PANDORE

J.P.Chancelier
C.Gomez
J.P.Quadrat
A.Sulem

INRIA
Domaine de Voluceau
78153 Le Chesnay
France

Abstract

The languages Lisp, Macsyma and Prolog provides a powerful environment for manipulating programs and reports. Macrofort is a language for interfacing Macsyma and Fortran. MacroTex does the samething for Latex. Using these facilities Pandore generates complete reports in Latex about studies in stochastic control . The user has only to specify its model in term of control of diffusion processes. Pandore does everthing else. The pandore paper in these proceedings has been completely generated by the system.

Introduction

- **Pandore** — Pandore is a system able to make complete engineering studies in optimization of dynamic sytem.
- **Macsyma** — It makes an intensive use of the three languages Macsyma, Prolog and Lisp. The complete
- **Prolog** — environment is Lisp since Macsyma and the Prolog used are written in Lisp.
- **Macrofort** — To facilitate the fortran generation from macsyma one uses a language called Macrofort.
- **Macrotex** — For Tex report generation one uses another language called Macrotex.

M a c s y m a

- **Formal calulus systems**
- **Trees**
- **Demos**
- **Interpret.**
- **Simplific.**
- **Evaluation**

Macsyma is one of the most famous formal calculus system.

Because it is written in Lisp it can manipulate easily trees representing algebraic formulas or typed expressions like rationals.

Mainly, the interpretor makes a "read,eval-simplify,print" loop, the simplifier and the evaluator calling each other recursively.

Most of the Macsyma power comes from its simplification possibilities which confer it surprising possibilities in partial evaluation.

Macsyma demos

The following simple demos show some algebraic possibilities of macsyma:

- rational calculus,
- differential calculus,
- matricial calculus,
- progamming language,
- equations solver.

Differential Calculus

```
(c2) 1/(X^3+2);

                                  1
(d2)                            ------
                                 3
                                x  + 2

                 ┌─── means previous expression
(c3) INTEGRATE(%,X);
                                              1/3
                                        2 x - 2
                                 atan(-----------)
            2     1/3       2/3              1/3                 1/3
       log(x  - 2    x + 2    )         2     sqrt(3)    log(x + 2   )
(d3) - ---------------------------- + ------------------ + -------------
                    2/3                   2/3                   2/3
                 6 2                     2    sqrt(3)          3 2
```

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

Differential calculus

```
(c4) DIFF(%,X);
                                                 1/3
                1                         2 x - 2                          1
(d4) ---------------------- - ---------------------------- + -----------------
                1/3 2              2/3    2     1/3       2/3      2/3         1/3
         (2 x - 2    )          6 2    (x  - 2     x + 2    )    3 2     (x + 2   )
     3 (------------- + 1)
                2/3
             3 2

             means rational simplification of the previous expression
(c5) RATSIMP(%);
                                  1
(d5)                            ------
                                 3
                                x  + 2
```

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

Programming language

```
(c18) FAC(N):=IF N=0 THEN 1 ELSE N*FAC(N-1);
(d18)              fac(n) := IF n = 0 THEN 1 ELSE n fac(n - 1)

(c19) FAC(5);
(d19)                                  120

(c20) G(N):=SUM(I*X^I,I,0,N);
                                        i
(d20)                       g(n) := sum(i x , i, 0, n)

(c21) G(10);
          10       9       8       7       6       5       4       3       2
(d21) 10 x   + 9 x  + 8 x  + 7 x  + 6 x  + 5 x  + 4 x  + 3 x  + 2 x  + x
```

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

Matricial Calculus

```
(c16) MAT:MATRIX([A,B,C],[D,E,F],[G,H,I]);
                                  [ a  b  c ]
                                  [         ]
(d16)                             [ d  e  f ]
                                  [         ]
                                  [ g  h  i ]

(c17) %^^2;
                [                  2                                    ]
                [ c g + b d + a    c h + b e + a b    c i + b f + a c    ]
                [                                                        ]
(d17)/R/        [                       2                                ]
                [ f g + d e + a d   f h + e  + b d    f i + e f + c d    ]
                [                                                        ]
                [                                    2                   ]
                [ g i + d h + a g   h i + e h + b g  i  + f h + c g      ]
```

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

Equation solver

```
(c13) SOLVE(X^6-1);
               sqrt(3) %i + 1      sqrt(3) %i - 1                       sqrt(3) %i + 1
(d13) [x = --------------, x = --------------, x = - 1, x = - --------------,
                     2                   2                                2

                                                          sqrt(3) %i - 1
                                                  x = - --------------, x = 1]
                                                                2

(c14) X^6-1,%[1];
                                                 6
                                (sqrt(3) %i + 1)
(d14)                           ----------------- - 1
                                        64

(c15) EXPAND(%);
(d15)                                   0
```

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

Rational Calculus

```
(c8) (X+3)^20;
                                             20
(d8)                                  (x + 3)

(c9) RAT(%);
           20          19           18             17               16                15
(d9)/R/ x    + 60 x    + 1710 x    + 30780 x    + 392445 x    + 3767472 x

                 14                13                 12                   11
 + 28256040 x    + 169536240 x    + 826489170 x    + 3305956680 x

                   10                  9                    8                     7
 + 10909657044 x    + 29753610120 x    + 66945622770 x    + 123591918960 x

                    6                     5                     4                     3
 + 185387878440 x    + 222465454128 x    + 208561363245 x    + 147219785820 x

                   2
 + 73609892910 x    + 23245229340 x + 3486784401
```

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

Rational Calculus

```
(c10) DIFF(%,X);
             19          18          17          16          15
(d10)/R/ 20 x   + 1140 x   + 30780 x   + 523260 x   + 6279120 x

             14              13              12              11
 + 56512080 x   + 395584560 x   + 2203971120 x   + 9917870040 x

               10                9                8                 7
 + 36365523480 x   + 109096570440 x  + 267782491080 x  + 535564982160 x

                6                 5                4                3
 + 865143432720 x  + 1112327270640 x  + 1112327270640 x  + 834245452980 x

                2
 + 441659357460 x  + 147219785820 x + 23245229340

(c11) FACTOR(%);
                                          19
(d11)                           20 (x + 3)
```

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

Interpretation

Macsyma objects

Almost all of the objects of macsyma are internally lists or typed lists.

