

An Introduction to GAMS¹

Véronique Robichaud

JULY 2010

Introduction

The GAMS software (*General Algebraic Modeling System*) was originally developed by a group of economists from the World Bank in order to facilitate the resolution of large and complex non linear models on personal computer. As a matter of fact, GAMS allows solving simultaneous non linear equation system, with or without optimization of some objective function.

(i) **Simplicity** of implementation, (ii) **portability and transferability** between users and systems and (iii) **easiness of technical update** because of the constant inclusion of new algorithms are the main advantages of GAMS.

The seminal GAMS system was file oriented. The program must be created in ASCII format with any one of the usual text editor run by a DOS command.

The development of GAMS-IDE interface in the late 1990s makes it even easier to use. GAMS-IDE works as a general text editor compatible with WINDOWS and offers the ability to launch and monitor the compilation and execution of typical GAMS programs. In this introduction note we will present the general structure of the GAMS program, followed by a detailed illustration, including a description of the output file.

Architecture of GAMS programming: input file, resolution procedure and output settings

The structure of the input GAMS Code

Typically, a computable general equilibrium (CGE) model programmed in GAMS can be decomposed in three modules corresponding respectively to data entry, model specification and solve procedure (see Figure 1 below). The following chart gives an overall illustration of the structure of the GAMS syntax. Note that the standard GAMS key words appear in bold.

It is important to note that **declaration and definition**, or assignment, must be completed for each of the element in use in the model (i.e. sets, parameters, variables and equations).

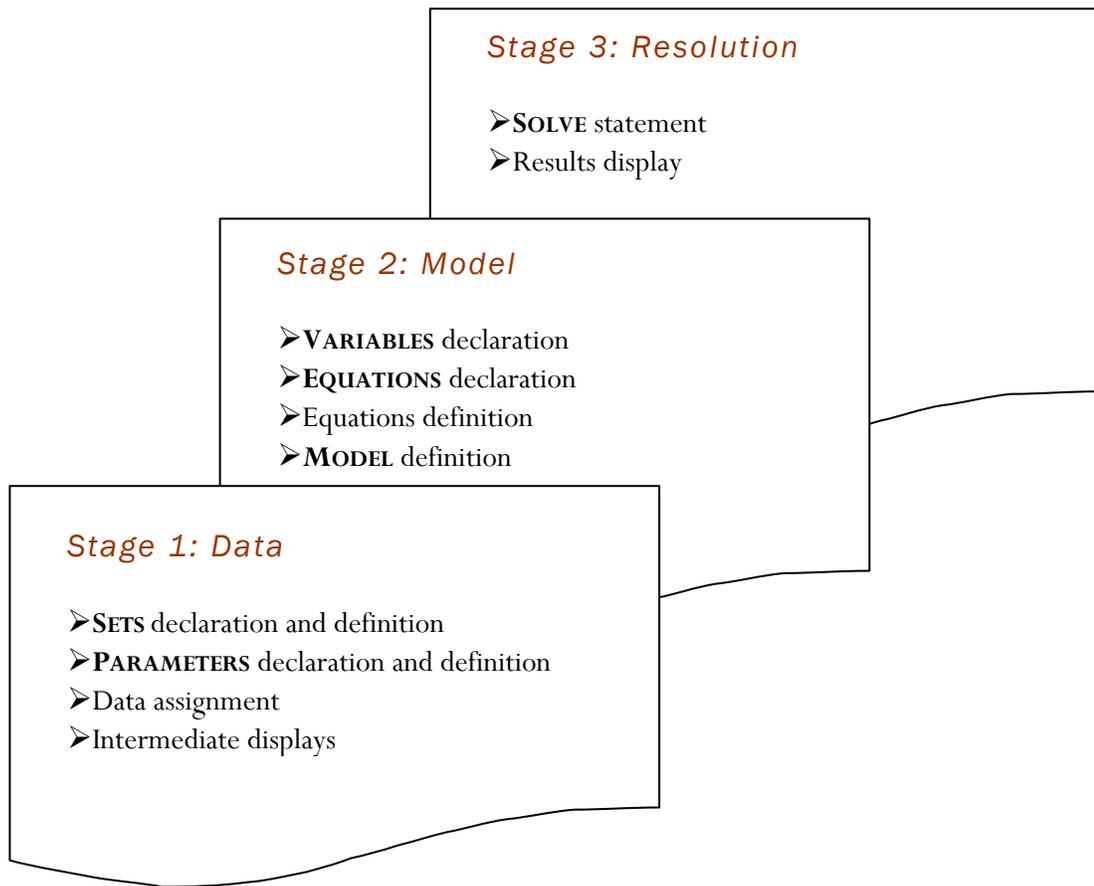
Some general rules must be set apart concerning GAMS programming:

¹ This document is an extract from “GAMS an Introduction”, a pedagogical paper written by Jean-Christophe Dumont and Véronique Robichaud in 2000.

- As a general rule, it is necessary to proceed to the declaration of any element before using it. In particular, sets must be declared at the very beginning of the program.
- Even if it is not always necessary, a good habit is to end every statement by a semicolon in order to avoid unexpected compilation error.
- GAMS allows for statement on several lines or several statement on the same line. This property can help to reduce the length of the code or facilitate printing.
- Capital and small letters are not distinguished in GAMS.
- Finally the main mathematical functions are described bellow:

Multiplication	*	Equality in an operation	=	Logarithm	<i>LOG(.)</i>
Subtraction	-	Summation	<i>SUM(set domain, element)</i>	Maximum	<i>MAX(.,.)</i>
Addition	+	Product	<i>PROD(set domain, element)</i>	Minimum	<i>MIN(.,.)</i>
Division	/	Absolute Value	<i>ABS(.)</i>		
Exponent	**	Exponential	<i>EXP(.)</i>		

FIGURE 1: ORGANIZATION CHART OF A TYPICAL GAMS CODE FOR CGE MODELS



Resolution procedure and output settings

Once the GAMS code has been written in a proper manner it must be saved with the extension **.gms**. To run the code, one can either choose the run command in the file menu, press F9 or click on the red arrow button.

The output file created by this process has the same prefix as the initial program but gets the extension **.lst**. It contains either the initial program with compilation error identification if any, either the default output settings as described below.

Output in the file 'name.lst'

- Echo print of the initial program
- Equation listing
- Column listing
- Model statistics
- Solve summary
- Results

The next two sections will give an illustration of the content and the nomenclature of GAMS programming. The example is taken from a small theoretical CGE Model (Model 0).

Writing a GAMS Program

All statements previously presented are used in a complete GAMS program and some additional options are also introduced. For further information on any of these commands, the reader should refer to the GAMS User's Guide² available under the help menu.

Calibration of a CGE model

The first part of the modeling process consists in entering data, which will be referred as benchmark data, and evaluating parameters consistent with these values. This process is called the *calibration*. In the case of CGE modeling, the benchmark data are usually drawn from the Social Accounting Matrix (SAM). Parameters that can be calibrated will be determined with those.

TITLE OPTION

Although not necessary, the title option allows a more comprehensible output listing. The text following the **\$TITLE** command will appear as a header on every page of the output listing. A subtitle can also be added to this header using the command **\$STITLE**.

```
$TITLE                                MODEL AUTA
```

² Brooke A., D. Kendrick & A. Meeraus (1998), A. *GAMS. A user's Guide*. The Scientific Press, San Francisco.

```

$STITLE                                AUTARKY WITHOUT GOVERNMENT
* Model of a closed economy without government producing 3 goods using
* 2 factors owned by 2 types of households.

```

In this example, `MODEL AUTA` will appear as the first line of each page in the output file and `AUTARKY WITHOUT GOVERNMENT` on the second one. The following lines are comments; `GAMS` will not read lines beginning with an asterisk (*). The use of comments can be useful to the modeller in order to organize its program in a comprehensive manner. Another way to insert comments is to put the text between the **\$ONTEXT** and **\$OFFTEXT** command.

