ISSN 0221-5225
BIGRE N° 70
Septembre 1990

BIGRE

Journées AFCET-GROPLAN

NICE, 24-25 janvier1990

LES AVANCEES EN PROGRAMMATION

CERISI GRECO-PRC PAOIA IRIT

PANDORE: Un système expert pour l'optimisation des systèmes dynamiques

J.P. Quadrat, A. Sulem, J.P. Chancelier, C. Gomez
INRIA
Domaine de Voluceau Rocquencourt BP105 78153 Le Chesnay Cedex France

Résumé

Le but du système *Pandore* est d'automatiser l'ensemble des tâches à accomplir pour faire une étude d'optimisation de systèmes dynamiques, utilisant des techniques de calcul formel, d'inférence, de manipulations symboliques et d'analyse numérique. Le système est écrit en *Macsyma*, *Lisp* et *Prolog* et tourne sur une machine Lisp Symbolics. Deux langages : MACROFORT et MACROTEX ont été développés permettant la génération de programmes *Fortran* et LaTeX à partir de *Macsyma*.
L'utilisateur spécifie son problème sous forme symbolique au moyen d'un éditeur spécialisé. Un langage de commandes permet d'interroger le système et de modifier la base de données du problème.
Lorsque le problème est bien posé, une méthode de résolution est choisie par le système. Les critères de choix des méthodes de résolution sont codés sous forme de clauses *Prolog*. Une étude théorique de l'existence et l'unicité de solutions d'équations aux dérivées partielles associées à ces problèmes est menée dans certains cas. *Pandore* génère ensuite les programmes *Fortran* pour l'étude numérique, puis un rapport en LaTeX comprenant le modèle, l'explication de la méthode de résolution choisie, l'étude numérique, les graphes des résultats.

1 Introduction

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. Le but est de coder l'état de l'art dans ce domaine et d'automatiser l'ensemble des tâches à accomplir pour réaliser une étude complète à partir de la spécification du problème de commande. Cette étude consiste à choisir une méthode de résolution, générer les programmes numériques associés, et écrire un rapport en LaTeX résumant l'ensemble de l'étude. Pour atteindre ces objectifs, *Pandore* a été développé 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 [15], et de calcul formel par *Macsyma*[12].
Macsyma est un logiciel interactif capable de faire du calcul numérique et symbolique. Il permet d'automatiser une couche d'activités en amont du calcul numérique lui-même faites le plus souvent manuellement : manipulations de formules algébriques ou de programmes vus comme des formules.
Prolog est utilisé pour coder la connaissance sur la théorie du contrôle et pour l'organisation logique du système.
On a rajouté à cet environnement les développements suivants:

- définition et écriture d'un compilateur permettant la génération de *Fortran* à partir de *Macsyma* appelé *MACROFORT*.
- définition et écriture d'un autre compilateur permettant la génération de LaTeX à partir de *Macsyma* appelé MACROTEX .

Pandore comprend quatre parties :

- un éditeur spécialisé d'entrée du problème,
- un langage de commandes 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'éditeur d'entrée du problème

2.1 Les problèmes à résoudre

Sans détailler les aspects mathématiques mais afin de comprendre les fonctionnalités de *Pandore*, énonçons un problème typique résoluble par *Pandore* [1,2,3,5,6]:
On considère un système dynamique dont l'état X_t est modélisé par l'équation différentielle stochastique suivante

$$dX_t = b(X_t, U_t)dt + \sigma(X_t)dW_t$$

où U_t est le contrôle, b la moyenne de la dynamique, $\sigma^2 t$ la variance.
Le problème est d'optimiser, par rapport aux variables de contrôle, un critère de la forme

$$E[\int_{to}^{T} c(X_t, U_t)\, dt + f(X_T, U_T)]$$

où $c(X_t, U_t)$ est appelé le coût instantané et $f(X_T, U_T)$ le coût d'arrêt.
L'utilisateur devra préciser à la machine quelles sont les variables d'état et de contrôle, leurs domaines de variation, la dynamique du système en explicitant les expressions de b et h, c et f, l'horizon, les conditions aux limites, les paramètres ...
Pour cela, il utilise l'éditeur décrit plus bas.

2.2 L'éditeur de Pandore

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. 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.
Les faits obtenus après la rentrée des données d'un exemple de problème de gestion de portefeuille sont de la forme

```
 (<-- (etat  !0   !x1 ))
 (<-- (etat  !0   !x2 ))
 (<-- (commande  !0   !u1 ))
 (<-- (condition-frontiere !0 !x1 0
!reflechi -1.45))
 (<-- (moyenne-dynamique !0 !x1 x2-u3+u2))
 (<-- (variance-dynamique !0 !x1 0.1))
 (<-- (sens-physique !0 !x2 !"le compte en
actions"))
```

La syntaxe du *Prolog Oblogis* utilisé est :
(← (conclusion)(hyp 1)(hyp 2) ...)
Les constantes sont précédées du caractère "!".

3 Le langage de commandes 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. Ce langage de commandes est hiérarchisé. Certaines commandes sont utilisées pour la gestion de la base de faits, elles permettent d'installer, de modifier ou de visualiser la base de faits. D'autres commandes permettent d'activer des modules de *Pandore*, pour vérifier la complétude d'un problème, faire générer du code *Fortran*, l'exécuter, faire des sorties graphiques...

