Introduction au Logiciel GAMS
(General Algebraic Modeling System)

J.-M. Reneaume

SOMMAIRE

A. *INTRODUCTION*

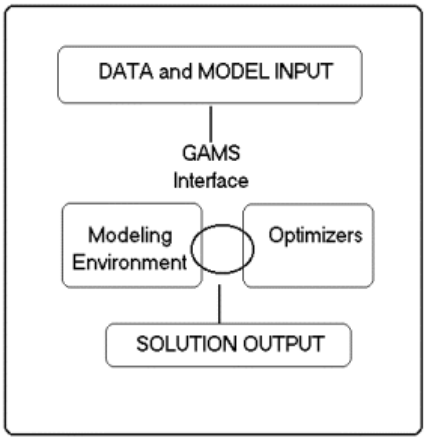
Notre objectif est de donner un rapide (mais complet !) aperçu de GAMS et de ses capacités. Face au développement des capacités de calcul et des algorithmes d'optimisation, ce logiciel a été créé afin de répondre à un certain nombre de besoins :

- Fournir un langage structuré pour la description synthétique de modèles complexes de grande taille,
- Permettre la modification du modèle de façon simple et sécurisée,
- Permettre la définition univoque de relations algébriques,
- Permettre la définition du modèle de façon indépendante des algorithmes de résolution utilisés.

Deux grands principes ont guidé les concepteurs de ce logiciel :

- Tous les algorithmes doivent être utilisables sans aucune modification du modèle (en particulier, l'introduction d'un nouvel algorithme n'affecte pas le modèle),
- La formulation du modèle doit être indépendante des données utilisées par ce dernier (en particulier, l'augmentation de la taille du modèle, n'affecte pas la complexité de la représentation de ce dernier).

L'environnement de modélisation et la bibliothèque d'algorithmes d'optimisation ont été conçus indépendamment.



Nous illustrerons notre propos avec l'exemple de l'atelier vu en cours : deux machines (A et B) peuvent fabriquer deux produits (1 et 2). L'objectif est de maximiser le chiffre d'affaire annuel de l'atelier en recherchant le temps passer par chaque machine à fabriquer l'un ou l'autre des produits. La production maximum en 2 est de 200 tonnes par an. L'atelier fonctionne 8330 hr/an. Le tableau 1 résume les données du problème en terme de :

- quantités produites par machine et par produit,
- Prix de vente des produits (les qualités, donc les prix de vente, des produits varient en fonction de la machine utilisée).

	Quantité [Kg/hr]		Prix de vente [FF/Kg]	
Machine	**1**	**2**	**1**	**2**
A	12.1	11.3	2.1	4.5
B	14.2	15.2	1.2	5.6

Tableau 1 : Données pour l'exemple de l'atelier

Donnons immédiatement la représentation de ce problème en « langage GAMS » :

```
*
* atelier de fabrication (2 machines, 2 produits)
*
sets
   i machines /A, B/
   j produits /1, 2/;

table m(i,j) production par machine et par produit en Kg par hr
                  1       2
         A        12.1    11.3
         B        14.2    15.2;
table p(i,j) prix de vente par machine et par produit en Francs par Kg
                  1        2
         A      2.1      4.5
         B      1.2      5.6;
scalar f fonctionnement de l atelier en h par an /8330/;
scalar l2 production annuelle maximale de 2 en Kg par an /200000./;

variables
       t(i,j) temps passe par la machine i a produire j
       z   chiffre d'affaire;
positive variable t;

equations
       temps(i)
       prodmax
       cost;
temps(i).. sum(j,t(i,j))=e=f;
prodmax..  sum(i,t(i,'2')*m(i,'2'))=l=l2;
cost..     z=e=sum((i,j),t(i,j)*m(i,j)*p(i,j));

model atelier /all/;
solve atelier using lp maximizing z;
display t.l, t.m ;
```

B. *DESCRIPTION DU MODELE*

1. Structure générale du modèle

a) Structure du fichier d'entrée

Le modèle comporte les paragraphes suivants :

