UNE INTRODUCTION A GAMS[1]

VÉRONIQUE ROBICHAUD, JANVIER 2017

Introduction

Le logiciel GAMS (*General Algebraic Modeling System*) a été initialement développé par un groupe d'économistes de la Banque mondiale pour faciliter la résolution modèles non linéaires volumineux et complexes sur les ordinateurs personnels. En effet, le GAMS nous permet de résoudre simultanément des systèmes d'équations non linéaires, avec ou sans l'optimisation de certaines fonctions objectif.

Les principaux avantages de GAMS sont (i) la **simplicité** de mise en œuvre, (ii) la **transférabilité** entre utilisateurs et systèmes, et (iii) la **facilité des mises à jour techniques** grâce à l'inclusion régulière de nouveaux algorithmes

À l'origine, le système GAMS était axé sur les fichiers de programme, lesquels devaient être créés en format ASCII, à l'aide avec n'importe quel éditeur de texte, puis exécutés par une commande DOS. Le développement de l'interface GAMS-IDE à la fin des années 1990 a grandement facilité l'utilisation du logiciel. GAMS-IDE fonctionne comme un éditeur de texte compatible avec Windows et permet le lancer et de monitorer la compilation et l'exécution de programmes GAMS. Au cours de cette introduction, nous présenterons la structure générale d'un code GAMS à l'aide d'un exemple, puis nous décrirons les différentes parties du fichier de sortie.

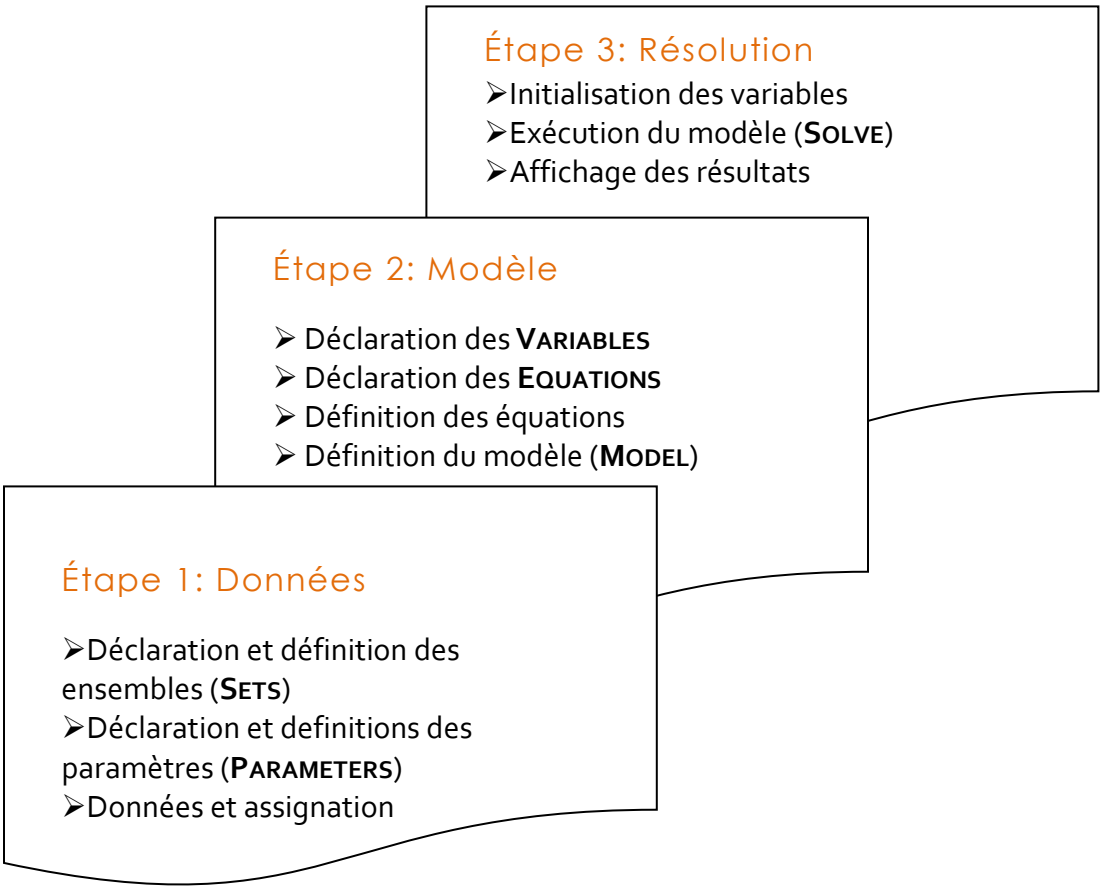
[1] Ce document est une traduction d'un extrait du document pédagogique "GAMS an Introduction", rédigé par Jean-Christophe Dumont et Véronique Robichaud en 2000.

