UN EXEMPLE DE SYSTEME SYMBOLIQUE NUMERIQUE: PANDORE-BASILE

F.Delebecque, J.P. Quadrat

January 12, 1990

1 PANDORE

Pandore est un système expert pour l'identification et la commande optimale stochastique ou déterministe.

Il est spécialisé dans la résolution des systèmes non-linéaires et l'exploitation du calcul formel pour la résolution numérique de ces problèmes.
Un interface avec Basile lui permet de générer des programmes numériques exploitables par ce dernier lorsque des fonctionalités lui manquent.

Il est souvent capable de réaliser une étude complète à partir de la spécification du problème de commande. Cette étude consiste à générer des programmes numériques, à faire du graphique sur les résultats et à générer un rapport en Latex résumant et expliquant les résultats obtenus.

Il a accès à la bibliothèque Basile par ses moyens de génération de programmes Fortran.

Il est capable de discrétiser avec un ordre de précision arbitraire, donné par l'utilisateur, des équations différentielles commandées et de résoudre le problème associé par une méthode de gradient.

Pour atteindre ces objectifs Pandore vit dans un univers Lisp auquel on a ajouté des possibilités d'inférence par un Prolog *Oblogis* — fait par P. Gloess à l'Université de Compiègne — et de calcul formel par *Macsyma.*

La structure de Pandore est donnée par la figure 1.

La partie numérique est soit effectuée par la génération de programmes Fortran et leur exploitation automatique par Pandore, soit par la génération de programmes Fortran utilisables par Basile grâce à des macros Basile classiques. La raison de ce choix est justifié par l'expérience. On se trouve généralement en présence de l'alternative suivante :

- soit le problème est facile et Pandore arrive alors à faire l'étude tout seul,

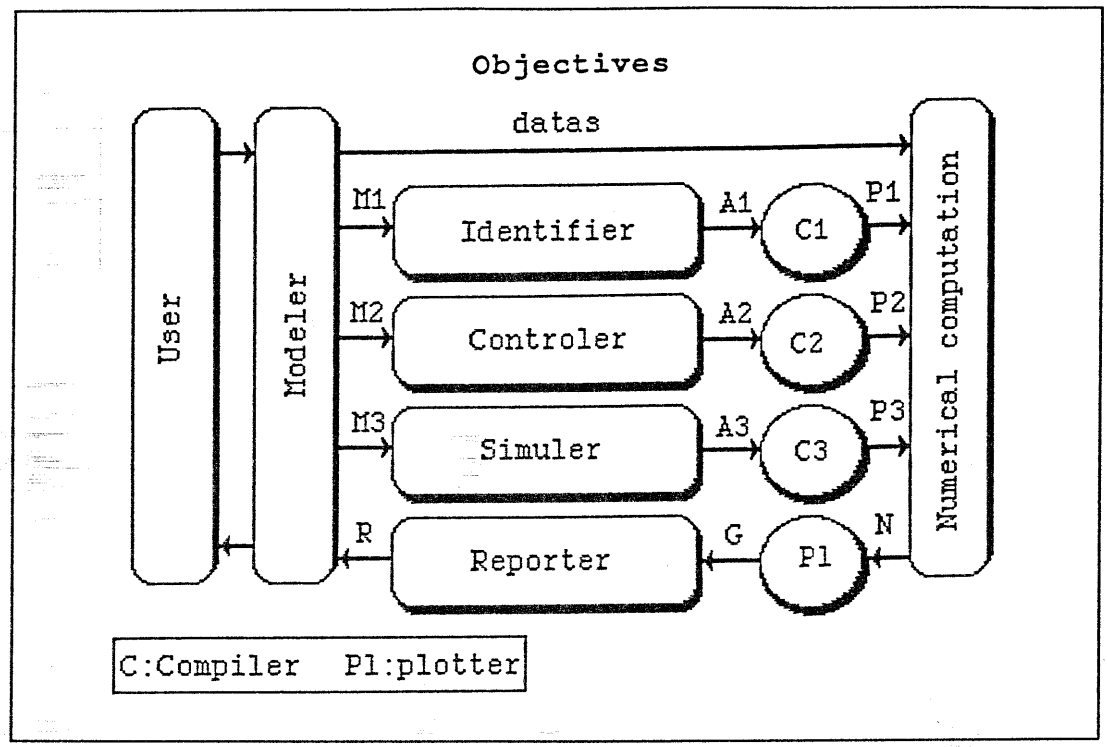


Figure 1: Structure de Pandore.

- soit le problème est trop difficile pour Pandore, alors de la programmation ou tout au moins de l'intéraction dans Basile et Macsyma est nécessaire. Pandore se contente de fournir des programmes Fortran nécessaires à Basile (qui sont dans ces cas difficiles, souvent infaisables à la main). Pandore génère alors également le rapport.

Pandore comprend quatre parties :

- un éditeur spécialisé d'entrée du problème,
- un langage de commande permettant d'interagir avec le système,
- un système de génération de programmes Fortran,
- un système de génération de rapports pouvant démontrer des théorèmes dans certains cas particuliers.

2 L'EDITEUR D'ENTREE DU PROBLEME

C'est un éditeur fenêtre-souris-menus permettant de spécifier le problème d'optimisation et les notations.

Cet éditeur est intelligent dans la mesure où il adapte automatiquement ses menus aux réponses précédemment faites.

Son rôle est d'installer des faits Prolog exploités par la suite par le langage de commande. Prenons l'exemple de la rentrée de la navette dans l'atmosphère qui sera l'application dont nous

nous servirons pour illustrer notre propos. Il est énoncé clairement en termes physiques au paragraphe 1.5 du chapitre précédent ou en termes mathématiques dans le rapport généré par Pandore en annexe B.

L'entrée des faits se fait par le menu donné dans la figure 2.

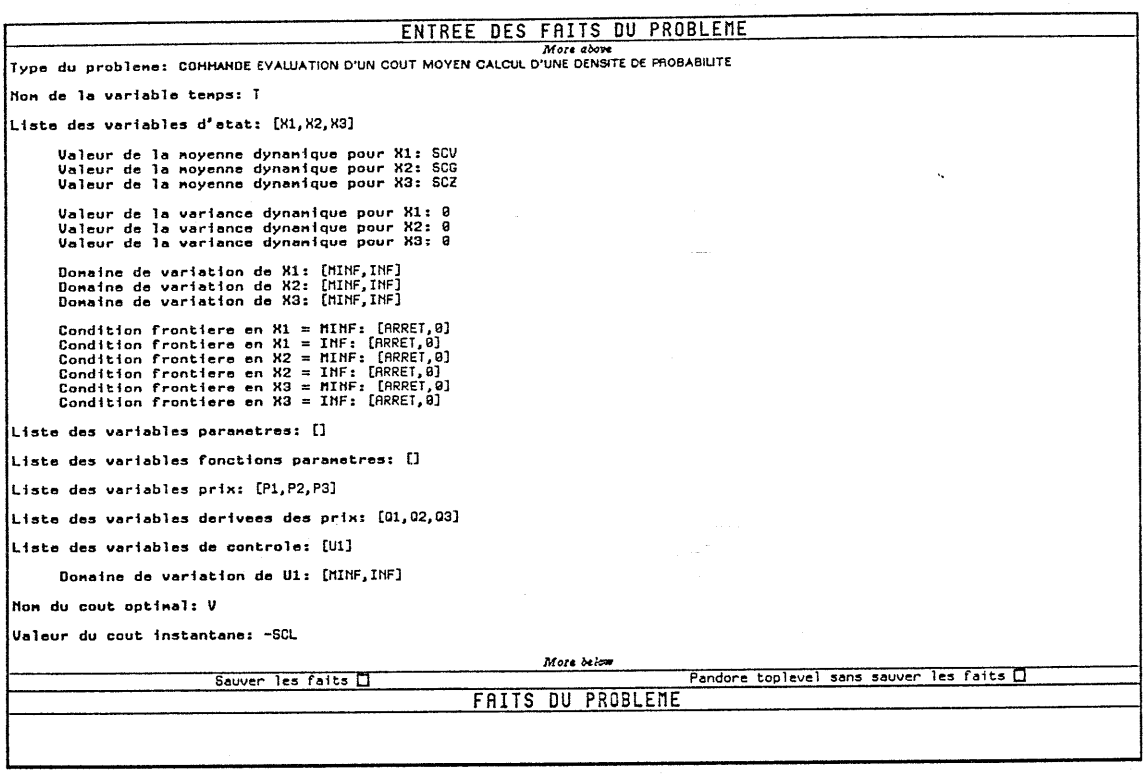


Figure 2: Editeur d'entrée.

Lorsque le problème est compliqué, un programme Macsyma peut être nécessaire.
Sur des exemples plus simples, on pourra entrer directement les formules à l'éditeur.
Une fois sauvés les faits par le menu correspondant de l'éditeur on dispose d'une base de données accessible par Prolog et mise à jour par les différentes commandes de Pandore et par les réponses de l'utilisateur à des questions.

3 LE LANGAGE DE COMMANDE DE PANDORE

Ce sont des commandes rajoutées au système de la machine Lisp, spécialisées à l'exploitation numérique et graphique de problèmes de commande optimale.

La liste complète des commandes est donnée dans la figure 3.

Leurs noms sont suffisamment explicites, nous ne les commenterons pas.

A titre d'illustration, la génération des programmes Fortran nécessaires à Basile pour résoudre le problème déjà cité par une méthode de gradient, nécessite l'appel de deux commandes :

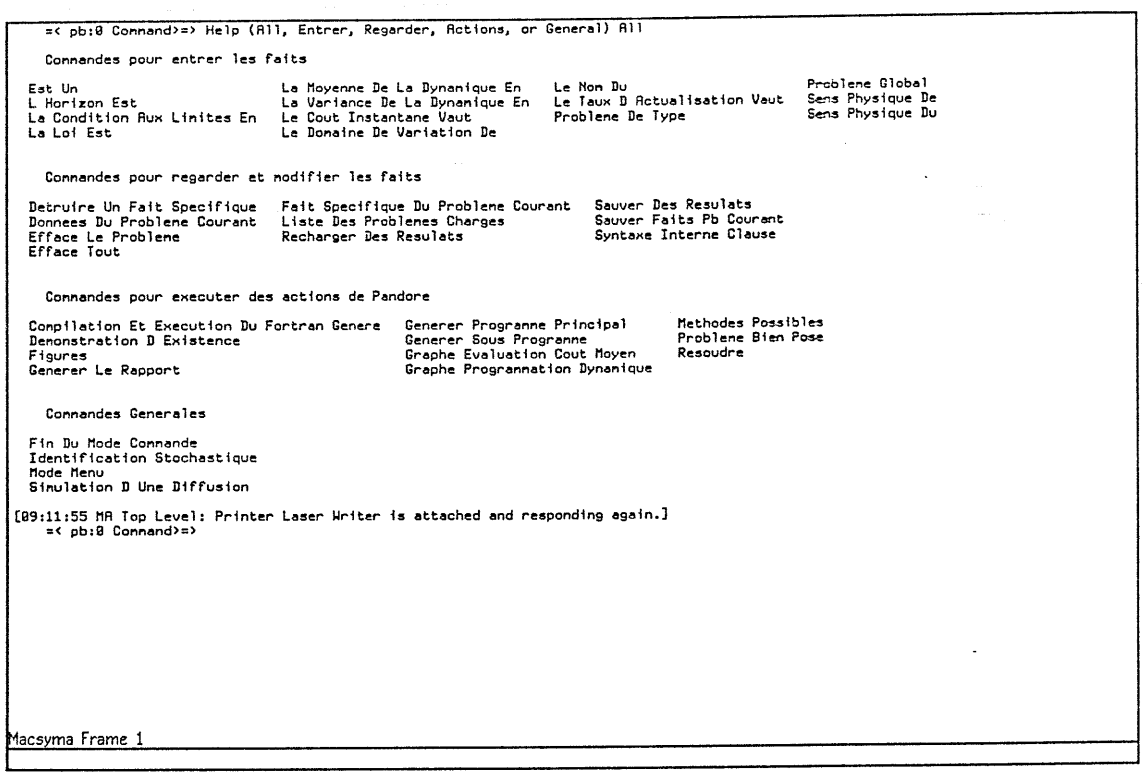


Figure 3: Commandes de Pandore.

⇒ *Méthodes possibles* . La réponse propose Pontriaguine-Basile entre autre.

⇒ *Générer sous programme Pontriaguine-Basile* génère les programmes Fortran suivants :

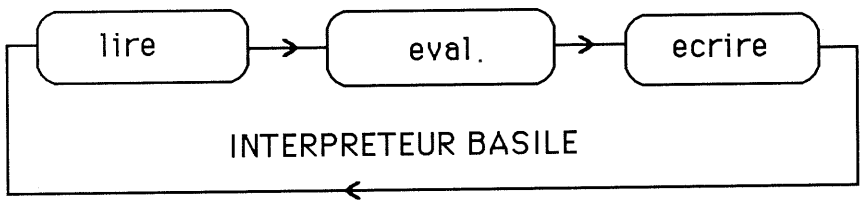
```
      SUBROUTINE CFX(X,CX)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(4),CX(4),T
        COMMON /ZPAR/ A(100)
          CX(1)=200*(X(1)-0.0976)
          CX(2)=40*(X(2)+1)
          CX(3)=200*(X(3)-0.2)
    etc...
      END
```

On peut alors utiliser BASILE pour exploiter ce programme.

