DEBUGATOR

How to debug a Computable General Equilibrium Model using GAMS

February 2012

Hélène Maisonnave [1]
Bernard Decaluwé [2]
Véronique Robichaud [3]
André Lemelin [4]

This work is a joint effort between IFPRI and PEP in the context of the AGRODEP project, and the authors acknowledge the financial support of the AGRODEP project.

[1] CIRPÉE et PEP, Université Laval, Québec

[2] Département d'économique, Université Laval, Québec

[3] CIRPÉE et PEP, Université Laval, Québec

[4] Centre INRS-UCS, Université du Québec, Montréal

DEBUGATOR

How to debug a Computable General Equilibrium Model using GAMS

This document provides some hints to debug a Computable General Equilibrium (CGE) model written in GAMS[5]. The aim of this document is to provide a methodology to help the readers debug their own errors. The document tries to be systematic and pedagogical. The model used for all the applications is PEP1-1 by Decaluwé et al (2010), available online at http://www.pep-net.org/programs/mpia/pep-standard-cge-models/pep-1-1-single-country-static-version/

Some errors have been introduced in the model. They are a sample of some the types of mistakes that may be encountered while running the GAMS code, such as compilation errors, execution, calibration and specification errors. This document is not an exhaustive list of all the different problems that may be encountered. The intention is to guide the reader on how to debug his or her code using hypothetical examples.

Before reading this document, we strongly recommend that the reader start with the description of PEP 1-1[6], as well as the User Guide of PEP 1-1[7].

There are 20 GAMS files that accompany this document, each corresponding to the 20 examples that are presented here. This should make it easier for the reader to follow all the steps developed in the processes.

This document is funded by AGRODEP and the PEP network. It was developed in a pedagogical perspective to help the researchers of both networks. Needless to say, we welcome comments that will help us improve and facilitate the use of this document. Readers are invited to send their comments to Hélène Maisonnave at the following address: hmaisonnave@hotmail.fr

[5] For a document that deals with using GAMS in general (not specifically applied to CGE), please refer to McCarl(2009) or Rosenthal (2008)

[6] See Decaluwé et al (2010)

[7] See Robichaud et al (2011)

As previously mentioned, this document is intended to be pedagogical and systematic. The different mistakes are presented in the order of their occurrence. Compilation errors are the first type of errors the user will encounter. Following are some examples of execution errors that may occur once all the compilation errors are corrected. Then, when the model has successfully run (meaning neither compilation nor execution errors have occurred), we will present some examples of calibration errors. Finally, once the model replicates the base year successfully, we want to check if the model is correctly specified before running the simulations we are interested in. We will present three different examples leading to erroneous results, due to the model not being correctly specified.

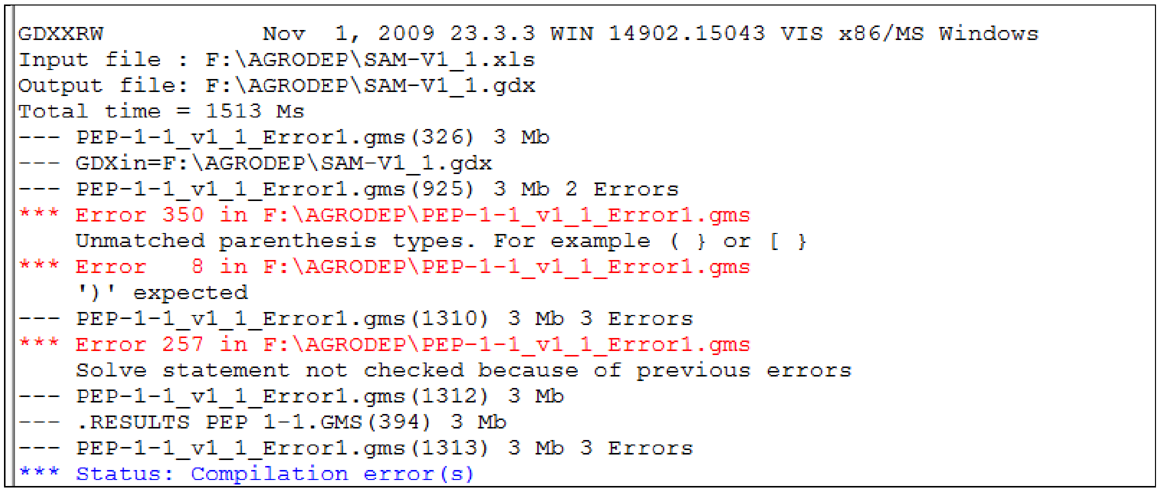
1. Compilation errors

Following are 7 different compilation errors. For each example, only one error has been introduced in the PEP 1-1 model.

Example 1:

A run of the file PEP-1-1_v1_1_Error1.gms yields the following process window:

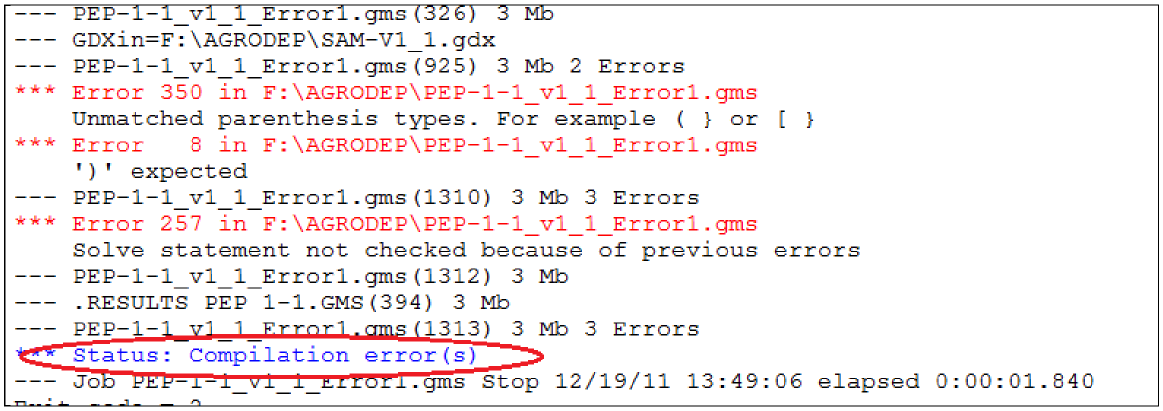
Figure 1 : Process window of example 1



```
GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 1513 Ms
--- PEP-1-1_v1_1_Error1.gms(326) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error1.gms(925) 3 Mb 2 Errors
*** Error 350 in F:\AGRODEP\PEP-1-1_v1_1_Error1.gms
    Unmatched parenthesis types. For example ( } or [ }
*** Error   8 in F:\AGRODEP\PEP-1-1_v1_1_Error1.gms
    ')' expected
--- PEP-1-1_v1_1_Error1.gms(1310) 3 Mb 3 Errors
*** Error 257 in F:\AGRODEP\PEP-1-1_v1_1_Error1.gms
    Solve statement not checked because of previous errors
--- PEP-1-1_v1_1_Error1.gms(1312) 3 Mb
--- .RESULTS PEP 1-1.GMS(394) 3 Mb
--- PEP-1-1_v1_1_Error1.gms(1313) 3 Mb 3 Errors
*** Status: Compilation error(s)
```

The status indicates that there are compilation errors, meaning that GAMS could not solve the model due to writing errors.

Figure 2 : Process window of example 1 (2)

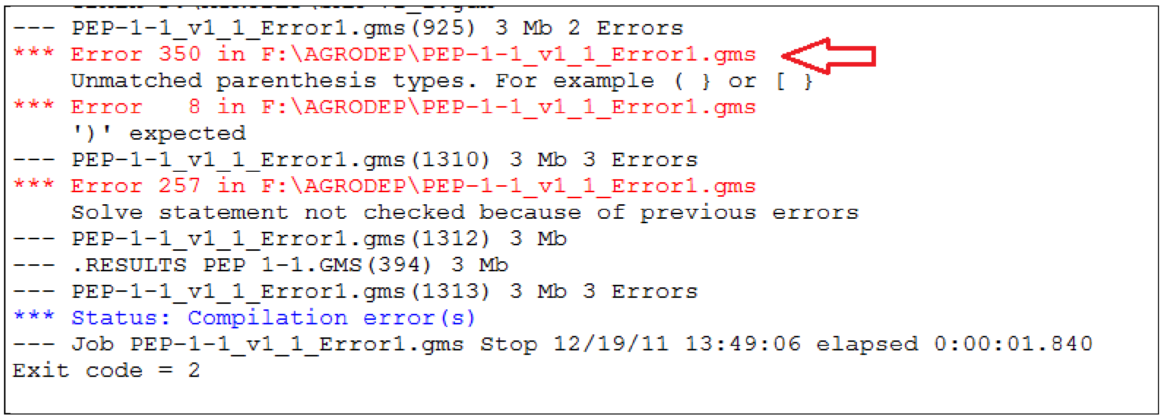


```
--- PEP-1-1_v1_1_Error1.gms(326) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error1.gms(925) 3 Mb 2 Errors
*** Error 350 in F:\AGRODEP\PEP-1-1_v1_1_Error1.gms
    Unmatched parenthesis types. For example ( } or [ }
*** Error   8 in F:\AGRODEP\PEP-1-1_v1_1_Error1.gms
    ')' expected
--- PEP-1-1_v1_1_Error1.gms(1310) 3 Mb 3 Errors
*** Error 257 in F:\AGRODEP\PEP-1-1_v1_1_Error1.gms
    Solve statement not checked because of previous errors
--- PEP-1-1_v1_1_Error1.gms(1312) 3 Mb
--- .RESULTS PEP 1-1.GMS(394) 3 Mb
--- PEP-1-1_v1_1_Error1.gms(1313) 3 Mb 3 Errors
*** Status: Compilation error(s)
--- Job PEP-1-1_v1_1_Error1.gms Stop 12/19/11 13:49:06 elapsed 0:00:01.840
```

GAMS gives an explanation of the possible origin of the error in order to help the modeller to debug the model. Here there are 3 errors written in red. The last one (*Error 257*) is not actually an error, it only indicates that the model did not run because of previous errors.

The best way to debug the model is to double-click on the very first error in the process window, as shown in the figure 3:

Figure 3 : Process window of example 1 (3)



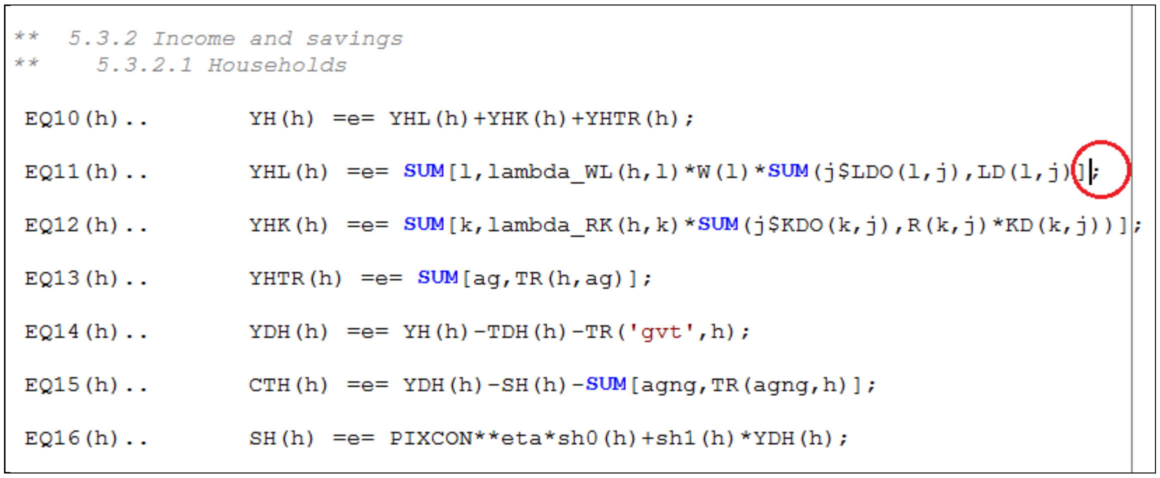
```
--- PEP-1-1_v1_1_Error1.gms(925) 3 Mb 2 Errors
*** Error 350 in F:\AGRODEP\PEP-1-1_v1_1_Error1.gms
    Unmatched parenthesis types. For example ( } or [ }
*** Error   8 in F:\AGRODEP\PEP-1-1_v1_1_Error1.gms
    ')' expected
--- PEP-1-1_v1_1_Error1.gms(1310) 3 Mb 3 Errors
*** Error 257 in F:\AGRODEP\PEP-1-1_v1_1_Error1.gms
    Solve statement not checked because of previous errors
--- PEP-1-1_v1_1_Error1.gms(1312) 3 Mb
--- .RESULTS PEP 1-1.GMS(394) 3 Mb
--- PEP-1-1_v1_1_Error1.gms(1313) 3 Mb 3 Errors
*** Status: Compilation error(s)
--- Job PEP-1-1_v1_1_Error1.gms Stop 12/19/11 13:49:06 elapsed 0:00:01.840
Exit code = 2
```

We start with the very first error, as the subsequent errors may be cumulative[8]. In other words, the first mistake may be repeated, or produce other errors, and then GAMS will compute it as another mistake, although it is in fact the same mistake.

By double-clicking on the first error in the process window, it brings us directly into the GAMS file. The cursor will actually show you where the error is, as shown in figure 4:

[8] Notice that in this particular case, the second message describing error 8 is also very instructive. However, we strongly recommend to start with the first error.

Figure 4 : GAMS file error 1



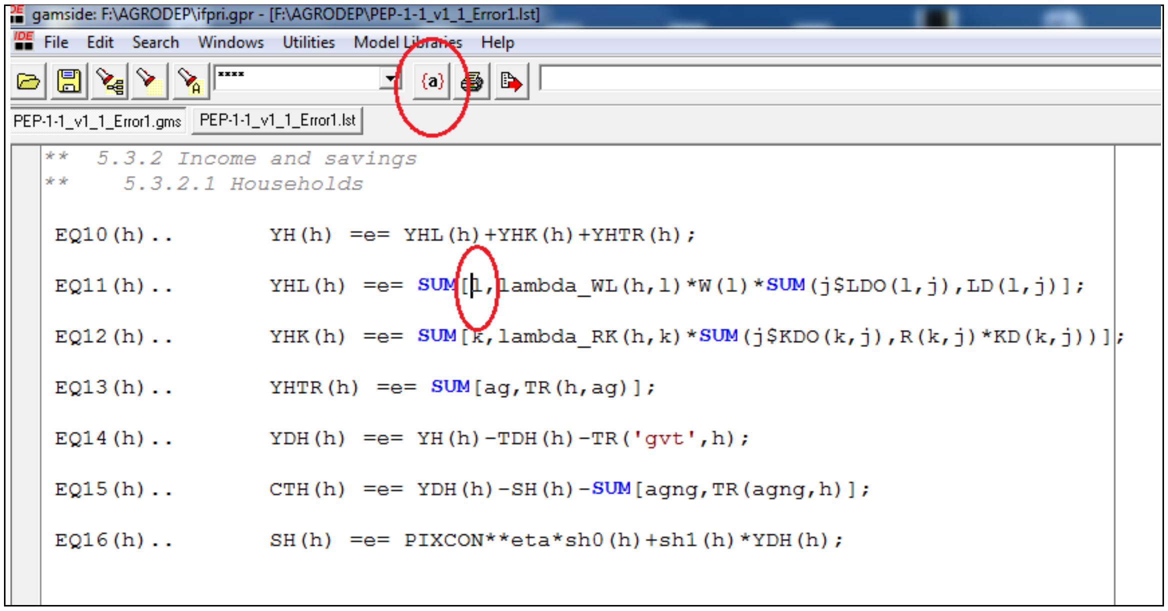
```
**   5.3.2 Income and savings
**     5.3.2.1 Households

 EQ10(h)..        YH(h)  =e= YHL(h)+YHK(h)+YHTR(h);

 EQ11(h)..        YHL(h)  =e= SUM[l,lambda_WL(h,l)*W(l)*SUM(j$LDO(l,j),LD(l,j)];

 EQ12(h)..        YHK(h)  =e= SUM[k,lambda_RK(h,k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ13(h)..        YHTR(h)  =e= SUM[ag,TR(h,ag)];

 EQ14(h)..        YDH(h)  =e= YH(h)-TDH(h)-TR('gvt',h);

 EQ15(h)..        CTH(h)  =e= YDH(h)-SH(h)-SUM[agng,TR(agng,h)];

 EQ16(h)..        SH(h)  =e= PIXCON**eta*sh0(h)+sh1(h)*YDH(h);
```

As previously mentioned, in the process window, GAMS provides some hints to the user for an explanation of the error. Looking back at figure 1, the error message shown in the process window for the very first error suggests that there are unmatched parenthesis types.

Looking carefully at equation 11 in figure 4 above, (where the cursor is), we notice that the type of parenthesis is different, and also that there is a parenthesis missing.

As some equations may be very long and complex, in order to help the user check where the open parenthesis matches up, GAMS has a facility: the user can put the cursor right after the open bracket and then press *F8* or the *{a}* touch, as shown in figure 5.

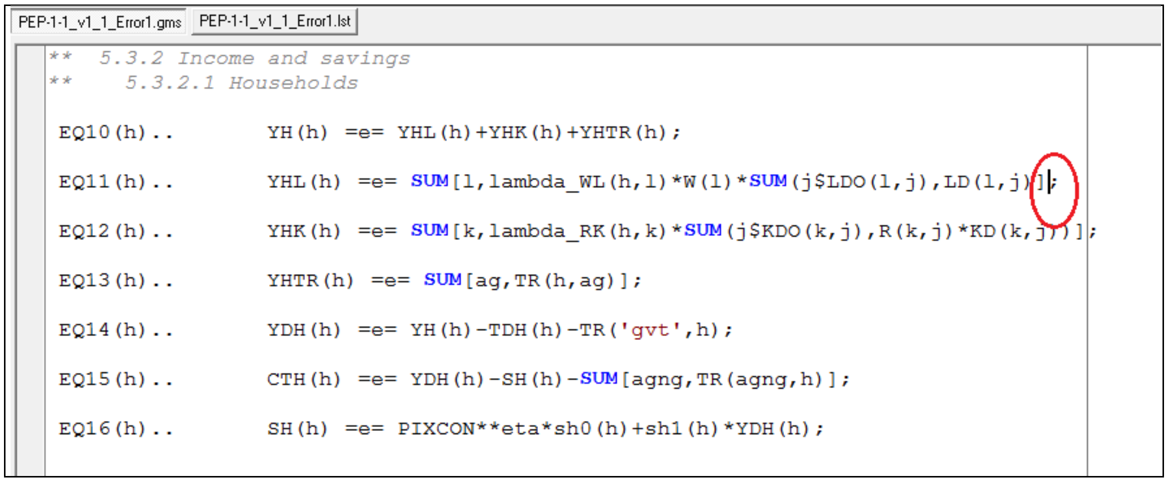
Figure 5 : Checking for parenthesis (F8 shortcut)


gamside: F:\AGRODEP\ifpri.gpr - [F:\AGRODEP\PEP-1-1_v1_1_Error1.lst]
File Edit Search Windows Utilities Model Libraries Help
PEP-1-1_v1_1_Error1.gms
PEP-1-1_v1_1_Error1.lst

```
**   5.3.2 Income and savings
**     5.3.2.1 Households

 EQ10(h)..        YH(h)  =e= YHL(h)+YHK(h)+YHTR(h);

 EQ11(h)..        YHL(h)  =e= SUM[l,lambda_WL(h,l)*W(l)*SUM(j$LDO(l,j),LD(l,j)];

 EQ12(h)..        YHK(h)  =e= SUM[k,lambda_RK(h,k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ13(h)..        YHTR(h)  =e= SUM[ag,TR(h,ag)];

 EQ14(h)..        YDH(h)  =e= YH(h)-TDH(h)-TR('gvt',h);

 EQ15(h)..        CTH(h)  =e= YDH(h)-SH(h)-SUM[agng,TR(agng,h)];

 EQ16(h)..        SH(h)  =e= PIXCON**eta*sh0(h)+sh1(h)*YDH(h);
```

This will bring the user exactly where the bracket closes, as shown below:

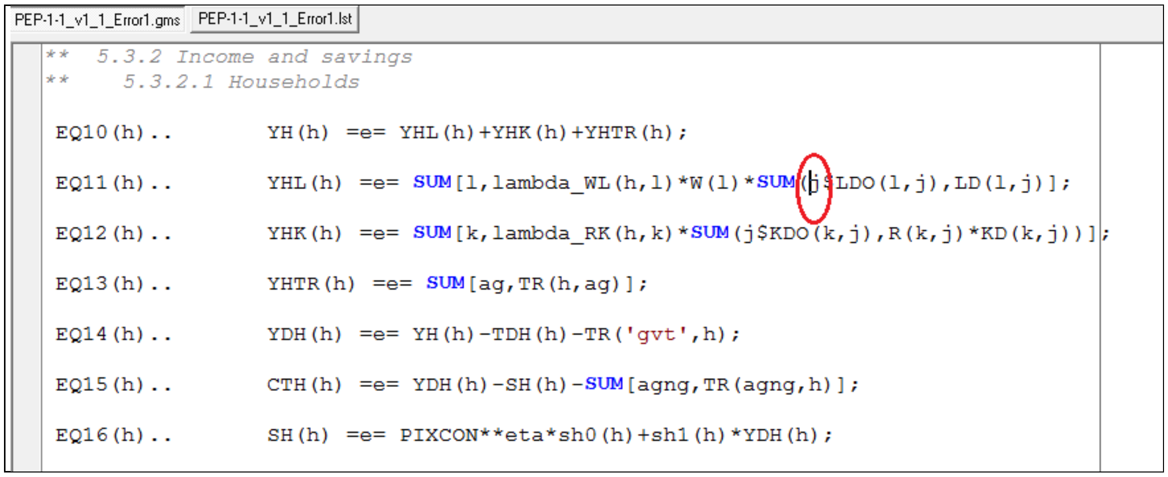
Figure 6 : Checking for parenthesis (2)



```
**   5.3.2 Income and savings
**     5.3.2.1 Households

 EQ10(h)..        YH(h)  =e= YHL(h)+YHK(h)+YHTR(h);

 EQ11(h)..        YHL(h)  =e= SUM[l,lambda_WL(h,l)*W(l)*SUM(j$LDO(l,j),LD(l,j)];

 EQ12(h)..        YHK(h)  =e= SUM[k,lambda_RK(h,k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ13(h)..        YHTR(h)  =e= SUM[ag,TR(h,ag)];

 EQ14(h)..        YDH(h)  =e= YH(h)-TDH(h)-TR('gvt',h);

 EQ15(h)..        CTH(h)  =e= YDH(h)-SH(h)-SUM[agng,TR(agng,h)];

 EQ16(h)..        SH(h)  =e= PIXCON**eta*sh0(h)+sh1(h)*YDH(h);
```

Repeating this exercise for each parenthesis in equation 11, we discover that there is one parenthesis that doesn't match up:

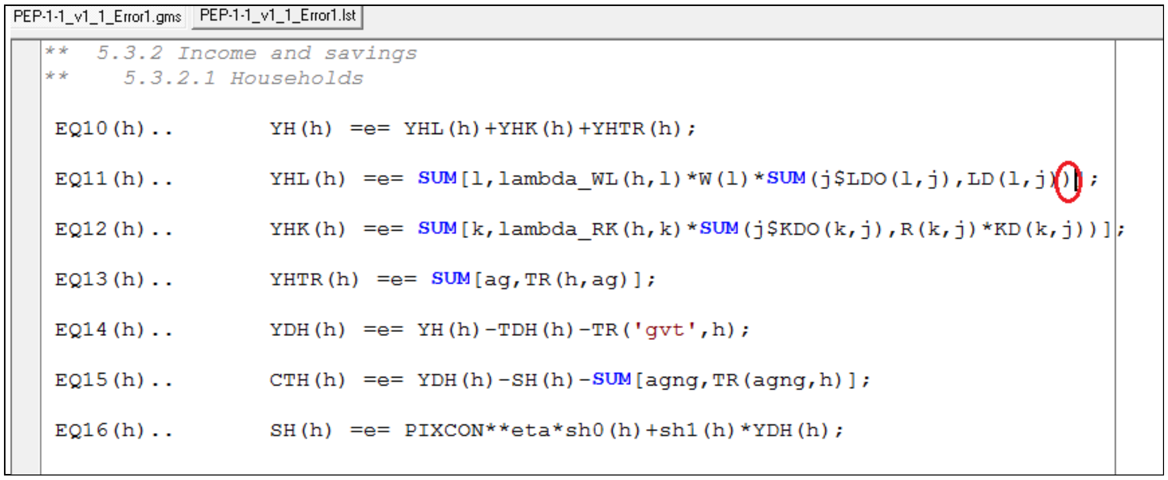
Figure 7 : Brackets in example 1:



```
PEP-1-1_v1_1_Error1.gms   PEP-1-1_v1_1_Error1.lst
**   5.3.2 Income and savings
**     5.3.2.1 Households

 EQ10(h)..        YH(h)  =e=  YHL(h)+YHK(h)+YHTR(h);

 EQ11(h)..        YHL(h)  =e=  SUM[l,lambda_WL(h,l)*W(l)*SUM(j$LDO(l,j),LD(l,j)];

 EQ12(h)..        YHK(h)  =e=  SUM[k,lambda_RK(h,k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ13(h)..        YHTR(h)  =e=  SUM[ag,TR(h,ag)];

 EQ14(h)..        YDH(h)  =e=  YH(h)-TDH(h)-TR('gvt',h);

 EQ15(h)..        CTH(h)  =e=  YDH(h)-SH(h)-SUM[agng,TR(agng,h)];

 EQ16(h)..        SH(h)  =e=  PIXCON**eta*sh0(h)+sh1(h)*YDH(h);
```

Thus, this bracket needs to be closed, and the correct place is just before the square bracket, as shown in figure 8.

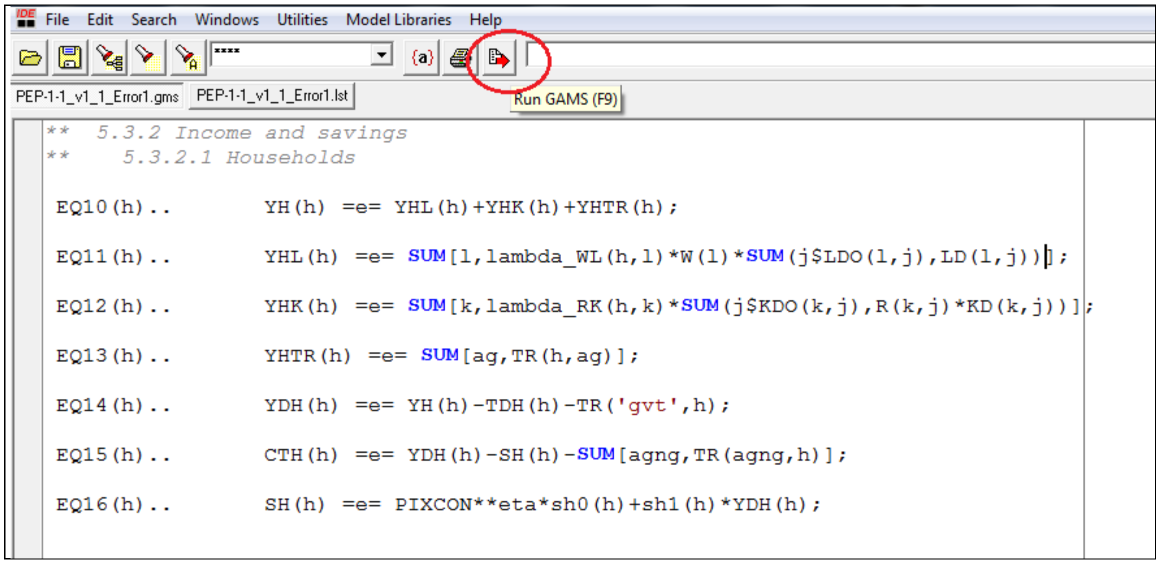
Figure 8 : Adding the missing bracket



```
PEP-1-1_v1_1_Error1.gms   PEP-1-1_v1_1_Error1.lst
**   5.3.2 Income and savings
**     5.3.2.1 Households

 EQ10(h)..        YH(h)  =e=  YHL(h)+YHK(h)+YHTR(h);

 EQ11(h)..        YHL(h)  =e=  SUM[l,lambda_WL(h,l)*W(l)*SUM(j$LDO(l,j),LD(l,j))];

 EQ12(h)..        YHK(h)  =e=  SUM[k,lambda_RK(h,k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ13(h)..        YHTR(h)  =e=  SUM[ag,TR(h,ag)];

 EQ14(h)..        YDH(h)  =e=  YH(h)-TDH(h)-TR('gvt',h);

 EQ15(h)..        CTH(h)  =e=  YDH(h)-SH(h)-SUM[agng,TR(agng,h)];

 EQ16(h)..        SH(h)  =e=  PIXCON**eta*sh0(h)+sh1(h)*YDH(h);
```

Once the bracket is added, the user may re-run the model by pressing *F9* or by clicking on the red arrow that appears on the GAMS window.

Figure 9 : Running a GAMS code:



```
**   5.3.2 Income and savings
**     5.3.2.1 Households

 EQ10(h)..         YH(h)  =e= YHL(h)+YHK(h)+YHTR(h);

 EQ11(h)..         YHL(h)  =e= SUM[l,lambda_WL(h,l)*W(l)*SUM(j$LDO(l,j),LD(l,j))];

 EQ12(h)..         YHK(h)  =e= SUM[k,lambda_RK(h,k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ13(h)..         YHTR(h)  =e= SUM[ag,TR(h,ag)];

 EQ14(h)..         YDH(h)  =e= YH(h)-TDH(h)-TR('gvt',h);

 EQ15(h)..         CTH(h)  =e= YDH(h)-SH(h)-SUM[agng,TR(agng,h)];

 EQ16(h)..         SH(h)  =e= PIXCON**eta*sh0(h)+sh1(h)*YDH(h);
```

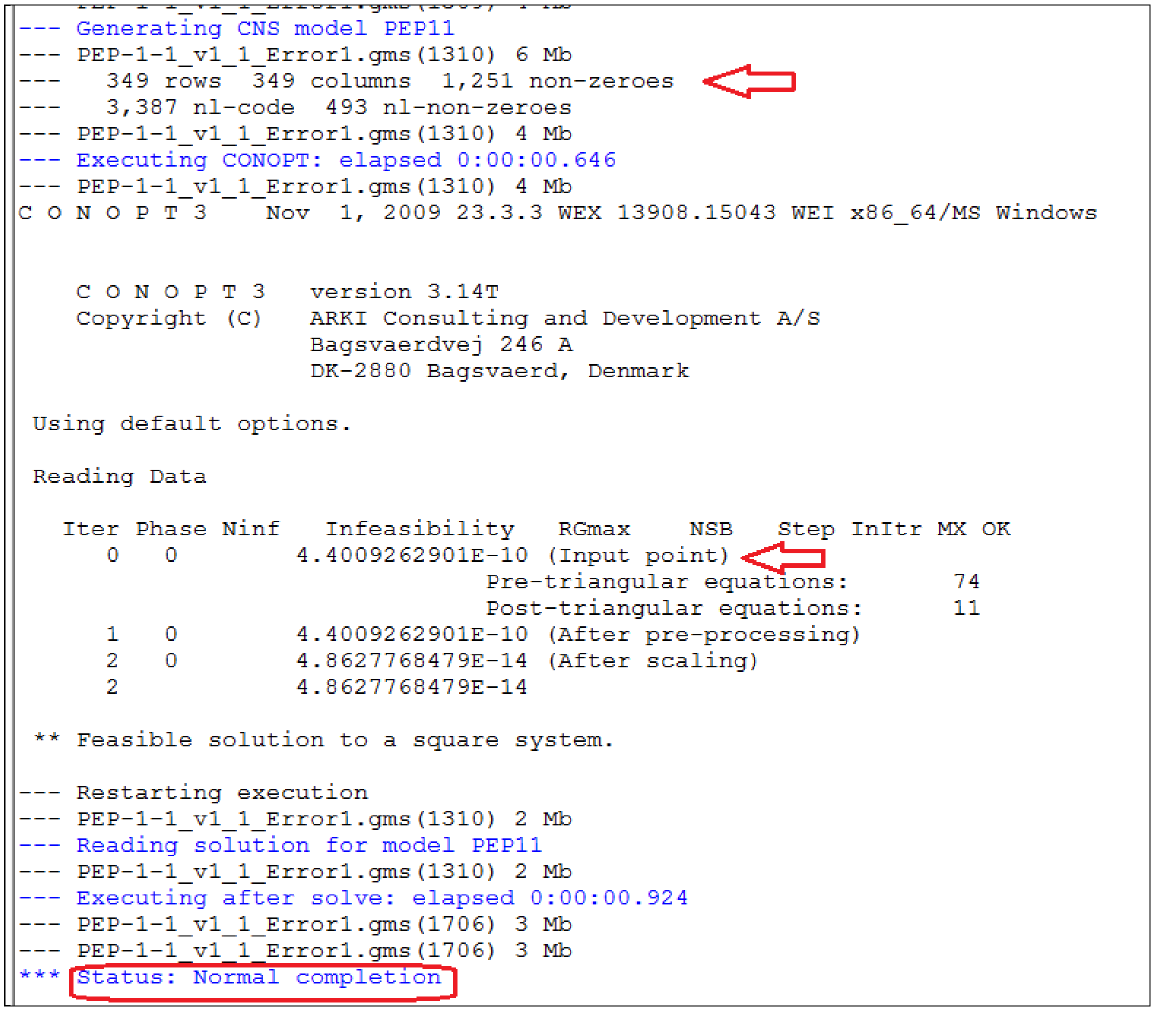
After pressing *F9*, a new process window appears where the status of the model is now “Normal completion”, as the errors have been corrected. Note that the unmatched bracket was causing 3 errors. It is always very important to start with the very first error in the list, as this error may be repeated or generate other errors.

After introducing a change, we want to be sure that there are less remaining errors than before the change was introduced. If by introducing a change, you generate more errors than the situation before, you might not have found the source of the problem.

In the new process window (see figure 10 below), the user wants to check that the status indicates “normal completion”, and that the model is square (first red arrow), meaning that it has the same number of equations as endogenous variables.

The second red arrow in the process window points to the infeasibility point. In the model we are running, there is no shock, meaning we want to reproduce the benchmark values (derived from the Social Accounting Matrix (SAM). Thus, GAMS is going to replace every variable and parameter by its value provided at the calibration process. This “input point” in the process window shows how far we are from the initial solution. GAMS provides an indication of the largest difference that exists in the whole system of equations between the initial data and the values computed by the model. If the model is calibrated correctly, the input point should be very small (as it is in figure 10).

Figure 10 : Process window of example 1

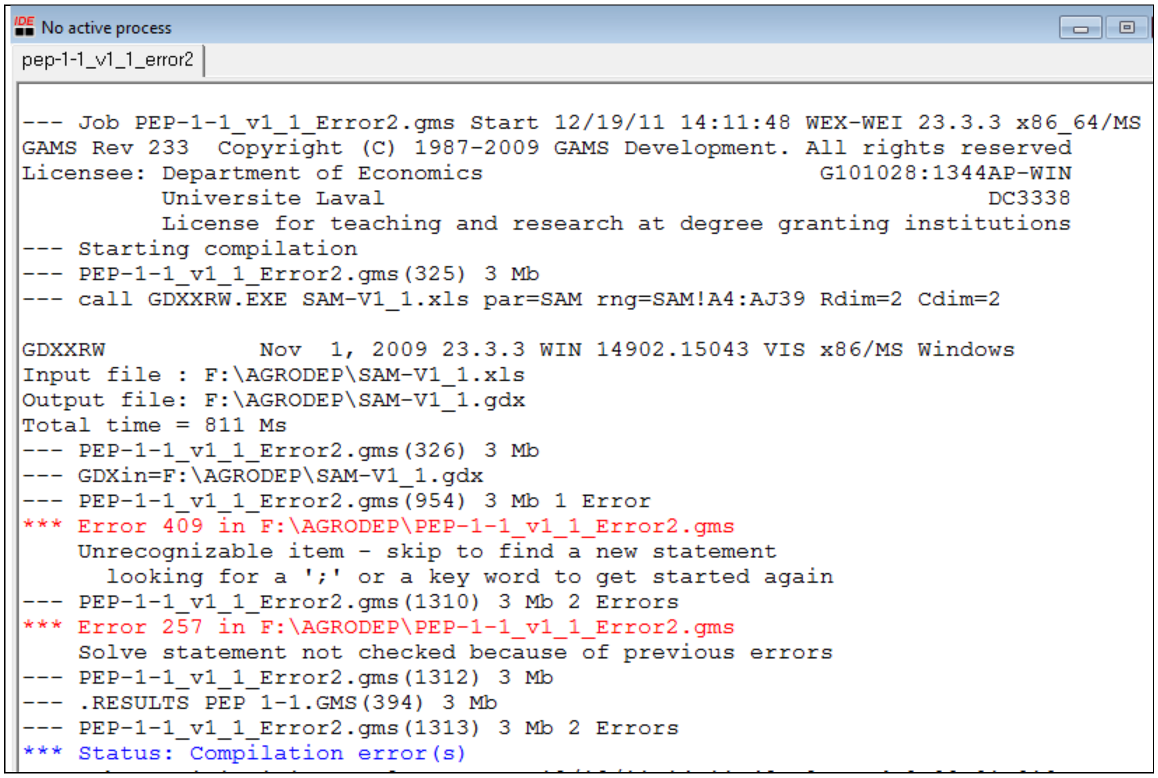


```
--- Generating CNS model PEP11
--- PEP-1-1_v1_1_Error1.gms(1310) 6 Mb
---   349 rows  349 columns  1,251 non-zeroes
---   3,387 nl-code  493 nl-non-zeroes
--- PEP-1-1_v1_1_Error1.gms(1310) 4 Mb
--- Executing CONOPT: elapsed 0:00:00.646
--- PEP-1-1_v1_1_Error1.gms(1310) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                 Pre-triangular equations:       74
                                 Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error1.gms(1310) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error1.gms(1310) 2 Mb
--- Executing after solve: elapsed 0:00:00.924
--- PEP-1-1_v1_1_Error1.gms(1706) 3 Mb
--- PEP-1-1_v1_1_Error1.gms(1706) 3 Mb
*** Status: Normal completion
```

Example 2:

The second process window is as follows:

Figure 11 : Process window of example 2



No active process

pep-1-1_v1_1_error2

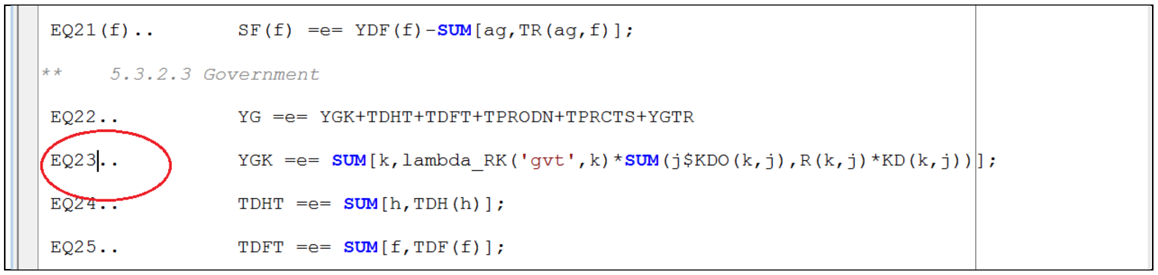
```
--- Job PEP-1-1_v1_1_Error2.gms Start 12/19/11 14:11:48 WEX-WEI 23.3.3 x86_64/MS
GAMS Rev 233  Copyright (C) 1987-2009 GAMS Development. All rights reserved
Licensee: Department of Economics                        G101028:1344AP-WIN
          Universite Laval                                           DC3338
          License for teaching and research at degree granting institutions
--- Starting compilation
--- PEP-1-1_v1_1_Error2.gms(325) 3 Mb
--- call GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2

GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 811 Ms
--- PEP-1-1_v1_1_Error2.gms(326) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error2.gms(954) 3 Mb 1 Error
*** Error 409 in F:\AGRODEP\PEP-1-1_v1_1_Error2.gms
    Unrecognizable item - skip to find a new statement
      looking for a ';' or a key word to get started again
--- PEP-1-1_v1_1_Error2.gms(1310) 3 Mb 2 Errors
*** Error 257 in F:\AGRODEP\PEP-1-1_v1_1_Error2.gms
    Solve statement not checked because of previous errors
--- PEP-1-1_v1_1_Error2.gms(1312) 3 Mb
--- .RESULTS PEP 1-1.GMS(394) 3 Mb
--- PEP-1-1_v1_1_Error2.gms(1313) 3 Mb 2 Errors
*** Status: Compilation error(s)
```

As indicated previously, we do not need to pay attention to *Error 257* since GAMS simply indicates that the model could not run properly due to previous errors.

Below *Error 409*, there is a short description that provides information on the source of the problem: GAMS is looking for a semi column or a key word.

Double-clicking on Error 409 in the process window brings the cursor directly in the GAMS code, just after *EQ23*, as shown in the figure below.