La structure de programmation en GAMS : fichier source, procédures de résolution et éléments de sortie

La structure d'un code GAMS

Généralement, un modèle d'équilibre général calculable (MEGC) programmé à l'aide de GAMS peut être divisé en trois modules: la saisie de données, la spécification du modèle et la résolution. Le diagramme suivant donne une illustration générale de la structure de la syntaxe GAMS. Notez que les mots-clés GAMS apparaissent en gras.

Figure 1: Diagramme de l'organisation typique d'un code GAMS d'un MEGC



Il est important de noter que la déclaration et la définition de chaque élément doivent être complétées avant l'utilisation de ceux-ci dans le modèle (i.e. ensembles, paramètres, variables et équations).

Voici quelques règles générales à respecter lors de la programmation avec GAMS:

- En général, il est nécessaire de déclarer tout élément avant de pouvoir l'utiliser. En particulier, les ensembles doivent être déclarés au tout début du programme.
- Bien que ce ne soit pas toujours nécessaire, il est préférable de prendre l'habitude de terminer une commande par un point-virgule afin d'éviter des erreurs de compilation.
- GAMS ne fait pas la distinction entre les lettres majuscules et minuscules.
- Voici les principales fonctions mathématiques:

Multiplication	*	Égalité dans une opération	=	Logarithme	LOG(.)
Soustraction	-	Sommation	SUM(indice de sommation, élément)	Maximum	MAX(.,.)
Addition	+	Produit	PROD(indice de produit, élément)	Minimum	MIN(.,.)
Division	/	Valeur absolue	ABS(.)		
Exposant	**	Exponentiel	EXP(.)		

Procédure de résolution et éléments de sortie

Une fois le code GAMS écrit de manière appropriée, il doit être sauvegardé en utilisant l'extension. gms. Pour faire résoudre le modèle, il s'agit de choisir la commande « run » dans le menu « File », d'appuyer sur la touche F9, ou de cliquer sur le bouton affichant une flèche rouge.

Le fichier de sortie créé par le processus de résolution portera le même nom que le programme initial mais avec l'extension. lst. Ce fichier contient le programme initial suivi soit de l'identification des erreurs, s'il y en a, soit, des éléments de sortie suivants, lesquels seront discutés ultérieurement :

- Programme initial
- Liste des équations
- Liste des variables
- Statistiques du modèle
- Résumé de résolution
- Résultats

Les deux sections qui suivent illustrent le contenu d'un programme GAMS et sa terminologie. L'exemple qui y est présenté est une adaptation du modèle AUTA présenté dans : Decaluwé, B., A. Martens and L. Savard (2001), "La politique économique du développement et les modèles d'équilibre général calculable. Une introduction", Montréal, Presses de l'Université de Montréal, p.524.

L'écriture d'un programme GAMS

Tous les énoncés et commandes présentés précédemment sont utilisés dans l'exemple ci-dessous, et quelques options additionnelles y sont présentées. Pour plus d'information sur l'une ou l'autre de ces commandes, l'utilisateur peut se référer au guide de l'utilisateur disponible dans le menu "Help"[2] .

Calibrage d'un MEGC

La première partie d'un modèle consiste à introduire les données de référence, lesquelles seront utilisées pour ensuite évaluer les valeurs des paramètres cohérentes avec les données et la structure du modèle. En général, ces données sont issues d'une matrice de comptabilité sociale (MCS).

L'option "Title"

Malgré que ce ne soit pas nécessaire, l'option "Title" permet d'obtenir un fichier de sortie plus élégant. Le texte qui suit la commande **$TITLE** apparaîtra automatiquement en en-tête de chaque page du fichier de sortie. Un sous-titre peut également être ajouter en utilisant l'option **$STITLE**.

Dans l'exemple ci-dessous, `MODELE AUTA` apparaîtra à la première ligne de chaque page du fichier de sortie et `ECONOMIE FERMÉE SANS GOUVERNEMENT` à la deuxième ligne du fichier de sortie. Les lignes qui suivent sont des commentaires : GAMS ne lira pas les ligne débutant par un astérisque (*). L'utilisation de commentaires peut s'avérer utile pour le modélisateur soucieux d'avoir un modèle organise et plus facile à comprendre. Une autre manière d'introduire des commentaires est en plaçant le texte entre les commandes **$ONTEXT** et **$OFFTEXT**. Notons que toutes les options dont la syntaxe début par un $ doivent obligatoirement débutée à la première position de la ligne.

```
$TITLE    MODELE AUTA
$STITLE   ECONOMIE FERMÉE SANS GOUVERNEMENT

* Modèle d'une économie en autarcie, sans gouvernement
* 3 branches et produits, deux ménages.
```

[2] GAMS — A User's Guide. Tutorial by Richard E. Rosenthal c 2006. GAMS Development Corporation, Washington, DC, USA.

