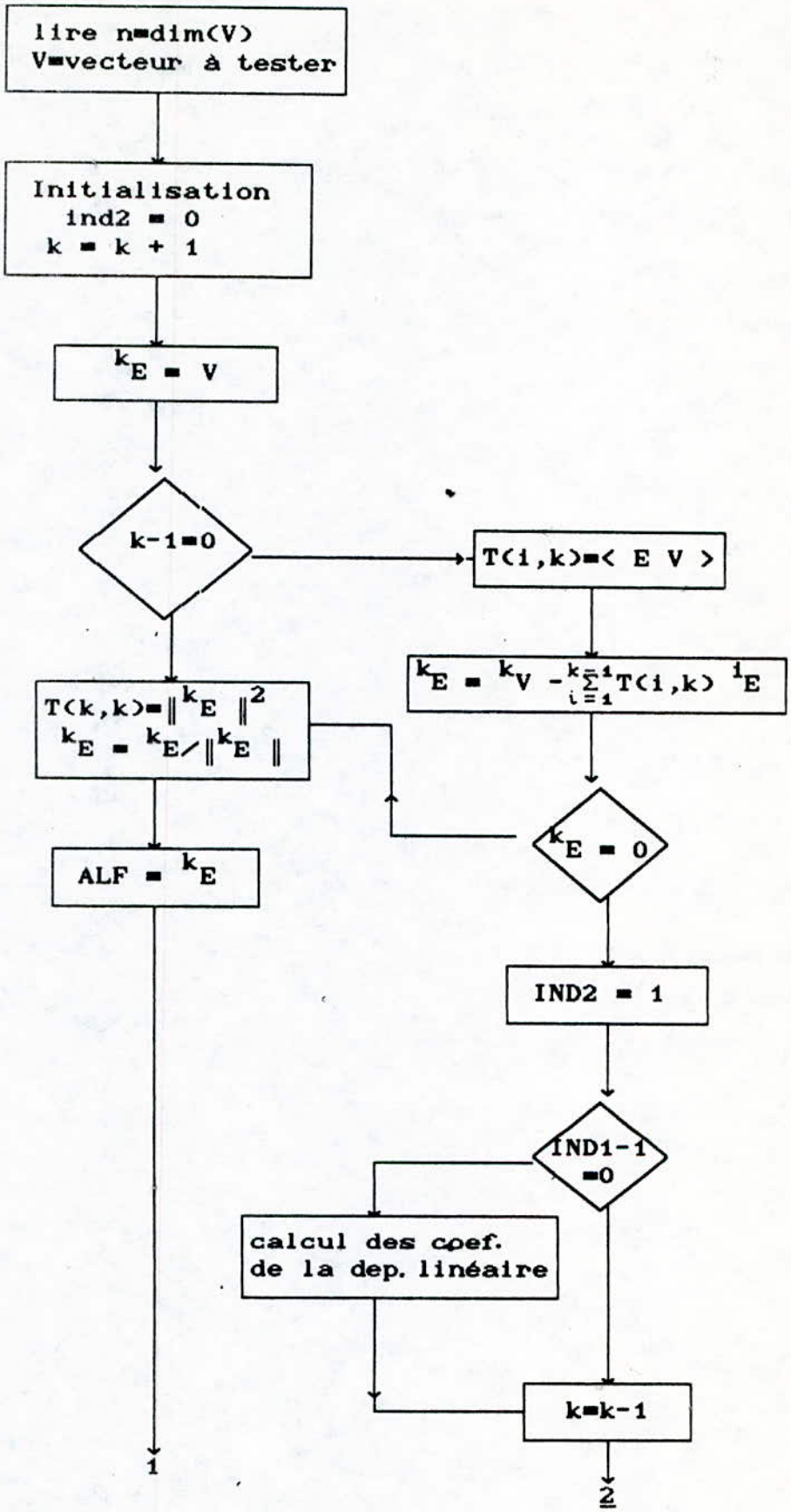
La dépendance de ${}^{k}V$ est alors caractérisée par ${}^{k}E = 0$.
On désigne par α_1 les coefficients de la relation linéaire .

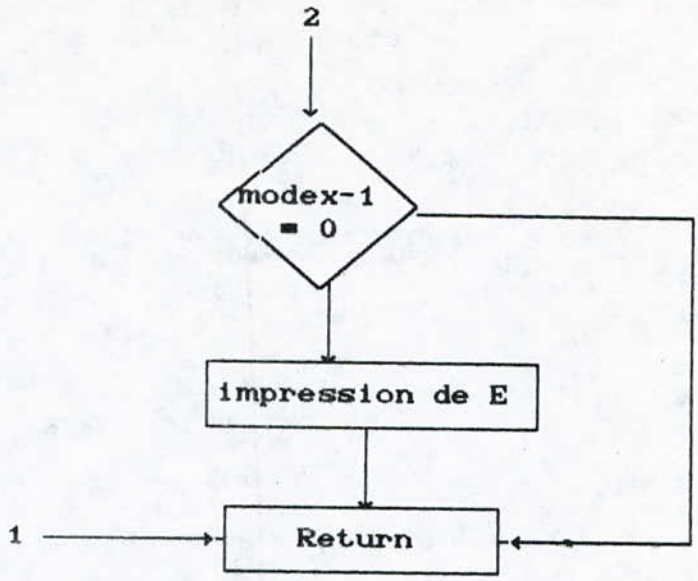
$$ {}^{k}V = \sum_{i=1}^{k-1} \alpha_1\, {}^{i}V \quad ; \quad t_{1k} = \langle {}^{i}E\ {}^{k}V \rangle \quad \text{et} \quad \sum_{i=1}^{k-1} t_{1k}\, {}^{i}E = \sum_{i=1}^{k-1} t_{j1}\, {}^{j}E $$

On en déduit:

$$\alpha_{k-1} = \frac{t_{k-i,k} - \sum_{\rho=1}^{i-k} \alpha_{k-i+\rho,k-i+\rho}}{t_{k-i,k-i}}$$

Ce qui nous donne l'organigramme suivant:


lire n=dim(V)
V=vecteur à tester
Initialisation
ind2 = 0
k = k + 1
$^{k}E = V$
k-1=0
T(i,k)=< E V >
$^{k}E = {}^{k}V - \sum_{i=1}^{k-1} T(i,k)\ {}^{i}E$
$T(k,k) = \| {}^{k}E \|^2$
$^{k}E = {}^{k}E / \| {}^{k}E \|$
$^{k}E = 0$
ALF = ^{k}E
IND2 = 1
IND1-1 =0
calcul des coef. de la dep. linéaire
k=k-1
1
2


2
modex-1
= 0
impression de E
1
Return

III - 2. Simulation

La simulation constitue la partie la plus importante du processus de conception de modèles.

En effet, dans le cadre de notre projet, ce module permet de simuler le comportement effectif des modèles dynamiques continus.

Ces modèles sont généralement décrits par un ensemble d'équations algébro-différentielles. Dans la réalité il existe une grande diversité de modèles dynamiques, du simple au complexe, du lent au rapide, du stiff au non stiff etc ...

Le problème est de trouver une méthode de simulation qui permettra de prendre en considération toutes ces catégories de modèles. Une telle méthode constitue la performance du système de simulation.

En effet, au début des années quatre vingt, les recherches ont abouti à une synthèse de la méthode de Gear adaptée à tout type de modèle dynamique, d'où la determination de la méthode de "Gear généralisée" [5] que nous avons retenue dans le cadre de la simulation.

III - 2 - 1 .Position du problème

Etant donné un modèle dynamique linéaire continu,

$$\dot{X}(t) = A\,X(t) + B\,U(t)$$
$$Y(t) = C\,X(t) \qquad (1)$$

où X: le vecteur d'etat de dimension N
U: le vecteur des entrées ou commande
Y: le vecteur des sorties ou observateur.

Le problème consiste à intégrer le système d'équations (1) sur un horizon fini (t_o,t_f) de telle sorte que la solution soit stable.

III - 2 - 2 .Déscription de la méthode de Gear généralisée

La méthode de Gear appliquée à un système différentiel est une méthode de résolution à pas liés.

L'algorithme de résolution contient deux parties essentielles:

- -prédiction-correction de la solution,
- -determination du pas d'integration (ajustement automatique du pas).

Considérons l'équation:

$$\sum_{j=o}^{k} \alpha_j \, x_{n-j+1} + \sum_{j=o}^{k} h_{n-j} \, \beta_j \, x_{n-j+1} = 0 \qquad (2)$$

Nous en déduisons:

$$\alpha_o \, x_{n+1} = -h_n \, \beta_o \, x_{n+1} - \sum_{j=o}^{k} (\alpha_j \, x_{n-j+1} + h_{n-j} \, \beta_j \, x_{n-j+1}) \qquad (3)$$

avec $h_n = t_{n+1} - t_n$

$$h_{n-j} = t_{n-j+1} - t_{n-j} \qquad j=0,1,\ldots,k$$

Nous recherchons par l'utilisation de cette méthode l'obtention d'une solution donnée par l'expression (3) satisfaisant en même temps le système (1).

Il faut donc déterminer les coefficients α_j et β_j à partir de la résolution de systèmes d'équations.

Pour cela nous avons fait appel aux polynômes d'interpolation qui approximent une fonction x(t) par un polynôme

$P_n(t)$ s'approchant au mieux de x(t) et dont la forme générale est:

$$x(t) \simeq P_n(t) = \sum_{i=o}^{n} a_i t^i$$

polynôme de degré n, ce qui revient à calculer les coefficients a_i grâce à:

$$v(t_o, t_1, \ldots, t_n)\ a = x$$

où $v(t_o, t_1, \ldots, t_n)$ est la matrice de Vandermonde.

Les polynômes d'interpolation qui vont être utilisés sont:

- le polynôme de Newton:

$$P_n(t) = a_o + a_1(t-t_o) + a_2(t-t_o)(t-t_1) + \ldots + a_n(t-t_o)\ldots(t-t_{n-1}) + (t-t_o)(t-t_1)\ldots(t-t_n)\ R_{n+1}(t)$$

- le polynôme de Lagrange:

$$P_n(t) = \sum_{i=o}^{n} x(t_i)\ L_i(t)$$

$$\text{avec}\quad L_i(t) = \prod_{j=o}^{n} \frac{t-t_j}{t_i-t_j} \quad ; \quad i \neq j$$

Pour le calcul des coefficients des polynômes d'interpolation on résoud le système $V^T\ a = f$.

C'est un système dual auquel il correspond un système direct de Vandermonde:

$$V\ x = b$$

a - Méthode prédiction-correction

La prédiction consiste à approcher le point de départ de la zone de convergence en moins d'itérations possibles et la correction améliore cette approche.

α_j et β_j servent à la déterminatin de x_{n+1} et $\dot{x}_{n+1}$, de plus une

valeur prédite est nécéssaire que nous désignerons par $x^{(P)}_{n+1}$.

L'algorithme qui nous concerne sera donc:

1) calcul de $x^{(P)}_{n+1}$

2) calcul de $h\dot{x}_{n+1}$ en prenant $x^{(P)}_{n+1}$ comme valeur de démarrage

3) résolution de $f(x_{n+1}, \dot{x}_{n+1}, t_{n+1})=0$ par la méthode de Newton

4) control d'erreur suivant le principe de Gear

1) Calcul de $x^{(P)}_{n+1}$

$$x^{(P)}_{n+1} = \sum_{i=1}^{k+1} b_i \, x_{n+1-i} \qquad \text{avec } x_{n+1-i} = x(t_{n+1-i}) \tag{4}$$

$$b_i(n,k) = \prod_{j=0}^{k} \frac{t_n - t_{n-j}}{t_{n-i} - t_{n-j}} \qquad i \neq j \tag{5}$$

2) Calcul de la dérivée

$$h\dot{x}_{n+1} = -\sum_{i=0}^{k} a'_i \, x(t_{n+1-i}) \tag{6}$$

$$\alpha_i(n+1, k) = \frac{t_{n+1} - t_n}{t_{n+1} - t_{n+1-i}} \prod_{j=1}^{k} \frac{t_{n+1} - t_{n+1-j}}{t_{n+1-i} - t_{n+1-j}} \tag{7}$$

$$\alpha_0 = -\sum_{i=1}^{k} \alpha_i \qquad i \neq j$$

3) Résolution de Newton

Le système $f(x, \dot{x}, t_{n+1})=0$ est résolu par une itérration du type Newton-Raphson:

$$\Delta x^m = -\left[\frac{\partial f}{\partial x} - \frac{\alpha_o}{h_{n+1}} \frac{\partial f}{\partial \dot{x}}\right]^{-1} f^m$$

$$\Delta \dot{x}^m = -\frac{\alpha_o}{h_{n+1}} \Delta x^m$$

4) Erreur local

Par définition nous avons:

$$E_k = h\,(\,\dot{x}(t_{n+1}) - \dot{x}_{n+1}) + o(h^{k+1})$$

avec $$\dot{x}_{n+1} = -\frac{1}{h} \sum_{i=o}^{k} \alpha_i\, x\,(t_{n+1-i})$$

aprés développement, l'erreur local devient :

$$E_k = \frac{h}{t_{n+1} - t_{n-k}} (x_{n+1} - x^{(P)}_{n+1}) \qquad (8)$$

où $$E_k = \left(\prod_{j=1}^{k} \frac{t_{n+1-j} - t_{n+1}}{h}\right) \frac{h^{k+1}\, x^{(k+1)}(t_{n+1})}{(k+1)!} \qquad (9)$$

Ces deux dérnières constituent l'expression de l'erreur locale.

b - contrôle d'erreur:

Il faut déterminer le facteur de correction adéquat pour le pas sachant qu'il n'est pas connu au départ:

$$h_{n+1} = F_{k,n}\, h_n \qquad (10)$$

avec $$F_{k,n} = \frac{h_{n+1}}{h_n} = \sqrt[k]{\frac{e(h_n)}{E_k(h_n)}} = \left[\frac{EPS\, h_n}{L\, E_k}\right]^{1/k} \qquad (11)$$

c - Détermination de E_k:

La valeur de E_k est le maximum de l'erreur de troncature commise, soit:

$$E_k = \max_j (E_k^j) \qquad (12)$$

j désignant la $j^{ème}$ inconnue.

d - Détermination de $F_{k,n}$:

Il existe un pas optimal qui rend $F_{k,n}$ maximum, celui-ci sera donné par :

$$F_{k,n} = \max (F_{i,n}) \qquad (i=k-1,\ k,\ k+1) \qquad (13)$$

Le nouveau pas sera donc:

$$h_{n+1} = F_{k,n}\ h_n \qquad (14)$$

e - Changement de pas:

Le pas optimal est calculé à chaque intégration, ainsi il est non uniforme et adapté à chaque instant.

f - Changement d'ordre:

Le changement d'ordre n'est fait que si k pas sont éffectués, ceci ellimine la possibilité d'erreur trés grande dans les dérivées de la variable.

g - Utilisation des relations aux différences:

L'utilisation de ces relations modifie les coefficients

b_i et α_i, mais les nouveau coefficients $\tilde{b}_i$ et $\tilde{\alpha}_i$ sont liés d'une façon simple aux premiers.