Figure 12 : Location of the error in example 2



```
EQ21(f)..        SF(f) =e= YDF(f)-SUM[ag,TR(ag,f)];

**    5.3.2.3 Government

EQ22..           YG =e= YGK+TDHT+TDFT+TPRODN+TPRCTS+YGTR

EQ23..           YGK =e= SUM[k,lambda_RK('gvt',k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

EQ24..           TDHT =e= SUM[h,TDH(h)];

EQ25..           TDFT =e= SUM[f,TDF(f)];
```

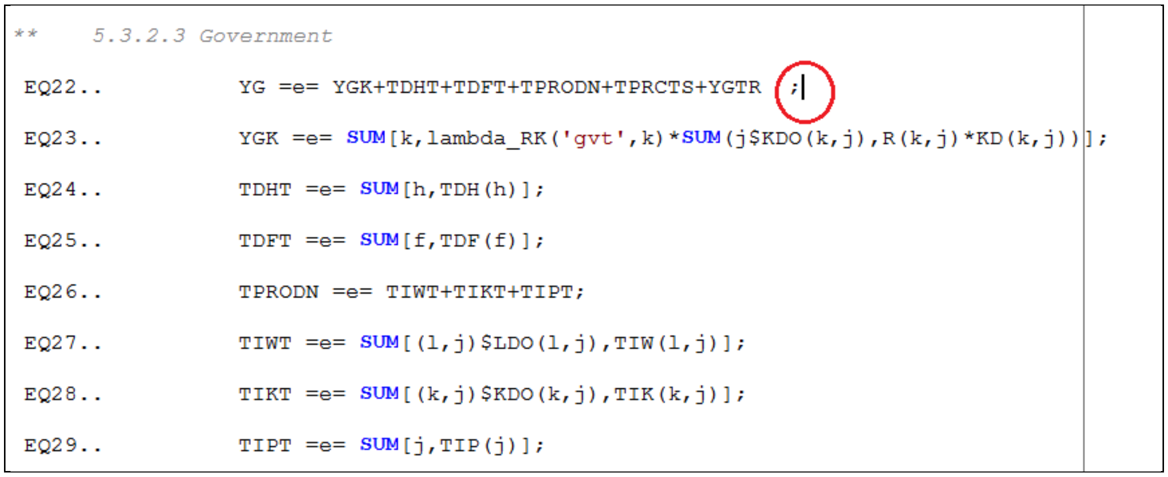
By looking at the location of the cursor in figure 12, it appears that there is no need to introduce neither a semicolon, nor a key word right after *EQ23*.

In this case, it can be particularly useful to go and check the previous line, meaning the end of EQ22 just above EQ23.

When looking at *EQ22*, we can see that there is a semicolon missing at the end of the equation (just after *YGTR*). Therefore, we need to add a semicolon right after *YGTR*.

We then re-run the model by pressing *F9* or the red arrow in the GAMS window.

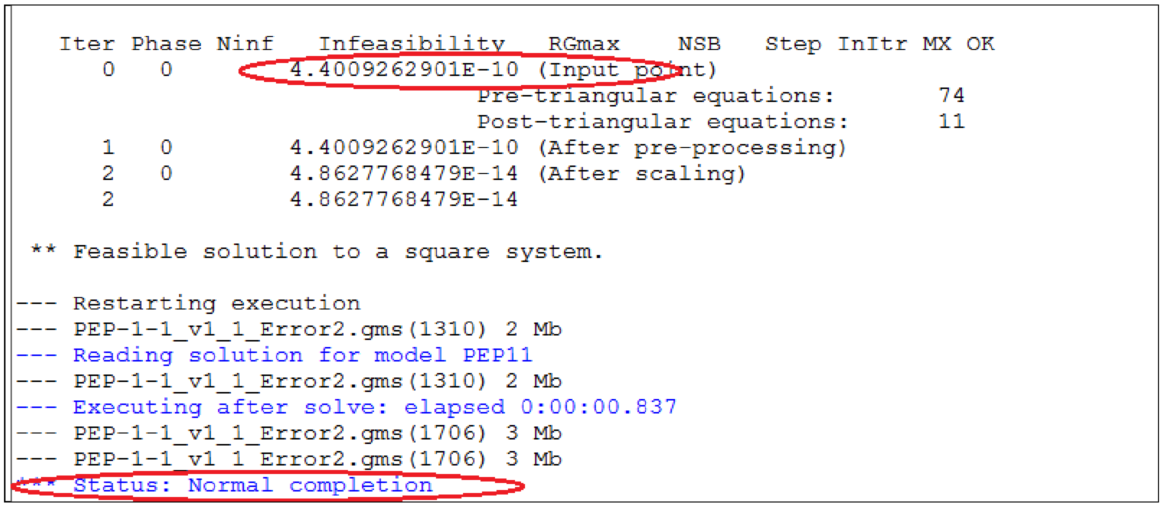
Figure 13 : Correction of example 2



```
**    5.3.2.3 Government

EQ22..           YG =e= YGK+TDHT+TDFT+TPRODN+TPRCTS+YGTR  ;

EQ23..           YGK =e= SUM[k,lambda_RK('gvt',k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

EQ24..           TDHT =e= SUM[h,TDH(h)];

EQ25..           TDFT =e= SUM[f,TDF(f)];

EQ26..           TPRODN =e= TIWT+TIKT+TIPT;

EQ27..           TIWT =e= SUM[(l,j)$LDO(l,j),TIW(l,j)];

EQ28..           TIKT =e= SUM[(k,j)$KDO(k,j),TIK(k,j)];

EQ29..           TIPT =e= SUM[j,TIP(j)];
```

Finally, in the process window, we check that the status is “normal completion”, and that the input-point is very small.

Figure 14 : Process window of example 2

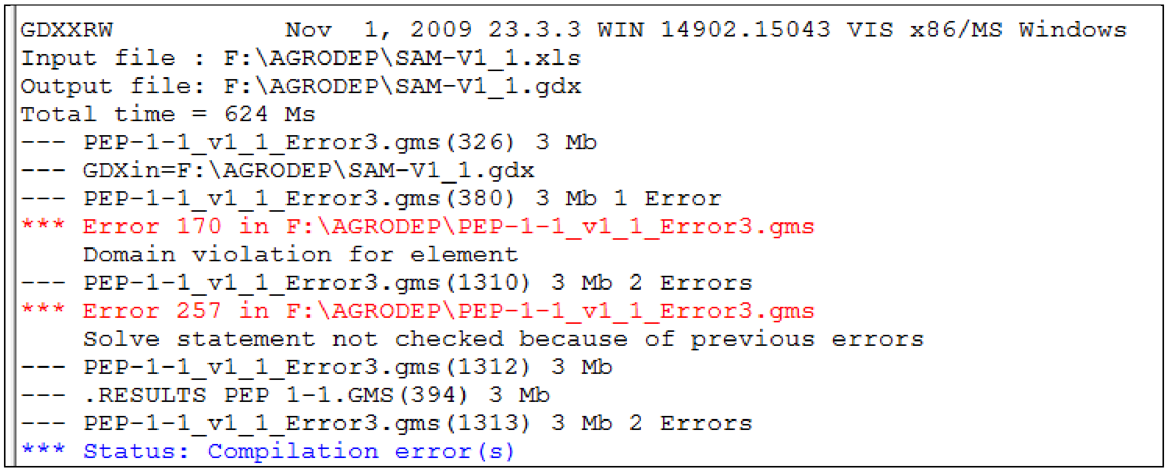


```
   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                 Pre-triangular equations:       74
                                 Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error2.gms(1310) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error2.gms(1310) 2 Mb
--- Executing after solve: elapsed 0:00:00.837
--- PEP-1-1_v1_1_Error2.gms(1706) 3 Mb
--- PEP-1-1_v1_1_Error2.gms(1706) 3 Mb
*** Status: Normal completion
```

Example 3:

Example 3 is presented as follows:

Figure 15 : Process window for example 3

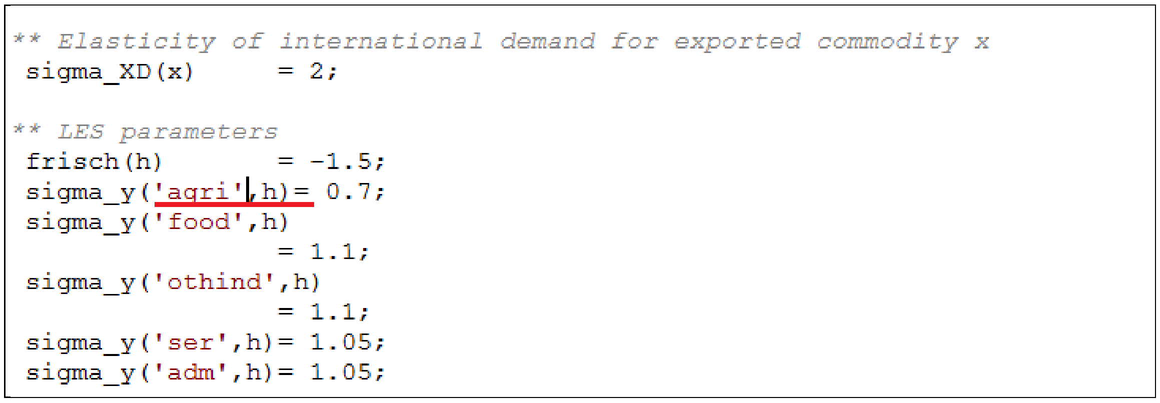


```
GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 624 Ms
--- PEP-1-1_v1_1_Error3.gms(326) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error3.gms(380) 3 Mb 1 Error
*** Error 170 in F:\AGRODEP\PEP-1-1_v1_1_Error3.gms
    Domain violation for element
--- PEP-1-1_v1_1_Error3.gms(1310) 3 Mb 2 Errors
*** Error 257 in F:\AGRODEP\PEP-1-1_v1_1_Error3.gms
    Solve statement not checked because of previous errors
--- PEP-1-1_v1_1_Error3.gms(1312) 3 Mb
--- .RESULTS PEP 1-1.GMS(394) 3 Mb
--- PEP-1-1_v1_1_Error3.gms(1313) 3 Mb 2 Errors
*** Status: Compilation error(s)
```

Example 3 consists of one error, *Error 170*. *Error 257* only indicates that GAMS could not solve the model because of previous errors. Below *Error 170*, there is a small definition that indicates that an element has been used that does not belong to a defined domain.

To locate the error in the GAMS file, we double-click on *Error 170* in the process window, and this will bring us directly in the GAMS code where the error is located.

Figure 16 : Error 3 in the GAMS file



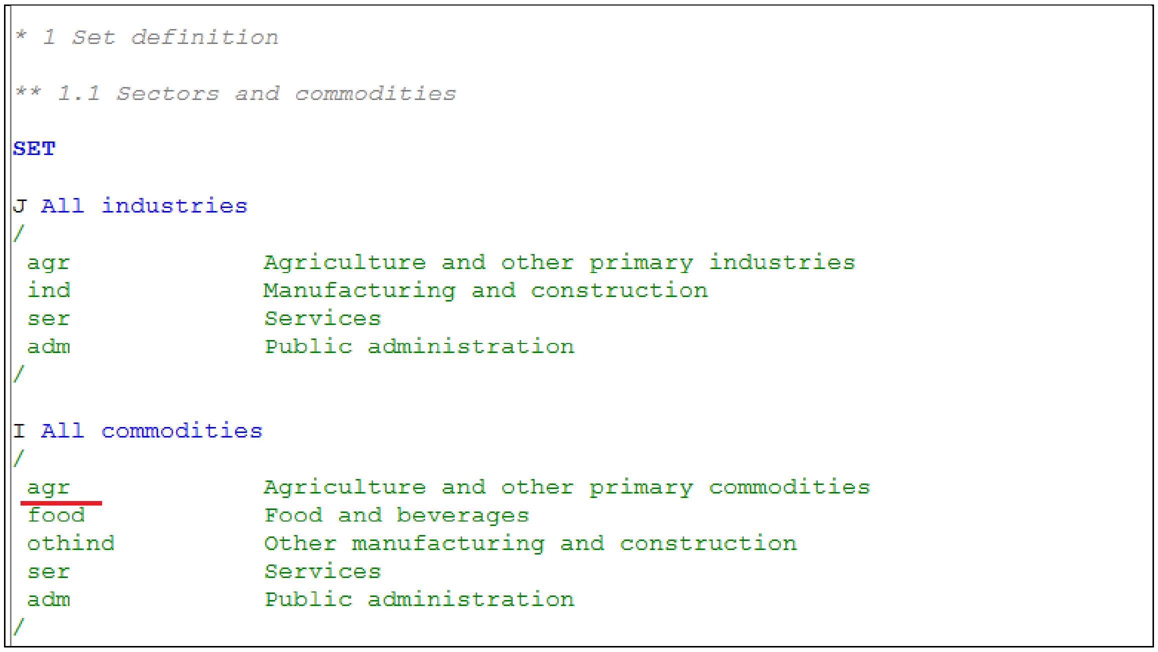
```
** Elasticity of international demand for exported commodity x
 sigma_XD(x)      = 2;

** LES parameters
 frisch(h)         = -1.5;
 sigma_y('agri',h)= 0.7;
 sigma_y('food',h)
                   = 1.1;
 sigma_y('othind',h)
                   = 1.1;
 sigma_y('ser',h)= 1.05;
 sigma_y('adm',h)= 1.05;
```

In the GAMS code, the cursor is located at the assignment of a value for *sigma_y* for the agricultural sector. From the definition of *Error 170* in the process window, we know that an element (in this case, *agri*) does not belong to a defined domain.

Therefore, we have to check if the element *agri* belongs to one of the domains defined. To be more specific, as the parameter *sigma_y* represents the income elasticity of a given commodity for households, it refers to the set *i* (commodity set defined in PEP 1-1).

Therefore, we have to check if the element *agri* belongs to the set *i*, where the sets are defined in the GAMS code. In the case of PEP 1-1, all the sets are defined at the very beginning of the GAMS file.

Figure 17 : Set declaration:



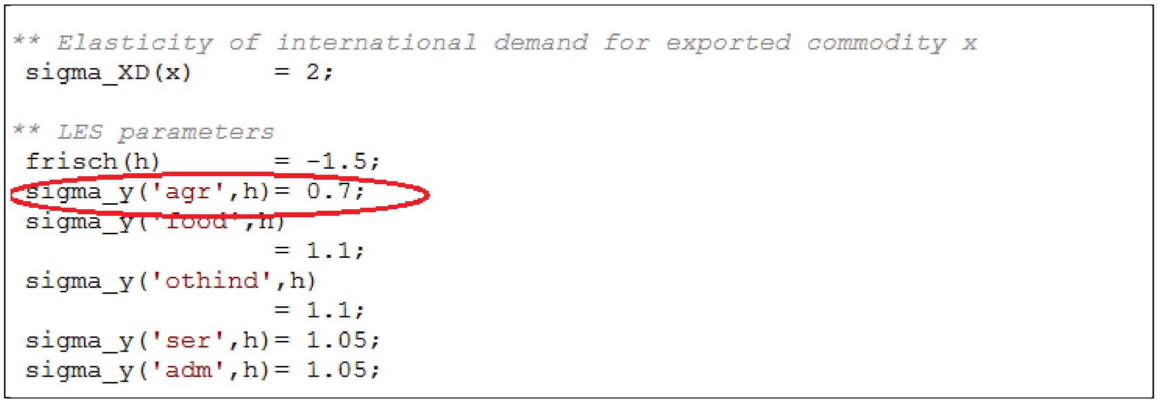
```
* 1 Set definition

** 1.1 Sectors and commodities

SET

J All industries
/
 agr                Agriculture and other primary industries
 ind                Manufacturing and construction
 ser                Services
 adm                Public administration
/

I All commodities
/
 agr                Agriculture and other primary commodities
 food               Food and beverages
 othind             Other manufacturing and construction
 ser                Services
 adm                Public administration
/
```

From Figure 17, we can see that the set *i* refers to the commodities set and includes five elements: *agr, food, othind, ser, adm*.

Thus, *agr* belongs to *i* but *agri* doesn't. In this case, GAMS doesn't recognize *agri* as an element of *i,* and therefore indicates that this element does not belong to the domain.

To correct the error, we simply need to write *agr* instead of *agri* as shown below.

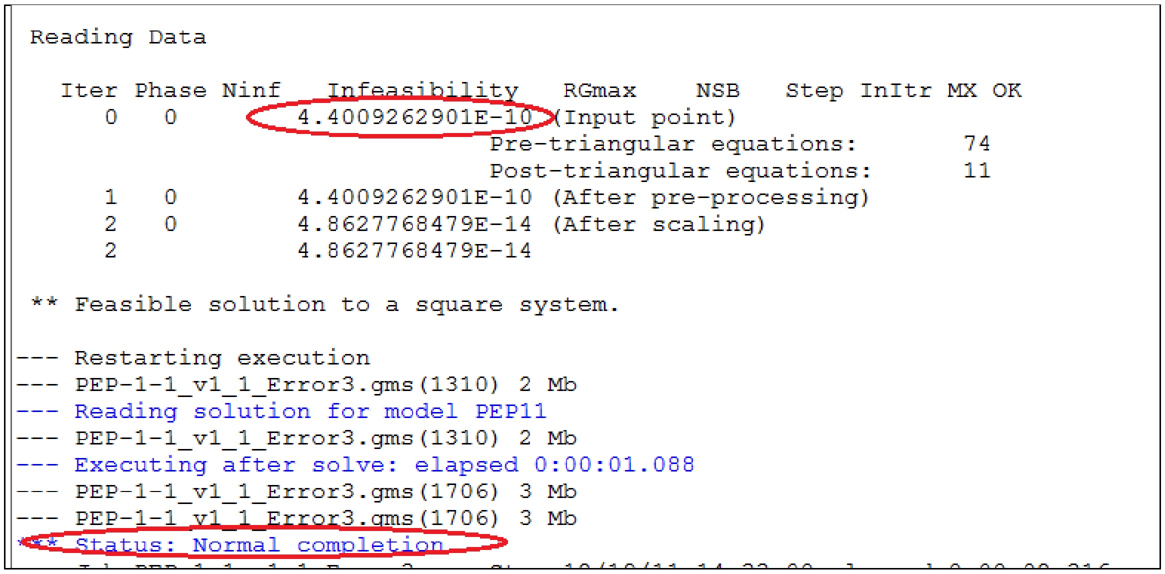
Figure 18 : Correction of example 3:



```
** Elasticity of international demand for exported commodity x
 sigma_XD(x)      = 2;

** LES parameters
 frisch(h)        = -1.5;
 sigma_y('agr',h)= 0.7;
 sigma_y('food',h)
                  = 1.1;
 sigma_y('othind',h)
                  = 1.1;
 sigma_y('ser',h)= 1.05;
 sigma_y('adm',h)= 1.05;
```

We then make sure that the model runs correctly by pressing *F9*. We obtain the following process window where we can check that the status is “normal completion”, and that the input point is small.

Figure 19 : Process window of example 3

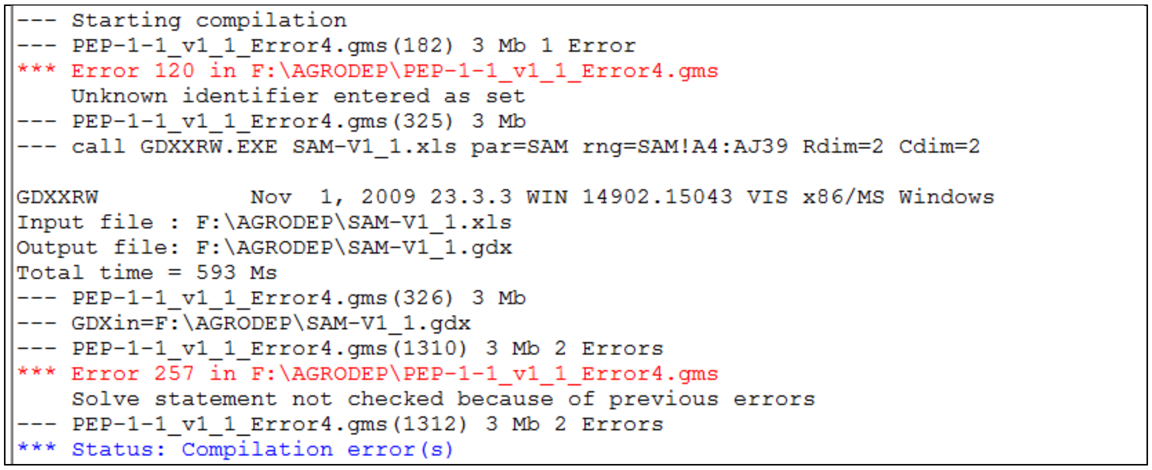


```
 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0           4.4009262901E-10 (Input point)
                                    Pre-triangular equations:       74
                                    Post-triangular equations:      11
      1   0           4.4009262901E-10 (After pre-processing)
      2   0           4.8627768479E-14 (After scaling)
      2               4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error3.gms(1310) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error3.gms(1310) 2 Mb
--- Executing after solve: elapsed 0:00:01.088
--- PEP-1-1_v1_1_Error3.gms(1706) 3 Mb
--- PEP-1-1_v1_1_Error3.gms(1706) 3 Mb
*** Status: Normal completion
```

Example 4:

Example 4 is as follows:

Figure 20 : Process window of example 4

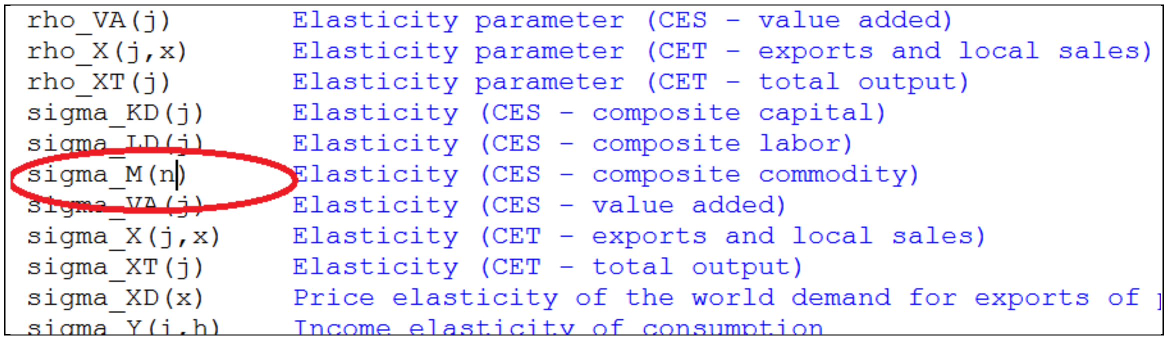


```
--- Starting compilation
--- PEP-1-1_v1_1_Error4.gms(182) 3 Mb 1 Error
*** Error 120 in F:\AGRODEP\PEP-1-1_v1_1_Error4.gms
    Unknown identifier entered as set
--- PEP-1-1_v1_1_Error4.gms(325) 3 Mb
--- call GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2

GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 593 Ms
--- PEP-1-1_v1_1_Error4.gms(326) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error4.gms(1310) 3 Mb 2 Errors
*** Error 257 in F:\AGRODEP\PEP-1-1_v1_1_Error4.gms
    Solve statement not checked because of previous errors
--- PEP-1-1_v1_1_Error4.gms(1312) 3 Mb 2 Errors
*** Status: Compilation error(s)
```

The example 4 consists of one error, *Error 120*, as *Error 257* only indicates that GAMS couldn't solve the model.

To locate the error in the GAMS file, we double-click on *Error 120* in the process window. The cursor is as follows:

Figure 21 : Error in example 4

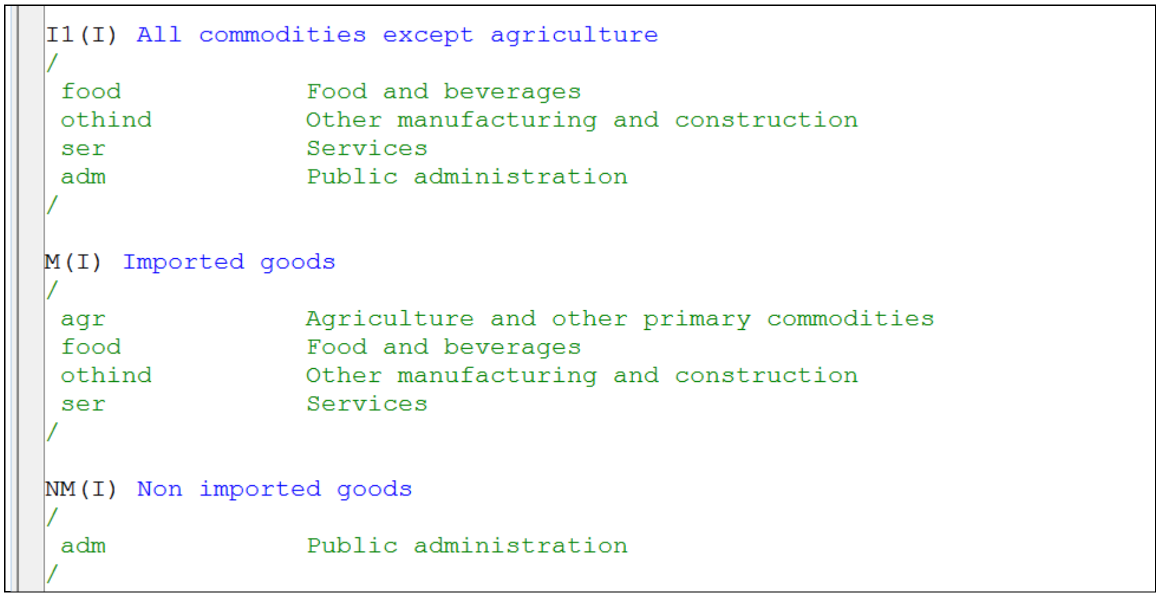


```
rho_VA(j)       Elasticity parameter (CES - value added)
rho_X(j,x)      Elasticity parameter (CET - exports and local sales)
rho_XT(j)       Elasticity parameter (CET - total output)
sigma_KD(j)     Elasticity (CES - composite capital)
sigma_LD(j)     Elasticity (CES - composite labor)
sigma_M(n)      Elasticity (CES - composite commodity)
sigma_VA(j)     Elasticity (CES - value added)
sigma_X(j,x)    Elasticity (CET - exports and local sales)
sigma_XT(j)     Elasticity (CET - total output)
sigma_XD(x)     Price elasticity of the world demand for exports of
sigma_Y(i,h)    Income elasticity of consumption
```

In the process window, the definition under *Error 120* indicates that an unknown identifier has been entered as set. The cursor in the error window indicates that the set entered for *sigma_M*(n) is unknown.

For this type of error, we have to go and check which sets were defined in the GAMS file~~.~~ In the case of PEP 1-1, the declaration of the sets is done at the very beginning of the GAMS file.

Figure 22 : Set declaration in PEP 1-1



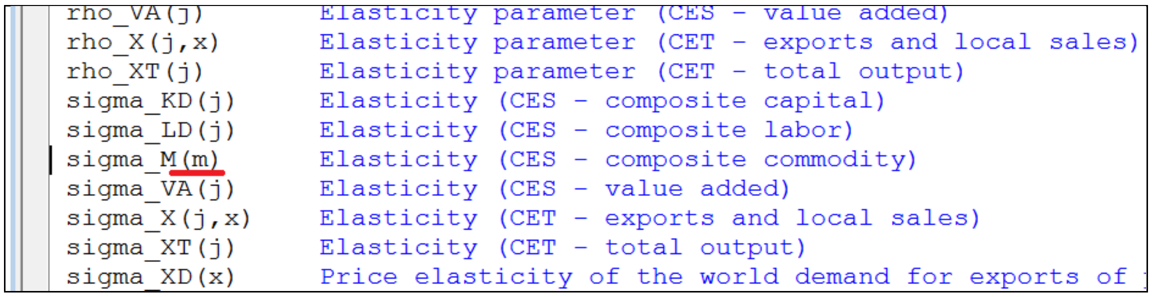
```
I1(I) All commodities except agriculture
/
 food             Food and beverages
 othind           Other manufacturing and construction
 ser              Services
 adm              Public administration
/

M(I) Imported goods
/
 agr              Agriculture and other primary commodities
 food             Food and beverages
 othind           Other manufacturing and construction
 ser              Services
/

NM(I) Non imported goods
/
 adm              Public administration
/
```

Figure 22 indicates some of the sets that are defined in PEP1-1. The cursor indicates an error for the parameter *sigma_M*, which refers to the elasticity for the composite commodity. Thus, *sigma_M* should refer to the imported goods.

By checking the different sets and subsets, it appears that the appropriate set for imported goods is *m* and not *n*.

To correct the error, we~~-~~simply replace the *n* by an *m* in the definition of *sigma_M*, where the mistake was identified.

Figure 23 : Correction of error 4

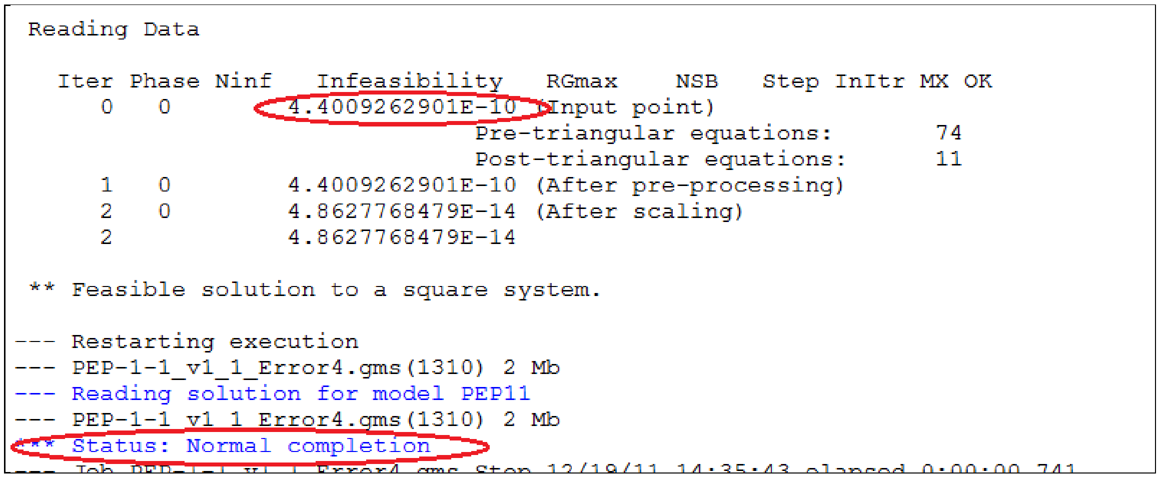


```
rho_VA(j)          Elasticity parameter (CES - value added)
rho_X(j,x)         Elasticity parameter (CET - exports and local sales)
rho_XT(j)          Elasticity parameter (CET - total output)
sigma_KD(j)        Elasticity (CES - composite capital)
sigma_LD(j)        Elasticity (CES - composite labor)
sigma_M(m)         Elasticity (CES - composite commodity)
sigma_VA(j)        Elasticity (CES - value added)
sigma_X(j,x)       Elasticity (CET - exports and local sales)
sigma_XT(j)        Elasticity (CET - total output)
sigma_XD(x)        Price elasticity of the world demand for exports of
```

Once the corrected value is introduced, we can run the model again by pressing F9, or the red arrow.

This allows the user to check that the model has run successfully.

Figure 24 : Process window after correction of example 4

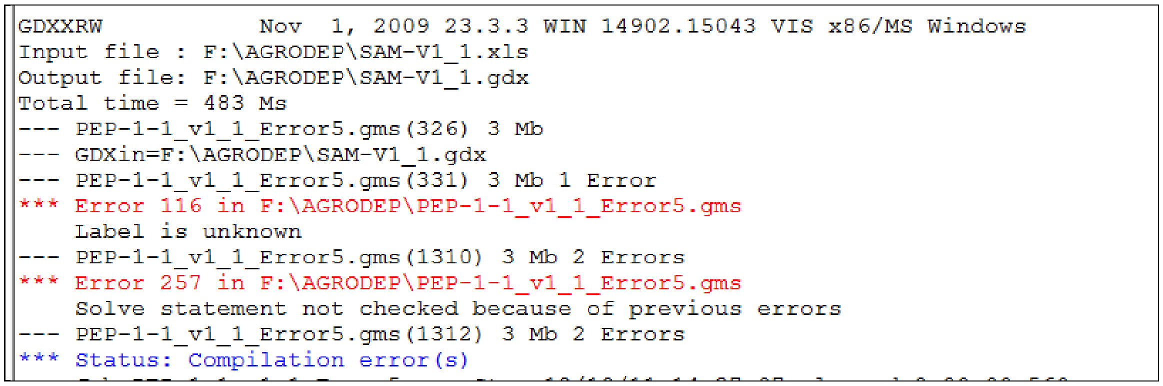


```
 Reading Data

   Iter  Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0    0        4.4009262901E-10 (Input point)
                               Pre-triangular equations:       74
                               Post-triangular equations:      11
      1    0        4.4009262901E-10 (After pre-processing)
      2    0        4.8627768479E-14 (After scaling)
      2             4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error4.gms(1310) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error4.gms(1310) 2 Mb
*** Status: Normal completion
--- Job PEP-1-1_v1_1_Error4.gms Stop 12/19/11 14:35:43 elapsed 0:00:00.741
```

Example 5:

Example 5 is presented in the figure below:

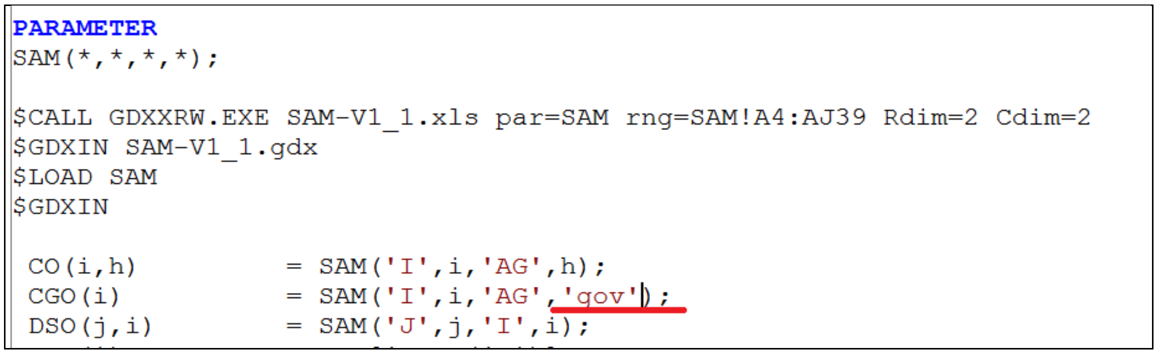
Figure 25 : Example 5



```
GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 483 Ms
--- PEP-1-1_v1_1_Error5.gms(326) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error5.gms(331) 3 Mb 1 Error
*** Error 116 in F:\AGRODEP\PEP-1-1_v1_1_Error5.gms
    Label is unknown
--- PEP-1-1_v1_1_Error5.gms(1310) 3 Mb 2 Errors
*** Error 257 in F:\AGRODEP\PEP-1-1_v1_1_Error5.gms
    Solve statement not checked because of previous errors
--- PEP-1-1_v1_1_Error5.gms(1312) 3 Mb 2 Errors
*** Status: Compilation error(s)
```

The example 5 consists in one error, *Error 116*. By double-clicking on the first red line, it brings us to the following figure in the GAMS code:

Figure 26 : Location of the error in the GAMS file (example 5)



```
PARAMETER
SAM(*,*,*,*);

$CALL GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2
$GDXIN SAM-V1_1.gdx
$LOAD SAM
$GDXIN

 CO(i,h)           = SAM('I',i,'AG',h);
 CGO(i)            = SAM('I',i,'AG','gov');
 DSO(j,i)          = SAM('J',j,'I',i);
```

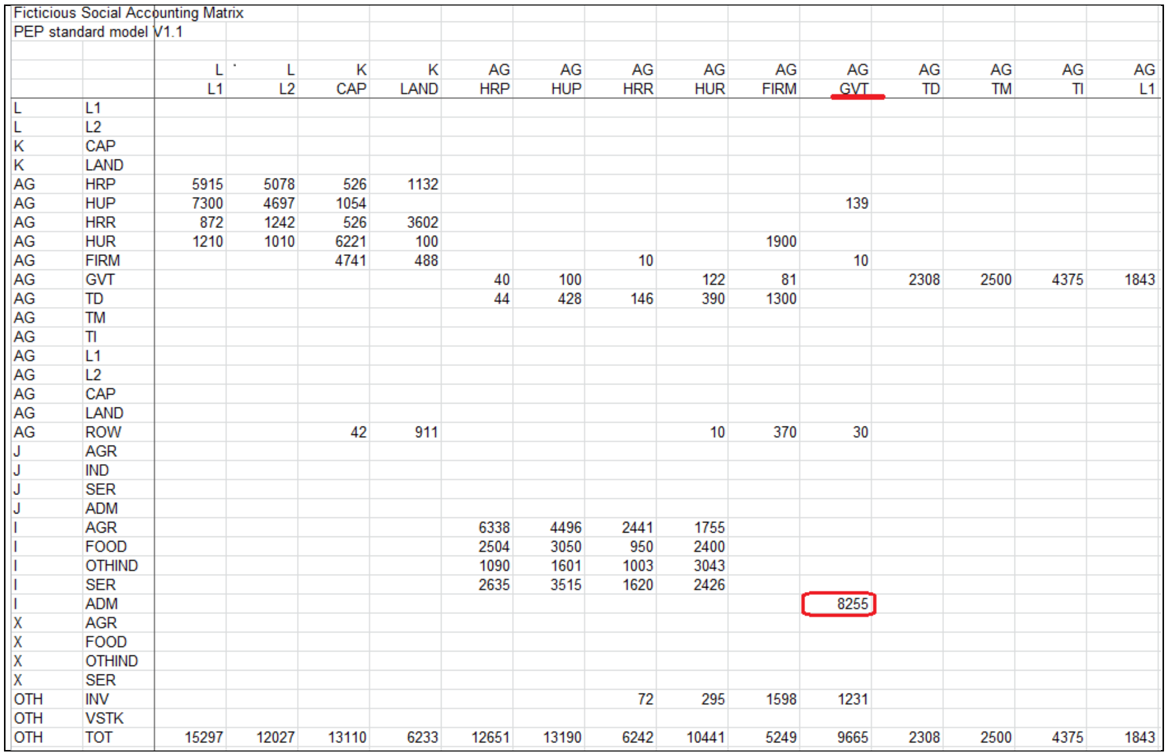
As shown in the figure above, the cursor brings us right after the element “gov”.

The definition of *Error 116* in the process window indicates that a label is unknown; namely the label *gov* is unknown.

At this specific place in the GAMS code (figure 26), we assign a value to the variable *CGO(i)*. We actually ask GAMS to find in the parameter SAM (created above) the corresponding value for *CGO(i).*

By using the gdx facility[9] to call the data, we know that the dimension of the rows and columns is 2, meaning that the first two labels in each row and column won't be data but labels. Therefore, the name we give when assigning a value to a variable in the GAMS code must be the same as the one in the excel file. The figure below represents the first part of the SAM used for PEP-1-1, in the SAM_V1_1.xls file.

Figure 27 : Social Accounting Matrix used in PEP-1-1



Ficticious Social Accounting Matrix
PEP standard model V1.1

		L	L	K	K	AG	AG	AG	AG	AG	AG	AG	AG	AG	AG
		L1	L2	CAP	LAND	HRP	HUP	HRR	HUR	FIRM	GVT	TD	TM	TI	L1
L	L1														
L	L2														
K	CAP														
K	LAND														
AG	HRP	5915	5078	526	1132										
AG	HUP	7300	4697	1054							139				
AG	HRR	872	1242	526	3602										
AG	HUR	1210	1010	6221	100					1900					
AG	FIRM			4741	488			10			10				
AG	GVT					40	100		122	81		2308	2500	4375	1843
AG	TD					44	428	146	390	1300					
AG	TM														
AG	TI														
AG	L1														
AG	L2														
AG	CAP														
AG	LAND														
AG	ROW			42	911				10	370	30				
J	AGR														
J	IND														
J	SER														
J	ADM														
I	AGR					6338	4496	2441	1755						
I	FOOD					2504	3050	950	2400						
I	OTHIND					1090	1601	1003	3043						
I	SER					2635	3515	1620	2426						
I	ADM										8255				
X	AGR														
X	FOOD														
X	OTHIND														
X	SER														
OTH	INV							72	295	1598	1231				
OTH	VSTK														
OTH	TOT	15297	12027	13110	6233	12651	13190	6242	10441	5249	9665	2308	2500	4375	1843

The figure clearly shows that the first two labels of rows and columns represent labels, and not data.

CGO(i) is public consumption of commodity *i*. In the SAM, (encircled in red in the figure above), we can see that the corresponding labels would be ("I",I,"AG","GVT").

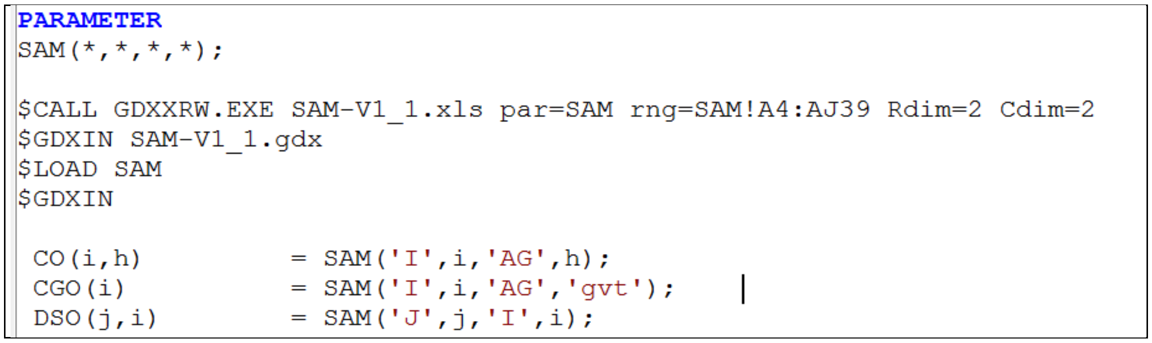
However, this is not what is written in the GAMS code. Indeed, in figure 26, we can see that we assigned *CGO(i)* as ("I",I,"AG","GOV"), and *GOV* has not been defined anywhere in the SAM.