Définition des ensembles

La définition d'ensembles est utile pour les variables et paramètres qui ont plusieurs dimensions. Ces ensembles correspondent en fait aux indices utilisés dans l'écriture mathématique du modèle.

```
SETS I Branches et produits
/ AGR agriculture
  MAN manufactures
  SER services /

BNS(I) Biens
/ AGR agriculture
  MAN manufactures /

H Ménages
/ SAL ménages salariés
  CAP ménages capitalistes /

ALIAS (i,j)
;
```

L'énoncé **SETS** définit les ensembles du modèle. Dans notre exemple, nous définissons trois ensemble. Le premier, `I`, contient les éléments AGR, MAN et SER. À la suite de chaque symbole (AGR, MAN et SER) une description peut être ajoutée. Les éléments d'un ensemble sont placés entre deux barres obliques « / ». Le deuxième ensemble, nommé BNS, est en fait un sous-ensemble de l'ensemble I, tel qu'indiqué entre parenthèses, et n'inclut que les deux premiers éléments. Le dernier ensemble représente les ménages. Une dernière commande, ALIAS, permet au modélisateur d'assigner un second indice (ici J) pour référer aux mêmes éléments d'un ensemble déjà défini (ici, l'ensemble I). Ce deuxième nom s'avère très utile lorsqu'une variable ou un paramètre est affecté de plus d'un indice.

Définition des paramètres

Nous définissons ensuite les paramètres. Un « paramètre » est un élément dans une équation dont la valeur ne sera pas affectée suite à une simulation, comme les élasticités, les taux de taxation, les différents coefficients utilisés dans les fonctions etc. En plus de ces paramètres, les variables de « référence » sont caractérisées par leur valeur initiale. Il est d'usage de définir pour chaque variable un paramètre correspondant qui portera le même nom que celle-ci suivi de la lettre « O ». On assignera ensuite à ce paramètre la valeur initiale (ou de référence) pour la variable en question. La définition des paramètres et des variables de référence débute avec la commande **PARAMETERS** et se termine par un point-virgule. Encore une fois, il est utile d'ajouter une description aux paramètres, comme dans notre exemple. Lorsqu'un paramètre est assujetti d'un indice, comme par exemple A_j, celui-ci apparaît entre parenthèses tout de suite après le nom du paramètre, A(j).

```
PARAMETERS

 A(j)             Paramètre d'échelle (Cobb-Douglas - fonction de production)
 aij(i,j)         Coefficient (Leontief - consommation intermédiaire)
 alpha(j)         Élasticité (Cobb-Douglas - fonction de production)
 gamma(i,h)       Part du produit i dans le budget de consommation du ménage h
 io(j)            Coefficient (Leontief - consommation intermédiaire totale)
 lambda           Part du revenu du capital reçue par les ménages capitalistes
 mu(i)            Part du produit i dans l'investissement total
 psi(h)           Propension moyenne à épargner du ménage h
 v(j)             Coefficient (Leontief - valeur ajoutée)

* Définition des variables pour la période de référence
* Variables en volume (quantité)
 CO(i,h)          Consommation du ménage h en produit i
 CIO(j)           Consommation intermédiaire totale de la branche j
 DIO(i,j)         Consommation intermédiaire en produit i par la branche j
 DITO(i)          Demande intermédiaire totale pour le produit i
 INVO(i)          Demande finale en produit i pour fins d'investissement
 KDO(j)           Demande de capital de la branche j
 KSO(j)           Offre de capital dans la branche j
 LDO(j)           Demande de travail de la branche j
 LSO              Offre totale de travail
 VAO(j)           Valeur ajoutée de la branche j
 XSO(j)           Production de la branche

* Prix
 PO(i)            Prix du produit i
 PCIO(j)          Indice de prix des consommations intermédiaires de la branche j
 PVAO(j)          Prix de la valeur ajoutée de la branche j
 RO(j)            Taux de rémunération du capital de la branche j
 WO               Taux de salaire
* Variables nominales (en valeur)
 CTHO(h)          Budget de consommation du ménage h
 DIVO             Dividendes
 ITO              Investissement total
 SFO              Épargne des entreprises
 SHO(h)           Épargne du ménage h
 YFO              Revenu des entreprises
 YHO(h)           Revenu du ménage h
;
```