Typed list

Typed lists are lists which have a first element specifying the type for example:

- the macsyma list [1,2] is represented by ((mlist) 1 2)
- the macsyma matrix "matrix([1,2])" is represented by

The other lists are similar to the lisp ones for example "a+b" is represented by ((mplus) $a $b)

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

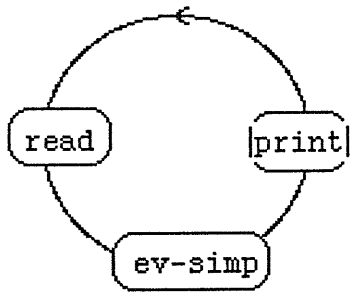
Interpretation

"read, eval-simp, print" loop

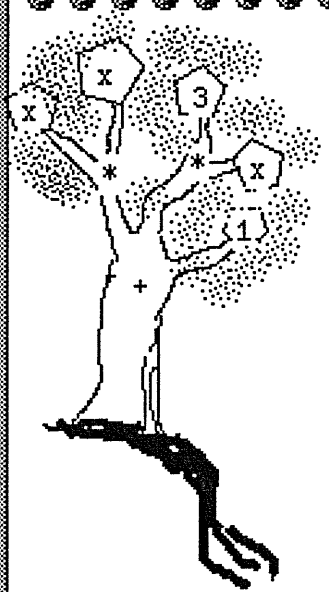
Macsyma manipulates these objects. Its interpretor does a "read eval-simp display loop".

For example suppose we want compute the derivative of (x+1)^3. The interpretor

1) reads "diff((x+1)^3,x)"
2) parses the expression to obtain:
   (($diff) ((mexpt) ((mplus) $x 1) 3) $x)
3) evaluates and simplifies the expression
   ((mtimes simp) 3
     ((mexpt simp) ((mplus simp) $x 1) 2))
4) 2D-prints the result: $3(x+1)^2$.



Trees



Very often formulas are represented by trees in Macsyma. Some typed expressions have a specific internal representation like rational expressions, Taylor series etc...

Trees are represented easily by list in Lisp. For example the macsyma expression x^2+3x+1 is written in Lisp by '(+ (* x x) (* 3 x) 1) which is the tree given by the picture.

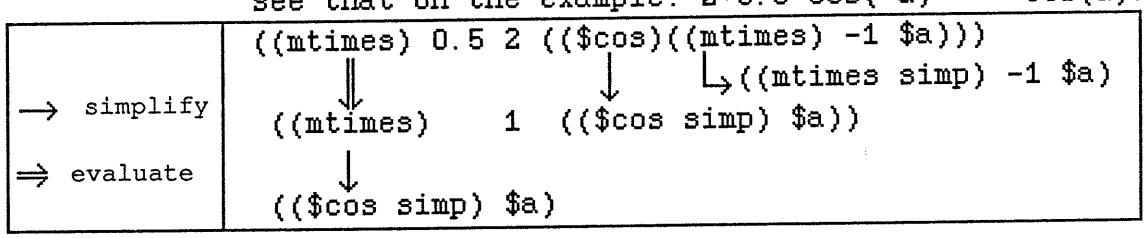
The rationals have a specialized internal representation nevertheless they have variable lengths and thus can be easily represented by lists.

Simplification

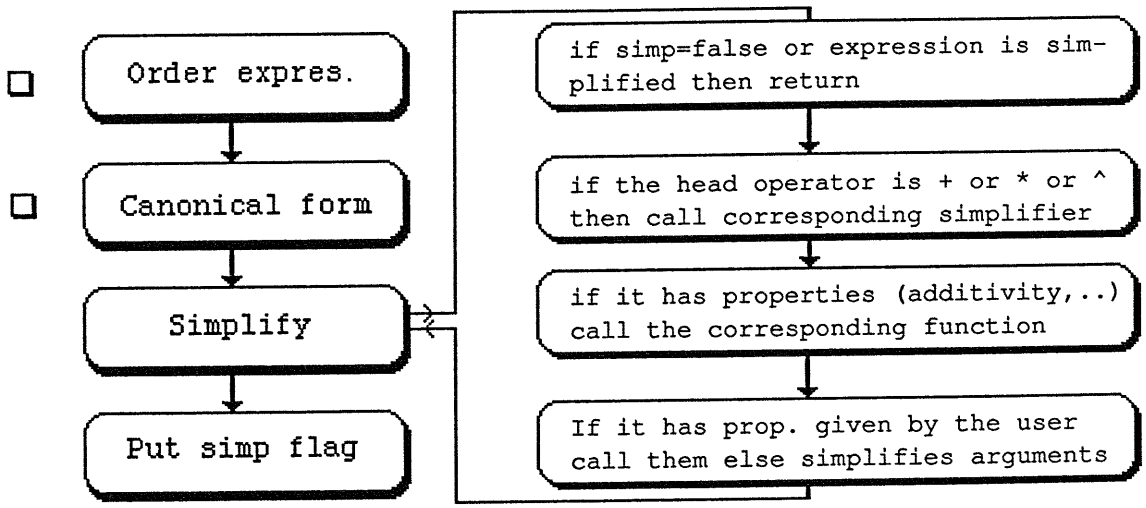
ev-simp

The simplification is the main improvement given by the formal calculus systems.

Because partial evaluation is an important part of the simplification process the simplifier and the evaluator calls each other recursively. Let us see that on the example: 2*0.5*cos(-a) --⇒ cos(a).