- **SETS**
  Déclaration
  Affectation

- **DATA** (**PARAMETERS**, **TABLES**, **SCALAR**)
  Déclaration
  Affectation

- **VARIABLES**
  Déclaration
  Affectation du type

- [Affectation des bornes et des valeurs initiales des variables]

- **EQUATIONS**
  Déclaration
  Définition

- Instruction **MODEL** et **SOLVE**

- [Instruction **DISPLAY**]

Les instructions entre [ ] sont facultatives.

b) Règles générales

Le modèle doit être décrit en utilisant l'environnement (éditeur) de GAMS. Ce dernier créera un fichier ayant le suffixe .gms. Par exemple : atelier.gms

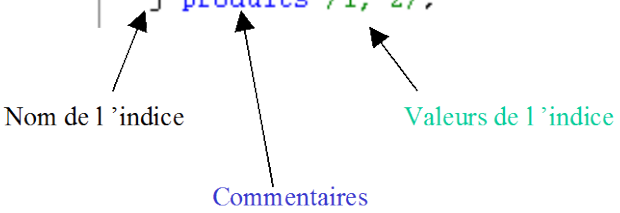
Les règles générales suivantes doivent être respectées :

1. Un élément (indice, paramètre, équation …) ne peut être référencé (affecté ou utilisé) avant d'avoir été déclaré.
2. Une instruction doit être suivie d'un point virgule (;)
3. Une ligne commençant par une * est un commentaire (ligne non interprétée par le compilateur)
4. Le nom d'un élément comporte au maximum 9 caractères alphanumériques. Le premier est toujours alphabétique.
5. Il existe un certain nombre de mots réservés. Ils ne doivent pas être utilisés en dehors du contexte prévu par le GAMS. Par exemple le mot **sets**, est un mot réservé à la déclaration des indices (Cf. paragraphe suivant). Il ne doit donc pas être utilisé comme identificateur d'une variable. Les mots réservés sont repérés en gras et en bleu par l'éditeur de texte.

2. SETS

Ce paragraphe permet de décrire les ensembles d'indices qui seront utilisés dans le modèle.

```
sets
   i machines /A, B/
   j produits /1, 2/;
```



Remarques:

- Ecriture équivalente:

```
set   i machines /A, B/;
set   j produits /1, 2/;
```

- Ici, la déclaration et l'affectation sont simultanées.
- Tous les éléments sont traités comme des caractères et non des nombres.

3. DATA

Les données (DATA) peuvent être de trois types :

- PARAMETERS
- TABLES
- SCALAR

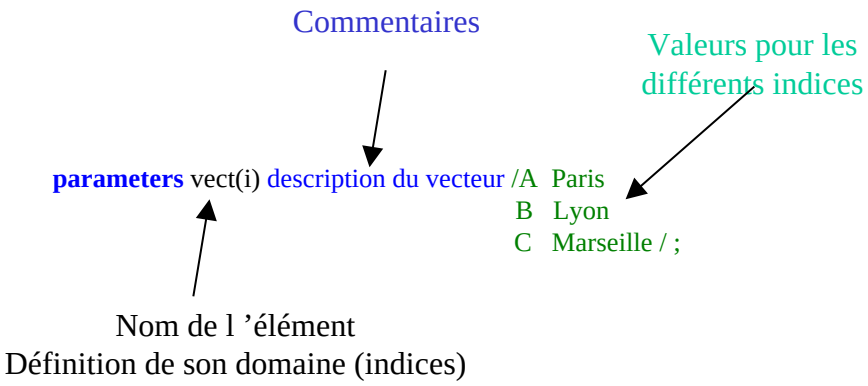
a) PARAMETERS

Les données sont ici entrées par une liste.

Le modèle de l'exemple de l'atelier ne comporte pas une telle liste. Prenons l'exemple suivant :
Un vecteur « vect » comporte trois éléments. Il existe donc trois valeurs de l'indice de ce vecteur, par exemple A, B et C. L'instruction suivante permet de déclarer et d'affecter les indices :

Set i indices du vecteur /A, B, C/ ;

Le vecteur sera déclaré et affecté de la façon suivante :



Remarques :

- Les paires (indice, valeur) sont séparées par des virgules ou données sur des lignes différentes.