-Détermination des $\tilde{b}_i$:

$$\tilde{b}_i = - \sum_{i=j+1}^{k+1} b_i \qquad (j=1,\ldots,k) \qquad (15)$$

-Détermination des $\tilde{\alpha}_i$:

$$\tilde{\alpha}_j = \sum_{i=0}^{j} \alpha_i \qquad (j=0,\ldots,k-1) \qquad (16)$$

$$\tilde{\alpha}_k = 0$$

Lors du calcul de $F_{k-1,n+1}$ et $F_{k+1,n+1}$ les valeurs E_{k+1} et E_{k-1} doivent être calculées, ce qui détermine $x_{n+1}^{(P)}(k-1)$ et $x_{n+1}^{(P)}(k+1)$.

Ces valeurs prédites constituent les valeurs de démarrage de la méthode de Gear, elles sont obtenues en résolvant le système (1) par la méthode de Crank-Nicholson [6].

III - 2 - 2. Démarrage de la méthode de Gear généralisée:

Algorithme de Crank-Nicholson:

Cet algorithme permet d'intégrer le système (1) de t_o à t_f:

$$R = I - \frac{h}{2} A + \frac{h^2}{4} A^2 - \frac{h^3}{12} A^3$$

$$S = I + \frac{h}{2} A + \frac{h^2}{4} A^2 + \frac{h^3}{12} A^3$$

$$T = I + \frac{h}{2} A + \frac{h^2}{6} A^2 + \frac{h^3}{24} A^3$$

$$V = I - \frac{h}{2} A + \frac{h^2}{6} A^2 - \frac{h^3}{24} A^3$$

$$w = I + \frac{h}{3} A + \frac{h^2}{12} A^2$$

$$X = I - \frac{h}{3} A + \frac{h^2}{12} A^2$$

$$Y = I + \frac{h}{4} A$$

$$Z = I - \frac{h}{4} A$$

$$Q = \frac{h^4}{48} B$$

- Calcul des valeurs propres λ
- Calcul du pas d'intégration

$$h = \frac{1}{2 \lambda_{max}} \quad , \text{ si } h > 0.1 \text{ alors } h = 0.1$$

- Calcul des dérivées première, deuxième, et troisième de u(t).

début

$t=t_o$, $x=x_o$, $h_k=h$

tant que $t \overset{<}{o} t \leq t_f$

faire

$$x(t+h) = R^{-1} S\, x(t) + R^{-1}\left(T \frac{h}{2} B\, u(t) + V \frac{h}{2} B\, u(t+h) + W \frac{h^2}{4} B\, \dot{u}(t) + X \frac{h^2}{4} B\, \dot{u}(t+h) + Y \frac{h^3}{12} B\, \ddot{u}(t) + Z \frac{h^3}{12} B\, \ddot{u}(t+h) + Q\, \ddot{u}(t) - Q\, \ddot{u}(t+h)\right)$$

fin faire

fin

Algorithme de la méthode de Gear généralisée:

Nous allons décrire les étapes de l'algorithme de la méthode.

Nous sommes à l'instant $t=t_n$, nous avons :

$$\Delta x = (\Delta x_n, \Delta x_{n-1}, \ldots, \Delta x_{n-k})$$

Le pas est h_n et nous voulons passer à h_{n+1}

1) a) calculer $b_i(k,n+1)$ suivant (5)

b) déduire $\tilde{b}_i(k,n+1)$ suivant (15)

c) calculer $x^{(P)}_{n+1} = x_n + \sum_{i=1}^{k} \tilde{b}_i\, \Delta x_{n-1}$

2) a) calculer $\alpha_i(k,n+1)$ suivant (7)

b) déduire $\tilde{\alpha}_i(k,n+1)$ suivant (16)

c) calculer $h_n \dot{x}_{n+1} = -\sum_{i=0}^{k-1} \tilde{\alpha}_i\, \Delta x_{n-i}$

3) Résoudre suivant Newton-Raphson $f(x_{n+1}, \dot{x}_{n+1}(x_{n+1}), t_{n+1}) = 0$

en prenant comme point de départ $(x_{n+1}^{(P)}$ et $\dot{x}_{n+1}(x_{n+1}^{(P)}))$

$$\Delta x^m = -\left[\frac{\partial f}{\partial x} - \frac{\tilde{\alpha}_0}{h_n \frac{\partial f}{\partial \dot{x}}}\right]^{-1} f^m$$

$$\Delta \dot{x}^m = -\frac{\tilde{\alpha}_0}{h_n \Delta x^m}$$

4) a) Calculer E_k suivant (8) en prenant x_{n+1} comme solution de Newton-Raphson.

b) Calculer $F_{k,n+1}$ suivant (11).

c) Si k+1 pas sont pris, calculer $F_{k-1,n+1}$ et $F_{k+1,n+1}$

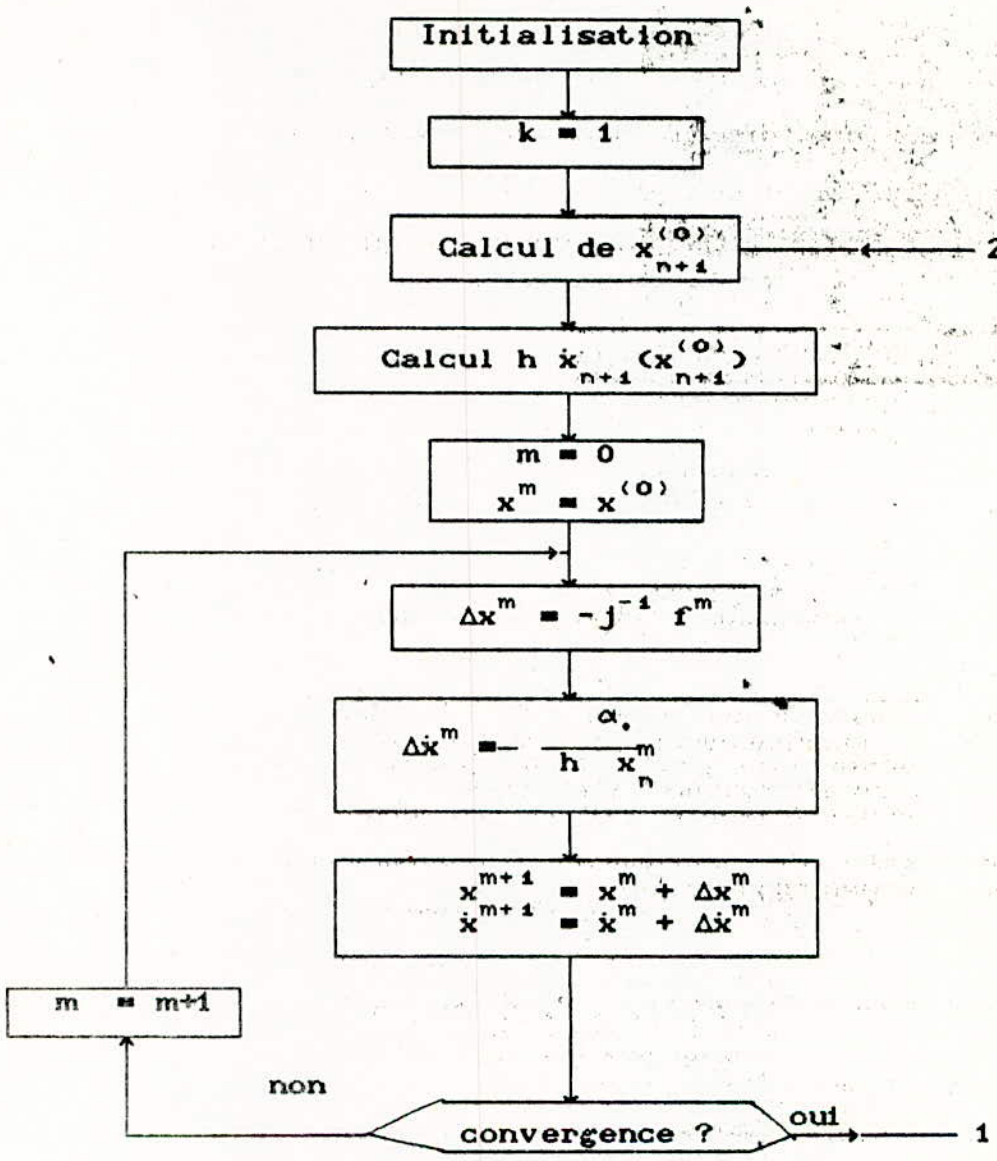
d) $F_{k,n+1} = \max_i (F_{i,n+1})$ i=k-1,k,k+1

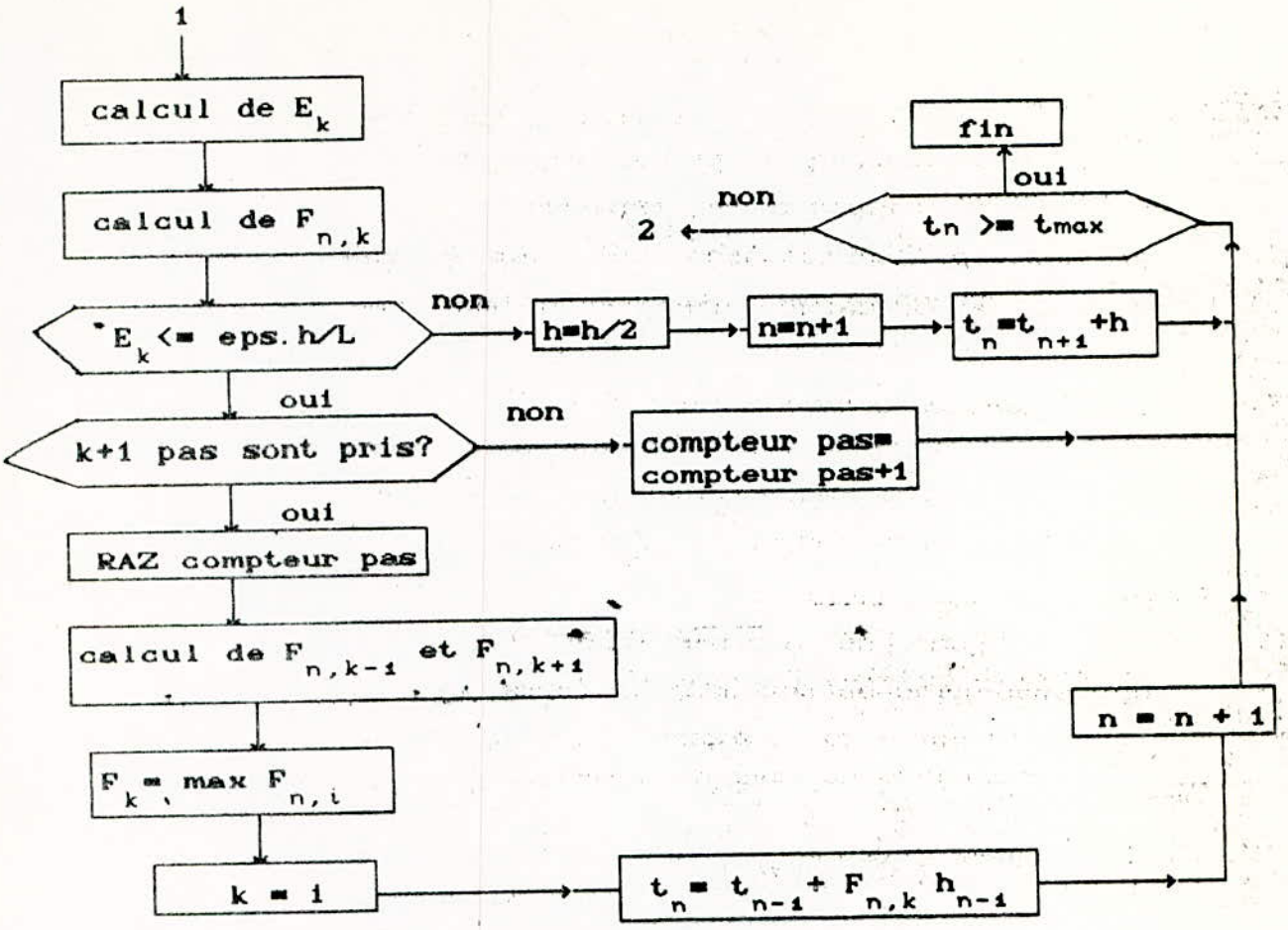
e) $h_{n+1} = F_{k,n+1}\, h_n$

III - 2 - A. Conclusion:

L'algorithme ainsi décrit est déduit à partir de la théorie de polynômes d'interpolation, il est très rapide et permet de résoudre éfficacement un système algébro-différenciel quelconque.