Simplification algorithm



Customizing the Simplification

declare

tellsimp

defrule

apply1

The user can customize the simplification of macsyma for that it has 3 possibilities:

1) puts preexisting properties on an operator:
   (declare(f,additive),f(x+y)) --> f(x)+f(y),
2) defines new rules for the simplifier:
   (matchdeclare(a,freeof(y)),
   tellsimp(f(a+y),f(a)),f(b+c+y)) --> f(b+c),
3) defines its own simplifier:
   (matchdeclare(x,true),defrule(r1,f(f(x)),g(x)),
apply1(f(f(f(x))),r1)) --> g(f(x)) .

ordering expression, canonical form

ordering expression

Macsyma order expression in the inverse lexicographic order for example:

```
%pi-3+a+x  ---> x+a+%pi-3
```

canonical form

Macsyma puts the intered expressions in canonical forms. In this canonical form the operators ""-" and "/" do not exist. For example a-b+1/c has the internal representation :

```
((mplus simp) ((mexpt simp) c -1)
                            b
              ((mtimes simp) a -1)))
```

Partial Evaluation

"foo"--->	f o o	diff(x^2,x) --------->	2 x	The simplification abilites of macsyma allows it to make good partial evaluation and to give an interesting answer to any syntactly correct expression.
2	2	'diff(x^2,x)	$\underline{d}(x^2)$	
a	a		d x	
a:2	2	declare(diff,noun)	done	
a	2	diff(x^2,x)	$\underline{d}(x^2)$	
'a	a		d x	
a^3+b	8 + b	remove(diff,noun)	done	
f(b)	f(b)	diff(x^2,x)	2 x	
f(a)	f(2)	sum(concat(x,i),i,1,2)	x1+x2	
f(x):=2*x	f(x):=2x	x1+x2,x1=2	x2+2	
f(b)	2 b			
f(a)	4			
a=b	2 = b			

Some Formal Calculus Systems

Macsyma Macsyma comes from MIT (Engelman, Martin, Moses and their students..). The project started in 1969. Its size is about 300.000 lines of Lisp. It represents more than 100 engineer-years of designing and 600 engineer-years of testing. The Lisp used are Maclisp, Zetalisp Franzlisp and Commonlisp.

Reduce Reduce started in 1969. Its author is Hearn. It works with Cambridge-Lisp, Standard-Lisp, Maclisp, Franzlisp, Le_lisp. It is almost as powerful as Macsyma now. It is the most available system.

Some Formal Calculus Systems

Scratchpad Scratchpad is the formal calculus system of IBM it is written in Lisp S/360, Lisp 1.5. It started in 1968. Two mains contributors are Griesmer and Jenks. Its new version has certainly the most beautiful design of all the systems.

Maple Maple comes from the university of Waterloo. Its authors are Char, Geddes and Gonnet. The project started in 1983. Maple is written in C. It is also almost as powerful as Macsyma. Its algorithms are more recent, thus sometimes it is much faster than Macsyma.

P r o l o g

Prolog allows to express easily relations between trees. It is very useful to build small data bases on such objects, with a powerful interrogating language.

- ☐ **Oblogis**
- ☐ **Clauses**
- ☐ **Unificat.**
- ☐ **And-or trees**

We use a Prolog written in Lisp called Oblogis. In this language Prolog is seen as two Lisp Macros, the first for installing the Prolog clauses, the second for asking questions.

Prolog adds two functionalities to Lisp: unification and exploration of and-or-trees.

Clauses

Prolog make inference on clauses of one of the following kind:

(1) hyp_1 & ... & hyp_n ==> concl

(2) ==> concl

- **literal**
- **variable**

where hyp_i and concl are literals

The second kind is called a fact and is a particular case of the first kind.

One asks some questions by asking a literal to be proved. That is finding the substitutions of the variables such that the litteral becomes true. True means proved or infered in this universe.

Literal

A literal is a tree without logic connectors, the leaves of which being of two kinds the variables and the constants.

variable The variables are universally quantified that means that each clauses where appears the variable is true for any substitution of the variable by a constant.

constant The constants are any symbol which is not a variable. Some syntactic sign permits the discrimination between the two kinds of symbols.

Exploration of and-or trees

Given a literal c to be proved. For all (or) the clauses for which the conclusion is c prolog tries to find all the substitutions which make the hypotheses true simultaneously (and).

Each hypothesis constrainted by the previous substitutions becomes a new literal to be proved.

Prolog makes the exploration by a depth first stategy that is at each level of its demonstration it starts by proving the first hypothesis. Only when it has achieved to prove it, it starts the second one.

Unification

Definition:

Two literals a and b are unifiable if it exists a substitution of the variables which appears in them denoted by s such that:

s(a)=s(b)

❑ **Oblogis**

Example using Oblogis syntax:

(+ !x z) and (+ y !x) are unifiable by the substitution:

s: { y ---> !x
   { z ---> !x.

Oblogis

❑ **example**

It is prolog written in Lisp by P.Gloess.
The syntax of the clauses are :
(1) (<- (concl) (hyp_1) ... (hyp _n))
(2) (<- (concl))
The litterals are lisp lists.
The constant are preceded by the macro character "!".
We ask the question "concl" by :

```
(? (concl))
```

Oblogis is able to manipulate structured objects but we don't discuss this feature here.

Oblogis example

```
(<- (hamiltonien b c (+ (* b !p) c)))
    (drift b)(cost c))
```

means that the expression (+ (* b p) c) is an hamiltonien if b is a drift and c is a cost.

```
(<- (drift (+ (* 2 !x) !u)))
(<- (cost (+ (* !x !x)(* 3 !u !u))))
```

are two facts which give a drift and a cost.
Then to the question (? (hamiltonien b c hm)) "what are the hamiltonians" the answer is:
b=(+ (* 2 x) u), c=(+ (* x x)(3 u u)),
hm=(+ (* (+ (* 2 x) u) p) (+ (* x x)(* 3 u u)))

M a c r o f o r t

- ☐ **Elem. Inst.**
- ☐ **Macros**
- ☐ **Stack Inst**
- ☐ **Fortran Generation**
- ☐ **Example**

On one hand Macrofort is a Pascal like language written in Macsyma, on the other hand it is a set of facilities added to Macsyma to generate Fortran. He takes sense only in the latter case.

There are three kinds of instructions: -the elementary instructions, -the macro instructions, -the stack manipulation instructions.

We use Macsyma to make the algebraic computations and the macrofort program manipulations the result is a Macsyma list the evaluation of which produce the fortran program.

Macsyma function	Fortran generated
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
writef(niv,fich,etiq,[liste])	WRITE (fich,etiq)liste
formatf(niv,etiq,[liste])	etiq FORMAT (liste)
gotof(niv,etiq)	GOTO etiq
ifgotof(niv,cond,etiq)	IF cond GOTO etiq
dof(niv,etiq,ind,fin)	DO etiq ind=1,fin
continuef(niv,etiq)	etiq CONTINUE
subroutinef(niv,nom,[liste])	SUBROUTINE nom(liste)
functionf(niv,type,nom,[liste])	typ FUNCTION nom(list)
callf(niv,nom,[liste])	COMMON /nom/liste
externalf(niv,liste)	EXTERNAL liste
declaref(niv,type,[liste])	type liste

Elementary Instructions

The elementary instructions are macsyma functions which prints one fortran instruction each.
"niv" is an indentation level.