- Le compilateur détectera une erreur si la valeur d'un indice n'est pas compatible avec la déclaration du domaine. Par exemple :

Parameters vect(i) description du vecteur /A Paris, X Lyon, C Marseille/ ;

Cette instruction n'est pas correcte car la valeur X n'est pas définie pour l'indice i.

- Zéro est la valeur par défaut. Seule l'affectation des valeurs non nulles est requise.

b) TABLES

Cette instruction permet de déclarer et d'affecter des tableaux.

```
table m(i,j) production par machine et par produit en Kg par hr
                    1       2
          A         12.1    11.3
          B         14.2    15.2;
table p(i,j) prix de vente par machine et par produit en Francs par Kg
                    1        2
          A       2.1       4.5
          B       1.2       5.6;
```

Indices de colonne

Indices de ligne

valeurs

c) SCALAR

Il s'agit du cas particulier d'une liste comportant un unique élément.

```
scalar f fonctionnement de l atelier en h par an /8330/;
scalar l2 production annuelle maximale de 2 en Kg par an /200000./;
```

d) Affectation directe

Il est toujours possible de découpler la déclaration et l'affectation. Par exemple, l'instruction

Parameters vect(i) description du vecteur /A Paris, B Lyon, C Marseille/ ;

Est équivalente aux instructions :

Parameters vect(i) description du vecteur ;
Vect('A') = Paris ;
Vect('B') = Lyon ;
Vect('C') = Marseille ;

Remarques :

- Il s'agit d'instructions différentes. Elles sont donc toutes suivies d'un point virgule.

- Il est aussi possible de rencontrer, par exemple, la syntaxe suivante :
  Set i indices du vecteur /A, B, C/ ;
  Parameters vect1(i) description du vecteur /A 2, B 4, C 6/ ;
  Parameters vect2(i) description du vecteur ;
  vect2(i) = vect1(i)+7 ;
  Ici on utilise le domaine (i) et non une valeur particulière d'indice, qui est alors placée entre ' '.

4. VARIABLES

Il s'agit maintenant de déclarer les variables d'optimisation.

```
variables
        t(i,j) temps passe par la machine i a produire j
        z    chiffre d'affaire;
positive variable t;
```

Remarques :

- La variables t est une matrice, deux domaines sont définis : i et j.
- La variable z est un scalaire. Il n'y a donc pas de domaine.
- La dernière ligne permet de déclarer le type. Différents types sont possibles :

Variable Type	Allowed Range of Variable
Free	$-\infty$ to $+\infty$
Positive	0 to $+\infty$
Negative	$-\infty$ to 0
Binary	0 or 1
Integer	0,1,...., 100

- Le type par défaut est « Free ».
- La définition du domaine n'est pas répétée dans la déclaration du type.

5. EQUATIONS

Les équations sont déclarées puis définies.

a) Déclaration

```
equations
        temps(i)contraintes sur les temps
        prodmax contrainte de production maximum
        cost fonction objectif;
```

Remarques :

- Il s'agit d'une unique instruction. Elle est terminée par un point virgule.
- Le domaine doit être défini si nécessaire.

b) Définition

```
temps(i).. sum(j,t(i,j))=e=f;
prodmax..  sum(i,t(i,'2')*m(i,'2'))=l=12;
cost..     z=e=sum((i,j),t(i,j)*m(i,j)*p(i,j));
```

Remarques :

- Le domaine est encore défini si nécessaire.
- Chaque instruction est terminée par un point virgule.
- Les valeurs particulières d'indices sont entre ' '.
- Les différents opérateurs sont : =e= pour les égalités, =l= pour les inégalités (≤), =g= pour les inégalités (≥). Il ne faut pas confondre le signe = (utilisé pour les affectations) et =e= (opérateur de relation lors de la définition des équations).

c) Opérateurs

Les opérateurs classiques sont utilisés :

+ : addition
- : soustraction
* : multiplication
/ : division
** : puissance

Les règles de priorité sont respectées.