Organigramme:




1
calcul de E_k
calcul de $F_{n,k}$
E_k <= eps.h/L
non
h=h/2
n=n+1
$t_n = t_{n+1} + h$
oui
k+1 pas sont pris?
non
compteur pas= compteur pas+1
oui
RAZ compteur pas
calcul de $F_{n,k-1}$ et $F_{n,k+1}$
$F_k = \max F_{n,i}$
k = i
$t_n = t_{n-1} + F_{n,k}\, h_{n-1}$
n = n + 1
tn >= tmax
non
2
oui
fin

III - 3. Synthèse [7], [8]

Le but de l'automatique est de réaliser la commande des systèmes. Commander un système consiste à agir sur les variables d'entrée de telle sorte que le système réalise le but assigné qui, dans le cas général, pourra se formuler sous la forme d'un certain critère à optimiser.

Dans le cadre de notre étude, nous avons utilisé le critère quadratique. IL permet d'exprimer de manière convenable les qualités globales recherchées pour la commande.

Celles-ci peuvent en fait se résumer, le plus généralement, par la détermination d'une commande assurant le meilleur compromis entre certaines performances (stabilité, précision, énergie appliquée au système...) représentées par des termes de pondération faisant intervenir, les sorties ou les variables d'état. Cette méthode a aussi l'avantage de se prêter à des développements mathématiques nombreux et puissants et correspond à une réalité physique certaine.

Le critère quadratique peut être considéré comme un simple outil mathématique, un moyen élégant pour parvenir à la forme de commande souhaitée.

III - 3 - 1. Principe de la méthode

Soit le système décrit par l'équation d'état:

$$\dot{X} = A X + B U \qquad (1,1)$$

le problème est de trouver une commande qui minimise le critère quadratique suivant:

$$J(x,u) = \int_0^\infty (x^T(t)\, Q\, x(t) + u^T(t)\, R\, u(t))\, dt \qquad (1,2)$$

où Q et R sont des matrices symétriques respectivement semi-définie positive et définie positive.

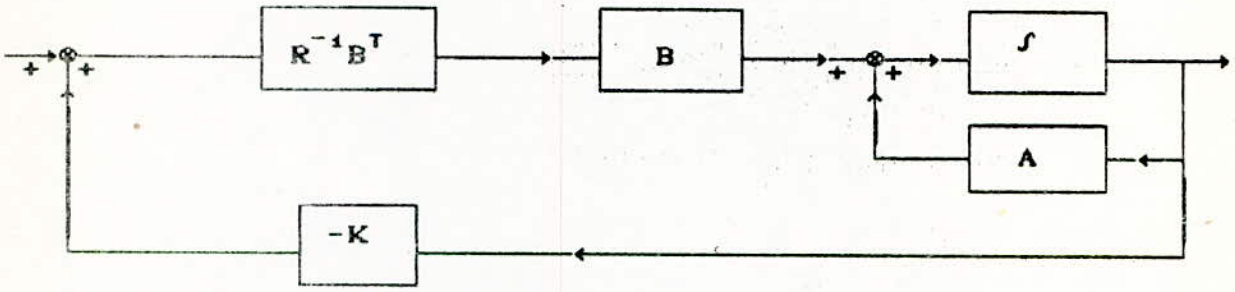
La commande par retour d'état linéaire minimisant le critère J est donnée par:

$$u = - R^{-1} B^T P x , \qquad (1,3)$$

avec P satisfaisant l'équation algébrique de Riccati

$$P A + A^T P - P B R^{-1} B^T P = 0 \qquad (1,4)$$

Le schema fonctionnel est alors:



Les équations (1,1) et (1,2) reprises avec l'expression (1,4) entrainent:

$$\dot{x}(t) = (A - B R^{-1} B^T P)\, x(t) = \tilde{A}\, x(t) \qquad (1,5)$$

et

$$J = \int_{to}^{\infty} x^T (Q + P B R^{-1} B^T P)\, x\, dt \qquad (1,6)$$

L'équation (1,4) peut s'écrire :

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

d'où

$$(A^T - PBR^{-1}B^T P)P + P(A - BR^{-1}B^T P) = - Q - PBR^{-1}B^T P$$

ou encore

$$\tilde{A}^T P + P \tilde{A} = - \tilde{Q}$$

Soit la fonction de Lyapounov V(x) définie par:

$$V(x) = x^T P x \qquad (1,7)$$

$$\dot{V}(x) = - x^T \tilde{Q} x \qquad (1,8)$$

Le critère quadratique s'écrit donc:

$$J = \int_{t_o}^{\infty} x^T \tilde{Q} x \, dt = - \int_{t_o}^{\infty} \dot{V}(x) \, dt$$

nous obtenons:

$$J = -V(x(\infty)) + V(x(t_o))$$

Dans le cas où le système est stable

$$V(x(\infty)) = 0$$

d'où

$$J(x,u) = V(x(t_o))$$

(1,7) devient alors:

$$J = x^T(t_o) P x(t_o)$$

C'est la valeur optimale du critère quadratique relatif au problème défini par les équations (1,1) et (1,2).

III - 3 - 2. Commande sous-optimale

Dans le cas où le système peut être décomposé en sous-systèmes, on peut calculer la loi de commande associée au sous-système commandable et sera utilisée comme loi de commande sous-optimale du système réel.

On définit alors un modèle réduit et un critère quadratique équivalent au critère initiale.

Soit le modèle réduit du système (1,1)

$$\dot{x}_r = A_r x_r + B_r u$$

$$x_r \in \mathbb{R}^m \qquad m < n$$

$$\hat{y} = C_r x_r$$

tel que l'état $x_r(t)$ est relié à l'état $x(t)$ quelque soit $u(t)$ par une relation suffisament précise.

$$x_r(t) \simeq \phi\, x(t) \qquad (1,9)$$

Au modèle (1,4) on associe le critère de performance J_r:

$$J_r = \int_0^\infty (x_r^T(t)\, Q_r\, x_r(t) + u^T(t)\, R\, u(t))\, dt \qquad (1,10)$$

et la commande optimale au sens de J_r :

$$u_r = k_r\, x_r(t) = - R^{-1} B_r^T P_r x_r(t) \qquad (1,11)$$

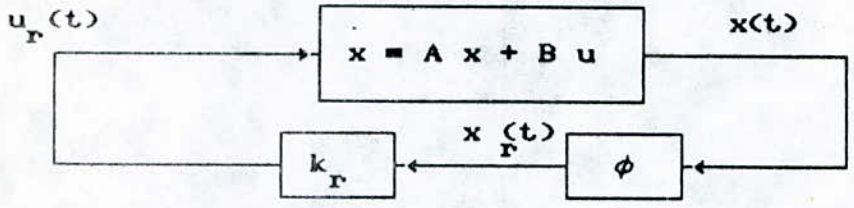
où P_r est la solution de l'équation de Riccati d'ordre m

$$P_r A_r + A_r^T P_r - P_r B_r R^{-1} B_r^T P_r + Q = 0 \qquad (1,12)$$

compte tenu de (1,9) $u_r(t)$ s'écrit:

$$u_r(t) \simeq - R^{-1} B_r^T P_r \phi \; x(t)$$

u_r représente la commande sous-optimale du système (1,1).



III - 3 - 3. Stabilité du système en boucle fermée

L'équation du système en boucle fermée auquel on applique la commande sous-optimale s'écrit:

$$\dot{x}(t) = A\,x(t) - Bk_r\phi(x)$$

$$\phi(x) = Lx(t)$$

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

$$\dot{x}(t) = (A - Bk_rL)x(t)$$

Le système bouclé est donc stable si les valeurs propres de la matrice $(A - Bk_rL)$ sont toutes à parties réelles négatives.

III - 3 - 4. Méthode de résolution

Nous venons de voir que la résolution de l'équation de Riccati est nécessaire à la détermination des élements de la structure de commande. Rappelons cette équation:

$$PA + A^TP - PBR^{-1}B^TP + Q = 0$$

où

$$R^T = R \ , \quad R > 0$$

$$Q^T = Q \ , \quad Q >= 0$$

L'équation est non linéaire en P et, en supposant que la paire (A,B) est commandable, on peut trouver une solution (unique si la paire $(A,Q^{1/2})$ est observable).

III - 3 - 5. Résolution de l'équation de Riccati par la méthode de Newton - Raphson

Cette méthode est basée sur la linéarisation de cette équation donnant un schema itératif de résolution, et nécessite la résolution d'une équation de Lyapounov s'exprimant par la formule suivante:

$$A^TP + PA = - Q \qquad (1,13)$$

La résolution de l'équation matricielle non linéaire (1,4) est effectuée sur la base de la linéarisation de l'équation:

$$H(P) = A^TP + PA - PBR^{-1}B^TP + Q \qquad (1,14)$$

Supposant qu'à l'étape k, la solution P_k trouvée est peu différente de la solution P_{k+1} .

$$P_{k+1} = P_k + V_k \qquad \text{avec} \quad H(P_{k+1}) = 0$$

Approximant $H(P_{k+1})$ autour de P_k et négligeant les termes quadratiques en V_K , nous obtenons:

$$Q + P_kBR^{-1}B^TP_k + P_{k+1}A_k + A_kP^T_{k+1} = 0 \qquad (1,15)$$

ou sous une autre forme:

$$Q + L_k^T R L_k + P_{k+1} A_k + A_k^T P_{k+1} = 0 \qquad (1,16)$$

avec

$$\left.\begin{aligned} L_k &= R^{-1} B^T P_k \\ A_k &= A - B L_k \end{aligned}\right\} \qquad (1,17)$$

Les équations (1,15), (1,16) et (1,17) donnent l'algorithme suivant:

a) Choisir L_o de façon que $A_o = A - BL_o$ soit asymptotiquement stable (assurant la convergence de la méthode itérative).

b) Les itérations P_{k+1} s'obtiennent de la résolution de l'équation de Lyapounov suivante:

$$P_{k+1} A_k + A_k^T P_{k+1} = - Q_k \qquad (1,18)$$

avec

$$\left.\begin{aligned} A_k &= A - B L_k \\ Q_k &= Q + L_k^T R L_k \end{aligned}\right\} \qquad (1,19)$$

c) La solution P de l'équation (1,3) est obtenue par la repétition du point (b) tout en faisant la différence entre P_{k+1} et P_k jusqu'à l'obtention de la précision voulue, soit

$$\| P_{k+1} - P_k \| < \varepsilon$$

- Résolution de l'équation matricielle de Lyapounov

La méthode de résolution consiste à convertir le système (1,13) en un système linéaire d'ordre M (M=N(N+1)/2) mettant à profit la symétrie de P.

$$U\ P_v = R_v$$

où P_v est un vecteur de dimension M constitué des lignes supérieures de P .

Les élements de la matrice u sont obtenus à partir de la matrice A par les formules de reccurences données dans les trois étapes suivantes:

- Première étape

Construction de la matrice dynamique L(NxN) des indices donnant les élements u_{ij}

- Deuxième étape

Construction de la matrice V(MxM) à partir de la matrice A, soit:

$$V_{nm} = A_{ij}$$

où les indices m et n sont donnés par la matrice auxiliaire L:

$$n = L_{ik}$$
$$m = L_{jk} \qquad i,j,k = 1,2,\ldots,N$$

- Troisième étape

La matrice u est obtenue par la multiplication par 2 de tous les élements de lignes de V dont les indices correspondent aux élements diagonaux de L.

III - 3 - 6. Algorithme de Kleiman

Cet algorithme est utilisé pour déterminer la solution de l'équation de Riccati. La méthode se base sur des itérations successives jusqu'à la convergence de la solution. A chaque étape l'équation itérative de Riccati est transformée en une équation de Lyapounov qui sera résolu par l'algorithme de Bingulac.

Soit V_k (k=0,1,2,...) l'unique solution définie positive de l'équation algébrique linéaire:

$$0 = A_k^T V_k + V_k A_k + Q_k + L_k^T R L_k$$

avec

$$Q_o > = 0$$

$$L_k = R^{-1} B^T V_{k-1} \qquad k = 1,2,...$$

$$A_k = A - BL_k$$

où L_o est choisie telle que $A_o = A - BL_o$ possède les valeurs propres à parties réelles négatives.