4 BASILE

BASILE est un système conçu pour l'étude et la résolution des problèmes relevant de l'Automatique classique. Ce système a été réalisé avec l'objectif de fournir aux experts en Automatique un puissant outil de calcul.

Le système est totalement interactif: il reçoit une ligne de commande, la traite immédiatement, le cas échéant imprime le résultat et attend la nouvelle commande. D'autre part, le système offre



PROG. DURE

OBJETS STANDARDS

-Matrices scalaires
-Matrices polynomiales
-Matrices de ch. de carac.

BIBLIO. STANDARD

Syst. Lin.
-LINPACK
-EISPACK
Eq. Dif.
-ODEPACK
Optimisation
-MODULOPT

PROG. BASILE

NOUVEAUX OBJETS

-Syst. Lin. (A,B,C)
-Matrices de Transfert

MACROS

-Automatique lin.
-Trait. du signa
-Robustesse

un grand confort pour la visualisation des solutions obtenues, visualisation alpha-numérique ou graphique.

On a cherché en outre à rendre le système le plus "ouvert" possible; l'utilisateur peut, soit créer des procédures (ou "macros"), écrites en langage BASILE, et directement utilisables de façon interactive, soit introduire lui-même une nouvelle primitive de calcul associée à un ou plusieurs programmes FORTRAN. Les possibilités d'extension des fonctionnalités du système sont donc pratiquement illimitées. Il est clair, cependant que le caractère interactif du système est limité par le temps de calcul nécessaire à l'exécution de la commande soumise au système. On a donc cherché à réaliser un bon compromis entre les primitives de calcul et les macros.

Un effort particulier a été déployé pour donner au langage BASILE la plus grande similitude possible avec l'écriture mathématique courante. Cependant la nature des objets de l'Automatique classique (matrices réelles ou complexes, fonctions de transfert, matrices polynomiales, etc.) et l'obligation de "linéarité" de l'écriture dans un ordinateur ont obligé parfois à s'éloigner du modèle choisi.

La portabilité a été assurée en écrivant l'ensemble du système en FORTRAN 77 standard. BASILE se compose d'une partie "système" (environ 20000 lignes de FORTRAN) qui interprète la ligne de commande et gère l'interface avec une bibliothèque de programmes d'intérêt général (60000 lignes) . La taille de la bibliothèque de programmes a volontairement été réduite au minimum puisque la plupart des algorithmes sont implantés par des macros regroupées en bibliothèques . La structure modulaire de l'ensemble permet de modifier facilement la configuration standard de BASILE.

Le système a été développé à partir du logiciel MATLAB dont on a conservé en partie l'interpréteur. La base de données, en revanche, a été complètement modifiée pour traiter les objets spécifiques de l'Automatique (fonctions non linéaires ou matrices polynomiales, par exemple). La syntaxe MATLAB est devenue un standard de fait de la CAO en Automatique .

Conception générale du système Comme on l'a dit plus haut BASILE se veut un système *ouvert*, c'est à dire pouvant facilement s'adapter aux besoins particuliers d'un utilisateur. Cette volonté nous a conduit à concevoir le système comme un *langage* utilisant une base de données spécifique.

En fait la plupart des systèmes modernes de CAO en Automatique reposent sur ce principe .

Il est clair en effet que la multiplicité des systèmes dynamiques étudiés (discrets, continus, linéaires, non linéaires, implicites, explicites, déterministes, stochastiques, ...), la multiplicité de leurs possibles représentations (réponses fréquentielles, variables d'état, fractions matricielles polynomiales, matrices de transfert, ...) ainsi que la multiplicité des problèmes posés (identification, réalisation, contrôle, optimisation, simulation, robustesse, ...) rendraient rapidement inutilisable un système fonctionnant selon le mode "menus". En revanche, un utilisateur pourra bien entendu définir lui-même ses propres menus en écrivant des macros appropriées sous forme de dialogue.

Une conséquence essentielle de cette conception est que BASILE peut paraître —à première vue— assez éloigné des préoccupations immédiates de l'Automaticien. En fait les *variables* manipulées et les *primitives* de calcul sont vues comme des "briques" à partir desquelles l'utilisateur doit construire la description du système qui l'intéresse ainsi que la solution qu'il envisage au problème posé. Il sera donc toujours demandé un minimum d'expertise et de programmation à l'utilisateur. Cela peut aller de quelques lignes de programme pour concevoir par exemple un observateur-contrôleur d'un système dynamique linéaire représenté en variables d'état jusqu'à un effort moins négligeable pour optimiser un système non linéaire. Cependant il faut noter que cet effort de programmation ne se fait que lorsqu'on se trouve devant un problème "nouveau".

En effet tout problème une fois résolu peut se résumer à une *macro* (c'est à dire, un programme

BASILE) qu'on pourra sauvegarder et charger à nouveau dans le système pour résoudre un problème analogue.

On a donc volontairement essayé de limiter au maximum le nombre de primitives de calcul (une centaine) de façon à ne pas trop encombrer la mémoire de l'utilisateur. Souvent ces primitives ont une syntaxe courte ou longue selon que des paramètres optionnels sont présents ou non. De plus on peut demander un ou plusieurs arguments de sortie à ces primitives.

Ces primitives ont été choisies d'une part en raison de leur caractère d'intéret général et d'autre part en raison de leurs qualités numériques: les algorithmes implantés sont en général numériquement stables . Typiquement un utilisateur définira ses propres programmes en langage BASILE (macros). Ces macros faisant appel aux puissantes capacités numériques du système, l'utilisateur se trouve affranchi d'un pénible travail de programmation, tout en pouvant traiter des problèmes très généraux.

Si l'usage "standard" du système consiste à *écrire des macros*, c'est à dire à écrire des programmes en langage BASILE enchaînant clauses, boucles et appels aux primitives de calcul, il est aussi possible à l'utilisateur de rajouter lui même au système une nouvelle primitive correspondant à un (ou plusieurs) sous-programmes FORTRAN.

5 BASILE-PANDORE

Pour résoudre un problème de contrôle optimal non linéaire dans Basile, les difficultés éssentielles proviennent d'une part de la necessité de faire des calculs symboliques (calcul de dérivées essentiellement) et d'autre part de la lenteur de l'interpréteur Basile qui rend les calculs parfois trop longs pour que le système garde son caractère intéractif . L'utilisation de Macsyma permet de résoudre ces deux difficultés grâce à Macrofort qui va nous donner un programme FORTRAN dans lequel sont intégrés tous les calculs symboliques dont on a besoin. Ces programmes FORTRAN sont compilés et appelés interactivement depuis Basile. Le code Basile de l'application se résume à quelques macros, les calculs de type scalaire étant faits en FORTRAN. Du point de vue d'un utilisateur de Basile la situation se résume à trois fichiers :

Un premier fichier "interface" contient les interfaces "soft" entre Basile et les programmes FORTRAN générés par Macrofort. On trouvera dans la partie consacrée à Pandore les exemples de tels programmes pour le test sur la navette qui a servi d'illustration à cette étude.

Un deuxième fichier "méthode" contient les macros Basile qui traitent le problème de contrôle selon une méthode donnée (par exemple gradient ou shooting). Ces macros seront décrites dans les paragraphes qui suivent et sont d'intérêt général, c'est-à-dire indépendantes des données de l'application particulière. Pour traiter un deuxième problème de même type on n'aura donc pas à les modifier. De même pour des problèmes de "petite" taille (ou des problèmes linéaires ou encore pour tester l'algorithme) on pourra écrire les macros spécifiques de l'application et décrites plus haut en langage Basile.

Un troisième fichier "exécution" initialise les paramètres et lance la suite de commandes Basile permettant de résoudre le problème. Evidemment cette partie ne peut se faire que de manière interactive : on suit les valeurs du coût pour une méthode de gradient, on fait tracer des résultats intermédiaires, on modifie les conditions initiales, on change un paramètre de pénalisation, on regarde les valeurs propres de la matrice jacobienne etc, etc... Ce troisième fichier peut charger automatiquement les deux fichiers précédents et donc une fois sous Basile l'utilisateur qui veut lancer un problème complet doit seulement lancer la commande `exec('hermes.ex')`, pour le problème de la navette par exemple. La résolution effective d'un problème se fait en appelant et en modifiant de facon convenable le contenu de ce dernier fichier.

5.1 Fichier interface

Pour le test sur la navette le fichier d'interface est le suivant :

```
//<pT>=dercout(xT)
//Derivee du cout
pT=fort('dercou',xT,cible,pen)
//end

//<ct>=coutf(xT)
//Calcul du cout final
ct=fort('coutf',xT,cible,pen)
//end

//<xdot>=mprimal(t,xt)
ut=u(:,mini(<t/pas+1,nt>))
xdot=fort('primal',t,xt,ut)
//end

//<jaco>=jpr(t,x)
ut=u(:,mini(<t/pas+1,nt>))
jaco=fort('jprima',t,x,ut)
//end

//<xpgdot>=mxpgdual(t,xpg)
ut=U(:,maxi(<t/pas+1,1>))
xpgdot=fort('dualgr',t,xpg,ut)
//end

//<j>=jxpd3(t,xpg)
ut=U(:,maxi(<t/pas+1,1>))
j=fort('jdualg',t,xpg,ut);
//end
```

5.2 Fichier Méthode

Voici le fichier :

```
//<X>=primal(T,xTmin,U)
X=ode(xTmin,Tmin,T,mprimal,jpr);
//end

//<G>=dualg(T,xTmax,pTmax,U)
nT=maxi(size(T));reverse=nT:-1:1;
Td=T(reverse);
xpgTmax=<xTmax;pTmax;0*U(:,1)>;
XPG=ode(xpgTmax,Tmax,Td,mxpgdual,jxpd3);
G=XPG(nx3,reverse)
g1=G(:,1:nt-1);g2=g(:,2:nt);G=<g2-g1,-G(:,nt)>;
//end
```

```
//<G,cout>=grad(T,U)
nt=maxi(size(T));
write(%io(2),'integration du primal')
X=primal(T,xTmin,U),xTmax=x(:,nt);
write(%io(2),'calcul du cout')
cout=coutf(xTmax)
write(%io(2),'calcul de pTmax')
pTmax=dercout(xTmax);
write(%io(2),'integration du dual')
G=dualg(T,xTmax,pTmax,U)
//end
```

5.3 Fichier d'exécution

Le fichier qui suit permet de lancer l'application sous Basile.

```
// Chargement des fichiers
getf('gradient.bas')
getf('hermes.pb')
//compilation des macros
comp(mxpgdual)
comp(mprimal)
comp(jpr)
comp(jxpd3)
comp(grad)
comp(primal)
comp(dualg)
comp(dercout)

//Initialisations
pas=0.005;          //pas de temps
xTmin=<1;-0.2;1;0>;          //etat initial
nu= 1;          //nombre de variables de controle
//Parametres lies a la taille de l'etat
nx=maxi(size(xTmin));nx1=1:nx;nx2=nx+1:2*nx;nx3=(2*nx+1):(2*nx+nu);
TT=0:pas:1;nt=maxi(size(tt));          //discretisation en temps
Tmin=TT(1);Tmax=TT(nt);          //Temps initial et temps final
u0=1*ones(nu,nt);          //controle initial

cible=<0.0976;-1;0.2>;          //cible a atteindre
pen=<10000;10000;10000>;          //coefficients de penalisation
nappels =10 ;          //On va essayer 10 iterations d'optimisation
<c,U,G>=hopti(TT,U0,nappels);
save('resultats')          //sauvegarde des resultats
plot(T2,U2(1,:))          //graphe du controle apres 10 iterations.
```

6 BASILE-MACSYMA

Les deux interprètes Macsyma et Basile peuvent s'"appeler" l'un l'autre par l'intermédiaire de chaînes de caractères qui sont envoyées par un interprète et interprêtées et exécutées par l'autre. De plus des vecteurs de données peuvent être transmis (tableaux unidimensionnels pour l'instant).

Il est clair qu'il suffit en fait de savoir transmettre des chaînes de caractères pour faire communiquer complêtement Macsyma et Basile, une simple chaîne de caractères pouvant représenter l'exécution d'une suite de commandes arbitrairement grande.

Dans Basile on a rajouté la primitive **macsyma** qui transmet son argument (une chaîne de caractères) à Macsyma. Le résulat de l'évaluation de Macsyma est retransmis à Basile toujours sous la forme d'une chaîne de caractères qui peut alors être évaluée sous Basile. La commande **macsyma('int')** permet d'entrer sous Macsyma de façon interactive : on reçoit alors sous Macsyma un prompt particulier et on repasse sous Basile par la "commande" **basile;**. Tout se passe dans ce Macsyma interactif comme il a été précédemment expliqué (cf **macsyma-interact**) c'est-à-dire que Macsyma "connait" Basile, et deux fonctions Basile **matz** et **bas** sont interfacées qui permettent respectivement de transmettre ou lire un tableau dans la pile de Basile et de faire exécuter une chaîne de caractères par l'interprète Basile.