d) Syntaxe pour les sommes

La syntaxe générale pour une sommation est la suivante :

Sum (indexe, opérandes)

Par exemple :

$\sum_i x_{i,j}$ s'écrit **sum** (i, x(i,j))

$\sum_i \sum_j c_{i,j}.x_{i,j}$ s'écrit **sum** ((i,j) , c(i,j)*x(i,j))

Pour les produits, la syntaxe est la même. On utilise alors **prod**.

e) Fonctions intrinsèques

Les fonctions intrinsèques suivantes sont disponibles :

Function	Description
`errorf(x)`	Integral of the standard normal distribution from -∞ to x
`exp(x)`	Exponential, e^x
`log(x)`	Natural logarithm, $\log_e(x)$
`log10(x)`	Common logarithm, $\log_{10}x$
`normal(x,y)`	Random number normally distributed with mean x and standard deviation y
`uniform(x,y)`	Random number with uniform distribution between x and y
`abs(x)`	Absolute Value of x, i.e. |x|
`ceil (x)`	Ceiling of x. Smallest integer ≥ x
`floor(x)`	Floor of x. Largest integer ≤ arg
`mapval(x)`	Mapping function. Assigns unique numbers to special values.
`max(x,y,..`	Largest value among all arguments.
`min(x,y,..)`	Smallest value among all arguments
`mod(x,y)`	Remainder. `x - y*trunc(x/y)`
`power(x,y)`	Integer power. x^y , where y must be an integer
`round(x)`	round x to the nearest integer
`round (x,y)`	Rounds x to y decimal places right (+) or left (-) to the decimal point
`sign(x)`	Returns 1 if x > 0, -1 if x < 0, and 0 if x = 0
`sqr(x)`	Square of x. x^2
`sqrt(x)`	Square root of x. $\sqrt{x}$
`trunc(x)`	sign (x) times `floor(abs(x))`
`arctan(x)`	Tan^{-1} (x). Result in radians
`cos(x)`	Cosine(x); x in radians
`sin(x)`	Sine(x). x in radians

Deux fonctions intéressantes :

ord : donne la position relative d'un élément dans un ensemble (set)

Exemple :

```
set i indice /101*150/ ;
parameter a(i) ;
a(i) = ord(i) ;
```

Ainsi on aura, par exemple, a(101) égal à 1, a(140) égal à 40 …

card : donne le nombre d'éléments d'un ensemble (set)

Exemple :

```
set i indice /101*150/ ;
scalar s;
v = card(i);
```

La valeur 50 est affectée au scalaire **v**.

6. MODEL

Cette instruction permet de définir un modèle en utilisant toutes les équations ou seulement une partie d'entre elles.

```
model atelier /all/;
```

Remarques :

- Dans ce cas toutes les équations sont utilisées.
- Si on ne souhaite pas utiliser toutes les équations, on donne alors la liste des équations entre / /. Les noms des équations sont séparés par des virgules.
- Il est possible de définir plusieurs modèles.

7. SOLVE

Cette instruction permet d'appeler le solveur.

```
solve atelier using lp maximizing z;
```

Cette instruction signifie : résoudre le modèle « atelier » en utilisant un algorithme de Programmation Linéaire et en maximisant la variable « z » (cette variable est définie par l'équation « cost » du modèle « atelier »).

Les différents types d'algorithmes sont : LP, NLP, MIP, MINLP.

Il est possible de maximiser (**maximizing**) ou de minimiser (**minimizing**).

8. Bornes, valeurs et sensibilités (.LO .UP .L .M)

GAMS permet de gérer un certain nombre d'informations concernant les variables et les équations :

.lo : borne inférieure
.up : borne supérieure

.l : valeur
.m : shadow price (analyse de sensibilité)

x.lo désigne par exemple la borne inférieure de la variable x.

Les bornes inférieures (.lo) et supérieures (.up) sont gérées par l'utilisateur.
Les valeurs des variables (.l) et des sensibilités (.m) peuvent être initialisées par l'utilisateur mais elles seront déterminées par le solveur.