4 Expertise

4.1 Démonstration théorique

Pandore est capable d'étudier l'existence et l'unicité de solutions de certaines équations aux dérivées partielles apparaissant dans les problèmes de contrôle. Les théorèmes d'analyse fonctionnelle utilisés sont codés sous forme de clauses *Prolog*. De nouveaux objets *Macsyma*, comme les espaces de Sobolev, ont été crées et les autres objets mathématiques utilisés, comme les opérateurs différentiels ou les formulations variationnelles, sont représentés par des objets *Lisp* structurés [4].

4.2 Choix automatique d'une méthode de résolution

Lorsque le problème est formulé par l'utilisateur, *Pandore* vérifie la cohérence et la complétude des données et choisit une méthode de résolution parmi les différentes méthodes implémentées :

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

Ces méthodes sont expliqués dans [1,2,3,5,6]. Les critères de choix des méthodes sont codés sous forme de clauses *Prolog*. Une fois les méthodes choisies, on peut demander au système de résoudre numériquement par l'une des méthodes possibles.

5 Génération de programmes

L'appel de commandes appropriées permettent de générer les sous programmes *Fortran* correspondant à la méthode de résolution choisie puis de générer un programme principal faisant un essai numérique particulier selon la précision demandée par l'utilisateur.
Les résultats numériques sont alors disponibles dans *Pandore*. On peut les visualiser en lançant la commande permettant de faire du graphique en deux et trois dimensions.
Détaillons les étapes de la procédure de génération des sous-programmes qui sont résumées dans le tableau ci-dessous.
Une grammaire permet de spécifier complètement les problèmes de contrôle stochastique et les algorithmes de résolution. Quand le problème est bien posé, une phrase syntaxiquement correcte par rapport à cette grammaire, est générée grâce à des règles *Prolog*. Cette phrase est une liste *Macsyma* contenant dans un certain ordre, les données du problème et les algorithmes de résolution.
Un compilateur écrit en *Macsyma*, traduit cette phrase en un programme *Fortran*. Le choix d'environnement de compilation (*Macsyma*), permet de faire du prétraitement de façon puissante (calcul de gradient, inversion formelle de petits systèmes linéaires, simplification de formules...) Dans un premier temps, il génère une liste *Macsyma* décrivant le programme *Fortran* au niveau *Macsyma*, puis traduit cette liste en *Fortran*. Cette liste *Macsyma* est une phrase d'un certain language appelé *Macrofort* implémenté comme une interface entre *Macsyma* et *Fortran* et qui permet une programmation structurée.

```
[Problèmes]
     ↓                       Editeur (Lisp)
[Faits Prolog]
     ↓                       Contrôleur (Prolog)
[Appel à un générateur]
     ↓                       Générateur (Macsyma)
[Programme Macrofort]
     ↓                       Evaluation
[Programme Fortran]
```

6 Macrofort

Macrofort consiste en un ensemble de facilités rajouté à *Macsyma* pour générer du code *Fortran*. *Macrofort* 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 [16]:

- **les instructions élémentaires** qui sont des fonctions *Macsyma* dont le rôle est d'imprimer une instruction *Fortran* paramétrée.
- **les macro instructions** qui sont des fonctions *Macsyma* dont le rôle est de générer plusieurs instructions élémentaires de façon à augmenter la puissance d'expression de *Macrofort*.
- **les instructions de gestion de piles spécialisées** qui permettent de mettre à jour des piles et de construire des programmes en ajoutant ou retranchant des instructions.

On écrit un programme *Macrofort* 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.

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 *vs*. En sortie nous obtenons le programme d'optimisation correspondant.

Le générateur écrit en Macsyma :

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

aplat(
 subroutinem(gradient,[ro,eps],
  untilm(e<eps,[equalf,e,100],
   [prog1,
    [equalf,e,sum(
     (concat(vs[i],n)-vs[i])∧ 2,i,1,d)],
    prog2,
    [writem,8,vs,[concat(d,f12\.5)]]]
))))$
```

On voit que cette fonction se divise en deux parties:

- le prétraitement algébrique
  - initialisation des variables x et y (initm)
  - définition de xn et yn (prog1m)
  - définition de x et y (prog2m)

– la construction du programme
  * construction du sous-programme
  * génération de code (appel du traducteur aplat)

L'appel du générateur se fait par gradient-const-step (x*x + y*y,[x,y]).

Le programme Fortran généré mis dans un fichier est alors