Macsyma functions	Fortran generated
fairem(ind,max,prog)	DO etiq ind=1,max prog etiq CONTINUE
if_then_elsem(cond,then,else)	IF.NOT.cond GOTO etiq then GOTO fin etiq else fin CONTINUE
casem([cond1,prog1], ... [condn,progn], else)	IF.NOT.cond1 GOTO etiq1 prog1 GOTO fin etiq1 IF.NOT.cond2 GOTO etiq2 ..... etiqn-1 IF.NOT.condn GOTO etiq progn GOTO fin etiq else fin CONTINUE

Macro Instructions

The macro instructions are macsyma functions which return a set of elementary instructions. Then elementary instructions generate fortran code.

Macsyma functions	Fortran generated
untilm(cond,init,until)	init nuntil=0 deb CONTINUE until IF cond GOTO fin IF nuntil.GT.mxntl GOTO max nuntil=nuntil+1 GOTO deb max CONTINUE ierr=numuntil WRITE(out,format) format FORMAT("until overflow") fin CONTINUE numuntil=numuntil+1
writem([liste],[format],fich)	READ (fich,etiq)liste etiq FORMAT(format)
readm([liste],[format],fich)	READ (fich,etiq)liste etiq FORMAT(format)

Macro Instructions

"prog","init", "until" etc... mean a list of elementary instructions in a list form*. "cond" is a condition in macsyma syntax.

Macsyma functions	Fortran generated
writetabm(tab,list,fich)	WRITE(out,etiq)((tab(i1,..,in), i1,list[1]),i2...) etiq FORMAT(..... (k2-1) ('[',(f12.5,','),f12.5,']')..)
mainm(prog,nom)	real integer commons init prog formats END
subroutinem(name,[param],prog)	SUBROUTINE name(param,arg,error) real integer commons init prog formats END

Macro Instructions

"real","integer","commons", "formats","arg" are updated by stack instructions*.
The function "writetabm" is used to print in a file an array readable by macsyma.

Macsyma functions	Fortran generated
functionm(type,name,[param],prog)	type FUNCTION name(param,arg,error) real integer commons init prog formats END

Macro Instructions

It exists two other utilitaries:

-declarelispm(namef,namel,dim) which gives the access to the fortran array nomf of dimension dim by namel (only on Symbolics Lisp Machine).
-commentf(list) which displays in 2D the comments in "list" as a fortran comment.

Macsyma functions	Updated stack
realm(name)	dimension_reel
realdpm(name)	dimension_double_precision_reel
intergerm(name)	dimension_entier
commonm([name,[list]])	commons
commentairem(string)	commentaires
formatm(form)	formats
externalm(name)	externals
initm(inst)	init
progm(inst)	prog
prog1m(inst)	prog1
prog2m(inst)	prog2
argumentm(name)	arguments

General stack manipulation

push(val,pile)	add val on the top of pile
pushe(val,pile)	add val on the bottom of pile
pop(pile)	pop from the top of pile
pope(pile)	pop from the bottom of pile

Stack Instruction

Macros use stacks to generate the fortran code.

These stacks are updated by special instructions. All these stacks are initialized by the call: "contexte()".

Fortran Code Generation

Instruct. in the list form

To generate fortran code one build a macsyma list (prog) of elementary instructions in the list form, that is the macsyma list composed of the name of the function followed by its arguments without the first one: "niv". For example [gotof,100] is the list form of gotof(niv,100).

aplat

The evaluation of this macrofort program is obtained by calling aplat on this: list aplat(prog). The result is the display of the corresponding fortran program. We are urged to use only macro-instructions to build prog.

Example of a Macrofort use

```
gradient_const_step(f,vars):=block(
 [dim:length(vars),var],contexte(),
 for i:1 thru dim do ( var:vars[i],
  initm([equalf,var,0]),
  proglm([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],
    [progl,
     [equalf,error,sum((concat(vars[i],n)
                    -vars[i])^2,i,1,dim],
     prog2
     [writem,8,vars,[concat(dim,f12\.5)]]]]
))))$
```

```
    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
```


Example of use of Macrofort

The fortran program has been obtained by the call:
gradient_const_step(x^2+y^2, [x,y]).

The macsyma generator is more compact and much more general than the FORTRAN program generated. The latter is very similar to a hand written one.

```
         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
```

M a c r o T e x

- ☐ **Elem. Inst.**
- ☐ **Macros**
- ☐ **Stack Inst**
- ☐ **Latex Generation**
- ☐ **Example**

On one hand MacroTex is a Latex like language written in Lisp, on the other hand it is a set of facilities added to Macsyma to generate Latex. He takes sense only in the latter case.

There are three kinds of instructions: -the elementary instructions, -the macro instructions, -the stack manipulation instructions.

We use Macsyma to make the algebraic computations and the MacroTex program manipulations the result is a Macsyma list the evaluation of which produce the Latex program.

Macsyma function	Text generated
tex(exp)	exp in math mode
exptxt(exp)	exp in text math mode
format(string,arg1,...argn)	string(arg1..argn) following the directives:
~k	-tex(argi)
~l	-exptxt(argi)
~n	-argi (no display math)
~y	-quotation argi
~z	-reference argi
decoupet(exp,[e1..en],[op1..])	exp = e1 op1 e2
documenstylet(string)	\documenstyle{string}
labelt(string)	\label{string}
reft(string)	\ref{string}
citet(string)	\cite{string}
.....	.....
titlet(string,arg1,..argn)	\title{string(arg1..argn)}
sectiont(string,arg1..argn)	\section{string(arg1..argn)}
authort([name1,ad1]..)	\author{name1\\ad1}

Elementary Instructions

The elementary instructions are macsyma functions which prints one latex instruction each. They have other optional arguments*.

Macsyma function	Latex instructions
decoupem(exp)	cut exp in subexpressions having algebraic meaning and generate corresponding latex code.
abstractm(prog)	\begin{abstract} prog \end{abstract}
itemizem(item1,..)	\begin{itemize} \item item1 ..... \end{itemize}
enumeratem(prog1,..)	similar to itemize
stylem(doc,font,height,width, top,odd,even,par)	\documenstyle[font]{doc} \textheight=height pt \textwidth=width pt \topmargin=top pt \oddsimarginde=odd pt \evensidemargin=even pt \marginparwidth=par pt

Macro Instructions

The macro instructions are macsyma functions which generate a list of elementary macrotex instructions in the list form* and update or use some stacks.