Pour échanger un tableau entre les deux interprètes on utilise la primitive **macsyma** qui fait exécuter par Macsyma la fonction **matz** qui suivant la valeur d'un "switch" transmet un vecteur dans un sens ou dans l'autre.

Un exemple :

La primitive **macsyma** permet de définir des macros de calcul formel dans Basile exécutées de façon transparente par Macsyma.

Un exemple de session :

```
// obtention dans y de la derivee formelle
// de sin(x)+X^3 +1/x par rapport a x
<>y=deriv('sin(x)+x^3+1/x',1)
y =

 COS(X)+3*X**2-1/X**2

// creation d'une macro
<>y=deriv1('sin(x)+x^3+1/x',1);

<> disp(y)

<h>=y(x)
        h=COS(X)+3*X**2-1/X**2
// utilisation de la macro pour evaluer
// la derivee de sin(x)+x^3+1/x en x =1
<>y(1)
ans =

     2.5403023
// creation de tableaux dans Macsyma
<>macarray('a1',20)
ans =
```

```
 done
<>arrayinfo('a1')
ans =

 [DECLARED,1,[20]]
<>arrayp('a1')
ans =

 1
<> a1=1:20;
// envoit du tableau basile a1 dans Macsyma
// partie reelle dans basile_r_a1
// partie complexe dans basile_i_a1
<> rtomac('a1')
ans =

 done
// appel de Macsyma a partir de Basile
<>macsyma("print(basile_r_a1[20])");
20

<> macarray('b1',5)
ans =

 done
<>macsyma("for i:0 thru 5 do
        (basile_r_b1[i]:float(i),basile_i_b1[i]:0.0)");
// pour rentrer dans un Macsyma interactif
<>macsyma("int")
(M-Basile) listarray(basile_r_b1);
(M-B) [0.0,1.0,2.0,3.0,4.0,5.0]
(M-Basile) basile;
   ans =
        int
<> b1=rtobas('b1');
b1 =
! 1. 2. 3. 4. 5. |
```

7 GENERATION DE PROGRAMMES

Une fois entré le problème, une suite de modules vont être appelés pour engendrer l'écriture d'un programme Fortran.

La commande *méthodes possibles* engendrera une réponse indiquant quelles sont les méthodes de résolution possibles connues de Pandore. Les points de vue connus de Pandore sont :

- la programmation dynamique,

- la méthode de découplage,
- la méthode du gradient stochastique,
- la méthode de Pontriaguine,
- la méthode de perturbations régulières.

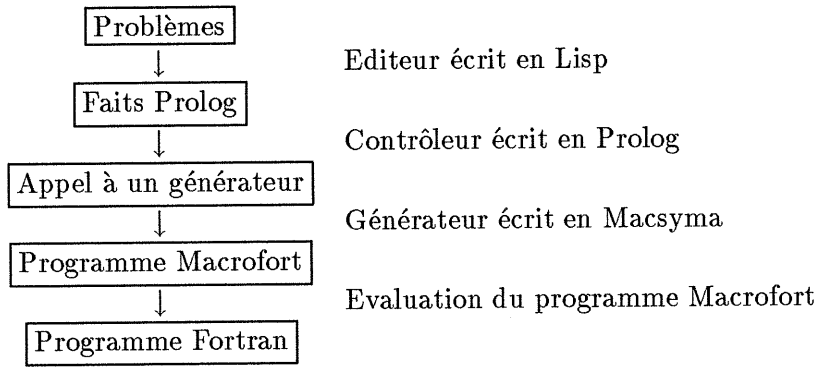
Nous ne parlerons que des deux derniers cas ici.

Pandore fera des choix à partir d'un module écrit en Prolog indiquant les conditions sous lesquelles une méthode particulière peut être employée. Une fois les méthodes connues, on pourra demander au système de résoudre numériquement par l'une des méthodes possibles. La commande *resoudre* fait ce travail. Mais à l'usage il apparait que la façon la plus sure et la plus rapide est de détailler les tâches. En demandant d'abord de générer les sous programmes par la commande *generer sous programme.* Les sous programmes résolvants le problème sont alors générés et mis dans une fenêtre EMACS. On pourra alors aller vérifier que tout c'est bien passé et lancer la compilation de ce fichier.

On demandera ensuite la génération du programme principal par la commande *generer programme principal* dont le rôle est de construire un programme faisant un essai numérique particulier — selon la précision demandée par l'utilisateur — et de donner l'accès de Macsyma aux résultats obtenus. Le programme est mis dans une fenêtre EMACS et peut être exploité de l'éditeur.

Les résultats numériques sont alors disponibles dans Pandore. Il reste à faire des dessins. La commande *figure* dessine les états, les commandes, les variables duales comme fonctions du temps.

Détaillons les étapes de la procédure de génération des sous programmes



Le point important est ici le langage Macrofort. Détaillons le et précisons son utilisation sur un exemple très simple.

8 MACROFORT

Macrofort peut être vu de deux façons. D'un côté c'est un langage de type Pascal écrit en Macsyma. D'un autre côté c'est un ensemble de facilités rajouté à Macsyma pour générer du code Fortran. Son intérêt réside pratiquement uniquement dans le second cas.

Ce dernier point de vue permet de cumuler la puissance de manipulation algébrique de Macsyma et la puissance numérique donnée par un compilateur Fortran. En effet Macsyma est capable de faire du calcul numérique mais de façon beaucoup moins efficace qu'un programme traditionnel Fortran ou C.

Macrofort possède 3 types d'instructions :

- les instructions élémentaires
- les macro instructions
- les instructions de gestion de piles spécialisées.

8.1 Instructions élémentaires

Ce sont des fonctions Macsyma dont le rôle est d'imprimer une instruction Fortran paramétrée.

Fonctions Macsyma	Fortran généré
equalf(niv,var,exp)	var=exp
stopf(niv)	STOP
returnf(niv)	RETURN
endf(niv)	END
programf(niv,nom)	PROGRAM nom
readf(niv,fich,etiq,[liste])	READ (fich,etiq)liste

niv représente un niveau d'indentation géré par le traducteur.

8.2 Macros instructions

Ce sont des fonctions Macsyma dont le rôle est de générer plusieurs instructions élémentaires de façon à augmenter la puissance d'expression de Fortran.

8.3 Les instructions de manipulation des piles

Elles permettent de mettre à jour des piles et de construire des programmes en ajoutant ou retranchant des instructions.

8.4 Génération du Code Fortran

On écrit un programme Fortran dont le but est de construire une liste d'instructions élémentaires en s'aidant éventuellement des macro instructions et des instructions de manipulation de piles.

Un traducteur appelé *aplat* traduit cette liste d'instructions élémentaires en un programme Fortran et met le résultat dans un fichier.

8.5 Exemple d'utilisation

L'exemple suivant montre un générateur de programmes Fortran qui résout par la méthode du gradient un problème d'optimisation. On donne un générateur, qui s'appelle *gradient-const-step*, qui a pour arguments la fonction à optimiser f et la liste des variables *vars* sur lesquelles on veut optimiser. En sortie nous obtenons le programme d'optimisation correspondant.

8.5.1 Le générateur écrit en Macsyma

```
gradient-const-step(f,vars):=block(
   [dim:length(vars),var],contexte(),
   for i:1 thru dim do ( var:vars[i],
    initm([equalf,var,0]),
    prog1m([equalf,concat(var,n),var-ro*diff(f,var)]),
    prog2m([equalf,var,concat(var,n)]) ),

aplat(
   subroutinem(gradient,[ro,eps],
    untilm(error<eps,[equalf,error,100],
     [prog1,
      [equalf,error,sum((concat(vars[i],n)-vars[i])^ 2,i,1,dim)],
      prog2
      [writem,8,vars,[concat(dim,f12\.5)]]]
))))$
```

8.5.2 L'appel du générateur

gradient-const-step (x*x + y*y,[x,y]).

8.5.3 Le programme Fortran généré

```
  SUBROUTINE gradient(ro,eps,ierr,maxuntil0)
   x=0
   y=0
   ierr=0
c-until error < eps faire liste-until
c-initialisation
    nuntil0=0
    error=100
c-debut-d'iteration-d'until
1000  CONTINUE
    nuntil0=nuntil0+1
c-debut-liste-until
      xn=x-2*ro*x
      yn=y-2*ro*y
      error=(xn-x)**2+(yn-y)**2
      x=xn
      y=yn
      WRITE(8,1003) x,y
c-fin-liste-until
c-tests-de-sortied'until
    IF (error.LT.eps)GOTO 1002
    IF (nuntil0.GT.maxuntil0) GOTO 1001
c-reiterer-until
    GOTO 1000
c-sortie-d'until-depassement-du-max-d'iter
 1001 CONTINUE
    WRITE (9,1004)
    ierr=1
 1002 CONTINUE
c-fin-d'until
 1003 FORMAT( 2 f12.5)
 1004 FORMAT( ' maxuntil0')
    END
```

9 GENERATION DE RAPPORT

Elle se fait de façon analogue à la génération de programmes, elle utilise Macrotex permettant de générer facilement du Latex à partir de Macsyma.

On dispose d'un ensemble de phrases variables — stockées sous formes de faits Prolog — et de fonctions Macsyma capables de faire les manipulations des formules nécessaires.

Un module écrit en Prolog gère alors cet ensemble de formats et de fonctions Macsyma et écrit un programme Macrotex dont l'évaluation conduit à l'écriture du rapport.

Décrivons maintenant le langage Macrotex permettant de faire de l'édition scientifique de qualité à partir de Macsyma.

10 MACROTEX

Macrotex peut être vu comme une extension de Tex connaissant de l'algèbre comme Macsyma, ou bien comme un ensemble de facilités ajouté à Macsyma pour éditer des expressions mathématiques.

On obtient ainsi un outil très puissant mariant la qualité d'édition de Tex et la puissance de manipulation algébrique de Macsyma.

Il est constitué de quatre types d'instructions :

- les instructions élémentaires,
- les macro instructions,
- les instructions de manipulation de piles,
- une instruction pour découper les grosses formules.

10.1 Generation du code Latex

Un programme *prog* Macrotex est construit en programmant dans Macsyma. Il est constitué d'instructions élémentaires. Sa construction utilise les utilitaires et les macro instructions de Macrotex. L'évaluation de la fonction *execute (prog)* génère le programme Latex en exécutant la liste d'instructions élémentaires de *prog*.

11 EXEMPLE D'UTILISATION DE PANDORE

Nous n'indiquerons pas précisément les connaissances mathématiques de Pandore dans le domaine de la commande optimale déterministe ou stochastique. Une fois intégré un ou deux modules supplémentaires existant actuellement seulement sous forme de générateurs de programmes ou de manipulateurs algébriques, l'état de l'art du domaine sera à peu près atteint.

Nous avons résolu par exemple un problème de rentrée de la navette dans l'atmosphère dans le cas le plus simple — trajectoire plane équatoriale.

RENTREE DE LA NAVETTE DANS L'ATMOSPHERE

Pandore
INRIA Domaine de Voluceau BP 105 Rocquencourt 78153 Le Chesnay France

March 29, 1989

Abstract

On considère le problème de rentrée de la navette dans l'atmosphère. Il s'agit de minimiser la longitude parcourue optimale plus un terme pénalisant l'écart à une cible. Le coût optimal satisfait une équation de Bellman obtenue par la méthode de la programmation dynamique. On ne résoudra pas ici cette équation. On applique la méthode de Pontriaguine qui permet d'obtenir les conditions nécessaires d'optimalité. On discrétise le problème en faisant des développements de Taylor formels à l'ordre 2 des équations d'évolution. On résout le problème discret par une méthode de gradient.