Thus, it is very simple to correct the error in the GAMS code simply by changing the label *GOV* for *GVT*[10].

[9] Please refer to Mc Carl et al (2008) for an exhaustive explanation of the gdx facility, and to Robichaud et al (2011) section 4.3 to have a complete explanation of the gdx facility used in PEP 1-1

[10] Note that one could have changed the label in the SAM. As in the rest of the GAMS code, we always refer to *gvt*, we decided to change *gov* for *gvt* in this particular case.

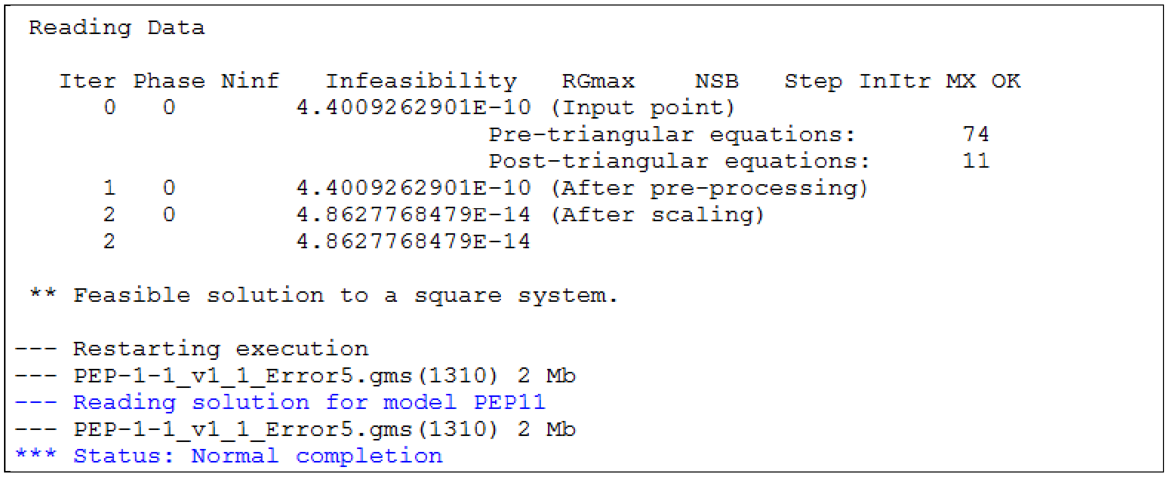
Figure 28 : Correction of example 5



```
PARAMETER
SAM(*,*,*,*);

$CALL GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2
$GDXIN SAM-V1_1.gdx
$LOAD SAM
$GDXIN

 CO(i,h)           = SAM('I',i,'AG',h);
 CGO(i)            = SAM('I',i,'AG','gvt');
 DSO(j,i)          = SAM('J',j,'I',i);
```

After substituting *GOV* for *GVT* in the figure above, we can run the model by pressing *F9* and check that the completion is normal.

Figure 29 : Checking example 5 solution

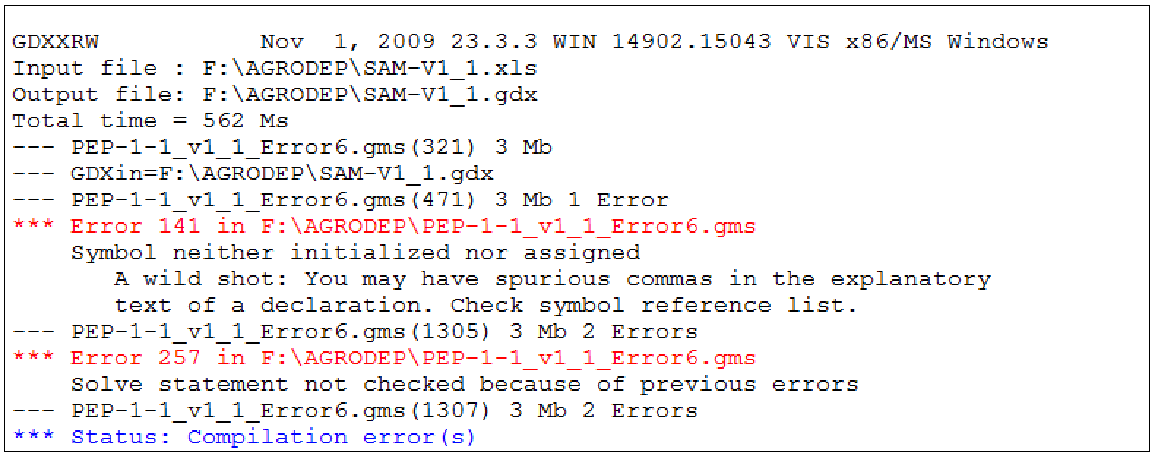


```
 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                Pre-triangular equations:       74
                                Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error5.gms(1310) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error5.gms(1310) 2 Mb
*** Status: Normal completion
```

Example 6:

Figure 30 presents Error 6:

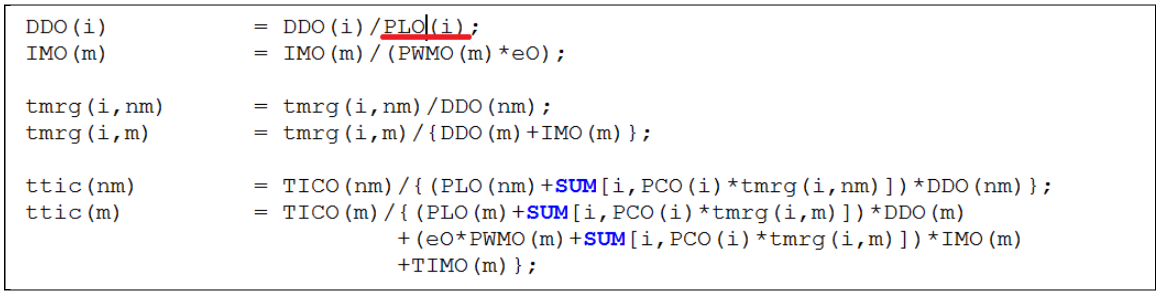
Figure 30 : Process window of example 6



```
GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 562 Ms
--- PEP-1-1_v1_1_Error6.gms(321) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error6.gms(471) 3 Mb 1 Error
*** Error 141 in F:\AGRODEP\PEP-1-1_v1_1_Error6.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.
--- PEP-1-1_v1_1_Error6.gms(1305) 3 Mb 2 Errors
*** Error 257 in F:\AGRODEP\PEP-1-1_v1_1_Error6.gms
    Solve statement not checked because of previous errors
--- PEP-1-1_v1_1_Error6.gms(1307) 3 Mb 2 Errors
*** Status: Compilation error(s)
```

Example 6 contains *Error 141.* By double-clicking on the red line *Error 141* in the process window, it brings us to the following part in the GAMS code:

Figure 31 : Location of error 6 in the GAMS code



```
DDO(i)              = DDO(i)/PLO(i);
IMO(m)              = IMO(m)/(PWMO(m)*eO);

tmrg(i,nm)          = tmrg(i,nm)/DDO(nm);
tmrg(i,m)           = tmrg(i,m)/{DDO(m)+IMO(m)};

ttic(nm)            = TICO(nm)/{(PLO(nm)+SUM[i,PCO(i)*tmrg(i,nm)])*DDO(nm)};
ttic(m)             = TICO(m)/{(PLO(m)+SUM[i,PCO(i)*tmrg(i,m)])*DDO(m)
                          +(eO*PWMO(m)+SUM[i,PCO(i)*tmrg(i,m)])*IMO(m)
                          +TIMO(m)};
```

Underlined in red in the figure above, the cursor brings us to the variable *PLO.*

In the process window (figure 30), right under the red line mentioning *Error 141*, GAMS gives a short definition of the error. It indicates that the symbol (in this case *PLO*) has been neither initialized nor assigned.

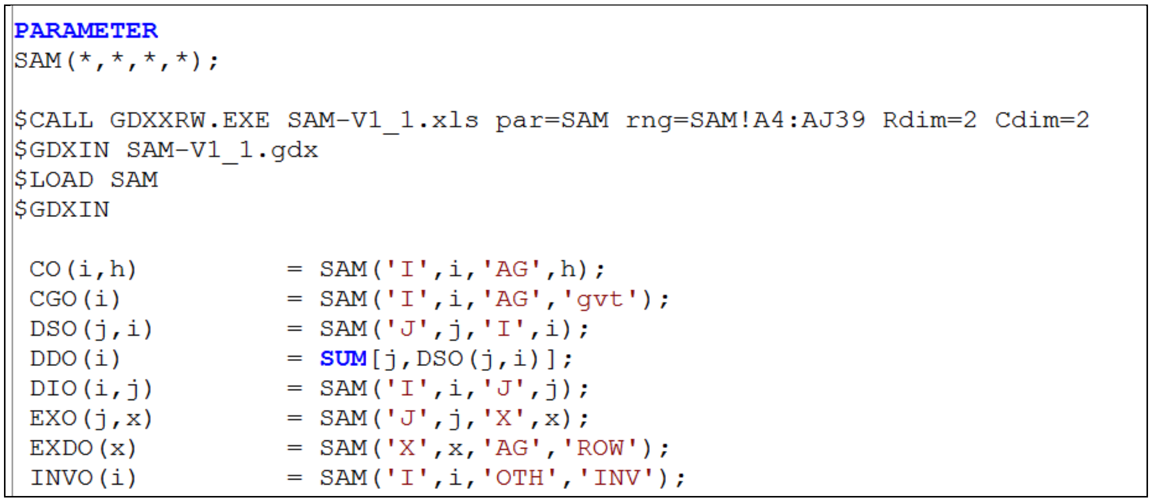
Therefore, we have to check that a value has been assigned to *PLO(i)* before this specific line in the GAMS code. Specifically, during the calibration step, the order of the variables/parameters matters. If a variable/parameter is used, it must have been defined previously in the GAMS code.

Effectively, we have to check in the GAMS code if a value has been assigned to *PLO(i)*.

We know that in the calibration, a value must be assigned to each parameter and variable. There are three different approaches to assigning these values:

- First approach: Reading the value of a variable from the SAM: in the previous example, we were assigning a value to *CGO(i)* directly from the SAM.

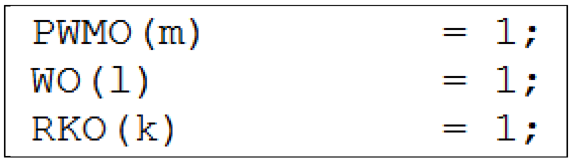
Figure 32 : Assignment of a value to a variable using the SAM



```
PARAMETER
SAM(*,*,*,*);

$CALL GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2
$GDXIN SAM-V1_1.gdx
$LOAD SAM
$GDXIN

 CO(i,h)            = SAM('I',i,'AG',h);
 CGO(i)             = SAM('I',i,'AG','gvt');
 DSO(j,i)           = SAM('J',j,'I',i);
 DDO(i)             = SUM[j,DSO(j,i)];
 DIO(i,j)           = SAM('I',i,'J',j);
 EXO(j,x)           = SAM('J',j,'X',x);
 EXDO(x)            = SAM('X',x,'AG','ROW');
 INVO(i)            = SAM('I',i,'OTH','INV');
```

-Second approach: Directly assigning a value to a variable in the GAMS code. Some variables cannot be read directly from the SAM (ex: all prices, elasticity…). In that case, we have to directly assign a value to those variables.

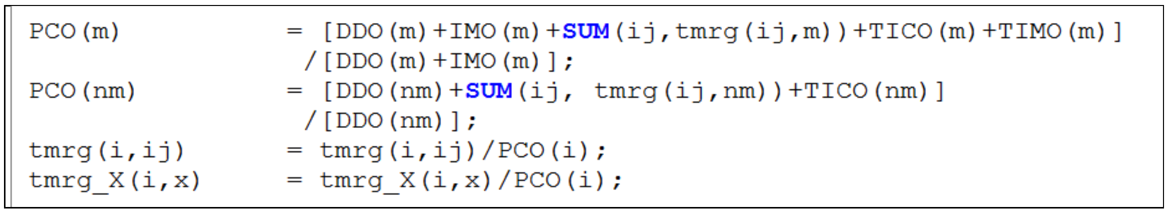
Figure 33 : Assignment of a value to a variable by giving it a value directly



```
 PWMO(m)             = 1;
 WO(l)               = 1;
 RKO(k)              = 1;
```

-Third approach: Computing the variable. This means that the value of the variable depends on the value of the other variables.

Figure 34 : Assignment of a value to a variable by calibrating it



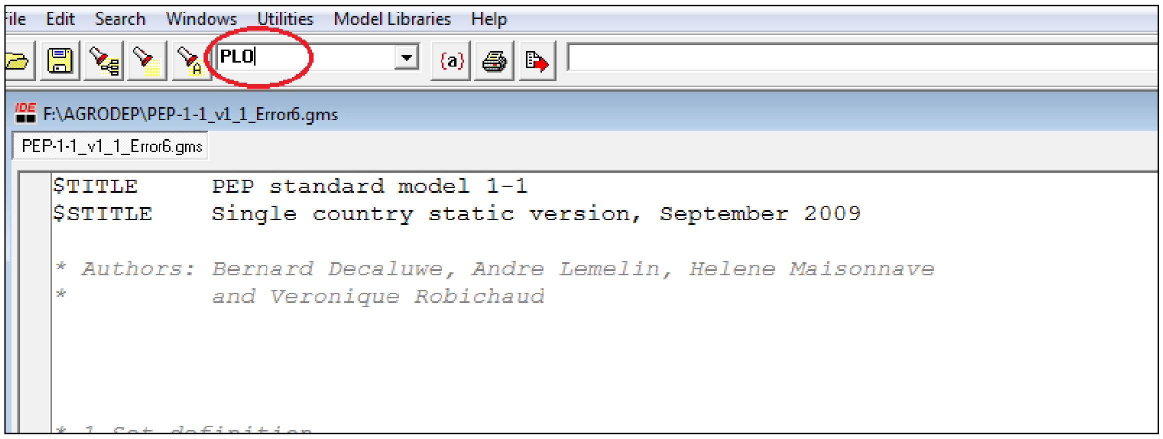
```
 PCO(m)             =  [DDO(m)+IMO(m)+SUM(ij,tmrg(ij,m))+TICO(m)+TIMO(m)]
                     /[DDO(m)+IMO(m)];
 PCO(nm)            =  [DDO(nm)+SUM(ij, tmrg(ij,nm))+TICO(nm)]
                     /[DDO(nm)];
 tmrg(i,ij)         = tmrg(i,ij)/PCO(i);
 tmrg_X(i,x)        = tmrg_X(i,x)/PCO(i);
```

We point out again that in the calibration process, the order of computation of different variables/parameters is very important. For example, in figure 34 above, to compute *tmrg(i,ij),* the variable *PCO(i)* has to be defined before.

In the case of example 6, we have to check if a value has been assigned to *PLO(i)* before it appears on the right hand side of an equation.

To find *PLO* from the beginning of the GAMS code, we write *PLO* in the search window and then hit the flashlight next to the search window.

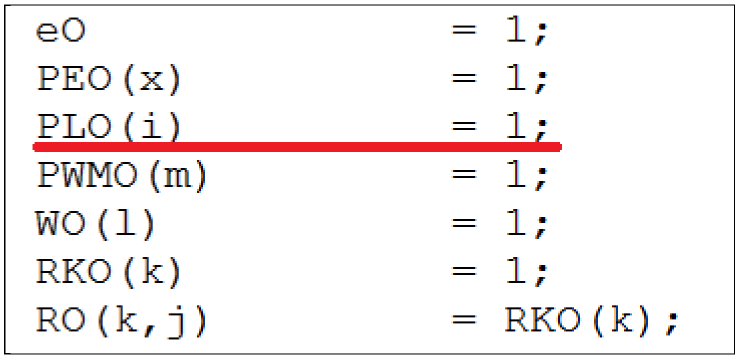
Figure 35 : Looking for PLO (i) in the GAMS code


File Edit Search Windows Utilities Model Libraries Help
PLO
F:\AGRODEP\PEP-1-1_v1_1_Error6.gms
PEP-1-1_v1_1_Error6.gms

```
$TITLE     PEP standard model 1-1
$STITLE    Single country static version, September 2009

* Authors: Bernard Decaluwe, Andre Lemelin, Helene Maisonnave
*          and Veronique Robichaud
```

We discover that *PLO (i)* has not been assigned a value at all. We then have to assign a value that is consistent with the price system of PEP 1-1.

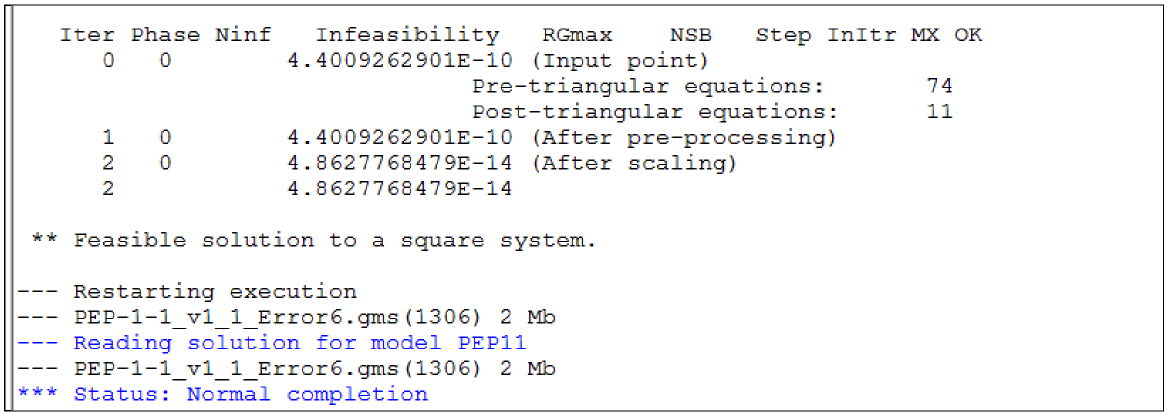
Figure 36 : Correction of example 6



```
eO               = 1;
PEO(x)           = 1;
PLO(i)           = 1;
PWMO(m)          = 1;
WO(l)            = 1;
RKO(k)           = 1;
RO(k,j)          = RKO(k);
```

Afterwards, we re-run the model by pressing F9 and check that the model runs successfully, without any errors.

Figure 37 : Verify example 6 solution



```
    Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
       0   0        4.4009262901E-10 (Input point)
                                 Pre-triangular equations:       74
                                 Post-triangular equations:      11
       1   0        4.4009262901E-10 (After pre-processing)
       2   0        4.8627768479E-14 (After scaling)
       2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error6.gms(1306) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error6.gms(1306) 2 Mb
*** Status: Normal completion
```

2. Execution errors:

In the following examples, the compilation errors have all been resolved. However, GAMS indicates that it cannot solve the model because it has encountered an execution problem.

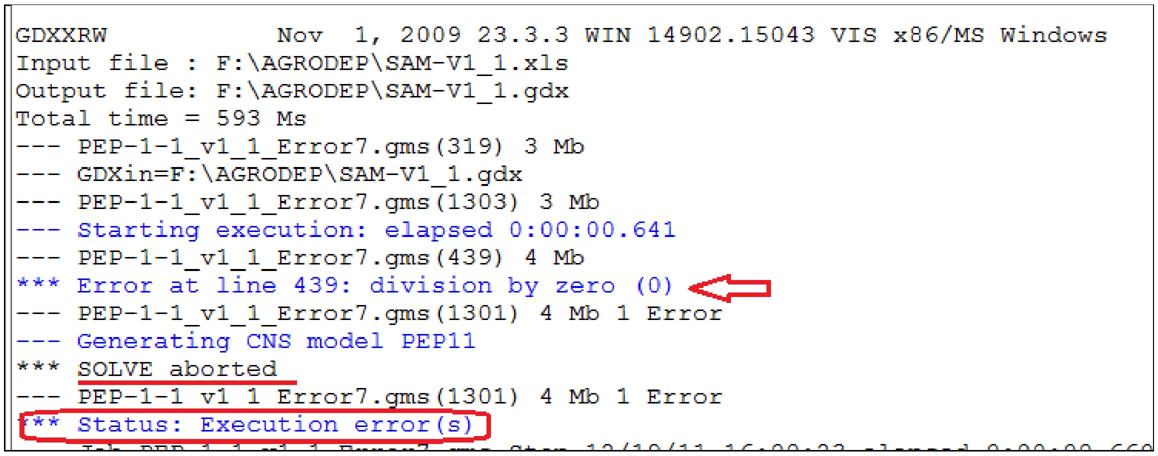
This can happen when the model is not square (i.e. the number of endogenous variables is different from the number of independent equations), or when GAMS has to divide by zero (i.e. a variable was assigned 0 as a value, and at some point in the GAMS code, it has to divide by this value, …)

For these types of errors, GAMS indicates that it could not solve the system of equations due to computational problems.

Example 7:

Example 7 is presented as follows:

Figure 38 : Process window of example 7

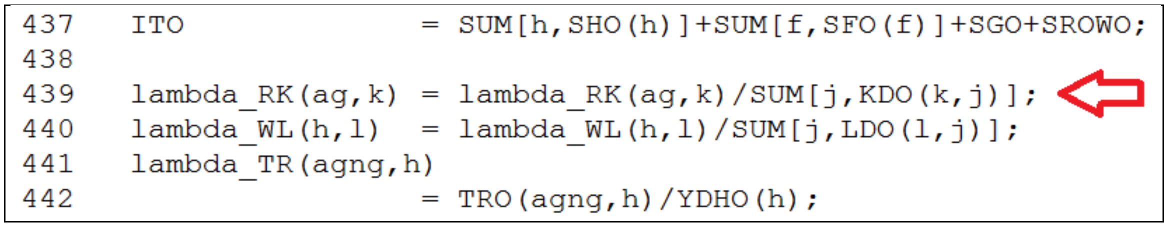


```
GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 593 Ms
--- PEP-1-1_v1_1_Error7.gms(319) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error7.gms(1303) 3 Mb
--- Starting execution: elapsed 0:00:00.641
--- PEP-1-1_v1_1_Error7.gms(439) 4 Mb
*** Error at line 439: division by zero (0)
--- PEP-1-1_v1_1_Error7.gms(1301) 4 Mb 1 Error
--- Generating CNS model PEP11
*** SOLVE aborted
--- PEP-1-1_v1_1_Error7.gms(1301) 4 Mb 1 Error
*** Status: Execution error(s)
```

In this example, the solution aborted due to a division by zero. In the process window, GAMS indicates the specific line where the division by zero would happen.

As in the previous examples, we double-click on the blue line of the error (indicated by the red arrow). This brings us to the output file (also called listing file). This file reproduces the entire GAMS code, numbering each line of the GAMS file. Subsequently, we can easily go to line 439, as indicated in the process window.

Figure 39 : Listing file of example 7



```
437    ITO                 = SUM[h,SHO(h)]+SUM[f,SFO(f)]+SGO+SROWO;
438
439    lambda_RK(ag,k)     = lambda_RK(ag,k)/SUM[j,KDO(k,j)];
440    lambda_WL(h,l)      = lambda_WL(h,l)/SUM[j,LDO(l,j)];
441    lambda_TR(agng,h)
442                        = TRO(agng,h)/YDHO(h);
```

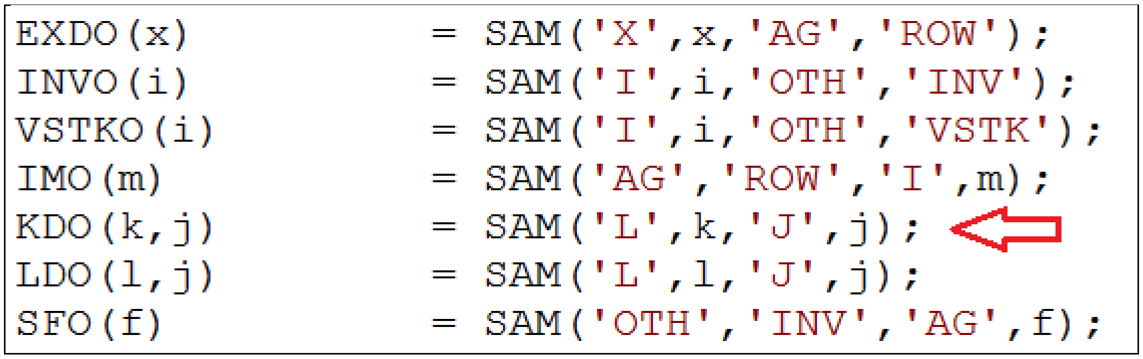
As we can see from figure 39, on line 439, GAMS indicates that the term in the denominator is equal to zero. In other words, *KDO(k,j)* may be equal to zero, at least for one of its values.

To determine the source of the problem, the first assumption here is that we have initialized *KDO(k,j)* to zero. If we had forgotten to assign a value to *KDO(k,j),* it would have created a compilation error, as in the case of example 6, where we had forgotten to assign a value to *PLO(i)*.

The next step is to check where in the GAMS code we assigned a value to *KDO(k,j).*

KDO(k,j) is a variable we can read from the SAM.

Figure 40 : Assigning the value of KDO(k,j) in example 7



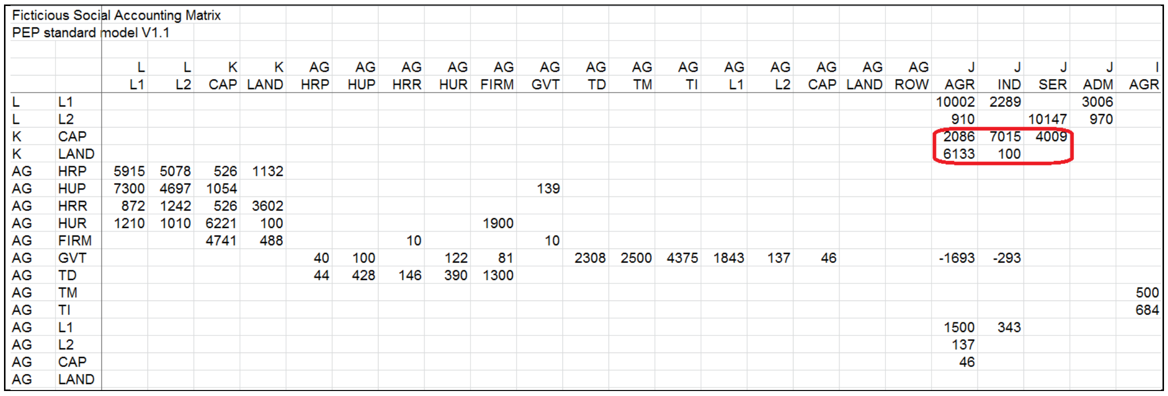
```
EXDO(x)          = SAM('X',x,'AG','ROW');
INVO(i)          = SAM('I',i,'OTH','INV');
VSTKO(i)         = SAM('I',i,'OTH','VSTK');
IMO(m)           = SAM('AG','ROW','I',m);
KDO(k,j)         = SAM('L',k,'J',j);
LDO(l,j)         = SAM('L',l,'J',j);
SFO(f)           = SAM('OTH','INV','AG',f);
```

From figure 40, we can see that we have assigned a value to *KDO(k,j)*. However, the way we assigned a value to *KDO(k,j)* must be incorrect, since the corresponding value is zero[11].

Let's have a look at the SAM and the different labels we use to read the value of *KDO(k,j).* Figure 41 shows the reproduction of the first half of the SAM used for PEP-1-1. The values of *KDO(k,j)* are encircled in red in the figure below.

[11] Note that if the value is actually zero for a sector, then we need to introduce a condition in the computation of lambda_RK(ag,k). It would then be: lambda_RK(ag,k)$KDO(k,j)= lambda_RK(ag,k)/SUM [j,KDO(k,j)];

Figure 41 : First half of the SAM used for PEP 1-1



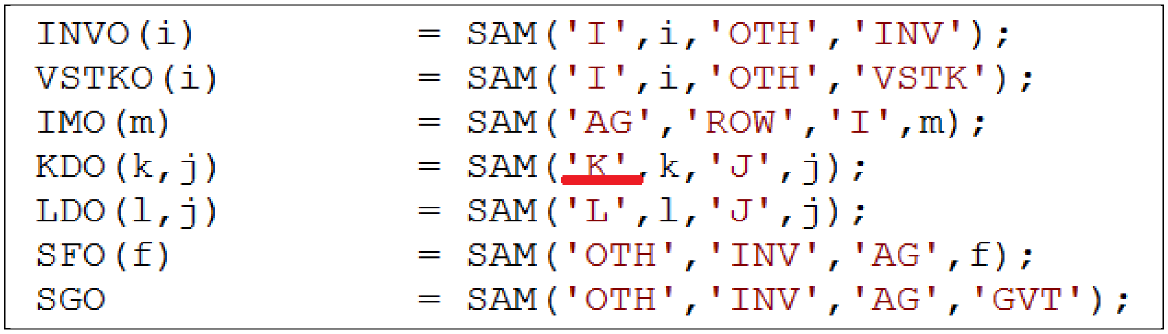
Ficticious Social Accounting Matrix
PEP standard model V1.1

		L	L	K	K	AG	AG	AG	AG	AG	AG	AG	AG	AG	AG	AG	AG	AG	AG	J	J	J	J	I
		L1	L2	CAP	LAND	HRP	HUP	HRR	HUR	FIRM	GVT	TD	TM	TI	L1	L2	CAP	LAND	ROW	AGR	IND	SER	ADM	AGR
L	L1																			10002	2289		3006	
L	L2																			910		10147	970	
K	CAP																			2086	7015	4009		
K	LAND																			6133	100			
AG	HRP	5915	5078	526	1132																			
AG	HUP	7300	4697	1054							139													
AG	HRR	872	1242	526	3602																			
AG	HUR	1210	1010	6221	100					1900														
AG	FIRM			4741	488			10			10													
AG	GVT					40	100		122	81		2308	2500	4375	1843	137	46			-1693	-293			
AG	TD					44	428	146	390	1300														
AG	TM																							500
AG	TI																							684
AG	L1																			1500	343			
AG	L2																			137				
AG	CAP																			46				
AG	LAND																							

Looking at the SAM from figure 41, to assign a value to KDO(k,j), KDO(k,j) should be assigned as (“K”, k,”J”,j).

We can see that this is not the way it was assigned in the GAMS code (see figure 40). We then need to change the way *KDO(k,j)* was assigned by substituting “K” for “L” for the first dimension.

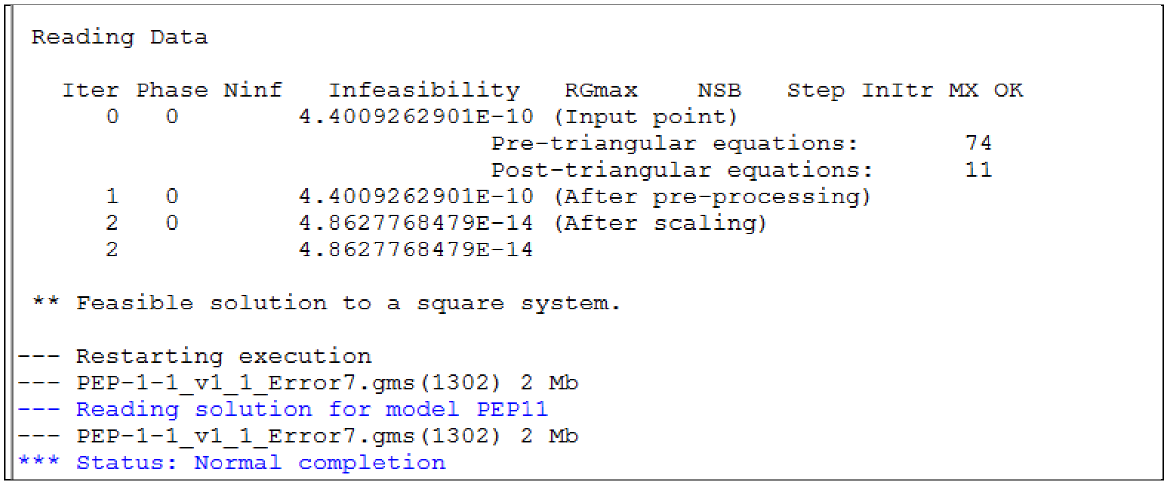
Figure 42 : Example 7 corrected



```
INVO(i)          = SAM('I',i,'OTH','INV');
VSTKO(i)         = SAM('I',i,'OTH','VSTK');
IMO(m)           = SAM('AG','ROW','I',m);
KDO(k,j)         = SAM('K',k,'J',j);
LDO(l,j)         = SAM('L',l,'J',j);
SFO(f)           = SAM('OTH','INV','AG',f);
SGO              = SAM('OTH','INV','AG','GVT');
```

We can then re-run the model to verify that the error has been corrected.

Figure 43 : Process window of example 7 after correction

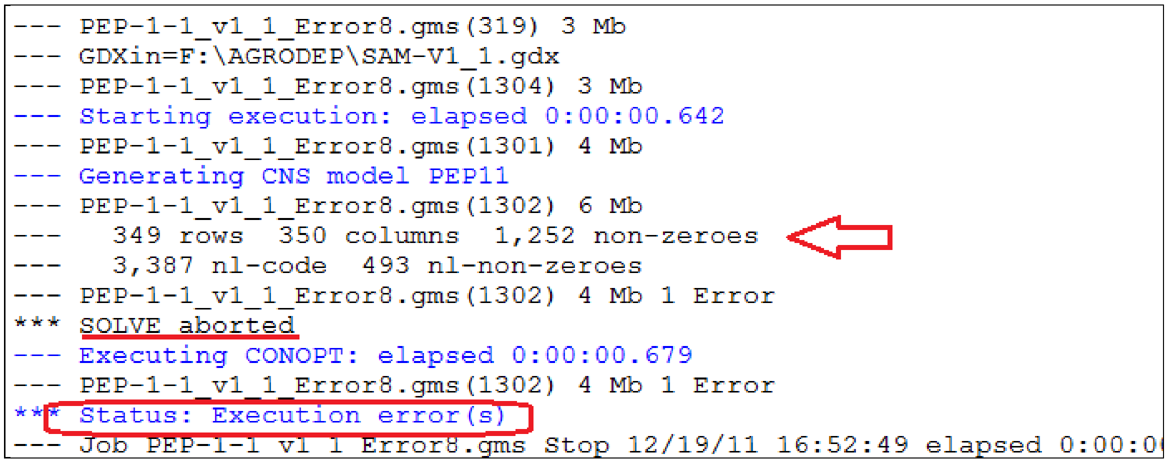


```
 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                               Pre-triangular equations:       74
                               Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error7.gms(1302) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error7.gms(1302) 2 Mb
*** Status: Normal completion
```

Example 8:

Example 8 is another example of an execution error. As indicated by the red arrow, the model is not squared.

Figure 44 : Process window of example 8

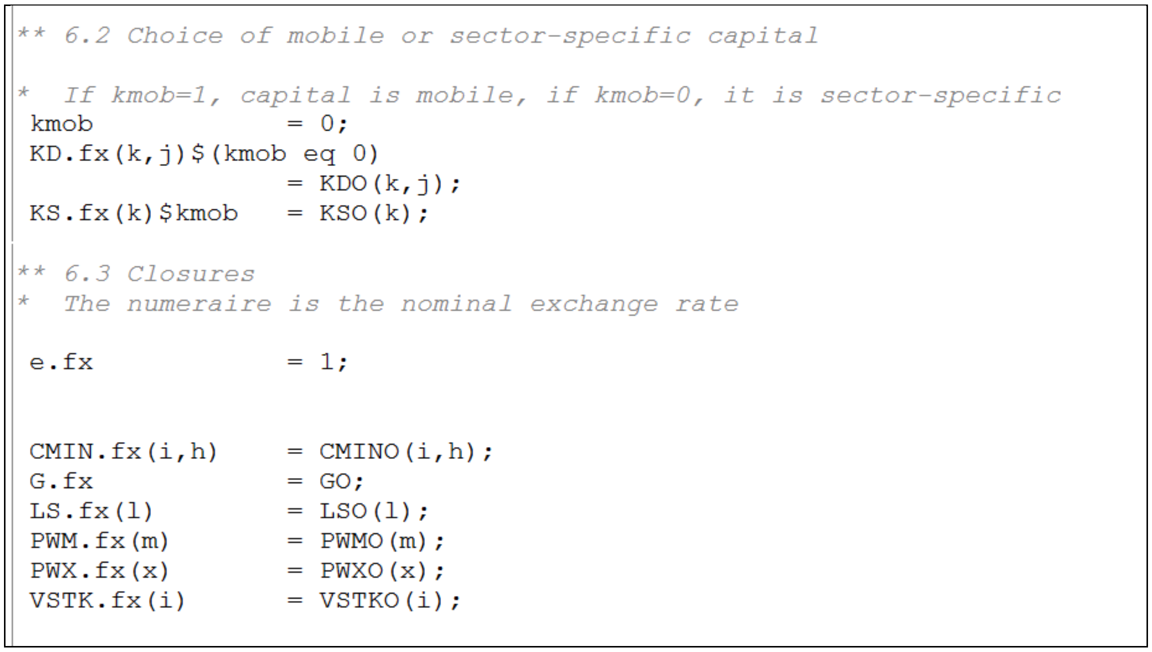


```
--- PEP-1-1_v1_1_Error8.gms(319) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error8.gms(1304) 3 Mb
--- Starting execution: elapsed 0:00:00.642
--- PEP-1-1_v1_1_Error8.gms(1301) 4 Mb
--- Generating CNS model PEP11
--- PEP-1-1_v1_1_Error8.gms(1302) 6 Mb
---   349 rows  350 columns  1,252 non-zeroes
---   3,387 nl-code  493 nl-non-zeroes
--- PEP-1-1_v1_1_Error8.gms(1302) 4 Mb 1 Error
*** SOLVE aborted
--- Executing CONOPT: elapsed 0:00:00.679
--- PEP-1-1_v1_1_Error8.gms(1302) 4 Mb 1 Error
*** Status: Execution error(s)
--- Job PEP-1-1_v1_1_Error8.gms Stop 12/19/11 16:52:49 elapsed 0:00:0
```

In this example, we have 349 rows, meaning equations, and 350 columns, referring to endogenous variables, indicating that we need to fix a variable or add an equation.

This type of error is not directly linked to the writing in GAMS. In this case, the modeller has to go back to his or her pen and paper, and check his list of variables and equations. Since there is only a difference of one variable or equation, the first hint would be to go and check the closure in the GAMS code, to make sure we have not forgotten an exogenous variable in the list. In PEP-1-1, the closure of the module is at the very end of the GAMS file.

Figure 45 : Closure rules in GAMS in example 8


```
** 6.2 Choice of mobile or sector-specific capital

*  If kmob=1, capital is mobile, if kmob=0, it is sector-specific
 kmob               = 0;
 KD.fx(k,j)$(kmob eq 0)
                    = KDO(k,j);
 KS.fx(k)$kmob      = KSO(k);

** 6.3 Closures
*  The numeraire is the nominal exchange rate

 e.fx               = 1;


 CMIN.fx(i,h)       = CMINO(i,h);
 G.fx               = GO;
 LS.fx(l)           = LSO(l);
 PWM.fx(m)          = PWMO(m);
 PWX.fx(x)          = PWXO(x);
 VSTK.fx(i)         = VSTKO(i);
```

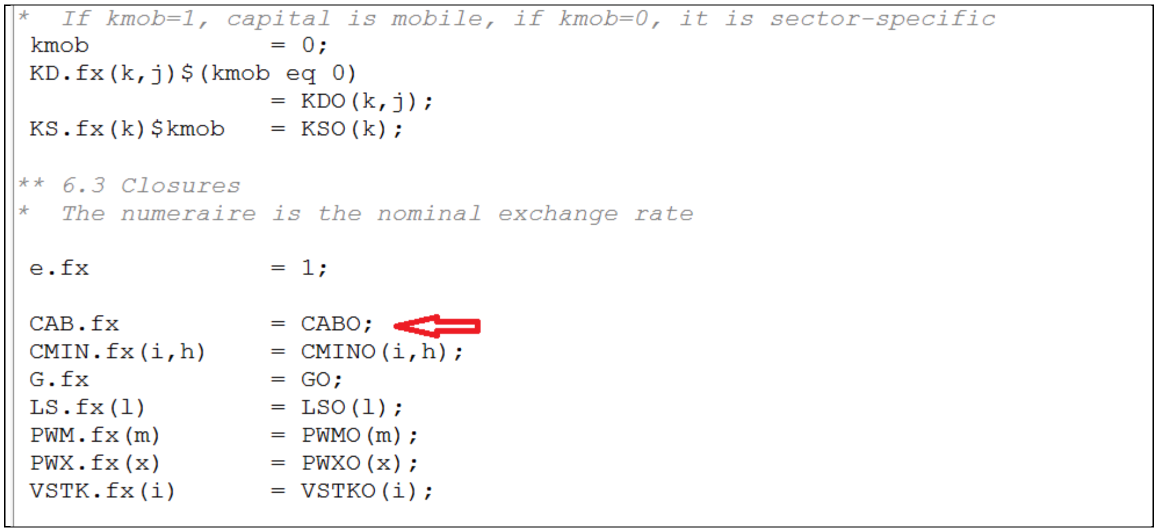
Of course, choices of exogenous variables depend on the modeller’s choices and are linked to the economic problem he or she wants to analyse. Consequently, the closure must be consistent with the economics and the way equations are written. Thus, for this specific type of error, the modeller wants to go back to his list of equations and variables that correspond with the economic closure he has chosen. The modeller then needs to check if the list of exogenous variables is complete.