```
  SUBROUTINE gradient(ro,eps,ierr,maxuntil0)
   x=0
   y=0
   ierr=0
c-until e < eps faire liste-until
c-initialisation
    nuntil0=0
    e=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
      e=(xn-x)**2+(yn-y)**2
      x=xn
      y=yn
      WRITE(8,1003) x,y
c-fin-liste-until
c-tests-de-sortie-d'until
    IF (e.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
```

7 Génération de rapport

Elle se fait de façon analogue à la génération de programmes.
Le module de génération de rapports "collecte" les résultats des autres modules et génère un rapport en syntaxe LaTeX utilisant pour cela le langage MACROTEX permettant de générer facilement du LaTeX à partir de *Macsyma*.
On dispose d'un ensemble de phrases variables — stockées sous forme 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 [8].
Donnons un exemple. Le paragraphe :

"Le coût optimal V satisfait l'équation

$$-V(x_1,x_2) + \min_{u_1,u_2}\left(u_1\frac{\partial V}{\partial x_1}+u_2\frac{\partial V}{\partial x_2}+u_2^2\right) + \frac{\partial^2 V}{\partial x_1^2}+\frac{\partial^2 V}{\partial x_2^2}-x_1\frac{\partial V}{\partial x_1}+x_1^2=0''$$

est obtenu par l'évaluation de:

[[formatt, "le coût optimal ~ N satisfait l'équation", V],
[tex, -V(x1,x2)+min[u1,u2](u1*diff(V,x1)
+u2*diff(V,x2**2)+diff(V,x1,2)+diff(V,x2,2)
-x1*diff(V,x1)+x1**2=0]]

Cette liste a été construite au moyen du programme *Prolog* suivant où *formatt* et *tex* sont des instructions du langage MACROTEX . *formatt* est analogue à la fonction *Lisp format* et *tex* traduit en syntaxe LaTeX une expression *Macsyma*.

(←	(val x x))	identité
(←	(satisfait "le coût optimal ~ N satisfait : "))	format
(←	(équation l)	conclusion
	(etat x) (controle u) (actualisation d) (derive b) (diffusion a) (cout c) (horizon !infini) (cout optimal v)	données
	(satisfait q)	recherche du format
	(val eq { hjb(d,v,b,a,c,u,x)})	appel Macsyma
	(val l { [[formatt,q,v],[tex,eq]]}))	construction paragraphe

où *hjb* est une fonction *Macsyma* définie par:
hjb(d,v,b,a,c,u,x) :=-d*apply (v,x) +
min[u](c+sum(b[i]*diff(v,x[i])+
a[i]*diff(v,x[i],2),i,1,length(b)))) = 0

8 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* [17,18,19].
Un utilisateur peut ainsi faire du calcul algébrique dans *Macsyma* et générer du code LaTeX sans avoir à connaitre la syntaxe LaTeX . Nous avons donc écrit un générateur de code LaTeX en *Common Lisp* que nous avons appelé MACROTEX . Ce programme tourne sur Machine Symbolics Lisp, release-7 en *Common Lisp* et sur Sun en *Franz Lisp.*
MACROTEX comprend un ensemble d'instructions pour la traduction d'expressions mathématiques *Macsyma* en syntaxe LaTeX (en particulier une instruction permettant de découper les grosses formules automatiquement). De plus, des facilités sont rajoutées pour générer un document LaTeX au niveau *Macsyma*.

9 Exemple d'utilisation de MACROTEX

L'exemple suivant montre l'utilisation de MACROTEX pour générer un texte.
On définit la fonction *Macsyma* :

```
control(n1,v,df,dr):=block(
 [DECOUPEM, A.v = factorise(df,dr,n1,v)],
 [FORMATT, "La matrice est:~K",
 matrice(df,dr,n1,v)]]]))$
```

Cette fonction utilise le programme *Macsyma* suivant:

```
/* liste des variables d'etat */
xx:makelist(\x[i],i,1,n1)$

/* discretisation des derivees */
d1(i,v):=(s[i].v-S[i]^^(-1).v)/(2*\h[i])$
d2(i,v):=(s[i]^^(-1).v+s[i].v-2*v)/\h[i]^2$

/* calcul de l'hamiltonien discret */
discret(a,b,n,v):= expand(sum(a[i]*d2(i,v)+
                     b[i]*d1(i,v),i,1,n))$

/*calcul de l\'equation */
factorise(a,b,n,v):= apply("+", map(lambda(
    [x],coeff(discret(a,b,n,v),x).x),
       append([v],makelist(s[i].v,i,1,n),
         makelist(s[i]^^(-1).v,i,1,n))))$

/*calcul des lignes de la matrice */
lign(1,i,s1,s2,n,v):=
  [xx, subst(xx[i]+s1,xx[i],xx),
    ratsimp(coeff(1,s2)/-coeff(1,v))]$