Macsyma function	Latex instructions
citem([key1,source1],..)	\cite{key1} update the biblography with source1 .....
notationm()	\section{notation} \begin{itemize} notation1 .. \end{itemize}
bibliom()	\begin{thebibliography}{3} \bibitem{key1} source1 .... \end{thebibliography}
reportm(title,authors,abstact, chapters)	style title authors \begin{document} \maketitle \tableofcontents abstract notations chapters biblio \end{document}

Macro Instructions

The function contextt() initialize the global variables and the stack used. It must be called before starting the generation.

Stack Manipulation Instructions

The two stacks : list_references and list_ notations are used by the macro reportm.

They are updated by the functions :

- add_referencet(key,source),
- add_notationt(notation).

Then the references can be invoked using "key" in the format instruction with the directive ~y. For example :

```
add_referencet([Flem,"FLEMING-RISHEL...."]),
formatt("The optimal cost satisfies the dynamic
programming equation ~y.",Flem).
```

Latex Generation

list form

To generate Latex code one build a macsyma list (prog) of elementary instructions in the list form, that is the macsyma list composed of the name of the function followed by its arguments. For example [authort,["Pandore","Inria"]] is the list form of authort(["Pandore",Inria"]).

execute

The evaluation of this macrotex program is obtained by calling "execute" on this list:

```
execute(prog).
```

The result is the display of the corresponding Latex program.

Example of MacroTex use

```
gradient_const_step(f,vars):=block(
 [dim:length(vars),var],contextt(),
 for i:1 thru dim do ( var:vars[i],
  initm([tex,var=0]),
  prog1m([tex,concat(var,n)=
                 var-ro*diff(f,var)]),
  prog2m([tex,var=concat(var,n)]) ),
execute(
 rapportm([titlet,"Gradient method"],
    [authort,["Pandore","Inria"]],
    [format,"Until error<eps do"]
    [enumeratem,
     [prog1,
      [tex,error=sum((concat(vars[i],n)
                  -vars[i])^2,i,1,dim],
      prog2]]]))$
```

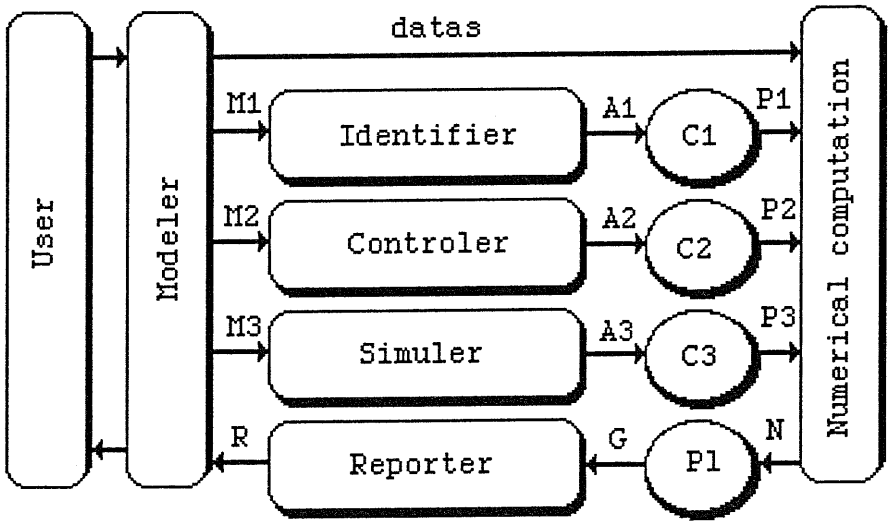

P a n d o r e

- **Costo**
- **Idsto**
- **Basile**
- **Objectives**

Pandore is an expert system on identificaton (idsto) and control (costo) of stochastic processes of diffusion type. It is oriented towards the resolution of nonlinear systems. The linear part is very rudimentary. In the future an interface with Basile (matlab based system for classical automatic control) will be done.

The objective is to automatize all the engineering tasks in these domains. The part achieved corresponds to the mouse sensitive blocks in the objective diagram.

Objectives



We want automatize all the tasks of an engineering study.