Traitement des données

Une fois les ensembles et les paramètres définis, les données doivent être saisies. Il existe différente méthode pour inclure des données dans un code GAMS, nous ne verrons que les plus faciles. D'abord, introduisons la MCS à l'aide de la commande **TABLE**. La syntaxe est la suivante : `TABLE` *nom* (*domaine des lignes*, *domaine des colonnes*) *description*. Par exemple:

```
TABLE MCS(*,*) Matrice de comptabilité sociale

          LD       KD      SAL      CAP        F      AGR      MAN      SER      ACC
LD                                                    300      100      200
KD                                                    100      150      100
SAL       600
CAP                210                          70
F                  140
AGR                          162       21               50      150       90       27
MAN                          108       84               20      150       90      173
SER                          270      105               30       75      120
ACC                           60       70       70
TOT       600      350      600      280      140      500      625      600      200
;
```

Le table de notre exemple se nomme "MCS" et a deux dimensions. Lorsque les titres des lignes ou des colonnes ne font pas référence aux éléments d'un ensemble, un astérisque est utilisé. Ici, nous avons introduit toute la MCS en utilisant le nom des éléments des ensembles I et H lorsque pertinent comme titre de ligne et de colonne, ce qui facilitera l'assignation des données. Les valeurs d'un tableau peuvent être exprimées avec ou sans décimales, mais chacune d'elle doit figurer en-dessous d'un seul titre de colonne. Un point-virgule indique la fin du tableau.

Lorsque des données sont introduites à l'aide d'un tableau, le modélisateur doit ensuite établir la correspondance entre celles-ci et les variables de référence du modèle. Dans notre exemple, la valeur de `CO(i,h)` se trouve à l'intersection des lignes dont le titre correspond aux noms des éléments de l'ensemble I et des colonnes dont le titre correspond aux noms des éléments de l'ensemble H. Pour faire référence à une ligne ou une colonne ne correspondant pas à un ensemble, le nom de celle-ci est placé entre guillemets. Par exemple, les la valeur des dividendes (DIVO) se trouve dans le tableau MCS à l'intersection de la ligne 'CAP' et de la colonne 'F'.

```
DIVO              = MCS('CAP','F');
ITO               = MCS('TOT','ACC');
SFO               = MCS('ACC','F');
SHO(h)            = MCS('ACC',h);
YFO               = MCS('TOT','F');
YHO(h)            = MCS('TOT',h);
XSO(j)            = MCS('TOT',j);
CO(i,h)           = MCS(i,h);
DIO(i,j)          = MCS(i,j);
INVO(i)           = MCS(i,'ACC');
KDO(j)            = MCS('KD',j);
LDO(j)            = MCS('LD',j);
```

On peut aussi assigner directement une valeur aux variables de référence ou aux paramètres comme suit. Dans notre exemple, nous donnons la valeur "1" à tous les prix.

```
* Les prix
WO                = 1;
RO(j)             = 1;
PO(i)             = 1;
```

Calibrage des autres variables et des paramètres

Les autres variables et les paramètres du modèle peuvent ensuite être calculés ou calibrés à partir des données saisies jusqu'ici. Chaque calcul peut occuper plus d'une ligne mais doit se terminer par un point-virgule. L'ordre dans lequel les calculs sont effectués est très important puisque GAMS utilisera la dernière valeur assignée à un symbole.

```
* Calcul des variables en volume
LDO(j)            = LDO(j)/WO;
KDO(j)            = KDO(j)/RO(j);
XSO(j)            = XSO(j)/PO(j);
CO(i,h)           = CO(i,h)/PO(i);
INVO(i)           = INVO(i)/PO(i);
DIO(i,j)          = DIO(i,j)/PO(i);

* Calibrage des autres variables
LSO               = SUM[i,LDO(i)];
KSO(j)            = KDO(j);
VAO(j)            = LDO(j)+KDO(j);
PVAO(j)           = {WO*LDO(j)+RO(j)*KDO(j)}/VAO(j);
DITO(i)           = SUM[j,DIO(i,j)];
CIO(j)            = SUM[i,DIO(i,j)];
PCIO(j)           = SUM[i,PO(i)*DIO(i,j)]/CIO(j);
CTHO(h)           = YHO(h)-SHO(h);
```

```
* Calibrage des paramètres
* Production (Cobb-Douglas et Leontief)
 alpha(j)          = WO*LDO(j)/{PVAO(j)*VAO(j)};
 A(j)              = VAO(j)/{LDO(j)**alpha(j)*KDO(j)**(1-alpha(j))};
 v(j)              = VAO(j)/XSO(j);
 io(j)             = CIO(j)/XSO(j);
 aij(i,j)          = DIO(i,j)/CIO(j);

* Paramètres distributifs
 gamma(i,h)        = PO(i)*CO(i,h)/CTHO(h);
 lambda            = {YHO('cap')-DIVO}/SUM[j,RO(j)*KDO(j)];
 mu(i)             = PO(i)*INVO(i)/ITO;
 psi(h)            = SHO(h)/YHO(h);
```