9. DISPLAY

Il est possible de spécifier un certain nombre d'affichages.

```
display t.l, t.m ;
```

Cette instruction indique que l'utilisateur souhaite faire afficher la valeur de la variable « t » (t.l) ainsi que sa sensibilité (t.m).

C. *SORTIES*

1. Messages d'erreur

Si une erreur de syntaxe est détectée par le compilateur, GAMS affiche à l'écran un message du type suivant :

atelier

```
GAMS Rev 118  Copyright (C) 1987-2000 GAMS Development. All rights reserved
Licensee: GAMS Development Corporation, Washington, DC   G871201:0000XX-XXX
          Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC9999
--- Starting compilation
--- ATELIER.GMS(13) 1 Mb 1 Error
*** Error 170 in D:\ENSEIGNEMENT\3A - OPTIMISATION\COURS\GAMS\EXEMPLE ATELIER GAMS\ATELIER.GMS
    Domain violation for element
--- ATELIER.GMS(35) 1 Mb 2 Errors
*** Error 257 in D:\ENSEIGNEMENT\3A - OPTIMISATION\COURS\GAMS\EXEMPLE ATELIER GAMS\ATELIER.GMS
    Solve statement not checked because of previous errors
--- ATELIER.GMS(37) 1 Mb 3 Errors
*** Error 141 in D:\ENSEIGNEMENT\3A - OPTIMISATION\COURS\GAMS\EXEMPLE ATELIER GAMS\ATELIER.GMS
    Symbol neither initialized nor assigned
       A wild shot: You may have spurious commas in the explanatory
       text of a declaration. Check symbol reference list.
--- ATELIER.GMS(37) 1 Mb 3 Errors
*** Status: Compilation error(s)
Exit code = 2
```

La première erreur est repérée par le code 170 : un élément est en dehors de son domaine.
Il faut se reporter alors au fichier de résultats (suffixe .lst, ici atelier.lst).

atelier.gms | atelier.lst

```
GAMS Rev 118  Windows NT/95/98                    03/05/03 14:13:35  PAGE    1
General  Algebraic  Modeling  System
Compilation


   1  *
   2  * atelier de fabrication (2 machines, 2 produits)
   3  *
   4  sets
   5   i machines /A, B/
   6   j produits /1, 2/;
   7
   8  table m(i,j) production par machine et par produit en Kg par hr
   9                    1       2
  10           A       12.1    11.3
  11           B       14.2    15.2;
  12  table p(i,j) prix de vente par machine et par produit en Francs par Kg
  13                    1        42
****                               $170
  14           A      2.1      4.5
  15           B      1.2      5.6;
  16  scalar f fonctionnement de l atelier en h par an /8330/;
  17  scalar l2 production annuelle maximale de 2 en Kg par an /200000./;
  18
  19  variables
  20         t(i,j) temps passe par la machine i a produire j
  21         z   chiffre d'affaire;
  22  positive variable t;
  23
```

Les erreurs sont toujours repérées par ****. Donc ici, la valeur 42 pour l'indice de colonne du tableau « m » n'est pas valable. En effet l'indice « j » a été défini pour 1 ou 2.

Une erreur peut amener le compilateur à mal interpréter le reste du fichier et donc à signaler d'autres erreurs de syntaxes. Ici cette seule erreur génère trois messages.

2. Sorties sans erreurs de syntaxe

Une fois qu'il n'y a plus de messages d'erreur relatifs à la syntaxe, le fichier de résultats (ici, atelier.lst) contient les principales information suivantes :

a) Rappel du fichier d'entrée

Le fichier de données est réécrit avec la numérotation des lignes

b) Explicitation des contraintes

Le fichier de données contenait l'instruction suivante :

temps(i).. **sum**(j,t(i,j))=e=f;

Ceci sera explicité par :

```
---- temps  =E=  contraintes sur les temps


temps(A)..  t(A,1) + t(A,2) =E= 8330 ; (LHS = 0, INFES = 8330 ***)


temps(B)..  t(B,1) + t(B,2) =E= 8330 ; (LHS = 0, INFES = 8330 ***)
```