In the case of PEP1-1, the nominal exchange rate is the “numéraire” of the model. Labour supplies are exogenous. When we apply the assumption of a small country, world prices are given, meaning that they are exogenous. Changes in inventories and minimal consumptions for households are also fixed. Finally, government’s spending and the current account balance are both exogenous. For capital supply, it depends on the choice of the modeller.

From this list of exogenous variables, if we compare with the list of exogenous variables in the GAMS code, we can see that the current account balance (*CAB*) is missing.

We then need to add the current account balance as an exogenous variable, as it is consistent with the underlying assumptions in the model.

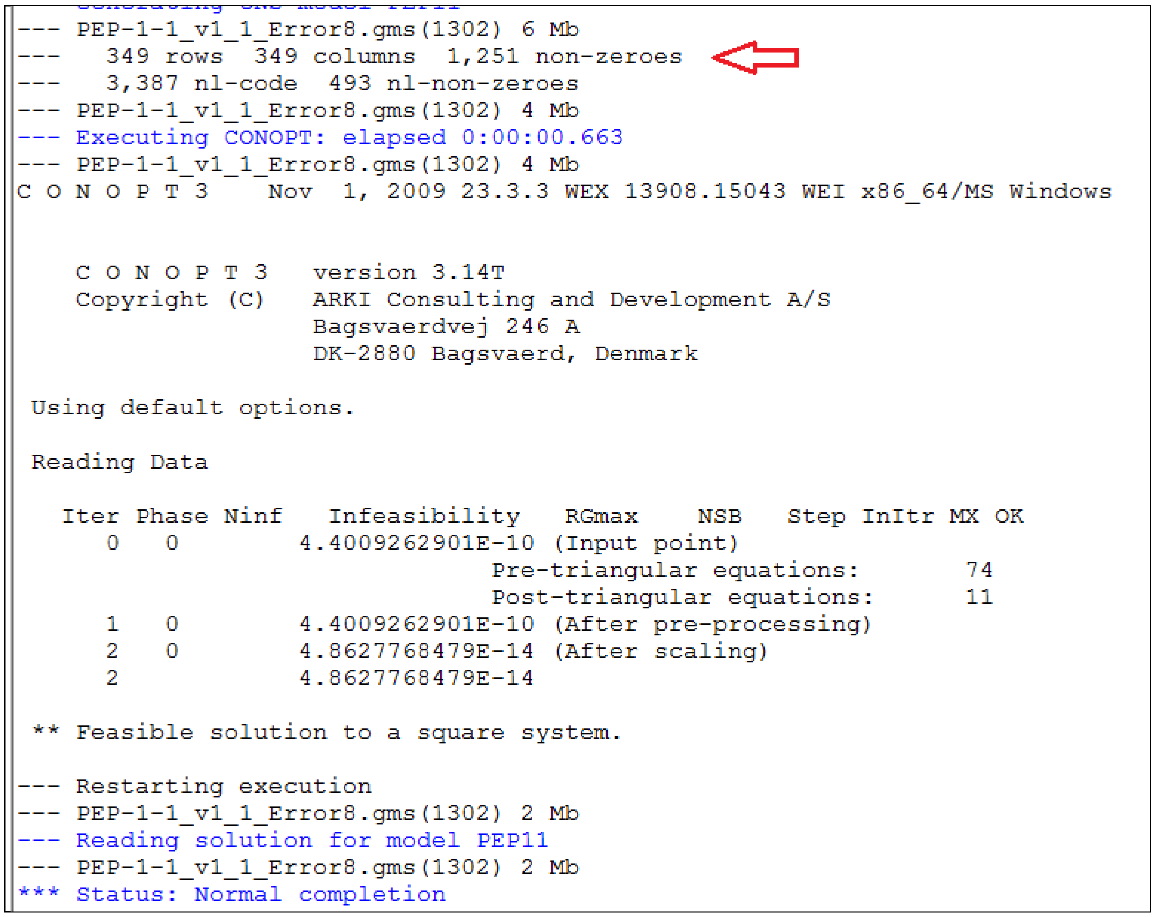
Figure 46 : Correction of example 8



```
*   If kmob=1, capital is mobile, if kmob=0, it is sector-specific
 kmob                = 0;
 KD.fx(k,j)$(kmob eq 0)
                     = KDO(k,j);
 KS.fx(k)$kmob       = KSO(k);

**  6.3 Closures
*   The numeraire is the nominal exchange rate

 e.fx                = 1;

 CAB.fx              = CABO;
 CMIN.fx(i,h)        = CMINO(i,h);
 G.fx                = GO;
 LS.fx(l)            = LSO(l);
 PWM.fx(m)           = PWMO(m);
 PWX.fx(x)           = PWXO(x);
 VSTK.fx(i)          = VSTKO(i);
```

Once we have added this exogenous variable, we can press *F9* and check if the model runs properly.

Figure 47 : Process window of example 8 after correction

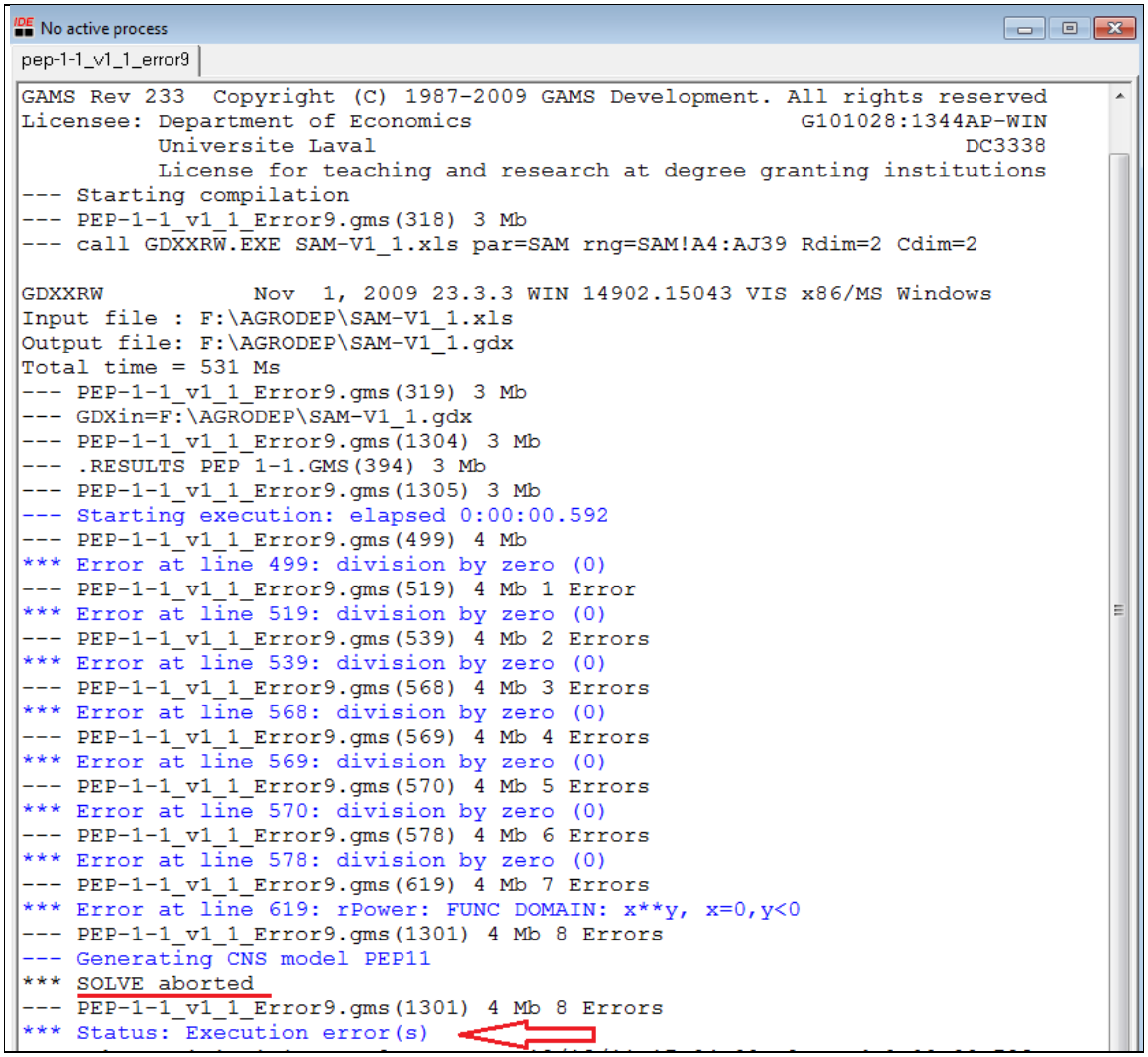


```
--- PEP-1-1_v1_1_Error8.gms(1302) 6 Mb
---   349 rows  349 columns  1,251 non-zeroes
---   3,387 nl-code  493 nl-non-zeroes
--- PEP-1-1_v1_1_Error8.gms(1302) 4 Mb
--- Executing CONOPT: elapsed 0:00:00.663
--- PEP-1-1_v1_1_Error8.gms(1302) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                 Pre-triangular equations:       74
                                 Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error8.gms(1302) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error8.gms(1302) 2 Mb
*** Status: Normal completion
```

Example 9:

Example 9 is presented as follows:

Figure 48 : Process window of example 9



No active process

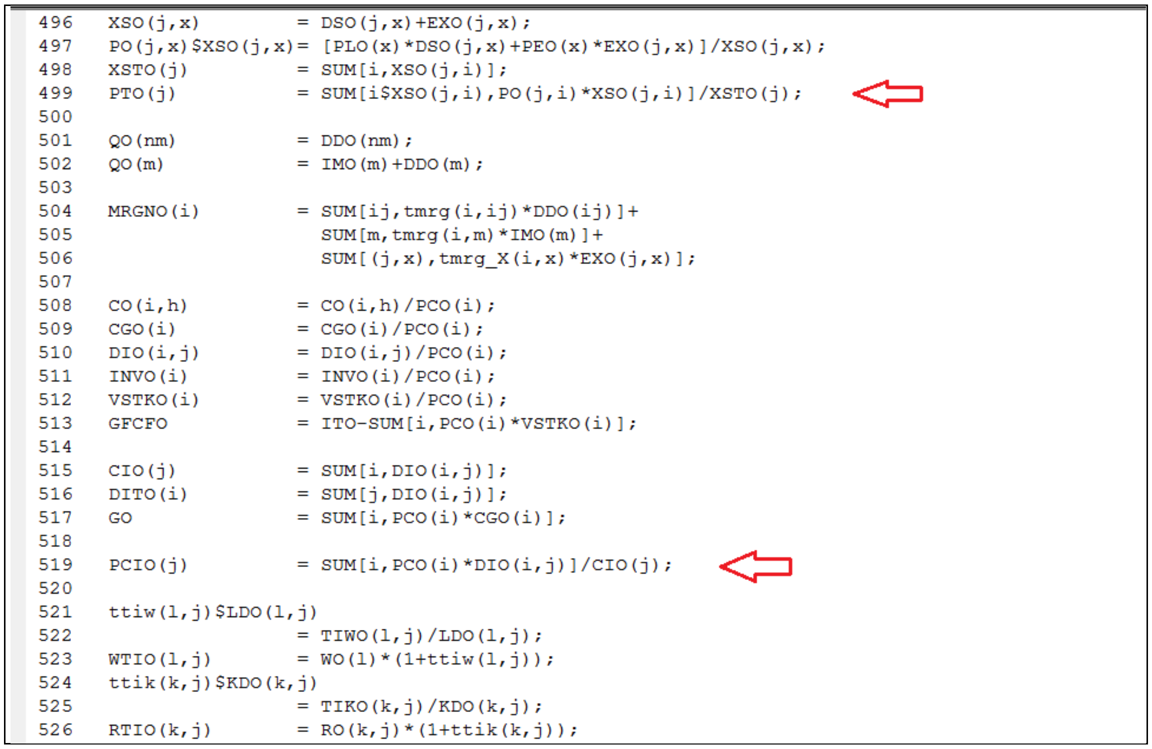
pep-1-1_v1_1_error9

```
GAMS Rev 233  Copyright (C) 1987-2009 GAMS Development. All rights reserved
Licensee: Department of Economics                           G101028:1344AP-WIN
          Universite Laval                                              DC3338
          License for teaching and research at degree granting institutions
--- Starting compilation
--- PEP-1-1_v1_1_Error9.gms(318) 3 Mb
--- call GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2

GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 531 Ms
--- PEP-1-1_v1_1_Error9.gms(319) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error9.gms(1304) 3 Mb
--- .RESULTS PEP 1-1.GMS(394) 3 Mb
--- PEP-1-1_v1_1_Error9.gms(1305) 3 Mb
--- Starting execution: elapsed 0:00:00.592
--- PEP-1-1_v1_1_Error9.gms(499) 4 Mb
*** Error at line 499: division by zero (0)
--- PEP-1-1_v1_1_Error9.gms(519) 4 Mb 1 Error
*** Error at line 519: division by zero (0)
--- PEP-1-1_v1_1_Error9.gms(539) 4 Mb 2 Errors
*** Error at line 539: division by zero (0)
--- PEP-1-1_v1_1_Error9.gms(568) 4 Mb 3 Errors
*** Error at line 568: division by zero (0)
--- PEP-1-1_v1_1_Error9.gms(569) 4 Mb 4 Errors
*** Error at line 569: division by zero (0)
--- PEP-1-1_v1_1_Error9.gms(570) 4 Mb 5 Errors
*** Error at line 570: division by zero (0)
--- PEP-1-1_v1_1_Error9.gms(578) 4 Mb 6 Errors
*** Error at line 578: division by zero (0)
--- PEP-1-1_v1_1_Error9.gms(619) 4 Mb 7 Errors
*** Error at line 619: rPower: FUNC DOMAIN: x**y, x=0,y<0
--- PEP-1-1_v1_1_Error9.gms(1301) 4 Mb 8 Errors
--- Generating CNS model PEP11
*** SOLVE aborted
--- PEP-1-1_v1_1_Error9.gms(1301) 4 Mb 8 Errors
*** Status: Execution error(s)
```

As shown in figure 48, the solving aborted due to execution errors. At different lines, GAMS has to divide by zero

We double-click on *Error at line 499*. This leads us to the listing file, where we can go to line 499.

Figure 49 : Listing file for example 9 (1)



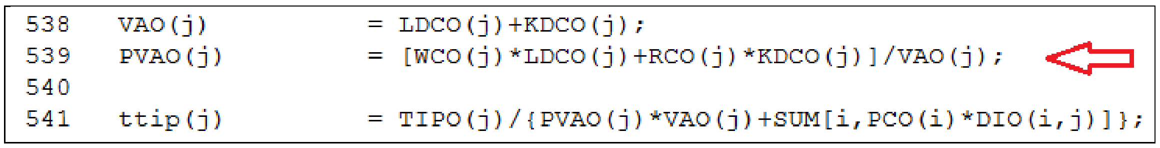
```
496   XSO(j,x)          = DSO(j,x)+EXO(j,x);
497   PO(j,x)$XSO(j,x)= [PLO(x)*DSO(j,x)+PEO(x)*EXO(j,x)]/XSO(j,x);
498   XSTO(j)           = SUM[i,XSO(j,i)];
499   PTO(j)            = SUM[i$XSO(j,i),PO(j,i)*XSO(j,i)]/XSTO(j);
500
501   QO(nm)            = DDO(nm);
502   QO(m)             = IMO(m)+DDO(m);
503
504   MRGNO(i)          = SUM[ij,tmrg(i,ij)*DDO(ij)]+
505                       SUM[m,tmrg(i,m)*IMO(m)]+
506                       SUM[(j,x),tmrg_X(i,x)*EXO(j,x)];
507
508   CO(i,h)           = CO(i,h)/PCO(i);
509   CGO(i)            = CGO(i)/PCO(i);
510   DIO(i,j)          = DIO(i,j)/PCO(i);
511   INVO(i)           = INVO(i)/PCO(i);
512   VSTKO(i)          = VSTKO(i)/PCO(i);
513   GFCFO             = ITO-SUM[i,PCO(i)*VSTKO(i)];
514
515   CIO(j)            = SUM[i,DIO(i,j)];
516   DITO(i)           = SUM[j,DIO(i,j)];
517   GO                = SUM[i,PCO(i)*CGO(i)];
518
519   PCIO(j)           = SUM[i,PCO(i)*DIO(i,j)]/CIO(j);
520
521   ttiw(l,j)$LDO(l,j)
522                     = TIWO(l,j)/LDO(l,j);
523   WTIO(l,j)         = WO(l)*(1+ttiw(l,j));
524   ttik(k,j)$KDO(k,j)
525                     = TIKO(k,j)/KDO(k,j);
526   RTIO(k,j)         = RO(k,j)*(1+ttik(k,j));
```

From figure 48, we know that the division by zero happens several times (i.e. at different lines).Our first assumption would be that there is only one variable or parameter that takes zero as a value, and we could have repeated errors.

However, from figure 49, we can check for both lines 499 and 519 (red arrows in figure 49). Each line refers to a different denominator, meaning a different variable. Indeed, at line 499, it seems that *XSTO(j)* is equal to zero for at least one of the sectors, whereas at line 519, it is *CIO(j)*.

Figures 50 to 52 reproduce the different lines where the denominator is equal to zero, for at least one element.

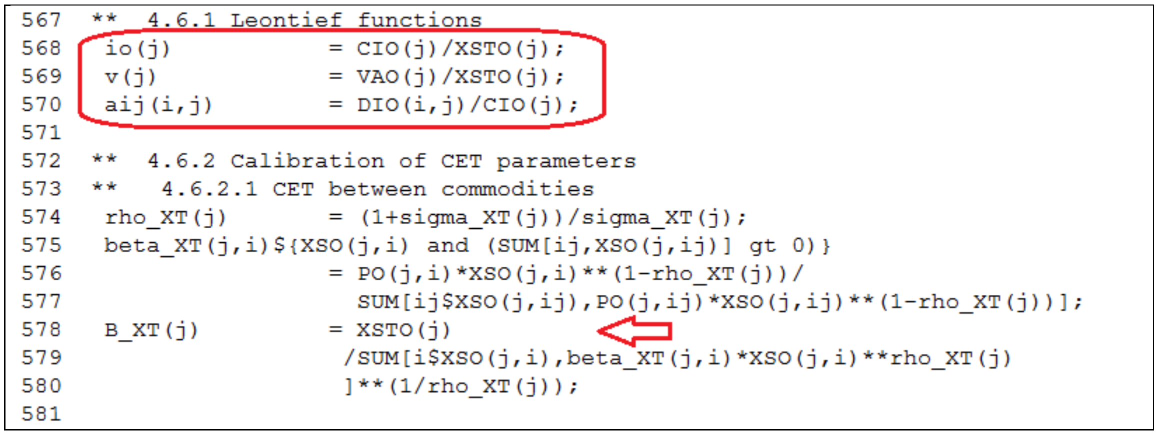
Figure 50 : Listing file for example 9 (2)



```
538   VAO(j)          = LDCO(j)+KDCO(j);
539   PVAO(j)         = [WCO(j)*LDCO(j)+RCO(j)*KDCO(j)]/VAO(j);
540
541   ttip(j)         = TIPO(j)/{PVAO(j)*VAO(j)+SUM[i,PCO(i)*DIO(i,j)]};
```

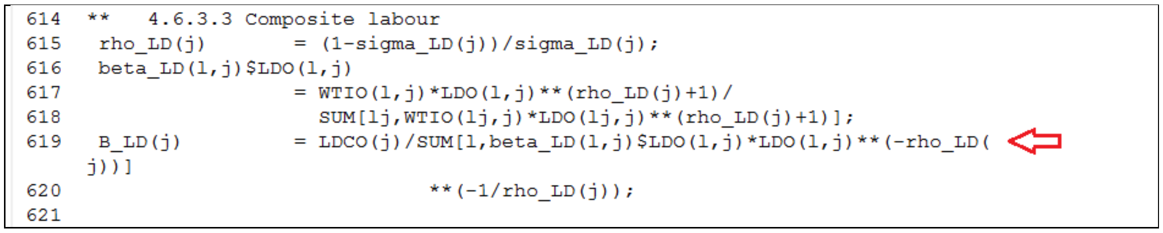
From figure 50, it is *VAO(j)* that takes zero for value at least for one of the sector.

Figure 51 : Listing file for example 9 (3)



```
**   4.6.1 Leontief functions
 io(j)               = CIO(j)/XSTO(j);
 v(j)                = VAO(j)/XSTO(j);
 aij(i,j)            = DIO(i,j)/CIO(j);

**   4.6.2 Calibration of CET parameters
**     4.6.2.1 CET between commodities
 rho_XT(j)           = (1+sigma_XT(j))/sigma_XT(j);
 beta_XT(j,i)${XSO(j,i) and (SUM[ij,XSO(j,ij)] gt 0)}
                     = PO(j,i)*XSO(j,i)**(1-rho_XT(j))/
                       SUM[ij$XSO(j,ij),PO(j,ij)*XSO(j,ij)**(1-rho_XT(j))];
 B_XT(j)             = XSTO(j)
                      /SUM[i$XSO(j,i),beta_XT(j,i)*XSO(j,i)**rho_XT(j)
                      ]**(1/rho_XT(j));

```

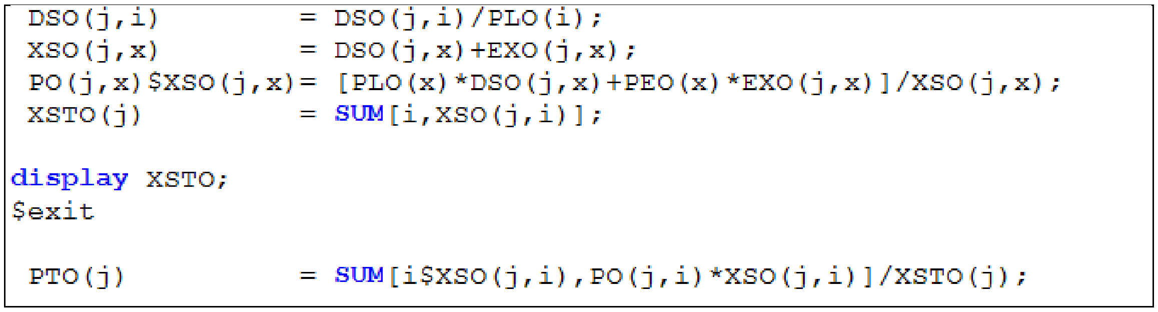
Figure 52 : Listing file for example 9 (4)



```
**   4.6.3.3 Composite labour
 rho_LD(j)          = (1-sigma_LD(j))/sigma_LD(j);
 beta_LD(l,j)$LDO(l,j)
                    = WTIO(l,j)*LDO(l,j)**(rho_LD(j)+1)/
                      SUM[lj,WTIO(lj,j)*LDO(lj,j)**(rho_LD(j)+1)];
 B_LD(j)            = LDCO(j)/SUM[l,beta_LD(l,j)$LDO(l,j)*LDO(l,j)**(-rho_LD(
j))]
                              **(-1/rho_LD(j));

```

Here, the only clue we have is that the division by zero affects sectors, and not commodities. Moreover, *VAO(j*), *CIO(j)* and *XSTO(j)* are linked together through the value added-input output coefficient. One way to proceed is to ask GAMS to display the value of the variable *XSTO(j).* For example, just before line 499, and right after the computation of *XSTO(j)*, we ask GAMS to stop the compilation immediately after the display command[12].

Figure 53 : Display of XSTO(j) in example 9



```
 DSO(j,i)            = DSO(j,i)/PLO(i);
 XSO(j,x)            = DSO(j,x)+EXO(j,x);
 PO(j,x)$XSO(j,x)=  [PLO(x)*DSO(j,x)+PEO(x)*EXO(j,x)]/XSO(j,x);
 XSTO(j)             = SUM[i,XSO(j,i)];

display XSTO;
$exit

 PTO(j)              = SUM[i$XSO(j,i),PO(j,i)*XSO(j,i)]/XSTO(j);
```

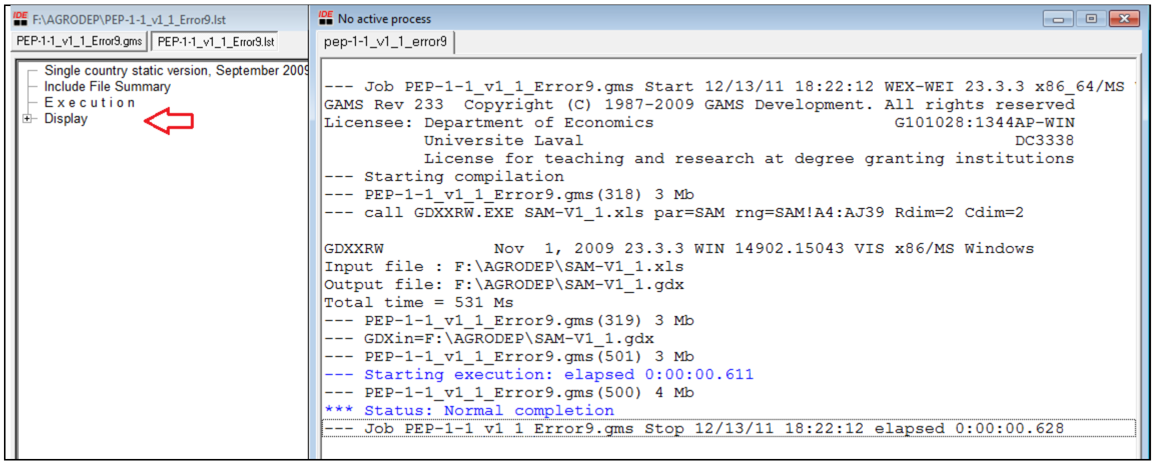
[12] Note that we could have asked GAMS to display the values of the three variables mentioned above. As *XSTO(j)* is the first one where the problem appears, we chose to have a look only at this variable.

Just before computing *PTO(j),* we ask GAMS to compute *XSTO*, using the command *DISPLAY*, and then we ask it to stop the solve right after the display, using the *$exit*.

Including *$exit* is not compulsory, but we find it useful as it will only solve the part of the model we are interested in. In other words, GAMS won't pay attention to the subsequent part of the GAMS code, and thus we should not have any errors up to this point.

Running the model with these new commands, we obtain the following listing window:

Figure 54 : Listing window of example 9

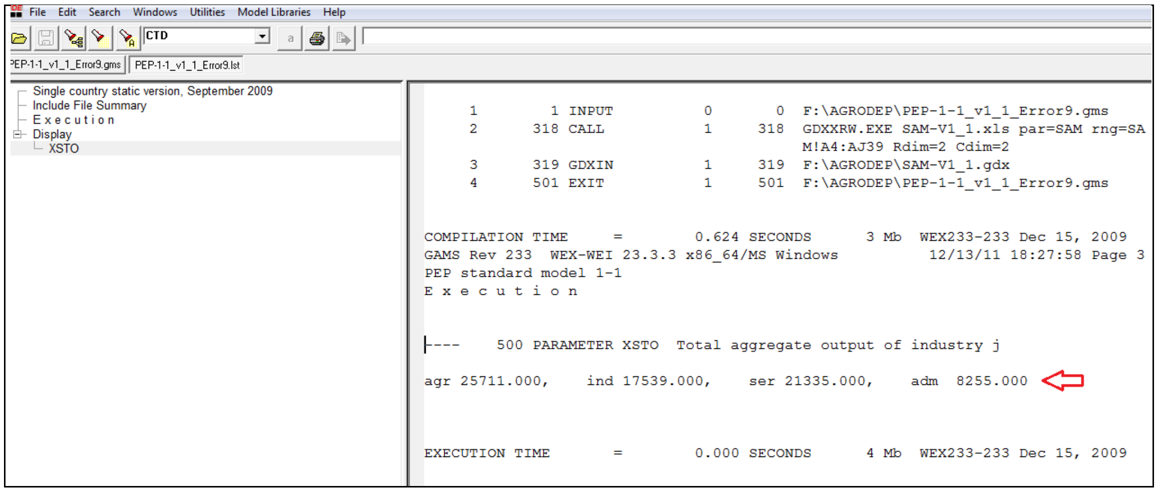


```
--- Job PEP-1-1_v1_1_Error9.gms Start 12/13/11 18:22:12 WEX-WEI 23.3.3 x86_64/MS
GAMS Rev 233  Copyright (C) 1987-2009 GAMS Development. All rights reserved
Licensee: Department of Economics                          G101028:1344AP-WIN
          Universite Laval                                             DC3338
          License for teaching and research at degree granting institutions
--- Starting compilation
--- PEP-1-1_v1_1_Error9.gms(318) 3 Mb
--- call GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2

GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 531 Ms
--- PEP-1-1_v1_1_Error9.gms(319) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error9.gms(501) 3 Mb
--- Starting execution: elapsed 0:00:00.611
--- PEP-1-1_v1_1_Error9.gms(500) 4 Mb
*** Status: Normal completion
--- Job PEP-1-1 v1 1 Error9.gms Stop 12/13/11 18:22:12 elapsed 0:00:00.628
```

Here we are interested in the display that appears in the listing file (red arrow).

By clicking on the "+" next to Display, *XSTO* appears, and when we double-click on it, it brings us in the listing file where GAMS computes the parameter *XSTO*.

Figure 55 : Display of XSTO in the listing file (example 9)


File Edit Search Windows Utilities Model Libraries Help
CTD
PEP-1-1_v1_1_Error9.gms
PEP-1-1_v1_1_Error9.lst
Single country static version, September 2009
Include File Summary
Execution
Display
XSTO

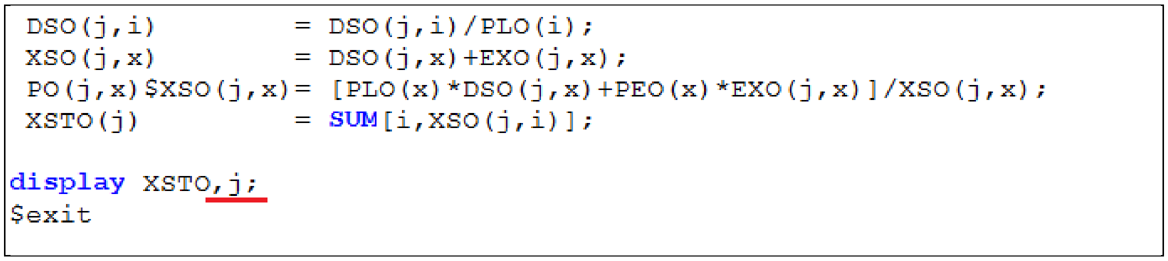
```
     1        1 INPUT          0       0  F:\AGRODEP\PEP-1-1_v1_1_Error9.gms
     2      318 CALL           1     318  GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SA
                                          M!A4:AJ39 Rdim=2 Cdim=2
     3      319 GDXIN          1     319  F:\AGRODEP\SAM-V1_1.gdx
     4      501 EXIT           1     501  F:\AGRODEP\PEP-1-1_v1_1_Error9.gms


COMPILATION TIME     =        0.624 SECONDS      3 Mb  WEX233-233 Dec 15, 2009
GAMS Rev 233  WEX-WEI 23.3.3 x86_64/MS Windows          12/13/11 18:27:58 Page 3
PEP standard model 1-1
E x e c u t i o n


----    500 PARAMETER XSTO  Total aggregate output of industry j

agr 25711.000,    ind 17539.000,    ser 21335.000,    adm  8255.000



EXECUTION TIME       =        0.000 SECONDS      4 Mb  WEX233-233 Dec 15, 2009
```

From figure 55, we find out that *XSTO* is computed for the four sectors (*agr,ind,ser adm*), and none of the values are equal to zero.

From this point, there are two ways to continue: either we check the sectors in the set declaration at the beginning of the GAMS file, or we can ask GAMS to display the different sectors. In this case, we will ask GAMS to display the different sectors.

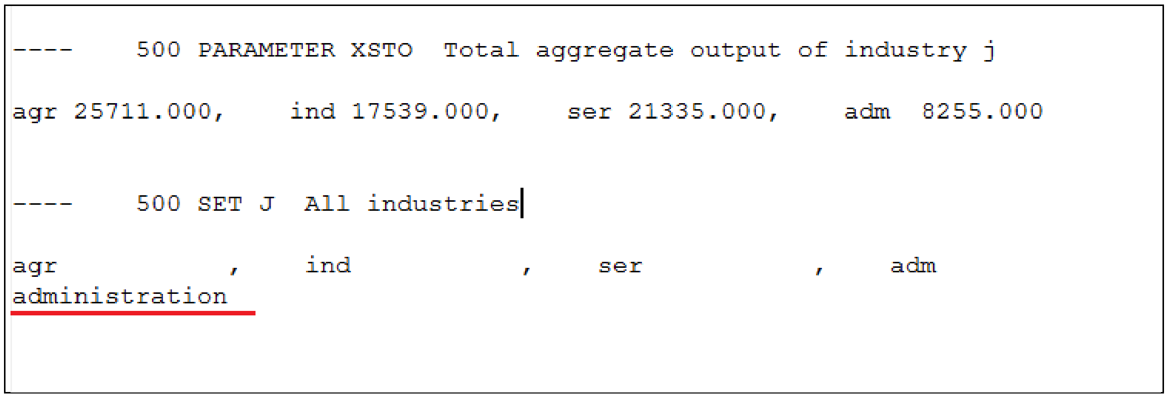
In the GAMS file, next to display *XSTO*, we add *j,* which refers to the set of activities.

Figure 56 : Display of XSTO(j) and j in example 9


```
 DSO(j,i)          = DSO(j,i)/PLO(i);
 XSO(j,x)          = DSO(j,x)+EXO(j,x);
 PO(j,x)$XSO(j,x)= [PLO(x)*DSO(j,x)+PEO(x)*EXO(j,x)]/XSO(j,x);
 XSTO(j)           = SUM[i,XSO(j,i)];

display XSTO,j;
$exit
```

After running the model, we have a look at the values displayed for *j* in the listing file:

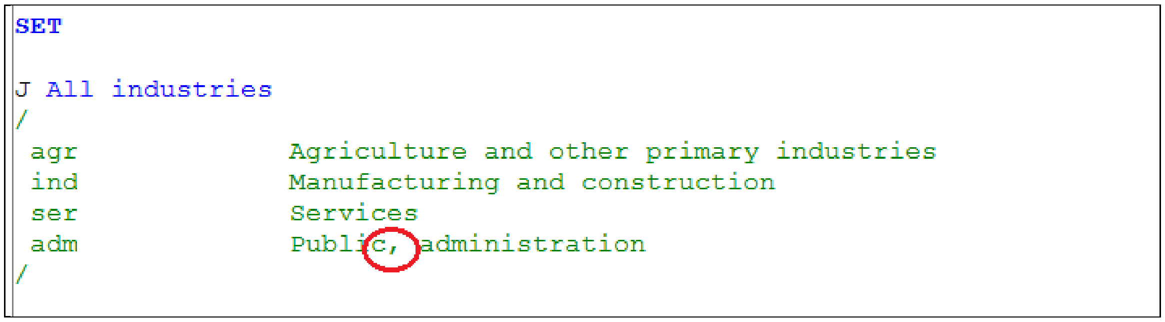
Figure 57 : Display of XSTO and j in the listing file (example 9)


```
----    500 PARAMETER XSTO  Total aggregate output of industry j

agr 25711.000,    ind 17539.000,    ser 21335.000,    adm  8255.000


----    500 SET J  All industries

agr            ,    ind            ,    ser            ,    adm
administration
```

From figure 57, we find out that *j* refers to all sectors: *agr,ind,ser,adm,administration*. We can immediately recognize the problem because we defined four different sectors, and GAMS displays five sectors, adding “administration” as an activity. We have to find out why GAMS considers “administration” as an activity.

For this, we need to go to where the sets are defined. In PEP 1-1, they are defined at the very beginning of the GAMS code.

Figure 58 : Set definition in example 9

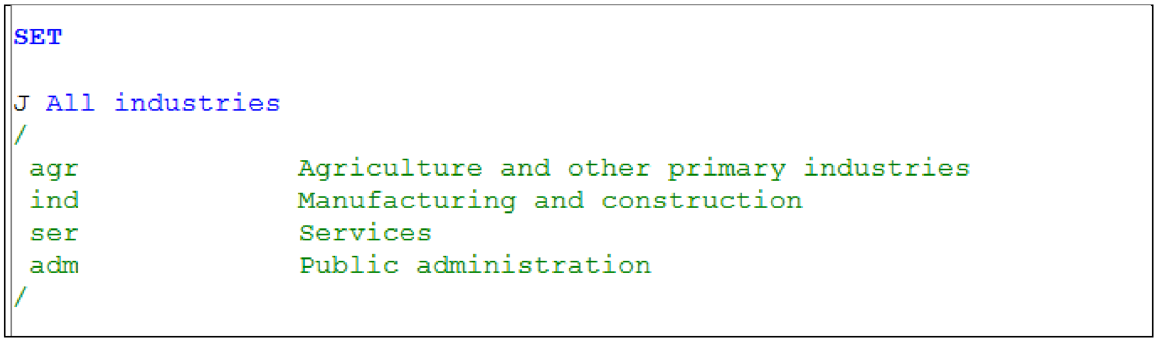

```
SET

J All industries
/
 agr                Agriculture and other primary industries
 ind                Manufacturing and construction
 ser                Services
 adm                Public, administration
/
```

From figure 58, we can see that the definition of *adm* sector is written as *public, administration*, with a comma in between. For GAMS, the fact that a comma has been introduced in the definition of the element *adm* is the same as if a new sector were declared. This means that according to GAMS, *j* refers to five elements and not only to four elements.

Since we only assign a value for the four sectors (*agr,ind,ser,adm*), by default, GAMS would assign the value zero for the fifth sector, *administration*. Note that GAMS does not report in the listing file the sectors which values are equal to zero.

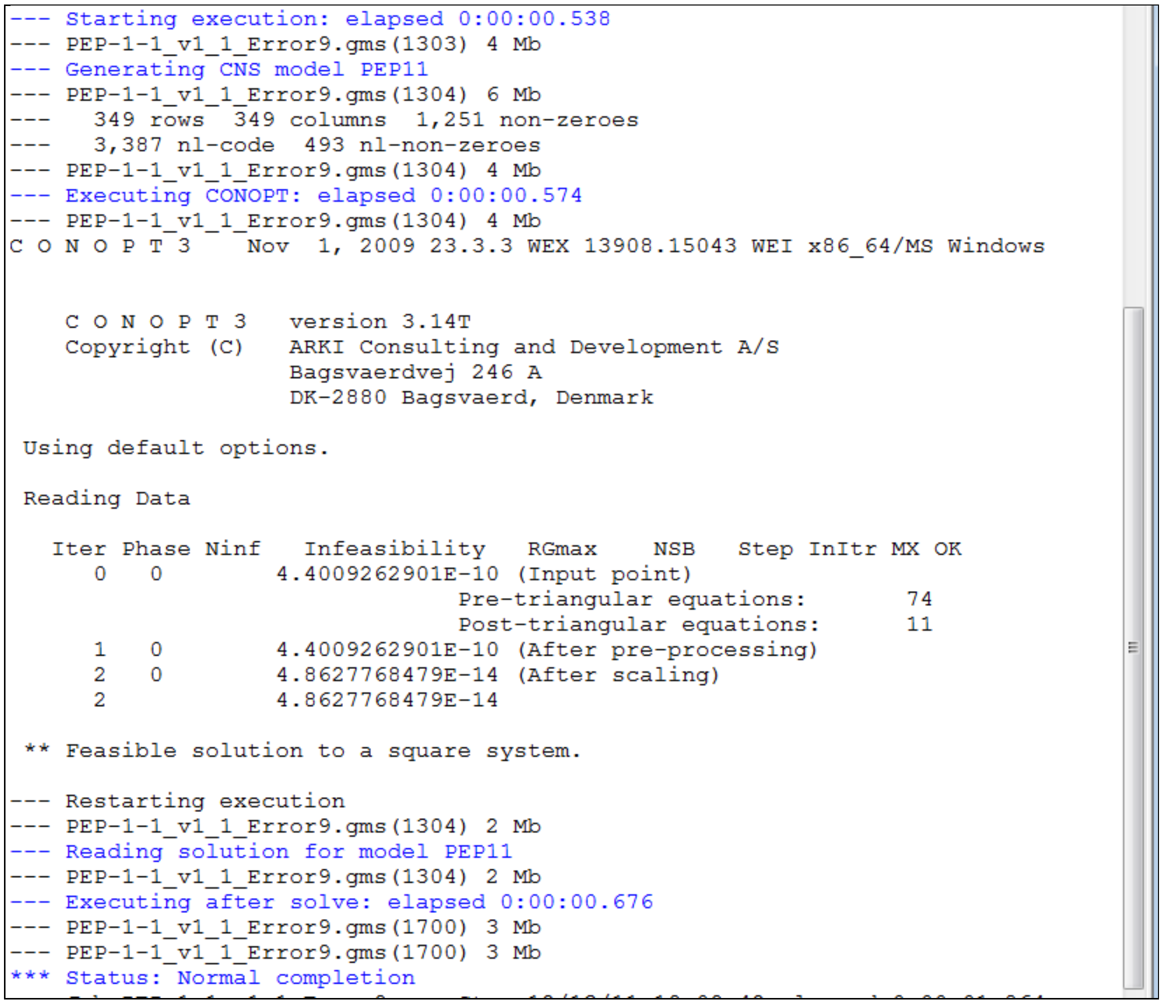
To correct the error, we simply need to remove the comma in the definition of the *adm* sector.