/*calcul de la matrice*/
matrice(a,b,n,v):= apply(matrix, append(
  makelist(lign(discret(a,b,n,v),
   i,\h[i],s[i].v,n,v),i,1,n),
  makelist(lign(discret(a,b,n,v),
   i,-\h[i],s[i]^^(-1).v,n,v),i,1,n)))$
```

L'appel de la fonction *control*(2,v,[1,1],[u[1],u[2]]) génère le texte LaTeX ci-dessous.

$$\begin{aligned}
AV &= E5 + E4 + E3 + E2 + E1 \\
E1 &= \left(\frac{0.5u_2}{h_2} + h_2^{-2}\right) S_2 V \\
E2 &= \left(h_2^{-2} - \frac{0.5u_2}{h_2}\right) S_2^{-1} V \\
E3 &= \left(-2h_1^{-2} - 2h_2^{-2}\right) V \\
E4 &= \left(\frac{0.5u_1}{h_1} + h_1^{-2}\right) S_1 V \\
E5 &= \left(h_1^{-2} - \frac{0.5u_1}{h_1}\right) S_1^{-1} V
\end{aligned}$$

La matrice est:

$$\begin{pmatrix}
[x_1, x_2] & [x_1 + h_1, x_2] & \dfrac{(h_1u_1+2)h_2^2}{(4h_2^2+4h_1^2)} \\
[x_1, x_2] & [x_1, x_2 + h_2] & \dfrac{\left(h_1^2h_2u_2+2h_1^2\right)}{(4h_2^2+4h_1^2)} \\
[x_1, x_2] & [x_1 - h_1, x_2] & -\dfrac{(h_1u_1-2)h_2^2}{(4h_2^2+4h_1^2)} \\
[x_1, x_2] & [x_1, x_2 - h_2] & -\dfrac{\left(h_1^2h_2u_2-2h_1^2\right)}{(4h_2^2+4h_1^2)}
\end{pmatrix}$$

10 Exemple d'utilisation de Pandore

Nous donnons dans l'annexe A le rapport généré pour la résolution d'un problème de gestion de portefeuille. Le titre du rapport est soit précisé par l'utilisateur, soit généré sous la forme d'un titre général par *Pandore*. Le rapport comprend un résumé du problème, le modèle et les notations, la méthode de résolution, les résultats numériques sous forme de graphes, les références et les programmes *Fortran* en annexe.

Pour générer ce rapport il aura fallu lancer la commande *generer le rapport* après avoir résolu numériquement le problème.

11 Conclusion

La puissance de *Pandore* vient de l'intégration de *Lisp*, *Macsyma*, *Prolog* utilisés comme un ensemble de fonctions au même niveau de programmation. *Pandore* offre à la fois des traitements spécifiques pour les problèmes de contrôle et des outils généraux comme *Macrofort* et MACROTEX .

Des utilisations de *Pandore* dans divers domaines ont été faites : optimisation de la trajectoire de rentrée de la navette Hermès dans l'atmosphère (modèle plan simplifié), gestion de barrages hydroélectriques pour E.D.F, gestion de portefeuilles...
De plus, *Pandore* est interfacé avec un système de C.A.O. pour automaticiens : BASILE [20]. C'est un système interactif spécialisé au calcul numérique pour l'Automatique classique. Il comprend un interprète du langage Basile gérant une bibliothèque scientifique. *Pandore* peut utiliser la bibliothèque de Basile et générer du code *Fortran* pour Basile.

Un travail analogue pour l'étude automatique de problèmes de traitement du signal et de filtrage non linéaire est mené à l'Université de Maryland, Electrical Engineering Department par G. Blankenship et al. [9,10].

Références

[1] THEOSYS (1984) Commande optimale de Système Stochastique *R.A.I.R.O. Automatique/ Systems Analysis and Control*, 18(2):225-250.

[2] GOMEZ C., QUADRAT J.P., SULEM A.(1984) Towards an Expert System in Stochastic Control : the Hamilton-Jacobi equation part, Volume 63 of *Lecture Notes in Control and Information Sciences*, Springer Verlag. Proceedings of the Sixth International Conference on Analysis and Optimization of Systems, Nice. Juin.