Affichage des paramètres et des variables de référence

Dans le fichier de sortie, la valeur des paramètres n'est pas automatiquement affichée. L'utilisation de la commande **DISPLAY** permet de de les faire afficher. Notons que le nom des paramètres ne doit pas inclure les dimensions.

```
* Paramètres à afficher dans le fichier de sortie
DISPLAY A, alpha, io, v, aij, gamma, psi, mu, lambda;
```

Le modèle

Déclaration des variables

Toutes les variables, endogènes et exogènes, qui apparaissent dans les équations doivent être déclarées. La commande **VARIABLES** débute cette procédure, laquelle se termine par un point-virgule. Suivant le nom d'une variable, une description peut être ajoutée. Dans notre exemple, les variables ont été regroupées par catégorie.

```
VARIABLES

* Variables en volume (quantité)
 C(i,h)            Consommation du ménage h en produit i
 CI(j)             Consommation intermédiaire totale de la branche j
 DI(i,j)           Consommation intermédiaire en produit i par la branche j
 DIT(i)            Demande intermédiaire totale pour le produit i
 INV(i)            Demande finale en produit i pour fins d'investissement
 KD(j)             Demande de capital de la branche j
 KS(j)             Offre de capital dans la branche j
 LD(j)             Demande de travail de la branche j
 LS                Offre totale de travail
 VA(j)             Valeur ajoutée de la branche j
 XS(j)             Production de la branche
```

```
* Prix
 P(i)              Prix du produit i
 PCI(j)            Indice de prix des consommations intermédiaires de la branche j
 PVA(j)            Prix de la valeur ajoutée de la branche j
 R(j)              Taux de rémunération du capital de la branche j
 W                 Taux de salaire

* Variables nominales (en valeur)
 CTH(h)            Budget de consommation du ménage h
 DIV               Dividendes
 IT                Investissement total
 SF                Épargne des entreprises
 SH(h)             Épargne du menage h
 YF                Revenu des entreprises
 YH(h)             Revenu du ménage h

* Autre variable
 LEON              Vérification de la loi de Walras
;
```

Déclaration des équations

Tout comme pour les autres symboles vus jusqu'ici, les équations doivent tout d'abord être déclarées. Cette étape débute par la commande **EQUATIONS** et se termine par un point-virgule. Les équations sont affectées des mêmes indices que les variables qui la composent.

Écriture des équations

La syntaxe pour définir les équations est la suivante. Premièrement, on écrit le nom de l'équation et sa dimension suivi de deux points. Ensuite, on écrit l'expression mathématique suivie d'un point-virgule. Le côté gauche et le côté droit de l'équation sont séparés par **=E=** lorsqu'une égalité stricte doit être respectée, par opposition à **=G=** ou **=L=** qui représentent respectivement « plus grand ou égal » et « plus petit ou égal ». Chaque équation ne peut être défini qu'une seule fois, mais les variables peuvent apparaître dans plusieurs équations et des deux côtés d'une même équation. Ici, l'ordre dans lequel les équations sont définies à peu d'importance puisqu'elles seront résolues de façon simultanée.