Figure 59 : Correction of example 9


```
SET

J All industries
/
 agr                Agriculture and other primary industries
 ind                Manufacturing and construction
 ser                Services
 adm                Public administration
/
```

Before re-running the model, we need to remove the *$exit* we inserted in the GAMS file. Then, we can re-run the model and check that it runs successfully.

Figure 60 : Process window after correction of example 9

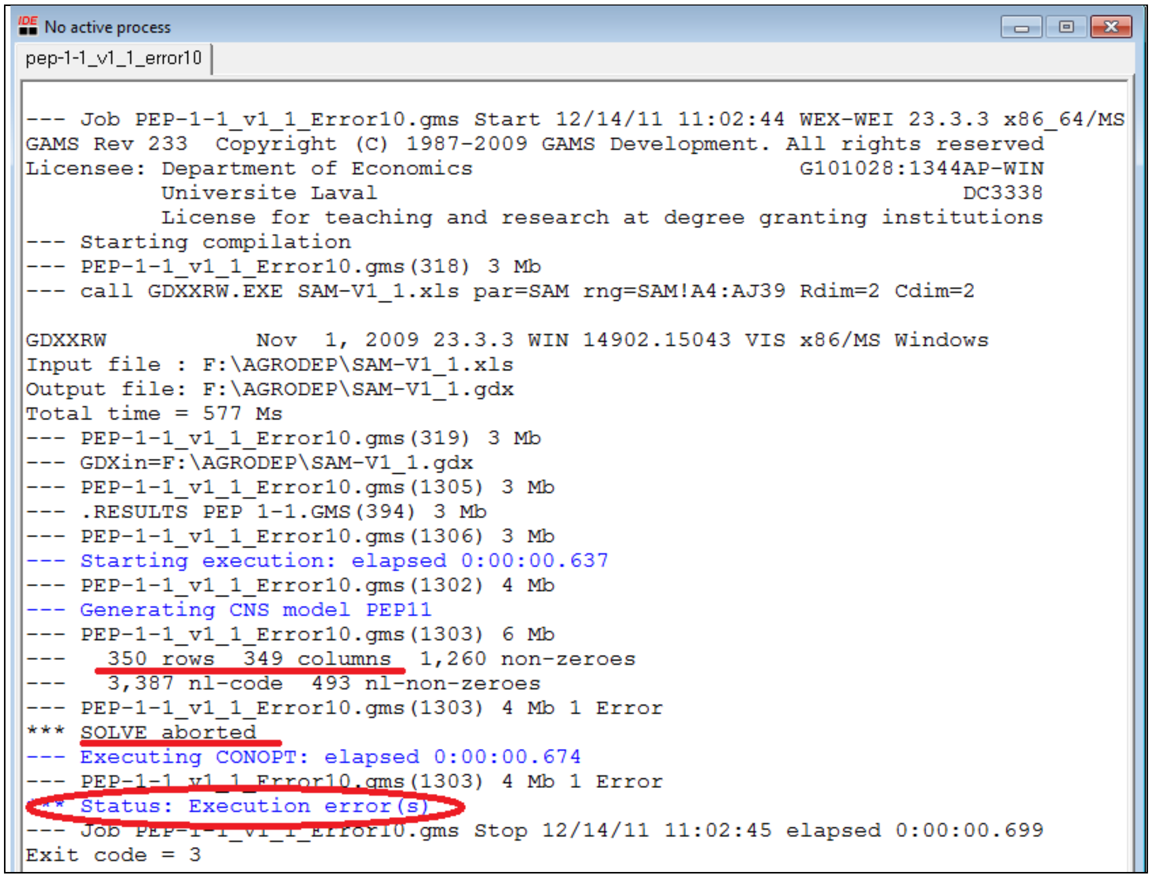


```
--- Starting execution: elapsed 0:00:00.538
--- PEP-1-1_v1_1_Error9.gms(1303) 4 Mb
--- Generating CNS model PEP11
--- PEP-1-1_v1_1_Error9.gms(1304) 6 Mb
---   349 rows  349 columns  1,251 non-zeroes
---   3,387 nl-code  493 nl-non-zeroes
--- PEP-1-1_v1_1_Error9.gms(1304) 4 Mb
--- Executing CONOPT: elapsed 0:00:00.574
--- PEP-1-1_v1_1_Error9.gms(1304) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                Pre-triangular equations:       74
                                Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error9.gms(1304) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error9.gms(1304) 2 Mb
--- Executing after solve: elapsed 0:00:00.676
--- PEP-1-1_v1_1_Error9.gms(1700) 3 Mb
--- PEP-1-1_v1_1_Error9.gms(1700) 3 Mb
*** Status: Normal completion
```

Example 10:

The process window of example 10 is given as follows:

Figure 61 : Process window of example 10



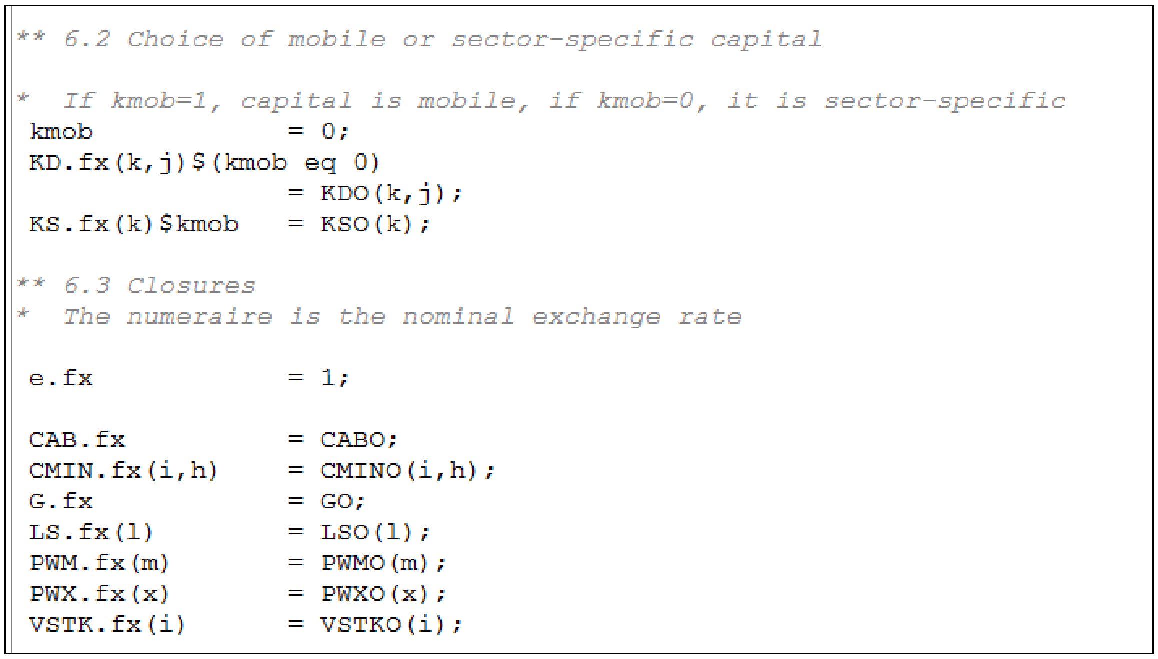
```
--- Job PEP-1-1_v1_1_Error10.gms Start 12/14/11 11:02:44 WEX-WEI 23.3.3 x86_64/MS
GAMS Rev 233  Copyright (C) 1987-2009 GAMS Development. All rights reserved
Licensee: Department of Economics                              G101028:1344AP-WIN
          Universite Laval                                                 DC3338
          License for teaching and research at degree granting institutions
--- Starting compilation
--- PEP-1-1_v1_1_Error10.gms(318) 3 Mb
--- call GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2

GDXXRW           Nov  1, 2009 23.3.3 WIN 14902.15043 VIS x86/MS Windows
Input file : F:\AGRODEP\SAM-V1_1.xls
Output file: F:\AGRODEP\SAM-V1_1.gdx
Total time = 577 Ms
--- PEP-1-1_v1_1_Error10.gms(319) 3 Mb
--- GDXin=F:\AGRODEP\SAM-V1_1.gdx
--- PEP-1-1_v1_1_Error10.gms(1305) 3 Mb
--- .RESULTS PEP 1-1.GMS(394) 3 Mb
--- PEP-1-1_v1_1_Error10.gms(1306) 3 Mb
--- Starting execution: elapsed 0:00:00.637
--- PEP-1-1_v1_1_Error10.gms(1302) 4 Mb
--- Generating CNS model PEP11
--- PEP-1-1_v1_1_Error10.gms(1303) 6 Mb
---   350 rows  349 columns  1,260 non-zeroes
---   3,387 nl-code  493 nl-non-zeroes
--- PEP-1-1_v1_1_Error10.gms(1303) 4 Mb 1 Error
*** SOLVE aborted
--- Executing CONOPT: elapsed 0:00:00.674
--- PEP-1-1_v1_1_Error10.gms(1303) 4 Mb 1 Error
*** Status: Execution error(s)
--- Job PEP-1-1_v1_1_Error10.gms Stop 12/14/11 11:02:45 elapsed 0:00:00.699
Exit code = 3
```

From figure 61, it appears that the solving aborted because the model is not square. We have 350 rows (i.e. equations) and 349 columns (i.e. endogenous variables). This scenario is the opposite of example 7, where we now have too many equations or not enough endogenous variables.

As in example 8, the modeller needs to go back to his pen and paper and check his variables and equations.

The first step in this case would be to go and check in the closure, to see if one variable was fixed when it shouldn't be. If this is the case, we will then have to release the variable.

Figure 62 : Closure rules in GAMS in example 10


```
** 6.2 Choice of mobile or sector-specific capital

*  If kmob=1, capital is mobile, if kmob=0, it is sector-specific
 kmob                = 0;
 KD.fx(k,j)$(kmob eq 0)
                     = KDO(k,j);
 KS.fx(k)$kmob       = KSO(k);

** 6.3 Closures
*  The numeraire is the nominal exchange rate

 e.fx                = 1;

 CAB.fx              = CABO;
 CMIN.fx(i,h)        = CMINO(i,h);
 G.fx                = GO;
 LS.fx(l)            = LSO(l);
 PWM.fx(m)           = PWMO(m);
 PWX.fx(x)           = PWXO(x);
 VSTK.fx(i)          = VSTKO(i);
```

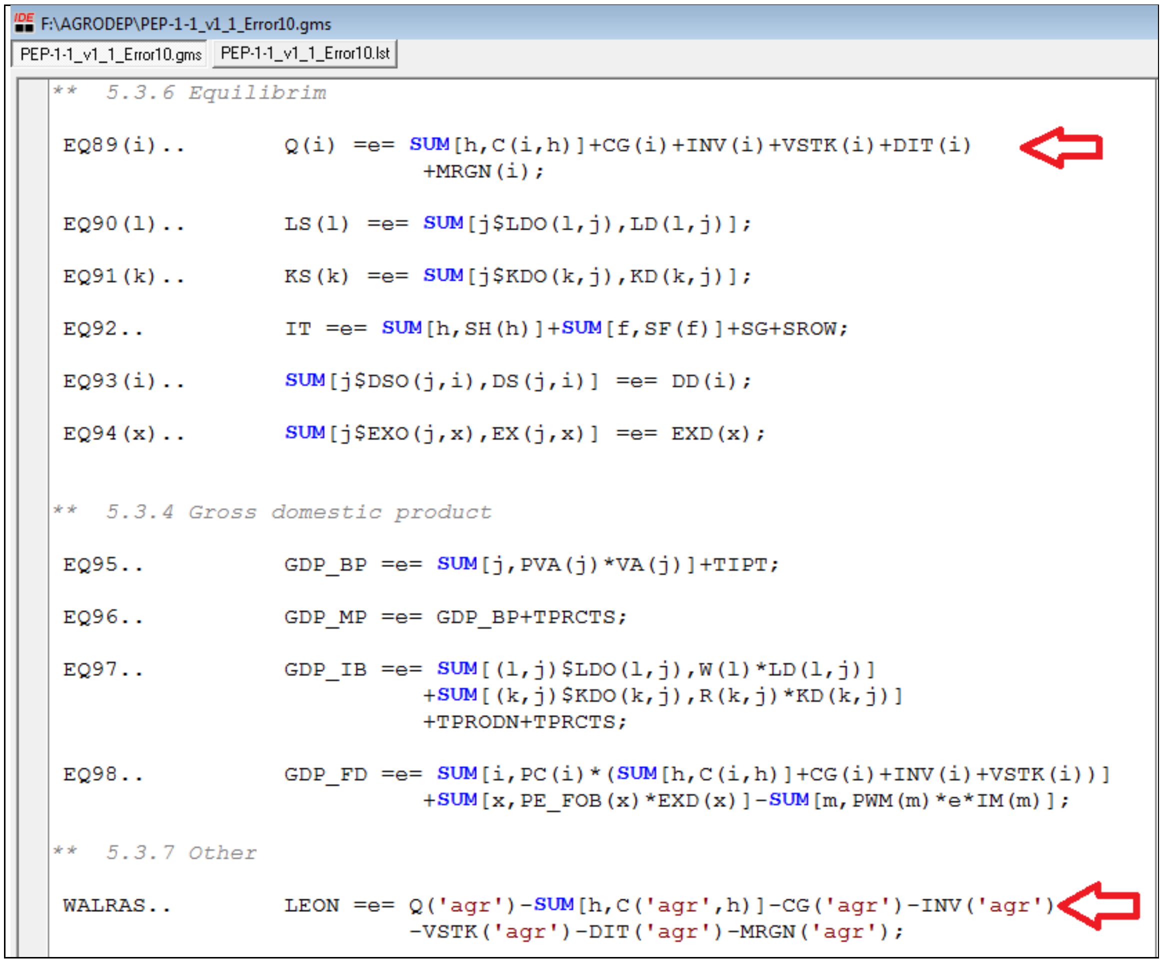
We explained the closure rules followed in PEP 1-1 above (see example 8), and we can see that all the variables that are supposed to be exogenous are indeed fixed.

Therefore, we need to look somewhere else. From figure 61, we know that there is only one extra equation. If there were several extra equations, it could mean that there is an equation defined on a set that should be removed. Here, there is a single extra equation: this could come from a set/subset use.

The modeller needs to go through his or her list of equations and variables, and check for each variable, what the corresponding equation is, and more importantly, what is the related set.

In our application, we go through the GAMS code of PEP 1-1, and check the definition of each variable and equation. When reviewing the equations, we find something interesting regarding the equilibrium equations, as shown in figure 63:

Figure 63 : Equilibrium equations in example 10


F:\AGRODEP\PEP-1-1_v1_1_Error10.gms

PEP-1-1_v1_1_Error10.gms | PEP-1-1_v1_1_Error10.lst

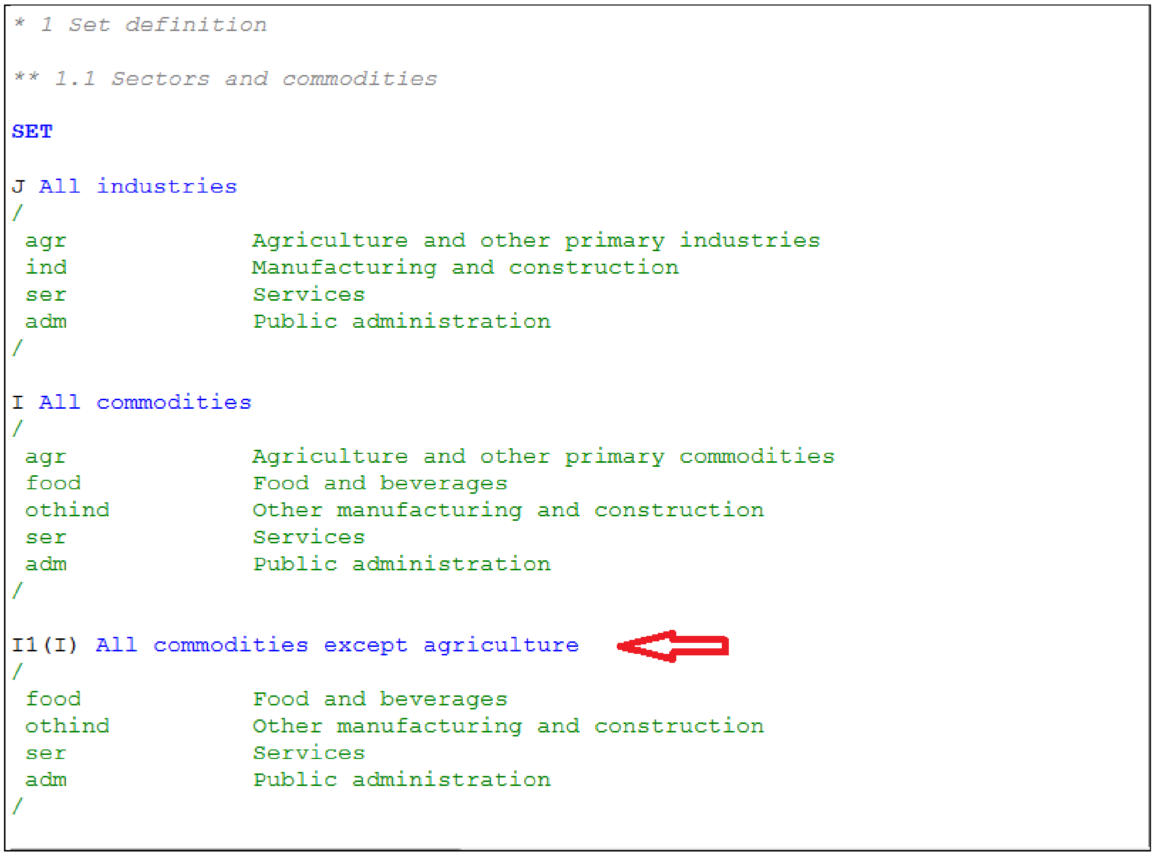
```
**  5.3.6 Equilibrim

 EQ89(i)..          Q(i) =e= SUM[h,C(i,h)]+CG(i)+INV(i)+VSTK(i)+DIT(i)
                             +MRGN(i);

 EQ90(l)..          LS(l)  =e= SUM[j$LDO(l,j),LD(l,j)];

 EQ91(k)..          KS(k)  =e= SUM[j$KDO(k,j),KD(k,j)];

 EQ92..             IT =e= SUM[h,SH(h)]+SUM[f,SF(f)]+SG+SROW;

 EQ93(i)..          SUM[j$DSO(j,i),DS(j,i)] =e= DD(i);

 EQ94(x)..          SUM[j$EXO(j,x),EX(j,x)] =e= EXD(x);


**  5.3.4 Gross domestic product

 EQ95..             GDP_BP  =e= SUM[j,PVA(j)*VA(j)]+TIPT;

 EQ96..             GDP_MP  =e= GDP_BP+TPRCTS;

 EQ97..             GDP_IB  =e= SUM[(l,j)$LDO(l,j),W(l)*LD(l,j)]
                             +SUM[(k,j)$KDO(k,j),R(k,j)*KD(k,j)]
                             +TPRODN+TPRCTS;

 EQ98..             GDP_FD  =e= SUM[i,PC(i)*(SUM[h,C(i,h)]+CG(i)+INV(i)+VSTK(i))]
                             +SUM[x,PE_FOB(x)*EXD(x)]-SUM[m,PWM(m)*e*IM(m)];

**  5.3.7 Other

 WALRAS..           LEON =e= Q('agr')-SUM[h,C('agr',h)]-CG('agr')-INV('agr')
                            -VSTK('agr')-DIT('agr')-MRGN('agr');
```

The two red arrows show the equilibrium on the commodities market. Equation 89 computes the equilibrium for each commodity, as it is defined over the set *i*. Equation Walras computes the variable LEON, that refers to the equilibrium for the agricultural commodity, and yet this equilibrium is already computed in equation 89. In other words, the equation that determines the equilibrium for the agricultural commodity is computed twice.

Now, we must determine which of the equations needs to be removed. Equation Walras computes the variable *LEON*, which represents the equilibrium on the agricultural market. If we were to remove equation WALRAS, the variable LEON would be undefined. If we have a close look at the meaning of equation 89, we find out that this equation represents the equilibrium conditions on the commodities markets. According to Walras' Law, if (n-1) markets are in equilibrium, then the last one is also in equilibrium. Thus, this equation should not be computed over all the commodities, but overall commodities minus one.

At the beginning of the PEP1-1 GAMS code, the sets are defined, and there is a set that refers to all the commodities except agriculture:

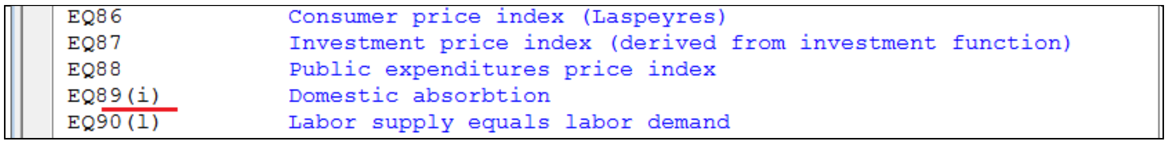
Figure 64 : Set definition in PEP 1-1 :



```
* 1 Set definition

** 1.1 Sectors and commodities

SET

J All industries
/
 agr                Agriculture and other primary industries
 ind                Manufacturing and construction
 ser                Services
 adm                Public administration
/

I All commodities
/
 agr                Agriculture and other primary commodities
 food               Food and beverages
 othind             Other manufacturing and construction
 ser                Services
 adm                Public administration
/

I1(I) All commodities except agriculture
/
 food               Food and beverages
 othind             Other manufacturing and construction
 ser                Services
 adm                Public administration
/
```

Equation 89 should be defined over this subset in accordance with Walras' law. We then need to change the set in the declaration of the equations:

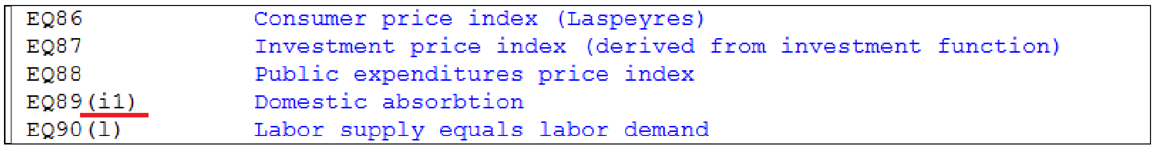
Figure 65 : Declaration of equation 89 in example 10



```
EQ86                Consumer price index (Laspeyres)
EQ87                Investment price index (derived from investment function)
EQ88                Public expenditures price index
EQ89(i)             Domestic absorbtion
EQ90(l)             Labor supply equals labor demand
```

Becomes:

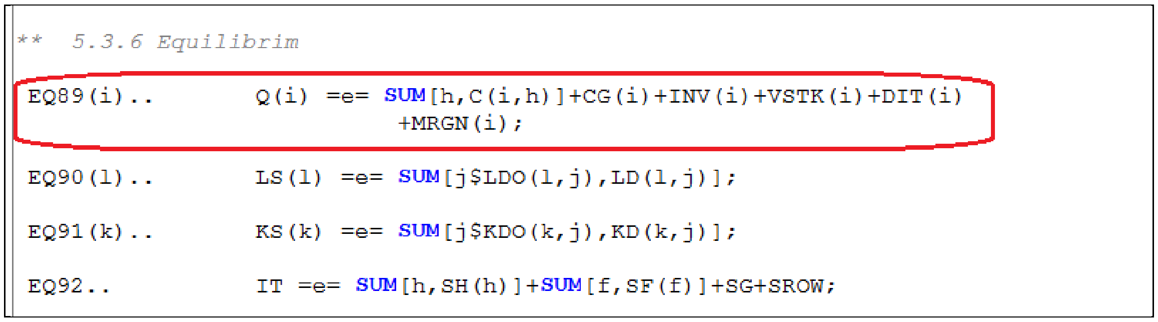
Figure 66 : Declaration of equation 89 after correction in example 10



```
EQ86                Consumer price index (Laspeyres)
EQ87                Investment price index (derived from investment function)
EQ88                Public expenditures price index
EQ89(i1)            Domestic absorbtion
EQ90(l)             Labor supply equals labor demand
```

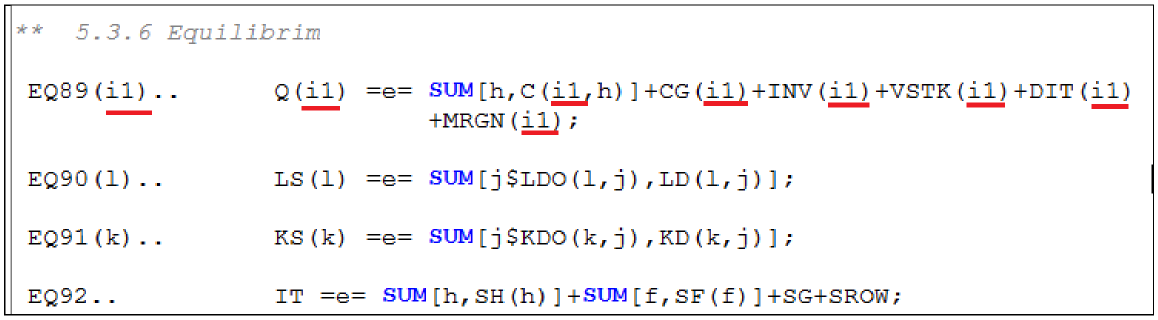
Now we also need to change the set in the definition of equation 89:

Figure 67 : Equation 89 in example 10


```
**   5.3.6 Equilibrim
EQ89(i)..        Q(i)  =e= SUM[h,C(i,h)]+CG(i)+INV(i)+VSTK(i)+DIT(i)
                           +MRGN(i);

EQ90(l)..        LS(l)  =e= SUM[j$LDO(l,j),LD(l,j)];

EQ91(k)..        KS(k)  =e= SUM[j$KDO(k,j),KD(k,j)];

EQ92..           IT  =e= SUM[h,SH(h)]+SUM[f,SF(f)]+SG+SROW;
```

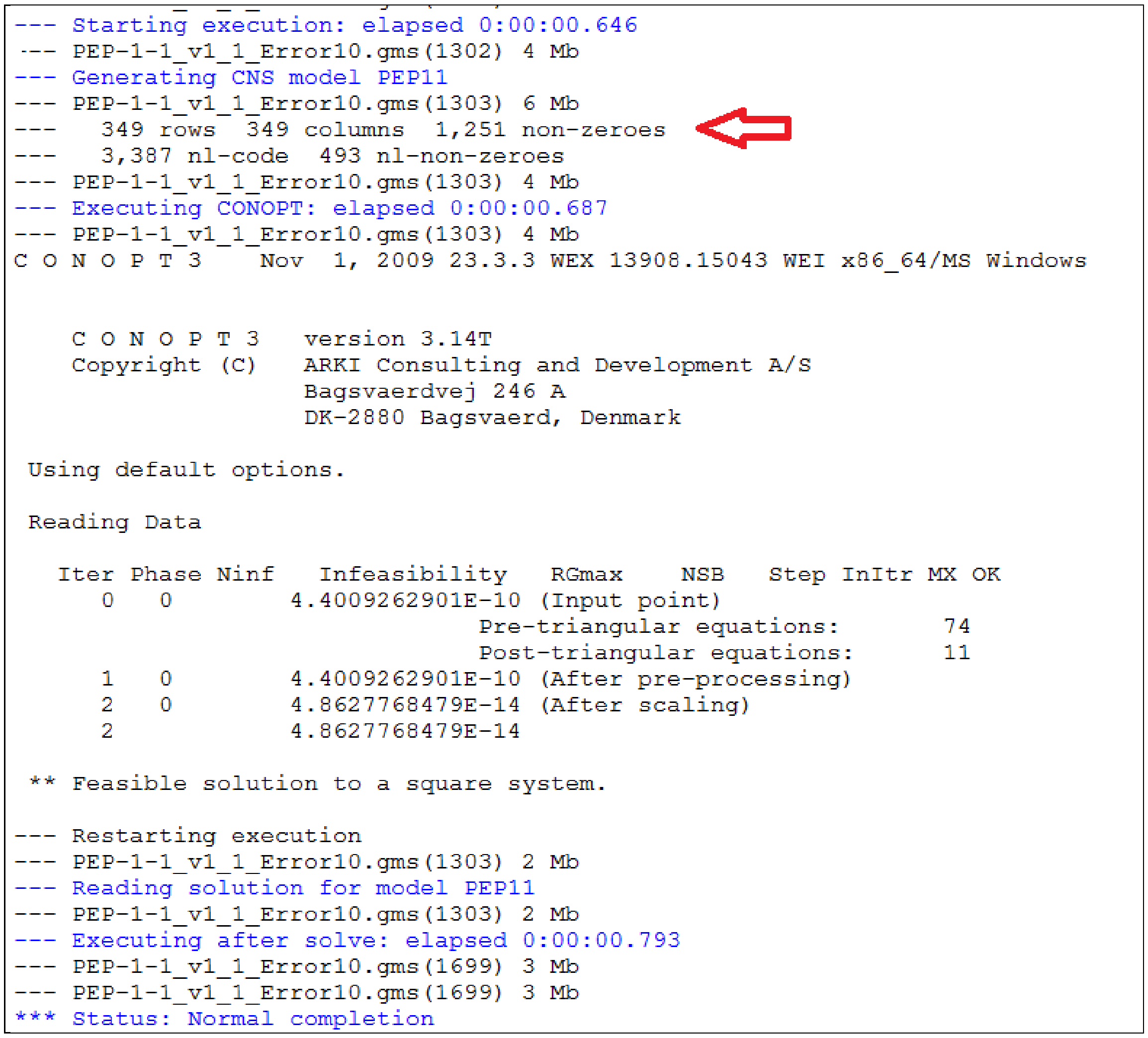
By:

Figure 68 : Equation 89 after correction in example 10


```
**   5.3.6 Equilibrim
EQ89(i1)..        Q(i1)  =e= SUM[h,C(i1,h)]+CG(i1)+INV(i1)+VSTK(i1)+DIT(i1)
                             +MRGN(i1);

EQ90(l)..         LS(l)  =e= SUM[j$LDO(l,j),LD(l,j)];

EQ91(k)..         KS(k)  =e= SUM[j$KDO(k,j),KD(k,j)];

EQ92..            IT  =e= SUM[h,SH(h)]+SUM[f,SF(f)]+SG+SROW;
```

Then, we can run the model again and check the solution:

Figure 69 : Process window after correction of example 10



```
--- Starting execution: elapsed 0:00:00.646
--- PEP-1-1_v1_1_Error10.gms(1302) 4 Mb
--- Generating CNS model PEP11
--- PEP-1-1_v1_1_Error10.gms(1303) 6 Mb
---   349 rows  349 columns  1,251 non-zeroes
---   3,387 nl-code  493 nl-non-zeroes
--- PEP-1-1_v1_1_Error10.gms(1303) 4 Mb
--- Executing CONOPT: elapsed 0:00:00.687
--- PEP-1-1_v1_1_Error10.gms(1303) 4 Mb
C O N O P T 3   Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                Pre-triangular equations:       74
                                Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error10.gms(1303) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error10.gms(1303) 2 Mb
--- Executing after solve: elapsed 0:00:00.793
--- PEP-1-1_v1_1_Error10.gms(1699) 3 Mb
--- PEP-1-1_v1_1_Error10.gms(1699) 3 Mb
*** Status: Normal completion
```

3. Calibration errors

The third type of error~~s~~ occurs when the model does not reproduce the base year values. We know that without a shock, GAMS replaces the parameters and variables by their computed values, and this should lead to reproduce the initial SAM. In the examples below, you will find some examples where the base year is not reproduced, and you will see that there are many ways to make calibration errors..

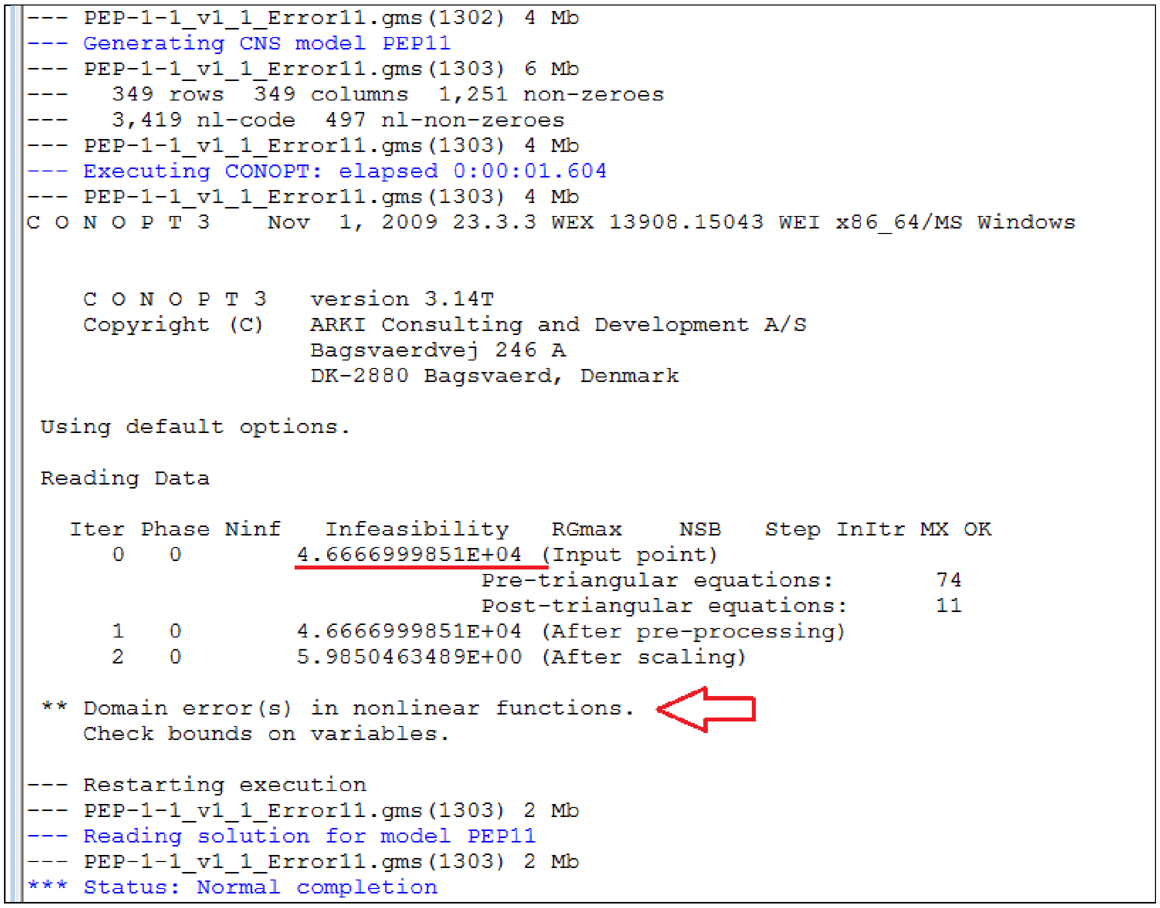
This type of error happens once the compilation errors (red lines in the process window) and execution errors have been corrected. In other words, though the model is running, we have to pay attention to the input point in order to check if the model replicates the base year.

We chose to initialize the variables at the benchmark values. This helps the model to solve but it is as well a diagnostic tool.

Example 11:

Example 11 is presented as follows:

Figure 70 : Process window of example 11



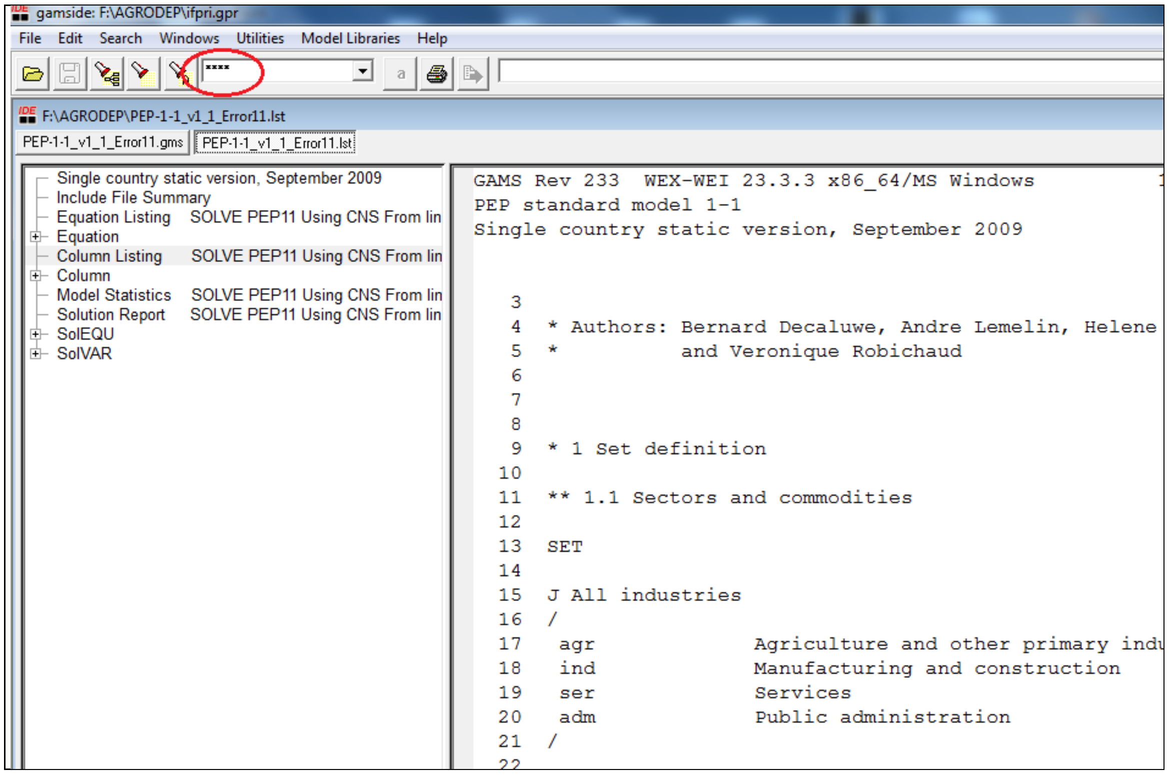
```
--- PEP-1-1_v1_1_Error11.gms(1302) 4 Mb
--- Generating CNS model PEP11
--- PEP-1-1_v1_1_Error11.gms(1303) 6 Mb
---   349 rows  349 columns  1,251 non-zeroes
---   3,419 nl-code  497 nl-non-zeroes
--- PEP-1-1_v1_1_Error11.gms(1303) 4 Mb
--- Executing CONOPT: elapsed 0:00:01.604
--- PEP-1-1_v1_1_Error11.gms(1303) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.6666999851E+04 (Input point)
                                Pre-triangular equations:       74
                                Post-triangular equations:      11
      1   0        4.6666999851E+04 (After pre-processing)
      2   0        5.9850463489E+00 (After scaling)

 ** Domain error(s) in nonlinear functions.
    Check bounds on variables.

--- Restarting execution
--- PEP-1-1_v1_1_Error11.gms(1303) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error11.gms(1303) 2 Mb
*** Status: Normal completion
```

From the process window above, we first want to check the input point, which indicates the magnitude of the biggest difference between the left and the right side of each equation. If the model was perfectly calibrated, the difference should be zero for each equation. If this is not the case, *and we have not introduced any shock*, then there is definitely a calibration error. In this example we can see that the infeasibility is quite big.