[3] GOMEZ C., QUADRAT J.P., SULEM A. (1984) Towards an Expert System in Stochastic Control : Optimization in the class of Local Feedback. Volume 1119 of *Lecture Notes in Mathematics*, Springer-Verlag. Proceedings of the Conference on Stochastic Control, Roma, March.

[4] GOMEZ C., QUADRAT J.P., SULEM A. (1985). Computer Algebra as a tool for solving optimal control problems. *Applications of Computer Algebra*. Kluwer Academic Publishers.

[5] CHANCELIER J.Ph., GOMEZ C., QUADRAT J.P., SULEM A.(1985) Un système expert pour l'optimisation de systèmes dynamiques, *Proceedings du septième colloque international sur les méthodes de calcul scientifique et technique, Versailles, Decembre).* North Holland

[6] CHANCELIER J.Ph., GOMEZ C., QUADRAT J.P., SULEM A.(1987) Automatic study in stochastic control. Volume 10 of *IMA Volumes in Mathematics and its Applications*, Springer Verlag. Proceedings of the Workshop on Stochastic Differential Systems, Stochastic Control Theory and Applications, Minneapolis, Minnesota, June 86.

[7] CHANCELIER J.Ph., GOMEZ C., QUADRAT J.P., SULEM A.(1986) Pandore. *Advanced Computing Concepts and Techniques in Control Engineering.* NATO ASI Series. Series F: Computer and Systems Sciences, Vol. 47. Edited by M.J. Denham and A.J. Laub

[8] CHANCELIER J.Ph., GOMEZ C., QUADRAT J.P., SULEM A.(1989) An Expert System for stochastic control problems : Automatic report generation. *Computer Science in Economics and Management 2 65-82.* Kluwer Academic Publishers.

[9] BLANKENSHIP G., GOMEZ C., QUADRAT J.P., SULEM A., YAN I. (1984) An Expert System for Stochastic Control and non-linear filtering In *Proceedings of 23rd IEEE Conference on Decision and Control, Las Vegas, NV*

[10] BLANKENSHIP G., CHANCELIER J.Ph., GOMEZ C., QUADRAT J.P., SULEM A.(1985) An Expert System for Stochastic Control and signal processing, *MTNS conference, Stockholm.*

[11] QUADRAT J.P. (1987) Génération automatique de programmes numériques en Contrôle Stochastique. *Calcul Formel pour l'Automatique.* Edité par Chenin.

[12] MACSYMA Reference Manual. Version 12 (1986). Symbolics, Inc.

[13] STEELE G. COMMON LISP. The language. *Digital Press.*

[14] LAMPORT L. LATEX . A Document Preparation System. *Addison-Wesley Publishing Company.*

[15] GLOESS P. (1984) Logis user's manual. *Université de Compiègne*

[16] CHANCELIER J.P., GOMEZ C., QUADRAT J.P. (October 1987) MACROFORT : A Fortran code generator in Macsyma. *MACSYMA Newsletter.*

[17] CHANCELIER J.Ph., SULEM A. (July 1988) MACROTEX : A LATEX code generator in Macsyma. *MACSYMA Newsletter.*

[18] CHANCELIER J.Ph., SULEM A. (Décembre 1987) MACROTEX : A LATEX code generator in Macsyma. *Rapport technique INRIA 93.*

[19] CHANCELIER J.Ph., SULEM A. (Octobre 1989) MACROTEX : Un générateur de code LATEX implémenté en MACSYMA. *cahiers Gutemberg 3.*

[20] DELEBECQUE F., KLIMANN C., STEER S.(1989) Basile. Guide de l'utilisateur. Version 3.0. Inria.

12 Annexe A : Exemple de rapport généré automatiquement

Gestion de portefeuille avec coûts de transaction

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

Résumé