Contents

1 Notations

- Variables d'état : X_1, X_2, X_3

  X_1 : le module de la vitesse

X_2 : l'angle du vecteur vitesse avec l'horizontale

X_3 : l'altitude

- Variables de commande : U_1

  U_1 : l'angle d'incidence

- Coût optimal : V

  V : la longitude parcourue optimale plus un terme pénalisant l'écart à une cible

- Temps : t
- Dimension de l'état : n
- I-ème variable d'état : x_i
- Opérateur dérivée par rapport au temps : ∂_0
- Opérateur dérivée par rapport à x_i : ∂_i

2 Equation d'évolution du système

On considère le processus commandé défini par le système d'équations différentielles:

- Evolution du module de la vitesse

$$dX_{1t} = b_1(X_2, X_3, X_1, U_1)dt \tag{1}$$

- Evolution de l'angle du vecteur vitesse avec l'horizontale

$$dX_{2t} = b_2(X_1, X_2, X_3, U_1)dt \tag{2}$$

- Evolution de l'altitude

$$dX_{3t} = b_3(X_1, X_2)dt \tag{3}$$

avec

-

$$\begin{aligned} b_1(X_2, X_3, X_1, U_1) &= e_4 + e_3 + e_2 + e_1 \\ e_1 &= -3.08\ 10^3\, {U_1}^2 {X_1}^2 e^{-18.X_3} \\ e_2 &= -1.33\ 10^3\, U_1 {X_1}^2 e^{-18.X_3} \\ e_3 &= -3.34\ 10^3\, {X_1}^2 e^{-18.X_3} \\ e_4 &= -\frac{4.5 \sin\left(8.73\ 10^{-2} X_2\right)}{\left(3.65\ 10^{-4} {X_3}^2 + 3.82\ 10^{-2} X_3 + 1\right)} \end{aligned} \tag{4}$$

- $$\begin{aligned} b_2(X_1, X_2, X_3, U_1) &= e_7 + e_6 + e_5 \\ e_5 &= 1.68\ 10^5\ U_1 X_1 e^{-18.X_3} \\ e_6 &= -\frac{52.\cos\left(8.73\ 10^{-2}\ X_2\right)}{\left(3.65\ 10^{-4}\ X_1 {X_3}^2 + 3.82\ 10^{-2}\ X_1 X_3 + X_1\right)} \\ e_7 &= \frac{50. X_1 \cos\left(8.73\ 10^{-2}\ X_2\right)}{\left(1.91\ 10^{-2}\ X_3 + 1\right)} \end{aligned} \tag{5}$$

- $$b_3(X_1, X_2) = 2.30\ 10^2\ X_1 \sin\left(8.73\ 10^{-2}\ X_2\right) \tag{6}$$

où

- $X_1 \in [-\infty, +\infty]$
- $X_2 \in [-\infty, +\infty]$
- $X_3 \in [-\infty, +\infty]$
- $U_1 \in [-\infty, +\infty]$

3 Critère à optimiser

On définit le critère comme la longitude parcourue optimale plus un terme pénalisant l'écart à une cible.

Il s'agit donc de minimiser la fonctionnelle:

$$J(t_0, S) = c_f(X_1, X_2, X_3) + \int_{t_0}^{T_F} c(X_1, X_2, X_3)_t\, dt$$

dans la classe des feedbacks, i.e. des applications S: $[t, X_1, X_2, X_3] \mapsto [U_1]$

où l'on a noté

$$c(X_1, X_2, X_3) = -\frac{0.70 X_1 \cos\left(8.73\ 10^{-2}\ X_2\right)}{\left(1.91\ 10^{-2}\ X_3 + 1\right)} \tag{7}$$

et

$$c_f(X_1, X_2, X_3) = 100(X_3 - 0.20)^2 + 20(X_2 + 1)^2 + 100\left(X_1 - 9.76\ 10^{-2}\right)^2 \tag{8}$$

4 Conditions d'optimalité

On définit la fonction de Bellman V par:

$$V(t, y_1, y_2, y_3) = \min_S \left(J(t, S) \,|\, X_{1t} = y_1, X_{2t} = y_2, X_{3t} = y_3 \right)$$

V satisfait l'équation de la Programmation Dynamique[6,4]

$$\min_{U_1} \left(A(U_1)V + c(U_1)\right) + \partial_0 V = 0 \tag{9}$$

$$V_{T_F}(X_1, X_2, X_3) = c_f(X_1, X_2, X_3)$$

avec

$$A(U_1)V = b_3\partial_3 V + b_2\partial_2 V + b_1\partial_1 V$$

où les b_i , c et c_f sont définis en (4)(5)(6)(7)(8)

5 Etude de l'existence d'une solution de l'équation de Bellman

D'après P.L.Lions [11] , on sait qu'il existe une solution de l'équation de Bellman; la démonstration est basée sur le Principe du Maximum.

6 Résolution du problème de commande par la méthode de Pontriaguine

La méthode de Pontriaguine ([12] [1] [2] [8] [13] [5] [10] ...) est applicable car on considère un problème de commande déterministe sur un horizon fini et sans contraintes sur l'état du système.

$$\begin{aligned} \dot{x}_t &= b(x_t, u_t) \\ V(0, y) &= \min_{u(.)} \left(\int_0^{t_f} c(x_t, u_t)dt + c_f(x_{t_f}) | x_0 = y \right) \end{aligned} \tag{10}$$

b représente le terme de dérive , c le coût instantané, et c_f le coût final.

Utilisons le principe de Lagrange [1] pour obtenir les conditions d'optimalité de Pontriaguine. Notons $\mathcal{L}$ le Lagrangien défini par

$$\mathcal{L}(x(.), u(.), p) = \int_0^{t_f} p(b(x_t, u_t) - \dot{x}_t)dt + \int_0^{t_f} c(x_t, u_t)dt + c_f(x_{t_f})$$

p désigne la variable adjointe associée à x.
On a

$$\mathcal{L}(x(.), u(.), p) = \int_0^{t_f} H(x, u, p) - p\dot{x}_t dt + c_f(x_{t_f})$$

où H désigne l'hamiltonien du problème déterministe défini par

$$H(x, u, p) = b(x, u)p + c(x, u)$$

6.1 Conditions d'optimalité de Pontriaguine

On les obtient en annulant le gradient du Lagrangien respectivement par rapport à l'état x, la variable adjointe p, et la commande u:

- Système primal

$$\frac{\partial \mathcal{L}}{\partial p} = 0$$

∂ désigne la différentielle au sens de Gâteaux.
On retrouve ainsi la contrainte

$$\dot{x} = H_p(x, u_d, p)$$

$$x_0 = y$$

- Système dual

$$\frac{\partial \mathcal{L}}{\partial x} = 0$$

On intègre par parties le terme $\int_0^{t_f} p\dot{x}_t dt$

$$\int_0^{t_f} p\dot{x}_t dt = -\int_0^{t_f} \dot{p}x_t dt + p(t_f)x(t_f) - p(t_0)x(t_0)$$

On obtient ainsi l'équation de l'état adjoint

$$\dot{p} = -H_x(x, u_d, p)$$

$$p(t_f) = (c_f)_x(x_{t_f})$$

- Conditions d'optimalité de l'hamiltonien

$$H_u(x, u_d, p) = 0$$

u_d représente la commande optimale déterministe.

6.2 Discrétisation du problème de commande

Soit n la dimension de l'état. On considère le coût intégral comme une variable d'état. On introduit donc une variable d'état supplémentaire x_{n+1} solution de l'équation différentielle

$$\begin{aligned} \dot{x}_{n+1}(t) &= c(x_t, u_t) \\ x_{n+1}(0) &= 0 \end{aligned}$$

Le coût à optimiser se réduit à un coût sur l'état final

$$J(u(.)) = \Phi(x_{t_f}) \equiv c_f(x(t_f)) + x_{n+1}(t_f)$$

On approxime l'équation d'état à l'ordre 2 :

$$x_{t+h} = x_t + \frac{h^2 b_x b}{2} + hb$$

où b désigne la nouvelle dynamique :

$$b = (b_1, \ldots, b_n, c)$$

h désigne le pas en temps.
Soit p_{t+h} la variable adjointe associée à x_{t+h} .
Le Lagrangien discrétisé s'écrit:

$$\mathcal{L}_h = \Phi(x_{t_f}) - \sum_{t=0}^{t_f-1} p_{t+h}(x_{t+h} - x_t - \frac{h^2 b_x b}{2} - hb)$$

On définit l'hamiltonien discrétisé par:

$$\mathcal{H}_h(x, u, p) = px + hp\left(b + \frac{hb_x b}{2}\right) \tag{11}$$

Les conditions d'optimalité de Pontriaguine s'écrivent

- Système primal

$$\frac{\partial \mathcal{L}_h}{\partial p_{t+h}} = 0$$

que l'on peut écrire

$$x_{t+h} = \frac{\partial \mathcal{H}_h}{\partial p}(x_t, u_t, p_t)$$

soit

$$x_{t+h} = x_t + \frac{h^2 b_x b}{2} + hb$$

$$x_0 = y$$

- Système dual

$$\frac{\partial \mathcal{L}_h}{\partial x_t} = 0$$

que l'on peut écrire

$$p_t = \frac{\partial \mathcal{H}_h}{\partial x}(x_t, u_t, p_{t+h})$$

$$p_{t_f} = \Phi_x(x_{t_f})$$

soit

$$p_t = \frac{(h^2 p_{t+h} b_{xx} b + h^2 p_{t+h} b_x^2)}{2} + h p_{t+h} b_x + p_{t+h}$$

- Optimalité de l'hamiltonien

$$\frac{\partial \mathcal{H}_h}{\partial u}(x_t, u_t, p_t) = 0$$

6.3 Spécialisation à notre cas particulier

Dans notre cas, l'hamiltonien H du problème continu vaut :

$$H(X_1, X_2, X_3, U_1, P_1, P_2, P_3) = c + b_3 P_3 + b_2 P_2 + b_1 P_1$$

où les b_i et c sont définis en (4)(5)(6)(7).

Les conditions d'optimalité de Pontriaguine du problème continu deviennent :

6.3.1 Système primal

$$\begin{aligned} \dot{X}_1 &= b_1(X_2, X_3, X_1, U_1) \\ \dot{X}_2 &= b_2(X_1, X_2, X_3, U_1) \\ \dot{X}_3 &= b_3(X_1, X_2) \end{aligned}$$

avec les conditions initiales:

$$X_1(0) = 1.0$$

$$X_2(0) = -0.20$$

$$X_3(0) = 1.0$$

6.3.2 Système dual

$$\begin{aligned}
\dot{P}_1 &= -\partial_1 b_3 P_3 - \partial_1 b_2 P_2 - \partial_1 b_1 P_1 - \partial_1 c \\
\dot{P}_2 &= -\partial_2 b_3 P_3 - \partial_2 b_2 P_2 - \partial_2 b_1 P_1 - \partial_2 c \\
\dot{P}_3 &= -\partial_3 b_3 P_3 - \partial_3 b_2 P_2 - \partial_3 b_1 P_1 - \partial_3 c
\end{aligned}$$

avec les conditions finales sur les variables duales:

$$\begin{aligned}
P1(t_f) &= \partial_1 c_f(X_1, X_2, X_3) \\
P2(t_f) &= \partial_2 c_f(X_1, X_2, X_3) \\
P3(t_f) &= \partial_3 c_f(X_1, X_2, X_3)
\end{aligned}$$

Explicitons les dérivées $\partial_i b$, $\partial_i c$ et $\partial_i c_f$

- $\partial_1 \mathbf{b_1}$

$$\begin{aligned}
& e_9 + e_8 + e_{10} \\
& e_{10} = -6.68\ 10^3\, X_1 e^{-18.X_3} \\
& e_8 = -6.16\ 10^3\, {U_1}^2 X_1 e^{-18.X_3} \\
& e_9 = -2.66\ 10^3\, U_1 X_1 e^{-18.X_3}
\end{aligned} \tag{12}$$

- $\partial_1 \mathbf{b_2}$

$$\begin{aligned}
& e_{13} + e_{12} + e_{11} \\
& e_{11} = 1.68\ 10^5\, U_1 e^{-18.X_3} \\
& e_{12} = \frac{e_{14}}{e_{15}} \\
& e_{14} = 52.\cos\left(8.73\ 10^{-2}\, X_2\right)\left(3.65\ 10^{-4}\, {X_3}^2 + 3.82\ 10^{-2}\, X_3 + 1\right) \\
& e_{15} = \left(3.65\ 10^{-4}\, X_1 {X_3}^2 + 3.82\ 10^{-2}\, X_1 X_3 + X_1\right)^2 \\
& e_{13} = \frac{50.\cos\left(8.73\ 10^{-2}\, X_2\right)}{\left(1.91\ 10^{-2}\, X_3 + 1\right)}
\end{aligned} \tag{13}$$

- $\partial_1 \mathbf{b_3}$

$$2.30\ 10^2 \sin\left(8.73\ 10^{-2}\, X_2\right) \tag{14}$$

- $\partial_2 \mathbf{b_1}$

$$-\frac{0.39\cos\left(8.73\ 10^{-2}\, X_2\right)}{\left(3.65\ 10^{-4}\, {X_3}^2 + 3.82\ 10^{-2}\, X_3 + 1\right)} \tag{15}$$

- $\partial_2 \mathbf{b_2}$

$$e_{17} + e_{16}$$

$$e_{16} = \frac{4.5 \sin\left(8.73\ 10^{-2}\ X_2\right)}{\left(3.65\ 10^{-4}\ X_1 {X_3}^2 + 3.82\ 10^{-2}\ X_1 X_3 + X_1\right)} \tag{16}$$

$$e_{17} = -\frac{4.4 X_1 \sin\left(8.73\ 10^{-2}\ X_2\right)}{\left(1.91\ 10^{-2}\ X_3 + 1\right)}$$