The next step is to find from which equation the problem comes from. To do this, we refer to the listing file, and write four asterisks or the word *INFES*[13] in the search window as shown in the next figure:

[13] *INFES* is the abbreviation of infeasible. Looking for an *INFES* in the listing file is a way to solve the calibration errors

Figure 71 : Listing file of example 11

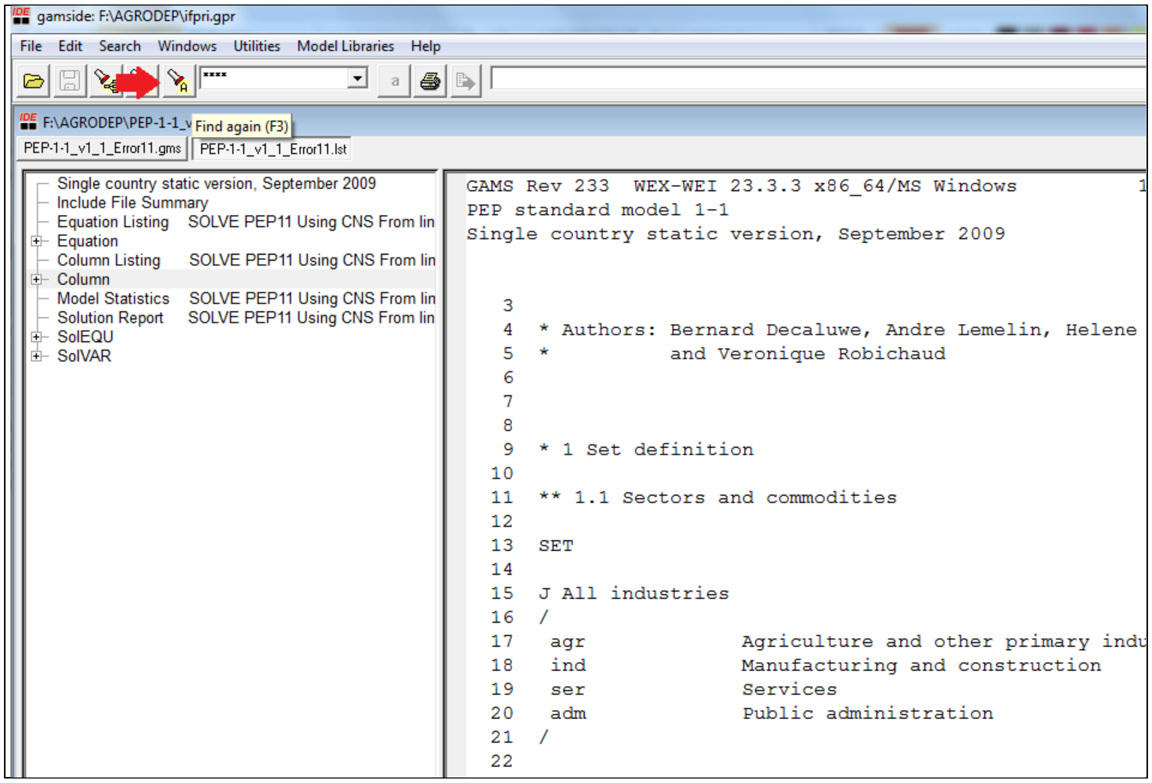

gamside: F:\AGRODEP\ifpri.gpr
File Edit Search Windows Utilities Model Libraries Help
****
F:\AGRODEP\PEP-1-1_v1_1_Error11.lst
PEP-1-1_v1_1_Error11.gms
PEP-1-1_v1_1_Error11.lst

```
Single country static version, September 2009
Include File Summary
Equation Listing    SOLVE PEP11 Using CNS From lin
Equation
Column Listing      SOLVE PEP11 Using CNS From lin
Column
Model Statistics    SOLVE PEP11 Using CNS From lin
Solution Report     SOLVE PEP11 Using CNS From lin
SolEQU
SolVAR
```

```
GAMS Rev 233  WEX-WEI 23.3.3 x86_64/MS Windows
PEP standard model 1-1
Single country static version, September 2009


* Authors: Bernard Decaluwe, Andre Lemelin, Helene
*          and Veronique Robichaud



* 1 Set definition

** 1.1 Sectors and commodities

SET

J All industries
/
 agr             Agriculture and other primary indu
 ind             Manufacturing and construction
 ser             Services
 adm             Public administration
/
```

The four asterisks indicate all the important components of the output file. By clicking on the flashlight located just to the left of the search window, you will find the “****” in the listing file.

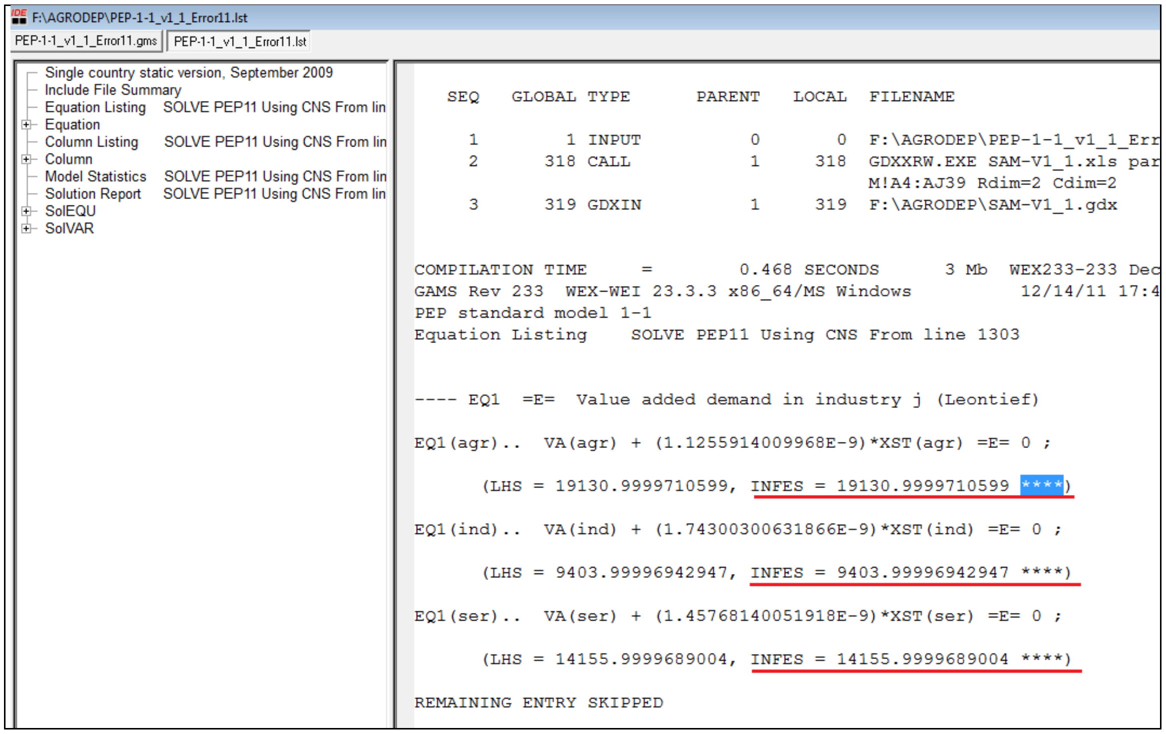
Figure 72 : Listing file of example 11 (2)


gamside: F:\AGRODEP\ifpri.gpr
File Edit Search Windows Utilities Model Libraries Help
F:\AGRODEP\PEP-1-1_v
Find again (F3)
PEP-1-1_v1_1_Error11.gms
PEP-1-1_v1_1_Error11.lst
Single country static version, September 2009
Include File Summary
Equation Listing SOLVE PEP11 Using CNS From lin
Equation
Column Listing SOLVE PEP11 Using CNS From lin
Column
Model Statistics SOLVE PEP11 Using CNS From lin
Solution Report SOLVE PEP11 Using CNS From lin
SolEQU
SolVAR

```
GAMS Rev 233  WEX-WEI 23.3.3 x86_64/MS Windows                1
PEP standard model 1-1
Single country static version, September 2009



   3
   4  * Authors: Bernard Decaluwe, Andre Lemelin, Helene
   5  *          and Veronique Robichaud
   6
   7
   8
   9  * 1 Set definition
  10
  11  ** 1.1 Sectors and commodities
  12
  13  SET
  14
  15  J All industries
  16  /
  17   agr              Agriculture and other primary indu
  18   ind              Manufacturing and construction
  19   ser              Services
  20   adm              Public administration
  21  /
  22
```

We click on the flashlight and obtain figure 73:

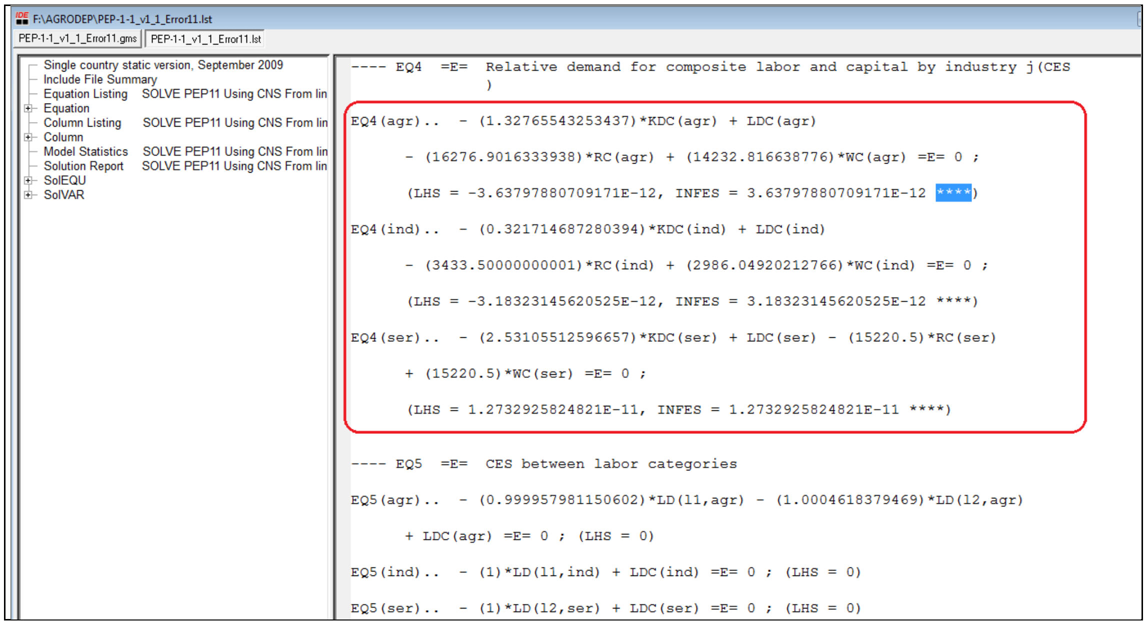
Figure 73 : Looking for an Infes in the listing file


F:\AGRODEP\PEP-1-1_v1_1_Error11.lst
PEP-1-1_v1_1_Error11.gms
PEP-1-1_v1_1_Error11.lst
Single country static version, September 2009
Include File Summary
Equation Listing SOLVE PEP11 Using CNS From lin
Equation
Column Listing SOLVE PEP11 Using CNS From lin
Column
Model Statistics SOLVE PEP11 Using CNS From lin
Solution Report SOLVE PEP11 Using CNS From lin
SolEQU
SolVAR

```
   SEQ   GLOBAL TYPE      PARENT   LOCAL  FILENAME

     1        1 INPUT          0       0  F:\AGRODEP\PEP-1-1_v1_1_Err
     2      318 CALL           1     318  GDXXRW.EXE SAM-V1_1.xls par
                                          M!A4:AJ39 Rdim=2 Cdim=2
     3      319 GDXIN          1     319  F:\AGRODEP\SAM-V1_1.gdx


COMPILATION TIME     =        0.468 SECONDS      3 Mb  WEX233-233 Dec
GAMS Rev 233  WEX-WEI 23.3.3 x86_64/MS Windows          12/14/11 17:4
PEP standard model 1-1
Equation Listing    SOLVE PEP11 Using CNS From line 1303


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

EQ1(agr)..  VA(agr) + (1.1255914009968E-9)*XST(agr) =E= 0 ;

      (LHS = 19130.9999710599, INFES = 19130.9999710599 ****)

EQ1(ind)..  VA(ind) + (1.74300300631866E-9)*XST(ind) =E= 0 ;

      (LHS = 9403.99996942947, INFES = 9403.99996942947 ****)

EQ1(ser)..  VA(ser) + (1.45768140051918E-9)*XST(ser) =E= 0 ;

      (LHS = 14155.9999689004, INFES = 14155.9999689004 ****)

REMAINING ENTRY SKIPPED
```

In the listing file shown in figure 73, we find an *infes* for equation 1, in each sector. We then check to see if there are any other *infes* by clicking on the flashlight again. We obtain the following figure:

Figure 74 : Looking for infes in example 11


F:\AGRODEP\PEP-1-1_v1_1_Error11.lst

PEP-1-1_v1_1_Error11.gms | PEP-1-1_v1_1_Error11.lst

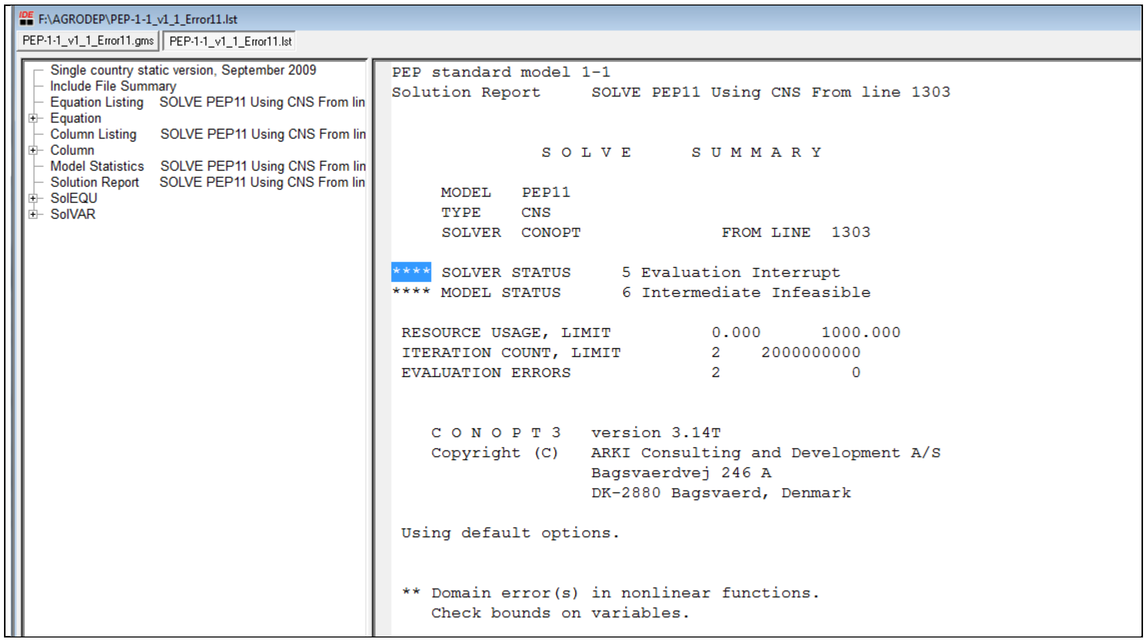
```
Single country static version, September 2009
Include File Summary
Equation Listing   SOLVE PEP11 Using CNS From lin
Equation
Column Listing     SOLVE PEP11 Using CNS From lin
Column
Model Statistics   SOLVE PEP11 Using CNS From lin
Solution Report    SOLVE PEP11 Using CNS From lin
SolEQU
SolVAR
```

```
---- EQ4  =E=  Relative demand for composite labor and capital by industry j(CES
     )

EQ4(agr)..  - (1.32765543253437)*KDC(agr) + LDC(agr)

      - (16276.9016333938)*RC(agr) + (14232.816638776)*WC(agr) =E= 0 ;

      (LHS = -3.63797880709171E-12, INFES = 3.63797880709171E-12 ****)

EQ4(ind)..  - (0.321714687280394)*KDC(ind) + LDC(ind)

      - (3433.50000000001)*RC(ind) + (2986.04920212766)*WC(ind) =E= 0 ;

      (LHS = -3.18323145620525E-12, INFES = 3.18323145620525E-12 ****)

EQ4(ser)..  - (2.53105512596657)*KDC(ser) + LDC(ser) - (15220.5)*RC(ser)

      + (15220.5)*WC(ser) =E= 0 ;

      (LHS = 1.2732925824821E-11, INFES = 1.2732925824821E-11 ****)

---- EQ5  =E=  CES between labor categories

EQ5(agr)..  - (0.999957981150602)*LD(l1,agr) - (1.0004618379469)*LD(l2,agr)

      + LDC(agr) =E= 0 ; (LHS = 0)

EQ5(ind)..  - (1)*LD(l1,ind) + LDC(ind) =E= 0 ; (LHS = 0)

EQ5(ser)..  - (1)*LD(l2,ser) + LDC(ser) =E= 0 ; (LHS = 0)
```

The *infes* that affects equation 4 is actually very small. If we look at the magnitude, we find that it is a number to the power of -12, or -11. In this case, we will not take into account the *infes* for which the magnitude is that low. Note that the balance of your SAM may influence your rounding here. We continue on in the listing file, looking for another *infes*. We will not report anymore *infes* that are greater than E-8.

Finally, we cannot find any significant *infes* and we obtain the following figure:

Figure 75 : Listing file of example 11



F:\AGRODEP\PEP-1-1_v1_1_Error11.lst
PEP-1-1_v1_1_Error11.gms
PEP-1-1_v1_1_Error11.lst
Single country static version, September 2009
Include File Summary
Equation Listing SOLVE PEP11 Using CNS From lin
Equation
Column Listing SOLVE PEP11 Using CNS From lin
Column
Model Statistics SOLVE PEP11 Using CNS From lin
Solution Report SOLVE PEP11 Using CNS From lin
SolEQU
SolVAR

```
PEP standard model 1-1
Solution Report     SOLVE PEP11 Using CNS From line 1303


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

     MODEL   PEP11
     TYPE    CNS
     SOLVER  CONOPT              FROM LINE  1303

**** SOLVER STATUS     5 Evaluation Interrupt
**** MODEL STATUS      6 Intermediate Infeasible

 RESOURCE USAGE, LIMIT          0.000      1000.000
 ITERATION COUNT, LIMIT         2     2000000000
 EVALUATION ERRORS              2              0


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.


 ** Domain error(s) in nonlinear functions.
    Check bounds on variables.
```

So for this example, we have just one *infes* that only affects equation 1, for all the reported sectors. As there is only one equation affected, we would proceed as follows:

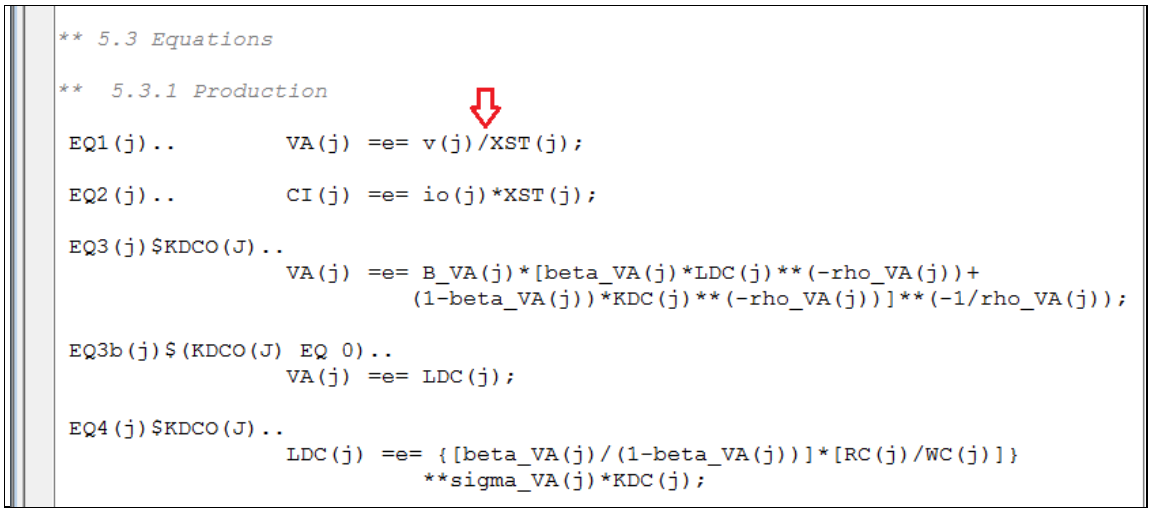
- check how the equation is written in the GAMS code

- check how the parameters that were entered in this equation are calibrated.

- check the initial data of the variables appearing in the equation

Let's go and see how equation 1 is written in the GAMS code.

Figure 76 : Equation 1 in example 11

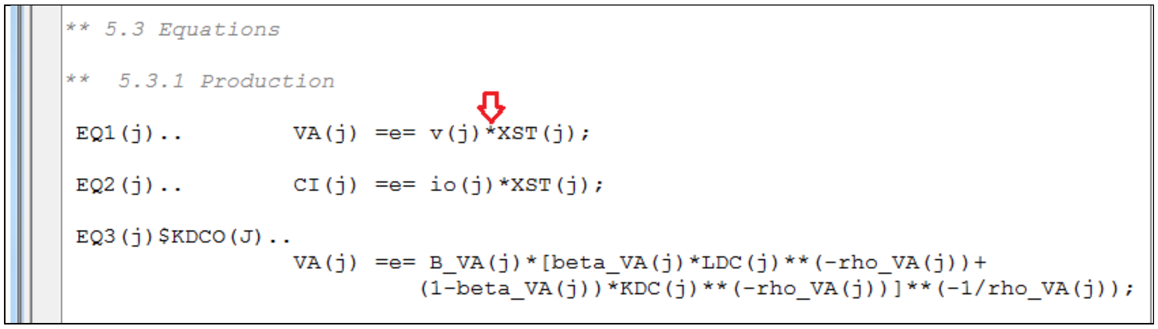


```
** 5.3 Equations

**  5.3.1 Production

 EQ1(j)..         VA(j)  =e= v(j)/XST(j);

 EQ2(j)..         CI(j)  =e= io(j)*XST(j);

 EQ3(j)$KDCO(J)..
                  VA(j)  =e= B_VA(j)*[beta_VA(j)*LDC(j)**(-rho_VA(j))+
                         (1-beta_VA(j))*KDC(j)**(-rho_VA(j))]**(-1/rho_VA(j));

 EQ3b(j)$(KDCO(J) EQ 0)..
                  VA(j)  =e= LDC(j);

 EQ4(j)$KDCO(J)..
                  LDC(j)  =e= {[beta_VA(j)/(1-beta_VA(j))]*[RC(j)/WC(j)]}
                          **sigma_VA(j)*KDC(j);
```

Equation 1 defines the value added of activity *j*. It is a share of the total production of the sector *j*, as assumed in the PEP1-1 model. However, in the GAMS code (figure 76) we can see that the parameter *v(j)* is divided by *XST(j)*, while in fact, it should be multiplied by *XST(j)*.

Thus, to correct the calibration error, we need to correct Equation 1 by changing "/"to "*", as shown in the following figure.

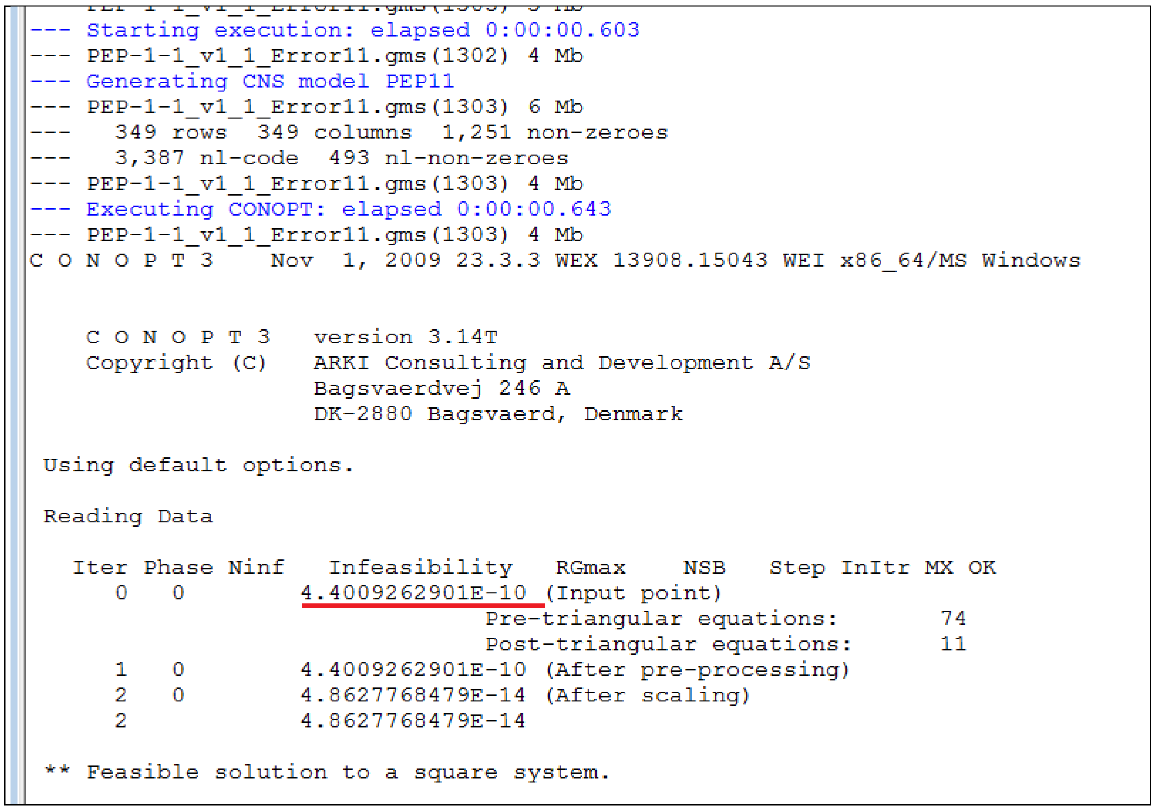
Figure 77 : Correction of example 11



```
** 5.3 Equations

**  5.3.1 Production

 EQ1(j)..          VA(j) =e= v(j)*XST(j);

 EQ2(j)..          CI(j) =e= io(j)*XST(j);

 EQ3(j)$KDCO(J)..
                   VA(j) =e= B_VA(j)*[beta_VA(j)*LDC(j)**(-rho_VA(j))+
                             (1-beta_VA(j))*KDC(j)**(-rho_VA(j))]**(-1/rho_VA(j));
```

Now, we can re-run the model and check if it reproduces the base year.

Figure 78 : Process window of example 11

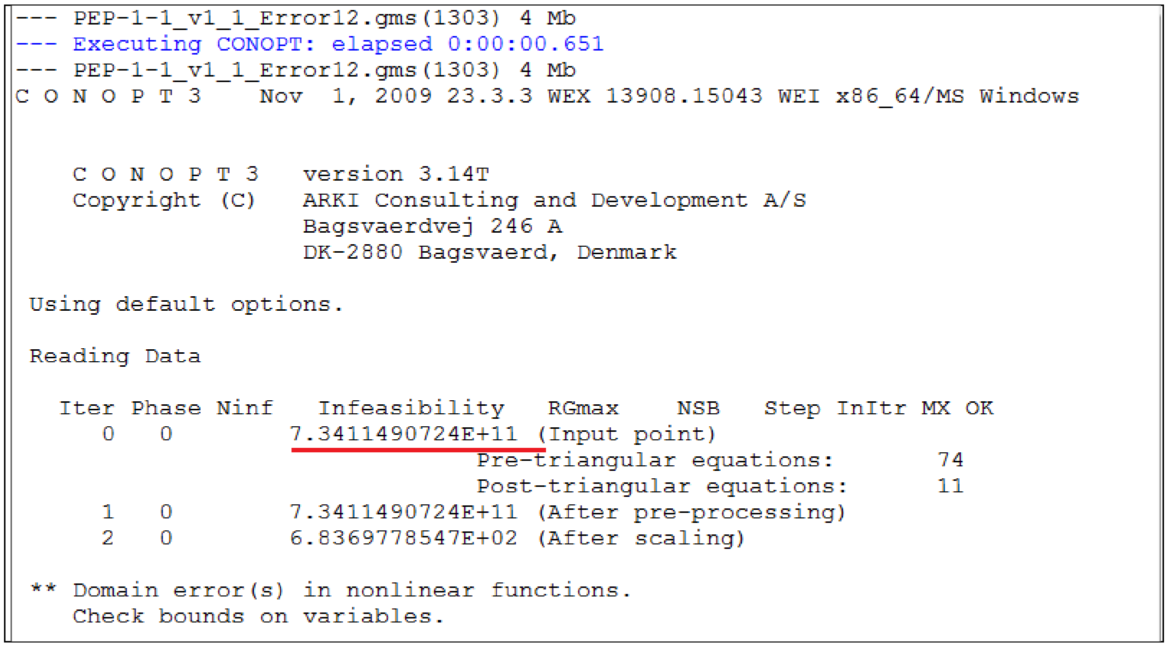


```
--- Starting execution: elapsed 0:00:00.603
--- PEP-1-1_v1_1_Error11.gms(1302) 4 Mb
--- Generating CNS model PEP11
--- PEP-1-1_v1_1_Error11.gms(1303) 6 Mb
---   349 rows  349 columns  1,251 non-zeroes
---   3,387 nl-code  493 nl-non-zeroes
--- PEP-1-1_v1_1_Error11.gms(1303) 4 Mb
--- Executing CONOPT: elapsed 0:00:00.643
--- PEP-1-1_v1_1_Error11.gms(1303) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                Pre-triangular equations:       74
                                Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.
```

Example 12:

Example 12 is presented in the following figure:

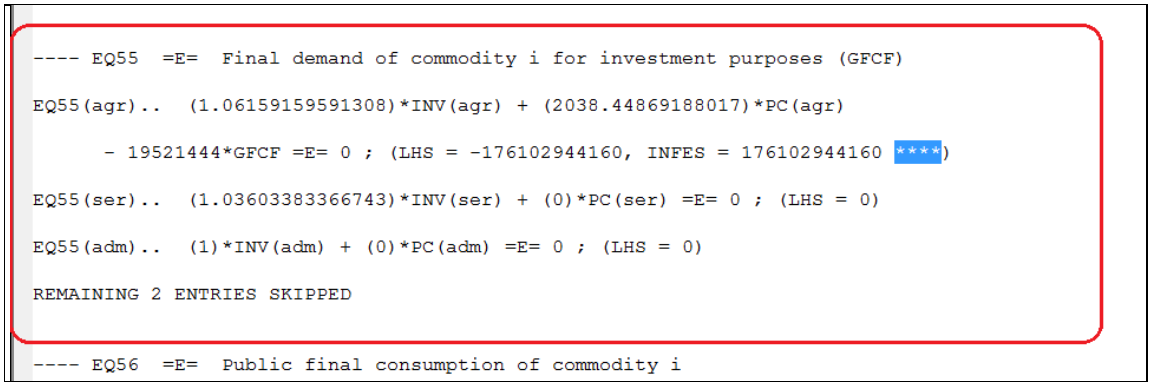
Figure 79 : Process window of example 12



```
--- PEP-1-1_v1_1_Error12.gms(1303) 4 Mb
--- Executing CONOPT: elapsed 0:00:00.651
--- PEP-1-1_v1_1_Error12.gms(1303) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        7.3411490724E+11 (Input point)
                                 Pre-triangular equations:       74
                                 Post-triangular equations:      11
      1   0        7.3411490724E+11 (After pre-processing)
      2   0        6.8369778547E+02 (After scaling)

 ** Domain error(s) in nonlinear functions.
    Check bounds on variables.
```

As underlined in the figure above, the input point is too high. The debugging procedure is the same as in the previous example. We go to the listing file, and by double-clicking on the flashlight, find the *infes*.

Figure 80 : Infes in example 11

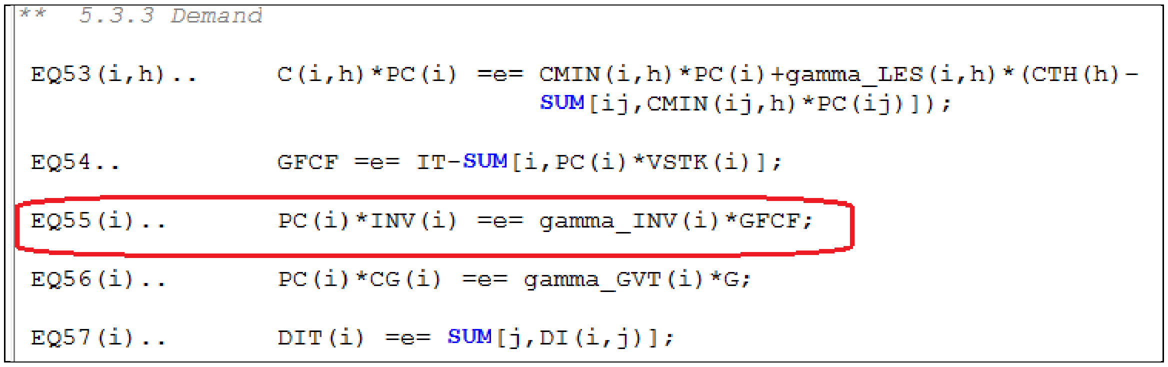


```
---- EQ55  =E=  Final demand of commodity i for investment purposes (GFCF)

EQ55(agr)..  (1.06159159591308)*INV(agr) + (2038.44869188017)*PC(agr)

     - 19521444*GFCF =E= 0 ; (LHS = -176102944160, INFES = 176102944160 ****)

EQ55(ser)..  (1.03603383366743)*INV(ser) + (0)*PC(ser) =E= 0 ; (LHS = 0)

EQ55(adm)..  (1)*INV(adm) + (0)*PC(adm) =E= 0 ; (LHS = 0)

REMAINING 2 ENTRIES SKIPPED

---- EQ56  =E=  Public final consumption of commodity i
```

We find that there is only one *infes* in this file[14].

We proceed by applying the same methods used in the previous example: we first check if the equation is correctly specified. If it is correctly specified, we check how the parameter used in the equation is calibrated.

Figure 81 : Writing of equation 55 in example 12

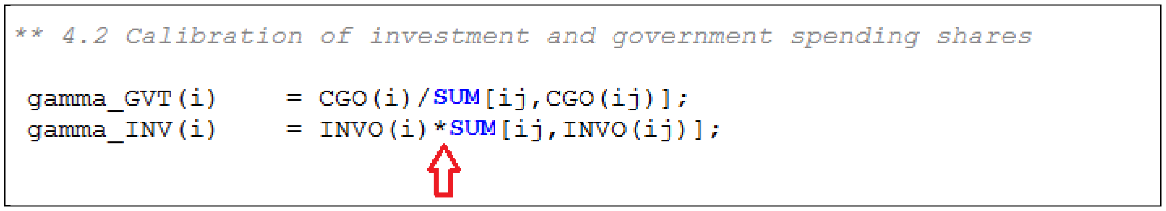


```
** 5.3.3 Demand

 EQ53(i,h)..      C(i,h)*PC(i)  =e= CMIN(i,h)*PC(i)+gamma_LES(i,h)*(CTH(h)-
                                        SUM[ij,CMIN(ij,h)*PC(ij)]);

 EQ54..           GFCF =e= IT-SUM[i,PC(i)*VSTK(i)];

 EQ55(i)..        PC(i)*INV(i)  =e= gamma_INV(i)*GFCF;

 EQ56(i)..        PC(i)*CG(i)  =e= gamma_GVT(i)*G;

 EQ57(i)..        DIT(i)  =e= SUM[j,DI(i,j)];
```

From figure 81, we check that equation 55 is correctly specified. *GFCF* expenditure is distributed among commodities in fixed shares (*gamma_INV(i)*).

As the equation is correctly specified, we continue by checking on the parameter *gamma_INV(i)*. The calibration of *gamma_INV(i)* is at the beginning of the GAMS code of PEP 1-1. Figure 82 reproduces the calibration for this parameter.

Figure 82 : Calibration of the parameter gamma_INV(i) in example 12

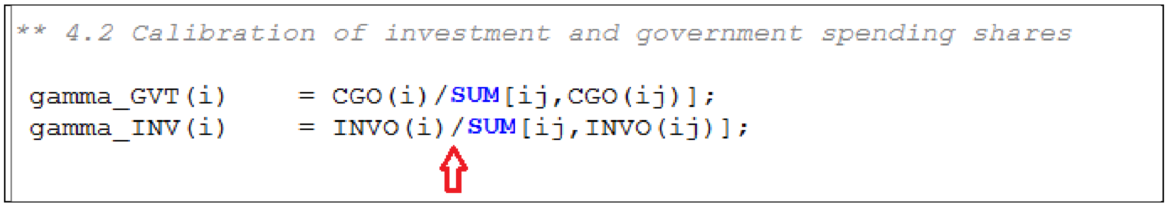


```
** 4.2 Calibration of investment and government spending shares

 gamma_GVT(i)      = CGO(i)/SUM[ij,CGO(ij)];
 gamma_INV(i)      = INVO(i)*SUM[ij,INVO(ij)];
```

The parameter *Gamma_INV(i)* is the share of commodity *i* in total investment expenditure. However, based on the equation in figure 82, the computation of *gamma_inv(i)* is incorrect. For shares, the parameter should be written as follows:

[14] We remind the reader that *infes* smaller than E–8 are not relevant *infes*. To avoid inflating the size of this document, we will henceforth report only the relevant *infes*.

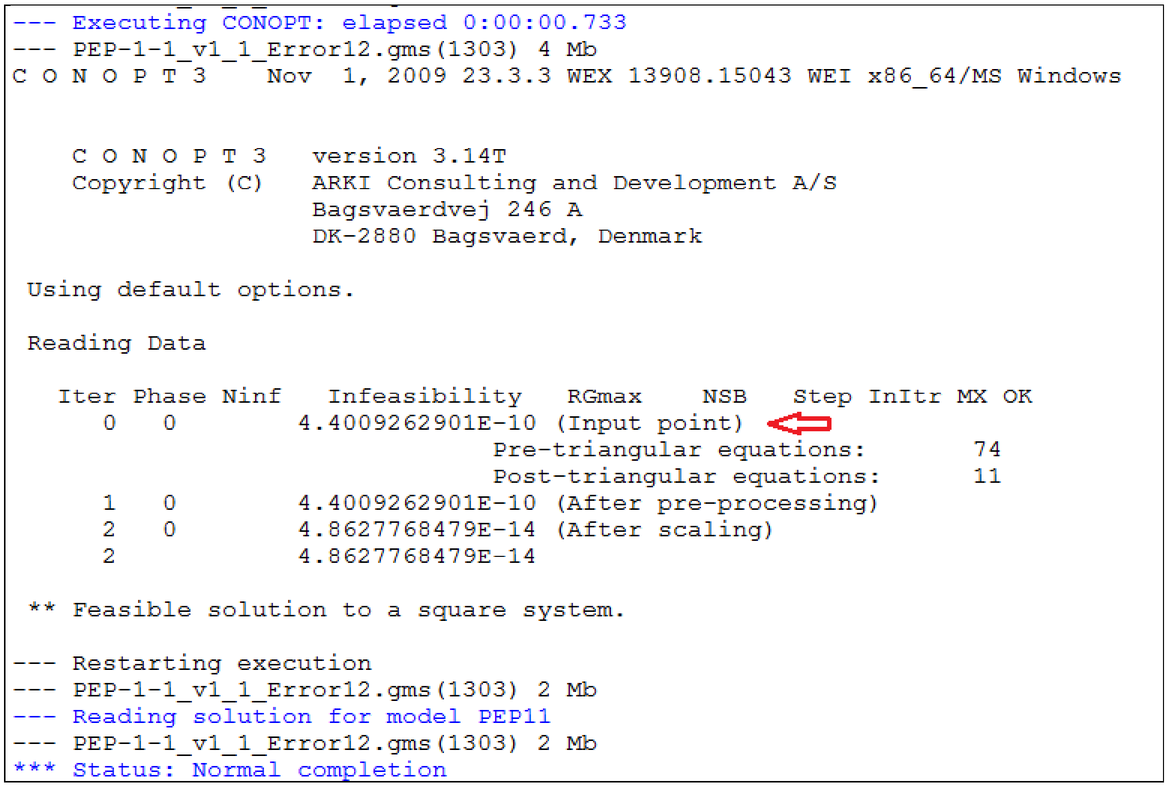
Figure 83 : Correction of the calibration of parameter gamma_INV(i) :



```
** 4.2 Calibration of investment and government spending shares

 gamma_GVT(i)      = CGO(i)/SUM[ij,CGO(ij)];
 gamma_INV(i)      = INVO(i)/SUM[ij,INVO(ij)];
```

Once we correct the symbol, we run the model again and check the input point.