Dans notre exemple, nous avons regroupé les équations par catégorie en utilisant plusieurs commandes **EQUATIONS**. Chaque fois que la commande **EQUATIONS** est utilisée, elle doit se terminer par un point-virgule.

```
* Production
EQUATIONS
 XSEQ(j)           Valeur ajoutée dans la branche j (Leontief)
 CIEQ(j)           Consommation intermédiaire totale de la branche j (Leontief)
 VAEQ(j)           Cobb-Douglas entre le travail et le capital
 LDEQ(j)           Demande de travail de la branche j
 KDEQ(j)           Demande de capital de la branche j
 DIEQ(i,j)         Consommation intermédiaire par produit (Leontief)
;
```

```
 CIEQ(j)..        CI(j) =e= io(j)*XS(j);

 VAEQ(j)..        VA(j) =e= A(j)*LD(j)**alpha(j)*KD(j)**(1-alpha(j)) ;

 LDEQ(j)..        W*LD(j) =e= alpha(j)*PVA(j)*VA(j);

 KDEQ(j)..        R(j)*KD(j) =e= (1-alpha(j))*PVA(j)*VA(j);

 DIEQ(i,j)..      DI(i,j) =e= aij(i,j)*CI(j);

* Revenu et épargne
EQUATIONS
 YHSEQ            Revenu des ménages salariés
 YHCEQ            Revenu des ménages capitalistes
 SHEQ(h)          Épargne du ménage h
 CTHEQ(h)         Budget de consommation du ménage h
 YFEQ             Revenu des entreprises
 SFEQ             Épargne des entreprises
;

 YHSEQ..          YH('sal') =e= W*SUM[j,LD(j)];

 YHCEQ..          YH('cap') =e= lambda*SUM[j,R(j)*KD(j)]+DIV;

 SHEQ(h)..        SH(h) =e= psi(h)*YH(h);

 CTHEQ(h)..       CTH(h) =e= YH(h)-SH(h);

 YFEQ..           YF =e= (1-lambda)*SUM[j,R(j)*KD(j)];

 SFEQ..           SF =e= YF-DIV;

* Demande
EQUATIONS
 CEQ(i,h)         Consommation du ménage h en produit i
 INVEQ(i)         Demande en produit i pour fins d'investissement
 DITEQ(i)         Demande intermédiaire en produit i
;

 CEQ(i,h)..       P(i)*C(i,h) =e= gamma(i,h)*CTH(h);

 INVEQ(i)..       P(i)*INV(i) =e= mu(i)*IT;

 DITEQ(i)..       DIT(i) =e= SUM[j,DI(i,j)];
```

```
* Prix
EQUATIONS
 PCIEQ(j)         Indice de prix des consommations intermédiaires de la branche j
 CPEQ(j)          Coûts de production de la branche j
;

 PCIEQ(j)..       PCI(j)*CI(j) =e= SUM[i,P(i)*DI(i,j)];

 CPEQ(j)..        P(j)*XS(j) =e= PVA(j)*VA(j)+PCI(j)*CI(j);


* Équilibre
EQUATIONS
 PEQ(bns)         Absorption domestique
 WEQ              Équilibre sur le marche du travail
 REQ(j)           Équilibre sur le marche du capital
 ITEQ             Équilibre épargne-investissement
;
 PEQ(bns)..       XS(bns) =e= SUM[h,C(bns,h)]+DIT(bns)+INV(bns);

 WEQ..            LS =e= SUM[j,LD(j)];

 REQ(j)..         KS(j) =e= KD(j);

 ITEQ..           IT =e= SUM[h,SH(h)]+SF;

* Autre
EQUATIONS
 WALRAS           Vérification de la loi de Walras
;

WALRAS..          LEON =e= XS('ser')-SUM(h,C('ser',h))-DIT('ser')-INV('ser');
```

L'initialisation

Nous devons maintenant assigner une valeur de départ à chacune des variables du modèle. Afin de résoudre le système d'équations, GAMS utilise comme point de départ la valeur à laquelle nous initialiserons chaque variable. À défaut d'assigner une valeur de départ, GAMS utilisera une valeur de zéro, ce qui pourrait causer des erreurs d'exécution, si une variable apparaît au dénominateur, et compliquerait grandement la recherche de solution, celle-ci se trouvant fort probablement bien loin de zéro. Pour procéder à l'initialisation des variables, le suffixe **.L** (qui signifie « *level* » ou « niveau ») est utilisé comme suit:

```
* Initialisation des variables

C.l(i,h)          = CO(i,h);
CI.l(j)           = CIO(j);
CTH.l(h)          = CTHO(h);
DI.l(i,j)         = DIO(i,j);
DIT.l(i)          = DITO(i);
DIV.l             = DIVO;
INV.l(i)          = INVO(i);
IT.l              = ITO;
KD.l(j)           = KDO(j);
KS.l(j)           = KSO(j);
LD.l(j)           = LDO(j);
LS.l              = LSO;
P.l(i)            = PO(i);
PCI.l(j)          = PCIO(j);
PVA.l(j)          = PVAO(j);
R.l(j)            = RO(j);
SF.l              = SFO;
SH.l(h)           = SHO(h);
VA.l(j)           = VAO(j);
W.l               = WO;
XS.l(j)           = XSO(j);
YF.l              = YFO;
YH.l(h)           = YHO(h);
```

Finalement, le suffixe **.FX** est utilisé pour préciser quelles variables sont exogènes:

```
* Fermetures
* P(AGR) est le numéraire
 P.fx('agr')      = PO('agr');
* Le capital est fixe par branche
 KS.fx(j)         = KSO(j);
* L'offre totale de travail est fixe
 LS.fx            = LSO;
* Les dividendes sont exogènes
 DIV.fx           = DIVO;
```

Dans les deux cas, la syntaxe est similaire: on inscrit le nom de la variable, le suffixe, les indices, le signe d'égalité, le paramètre correspondant à la variable et un point-virgule.