Il est alors possible de vérifier qu'il n'y a pas d'erreurs dans l'écriture du modèle.

c) Statistiques du modèle

```
MODEL STATISTICS

BLOCKS OF EQUATIONS       3     SINGLE EQUATIONS        4
BLOCKS OF VARIABLES       2     SINGLE VARIABLES        5
NON ZERO ELEMENTS        11
```

d) Résumé de la résolution

```
               S O L V E      S U M M A R Y

     MODEL   atelier              OBJECTIVE  z
     TYPE    LP                   DIRECTION  MAXIMIZE
     SOLVER  BDMLP                FROM LINE  35

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE           1085926.3124

 RESOURCE USAGE, LIMIT          0.020     1000.000
 ITERATION COUNT, LIMIT         2        10000
 Relaxed MINLP gives integer solution.
```

Les différentes valeurs du « SOLVER STATUS » sont :

Solver Status	Meaning
1 NORMAL COMPLETION	This means that the solver terminated in a normal way: i.e., it was not interrupted by an iteration or resource limit or by internal difficulties. The model status describes the characteristics of the accompanying solution.
2 ITERATION INTERRUPT	This means that the solver was interrupted because it used too many iterations. Use option `iterlim` to increase the iteration limit if everything seems normal.
3 RESOURCE INTERRUPT	This means that the solver was interrupted because it used too much time. Use option `reslim` to increase the time limit if everything seems normal.
4 TERMINATED BY SOLVER	This means that the solver encountered difficulty and was unable to continue. More detail will appear following the message.
5 EVALUATION ERROR LIMIT	Too many evaluations of nonlinear terms at undefined values. You should use bounds to prevent forbidden operations, such as division by zero. The rows in which the errors occur are listed just before the solution.
6 UNKNOWN ERROR PREPROCESSOR(S) ERROR SETUP FAILURE ERROR SOLVER FAILURE ERROR INTERNAL SOLVER ERROR ERROR POST-PROCESSOR ERROR(S) ERROR SYSTEM FAILURE	All these messages announce some sort of unanticipated failure of GAMS, a solver, or between the two. Check the output thoroughly for hints as to what might have gone wrong.

La valeur recherchée est « 1 NORMAL COMPLETION ».

Les différentes valeurs du « MODEL STATUS » sont :

Here is a list of possible MODEL STATUS messages:

Model Status	Meaning
1 OPTIMAL	This means that the solution is optimal. It only applies to linear problems or relaxed mixed integer problems (RMIP).
2 LOCALLY OPTIMAL	This message means that a local optimal. This is the message to look for if the problem is nonlinear, since all that can guarantee for general nonlinear problems is a local optimum.
3 UNBOUNDED	That is means that the solution is unbounded. This message is reliable if the problem is linear, but occasionally it appears for difficult nonlinear problem that are not truly unbounded, but that lack some strategically paced bounds to limit the variables to sensible values.
4 INFEASIBLE	This means that the linear problem is infeasible. Something is probably wrongly specified in the logic or the data.
5 LOCALLY INFEASIBLE	This message means that no feasible point could be found for the nonlinear problem from the given starting point. It does not necessarily mean that no feasible point exists
6 INTERMEDIATE INFEASIBLE	This means that the current solution is not feasible, but that the solver program stopped, either because of a limit (iteration or resource), or because of some sort of difficulty. Check the solver status for more information.
7 INTERMEDIATE NONOPTIMAL	This is again an incomplete solution, but it appears to be feasible.
8 INTEGER SOLUTION	An integer solution has been found to a MIP (mixed integer problem). There is more detail following about whether this solution satisfies the termination criteria (set by options optcr or optca
9 INTERMEDIATE NON-INTEGER	This is an incomplete solution to a MIP. An integer solution has not yet been found.
10 INTEGER INFEASIBLE	There is no integer solution to a MIP. This message should be reliable.
ERROR UNKNOWN ERROR NO SOLUTION	There is no solution in either of these cases. Look carefully for more detail about what might have happened.