M:Model
A:Algorithm
P:Program
R:Report
G:Graph
N:numbers

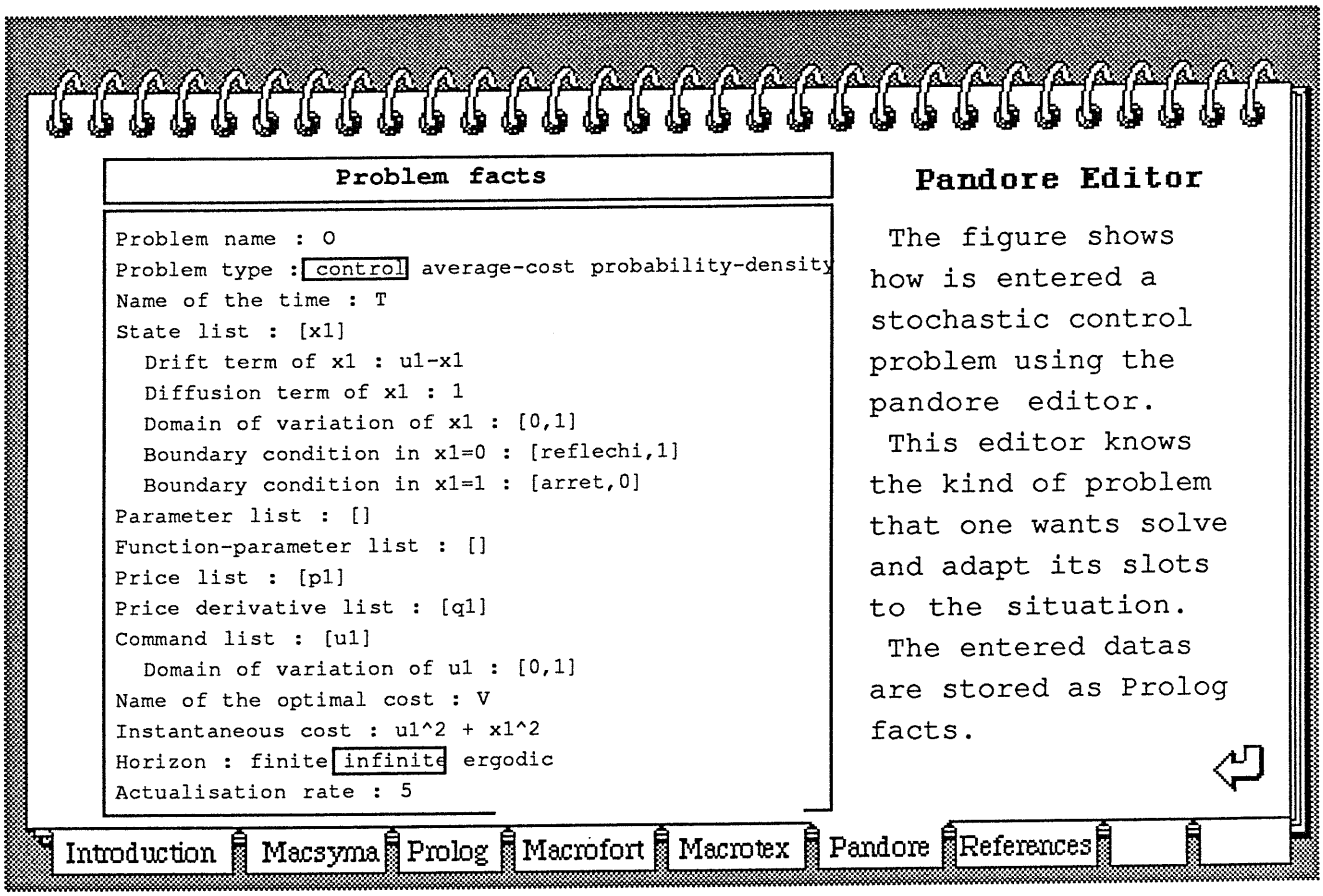
C:Compiler P1:plotter

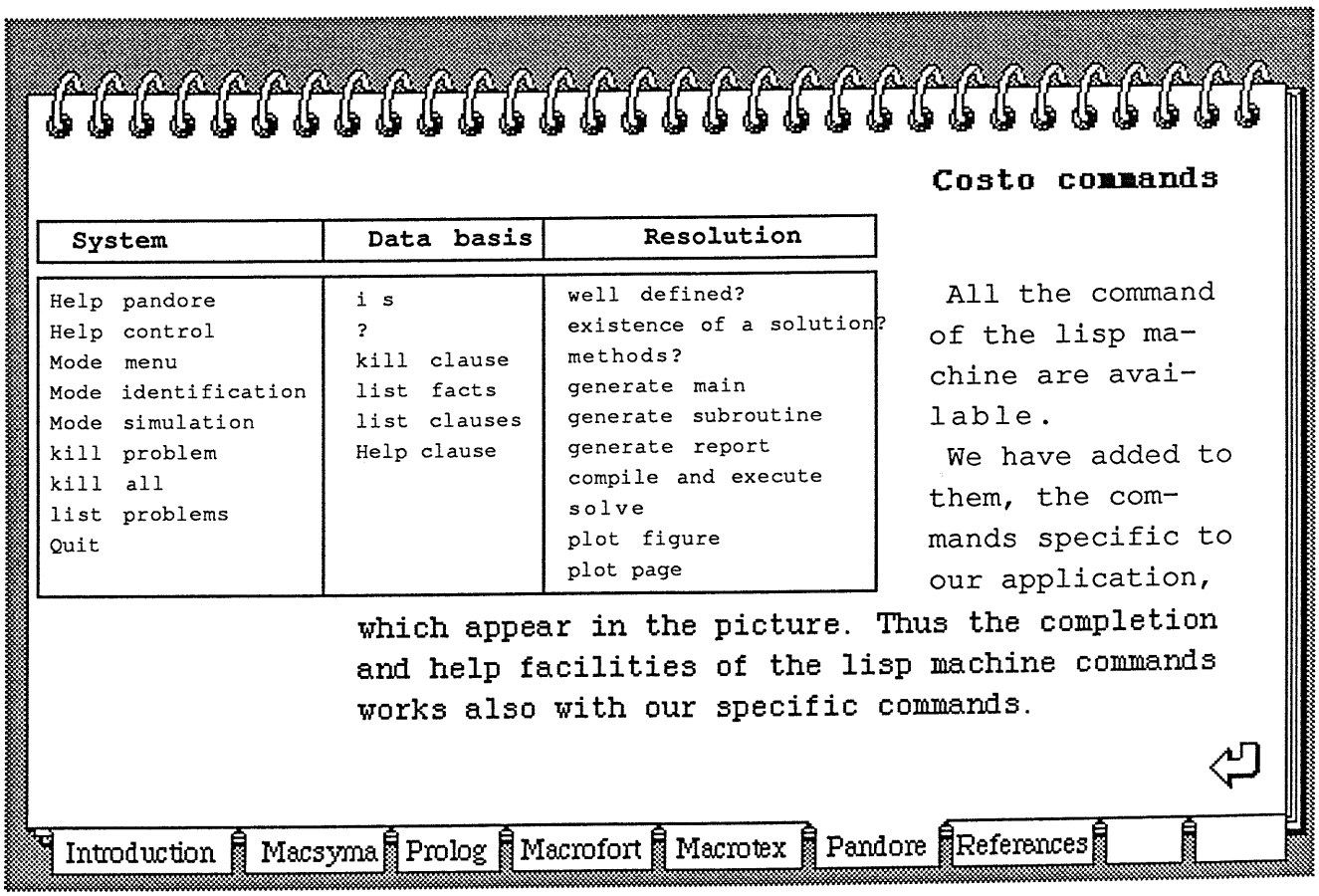
Modeler

- **pandore editor**
- **control commands**
- **identif. commands**

Now the modeler is only an interface with the user. The latter enter the model using a specialized editor, by filling some named slots. The slots appearing being not fixed but depending of the previous answers.

Then a specialized command language permit to ask questions about the model entered. Some of these commands are of very high level. They are partitioned according the problem to solve : control, identification, or simulation.