SET DEFINITION

The definition of sets is useful for multidimensional variables. It corresponds to the indexes in mathematical representations of models.

```

SET I Sectors
/ AGR agriculture
  IND industry
  SER services /

GOOD(I) Goods
/ AGR agriculture
  IND industry /

H Households
/ HW labour endowed households
  HC capital endowed households /

ALIAS (I, J)

```

The **SET** statement defines the set of sectors. In this example, three sets are defined. The first one, denoted by the letter I, contains elements are AGR, IND and SER. After each symbol (AGR, IND and SER) a description can be added. The elements of a set are placed between slashes (/). The second set, named GOOD, is in fact a subset of I, indicated by the (I) part and includes only the two first elements. The last set represents households. A final command, ALIAS, is used allow the modeller to use a second index (J) that represents exactly the same elements as the set I. This is useful when using multi-indexed variables.

PARAMETERS DEFINITION

The parameters should now be defined. Parameters are the elements in the equations that will not change after a simulation, such as elasticity, tax rates, distribution and scale coefficients. In addition to these parameters, benchmark variables are also defined for their value at the base year will not change after simulation. A common way to define these variables is to add an "O" after the variable name so it will not be confused with the 'true' variable. Parameters and benchmark variables definition begins with the statement **PARAMETER** and end with a semicolon. Once again, it is useful to put a description after the parameter designation, as it is done in the example. When a parameter is subject to an index, like A_j , the set over which it is defined is put between parentheses, A(j).

PARAMETER

A(j)	Scale coefficient (Cobb-Douglas production function)
aij(i,j)	Input output coefficient
alpha(j)	Share coefficient (Cobb-Douglas production function)
gamma(i,h)	Share of the value of commodity i in total consumption
io(j)	Coefficient (Leontief total intermediate consumption)
lambda	Share of capital income received by capitalists
mu(i)	Share of the value of commodity i in total investment
psi(h)	Propensity to save for household h
v(j)	Coefficient (Leontief value added)

* Variables definition for the base year

CO(i,h)	Consumption of commodity i by type h households
CIO(j)	Total intermediate consumption of industry j
DIO(i,j)	Intermediate consumption of commodity i in industry j
DITO(i)	Total intermediate demand for commodity i
DIVO	Dividends
INVO(i)	Final demand of commodity i for investment purposes
ITO	Total investment
KDO(j)	Sector j demand for capital
LDO(j)	Industry j demand for labour
LSO	Total labour supply
PO(i)	Price of commodity i
PVAO(j)	Value added price for industry j
RO(j)	Rate of return to capital from industry j
SFO	Firms' savings
SHO(h)	Savings of type h household
VAO(j)	Value added of industry j
WO	Wage rate
XSO(j)	Output of activity j
YFO	Firms' income
YHO(h)	Income of type h household

;

DATA HANDLING

Once the sets, parameters and benchmark variables are defined, data must be entered. This can be done using the **TABLE** command, which is useful for multidimensional variables. The syntax in the following: **TABLE name(row domain, column domain) description**. Here is an example:

TABLE SAM(*,*) Social accounting matrix

	LD	KD	HW	HC	F	AGR	IND	SER	ACC
LD						5760	7560	15540	
KD						1440	11340	5720	
HW	28860								
HC		11100			1900				
F		7400							
AGR			4329	650		120	2526.9	275.5	1098.6
IND			11544	3900		1544	21709.1	5815.5	9887.4
SER			10101	5850		136	11264	3349	
ACC			2886	2600	5500				
TOT	28860	18500	28860	13000	7400	9000	54400	30700	10986

;

The table is named SAM and has two dimensions. When the column/row labels do not refer to any specific set, the asterisk is used. Here, we have simply put the entire SAM but using the elements of the sets, when relevant, as the title of each column and row. This will greatly facilitate data assignment. Values can be expressed without or with decimals but must fall under only one column title. A semicolon indicates the end of the table.

When data is entered in a table, it then has to be assigned to the relevant variable. In the example, the value of $CO(i, h)$ can be found in the table SAM at the intersection of the row named i and the column h . When referring to a row or a column that does not represent an element of a defined set, brackets are used. For example, the value of dividends (DIVO) is given in table SAM at the intersection of row 'HC' and column 'F'.