- $\partial_2 \mathbf{b_3}$

$$20. X_1 \cos\left(8.73\ 10^{-2}\ X_2\right) \tag{17}$$

- $\partial_3 \mathbf{b_1}$

$$e_{21} + e_{20} + e_{19} + e_{18}$$

$$e_{18} = 5.61\ 10^4\ {U_1}^2 {X_1}^2 e^{-18. X_3}$$

$$e_{19} = 2.42\ 10^4\ U_1 {X_1}^2 e^{-18. X_3}$$

$$e_{20} = 6.08\ 10^4\ {X_1}^2 e^{-18. X_3} \tag{18}$$

$$e_{21} = \frac{e_{22}}{e_{23}}$$

$$e_{22} = 4.5 \sin\left(8.73\ 10^{-2}\ X_2\right) \left(7.31\ 10^{-4}\ X_3 + 3.82\ 10^{-2}\right)$$

$$e_{23} = \left(3.65\ 10^{-4}\ {X_3}^2 + 3.82\ 10^{-2}\ X_3 + 1\right)^2$$

- $\partial_3 \mathbf{b_2}$

$$e_{26} + e_{25} + e_{24}$$

$$e_{24} = -3.05\ 10^6\ U_1 X_1 e^{-18. X_3}$$

$$e_{25} = \frac{e_{27}}{e_{15}} \tag{19}$$

$$e_{15} = \left(3.65\ 10^{-4}\ X_1 {X_3}^2 + 3.82\ 10^{-2}\ X_1 X_3 + X_1\right)^2$$

$$e_{27} = 52. \cos\left(8.73\ 10^{-2}\ X_2\right) \left(7.31\ 10^{-4}\ X_1 X_3 + 3.82\ 10^{-2}\ X_1\right)$$

$$e_{26} = -0.96 X_1 \cos\left(8.73\ 10^{-2}\ X_2\right) \left(1.91\ 10^{-2}\ X_3 + 1\right)^{-2}$$

- $\partial_3 \mathbf{b_3}$

$$0 \tag{20}$$

- $\partial_1 \mathbf{c}$

$$-\frac{0.70 \cos\left(8.73\ 10^{-2}\ X_2\right)}{\left(1.91\ 10^{-2}\ X_3 + 1\right)}$$

- $\partial_2 c$

$$\frac{6.12\ 10^{-2}\ X_1 \sin\left(8.73\ 10^{-2}\ X_2\right)}{\left(1.91\ 10^{-2}\ X_3 + 1\right)}$$

- $\partial_3 c$

$$1.34\ 10^{-2}\ X_1 \cos\left(8.73\ 10^{-2}\ X_2\right)\left(1.91\ 10^{-2}\ X_3 + 1\right)^{-2}$$

- $\partial_1 c_f$

$$200\left(X_1 - 9.76\ 10^{-2}\right)$$

- $\partial_2 c_f$

$$40(X_2 + 1)$$

- $\partial_3 c_f$

$$200(X_3 - 0.20)$$

6.3.3 Optimalité de l'hamiltonien

$$\partial_{U_1} c + P_3 \partial_{U_1} b_3 + P_2 \partial_{U_1} b_2 + P_1 \partial_{U_1} b_1 \quad = \quad 0$$

avec

- $\partial_{U_1} b_1$

$$-1.33\ 10^3\ {X_1}^2 e^{-18.X_3} - 6.16\ 10^3\ U_1 {X_1}^2 e^{-18.X_3} \tag{21}$$

- $\partial_{U_1} b_2$

$$1.68\ 10^5\ X_1 e^{-18.X_3} \tag{22}$$

- $\partial_{U_1} b_3$

$$0 \tag{23}$$

- $\partial_{U_1} c$

$$0 \tag{24}$$

6.3.4 Le problème discret

L'hamiltonien du problème discret est donné par (11) avec

$$b = [b_1, b_2, b_3, c]$$

$$b_x = \begin{pmatrix} \partial_1 b_1 & \partial_2 b_1 & \partial_3 b_1 & 0 \\ \partial_1 b_2 & \partial_2 b_2 & \partial_3 b_2 & 0 \\ \partial_1 b_3 & \partial_2 b_3 & 0 & 0 \\ \partial_1 c & \partial_2 c & \partial_3 c & 0 \end{pmatrix}$$

les b_i et $\partial_i b_j$ sont définis précédemment.

b_{xx} est le tenseur des dérivées secondes. Explicitons les :

1. Dérivées secondes de b_1

$$\begin{pmatrix} \partial_{11}b_1 & 0 & \partial_{13}b_1 & 0 \\ 0 & \partial_{22}b_1 & \partial_{23}b_1 & 0 \\ \partial_{31}b_1 & \partial_{32}b_1 & \partial_{33}b_1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- $\partial_{11}\mathbf{b_1}$

$$e_{30} + e_{29} + e_{28}$$

$$e_{28} = -6.16\ 10^3\, {U_1}^2 e^{-18.X_3}$$

$$e_{29} = -2.66\ 10^3\, U_1 e^{-18.X_3}$$

$$e_{30} = -6.68\ 10^3\, e^{-18.X_3}$$

- $\partial_{22}\mathbf{b_1}$

$$\frac{3.44\ 10^{-2}\ \sin\left(8.73\ 10^{-2}\ X_2\right)}{\left(3.65\ 10^{-4}\ {X_3}^2 + 3.82\ 10^{-2}\ X_3 + 1\right)}$$

- $\partial_{31}\mathbf{b_1}$

$$0.29e_5 + e_{32} + e_{31}$$

$$e_{31} = 1.12\ 10^5\, {U_1}^2 X_1 e^{-18.X_3}$$

$$e_{32} = 1.22\ 10^5\, X_1 e^{-18.X_3}$$

$$e_5 = 1.68\ 10^5\, U_1 X_1 e^{-18.X_3}$$

- $\partial_{32}\mathbf{b_1}$

$$\frac{e_{33}}{e_{23}}$$

$$e_{23} = \left(3.65\ 10^{-4}\ {X_3}^2 + 3.82\ 10^{-2}\ X_3 + 1\right)^2$$

$$e_{33} = 0.39 \cos\left(8.73\ 10^{-2}\ X_2\right)\left(7.31\ 10^{-4}\ X_3 + 3.82\ 10^{-2}\right)$$

- $\partial_{33}\mathbf{b_1}$

$$e_{38} + e_{37} + e_{36} + e_{35} + e_{34}$$

$$e_{34} = -1.02\ 10^{6}\, {U_1}^2 {X_1}^2 e^{-18.X_3}$$

$$e_{35} = -4.40\ 10^{5}\, U_1 {X_1}^2 e^{-18.X_3}$$

$$e_{36} = -1.11\ 10^{6}\, {X_1}^2 e^{-18.X_3}$$

$$e_{37} = 3.30\ 10^{-3} \sin\left(8.73\ 10^{-2}\, X_2\right) \left(3.65\ 10^{-4}\, {X_3}^2 + 3.82\ 10^{-2}\, X_3 + 1\right)^{-2}$$

$$e_{38} = -\frac{e_{40}}{e_{41}}$$

$$e_{40} = \sin\left(8.73\ 10^{-2}\, X_2\right) \left(7.31\ 10^{-4}\, X_3 + 3.82\ 10^{-2}\right)^{2}$$

$$e_{41} = \left(3.65\ 10^{-4}\, {X_3}^2 + 3.82\ 10^{-2}\, X_3 + 1\right)^{3}$$

2. Dérivées secondes de b_2

$$\begin{pmatrix} \partial_{11}b_2 & \partial_{12}b_2 & \partial_{13}b_2 & 0 \\ \partial_{21}b_2 & \partial_{22}b_2 & \partial_{23}b_2 & 0 \\ \partial_{31}b_2 & \partial_{32}b_2 & \partial_{33}b_2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- $\partial_{11}\mathbf{b_2}$

$$-\frac{e_{43}}{e_{44}}$$

$$e_{43} = \cos\left(8.73\ 10^{-2}\, X_2\right) \left(3.65\ 10^{-4}\, {X_3}^2 + 3.82\ 10^{-2}\, X_3 + 1\right)^{2}$$

$$e_{44} = \left(3.65\ 10^{-4}\, X_1 {X_3}^2 + 3.82\ 10^{-2}\, X_1 X_3 + X_1\right)^{3}$$

- $\partial_{21}\mathbf{b_2}$

$$e_{46} + e_{45}$$

$$e_{45} = -\frac{e_{48}}{e_{15}}$$

$$e_{15} = \left(3.65\ 10^{-4}\, X_1 {X_3}^2 + 3.82\ 10^{-2}\, X_1 X_3 + X_1\right)^{2}$$

$$e_{48} = \sin\left(8.73\ 10^{-2}\, X_2\right) \left(3.65\ 10^{-4}\, {X_3}^2 + 3.82\ 10^{-2}\, X_3 + 1\right)$$

$$e_{46} = -\frac{4.4 \sin\left(8.73\ 10^{-2}\, X_2\right)}{\left(1.91\ 10^{-2}\, X_3 + 1\right)}$$

- $\partial_{22}\mathbf{b_2}$

$$e_{50} + e_{49}$$

$$e_{49} = \frac{0.39 \cos\left(8.73\ 10^{-2}\ X_2\right)}{\left(3.65\ 10^{-4}\ X_1 {X_3}^2 + 3.82\ 10^{-2}\ X_1 X_3 + X_1\right)}$$

$$e_{50} = -\frac{0.38 X_1 \cos\left(8.73\ 10^{-2}\ X_2\right)}{\left(1.91\ 10^{-2}\ X_3 + 1\right)}$$

- $\partial_{31}\mathbf{b_2}$

$$e_{54} + e_{53} + e_{52} + e_{51}$$

$$e_{51} = -3.05\ 10^{6}\ U_1 e^{-18.X_3}$$

$$e_{52} = \frac{1.31\ 10^{2}\ e_{33}}{e_{15}}$$

$$e_{15} = \left(3.65\ 10^{-4}\ X_1 {X_3}^2 + 3.82\ 10^{-2}\ X_1 X_3 + X_1\right)^2$$

$$e_{33} = 0.39 \cos\left(8.73\ 10^{-2}\ X_2\right)\left(7.31\ 10^{-4}\ X_3 + 3.82\ 10^{-2}\right)$$

$$e_{53} = -\frac{e_{56}}{e_{44}}$$

$$e_{44} = \left(3.65\ 10^{-4}\ X_1 {X_3}^2 + 3.82\ 10^{-2}\ X_1 X_3 + X_1\right)^3$$

$$e_{56} = e_{57} e_{58} e_{59}$$

$$e_{57} = \cos\left(8.73\ 10^{-2}\ X_2\right)$$

$$e_{58} = 7.31\ 10^{-4}\ X_1 X_3 + 3.82\ 10^{-2}\ X_1$$

$$e_{59} = 3.65\ 10^{-4}\ {X_3}^2 + 3.82\ 10^{-2}\ X_3 + 1$$

$$e_{54} = -0.96 \cos\left(8.73\ 10^{-2}\ X_2\right)\left(1.91\ 10^{-2}\ X_3 + 1\right)^{-2}$$

- $\partial_{32}\mathbf{b_2}$

$$e_{61} + e_{60}$$

$$e_{60} = 8.42\ 10^{-2}\ X_1 \sin\left(8.73\ 10^{-2}\ X_2\right)\left(1.91\ 10^{-2}\ X_3 + 1\right)^{-2}$$

$$e_{61} = -\frac{e_{63}}{e_{15}}$$

$$e_{15} = \left(3.65\ 10^{-4}\ X_1 {X_3}^2 + 3.82\ 10^{-2}\ X_1 X_3 + X_1\right)^2$$

$$e_{63} = \sin\left(8.73\ 10^{-2}\ X_2\right)\left(7.31\ 10^{-4}\ X_1 X_3 + 3.82\ 10^{-2}\ X_1\right)$$

- $\partial_{33}\mathbf{b_2}$

$$e_{66} + e_{65} + e_{64} + 3.31\ 10^2\, e_5$$

$$e_5 = 1.68\ 10^5\, U_1 X_1 e^{-18.X_3}$$

$$e_{64} = \frac{e_{67}}{e_{15}}$$

$$e_{15} = \left(3.65\ 10^{-4}\, X_1 X_3{}^2 + 3.82\ 10^{-2}\, X_1 X_3 + X_1\right)^2$$

$$e_{67} = 3.79\ 10^{-2}\, X_1 \cos\left(8.73\ 10^{-2}\, X_2\right)$$

$$e_{65} = -\frac{e_{69}}{e_{44}}$$

$$e_{44} = \left(3.65\ 10^{-4}\, X_1 X_3{}^2 + 3.82\ 10^{-2}\, X_1 X_3 + X_1\right)^3$$

$$e_{69} = \cos\left(8.73\ 10^{-2}\, X_2\right)\left(7.31\ 10^{-4}\, X_1 X_3 + 3.82\ 10^{-2}\, X_1\right)^2$$

$$e_{66} = 3.69\ 10^{-2}\, X_1 \cos\left(8.73\ 10^{-2}\, X_2\right)\left(1.91\ 10^{-2}\, X_3 + 1\right)^{-3}$$