Donc 1) $k \leq V_{k+1} \leq V_k \leq \ldots \qquad k = 0,1,...$

2) $\lim_{k \longrightarrow \infty} V_k = k$

ou k est la solution exacte de l'équation

$$0 = KA + A^T K + Q_o + KBR^{-1} B^T K$$

1- Initialiser $A_o = A$ (A stable)

2- Résolution de $A_o^T V_o + V_o A_o = -Q_o$

3- Former L_k, A_k $\qquad k = 1,2,...$

4- Former $Q = Q_o + L_k^T R L_k$

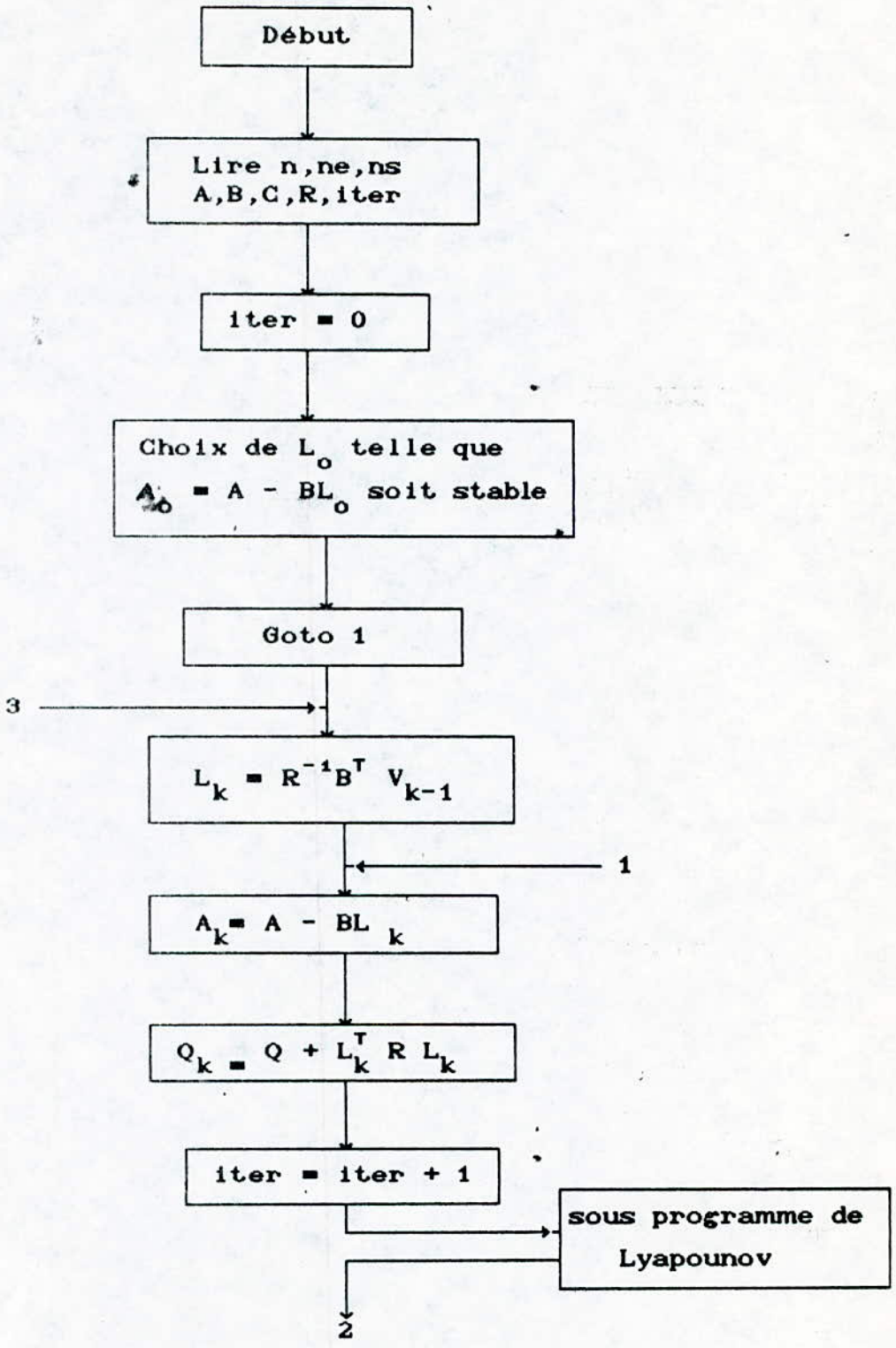
5- Résolution de $A_k^T V_k + V_k A_k + Q = 0$

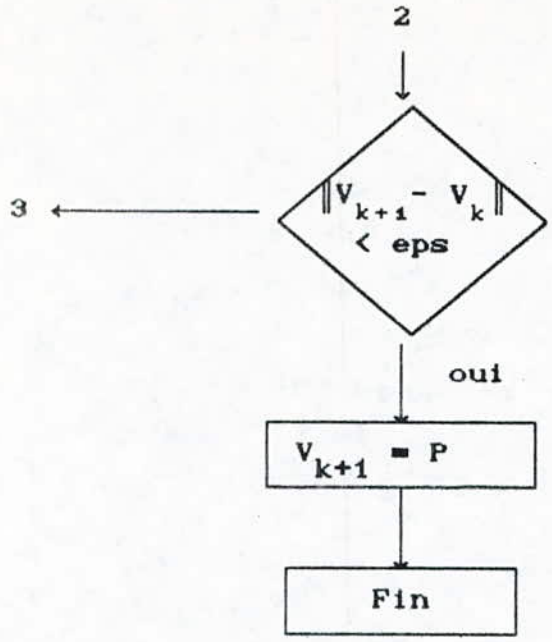
6- Evaluer la norme de $\| V_{k+1} - V_k \| = E$