Figure 84 : Process window of example 12 after correction

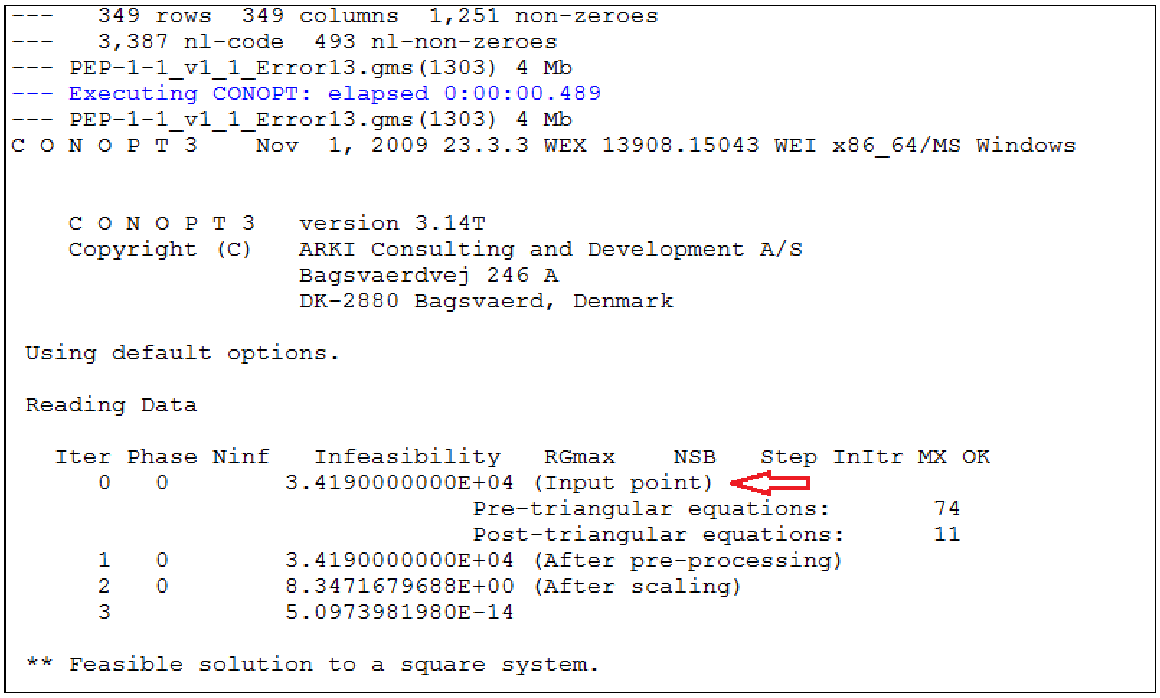


```
--- Executing CONOPT: elapsed 0:00:00.733
--- PEP-1-1_v1_1_Error12.gms(1303) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                 Pre-triangular equations:       74
                                 Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error12.gms(1303) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error12.gms(1303) 2 Mb
*** Status: Normal completion
```

Example 13:

The process window of Example 13 is reproduced below:

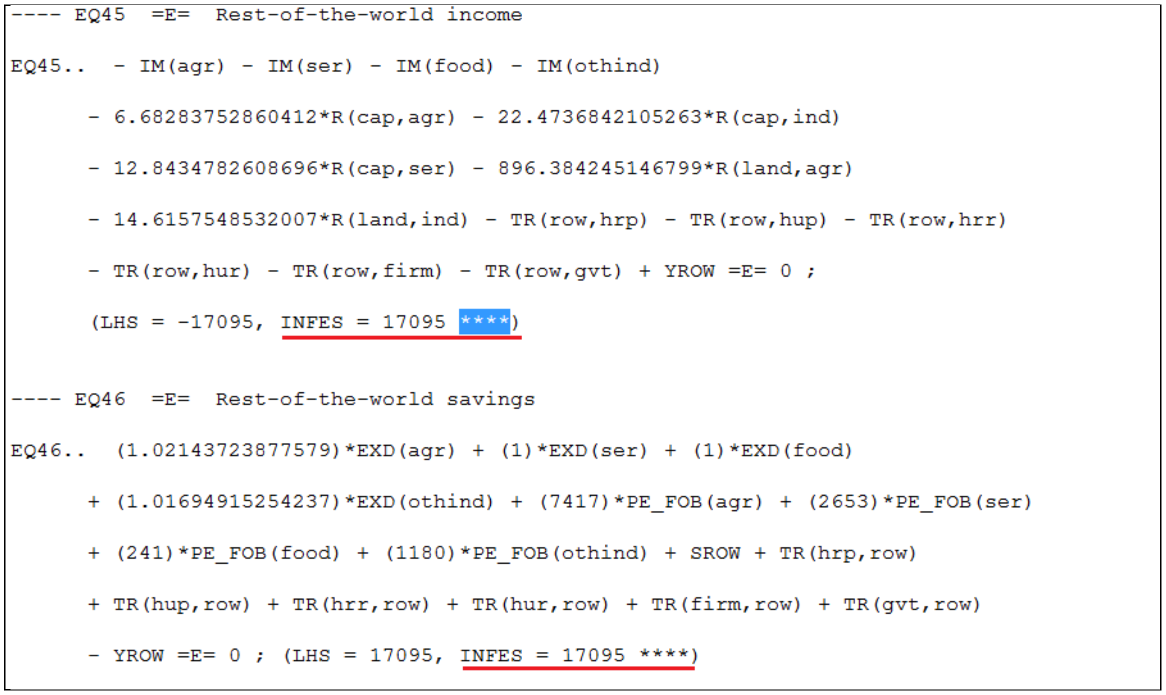
Figure 85 : Process window of example 13



```
---   349 rows  349 columns  1,251 non-zeroes
---   3,387 nl-code  493 nl-non-zeroes
--- PEP-1-1_v1_1_Error13.gms(1303) 4 Mb
--- Executing CONOPT: elapsed 0:00:00.489
--- PEP-1-1_v1_1_Error13.gms(1303) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        3.4190000000E+04 (Input point)
                                 Pre-triangular equations:       74
                                 Post-triangular equations:      11
      1   0        3.4190000000E+04 (After pre-processing)
      2   0        8.3471679688E+00 (After scaling)
      3            5.0973981980E-14

 ** Feasible solution to a square system.
```

The input point for this example is quite high. We have to go into the listing file and look for *infes*, by searching for four asterisks as in Figure 72. We find the following:

Figure 86 : Listing file for example 13



```
---- EQ45  =E=  Rest-of-the-world income

EQ45..  - IM(agr) - IM(ser) - IM(food) - IM(othind)

      - 6.68283752860412*R(cap,agr) - 22.4736842105263*R(cap,ind)

      - 12.8434782608696*R(cap,ser) - 896.384245146799*R(land,agr)

      - 14.6157548532007*R(land,ind) - TR(row,hrp) - TR(row,hup) - TR(row,hrr)

      - TR(row,hur) - TR(row,firm) - TR(row,gvt) + YROW =E= 0 ;

      (LHS = -17095, INFES = 17095 ****)

---- EQ46  =E=  Rest-of-the-world savings

EQ46..  (1.02143723877579)*EXD(agr) + (1)*EXD(ser) + (1)*EXD(food)

      + (1.01694915254237)*EXD(othind) + (7417)*PE_FOB(agr) + (2653)*PE_FOB(ser)

      + (241)*PE_FOB(food) + (1180)*PE_FOB(othind) + SROW + TR(hrp,row)

      + TR(hup,row) + TR(hrr,row) + TR(hur,row) + TR(firm,row) + TR(gvt,row)

      - YROW =E= 0 ; (LHS = 17095, INFES = 17095 ****)
```

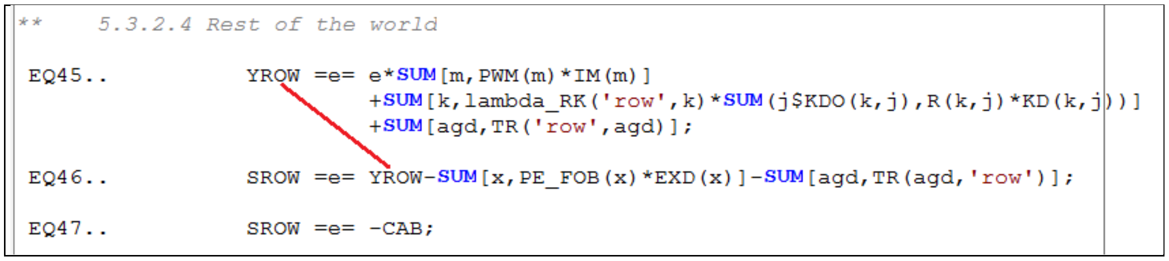
We find two *infes* in the listing file, at equations 45 and 46.

Firstly, we note that both equations are related to the agent “rest -of -the –world”. The second hint is that the value of the *infes* is exactly the same for both equations, and it is an exact value, so we can assume it comes from the SAM.

In this case we can proceed by checking how these two equations are written; notably, if there is a variable common to both of them. If there is, we will then have to look at the initialization of the variable, before going through the GAMS code for further investigations.

Let’s have a look first at equations 45 and 46:

Figure 87 : Equations 45 and 46 in example 13

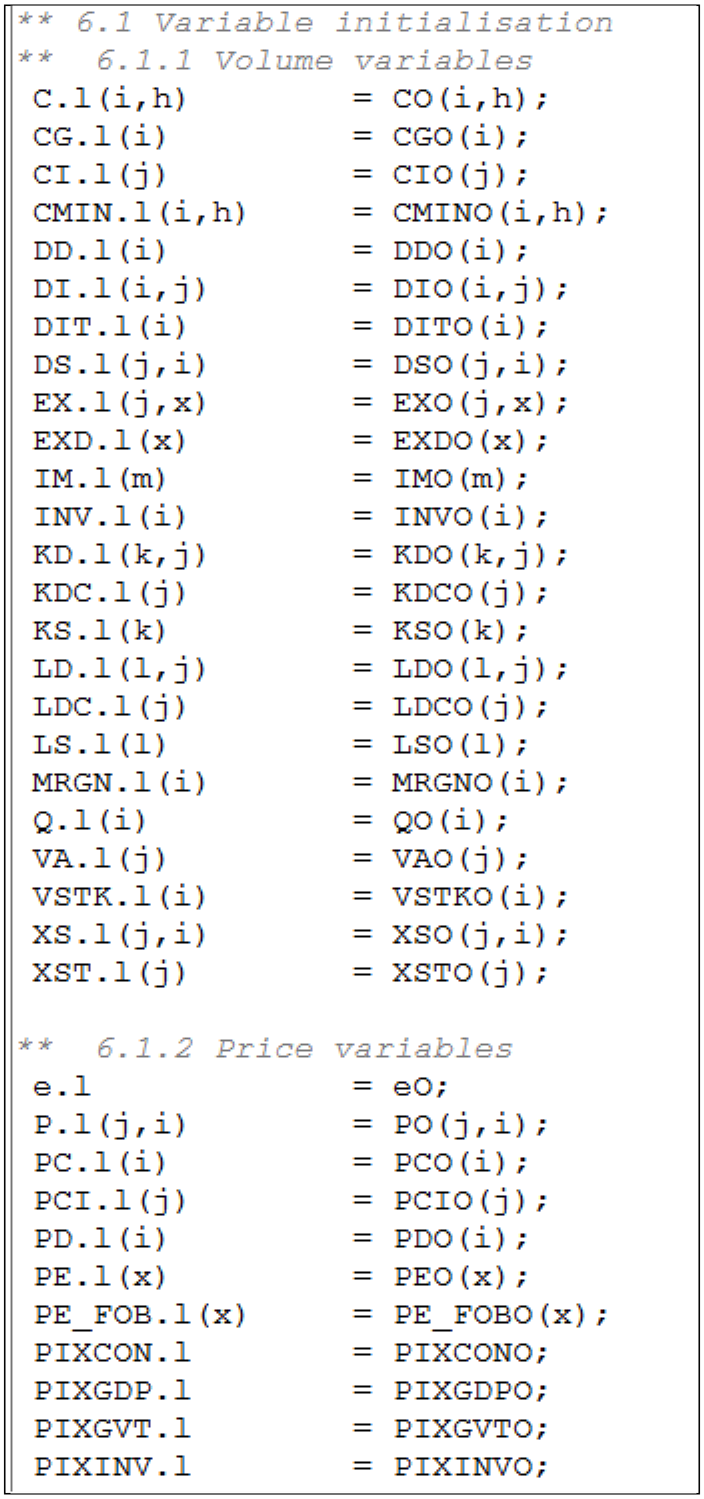


```
**    5.3.2.4 Rest of the world

 EQ45..          YROW =e= e*SUM[m,PWM(m)*IM(m)]
                          +SUM[k,lambda_RK('row',k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))]
                          +SUM[agd,TR('row',agd)];

 EQ46..          SROW =e= YROW-SUM[x,PE_FOB(x)*EXD(x)]-SUM[agd,TR(agd,'row')];

 EQ47..          SROW =e= -CAB;
```

From the figure above, we can see that both equations are correctly specified. We notice as well that the income of the *Rest of the World* (*YROW*) is common to both.

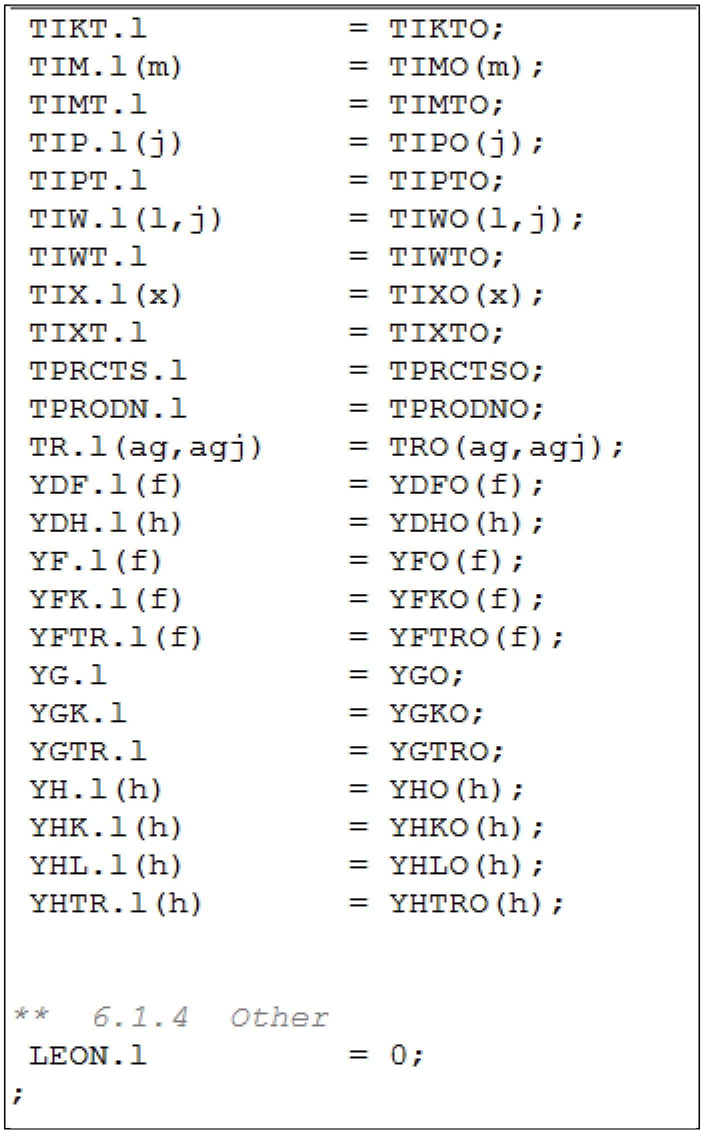
We now go and see how this variable has been initialized. In PEP 1-1, variables are initialized at the end of the GAMS file, just before the closure rules.

Figure 88 : Variable initialization in example 13



```
** 6.1 Variable initialisation
**  6.1.1 Volume variables
 C.l(i,h)           = CO(i,h);
 CG.l(i)            = CGO(i);
 CI.l(j)            = CIO(j);
 CMIN.l(i,h)        = CMINO(i,h);
 DD.l(i)            = DDO(i);
 DI.l(i,j)          = DIO(i,j);
 DIT.l(i)           = DITO(i);
 DS.l(j,i)          = DSO(j,i);
 EX.l(j,x)          = EXO(j,x);
 EXD.l(x)           = EXDO(x);
 IM.l(m)            = IMO(m);
 INV.l(i)           = INVO(i);
 KD.l(k,j)          = KDO(k,j);
 KDC.l(j)           = KDCO(j);
 KS.l(k)            = KSO(k);
 LD.l(l,j)          = LDO(l,j);
 LDC.l(j)           = LDCO(j);
 LS.l(l)            = LSO(l);
 MRGN.l(i)          = MRGNO(i);
 Q.l(i)             = QO(i);
 VA.l(j)            = VAO(j);
 VSTK.l(i)          = VSTKO(i);
 XS.l(j,i)          = XSO(j,i);
 XST.l(j)           = XSTO(j);

**  6.1.2 Price variables
 e.l                = eO;
 P.l(j,i)           = PO(j,i);
 PC.l(i)            = PCO(i);
 PCI.l(j)           = PCIO(j);
 PD.l(i)            = PDO(i);
 PE.l(x)            = PEO(x);
 PE_FOB.l(x)        = PE_FOBO(x);
 PIXCON.l           = PIXCONO;
 PIXGDP.l           = PIXGDPO;
 PIXGVT.l           = PIXGVTO;
 PIXINV.l           = PIXINVO;
```

```
PIXGDP.l          = PIXGDPO;
PIXGVT.l          = PIXGVTO;
PIXINV.l          = PIXINVO;
PL.l(i)           = PLO(i);
PM.l(m)           = PMO(m);
PP.l(j)           = PPO(j);
PT.l(j)           = PTO(j);
PVA.l(j)          = PVAO(j);
PWM.l(m)          = PWMO(m);
PWX.l(x)          = PWXO(x);
R.l(k,j)          = RO(k,j);
RC.l(j)           = RCO(j);
RK.l(k)           = RKO(k);
RTI.l(k,j)        = RTIO(k,j);
W.l(l)            = WO(l);
WC.l(j)           = WCO(j);
WTI.l(l,j)        = WTIO(l,j);

**  6.1.3 Nominal (value) variables
CAB.l             = CABO;
CTH.l(h)          = CTHO(h);
G.l               = GO;
GDP_BP.l          = GDP_BPO;
GDP_FD.l          = GDP_FDO;
GDP_IB.l          = GDP_IBO;
GDP_MP.l          = GDP_MPO;
GFCF.l            = GFCFO;
IT.l              = ITO;
SF.l(f)           = SFO(f);
SG.l              = SGO;
SH.l(h)           = SHO(h);
SROW.l            = SROWO;
TDF.l(f)          = TDFO(f);
TDFT.l            = TDFTO;
TDH.l(h)          = TDHO(h);
TDHT.l            = TDHTO;
TIC.l(i)          = TICO(i);
TICT.l            = TICTO;
TIK.l(k,j)        = TIKO(k,j);
```

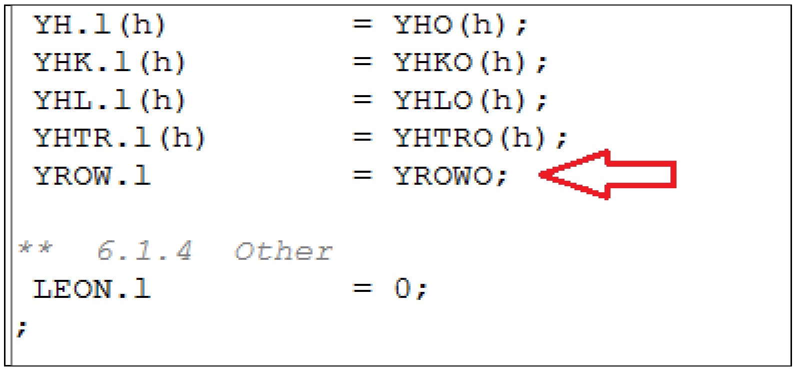


```
 TIKT.l                = TIKTO;
 TIM.l(m)              = TIMO(m);
 TIMT.l                = TIMTO;
 TIP.l(j)              = TIPO(j);
 TIPT.l                = TIPTO;
 TIW.l(l,j)            = TIWO(l,j);
 TIWT.l                = TIWTO;
 TIX.l(x)              = TIXO(x);
 TIXT.l                = TIXTO;
 TPRCTS.l              = TPRCTSO;
 TPRODN.l              = TPRODNO;
 TR.l(ag,agj)          = TRO(ag,agj);
 YDF.l(f)              = YDFO(f);
 YDH.l(h)              = YDHO(h);
 YF.l(f)               = YFO(f);
 YFK.l(f)              = YFKO(f);
 YFTR.l(f)             = YFTRO(f);
 YG.l                  = YGO;
 YGK.l                 = YGKO;
 YGTR.l                = YGTRO;
 YH.l(h)               = YHO(h);
 YHK.l(h)              = YHKO(h);
 YHL.l(h)              = YHLO(h);
 YHTR.l(h)             = YHTRO(h);


**   6.1.4   Other
 LEON.l                = 0;
;
```

The figure above reproduces the initialization in the GAMS code for example 13. We cannot find *YROW* in the list. In other words, *YROW* has not been initialized to its benchmark value, and by default, GAMS has initialized it to zero.

Thus, to correct the error, we need to initialize *YROW*.

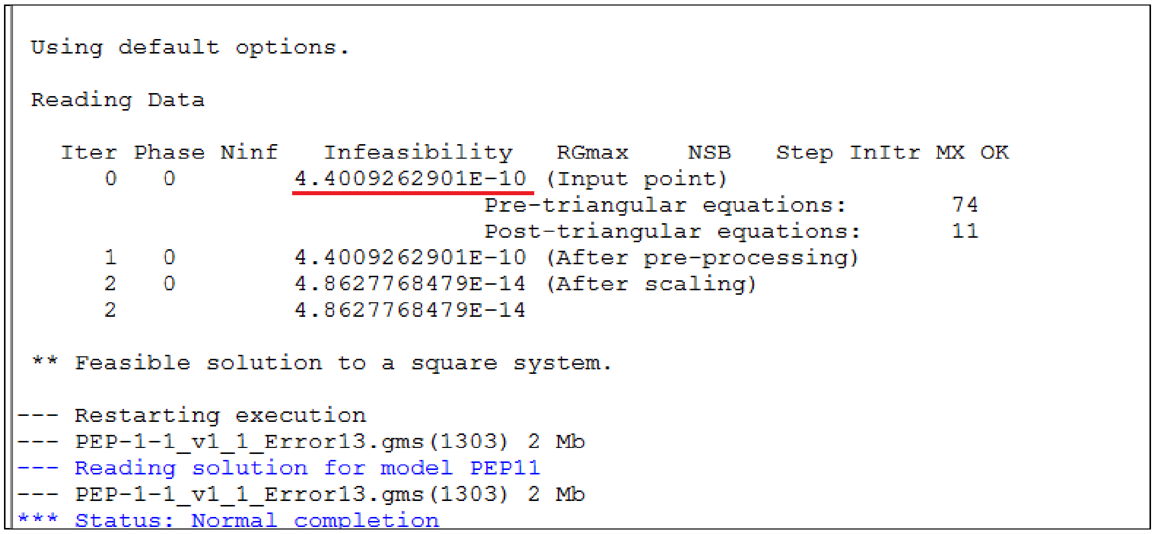
Figure 89 : Initialization of YROW in example 13



```
 YH.l(h)                = YHO(h);
 YHK.l(h)               = YHKO(h);
 YHL.l(h)               = YHLO(h);
 YHTR.l(h)              = YHTRO(h);
 YROW.l                 = YROWO;

**   6.1.4   Other
 LEON.l                 = 0;
;
```

We can now run the model again to verify that the error has been corrected.

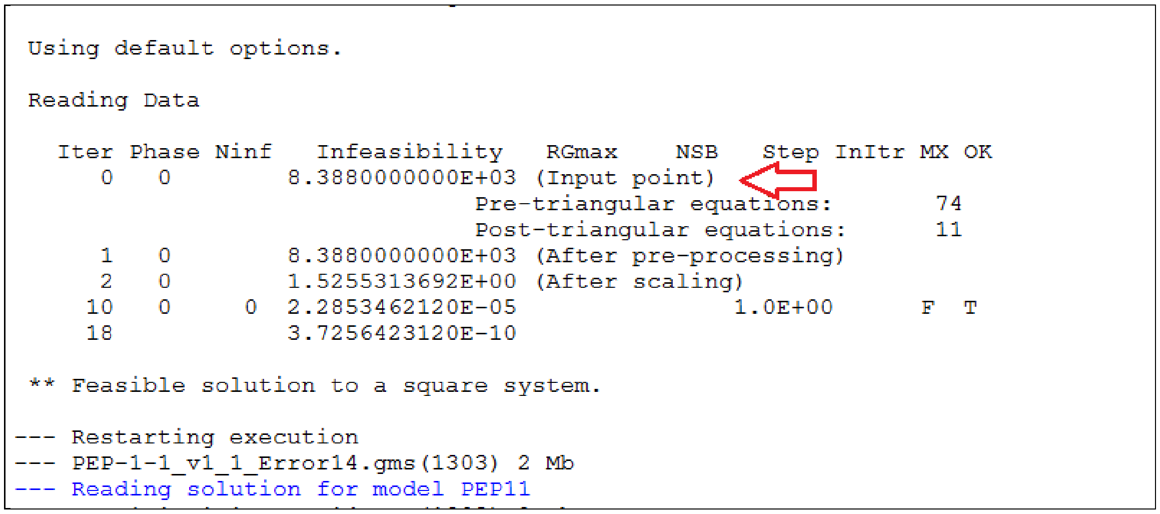
Figure 90 : Process window of example 13 after correction



```
 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                Pre-triangular equations:       74
                                Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error13.gms(1303) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error13.gms(1303) 2 Mb
*** Status: Normal completion
```

Example 14:

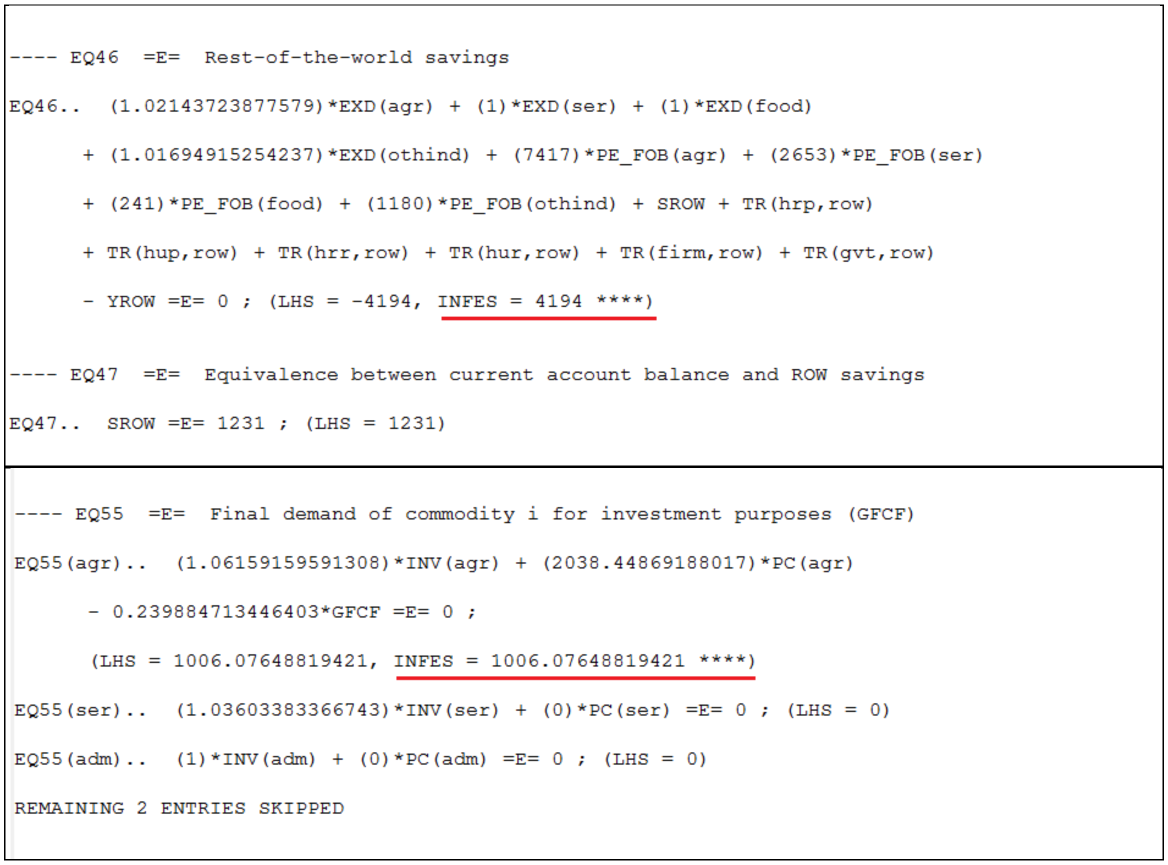
Figure 91 : Process window of example 14



```
 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        8.3880000000E+03 (Input point)
                             Pre-triangular equations:       74
                             Post-triangular equations:      11
      1   0        8.3880000000E+03 (After pre-processing)
      2   0        1.5255313692E+00 (After scaling)
     10   0      0 2.2853462120E-05              1.0E+00      F  T
     18            3.7256423120E-10

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error14.gms(1303) 2 Mb
--- Reading solution for model PEP11
```

As the input point is quite high, we proceed to look for *infes* in the listing file.

Figure 92 : Listing file in example 14



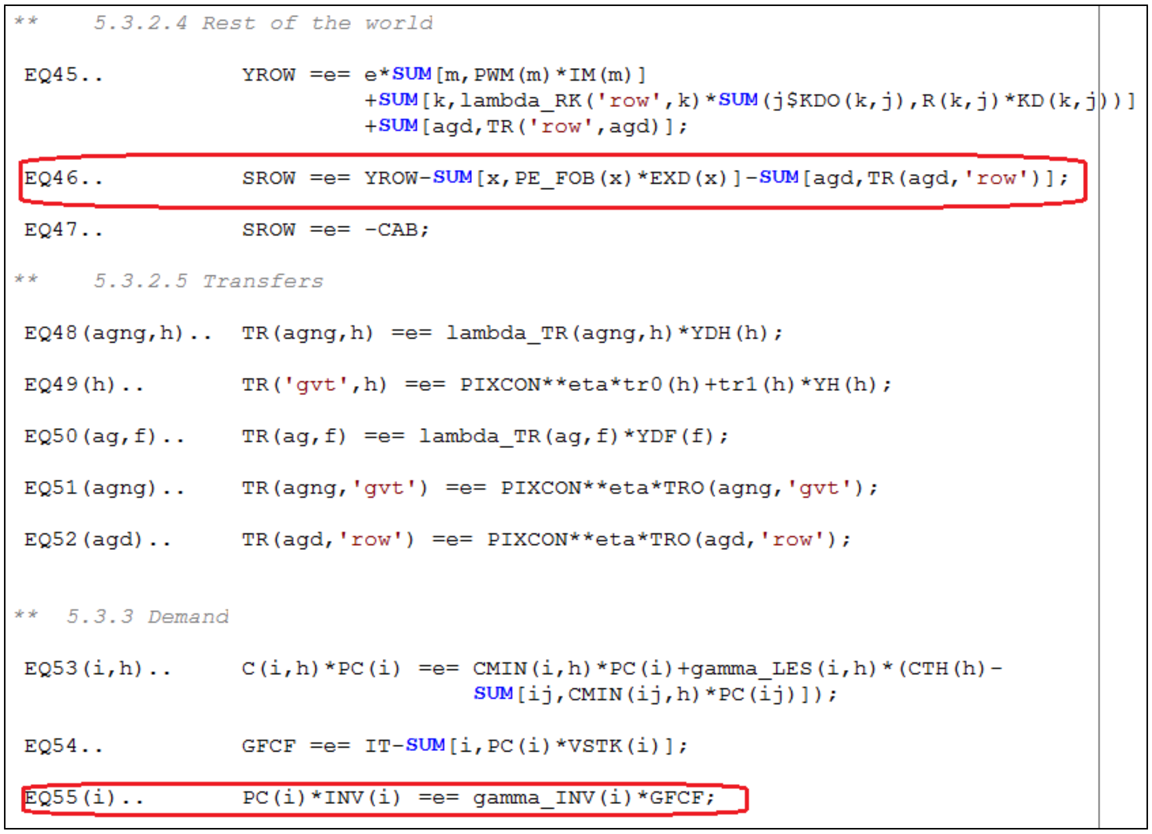
```
---- EQ46  =E=  Rest-of-the-world savings

EQ46..  (1.02143723877579)*EXD(agr) + (1)*EXD(ser) + (1)*EXD(food)

      + (1.01694915254237)*EXD(othind) + (7417)*PE_FOB(agr) + (2653)*PE_FOB(ser)

      + (241)*PE_FOB(food) + (1180)*PE_FOB(othind) + SROW + TR(hrp,row)

      + TR(hup,row) + TR(hrr,row) + TR(hur,row) + TR(firm,row) + TR(gvt,row)

      - YROW =E= 0 ; (LHS = -4194, INFES = 4194 ****)

---- EQ47  =E=  Equivalence between current account balance and ROW savings

EQ47..  SROW =E= 1231 ; (LHS = 1231)
```

```
---- EQ55  =E=  Final demand of commodity i for investment purposes (GFCF)

EQ55(agr)..  (1.06159159591308)*INV(agr) + (2038.44869188017)*PC(agr)

      - 0.239884713446403*GFCF =E= 0 ;

      (LHS = 1006.07648819421, INFES = 1006.07648819421 ****)

EQ55(ser)..  (1.03603383366743)*INV(ser) + (0)*PC(ser) =E= 0 ; (LHS = 0)

EQ55(adm)..  (1)*INV(adm) + (0)*PC(adm) =E= 0 ; (LHS = 0)

REMAINING 2 ENTRIES SKIPPED
```

We find two *infes* in the listing file, at equations 46 and 55. We cannot have a clear idea of what is wrong at this point. However, notice that *infes* in equation 46 is an exact value. This leads us to believe that it might be a value problem, or possibly an initialization problem as in the previous example.

We begin by checking the equations in the GAMS code.

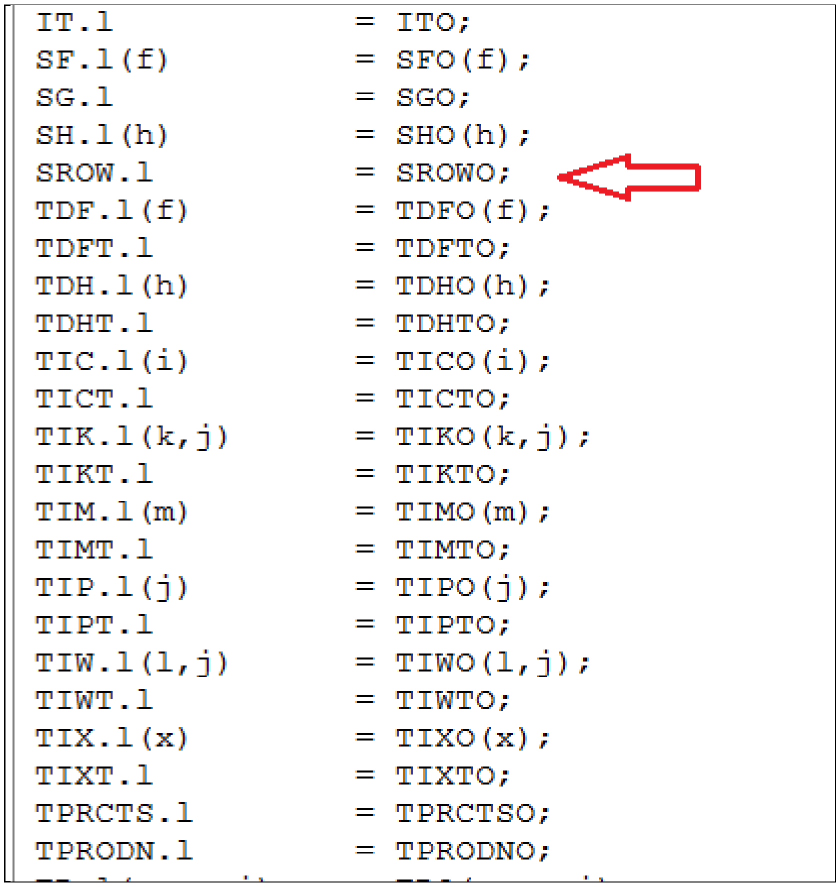
Figure 93 : Equations in example 14



```
**     5.3.2.4 Rest of the world

 EQ45..            YROW =e= e*SUM[m,PWM(m)*IM(m)]
                            +SUM[k,lambda_RK('row',k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))]
                            +SUM[agd,TR('row',agd)];

 EQ46..            SROW =e= YROW-SUM[x,PE_FOB(x)*EXD(x)]-SUM[agd,TR(agd,'row')];

 EQ47..            SROW =e= -CAB;

**     5.3.2.5 Transfers

 EQ48(agng,h)..    TR(agng,h)  =e= lambda_TR(agng,h)*YDH(h);

 EQ49(h)..         TR('gvt',h)  =e= PIXCON**eta*tr0(h)+tr1(h)*YH(h);

 EQ50(ag,f)..      TR(ag,f)  =e= lambda_TR(ag,f)*YDF(f);

 EQ51(agng)..      TR(agng,'gvt')  =e= PIXCON**eta*TRO(agng,'gvt');

 EQ52(agd)..       TR(agd,'row')  =e= PIXCON**eta*TRO(agd,'row');


**   5.3.3 Demand

 EQ53(i,h)..       C(i,h)*PC(i)  =e= CMIN(i,h)*PC(i)+gamma_LES(i,h)*(CTH(h)-
                                     SUM[ij,CMIN(ij,h)*PC(ij)]);

 EQ54..            GFCF =e= IT-SUM[i,PC(i)*VSTK(i)];

 EQ55(i)..         PC(i)*INV(i)  =e= gamma_INV(i)*GFCF;
```

Both equations are correctly specified. Our first intuition is to focus on equation 46, and check the value for *SROW*. We first check to see if the value was initialized. If it wasn't (as in the previous example), we can correct it. If the value was in fact initialized, we must verify the value assigned to it.

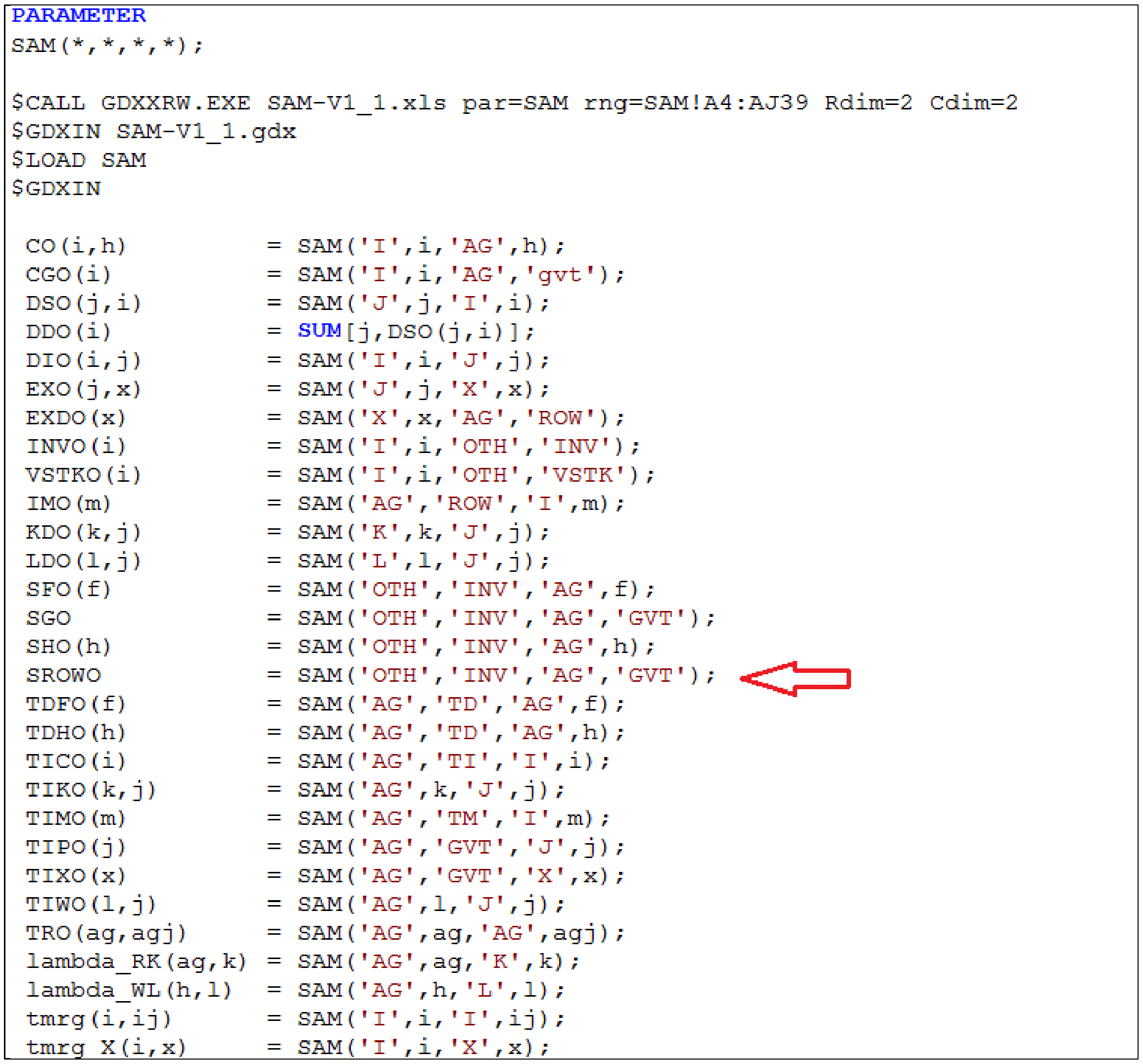
Figure 94 : Initialization in example 14



```
IT.l                = ITO;
SF.l(f)             = SFO(f);
SG.l                = SGO;
SH.l(h)             = SHO(h);
SROW.l              = SROWO;
TDF.l(f)            = TDFO(f);
TDFT.l              = TDFTO;
TDH.l(h)            = TDHO(h);
TDHT.l              = TDHTO;
TIC.l(i)            = TICO(i);
TICT.l              = TICTO;
TIK.l(k,j)          = TIKO(k,j);
TIKT.l              = TIKTO;
TIM.l(m)            = TIMO(m);
TIMT.l              = TIMTO;
TIP.l(j)            = TIPO(j);
TIPT.l              = TIPTO;
TIW.l(l,j)          = TIWO(l,j);
TIWT.l              = TIWTO;
TIX.l(x)            = TIXO(x);
TIXT.l              = TIXTO;
TPRCTS.l            = TPRCTSO;
TPRODN.l            = TPRODNO;
```

From figure 94, we can see that SROW was initialized at its benchmark value. Now we verify the value assigned to it.