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

Values for variables and parameters can also be entered simply as follows. In our example, all prices are set to one.

```
WO                = 1;
RO(j)             = 1;
PO(i)             = 1;
```

CALCULATION OF VARIABLES AND CALIBRATION

Other variables and parameters can be calculated or calibrated from the ones entered previously. At the end of each calculation, a semicolon is put. The sequence in which they are calculated is very important because GAMS will use the last value assigned to a variable or a parameter.

```
LDO(i)            = LDO(i)/WO;
KDO(i)            = KDO(i)/RO(i);
XSO(i)            = XSO(i)/PO(i);
CO(i, h)          = CO(i, h)/PO(i);
INVO(i)           = INVO(i)/PO(i);
DIO(i, j)         = DIO(i, j)/PO(i);

DITO(i)           = SUM[j, DIO(i, j)];
CIO(J)            = SUM[i, DIO(i, j)];
VAO(i)            = LDO(i)+KDO(i);
LSO               = SUM[i, LDO(i)];
PVAO(i)           = {PO(i)*XSO(i)-SUM[j, PO(J)*DIO(j, i)]}/VAO(i);
```

```

* Calibration of parameters
* Production (Cobb-Douglas and Leontief)
alpha(i)      = WO*LDO(i)/{PVAO(i)*VAO(i)};
A(i)          = VAO(i)/{LDO(i)**alpha(i)*KDO(i)**(1-alpha(i))};
io(i)         = CIO(i)/XSO(i);
v(i)          = VAO(i)/XSO(i);

* Share parameters
gamma(i,h)    = PO(i)*CO(i,h)/YHO(h);
psi(h)        = SHO(h)/YHO(h);
mu(i)         = PO(i)*INVO(i)/ITO;
aij(i,j)      = DIO(i,j)/CIO(J);
lambda        = lambda/SUM[i,RO(i)*KDO(i)];

```

DISPLAY OF PARAMETERS AND VARIABLES

In the output listing, values of parameters are not automatically printed. The **DISPLAY** option will allow displaying these values. Note that the names of parameters and benchmark variables are entered without dimension.

```
DISPLAY A, alpha, io, v, aij, gamma, psi, mu, lambda;
```

The model

VARIABLE DECLARATION

All variables, endogenous or exogenous, appearing in the equations must first be declared. The statement **VARIABLES** begins this procedure and ends with a semicolon. Following the variable name, for example W, a brief description can be added.

VARIABLES

```

C(i,h)        Consumption of commodity i by type h households
CI(j)         Total intermediate consumption of industry j
DI(i,j)       Intermediate consumption of commodity i in industry j
DIT(i)        Total intermediate demand for commodity i
DIV           Dividends
INV(i)        Final demand of commodity i for investment purposes
IT            Total investment
KD(j)         Sector j demand for capital
LD(j)         Industry j demand for labour
LS            Total labour supply
P(i)          Price of commodity i
PVA(j)        Value added price for industry j
R(j)          Rate of return to capital from industry j
SF            Firms' savings
SH(h)         Savings of type h household
VA(j)         Value added of industry j
W             Wage rate
XS(j)         Output of activity j
YF            Firms' income
YH(h)         Income of type h household

```

```
LEON           Walras law verification variable
;
```

EQUATIONS DECLARATION

As with the previous components of the model, the equation must be declared and defined. This step begins with the **EQUATIONS** statement followed by the symbols for which a description can be added. For example, the equation named EQ1(j) is followed by a short description. When all equations are declared, a semicolon indicates the end.