*- si $E \leq \varepsilon$ afficher V_{k+1}

*- sinon retour en 3

III - 3 - 7. Organigramme de la résolution de l'équation de Riccati

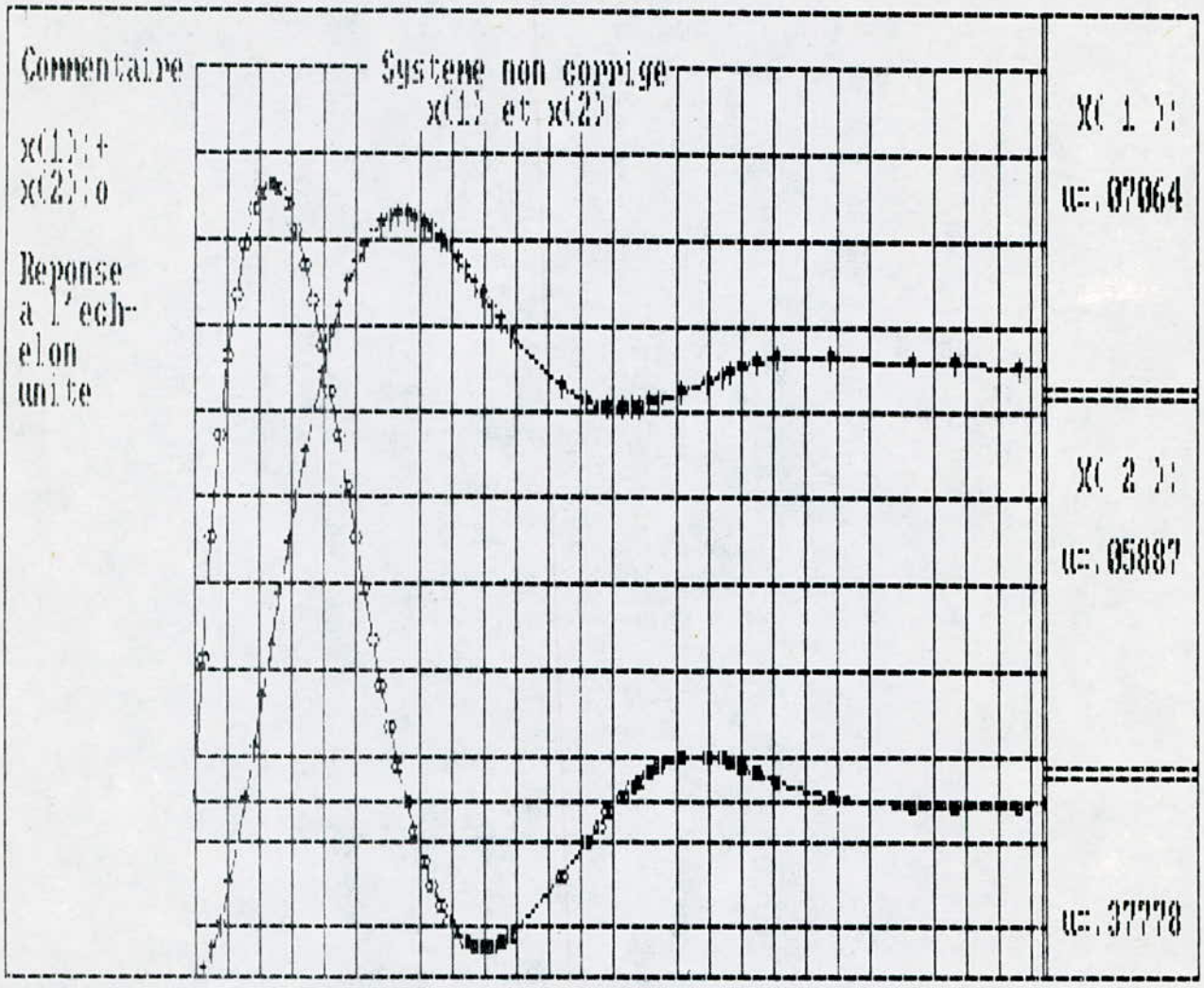


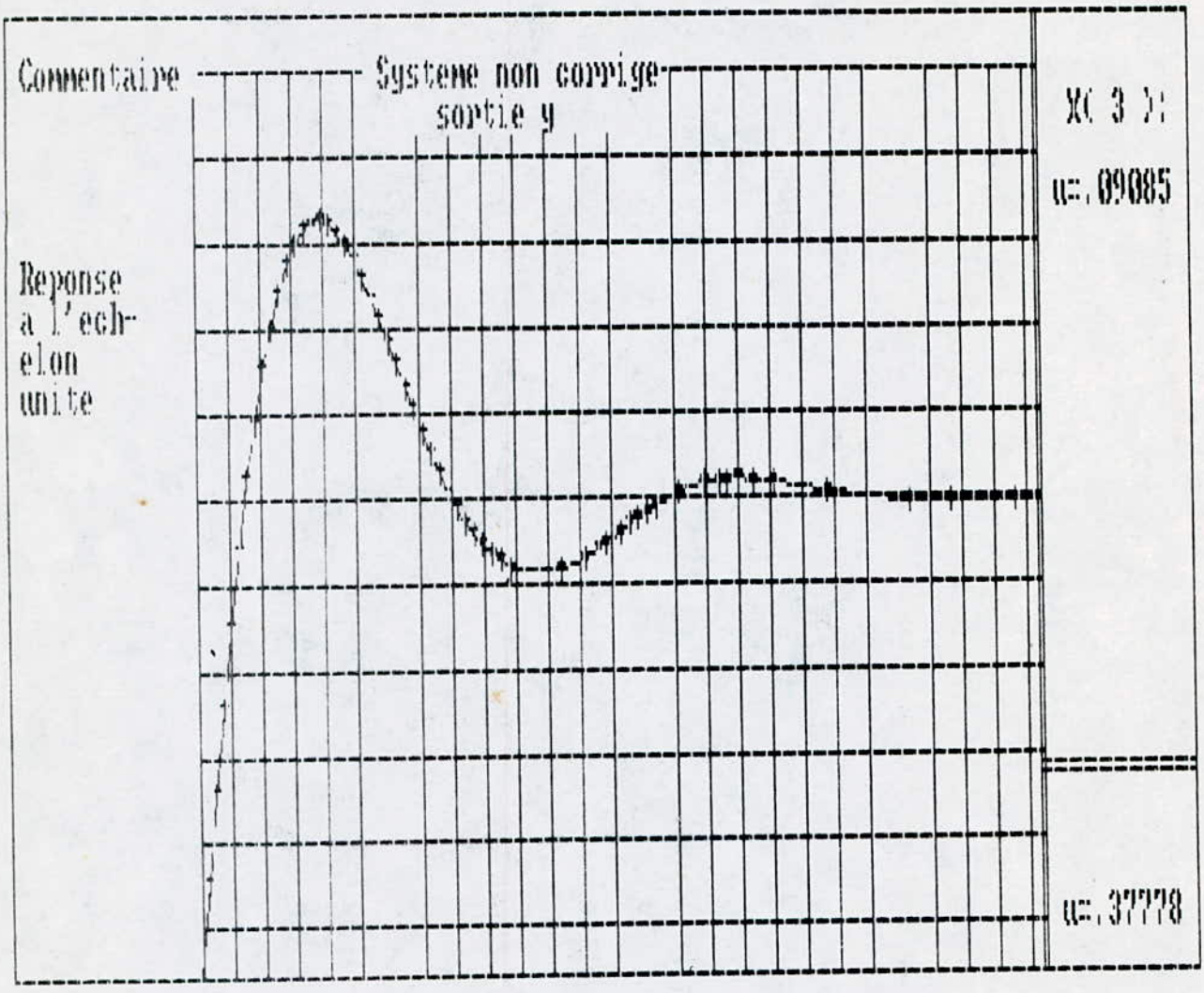

2
3
$\|V_{k+1} - V_k\| < eps$
oui
$V_{k+1} = P$
Fin

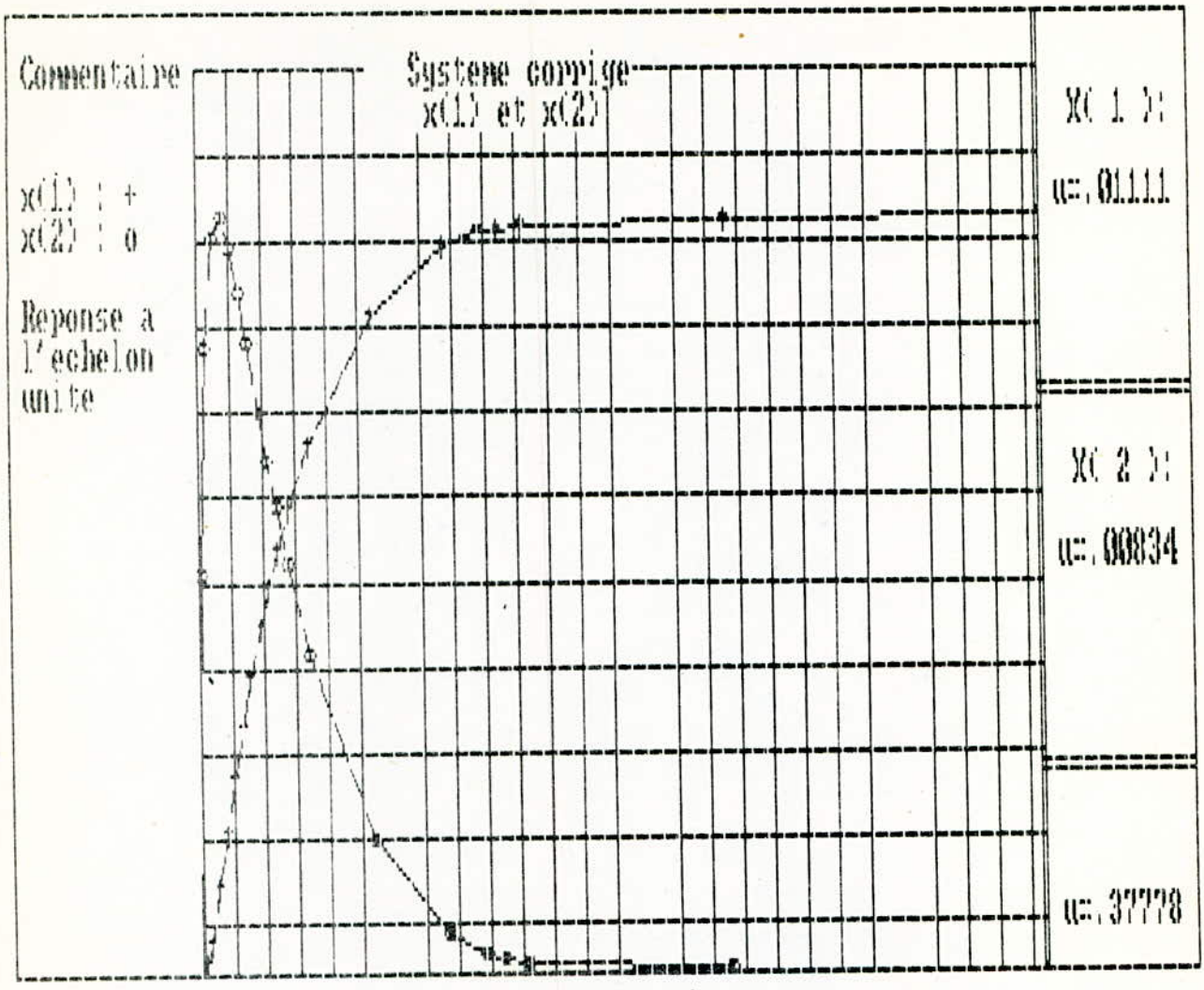
CHAPITRE IV

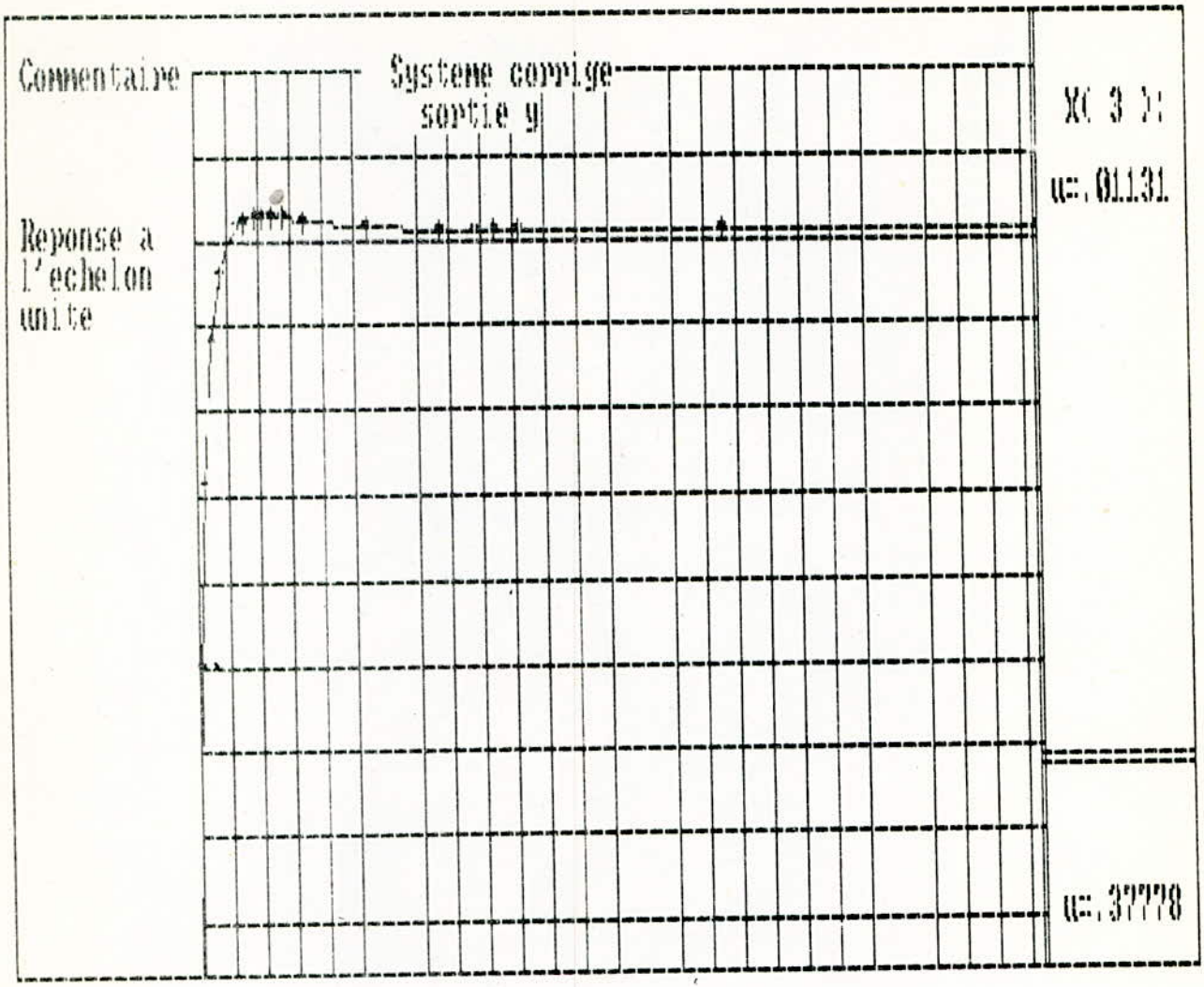
EXEMPLES D'APPLICATION

```
Exemple 1:

L'ordre de la matrice A est:2
L'ordre de la matrice B est:1
L'ordre de la matrice C est:1
L'horizon de simulation (ti,tf):0,10
Introduire les donnees dans l'ordre suivant:A,B,C,Xi,Yi

la matrice A:

0   1
-9      1

la matrice B:

0   1

la matrice C:

1   1

les conditions initiales:

0      0

0

0



Analyse:

Les valeurs propres en boucle ouverte sont:

   -5+j*1.322875A18934A31
   -5+j*-1.322875A18934631

Le systeme est stable
Le systeme est commandable
Le systeme est observable



Synthese:

La matrice gain est:

    -8.198039810119629     -9.834946632385254

La matrice A en boucle fermée est:

    0    1
   -10.198039810119629    -10.834946632385254

Les valeurs propres en boucle fermée sont:

   -1.0412707983213125+j*0
   -9.793656349182129+j*0

Le cout optimal est:
  0

Nombre d'iterations= 8
La convergence= 0
La matrice K:0.1
```


Commentaire
x(1):+
x(2):o
Reponse
a l'ech-
elon
unite
Systeme non corrige
x(1) et x(2)
X(1):
u=.07064
X(2):
u=.05887
u=.37778


Commentaire
Systeme non corrige
sortie y
Reponse
a l'ech-
elon
unite
X(3):
a=.09085
a=.37778


Commentaire
x(1) : +
x(2) : o
Reponse a l'echelon unite
Systeme corrige
x(1) et x(2)
X(1):
u=.01111
X(2):
u=.00834
u=.37778


Commentaire
Reponse a l'echelon unite
Systeme corrige sortie y
X(3):
u=.01131
u=.37778

Exemple 2: [9]

```
L'ordre de la matrice A est:5
L'ordre de la matrice B est:1
L'ordre de la matrice C est:1
L'horizon de simulation (ti,tf):0,2
Introduire les donnees dans l'ordre suivant:A,B,C,Xi,Yi

la matrice A:

-160        0.087      0.71       -2025       -20250
71.5        -0.087     0          0           0
88.5        0          -0.71      0           0
0.087       0          0          -0.186      0.186
0           0          0          0.385       -12.085

la matrice B:

675
0
0
0
0

la matrice C:

1
0
0
0
0

les conditions initiales:

-270   -221897.7   -33254.9   -52.7    1.88

0
```

```
Analyse:

Les valeurs propres en boucle ouverte sont:

  -159.9767303466797+j 0
  -11.02512893676758+j 0
  -.5425007939338684+j .4268013536930084
  -.5425007939338684+j-.4268013536930084
  -8.113475143909454E-002+j 0

Le systeme est stable
Le systeme est commandable
Le systeme est observable
```

Synthese:

La matrice gain est:

```
  -2.934112071990967   -1.014381487038918E-0
0a   -9.459680295549333E-004    2.45758008956
9092    27.57451438903809
```

La matrice A en boucle fermée est:

```
  -2140.525634765625   1.578601449728012E-002
   7.194632291793823E-002  -366.05078125  -16
37.189453125
   71.5   -8.699999749660492E-002   0   0   0
   88.5    0   -.7099999785423279   0   0
   3.700000047683716E-002   0   0  -.18600000
44107437    .1860000044107437
   0   0   0   .3849999904632568  -12.0850000
3814697
```

Les valeurs propres en boucle fermée sont:

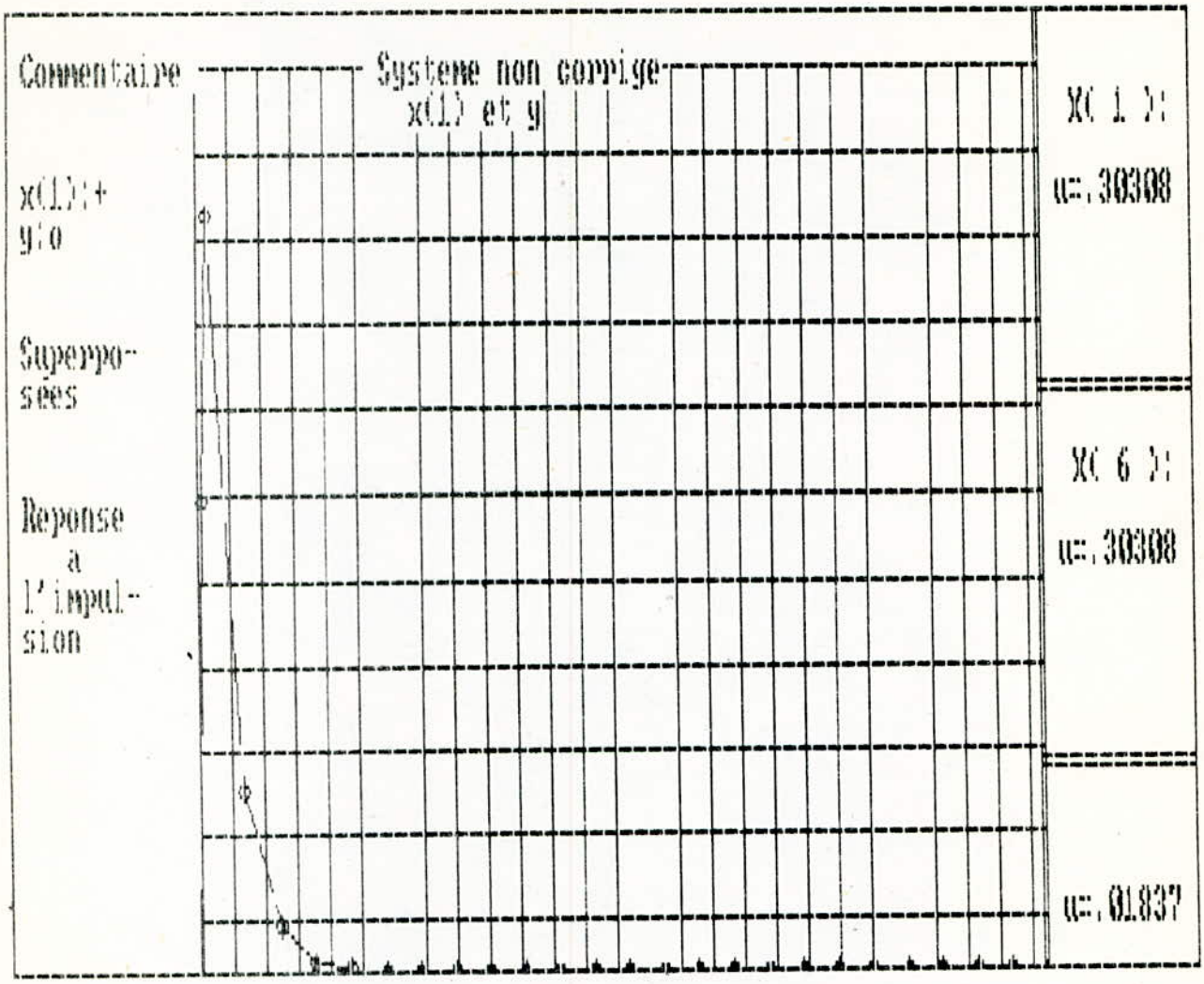
```
  -2140.522216796875+j 0
  -12.09009838104248+j 0
  -.1872281432151794+j 0
  -.7069852948188782+j 0
  -8.65083634853363E-002+j 0
```