On considère un problème de gestion de portefeuille avec coûts de transaction. Il s'agit de minimiser l'espérance mathématique d'un critère actualisé comprenant un coût intégral et des coûts de réflexion. Le coût optimal satisfait une équation de Bellman obtenue par la méthode de la programmation dynamique. Cette équation est résolue numériquement.

1 Notations

- Variables d'état : $X1, X2$
  $X1$: le compte à intérêt fixe
  $X2$: le compte en actions
- Variables de commande : $U1, U2, U3$
  $U1$: la consommation
  $U2$: le montant d'actions achetées
  $U3$: le montant d'actions vendues
- Coût optimal : V
  V : l'opposé du maximum de la fonction utilité
- Temps : t
- Dimension de l'état : n
- I-ème variable d'état : x_i
- Opérateur dérivée par rapport à x_i : ∂_i

2 Equation d'évolution du système:

On considère le processus de diffusion controlé défini par l'équation différentielle stochastique:
Evolution du compte à intérêt fixe

$$dX1_t = dZ^0_{1_t} - dZ^1_{1_t} + (0.2X1 + 0.9U3 - U2 - U1)dt + 0.45dW_{1_t} \quad (1)$$

Evolution du compte en actions

$$dX2_t = dZ^0_{2_t} - dZ^1_{2_t} + X2dW_{2_t} + (X2 - U3 + U2)dt \quad (2)$$

où

- $X1 \in [0,1]$
- $X2 \in [0,1]$
- $U1 \in [0.10, 1]$
- $U2 \in [0,1]$
- $U3 \in [0,1]$

$Z^j_{i\,t}$ représente un processus croissant, strictement croissant lorsque x_{i_t} est sur la frontière $x_i = j$.

- $X1_t$ est réfléchie sur la frontière $X1 = 0$.
- $X2_t$ est réfléchie sur la frontière $X2 = 0$.
- $X1_t$ est réfléchie sur la frontière $X1 = 1$.
- $X2_t$ est réfléchie sur la frontière $X2 = 1$.

W_{i_t} désigne un processus de Wiener, i.e. un processus continu gaussien à accroissements indépendants.

Ce processus de diffusion est bien défini [8] . Il est la limite lorsque le pas en temps h tends vers 0 d'un processus discret markovien X^h_n vérifiant:

$$E(X^h_{n+1} - X^h_n \mid \mathcal{F}_n) = h \begin{pmatrix} 0.2X1 + 0.9U3 - U2 - U1 \\ X2 - U3 + U2 \end{pmatrix} + o(h)$$

$$E\left(\left(X^h_{n+1} - X^h_n\right)^{\otimes 2} \mid \mathcal{F}_n \right) = 2h \begin{pmatrix} 0.1 & 0 \\ 0 & 0.5X2^2 \end{pmatrix} + o(h)$$

et une condition d'uniforme intégrabilité de l'accroissement $X^h_{n+1} - X^h_n$.
$\mathcal{F}_n$ représente la σ-algèbre générée par X_0, X_1 ... X_n .

3 Critère à optimiser

Il s'agit de minimiser la fonctionnelle:

$$\begin{aligned} J(S) &= e_1+e_5+e_4+e_3+e_2 \\ e_2 &= \int_0^{+\infty} e^{-0.3t}\, dZ^1_{2\,t} \\ e_3 &= -2\int_0^{+\infty} e^{-0.3t}\, dZ^0_{2\,t} \\ e_4 &= \int_0^{+\infty} e^{-0.3t}\, dZ^1_{1\,t} \\ e_5 &= -1.45\int_0^{+\infty} e^{-0.3t}\, dZ^0_{1\,t} \\ e_1 &= \int_0^{+\infty} e^{-0.3t}\left(-2.5U1^{0.4}\right)_t dt \end{aligned}$$

dans la classe des feedbacks, i.e. des applications S: $[X1, X2] \mapsto [U1, U2, U3]$.

4 Conditions d'optimalité

On définit la fonction de Bellman V par:

$V(y_1, y_2) = \min_S \left(E\left[J(S) \,|\, X1_0 = y_1, X2_0 = y_2 \right] \right)$

V satisfait l'équation de la Programmation Dynamique [2,1]

$$\min_{U1,U2,U3}(A(U1,U2,U3)V + C(U1)) - 0.3V = 0 \tag{3}$$

$$\partial_1 V(0, X2) = -1.45$$

$$\partial_2 V(X1, 0) = -2$$

$$\partial_1 V(1, X2) = -1$$

$$\partial_2 V(X1, 1) = -1$$

avec

$$\begin{aligned} A(U1,U2,U3)V &= e_9+e_8+e_7+e_6 \\ e_6 &= 0.5X2^2\partial_2^2 V \\ e_7 &= 0.1\partial_1^2 V \\ e_8 &= \partial_2 V(X2-U3+U2) \\ e_9 &= \partial_1 V(0.2X1+0.9U3-U2-U1) \end{aligned}$$