EQUATIONS

```
* Production
EQ1(j) Value added demand in industry i (Leontief)
EQ2(j) Total intermediate consumption demand in industry j (Leontief)
EQ3(j) Cobb-Douglas between labour and capital
EQ4(j) Demand for labour by industry j
EQ5(j) Demand for capital by industry j
EQ6(i,j) Intermediate consumption of commodity i by sector j

* Income and savings
EQ7 Household income (workers)
EQ8 Household income (capitalists)
EQ9(h) Household h savings
EQ10 Firms income
EQ11 Firms savings

* Demand
EQ12(i,h) Household h consumption of commodity i
EQ13(i) Investment in commodity i
EQ14(i) Intermediate demand for commodity i

* Prices
EQ15(j) Output price for industry j

* Equilibrium
EQ16(good) Domestic absorption
EQ17 Labour market equilibrium
EQ18 Investment-savings equilibrium

* Other
WALRAS Verification of the Walras law
;
```

EQUATIONS ASSIGNMENT

The syntax for equation definition is the following. First the name and dimension is stated followed by two dots. Then the equation is defined and ends with a semicolon. The **=E=** between the left hand side and the right hand side of the equation simply means "equals to" as opposed to **=G=** or **=L=** which respectively represent "greater than" and "less than". Each equation can only be defined once but the variables can appear in several equations and on both sides of the same

equation. Here, the order in which the equations are defined does not matter since all equations are solved simultaneously.

```

* Production
EQ1(j)..      VA(j) =e= v(j)*XS(j);
EQ2(j)..      CI(j) =e= io(j)*XS(j);
EQ3(j)..      VA(j) =e= A(j)*LD(j)**alpha(j)*KD(j)**(1-alpha(j)) ;
EQ4(j)..      W*LD(j) =e= alpha(j)*PVA(j)*VA(j);
EQ5(j)..      R(j)*KD(j) =e= (1-alpha(j))*PVA(j)*VA(j);
EQ6(i,j)..    DI(i,j) =e= aij(i,j)*CI(J);

* Income and savings
EQ7..         YH("HW") =e= W*SUM[i,LD(i)] ;
EQ8..         YH("HC") =e= lambda*SUM[j,R(j)*KD(j)] + DIV;
EQ9(h)..      SH(h) =e= psi(h)*YH(h);
EQ10..        YF =e= (1-lambda)*SUM[j,R(j)*KD(j)];
EQ11..        SF =e= YF-DIV;

* Demand

EQ12(i,h)..   P(i)*C(i,h) =e= gamma(i,h)*YH(h);
EQ13(i)..     P(i)*INV(i) =e= mu(i)*IT;
EQ14(i)..     DIT(i) =e= SUM[j,DI(i,j)];

* Prices
EQ15(j)..     P(j)*XS(j) =e= PVA(j)*VA(j)+SUM[i,P(i)*DI(i,j)];

* Equilibrium
EQ16(GOOD)..  XS(good) =e= SUM[h,C(good,h)]+DIT(good)+INV(good);
EQ17..        LS =e= SUM[j,LD(j)];
EQ18..        IT =e= SUM[h,SH(h)]+SF;

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

```

INITIALIZATION PROCEDURE

For all variables declared previously, a value must now be assigned. In order to solve the equations, GAMS has to start from a benchmark value. If no value is associated to a variable, GAMS will use zero, which could cause problems if the variable appears at the denominator of a division. For the endogenous variables, that is the ones whom values will be affected by a simulation, the suffix **.L** (for level) is used as follow:

```

C.l(i,h)      = CO(i,h);
CI.l(i)       = CIO(i);
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);
LD.l(j)       = LDO(j);
LS.l          = LSO;

```

```

P.l(i)           = PO(i);
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);

```

For the exogenous variables, that is the ones whom values will not be affected by a simulation, the suffix **.FX** (for fixed) is used:

```

* P(AGR) is the numeraire, capital is sector specific
P.fx('agr')     = PO('agr');
LS.fx           = LSO;
KD.fx(i)        = KDO(i);
DIV.fx          = DIVO;

```

In both cases, the syntax is very similar that is, the name of the variable, the suffix, the variable dimension, the equal sign (=), the benchmark value defined in the calibration process and a semicolon.