3. Dérivées secondes de b_3

$$\begin{pmatrix} 0 & \partial_{12}b_3 & 0 & 0 \\ \partial_{21}b_3 & \partial_{22}b_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- $\partial_{21}\mathbf{b_3}$

$$20.\cos\left(8.73\ 10^{-2}\, X_2\right)$$

- $\partial_{22}\mathbf{b_3}$

$$-1.8 X_1 \sin\left(8.73\ 10^{-2}\, X_2\right)$$

4. Dérivées secondes de c

$$\begin{pmatrix} 0 & \partial_{12}c & \partial_{13}c & 0 \\ \partial_{21}c & \partial_{22}c & \partial_{23}c & 0 \\ \partial_{31}c & \partial_{32}c & \partial_{33}c & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- $\partial_{21}\mathbf{c}$

$$\frac{6.12\ 10^{-2}\, \sin\left(8.73\ 10^{-2}\, X_2\right)}{\left(1.91\ 10^{-2}\, X_3 + 1\right)}$$

- $\partial_{22}\mathbf{c}$

$$\frac{5.34\ 10^{-3}\ X_1 \cos\left(8.73\ 10^{-2}\ X_2\right)}{\left(1.91\ 10^{-2}\ X_3 + 1\right)}$$

- $\partial_{31}\mathbf{c}$

$$1.34\ 10^{-2}\ \cos\left(8.73\ 10^{-2}\ X_2\right)\left(1.91\ 10^{-2}\ X_3 + 1\right)^{-2}$$

- $\partial_{32}\mathbf{c}$

$$-1.17\ 10^{-3}\ X_1 \sin\left(8.73\ 10^{-2}\ X_2\right)\left(1.91\ 10^{-2}\ X_3 + 1\right)^{-2}$$

- $\partial_{33}\mathbf{c}$

$$-5.12\ 10^{-4}\ X_1 \cos\left(8.73\ 10^{-2}\ X_2\right)\left(1.91\ 10^{-2}\ X_3 + 1\right)^{-3}$$

6.3.5 Méthode de gradient

La commande optimale se calcule en utilisant un algorithme de gradient à pas ρ adaptatif :

$$u_{n+1} = u_n - \rho \frac{dH}{du_n}$$

avec

$$\frac{dH}{dU_1} = \partial_{U_1} c + P_3 \partial_{U_1} b_3 + P_2 \partial_{U_1} b_2 + P_1 \partial_{U_1} b_1$$

où les $\partial_{U_1} b_1, \partial_{U_1} b_2, \partial_{U_1} b_3, \partial_{U_1} c$ sont définis en (21)(22)(23)(24).

L'adaptation se fait au moyen d'un algorithme de type gradient stochastique cherchant à maximiser la descente de chaque itération.

6.4 Résultats numériques

On représente sur les courbes suivantes les trajectoires des états, des commandes optimales et des variables duales. On donne également l'évolution du côut avec les itérations de gradient.