e) Valeurs des contraintes à la solution

```
---- EQU temps  contraintes sur les temps

     LOWER      LEVEL      UPPER     MARGINAL

A  8330.000  8330.000  8330.000     25.410
B  8330.000  8330.000  8330.000     50.900

                       LOWER      LEVEL      UPPER     MARGINAL

---- EQU prodmax        -INF  2.0000E+5 2.0000E+5      2.251
---- EQU cost            .          .          .         1.000

  prodmax  contrainte de production maximum
  cost  fonction objectif
```

f) Valeurs des variables à la solution

```
---- VAR t  temps passe par la machine i a produire j

       LOWER      LEVEL      UPPER     MARGINAL

A.1        .     1835.841     +INF         .
A.2        .     6494.159     +INF         .
B.1        .          .       +INF      -33.860
B.2        .     8330.000     +INF         .

                      LOWER      LEVEL      UPPER     MARGINAL

---- VAR z              -INF  1.0859E+6      +INF         .

  z  chiffre d'affaire
```

g) DISPLAY

L'instruction

```
display t.l, t.m ;
```

produit la sortie :

```
----     37 VARIABLE  t.L  temps passe par la machine i a produire j

             1           2

A    1835.841    6494.159
B                8330.000


----     37 VARIABLE  t.M  temps passe par la machine i a produire j

             1

B     -33.860
```

D. *INSTRUCTIONS COMPLEMENTAIRES*

1. ALIAS

L'instruction Alias permet de donner des noms différents à un même indice. Par exemple :

Set i indices du vecteur /A, B, C/ ;

Alias (i, i1, i2);

L'ordre des identificateurs entre les parenthèses est indifférent.

2. Expressions logiques - Instruction $

Les **expressions logiques** sont des expressions qui peuvent avoir deux valeurs : VRAI ou FAUX.

Ces expressions utilisent des **opérateurs logiques** qui sont différents des opérateurs utilisés lors de la définition des équations ou lors des affectations. Le tableau suivant présente les différents opérateurs logiques :

Opérateur	Signification
ge	**Supérieur ou égal**
gt	**Strictement supérieur**
le	**Inférieur ou égal**
lt	**Strictement inférieur**
ne	**Différent**
eq	**Egal**
not	**Non**
and	**Et (conjonction)**
or	**Ou (disjonction)**
xor	**Ou exclusif**

Exemple :

(sqr(x) gt x)

Cette expression est fausse si x est compris entre -1 et 1. Sinon, elle est vraie.

L'instruction $ permet de conditionner une « opération » à la véracité d'une condition logique. Ainsi, la syntaxe :

$(condition)

est équivalente à : « si la condition est satisfaite ».

a) Affectation conditionnelle

La syntaxe suivante :

a$(b gt 1.5) = 2;

est équivalente à :

si (b est strictement supérieur à 15) alors (a prend la valeur 2).

Noter la différence entre l'opérateur d'affectation **=** et l'opérateur logique **gt**.

b) Filtrage des indices d'une expression

La syntaxe suivante:

Sum (i $(ord(i) le 4), x(i))

indique que l'on souhaite effectuer la somme des éléments du vecteur **x(i)**, pour les valeurs de l'indice **i** inférieures ou égales à 4.

c) Filtrage des indices d'une équation

La syntaxe suivante :

Equation (k) $(ord(k) gt 2) .. Sum (i,x(i,k)) =j= y(k)+z(k);

indique que l'équation est définie pour les indices **k** strictement supérieurs à 2.

3. OPTION

Il est possible de choisir le solveur que l'on souhaite utiliser. Le tableau suivant donne, pour chaque type de problème, les principaux solveurs disponibles ainsi que la méthode mise en œuvre.

Solveur	**Nature du problème**				**Méthode**
	LP	*NLP*	*MILP*	*MINLP*	
BDMLP	*X*		*X*		Simplex - BB
CPLEX	*X*		*X*		Simplex - BB
CONOPT	*X*	*X*			GRG
MINOS5	*X*	*X*			GRG
DICOPT				*X*	AP/OA/ER

Pour choisir le solveur, l'instruction suivante doit être placée avant l'instruction **solve** :

Option nlp=minos5 ;