L'exécution du modèle

Pour pouvoir résoudre le modèle, nous devons tout d'abord le définir. Cette procédure débute avec la commande **MODEL**, suivie de nom à donner au modèle, une description facultative, les équations qui doivent être considérées et finalement un point-virgule. Dans notre exemple, le modèle se nomme "AUTA", la description est "Autarcie sans gouvernement" et toutes les équations doivent être considérées, d'où l'utilisation du mot-clé **ALL**.

La seconde ligne définit la procédure de résolution à utiliser. La commande **SOLVE** est suivi du nom du modèle, puis **USING** et la procédure à utiliser pour résoudre le modèle. La procédure est déterminée par le type de problème dont il est question. Par exemple, un modèle linéraire peut être résolu en utilisant la procédure **LP**, alors qu'un problème non-linéaire peut être résolu en suivant **NLP**. Chaque procédure fait appel à un solveur qui lui est propre (MINOS, CONOPT, MILES, ...), et chaque solveur utilise un algorithme particulier. Dans notre cas, nous cherchons à résoudre un système d'équations, dont certaines sont non-linéaires et la procédure adaptée à ce genre de problème est **CNS** (*Constrained non-linear system*).

```
* Exécution du modèle

MODEL AUTA Autarcie sans gouvernement /ALL/;
SOLVE AUTA USING CNS;
```

Lire un fichier de sortie GAMS

Une fois le modèle résolu, GAMS créé un fichier de sortie. Cette section décrit les principales parties du fichier de sortie produit par la résolution du modèle présenté précédemment.

Programme initial

La première partie du fichier de sortie reproduit simplement le code du modèle mais où chaque ligne est numérotée. S'il y a des erreurs dans le modèle, seule cette première partie apparaîtra dans le fichier de sortie. Dans l'exemple qui suit, il manqué une parenthèse dans le calcul du paramètre alpha. Dans le fichier de sortir, quatre astérisques (****) apparaîtront sous la ligne où l'erreur a été détectée, suivis d'un signe de dollar ($) et d'un numéro correspondant au type d'erreur rencontrée.

```
136   alpha(i)         = WO*LDO(i)/{PVAO(i)*VAO(i);
****                                            $8
```

À cause de cette erreur, GAMS n'a pu résoudre le modèle et ainsi, une deuxième erreur est automatiquement indiquée à la toute fin du programme.

```
306  SOLVE AUTA USING CNS;
****                      $257
```

Une description des codes d'erreur est donnée à la toute fin, afin de faciliter la correction:

```
Error Messages
  8  ')' expected
257  Solve statement not checked because of previous errors
```

Dans un programme sans erreur, l'information qui suit apparaît également dans le fichier de sortie.

Affichage facultatif

Dans la partie "calibrage" du modèle, la commande **DISPLAY** a été utilisée pour afficher la valeur des paramètres dans le fichier de sortie. Ces valeurs apparaissent à la suite du programme:

```
----    152 PARAMETER A  Paramètre d'échelle (Cobb-Douglas - fonction de product
                         ion)

AGR 1.755,    MAN 1.960,    SER 1.890


----    152 PARAMETER alpha  Élasticité (Cobb-Douglas - fonction de production)

AGR 0.750,    MAN 0.400,    SER 0.667
```

Liste des équations

La section « *Equation listing* » présente, pour chacune des équations, la valeur de celle-ci lorsque les valeurs des paramètres et celles d'initialisation des variables sont utilisées. À la fin de l'équation, la valeur du côté gauche de l'équation est indiquée entre parenthèses (LHS= " "). Si cette valeur est différente de celle pour le côté droit, GAMS ajoutera quatre astérisques (****). Cette information s'avère fort utile pour déceler tout problème de calibrage des paramètres, d'initialisation ou d'erreur dans l'écriture de l'équation.

```
---- XSEQ  =E=  Valeur ajoutée dans la branche j (Leontief)

XSEQ(AGR)..  VA(AGR) - 0.8*XS(AGR) =E= 0 ; (LHS = 0)

XSEQ(MAN)..  VA(MAN) - 0.4*XS(MAN) =E= 0 ; (LHS = 0)

XSEQ(SER)..  VA(SER) - 0.5*XS(SER) =E= 0 ; (LHS = 0)
```