MODEL EXECUTION

Finally, two lines must be added in order to solve the equation system. The first one defines the model. It begins with the **MODEL** statement, followed by a name (AUTA in the example), a brief description (AUTARKY WITHOUT GOVERNMENT), the list of equations to be resolved between slashes and a semicolon. When all equations are to be solved the key word **ALL** is put instead of the complete list of equations.

```
MODEL AUTA AUTARKY WITHOUT GOVERNMENT /ALL/;
```

The second line defines the procedure to be used in resolving the model. The **SOLVE** statement is followed by the name of the model, **USING** and the procedure to be used to resolve the model. The procedure to be used is determined by the type of model to solve. For example, linear programs can be solved using the **LP** procedure, while non-linear programs can be solved using **NLP**. Each procedure uses a different solver (MINOS, CONOPT, MILES, ...). These solvers use different algorithms that can be efficient in resolving some systems but less in other cases. In our example, the **CNS** procedure is used.

```
SOLVE AUTA USING CNS;
```

Reading a GAMS output file

Once GAMS has solved an equation system, it creates an output file. In this section the output file of the program presented in the previous section is described.

Echo print

The first part of the output file is simply the reproduction of the model with line numbers on the left hand side, for further references in the file. If there are errors in the program, this is the only part of the output that will appear. Here is an example. There is a parenthesis missing at the end of the calculation of the parameter alpha. In the echo print of the initial program, GAMS will put four asterisks (****) below the line where a mistake has been detected followed by a dollar sign (\$) and a number corresponding to a specific error.

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

Because of this error, GAMS was unable to solve the model, thus another error is put on the last line of the program.

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

At the end the echo print of the program, a description of the error code is given in order to facilitate the correction:

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

When no errors are encountered, the following information also appears in the listing.

Displayed parameters

In the calibration process, the **DISPLAY** statement was used in order to print the value of some parameters in the output file. These values appear after the echo print as follow:

```
---- 150 PARAMETER A Scale coefficient (Cobb-Douglas production function)
AGR 1.649,   IND 1.960,   SER 1.790

---- 150 PARAMETER alpha Share coefficient (Cobb-Douglas production function
)
AGR 0.800,   IND 0.400,   SER 0.731
```

Equation listing

This section presents, for each equation, the specific instance of the model that is created when actual values of the sets and parameters are used. At the end of the equation, the value of the left hand side is put between parenthesis (LHS= " "). When the value of the left hand side differs from the one on the right hand side, GAMS add four asterisks (****). This indication can be useful in finding problems linked to incorrect parameter calibration, or improper initialization of variable.

```
Equation Listing  SOLVE AUTA Using CNS From line 306
```

```

---- EQ1  =E= Value added demand in industry i (Leontief)

EQ1 (AGR).. VA(AGR) - 0.8*XS(AGR) =E= 0 ; (LHS = 0)

EQ1 (IND).. VA(IND) - 0.347426470588235*XS(IND) =E= 0 ; (LHS = 0)

EQ1 (SER).. VA(SER) - 0.692508143322476*XS(SER) =E= 0 ; (LHS = 0)

```

Column listing

Following the equation listing, GAMS prints the equivalent information by column instead of row. For each variable, the lower and upper bounds, the level calculated by the solver and the value of the coefficient affecting the actual variable in each equation are given. If the coefficient is non-linear, it is put between parentheses. In the example below, consumption (noted C) has no lower bounds (thus -INF for minus infinity), no upper bounds (+INF) and the levels that satisfy equilibrium are 4,329, 650 and 11,544. The variable appears in two equations, EQ12(I,H) and EQ16(I).

```

Column Listing          SOLVE AUTA Using CNS From line 306

---- C Consumption of commodity i by type h households (volume)

C (AGR,HW)
      (.LO, .L, .UP, .M = -INF, 4329, +INF, 0)
      1      EQ12 (AGR,HW)
      -1     EQ16 (AGR)

C (AGR,HC)
      (.LO, .L, .UP, .M = -INF, 650, +INF, 0)
      1      EQ12 (AGR,HC)
      -1     EQ16 (AGR)

C (IND,HW)
      (.LO, .L, .UP, .M = -INF, 11544, +INF, 0)
      (1)    EQ12 (IND,HW)
      -1     EQ16 (IND)