A Appendix:Les programmes Fortran générés

```
      SUBROUTINE PONTRIAGN(ETAT1,ETAT2,ETAT3,ETAT4,VAL1,VAL2,VAL3,VAL4,C
     1   OM1,HT,NT,CNEW,RO,EPS,IERRO,MAXUNTILO)
      REAL VAL1(NT),VAL2(NT),VAL3(NT),VAL4(NT),ETAT1(NT),ETAT2(NT),ETAT3
     1   (NT),ETAT4(NT),COM1(NT)
      EXTERNAL RO
c
c
C                Resout le probleme de controle optimal defini par:
C             - l'hamiltonien continu: P1
C                          2   2    - 18.197014 X3
C         (- 3081.2178 U1  X1  %E
C                          2   - 18.197014 X3
C         - 1328.6561 U1 X1  %E
C                       2   - 18.197014 X3
C         - 3339.4087 X1  %E
C               4.520514 SIN(0.08726646 X2)
C         - ------------------------------------)
C                         2
C           3.6539137e-4 X3  + 0.038230427 X3 + 1
C                              - 18.197014 X3
C         + P2 (167707.0 U1 X1 %E
C                  51.801273 COS(0.08726646 X2)
C         - ----------------------------------------
C                           2
C           3.6539137e-4 X1 X3  + 0.038230427 X1 X3 + X1
C           50.46856 X1 COS(0.08726646 X2)
C         + ----------------------------)
C                0.019115213 X3 + 1
C           0.7009522 P4 X1 COS(0.08726646 X2)
C         - --------------------------------
C                  0.019115213 X3 + 1
C         + 230.40355 P3 X1 SIN(0.08726646 X2)
C             l'hamiltonien discret est obtenu en faisant
C             un developpement limite a l'ordre 2
c
C             - le cout sur l'etat final:
C                                  2   2         2
C                X4 + 100 (X3 - 0.2)  + 20 (X2 + 1)  + 100 (X1 - 0.0976)
C             - les contraintes sur le controle :
C                       MINF =< U1 =< INF
c
C                 Les parametres du sousprogramme sont:
C             - ETAI,1=<i=< 4 , les variables d'etat,
C             - COMI,1=<i=< 1 , les variables de commande,
C             - VALI,1=<i=< 4  , les variables duales,
C             - HT  le pas en temps,
```

```
C            - NT le nombre de pas en temps,
C            - CNEW le nouveau cout dans l'iteration,
C            - EPS le test de convergence de la methode du gradient,
C            - MAXUNTIL0 le nombre maxi d'iterations,
C            - IERR indicateur de sortie par maxuntil0: ierr=1,
C            - R0(N) la fonction definissant le pas  de la methode du
C            gradient comme fonction du numero d'iteration N.
c
        IERR0=0
c-faire
                DO 1000 I0=1,NT
                    COM1(I0)=1.0
 1000           CONTINUE
c-fin-de-faire.
c-until ABS(COLD - CNEW) . LT . EPS - faire-liste_until
c-initialisation
                NUNTIL0=0
                CNEW=1.0e10
                WRITE(8,1008)
                NT1=NT-1
c-debut-d'iteration-d'until
 1001          CONTINUE
               NUNTIL0=NUNTIL0+1
c-debut-liste_until
                     X1=ETAT1(1)
                     X2=ETAT2(1)
                     X3=ETAT3(1)
                     X4=ETAT4(1)
c-faire
                       DO 1002 I0=1,NT1
                           X0=HT*(I0-1)
                           U1=COM1(I0)
                                 RX1=0.08726646*X2
                                 RX2=SIN(RX1)
                                 RX3=X3**2
                                 RX4=0.038230427*X3+3.6539137e-4*RX3+1
                                 RX5=1/RX4
                                 RX6=-4.520514*RX2*RX5
                                 RX7=X1**2
                                 RX8=EXP(-18.197014*X3)
                                 RX9=-3339.4087*RX7*RX8
                                 RX10=-1328.6561*RX7*RX8*U1
                                 RX11=U1**2
                                 RX12=-3081.2178*RX11*RX7*RX8
                                 RX13=RX9+RX6+RX12+RX10
                                 RX14=HT**2
                                 RX15=X1**3
                                 RX16=COS(RX1)
```

```
                              RX17=1+0.019115213*X3
                              RX18=1/RX17
                              RX19=0.038230427*X1*X3+3.6539137e-4*RX3*
     1   X1+X1
                              RX20=1/RX19
                              RX21=167707.0*RX8*U1*X1+50.46856*RX16*RX
     1   18*X1-51.801273*RX16*RX20
                              RX22=1/RX17**2
                                XN0=0.5*RX14*(4.520514*RX2*(0.16837494
     1   *RX2*X1*X3+8.808426*RX2*X1)/RX4**2-2657.3123*RX13*RX8*U1*X1-616
     2   2.4355*RX11*RX13*RX8*X1-6678.8174*RX13*RX8*X1+5570599.0*RX15*RX
     3   2*RX8*U1+1.2918488e7*RX11*RX15*RX2*RX8+1.4000994e7*RX15*RX2*RX8
     4   -0.39448926*RX16*RX21*RX5)+X1+HT*RX13
                                XN1=0.5*RX14*(51.801273*RX16*(0.168374
     1   94*RX2*RX7*X3+0.038230427*RX13*X3+RX9+8.808426*RX2*RX7+RX6+3.65
     2   39137e-4*RX13*RX3+RX12+RX10)/RX19**2-4.4042125*RX18*RX2*RX21*X1
     3   -7.0313786e8*RX2*RX7*RX8*U1+167707.0*RX13*RX8*U1-222.27428*RX16
     4   *RX2*RX22*RX7+4.5205135*RX2*RX20*RX21+50.46856*RX13*RX16*RX18)+
     5   X2+HT*RX21
                                XN2=X3+0.5*RX14*(20.106503*RX16*RX21*X
     1   1+230.40355*RX13*RX2)+230.40355*HT*RX2*X1
                                XN3=X4+0.5*RX14*(0.061169613*RX18*RX2*
     1   RX21*X1+3.0871427*RX16*RX2*RX22*RX7-0.7009522*RX13*RX16*RX18)-0
     2   .7009522*HT*RX16*RX18*X1
                        ETAT1(IO+1)=XN0
                        ETAT2(IO+1)=XN1
                        ETAT3(IO+1)=XN2
                        ETAT4(IO+1)=XN3
                        X1=XN0
                        X2=XN1
                        X3=XN2
                        X4=XN3
 1002                 CONTINUE
c-fin-de-faire.
                    X0=HT*(NT-1)
                    X1=ETAT1(NT)
                    X2=ETAT2(NT)
                    X3=ETAT3(NT)
                    X4=ETAT4(NT)
                    P1=200*(X1-0.0976)
                    P2=40*(X2+1)
                    P3=200*(X3-0.2)
                    P4=1
                    VAL1(NT)=P1
                    VAL2(NT)=P2
                    VAL3(NT)=P3
                    VAL4(NT)=P4
c-faire
```

```
                DO 1003 IO=1,NT1
                    XO=HT*(NT-IO)
                    X1=ETAT1(NT-IO)
                    X2=ETAT2(NT-IO)
                    X3=ETAT3(NT-IO)
                    X4=ETAT4(NT-IO)
                    U1=COM1(NT-IO)
                          RX1=0.08726646*X2
                          RX2=SIN(RX1)
                          RX3=COS(RX1)
                          RX4=1+0.019115213*X3
                          RX5=1/RX4
                          RX6=X3**2
                          RX7=0.038230427*X3+3.6539137e-4*RX6+1
                          RX8=0.038230427*X1*X3+3.6539137e-4*RX6*X
     1    1+X1
                          RX9=1/RX8**2
                          RX10=EXP(-18.197014*X3)
                          RX11=167707.0*RX10*U1+51.801273*RX3*RX7*
     1    RX9+50.46856*RX3*RX5
                          RX12=-6678.8174*RX10*X1
                          RX13=-2657.3123*RX10*U1*X1
                          RX14=U1**2
                          RX15=-6162.4355*RX10*RX14*X1
                          RX16=RX12+RX13+RX15
                          RX17=HT**2
                          RX18=1/RX4**2
                          RX19=1/RX8
                          RX20=167707.0*RX10*U1*X1+50.46856*RX3*RX
     1    5*X1-51.801273*RX19*RX3
                          RX21=1/RX7**2
                          RX22=X1**2
                          RX23=1/RX7
                          RX24=-4.520514*RX2*RX23
                          RX25=-3339.4087*RX10*RX22
                          RX26=-1328.6561*RX10*RX22*U1
                          RX27=-3081.2178*RX10*RX14*RX22
                          RX28=RX24+RX25+RX26+RX27
                          RX29=1/RX8**3
                          RX30=0.038230427*RX28*X3+0.16837494*RX2*
     1    RX22*X3+3.6539137e-4*RX28*RX6+RX27+RX26+RX25+RX24+8.808426*RX2*
     2    RX22
                          RX31=4.5205135*RX19*RX2-4.4042125*RX2*RX
     1    5*X1
                          RX32=RX3**2
                          RX33=RX2**2
                          RX34=0.16837494*RX2*X1*X3+8.808426*RX2*X
     1    1
```

```
                              RX35=X1**3
                              RX36=0.038230427*X1+7.3078275e-4*X1*X3
                              RX37=-3051766.5*RX10*U1*X1-0.96471727*RX
     1   18*RX3*X1+51.801273*RX3*RX36*RX9
                              RX38=0.038230427+7.3078275e-4*X3
                              RX39=4.520514*RX2*RX21*RX38
                              RX40=60767.266*RX10*RX22
                              RX41=24177.574*RX10*RX22*U1
                              RX42=56068.96*RX10*RX14*RX22
                              RX43=RX39+RX40+RX41+RX42
                              RX44=1/RX4**3
                                PN0=0.5*RX17*(P2*(51.801273*RX3*RX9*(0
     1   .33674988*RX2*X1*X3+0.038230427*RX16*X3+17.616852*RX2*X1+3.6539
     2   137e-4*RX16*RX6+RX15+RX13+RX12)-1.4062757e9*RX10*RX2*U1*X1-4.40
     3   42125*RX11*RX2*RX5*X1-444.54855*RX18*RX2*RX3*X1+167707.0*RX10*R
     4   X16*U1-4.5205135*RX2*RX20*RX7*RX9-103.60255*RX29*RX3*RX30*RX7+5
     5   0.46856*RX16*RX3*RX5-4.4042125*RX2*RX20*RX5+4.5205135*RX11*RX19
     6   *RX2)+P1*(4.520514*RX2*RX21*(0.16837494*RX2*X3+8.808426*RX2)-26
     7   57.3123*RX10*RX16*U1*X1-6162.4355*RX10*RX14*RX16*X1-6678.8174*R
     8   X10*RX16*X1-2657.3123*RX10*RX28*U1+1.6711797e7*RX10*RX2*RX22*U1
     9   -0.39448926*RX11*RX23*RX3-6162.4355*RX10*RX14*RX28-6678.8174*RX
     :   10*RX28+3.8755464e7*RX10*RX14*RX2*RX22+4.2002984e7*RX10*RX2*RX2
     ;   2)+0.061169613*P4*RX11*RX2*RX5*X1+6.1742854*P4*RX18*RX2*RX3*X1+
     <   20.106503*P3*RX11*RX3*X1-0.7009522*P4*RX16*RX3*RX5+0.061169613*
     =   P4*RX2*RX20*RX5+20.106503*P3*RX20*RX3+230.40355*P3*RX16*RX2)+HT
     >   *(-0.7009522*P4*RX3*RX5+230.40355*P3*RX2+P1*RX16+P2*RX11)+P1
                                PN1=0.5*RX17*(P1*(4.520514*RX2*RX21*(0
     1   .014693485*RX3*X1*X3+0.76868016*RX3*X1)+1048.2811*RX10*RX23*RX3
     2   *U1*X1+2431.0146*RX10*RX14*RX23*RX3*X1+2634.7217*RX10*RX23*RX3*
     3   X1+486126.47*RX10*RX3*RX35*U1+1127350.8*RX10*RX14*RX3*RX35+1221
     4   817.1*RX10*RX3*RX35+0.39448926*RX21*RX3*RX34-0.39448926*RX23*RX
     5   3*RX31+0.03442568*RX2*RX20*RX23)+P2*(51.801273*RX3*RX9*(-0.0150
     6   81492*RX23*RX3*X3+0.014693485*RX22*RX3*X3-1.4414298e-4*RX23*RX3
     7   *RX6-0.39448926*RX23*RX3+0.76868016*RX22*RX3)-4.4042125*RX2*RX3
     8   1*RX5*X1-0.38434002*RX20*RX3*RX5*X1-66158.61*RX10*RX23*RX3*U1-6
     9   .136035e7*RX10*RX22*RX3*U1-4.5205135*RX2*RX30*RX9-19.909304*RX2
     :   3*RX32*RX5-4.4042125*RX2*RX28*RX5+19.397089*RX18*RX22*RX33-19.3
     ;   97089*RX18*RX22*RX32+4.5205135*RX19*RX2*RX31+0.3944892*RX19*RX2
     <   0*RX3)+0.061169613*P4*RX2*RX31*RX5*X1+0.0053380555*P4*RX20*RX3*
     =   RX5*X1+20.106503*P3*RX3*RX31*X1-1.7546233*P3*RX2*RX20*X1+0.2765
     >   181*P4*RX23*RX32*RX5+0.061169613*P4*RX2*RX28*RX5-0.26940402*P4*
     ?   RX18*RX22*RX33+0.26940402*P4*RX18*RX22*RX32+20.106503*P3*RX28*R
     @   X3-90.89172*P3*RX2*RX23*RX3)+HT*(0.061169613*P4*RX2*RX5*X1+20.1
     1   06503*P3*RX3*X1+P2*RX31-0.39448926*P1*RX23*RX3)+P2
                                PN2=0.5*RX17*(P2*(51.801273*RX3*RX9*(0
     1   .038230427*RX43*X3+7.3078275e-4*RX28*X3+3.6539137e-4*RX43*RX6+R
     2   X42+RX41+RX40+RX39+0.038230427*RX28+0.16837494*RX2*RX22)-4.4042
     3   125*RX2*RX37*RX5*X1+0.08418746*RX18*RX2*RX20*X1+167707.0*RX10*R
```

```
     4    X43*U1-3051766.5*RX10*RX28*U1+1.2795009e10*RX10*RX2*RX22*U1-4.5
     5    205135*RX2*RX20*RX36*RX9+50.46856*RX3*RX43*RX5+8.497641*RX2*RX2
     6    2*RX3*RX44+4.5205135*RX19*RX2*RX37-103.60255*RX29*RX3*RX30*RX36
     7    -0.96471727*RX18*RX28*RX3)+P1*(-2657.3123*RX10*RX43*U1*X1+48355
     8    .15*RX10*RX28*U1*X1-6162.4355*RX10*RX14*RX43*X1-6678.8174*RX10*
     9    RX43*X1+0.7611413*RX21*RX33*X1+112137.92*RX10*RX14*RX28*X1+1215
     :    34.53*RX10*RX28*X1-1.01368264e8*RX10*RX2*RX35*U1-9.041028*RX2*R
     ;    X34*RX38/RX7**3+0.39448926*RX20*RX21*RX3*RX38-0.39448926*RX23*R
     <    X3*RX37-2.350779e8*RX10*RX14*RX2*RX35-2.5477629e8*RX10*RX2*RX35
     =    )+0.061169613*P4*RX2*RX37*RX5*X1+20.106503*P3*RX3*RX37*X1-0.001
     >    1692703*P4*RX18*RX2*RX20*X1-0.7009522*P4*RX3*RX43*RX5-0.1180227
     ?    85*P4*RX2*RX22*RX3*RX44+230.40355*P3*RX2*RX43+0.01339885*P4*RX1
     @    8*RX28*RX3)+HT*(0.01339885*P4*RX18*RX3*X1+P1*RX43+P2*RX37)+P3
                                  PN3=P4
                             VAL1(NT-I0)=PN0
                             VAL2(NT-I0)=PN1
                             VAL3(NT-I0)=PN2
                             VAL4(NT-I0)=PN3
                             P1=PN0
                             P2=PN1
                             P3=PN2
                             P4=PN3
      1003                CONTINUE
c-fin-de-faire.
c-faire
                          DO 1004 I0=1,NT1
                             X1=ETAT1(I0)
                             X2=ETAT2(I0)
                             X3=ETAT3(I0)
                             X4=ETAT4(I0)
                             P1=VAL1(I0+1)
                             P2=VAL2(I0+1)
                             P3=VAL3(I0+1)
                             P4=VAL4(I0+1)
                             U1=COM1(I0)
                                   RX1=EXP(-18.197014*X3)
                                   RX2=X1**2
                                   RX3=-1328.6561*RX1*RX2
                                   RX4=-6162.4355*RX1*RX2*U1
                                   RX5=RX3+RX4
                                   RX6=0.08726646*X2
                                   RX7=COS(RX6)
                                   RX8=SIN(RX6)
                                   RX9=1/(0.019115213*X3+1)
                                   RX10=X1**3
                                   RX11=X3**2
                                   RX12=1/(0.038230427*X3+3.6539137e-4*RX11
     1    +1)
```

```
                              RX13=U1**2
                              RX14=-1328.6561*RX1*RX2*U1-4.520514*RX12
     1   *RX8-3081.2178*RX1*RX13*RX2-3339.4087*RX1*RX2
                              RX15=0.038230427*X1*X3+3.6539137e-4*RX11
     1   *X1+X1
                                UNO=0.5*HT**2*(P2*(51.801273*RX7*(0.03
     1   8230427*RX5*X3+3.6539137e-4*RX11*RX5+RX4+RX3)/RX15**2+758121.75
     2   *RX1*RX8*X1/RX15+167707.0*RX1*RX5*U1-738617.25*RX1*RX2*RX8*RX9+
     3   50.46856*RX5*RX7*RX9-7.0313786e8*RX1*RX2*RX8+167707.0*RX1*RX14)
     4   +P1*(-2657.3123*RX1*RX5*U1*X1-12324.871*RX1*RX14*U1*X1-66158.61
     5   *RX1*RX12*RX7*X1-6162.4355*RX1*RX13*RX5*X1-6678.8174*RX1*RX5*X1
     6   -2657.3123*RX1*RX14*X1+2.5836976e7*RX1*RX10*RX8*U1+5570599.0*RX
     7   1*RX10*RX8)+10258.572*P4*RX1*RX2*RX8*RX9-0.7009522*P4*RX5*RX7*R
     8   X9+230.40355*P3*RX5*RX8+3372001.3*P3*RX1*RX2*RX7)+HT*(167707.0*
     9   P2*RX1*X1+P1*RX5)
                           COM1(IO)=U1-RO(NUNTILO)*UNO
 1004                 CONTINUE
c-fin-de-faire.
                   X1=ETAT1(NT)
                   X2=ETAT2(NT)
                   X3=ETAT3(NT)
                   X4=ETAT4(NT)
                   COLD=CNEW
                   CNEW=X4+100*(X3-0.2)**2+20*(X2+1)**2+100*(X1-0.0976)
     1   **2
                   WRITE(8,1009) CNEW
c-fin-liste_until.
c-tests-de-sortie-d'until
              IF(ABS(COLD-CNEW) . LT . EPS)GOTO 1006
              IF(NUNTILO.gt.MAXUNTILO)GOTO 1005
c-reiterer-until
              GOTO 1001
c-sortie-d'until-depassement-du-maximum-d'iterations
 1005         CONTINUE
              WRITE(8,1007)
              IERRO=1
 1006         CONTINUE
c-fin-d'until.
          WRITE(8,1010)
 1007 FORMAT(' MAXUNTILO' )
 1008 FORMAT('COUT_PONTRI:[')
 1009 FORMAT(' ',e12.5,',')
 1010 FORMAT(' []]$')
      RETURN
      END


        PROGRAM M_PONTRIN
        REAL ETAT4(334),ETAT3(334),ETAT2(334),ETAT1(334),VAL4(334),VAL3(
```

```
1    334),VAL2(334),VAL1(334),COM1(334)
     EXTERNAL RO
      LISPFUNCTION FLETAT1 'CLIMAX::FORT-MACS' (REAL(334),CHARACTER*
1    20,INTEGER(1))
      INTEGER NETAT1(1)
      CHARACTER*20 ETAT1F
      LISPFUNCTION FLVAL1 'CLIMAX::FORT-MACS' (REAL(334),CHARACTER*2
1    0,INTEGER(1))
      INTEGER NVAL1(1)
      CHARACTER*20 VAL1F
      LISPFUNCTION FLETAT2 'CLIMAX::FORT-MACS' (REAL(334),CHARACTER*
1    20,INTEGER(1))
      INTEGER NETAT2(1)
      CHARACTER*20 ETAT2F
      LISPFUNCTION FLVAL2 'CLIMAX::FORT-MACS' (REAL(334),CHARACTER*2
1    0,INTEGER(1))
      INTEGER NVAL2(1)
      CHARACTER*20 VAL2F
      LISPFUNCTION FLETAT3 'CLIMAX::FORT-MACS' (REAL(334),CHARACTER*
1    20,INTEGER(1))
      INTEGER NETAT3(1)
      CHARACTER*20 ETAT3F
      LISPFUNCTION FLVAL3 'CLIMAX::FORT-MACS' (REAL(334),CHARACTER*2
1    0,INTEGER(1))
      INTEGER NVAL3(1)
      CHARACTER*20 VAL3F
      LISPFUNCTION FLETAT4 'CLIMAX::FORT-MACS' (REAL(334),CHARACTER*
1    20,INTEGER(1))
      INTEGER NETAT4(1)
      CHARACTER*20 ETAT4F
      LISPFUNCTION FLVAL4 'CLIMAX::FORT-MACS' (REAL(334),CHARACTER*2
1    0,INTEGER(1))
      INTEGER NVAL4(1)
      CHARACTER*20 VAL4F
      LISPFUNCTION FLCOM1 'CLIMAX::FORT-MACS' (REAL(334),CHARACTER*2
1    0,INTEGER(1))
      INTEGER NCOM1(1)
      CHARACTER*20 COM1F
      ETAT1(1)=1.0
      ETAT2(1)=-0.2
      ETAT3(1)=1.0
      ETAT4(1)=0.0
      NETAT1(1)=334
      ETAT1F= 'ETAT1'
      NVAL1(1)=334
      VAL1F= 'VAL1'
      NETAT2(1)=334
      ETAT2F= 'ETAT2'
```

```
      NVAL2(1)=334
      VAL2F= 'VAL2'
      NETAT3(1)=334
      ETAT3F= 'ETAT3'
      NVAL3(1)=334
      VAL3F= 'VAL3'
      NETAT4(1)=334
      ETAT4F= 'ETAT4'
      NVAL4(1)=334
      VAL4F= 'VAL4'
      NCOM1(1)=334
      COM1F= 'COM1'
      CALL PONTRIAGN(ETAT1,ETAT2,ETAT3,ETAT4,VAL1,VAL2,VAL3,VAL4,COM
     1   1,0.003,334,COUTN,RO,9.0e-6,IER,100)
      CALL FLETAT1(ETAT1,ETAT1F,NETAT1)
      CALL FLVAL1(VAL1,VAL1F,NVAL1)
      CALL FLETAT2(ETAT2,ETAT2F,NETAT2)
      CALL FLVAL2(VAL2,VAL2F,NVAL2)
      CALL FLETAT3(ETAT3,ETAT3F,NETAT3)
      CALL FLVAL3(VAL3,VAL3F,NVAL3)
      CALL FLETAT4(ETAT4,ETAT4F,NETAT4)
      CALL FLVAL4(VAL4,VAL4F,NVAL4)
      CALL FLCOM1(COM1,COM1F,NCOM1)
    STOP
    END
    REAL FUNCTION RO(N)
    INTEGER N
    RO=10./(100.+40.*n)
    END
```

References

[1] ALEXEEV V., TIKHOMIROV V., FOMINE V. (1982). Commande optimale. Edition MIR.