Problem facts
Problem name : O
Problem type : control average-cost probability-density
Name of the time : T
State list : [x1]
Drift term of x1 : u1-x1
Diffusion term of x1 : 1
Domain of variation of x1 : [0,1]
Boundary condition in x1=0 : [reflechi,1]
Boundary condition in x1=1 : [arret,0]
Parameter list : []
Function-parameter list : []
Price list : [p1]
Price derivative list : [q1]
Command list : [u1]
Domain of variation of u1 : [0,1]
Name of the optimal cost : V
Instantaneous cost : u1^2 + x1^2
Horizon : finite infinite ergodic
Actualisation rate : 5
Pandore Editor
The figure shows how is entered a stochastic control problem using the pandore editor.
This editor knows the kind of problem that one wants solve and adapt its slots to the situation.
The entered datas are stored as Prolog facts.
Introduction
Macsyma
Prolog
Macrofort
Macrotex
Pandore
References


Costo commands
System
Data basis
Resolution
Help pandore
Help control
Mode menu
Mode identification
Mode simulation
kill problem
kill all
list problems
Quit
i s
?
kill clause
list facts
list clauses
Help clause
well defined?
existence of a solution?
methods?
generate main
generate subroutine
generate report
compile and execute
solve
plot figure
plot page
All the command of the lisp machine are available.
We have added to them, the commands specific to our application,
which appear in the picture. Thus the completion and help facilities of the lisp machine commands works also with our specific commands.
Introduction
Macsyma
Prolog
Macrofort
Macrotex
Pandore
References

Controler

- **Control of diffusion**
- **Existence**
- **Dynamic program.**
- **Decoupling**
- **Stoch.grad.**
- **Regul.pert.**
- **Fortran generation**

The controler is a specialist of control of diffusion processes.

It is able to study the existence of a solution to the control problem.

It choses between four different approches to solve numerically the problem : dynamic programming, decoupling, stochastic gradient, regular perturbation. For each one it is able to generate a specific fortran routine and to call it to obtain numerical results.

Control of diffusion processes

The dynamic of the systems is:

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

where t denotes the time, X_t the state, U_t the control, W_t a Wiener process.

We want minimize, with respect to the control variable one of the following criterium :

(2) $E \int_0^T c(X_t, U_t)dt$

(3) $E \int_0^\infty e^{-lt} c(X_t, U_t)dt$

Dynamic programming method

The otimal cost $V(t,x)$ of the stochastic control problem satisfies the parabolic partial differential equation :

$$D_t V + \text{Min}\,(b(x,u).D_x V + c(x,u)) + \text{tr}(aD_{xx}V)=0$$

for the criterium (1) and the elliptic equation :

$$-lV + \text{Min}\,(b(x,u).D_x V + c(x,u)) + \text{tr}(aD_{xx}V)=0$$

for the criterium (2).

These equations are solved numerically after discretization by finite difference methods.

A complete description of the method is given in the generated report.

Decoupling method

In the case where the dynamic is uncoupled and the criterium is a general function of a sum of local cost we can optimize in the class of local feedbacks which does not change the coupling structure of the system.

Then we have to solve a control of the system of partial differential equations which describes the probability density of the state. Indeed in this case this density is the product of the density of each subsystem. The complete method is described in the generated report.

Stochastic gradient method

The stochastic gradient method consists in
1) parametrizing the feedback law,
2) optimizing the parameter a, seeing it as open loop control, by a stochastic iteration.

In the stochastic iteration we first compute the gradient $J(\omega)$ of the deterministic control obtained by simulating a trajectory of the noise and considering it as a known parameter, then we do the iteration:

$a \longrightarrow a - r_n J(\omega)$ with r_n satisfying

$\Sigma r_n = \infty$, $r_n > 0$, $r_n \longrightarrow 0$

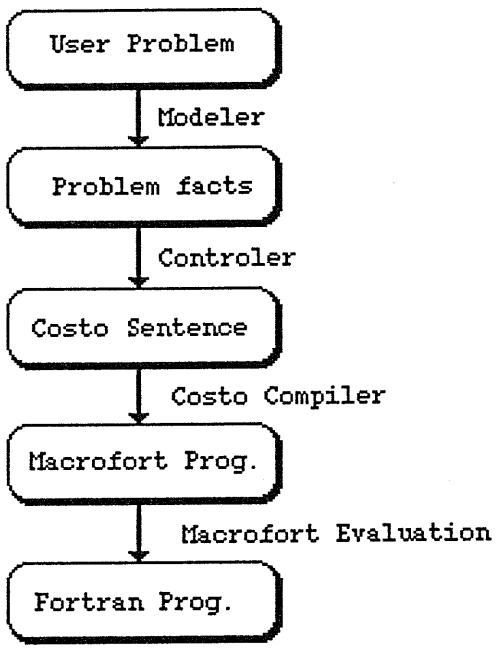
Regular perturbation method

When the intensity of the noise σ is small we know that the affine feedback :

$u(t,x) = u_0(t) + K(t)(x - x_0(t))$,

(where : $u_0(t)$ and $x_0(t)$ are respectively the optimal control and trajectory of the deterministic control problem obtained by annulating the noise, and $k(t)$ the gain of the linear quadratic control problem osculator to the optimal deterministic trajectory) leads to a cost optimal up to σ^4.

Thus the method consists in computing this affine Feedback.



Generation of numerical routines

The steps to generate a numerical routine to solve the control problem are given in the diagram.

The controler choses a method makes the algorithmic choices and expresses its choices in a language called Costo. A compiler written in Mac-syma produces the fortran code using the Macrofort facilities

Existence of a solution

The system is able to prove existence result of the solution of the the dynamic programming equation.

It verifies the hypotheses of the Lax-Milgram* theorem or those of the monotony theorem* of J.L. Lions. For that it builds the variational formulation of the problem and proves some coercivity.

To be able to do that some estimation facilities and a function able to apply the Green formula to a differential operator have been to be added to Macsyma.