```

Model statistics

The purpose of this section is to provide information on the size and non-linearity of the model. The BLOCK gives the number of equations and variables in the model, in our example 19 and 21 respectively, while SINGLE gives the number of rows and column generated by the solver, here 50 and 56. The NON-ZERO ELEMENTS count refers to the number of non-zero coefficients. The four remaining entries give information about the non-linearity of the model. The NON LINEAR N-Z indicates the number of nonlinear matrix entries in the model. The CODE LENGH and the CONSTANT POOL give the user the level of complexity on the non-linearity. Finally, this section indicates the time used to generate the model (GENERATION TIME).

```

MODEL STATISTICS

BLOCKS OF EQUATIONS          19      SINGLE EQUATIONS          50
BLOCKS OF VARIABLES         21      SINGLE VARIABLES          56
NON ZERO ELEMENTS           179     NON LINEAR N-Z           91

```

DERIVATIVE POOL	16	CONSTANT POOL	25
CODE LENGTH	512		
FIXED EQUATIONS	50	FREE VARIABLES	50

GENERATION TIME = 0.015 SECONDS 4 Mb WEX233-233 Nov 17, 2009

Solve Summary

While all previous information referred to the solving process, the followings concern the solution itself. The first section on this matter is the solve summary. It recalls the name of the model (here THEORY), the type of model (CNS) and the solver used (CONOPT). The SOLVER STATUS indicates the way the solver terminated. If the resolution has been interrupted by the user or by the solver (because of errors) this is where it will be indicated. In our example, the solver terminated in a normal way, which is indicated by 1 NORMAL COMPLETION. While the SOLVER STATUS refers to the resolution process, the MODEL STATUS refers to the solution itself. The code 1 OPTIMAL indicates that the solution is optimal. The RESOURCE USAGE, LIMIT, gives information about the time, in seconds, taken by the solver and the upper limit allowed for the solver. The default upper bound is 1000 but can be changed by the user. The ITERATION COUNT, LIMIT provides the number of iterations used by the solver and the upper limit (default being 10,000). Finally, EVALUATION ERRORS indicates the number of errors encountered by the solver and the maximum error allowed (default being 0).

```

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

MODEL   AUTA
TYPE    CNS
SOLVER  CONOPT                FROM LINE 306

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

RESOURCE USAGE, LIMIT      0.016      1000.000
ITERATION COUNT, LIMIT     2      2000000000
EVALUATION ERRORS          0          0

```

Solution Listing

In this section, GAMS lists the solution found by the solver row-by-row then column-by-column. For each equation and each variable it indicates the lower and upper bounds, the level and the marginal. While the bounds are defined by the modeller, the level and the marginal are calculated by the solver. The dot "." means zero. Here is an example of the equation list:

```

---- EQU EQ1  Value added demand in industry i (Leontief)

      LOWER      LEVEL      UPPER

AGR      .          .          .
IND      .          .          .
SER      .          .          .

---- EQU EQ2  Total intermediate consumption demand in industry j (Leontief)

```

	LOWER	LEVEL	UPPER
AGR	.	.	.
IND	.	.	.
SER	.	.	.

And here is an example of the variable listing:

---- VAR C Consumption of commodity i by type h households (volume)

	LOWER	LEVEL	UPPER
AGR.HW	-INF	4329.000	+INF
AGR.HC	-INF	650.000	+INF
IND.HW	-INF	11544.000	+INF
IND.HC	-INF	3900.000	+INF
SER.HW	-INF	10101.000	+INF
SER.HC	-INF	5850.000	+INF

Report Summary

The report summary section indicates the count of rows or column that GAMS marked with the following remarks and the counts of errors.

```
**** REPORT SUMMARY :           0 INFEASIBLE
                               0 DEPENDENT
                               0 ERRORS
```

File Summary

The last part of the output file indicates the names of the input and output files. In the following example, the input file name is AUTA.GMS and the output file is AUTA.LST and files are located in the same directory.

```
**** FILE SUMMARY

Input      C:\auta.gms
Output     C:\auta.lst
```