$$C(U1) = -2.5U1^{0.40} \tag{4}$$

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

D'après P.L.Lions [5] , on sait qu'il existe une solution de l'équation de Bellman; la démonstration est basée sur le Principe du Maximum.

Les méthodes de discrétisation exposées plus bas sont elles aussi basées sur le Principe du Maximum et son interprétation probabiliste.

6 Méthode de la programmation dynamique

On se propose de résoudre l'équation de Bellman (3) après discrétisation. Ceci est possible car la dimension de l'état du système est petite.[6,7,3,4]

6.1 Discrétisation

On note h_i le pas de discrétisation pour la i-ème coordonnée d'espace.

On définit les opérateurs:

$S_i \;:\; V(x_1,\ldots,x_i,\ldots,x_n) \;\mapsto\; V(x_1,\ldots,x_i+h_i,\ldots,x_n) \quad i=1,\ldots,n$

$$\partial_i^{h+} = \frac{(S_i-1)}{h_i}$$

$$\partial_i^{h} = \frac{0.5(S_i-S_i^{-1})}{h_i}$$

$$\Delta_i^{h} = \frac{(S_i^{-1}+S_i-2)}{h_i^2}$$

On approxime alors:

$\partial_i^2 V$ par $\Delta_i^h V$

$\partial_1 V$ par $\partial_1^h V$

$\partial_2 V$ par $\partial_2^h V$

L'équation de Bellman discrétisée s'écrit:

$$\min_{U1,U2,U3}\left(C(U1) + A^h(U1,U2,U3)V^h\right) - 0.30V^h = 0$$

avec

$$\begin{aligned} A^h(U1,U2,U3)V &= e_{13}+e_{12}+e_{11}+e_{10} \\ e_{10} &= \partial_2^h V(X2-U3+U2) \\ e_{11} &= 0.5\Delta_2^h V(X2+0.4)^2 \\ e_{12} &= \partial_1^h V(0.2X1+0.9U3-U2-U1) \\ e_{13} &= 0.1\Delta_1^h V \end{aligned}$$

6.2 Interprétation probabiliste de l'équation de Bellman discrétisée:

La discrétisation de l'équation de Bellman:

$$\min_{U1,U2,U3}(A(U1,U2,U3)V + C(U1,U2,U3)) - \lambda V = 0$$

s'interprète comme un problème de contrôle de chaine de Markov avec taux d'actualisation λk et coût instantané kC. Le coût correspondant est

$$\sum_{n=0}^{\infty} k(\lambda k+1)^{-1-n} C(X_n, U_n)$$

et la matrice de Markov M s'écrit:

$$M = kA + I$$

où I désigne la matrice identité, λ le taux d'actualisation et k l'inverse du maximum de la diagonale de A.
M est définie par:

$$\left(\begin{array}{ccc}
Pt-init & Pt-final & Probabilit\acute{e}-transition \\
[X1,X2] & [X1,X2] & 0 \\
[X1,X2] & [X1+h_1,X2] & \dfrac{0.2\left(X1+h_1^{-2}\right)+0.9U3-U2-U1}{2h_1h_2^{-2}X2^2+0.4h_1^{-1}} \\
[X1,X2] & [X1,X2+h_2] & \dfrac{X2^2+(X2-U3+U2)h_2}{2X2^2+0.4h_1^{-2}h_2^2} \\
[X1,X2] & [X1-h_1,X2] & \dfrac{-0.2\left(X1+h_1^{-2}\right)-0.9U3+U2+U1}{2h_1h_2^{-2}X2^2+0.4h_1^{-1}} \\
[X1,X2] & [X1,X2-h_2] & \dfrac{X2^2-(X2-U3+U2)h_2}{2X2^2+0.4h_1^{-2}h_2^2} \\
[0,X2] & [0,X2] & 0 \\
[0,X2] & [h_1,X2] & \dfrac{0.5h_2^2}{h_1^{-2}X2^2+0.5h_2^2} \\
[1,X2] & [1,X2] & 0 \\
[1,X2] & [1-h_1,X2] & \dfrac{0.5h_2^2}{\left(h_1^{-2}X2^2+0.5h_2^2\right)} \\
[X1,0] & [X1,0] & 0 \\
[X1,0] & [X1,h_2] & \dfrac{h_1^2}{1.25h_2^2+h_1^2} \\
[X1,1] & [X1,1] & 0 \\
[X1,1] & [X1,1-h_2] & \dfrac{49h_1^2}{5h_2^2+49h_1^2}
\end{array}\right)$$

Sous les conditions suivantes:

$$h_1 \leq \min_{U1,U2,U3,X1}\left(\frac{0.2}{|0.2X1+0.9U3-U2-U1|}\right)$$

$$h_2 \leq \min_{X2,U2,U3}\left(\frac{(X2+0.40)^2}{|X2-U3+U2|}\right)$$

les coefficients de la matrice sont positifs et leur somme en ligne vaut 1. La matrice M s'interprète bien comme une matrice de transition d'une chaine de Markov.