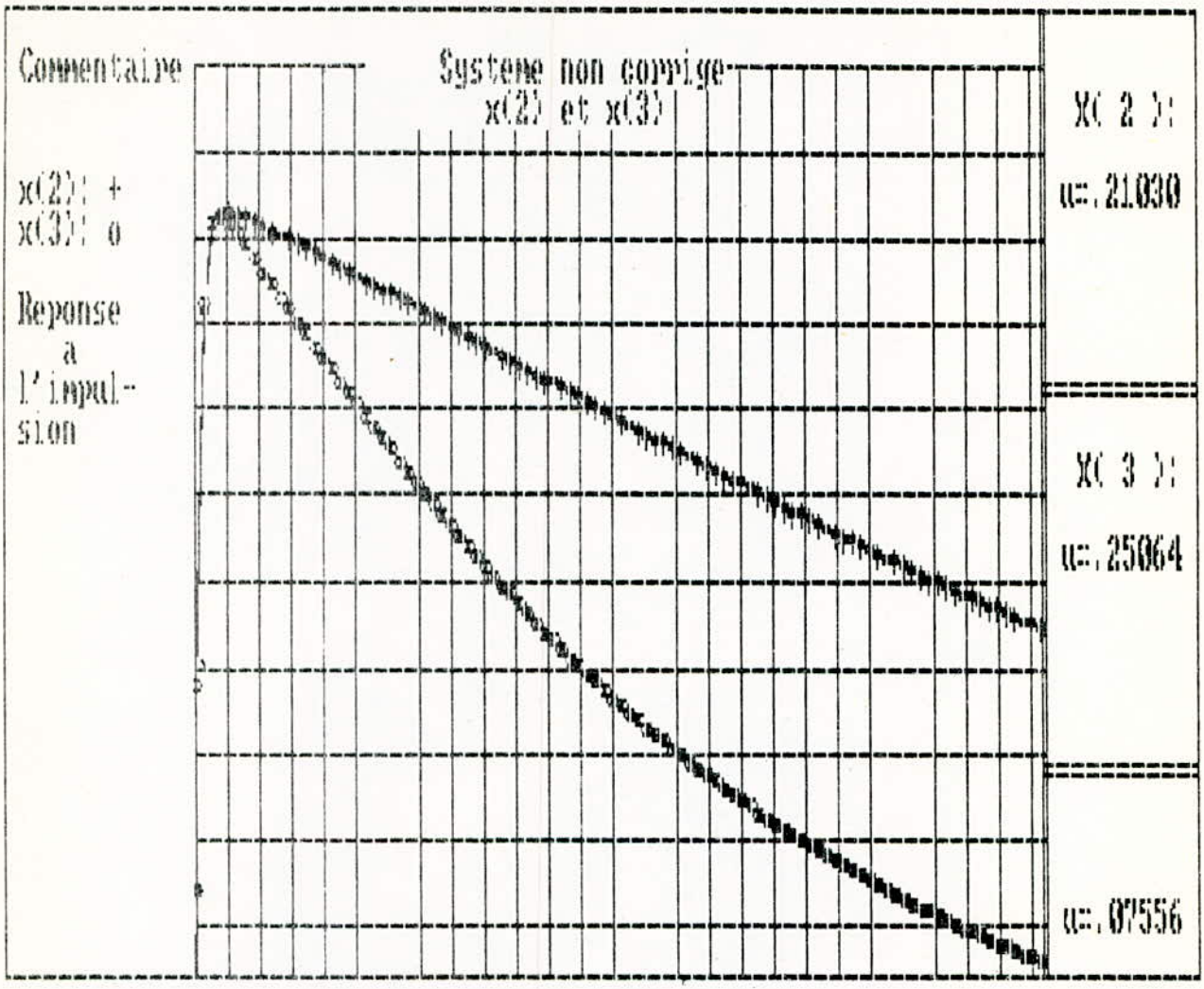
Le cout optimal est:
 0

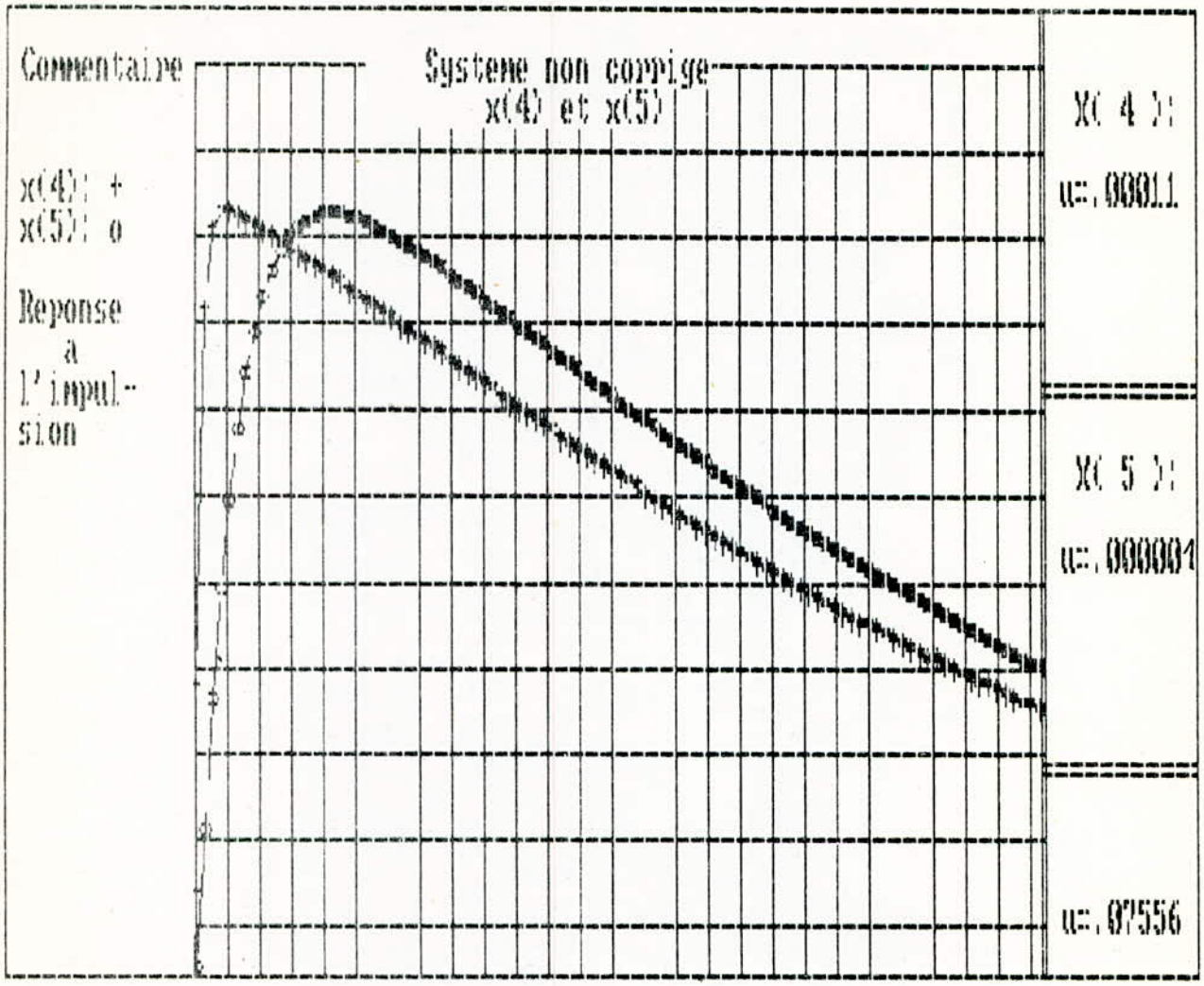
Nombre d'iterations= 20

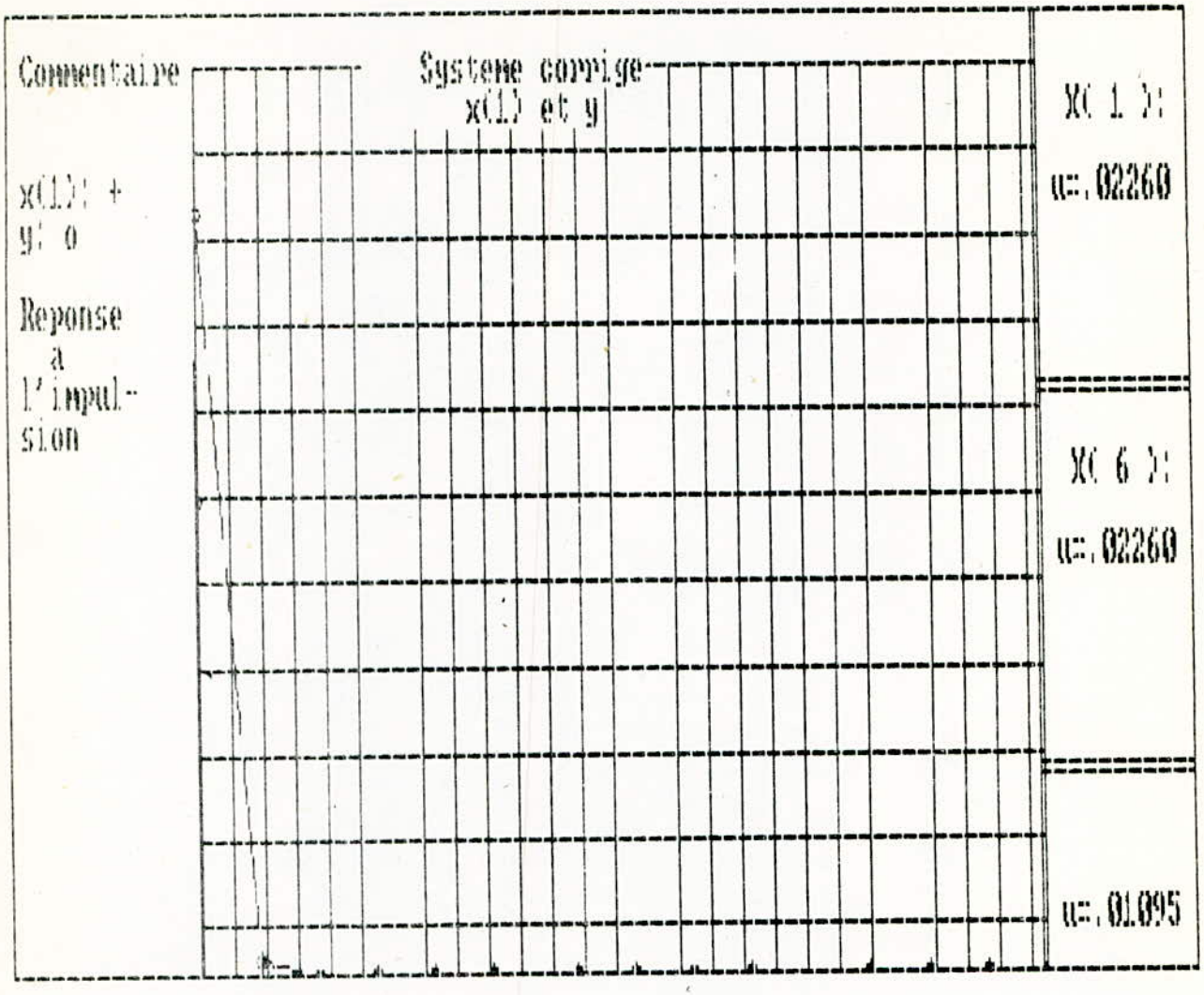
La convergence= 8.768204133957624E-005

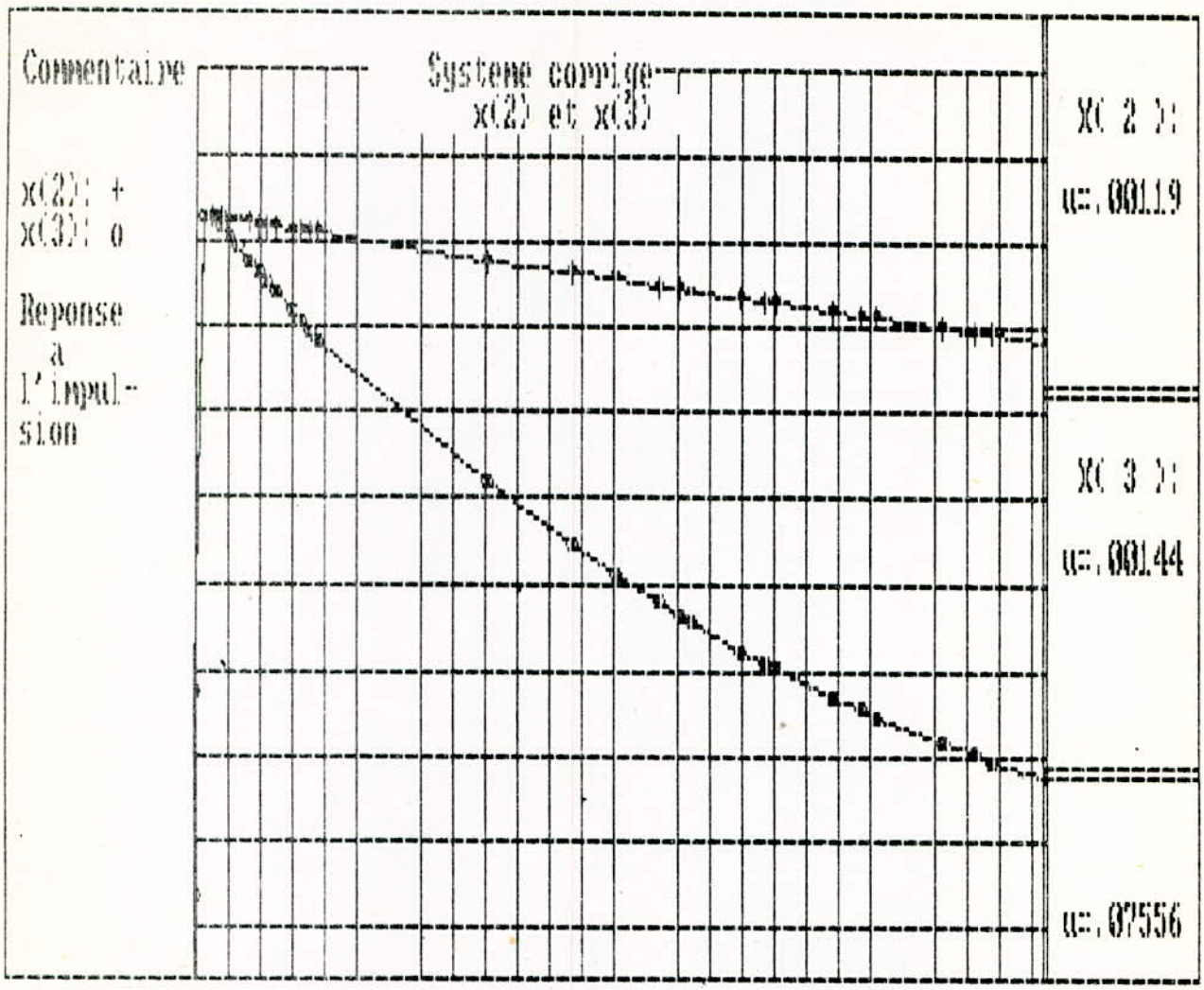
La matrice R:0.1

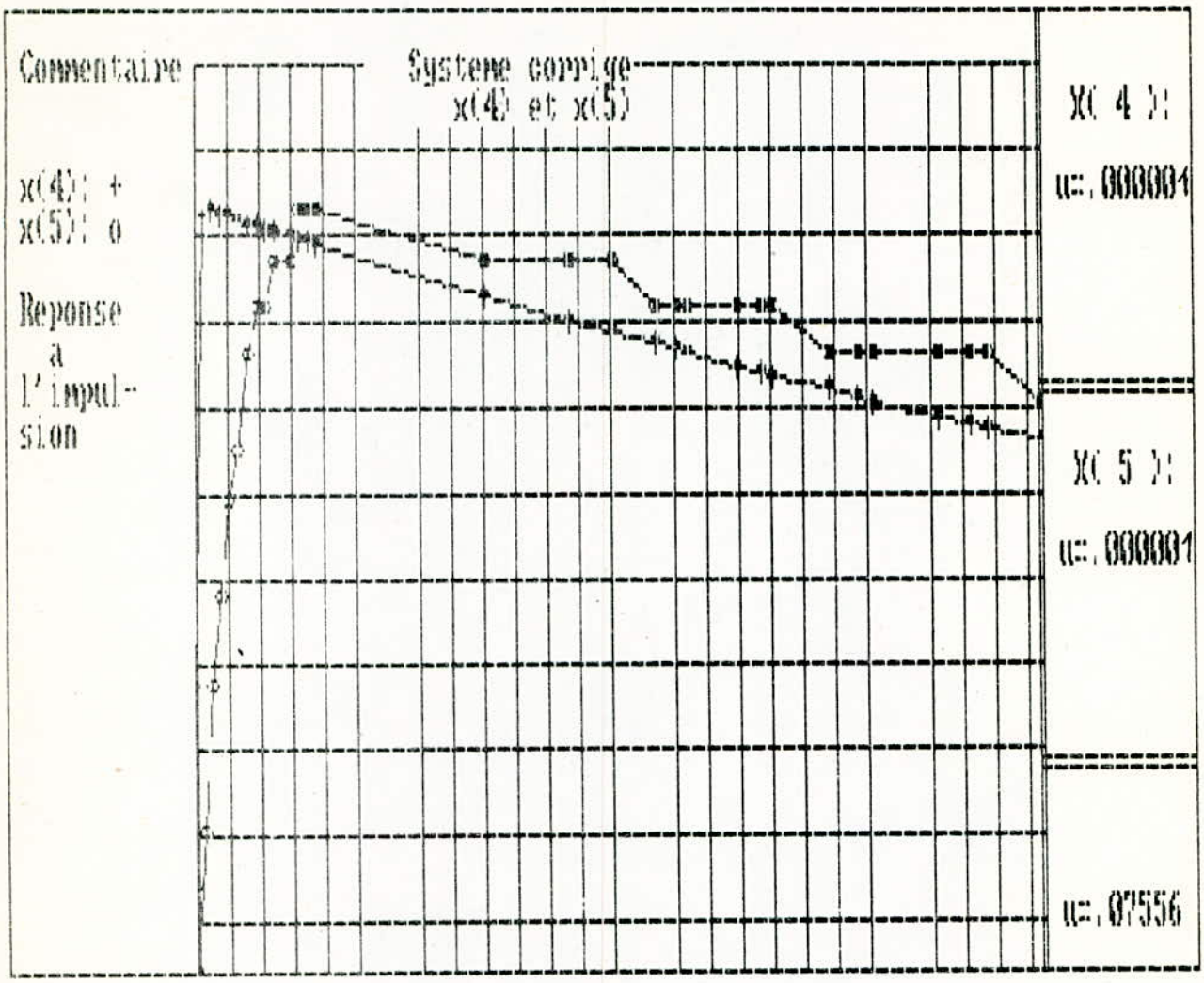

Commentaire
Systeme non corrige
x(1) et y
x(1)+
y:0
Superpo-
sees
Reponse
a
l'impul-
sion
X(1):
a=.30308
X(6):
a=.30308
a=.01837


Commentaire
x(2): +
x(3): o
Reponse
a
l'impul-
sion
Systeme non corrige
x(2) et x(3)
X(2):
a=.21030
X(3):
a=.25064
a=.07556


Commentaire
x(4): +
x(5): o
Reponse
à
l'impul-
sion
Systeme non corrige
x(4) et x(5)
X(4):
u=.00011
X(5):
u=.000001
u=.07556


Commentaire
x(1): +
y: 0
Reponse
a
l'impul-
sion
Systeme corrige
x(1) et y
X(1):
u=.02260
X(6):
u=.02260
u=.01095


Commentaire
x(2): +
x(3): o
Reponse
a
l'impul-
sion
Systeme corrige
x(2) et x(3)