Liste des variables

Dans la section qui suit « *Column listing* », GAMS affiche l'information équivalente mais cette fois pour chaque colonne au lieu de chaque ligne. Pour chaque variable, les bornes inférieure et supérieure sont indiquées, de même que le niveau initial et la valeur marginale. Suivent ensuite la liste dans équations dans lesquelles apparaissent la variable en question et la valeur du coefficient qui affecte la variable dans l'équation en question. Dans l'exemple ci-dessous, la variable C n'a pas de borne inférieure (donc -INF pour « moins l'infini ») ni de borne supérieure

(+INF) et les niveau d'initialisation sont respectivement de 162, 21 et 108. Cette variables apparaît dans deux équations, CEQ(I,H) et PEQ(I).

```
---- C  Consommation du ménage h en produit i

C(AGR,SAL)
                (.LO, .L, .UP, .M = -INF, 162, +INF, 0)
        1       CEQ(AGR,SAL)
       -1       PEQ(AGR)

C(AGR,CAP)
                (.LO, .L, .UP, .M = -INF, 21, +INF, 0)
        1       CEQ(AGR,CAP)
       -1       PEQ(AGR)

C(MAN,SAL)
                (.LO, .L, .UP, .M = -INF, 108, +INF, 0)
       (1)      CEQ(MAN,SAL)
       -1       PEQ(MAN)
```

Statistiques du modèle

L'objectif de la section « *Model statistics* » est de fournir de l'information concernant la taille et la non-linéarité du modèle. Les « *Blocks* » donnent de l'information sur le nombre d'équations et de variables dans le modèle sans tenir compte des indices. La variable C(I,H), par exemple, constitue un bloc de variable. Les « *Single* » comptent plutôt les variables et équations en tenant compte des indices. La variable C(I,H) compte ici pour six variables puisqu'il y a trois produits et deux ménages. Le « *NON-ZERO ELEMENTS* » indique le nombre de coefficients non nulles. Les autres éléments donnent de l'information sur la non-linéarité et la complexité du modèle. Finalement, cette section informe le modélisateur du temps qu'il a fallu pour générer le modèle (*GENERATION TIME*).

```
MODEL STATISTICS

BLOCKS OF EQUATIONS          22     SINGLE EQUATIONS           58
BLOCKS OF VARIABLES          21     SINGLE VARIABLES           58
NON ZERO ELEMENTS           193     NON LINEAR N-Z             92
DERIVATIVE POOL              20     CONSTANT POOL              24
CODE LENGTH                 195
FIXED EQUATIONS              58     FREE VARIABLES             58


GENERATION TIME      =  0.000E+0000 SECONDS      4 MB  24.6.1 r55820 WEX-WEI
```

Sommaire de résolution

Alors que la section précédente portait sur le processus de résolution, la section « *Solve summary* » concerne la résolution du modèle en tant que tel. La première partie de cette section reprend les informations de la commande SOLVE à savoir : le nom du modèle (ici AUTA), le type de problème à résoudre (CNS) et le solveur utilisé (CONOPT). « SOLVER STATUS » indique comment s'est conclu la résolution. Si celle-ci a été interrompue par l'utilisateur ou par le solveur à cause d'erreurs, c'est ici que cette information apparaîtra. Dans notre exemple, la résolution s'est déroulée normalement, ce qui est indiqué par « *1 NORMAL COMPLETION* ». MODEL STATUS informe l'utilisateur sur la solution du modèle. Le code « 16 Solved » indique qu'une solution au système d'équations a été trouvée.

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

     MODEL   AUTA
     TYPE    CNS
     SOLVER  CONOPT               FROM LINE  334

**** SOLVER STATUS     1 Normal Completion
**** MODEL STATUS      16 Solved
```

Résultats

Dans la section « Solution listing », GAMS affiche la solution qui a été trouvée ligne par ligne, puis colonne par colonne. Pour chaque équation (ligne) et pour chaque variable (colonne) on trouvera la valeur des bornes inférieure et supérieure, et la valeur de la solution (LEVEL). Les bornes sont définies par le modélisateur (dans notre exemple, nous n'avons défini aucune borne) alors que la solution est calculée par le solveur. Un point "." signifie zéro. Voici un exemple pour les équations:

```
---- EQU XSEQ  Valeur ajoutée dans la branche j (Leontief)

       LOWER     LEVEL     UPPER

AGR      .         .         .
MAN      .         .         .
SER      .         .         .
```

Et pour les variables :

```
---- VAR C  Consommation du ménage h en produit i

            LOWER      LEVEL      UPPER

AGR.SAL      -INF    162.000      +INF
AGR.CAP      -INF     21.000      +INF
MAN.SAL      -INF    108.000      +INF
MAN.CAP      -INF     84.000      +INF
SER.SAL      -INF    270.000      +INF
SER.CAP      -INF    105.000      +INF
```