Le coût optimal vérifie l'équation:

$$(\lambda k+1)V^h = \min_{U1,U2,U3}\left(M(U1,U2,U3)V^h + kC(U1,U2,U3)\right)$$

que l'on résout par l'itération contractante:

$$V_{n+1}^h = \frac{\min_{U1,U2,U3}\left(M(U1,U2,U3)V_n^h + kC(U1,U2,U3)\right)}{(\lambda k+1)}$$

6.3 Optimisation de l'Hamiltonien

Il s'agit de minimiser la partie $\mathcal{H}^h$ de H^h dépendant du contrôle :

$$\mathcal{H}^h = -\partial_2^h VU3 - 0.9\partial_1^h VU3 - \partial_2^h VU2 + \partial_1^h VU2 + \partial_1^h VU1 + 2.5U1^{2/5}$$

On minimise $\mathcal{H}^h$ en $U = [U1, U2, U3]$ par une méthode de gradient projeté.

$$U_{n+1} = \mathcal{P}_{[0.1,1]\otimes[0,1]\otimes[0,1]}\left(U_n - \rho\frac{d\mathcal{H}^h(U_n)}{dU_n}\right)$$

soit

$$\left(\begin{array}{c}
U1_{n+1} = \mathcal{P}_{[0.10,1]}\left(\rho_1\left(U1_n^{-3/5} + \partial_1^h V\right) + U1_n\right) \\
U2_{n+1} = \mathcal{P}_{[0,1]}\left(\rho_2(\partial_1^h V - \partial_2^h V) + U2_n\right) \\
U3_{n+1} = \mathcal{P}_{[0,1]}\left(\rho_3(10\partial_2^h V - 9\partial_1^h V) + U3_n\right)
\end{array}\right)$$

Cet algorithme converge dès que le pas ρ vérifie:

$$0 < \rho < 2kK^{-2}$$

avec

$$k|V|^2 \leq D_U^2\mathcal{H}^h(V)V \leq K|V|^2$$

6.4 Résultats numériques

On fait un test numérique en prenant:

nombre de points de discrétisation : [13, 15]
précision pour la résolution implicite: 6.45 10^{-2}
pas pour la résolution implicite: 3.31 10^{-3}
nombre maximal d'itérations pour la résolution implicite: 4640
nombre maximal d'itérations pour l'optimisation de l'hamiltonien: 36
tests de convergence: précision 0.10

Les courbes suivantes représentent la fonction optimale d'utilité et les contrôles optimaux.

Références

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

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

[3] GOURSAT M.,QUADRAT J.P.(1975). Analyse numérique d'inéquations quasi variationnelles elliptiques associées à des problèmes de contrôle impulsionnel. IRIA Report.

[4] KUSHNER H.J.(1977). Probability methods in stochastic control and for elliptic equations. Academic Press.

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

[6] QUADRAT J.P.(1980). Existence de solution et algorithme de résolutions numériques de problèmes stochastiques dégénérés ou non. SIAM Journal of Control.

[7] QUADRAT J.P.(1975). Analyse numérique de l'équation de Bellman stochastique. IRIA Report.

[8] STROOCK F., VARADHAN S.R.S.(1979). Multidimensional Diffusion Process. Springer Verlag.

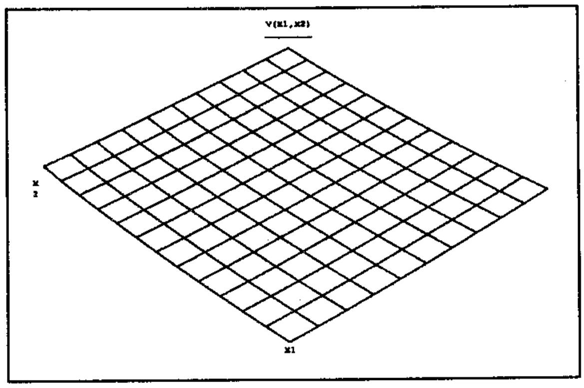

Figure 1 : coût optimal V

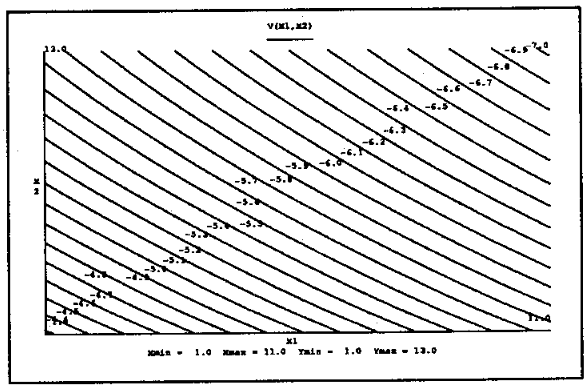

Figure 2 : Courbes de niveaux du coût optimal V

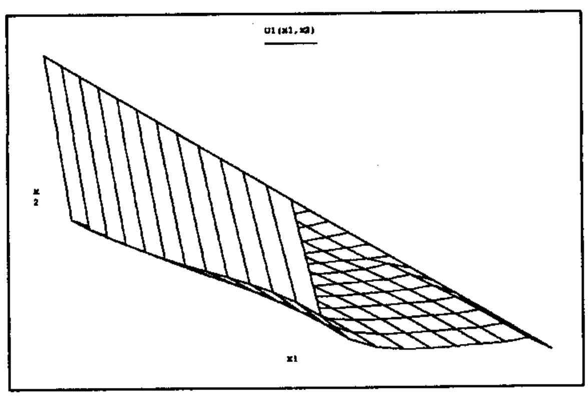

Figure 3 : contrôle optimal U1

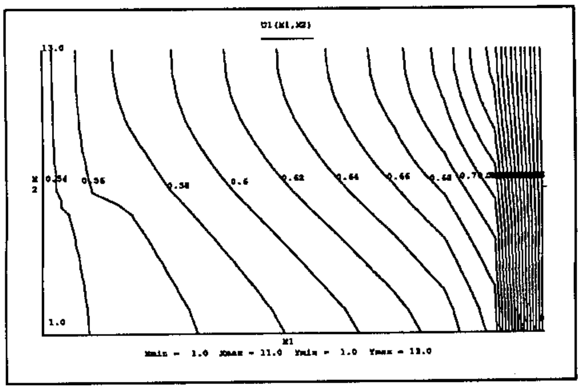

Figure 4 : Courbes de niveaux du contrôle optimal U1

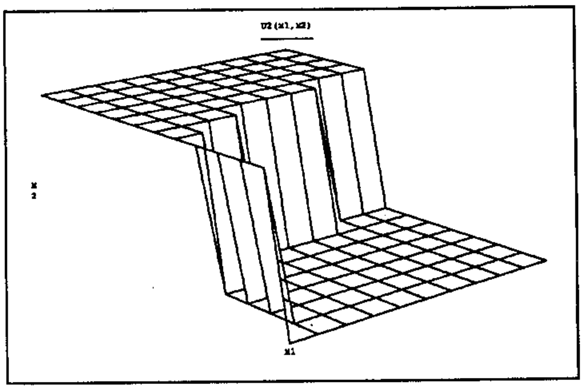

Figure 5 : contrôle optimal U2

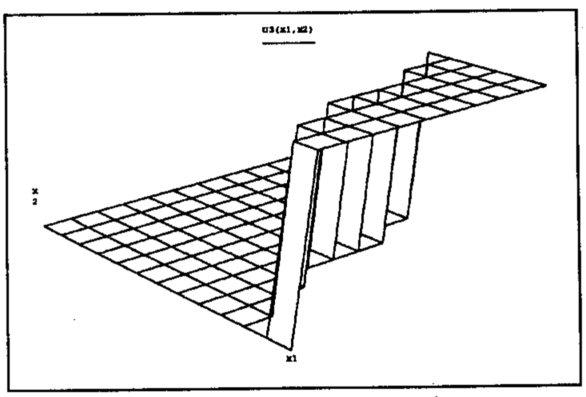

Figure 6 : contrôle optimal U3

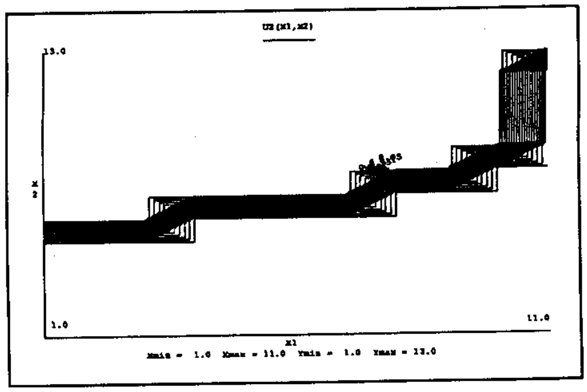

Figure 7 : Courbes de niveaux du contrôle optimal U2

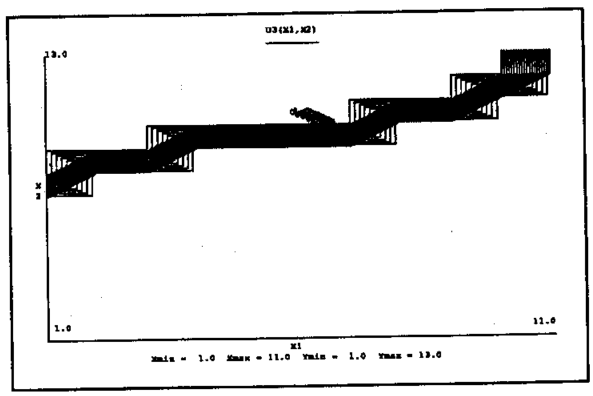

Figure 8 : Courbes de niveaux du contrôle optimal U3