X(2):
u=.001.19
X(3):
u=.001.44
u=.07556


Commentaire
x(4) : +
x(5) : o
Reponse
a
l' impul-
sion
Systeme corrige
x(4) et x(5)
X(4):
u=.000001
X(5):
u=.000001
u=.07556

Exemple :

L'ordre de la matrice A est:2
L'ordre de la matrice B est:1
L'ordre de la matrice C est:1
L'horizon de simulation (ti,tf):0,10
Introduire les donnees dans l'ordre suivant:A,B,C,Xi,Yi

la matrice A:

0 1
-20 -1

la matrice B:

1
0

la matrice C:

1 .
0

les conditions initiales:

0 0

0

Analyse:

Les valeurs propres en boucle ouverte sont:

-.5+i 4.44097704208374
-.5-i 4.44097704208374

Le systeme est stable
Le systeme est commandable
Le systeme est observable

Synthese:

La matrice gain est:

-31.1843624149902 -.68838930130000488

La matrice A en boucle fermée est:

-31.1843760040893555 .3116101026535034
-20 -1

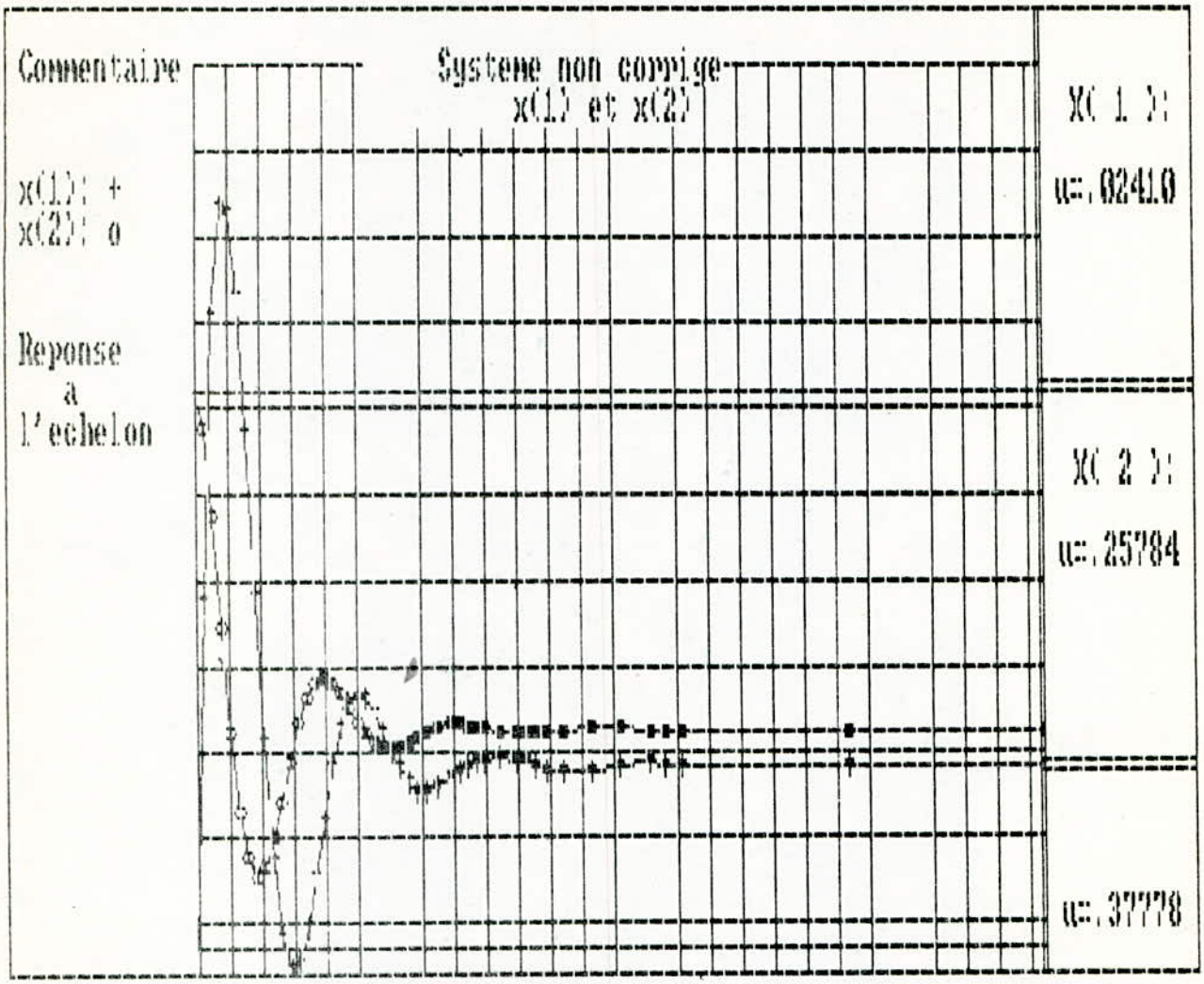
Les valeurs propres en boucle fermée sont:

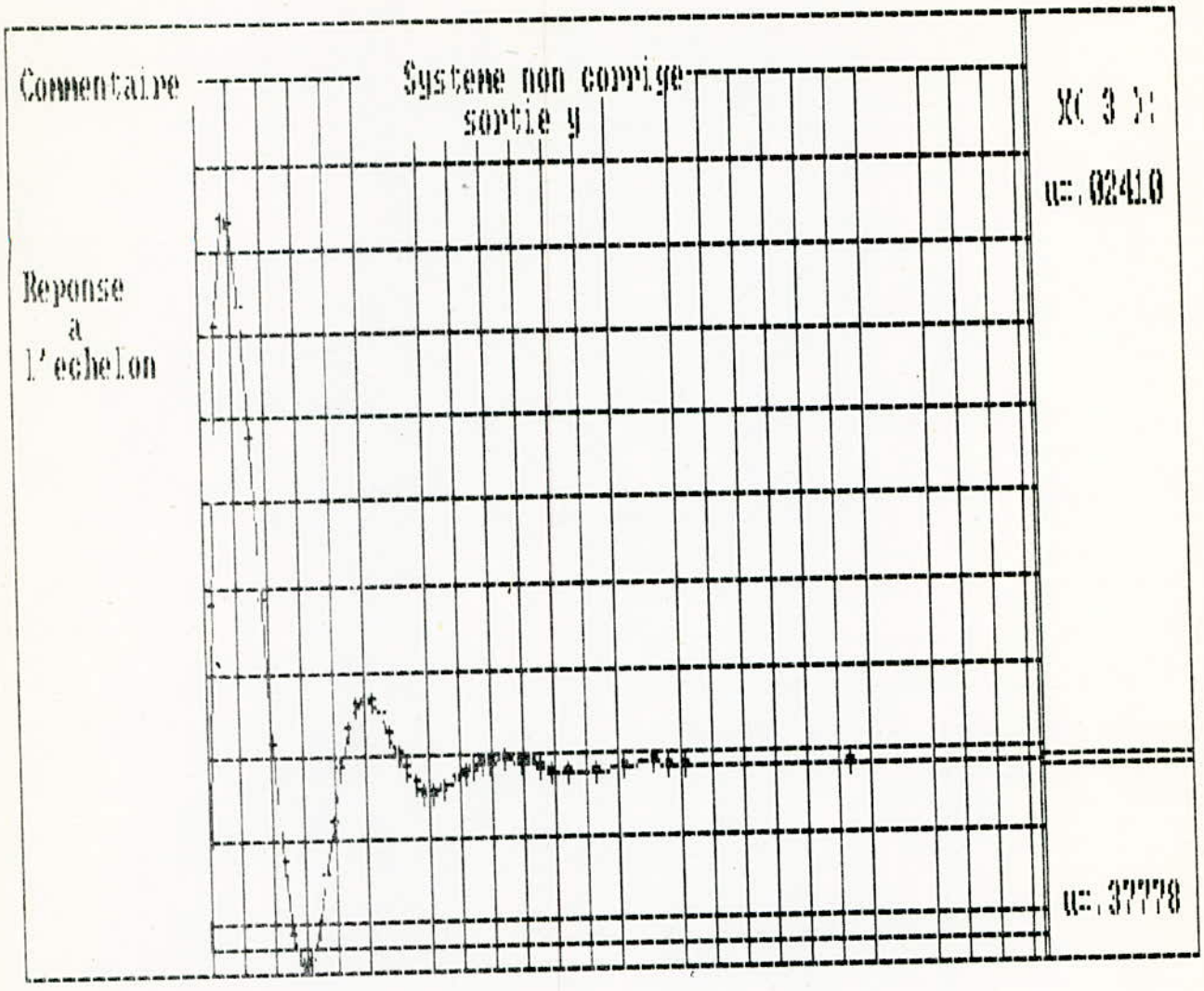
-30.9776677132568836+j 0
-1.2079031467437744+j 0

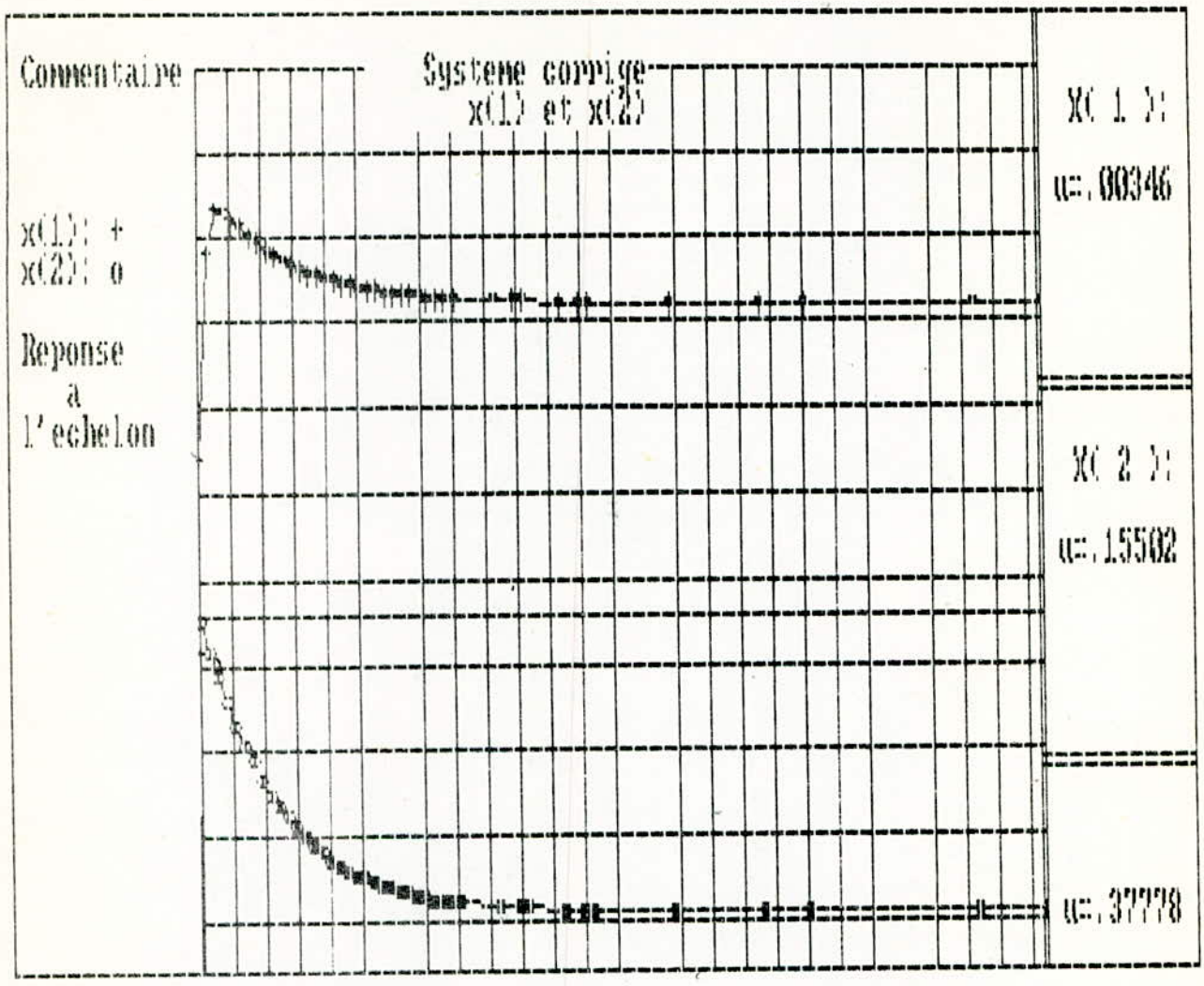
Le cout optimal est:
0

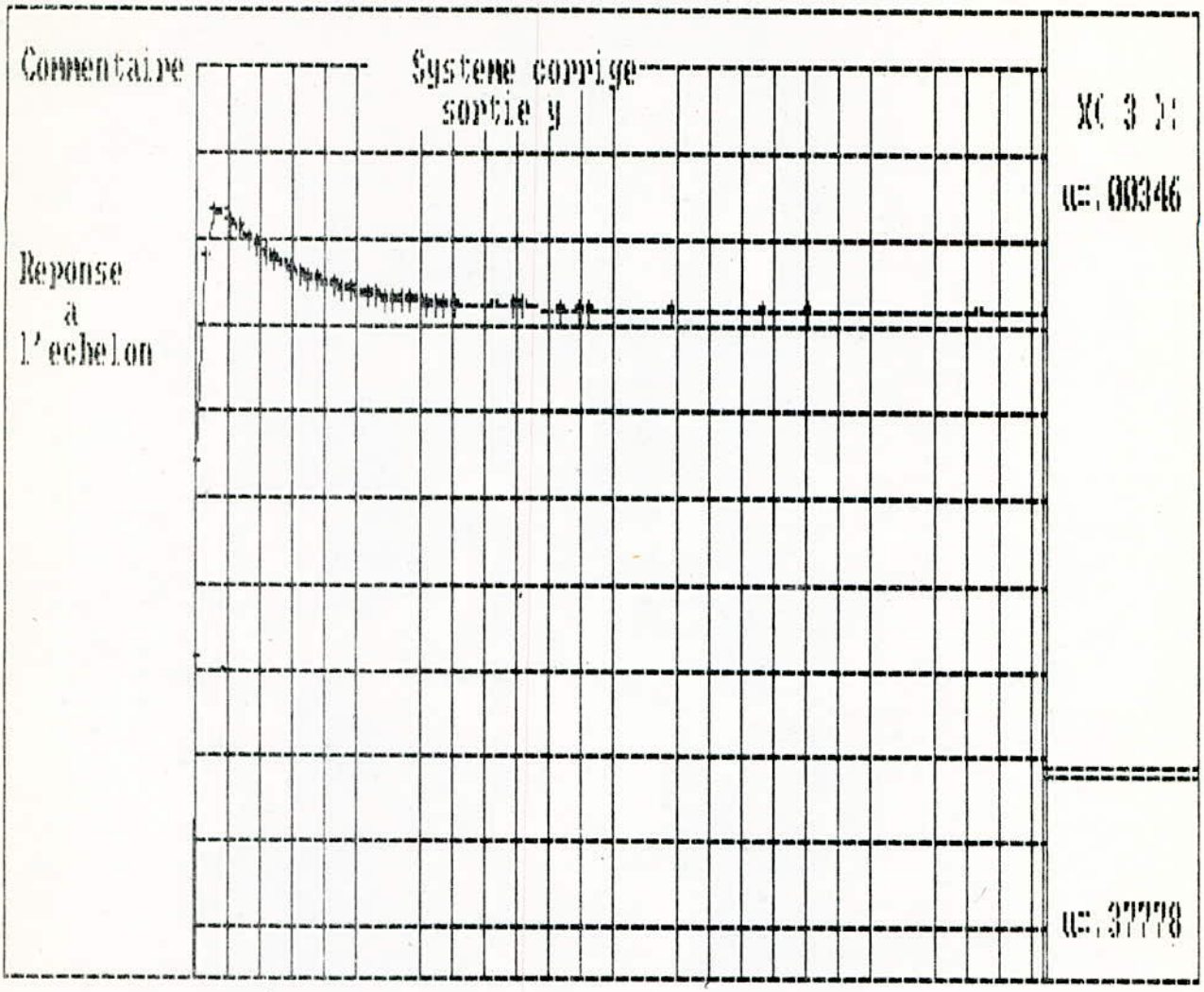
Nombre d'iterations= 8
La convergence= 8.032657206058502E-009

La matrice K:0.001


Commentaire
x(1): +
x(2): o
Reponse
a
l'echelon
Systeme non corrige
x(1) et x(2)
X(1):
u=.8241.0
X(2):
u=.25784
u=.37778


Commentaire
Reponse
à
l'echelon
Systeme non corrige
sortie y
X(3):
u=.0241.0
u=.37778


Commentaire
Systeme corrige
x(1) et x(2)
x(1): +
x(2): o
Reponse
a
l'echelon
X(1):
u=.00346
X(2):
u=.15502
u=.37778


Commentaire
Reponse
à
l'echelon
Systeme corrige
sortie y
XC 3 1:
c=.00346
c=.37778

CONCLUSION

Le travail realisé est un exemple parmi d'autres pour L'automatique.

Tout au long de ce travail, nous avons cherché à utiliser les méthodes et les algorithmes les plus performants afin d'obtenir d'assez bons résultats en un temps minime.

Nous avons pu avoir une idée sur les systèmes de CAO, leur mise en oeuvre et les nombreuses possibilités de conception dans le domaine automatique. Ceci nous a été bénéfique parce que nous avons pu réalisé un système (même s'il n'a pas l'envergure de ceux cités au premier chapitre) qui assiste l'utilisateur dans plusieurs tâches.

Le comportement du modèle mathématique ,ou du système physique a été cerné par une étude automatique comprenant l'analyse, la simulation et la synthèse.

Ce dernier module calcule les lois de commande de ce modèle ou de ce système à l'aide du critère quadratique; critère assez performant.

Les résultats des exemples traités donnent une idée sur la rapidité des algorithmes utilisés et leur efficacité.

En outre ,ce logiciel permet,grace à son menu:
-la simplicité d'introduction du problème posé.
-la rapidité d'obtention des résultats correspondants.
-la représentation qualitative des résultats par des courbes.

Le problème que nous avons eu ,était dans le langage du TURBO-BASIC.

Celui-ci n'est pas trés puissant et est limité dans son graphisme.

Nous pouvons envisager d'améliorer la structure du logiciel et le développer en introduisant ,par exemple, la commande non-linéaire, nous pouvons prévoir une décomposition du système en deux sous systèmes: l'un lent , l'autre rapide.

LÀ, découleront d'autres calculs tels que le calcul du Jacobien. Il y'aura donc linéarisation des équations non- lineaires.

REFERENCES

[1] M GACI
La CAO en automatique

[2] CADCE' 85
Computer Aided Design in control and Engineering systems
3 rd IFAC/ IFIP 1985

[3] Manuel Turbo Basic
Conçu et commercialisé par Borland International INCO
ed P. S. I

[4] A. FOSSARD
Commande dessystèmes multidimentionnels
ed Dunod 1972

[5] Y. MONCEF
Un algorithme général de résolution des équations différencielles
CEA - N - 1719 Mai 1972

[6] E. J. DAVISON
Dept. of Electrical Engineering, University of Califonia, Berkley, California U.S.A

[7] A. BOUNEMRI
Approximation des systèmes de grande dimension Analyse et commande
Thèse de magister C. E. N 1986

[8] A. BOUHAROUF . B. BENDJAIMA
Application de la commande robuste au modèle nucléaire
Projet de fin d'étude Juin 1985

[9] DJENNOUN
Contribution à la mise au point d'algorithmes d'analyse

et de commande: "Application aux réacteurs nucléaires".
Thèse de magister C. E. N. S. Alger Juin 1986