[1]J.P.AUBIN, Approximation of elliptic Boundary-value Problems. Wiley Inter-science, New-York, 1972.
[2]F.BASKET,M.CHANDY,R.MUNTZ,J.PALACIOS Open Closed Mixed Network Of Queus With Different Class of Customers, J.A.C.M n°22, p. 248-260, 1975.
[3]R.BELLMAN, Dynamic Programming,Princeton University Press,1957.
[4]A.BENSOUSSAN, Stochastic Control by Functional Analysis Method, North-Holland 1982.
[5]A.BENSOUSSAN, Méthodes de Perturbartion en Contrôle Optimal,1985.
[6]A.BENSOUSSAN,J.L.LIONS, Applications des inéquations variationnelles en contrôle stochastique, Dunod,1978.
[7]P.F.CIARLET, The finite element method for elliptic problems, North Holland, 1978.
[8]D.CLAUDE, Decoupling of Nonlinear Systems, Systems Control Letters n°1, p242-248, 1982.
[9]A.etC.COLMERAUER, Prolog en 10 figures, La Recherche 1985.
[10]J.B.CRUZ, Feedback Systems, McGraw-Hill, 1972.
[11]F.DELEBECQUE,J.P.QUADRAT,
-Sur l'Estimation des Caractéristiques Locales d'un Processus de diffusion avec Sauts, Rapport INRIA 1978.
-Contribution of Stochastic Control Singular Perturbation Averaging and Team Theories to an Example of Large-Scale System: Management of Hydropower Production, IEEE AC23 n°2, p209-221 1978.

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

[12]J.C.DODU,M.GOURSAT,A.HERTZ,J.P.QUADRAT,M.VIOT Méthodes de gradient stochastique pour l'optimisation des investissements dans un réseau électrique, EDF Bulletin Serie C, n°2, 1981
[13]I.EKLAND,R.TEMAM, Analyse convexe et problèmes variationnels, Dunod, 1974.
[14]W.H.FLEMING, Control for small noise intensities, SIAM J.Control, vol.9, n°3, 1971.
[15]W.H.FLEMING-R.RISHEL, Optimal Deterministic and Stochastic Control, Springer-Verlag, 1975.
[16]F.GEROMEL,J.LEVINE,P.WILLIS, A fast Algorithm for Systems Decoupling using Formal Calculus, L.N.C.I.S n°63, Springer-Verlag 1984.
[17]P.GLOESS, Logis User's manual, Université de Compiegne, janvier 1984.
[18]P.GLOESS, Understanding Expert Systems,Université de Compiegne, janvier 1984.
[19]C.GOMEZ,J.P.QUADRAT,A.SULEM, Towards an Expert System in Stochastic Control : the Hamilton-Jacobi equation Part, L.N.C.I.S n°63, Springer-Verlag 1984.
[20]C.GOMEZ,J.P.QUADRAT,A.SULEM, Towards an Expert System in Stochastic Control : the Local-feedback Part, Congrés Rome sur le contrôle Stochastique, L.N.C.I.S Springer-Verlag 1985.
[21]THEOSYS Numerical methods in stochastic control., RAIRO Automatique 1983.
[22]M.GOURSAT,J.P.QUADRAT,
- Analyse numériques d'inéquations quasi variationnelles elliptiques associées à des problèmes de contrôle impulsionnel, IRIA Rapport,1975.
- Analyse numériques d'inéquations variationnelles elliptiques associées à des problèmes de temps d'arrêt optimaux, IRIA Rapport,1975.

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

[23]J.JACOD, Calcul Stochastique et Problèmes de Martingale, Springer-Verlag, 1979.
[24]E.KIEFER,J.WOLFOWITZ, Stochastic estimation of the maximum of a regression function, Ann. Math. Statistic, 23,n°3,1952.
[25]H.J.KUSHNER, Probabilty methods in stochastic control and for elliptic equations, Academic Press, 1977.
[26]H.J.KUSHNER,D.S.CLARK, Stochastic approximation methods for constrained and unconstrained systems,Springer Verlag,1978.

[27]P.L.LIONS,B.MERCIER, Approximation numérique des équations de Jacobi-Bellman, RAIRO,14 , pp.369-393, 1980.
[28]P.LIPCER-A.SHIRIAEV Statistique des Processus Stochastiques, Presse Universitaire Moscou, 1974.
[29]LISP Machine Lisp Manual, MIT Press,1982.

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

[30]MACSYMA Manual,MIT Press,1983.
[31]B.T.POLYAK, Convergence and convergence rate of iterative stochastic algorithms. Automatica i Telemekhanika, Vol.12,1976,pp.83-94.
[32]B.T.POLYAK, Subgradient methods a survey of soviet research in nonsmooth optimization, C. Lemarechal and K.Mifflin eds.Pergamon Press, 1978.
[33]B.T.POLYAK,Y.Z.TSYPKIN, Pseudogradient adaptation and training algorithms.Automatica i Telemekhanika,Vol.3,1973.
[34]J.P.QUADRAT, Existence de solution et algorithme de résolutions numériques de problèmes stochastiques dégénérés ou non, SIAM Journal of Control, mars 1980.
[35]J.P.QUADRAT, Analyse numérique de l'équation de Bellman stochastique, IRIA Report,1975.
[36]J.P.QUADRAT, On optimal stochastic control problem of large systems, Advances in filtering and optimal stochastic control, Lecture Notes in Control and Computer Science n° 42, Springer-Verlag,1982.
[37]J.P.QUADRAT,M.VIOT, Product form and optimal local feedback for multi-index Markov chains, Allerton Conference,1980.
[38]C.QUEINNEC, LISP Langage d'un autre type, Eyrolles, 1983.
[39]H.ROBBINS,S.MONRO, A stochastic approximation method. Ann. Math. Statist., 22, pp400-407, 1951.

Introduction | Macsyma | Prolog | Macrofort | Macrotex | Pandore | References

[37]J.P.QUADRAT,M.VIOT, Product form and optimal local feedback for multi-index Markov chains, Allerton Conference,1980.
[38]C.QUEINNEC, LISP Langage d'un autre type, Eyrolles, 1983.
[39]H.ROBBINS,S.MONRO, A stochastic approximation method. Ann. Math. Statist., 22, pp400-407, 1951.
[40]F.STROOCK,S.R.S.VARADHAN, Multidimensional Diffusion Processes, Springer-Verlag,1979.
[41]TORRION, Differentes méthodes d'optimisation appliquées à la gestion annuelle du système offre-demande français. Note EDF, EEG 1985.
[42]TURGEON, Optimal operation of multi-reservoir power system with stochastic inflows.Water resource resarch. April 1980
[43]CHANCELIER-GOMEZ-QUADRAT-SULEM Un systeme expert pour l'optimisation de système dynamique, Congres d'Analyse Numerique INRIA Versailles Decembre 85, North Holland.