[2] ATHANS M., FALB P.L. (1966). Optimal control. Mc Graw Hill.

[3] BELLMAN R. (1971). Introduction to the mathematical theory of control processes. Academic press.

[4] BENSOUSSAN A., LIONS J.L.(1978). Applications des inéquations variationnelles en contrôle stochastique. Dunod.

[5] EKELAND I., TEMAM R. (1974). Analyse Convexe et problèmes variationnels. Dunod.

[6] FLEMING W.H., RISHEL R. (1975). Deterministic and Stochastic Optimal Control. Springer Verlag, New York.

[7] JACOBSON D.H., MAYNE D.O. (1970). Differential dynamic programming. Elsevier.

[8] LEE E.B., MARKUS L. (1967). Fundations of optimal control theory. J.Wiley.

[9] LIONS J.L. (1966). Contrôle optimal des systèmes gouvernés par des équations aux dérivées partielles. Dunod.

[10] PALLU DE LA BARRIERE R. (1966). Cours d'automatique théorique. Dunod

[11] LIONS P.L. (1982). Generalized Solutions of Hamilton-Jacobi Equations. Research Notes in Mathematics, 69 , Pitman.

[12] PONTRYAGIN L.S., BOLTYANSKII V.GR, GAMKRELIDZE V., MISCHENKO E.F. (1964). The mathematical theory of optimal processes. Pergamon Press. Edition Mir en français 1974.

[13] YOUNG L.C. (1969). Lectures on the calculus of variations and optimal control. Saunders.

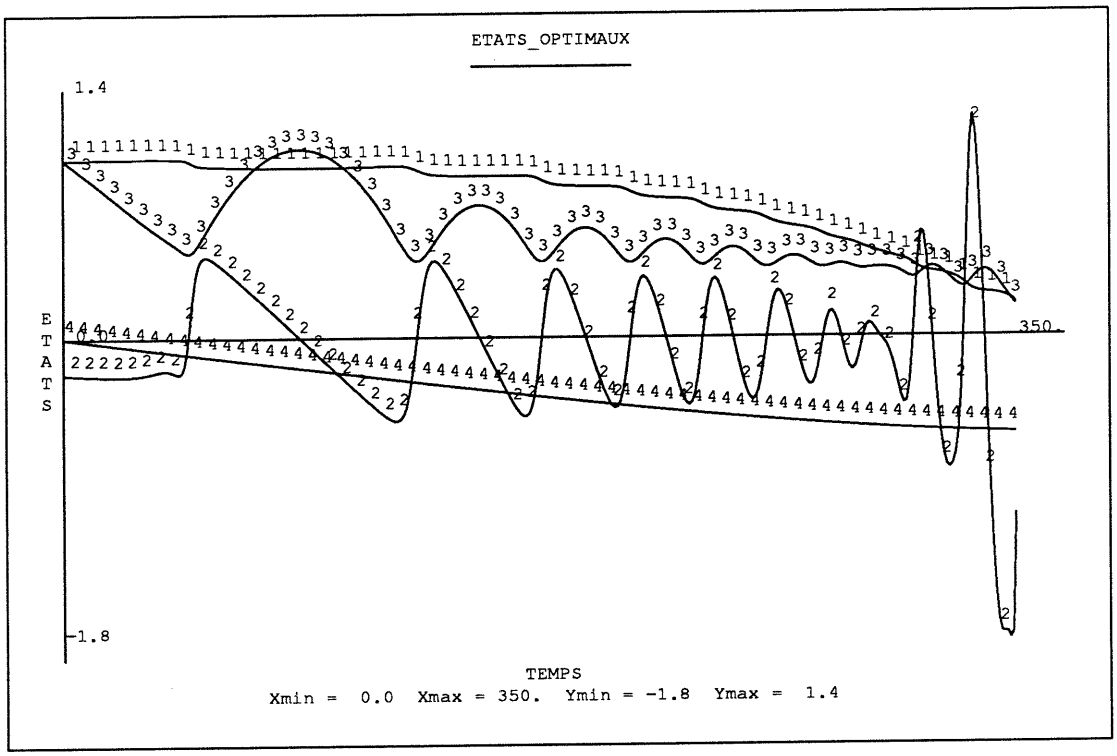


Figure 1: Trajectoire optimale déterministe. La courbe annotée par 4 représente le coût.

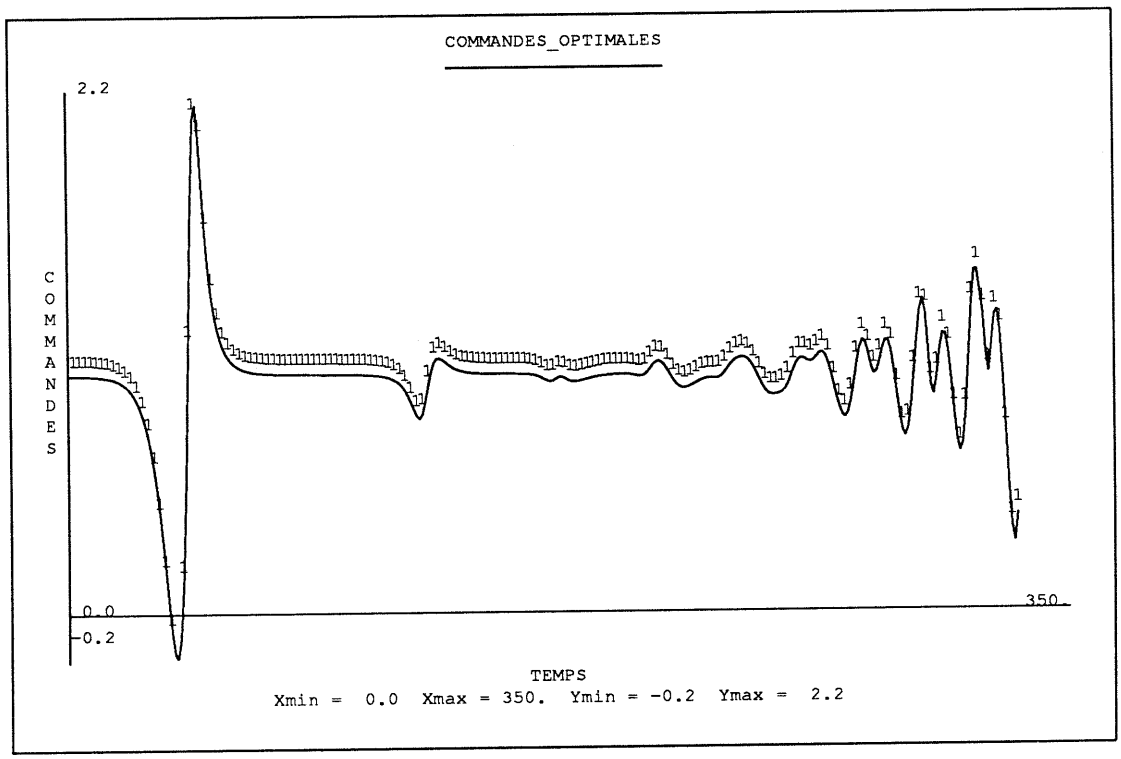


Figure 2: Commandes optimales.

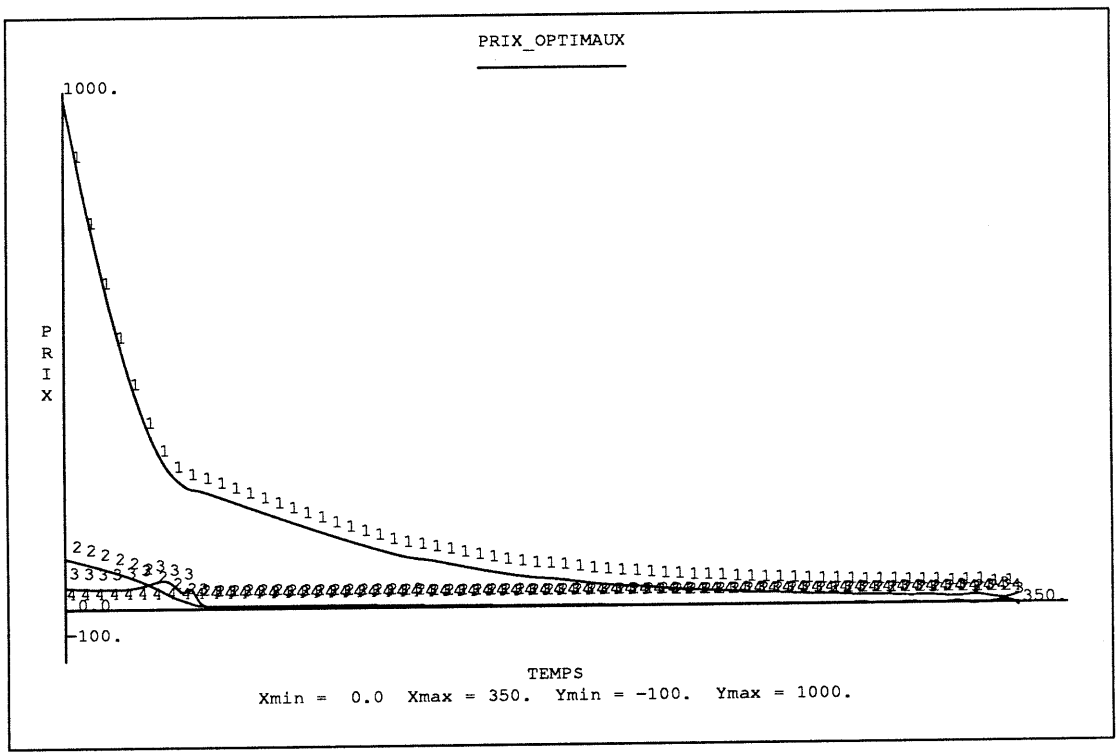


Figure 3: Variables duales optimales.

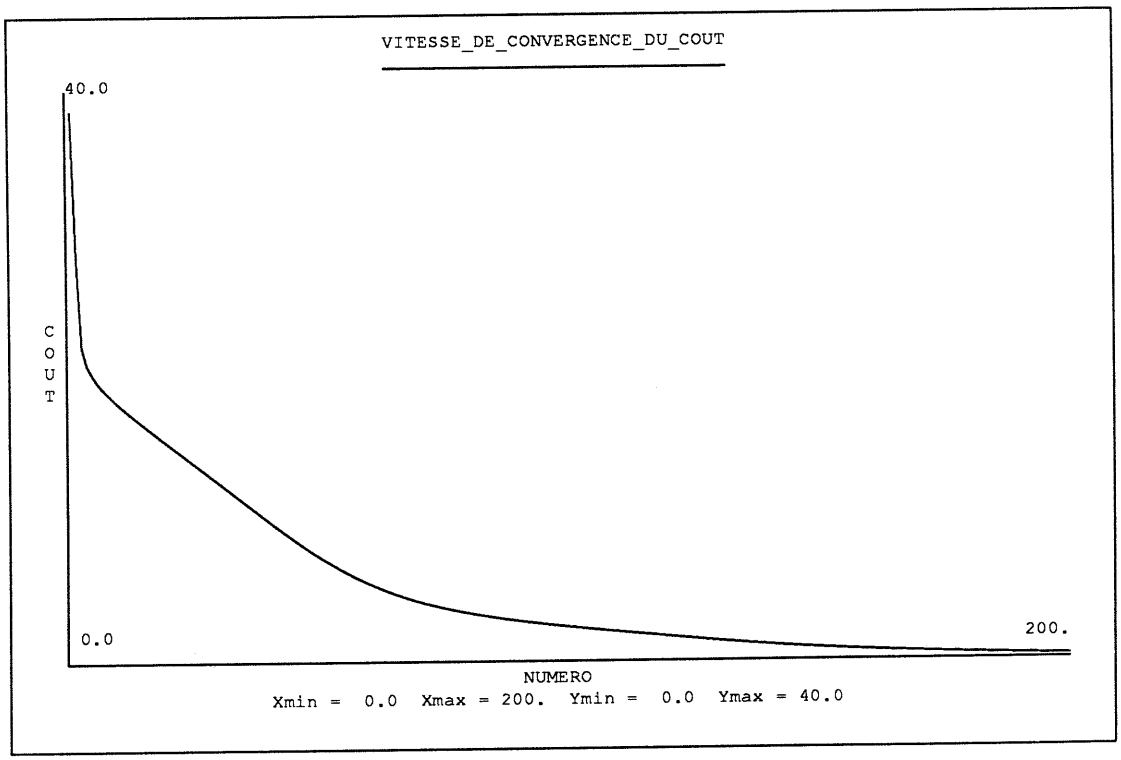


Figure 4: Convergence du coût en fonction du numéro d'itération.