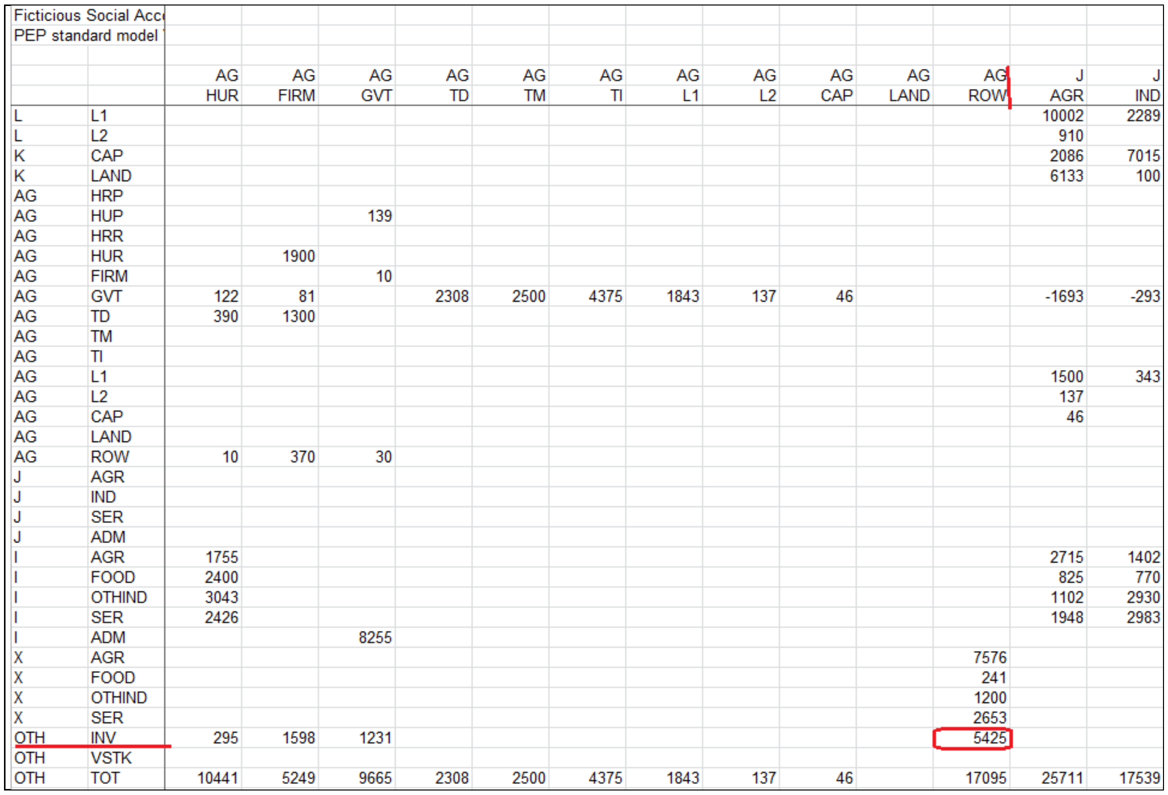
Figure 95 : Assignment of variables in example 14



```
PARAMETER
SAM(*,*,*,*);

$CALL GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2
$GDXIN SAM-V1_1.gdx
$LOAD SAM
$GDXIN

 CO(i,h)           = SAM('I',i,'AG',h);
 CGO(i)            = SAM('I',i,'AG','gvt');
 DSO(j,i)          = SAM('J',j,'I',i);
 DDO(i)            = SUM[j,DSO(j,i)];
 DIO(i,j)          = SAM('I',i,'J',j);
 EXO(j,x)          = SAM('J',j,'X',x);
 EXDO(x)           = SAM('X',x,'AG','ROW');
 INVO(i)           = SAM('I',i,'OTH','INV');
 VSTKO(i)          = SAM('I',i,'OTH','VSTK');
 IMO(m)            = SAM('AG','ROW','I',m);
 KDO(k,j)          = SAM('K',k,'J',j);
 LDO(l,j)          = SAM('L',l,'J',j);
 SFO(f)            = SAM('OTH','INV','AG',f);
 SGO               = SAM('OTH','INV','AG','GVT');
 SHO(h)            = SAM('OTH','INV','AG',h);
 SROWO             = SAM('OTH','INV','AG','GVT');
 TDFO(f)           = SAM('AG','TD','AG',f);
 TDHO(h)           = SAM('AG','TD','AG',h);
 TICO(i)           = SAM('AG','TI','I',i);
 TIKO(k,j)         = SAM('AG',k,'J',j);
 TIMO(m)           = SAM('AG','TM','I',m);
 TIPO(j)           = SAM('AG','GVT','J',j);
 TIXO(x)           = SAM('AG','GVT','X',x);
 TIWO(l,j)         = SAM('AG',l,'J',j);
 TRO(ag,agj)       = SAM('AG',ag,'AG',agj);
 lambda_RK(ag,k)   = SAM('AG',ag,'K',k);
 lambda_WL(h,l)    = SAM('AG',h,'L',l);
 tmrg(i,ij)        = SAM('I',i,'I',ij);
 tmrg_X(i,x)       = SAM('I',i,'X',x);
```

We find *SROWO* directly from the SAM.

Figure 96 : partial reproduction of the SAM:



Ficticious Social Acc														
PEP standard model														
		AG	AG	AG	AG	AG	AG	AG	AG	AG	AG	AG	J	J
		HUR	FIRM	GVT	TD	TM	TI	L1	L2	CAP	LAND	ROW	AGR	IND
L	L1												10002	2289
L	L2												910	
K	CAP												2086	7015
K	LAND												6133	100
AG	HRP													
AG	HUP			139										
AG	HRR													
AG	HUR		1900											
AG	FIRM			10										
AG	GVT	122	81		2308	2500	4375	1843	137	46			-1693	-293
AG	TD	390	1300											
AG	TM													
AG	TI													
AG	L1												1500	343
AG	L2												137	
AG	CAP												46	
AG	LAND													
AG	ROW	10	370	30										
J	AGR													
J	IND													
J	SER													
J	ADM													
I	AGR	1755											2715	1402
I	FOOD	2400											825	770
I	OTHIND	3043											1102	2930
I	SER	2426											1948	2983
I	ADM			8255										
X	AGR											7576		
X	FOOD											241		
X	OTHIND											1200		
X	SER											2653		
OTH	INV	295	1598	1231								5425		
OTH	VSTK													
OTH	TOT	10441	5249	9665	2308	2500	4375	1843	137	46		17095	25711	17539

From figure 96, we read the "rest of the world's saving", SROW, (encircled in red) at the intersection of the line "OTH","INV" (underlined in red), and the column "AG","ROW".

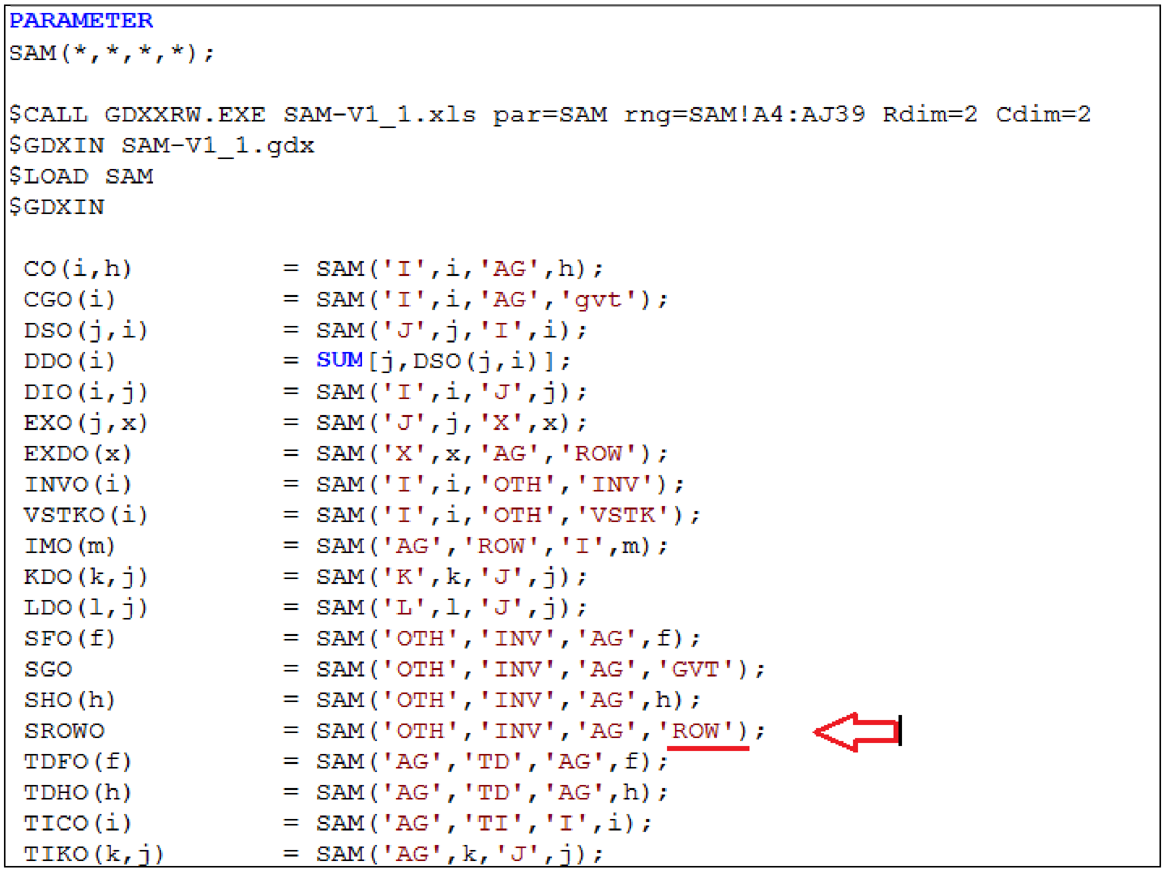
If we look at figure 95, we can see that it is not exactly how SROWO was assigned.

It is written:

SROWO= SAM("OTH","INV","AG","GVT") instead of SROWO= SAM("OTH","INV","AG","ROW"). In other words, we have assigned the value of "government's savings" for the "rest of the world's savings".

We then need to correct the error.

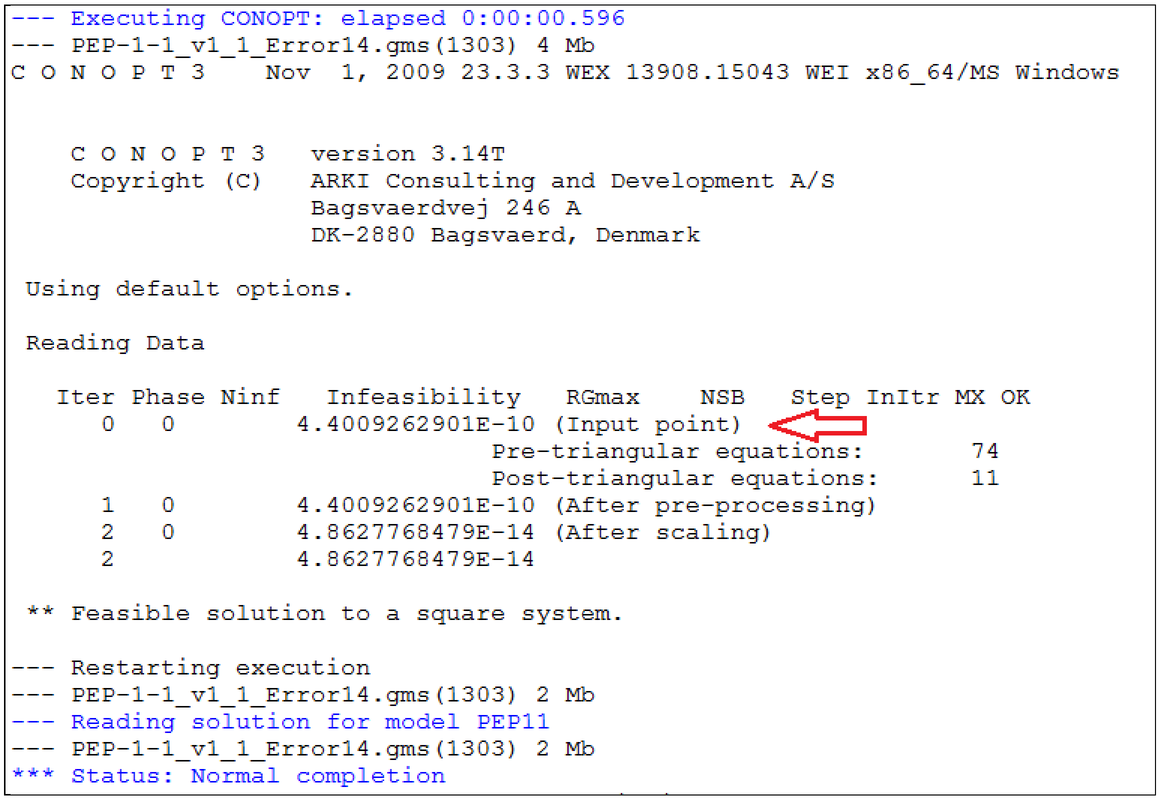
Figure 97 : Assignment of SROWO after correction in example 14



```
PARAMETER
SAM(*,*,*,*);

$CALL GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2
$GDXIN SAM-V1_1.gdx
$LOAD SAM
$GDXIN

 CO(i,h)            = SAM('I',i,'AG',h);
 CGO(i)             = SAM('I',i,'AG','gvt');
 DSO(j,i)           = SAM('J',j,'I',i);
 DDO(i)             = SUM[j,DSO(j,i)];
 DIO(i,j)           = SAM('I',i,'J',j);
 EXO(j,x)           = SAM('J',j,'X',x);
 EXDO(x)            = SAM('X',x,'AG','ROW');
 INVO(i)            = SAM('I',i,'OTH','INV');
 VSTKO(i)           = SAM('I',i,'OTH','VSTK');
 IMO(m)             = SAM('AG','ROW','I',m);
 KDO(k,j)           = SAM('K',k,'J',j);
 LDO(l,j)           = SAM('L',l,'J',j);
 SFO(f)             = SAM('OTH','INV','AG',f);
 SGO                = SAM('OTH','INV','AG','GVT');
 SHO(h)             = SAM('OTH','INV','AG',h);
 SROWO              = SAM('OTH','INV','AG','ROW');
 TDFO(f)            = SAM('AG','TD','AG',f);
 TDHO(h)            = SAM('AG','TD','AG',h);
 TICO(i)            = SAM('AG','TI','I',i);
 TIKO(k,j)          = SAM('AG',k,'J',j);
```

We can now run the model again and see if we have removed at least one *infes*.

Figure 98 : Process window after correction in example 14

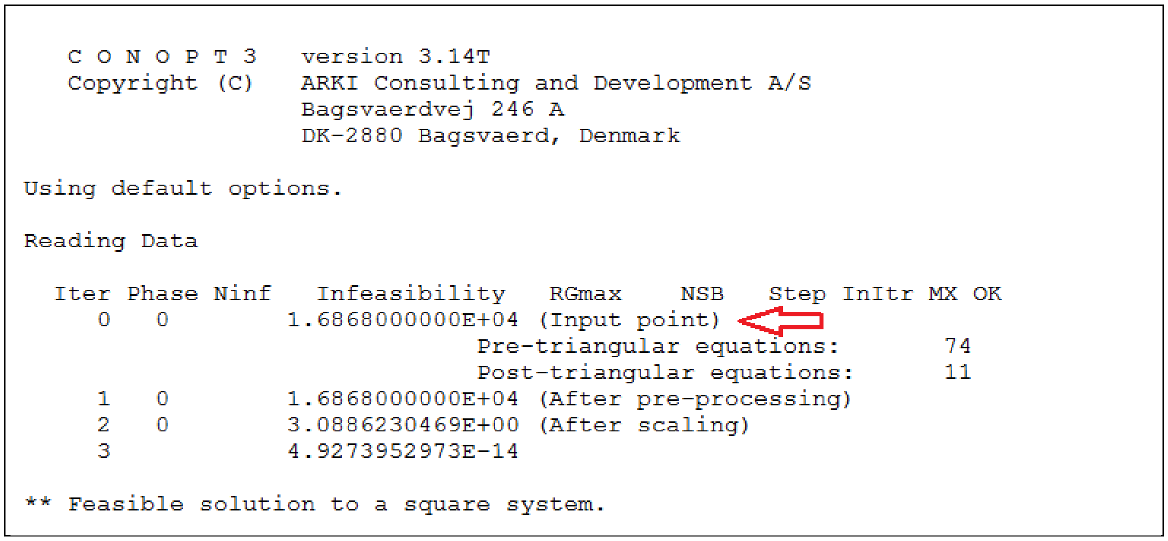

```
--- Executing CONOPT: elapsed 0:00:00.596
--- PEP-1-1_v1_1_Error14.gms(1303) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                 Pre-triangular equations:      74
                                 Post-triangular equations:     11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error14.gms(1303) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error14.gms(1303) 2 Mb
*** Status: Normal completion
```

From figure 98, we can see that we have an ideal input point, meaning that by correcting the value of *SROW*, we also corrected the second *infes*. In effect, the incorrect value of *SROW* was affecting equation 55, (via the computation of *IT*).

Example 15:

Example 15 is described in the figure below:

Figure 99 : Process window in example 15

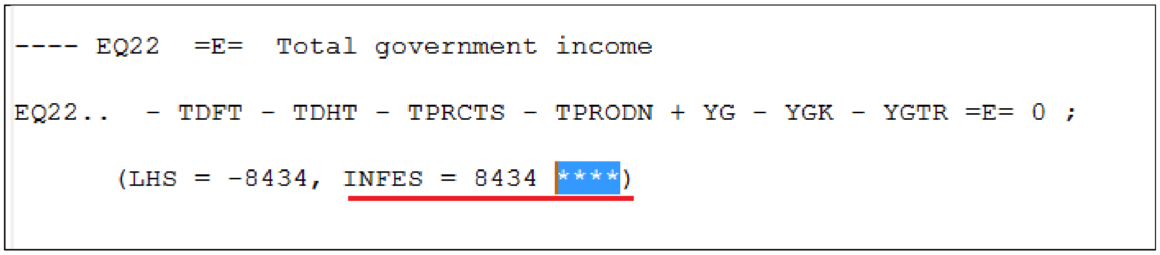


```
   C O N O P T 3   version 3.14T
   Copyright (C)   ARKI Consulting and Development A/S
                   Bagsvaerdvej 246 A
                   DK-2880 Bagsvaerd, Denmark

Using default options.

Reading Data

  Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
     0   0        1.6868000000E+04 (Input point)
                               Pre-triangular equations:      74
                               Post-triangular equations:     11
     1   0        1.6868000000E+04 (After pre-processing)
     2   0        3.0886230469E+00 (After scaling)
     3            4.9273952973E-14

** Feasible solution to a square system.
```

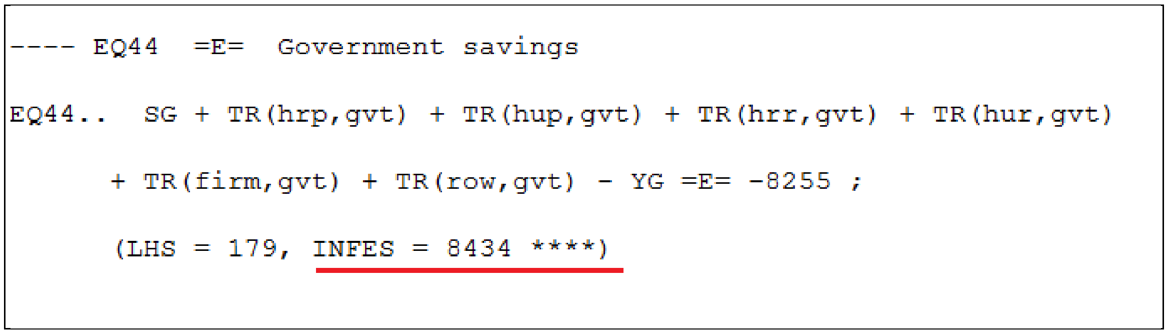
To find the *infes*, we go in the listing file and search for the four asterisks using the flashlight.

We find the following:

Figure 100 : Infes found in the listing file example 15



```
---- EQ22  =E=  Total government income

EQ22..  - TDFT - TDHT - TPRCTS - TPRODN + YG - YGK - YGTR =E= 0 ;

       (LHS = -8434, INFES = 8434 ****)
```



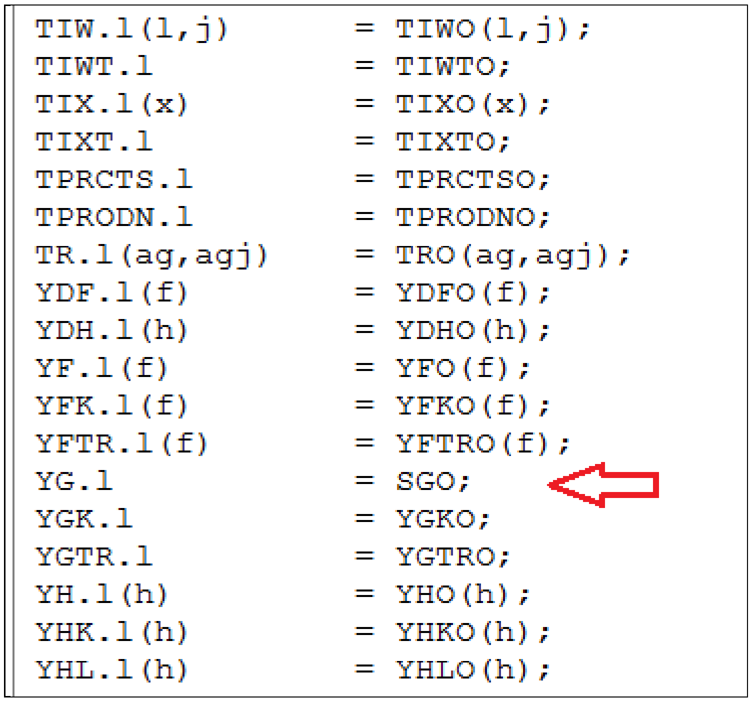
```
---- EQ44  =E=  Government savings

EQ44..  SG + TR(hrp,gvt) + TR(hup,gvt) + TR(hrr,gvt) + TR(hur,gvt)

      + TR(firm,gvt) + TR(row,gvt) - YG =E= -8255 ;

      (LHS = 179, INFES = 8434 ****)
```

We find two *infes*, one at equation 22, dealing with government's income, and the second at equation 44, dealing with government's savings.

Before we begin our investigation, we may notice that for both equations, we have an exact value, and that it is the same value in both equations. This leads to the intuition that the *infes* should come from a data assignment or initialization.

In both equations, we have the common variable *YG*. We should start with checking the initialization of this variable. If the initialization is in fact correct, we will then check the assignment of the variable.

Figure 101 : Initialization of YG in example 15

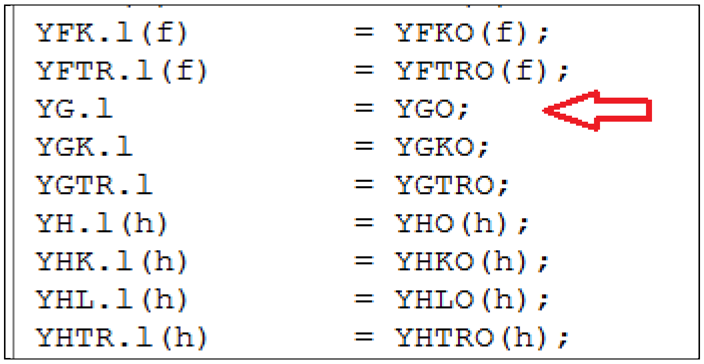


```
TIW.l(l,j)        = TIWO(l,j);
TIWT.l            = TIWTO;
TIX.l(x)          = TIXO(x);
TIXT.l            = TIXTO;
TPRCTS.l          = TPRCTSO;
TPRODN.l          = TPRODNO;
TR.l(ag,agj)      = TRO(ag,agj);
YDF.l(f)          = YDFO(f);
YDH.l(h)          = YDHO(h);
YF.l(f)           = YFO(f);
YFK.l(f)          = YFKO(f);
YFTR.l(f)         = YFTRO(f);
YG.l              = SGO;
YGK.l             = YGKO;
YGTR.l            = YGTRO;
YH.l(h)           = YHO(h);
YHK.l(h)          = YHKO(h);
YHL.l(h)          = YHLO(h);
```

From figure 101 we find that *YG* is in fact initialized, but not to its correct value. Here *YG* is initialized as *SGO* instead of *YGO*.

To correct the error, we simply need to initialize *YG* at its correct benchmark value, *YGO*.

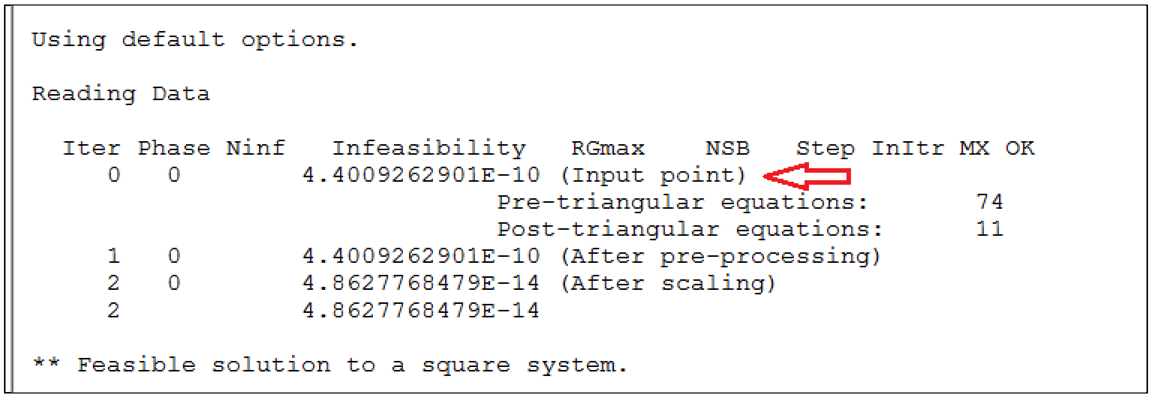
Figure 102 : Correction of initial value for YG in example 15



```
YFK.l(f)          = YFKO(f);
YFTR.l(f)         = YFTRO(f);
YG.l              = YGO;
YGK.l             = YGKO;
YGTR.l            = YGTRO;
YH.l(h)           = YHO(h);
YHK.l(h)          = YHKO(h);
YHL.l(h)          = YHLO(h);
YHTR.l(h)         = YHTRO(h);
```

Once again, we can run the model and check that the input point is very small, as shown in the figure below:

Figure 103 : Process window of example 15 after correction

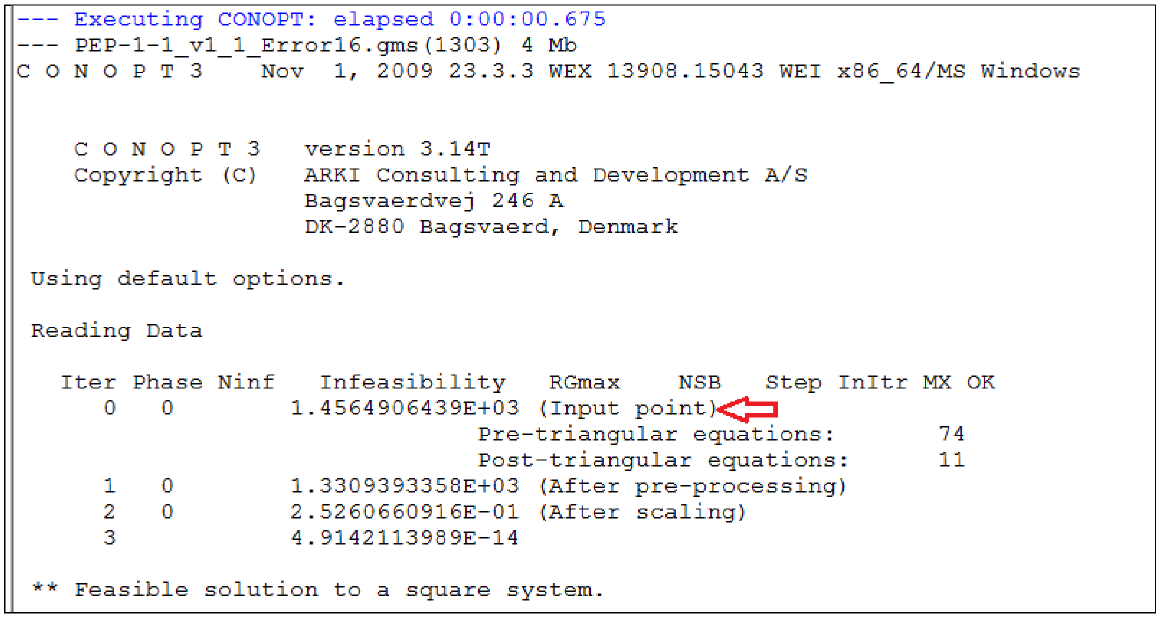


```
Using default options.

Reading Data

  Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
     0   0        4.4009262901E-10 (Input point)
                               Pre-triangular equations:       74
                               Post-triangular equations:      11
     1   0        4.4009262901E-10 (After pre-processing)
     2   0        4.8627768479E-14 (After scaling)
     2            4.8627768479E-14

** Feasible solution to a square system.
```

Example 16:

Example 16 is defined as follows:

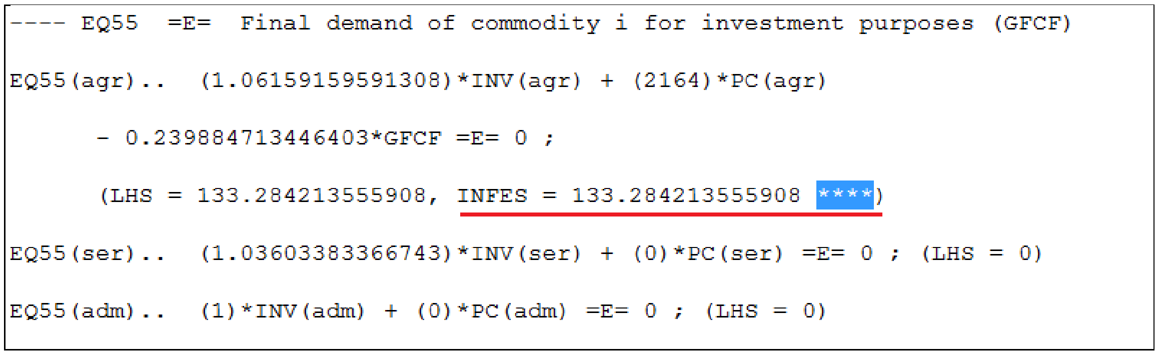
Figure 104 : Process window of example 16



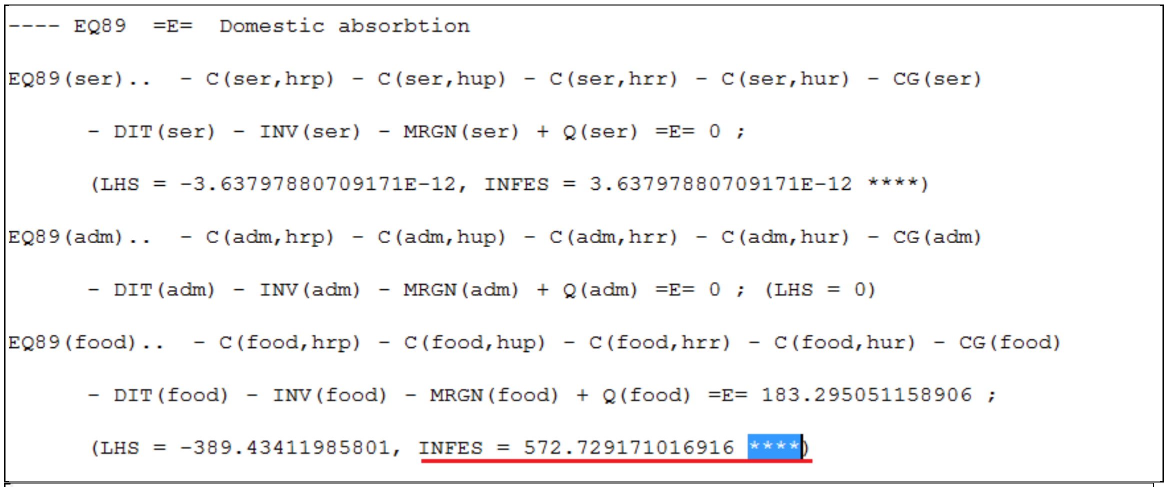
```
--- Executing CONOPT: elapsed 0:00:00.675
--- PEP-1-1_v1_1_Error16.gms(1303) 4 Mb
C O N O P T 3    Nov  1, 2009 23.3.3 WEX 13908.15043 WEI x86_64/MS Windows


    C O N O P T 3   version 3.14T
    Copyright (C)   ARKI Consulting and Development A/S
                    Bagsvaerdvej 246 A
                    DK-2880 Bagsvaerd, Denmark

 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        1.4564906439E+03 (Input point)
                                Pre-triangular equations:       74
                                Post-triangular equations:      11
      1   0        1.3309393358E+03 (After pre-processing)
      2   0        2.5260660916E-01 (After scaling)
      3            4.9142113989E-14

 ** Feasible solution to a square system.
```

We go in the listing file to find out where the *infes* are:

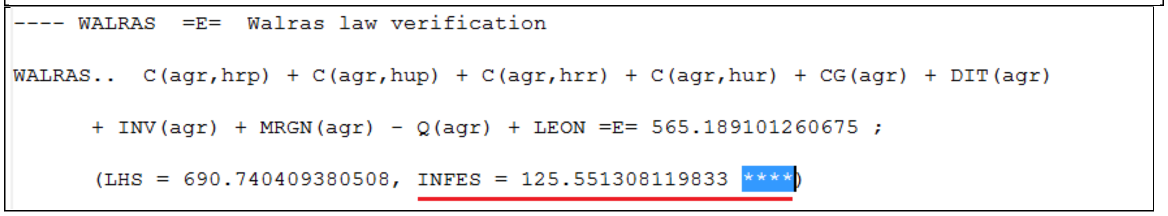
Figure 105 : Listing file in example 16



```
---- EQ55  =E=  Final demand of commodity i for investment purposes (GFCF)

EQ55(agr)..   (1.06159159591308)*INV(agr) + (2164)*PC(agr)

      - 0.239884713446403*GFCF =E= 0 ;

      (LHS = 133.284213555908, INFES = 133.284213555908 ****)

EQ55(ser)..   (1.03603383366743)*INV(ser) + (0)*PC(ser) =E= 0 ; (LHS = 0)

EQ55(adm)..   (1)*INV(adm) + (0)*PC(adm) =E= 0 ; (LHS = 0)
```



```
---- EQ89  =E=  Domestic absorbtion

EQ89(ser)..  - C(ser,hrp) - C(ser,hup) - C(ser,hrr) - C(ser,hur) - CG(ser)

      - DIT(ser) - INV(ser) - MRGN(ser) + Q(ser) =E= 0 ;

      (LHS = -3.63797880709171E-12, INFES = 3.63797880709171E-12 ****)

EQ89(adm)..  - C(adm,hrp) - C(adm,hup) - C(adm,hrr) - C(adm,hur) - CG(adm)

      - DIT(adm) - INV(adm) - MRGN(adm) + Q(adm) =E= 0 ; (LHS = 0)

EQ89(food)..  - C(food,hrp) - C(food,hup) - C(food,hrr) - C(food,hur) - CG(food)

      - DIT(food) - INV(food) - MRGN(food) + Q(food) =E= 183.295051158906 ;

      (LHS = -389.43411985801, INFES = 572.729171016916 ****)
```

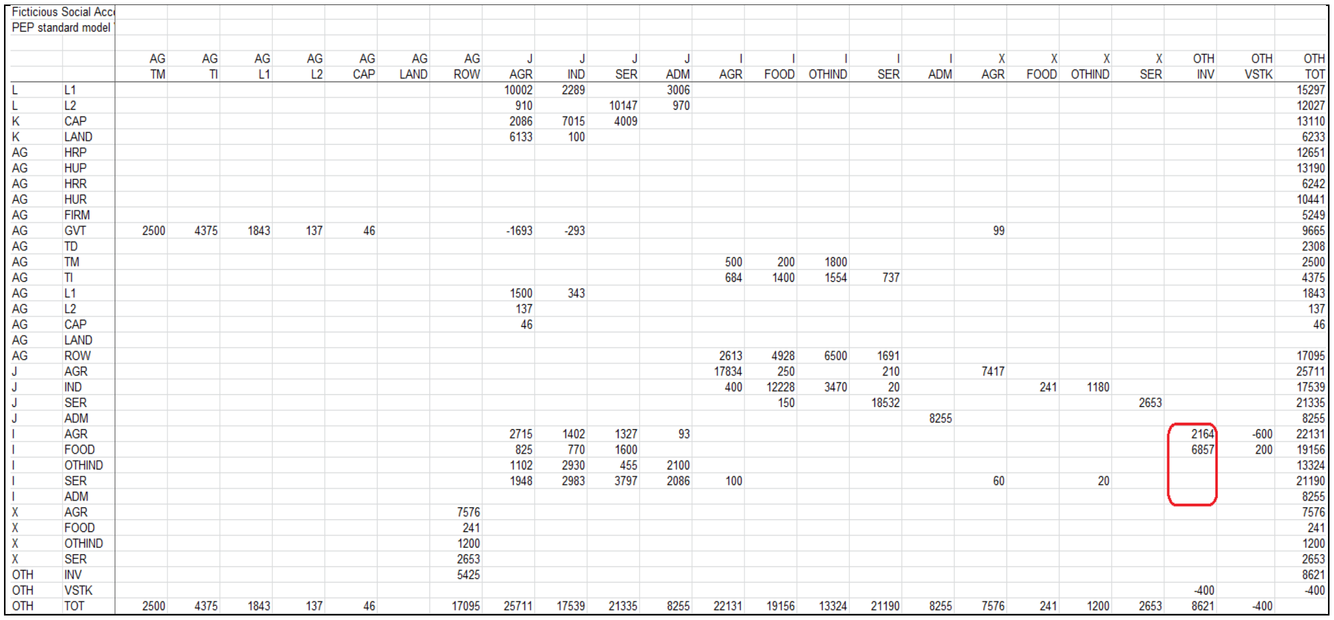


```
---- WALRAS  =E=  Walras law verification

WALRAS..  C(agr,hrp) + C(agr,hup) + C(agr,hrr) + C(agr,hur) + CG(agr) + DIT(agr)

      + INV(agr) + MRGN(agr) - Q(agr) + LEON =E= 565.189101260675 ;

      (LHS = 690.740409380508, INFES = 125.551308119833 ****)
```

We find three *infes* for this example.

What is interesting about these *infes* is that they only appear for the agricultural and food commodities. Equation 55 computes the *INV(i)* variable, and this variable only exists for agriculture and food. If you have a look in the SAM, you will see that there is no final demand for investment purposes for services, administration or other food commodities.

Figure 106 : SAM of PEP 1-1



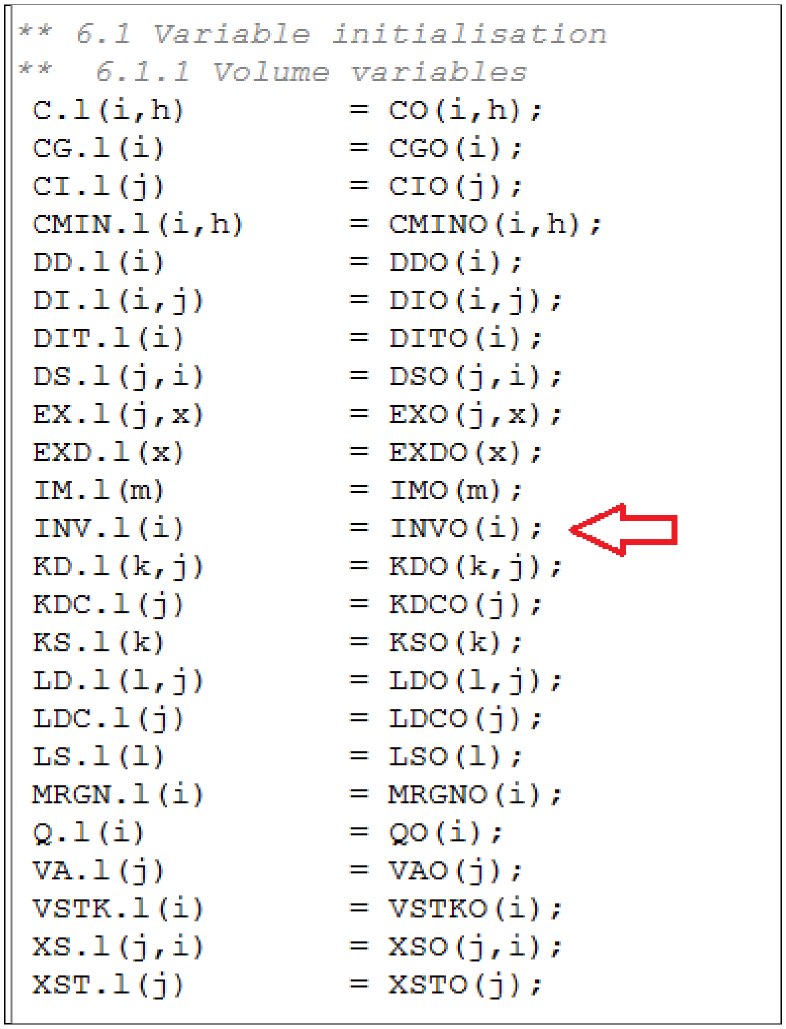
Ficticious Social Acc																								
PEP standard model																								
		AG	AG	AG	AG	AG	AG	AG	J	J	J	J	I	I	I	I	I	X	X	X	X	OTH	OTH	OTH
		TM	TI	L1	L2	CAP	LAND	ROW	AGR	IND	SER	ADM	AGR	FOOD	OTHIND	SER	ADM	AGR	FOOD	OTHIND	SER	INV	VSTK	TOT
L	L1								10002	2289		3006												15297
L	L2								910		10147	970												12027
K	CAP								2086	7015	4009													13110
K	LAND								6133	100														6233
AG	HRP																							12651
AG	HUP																							13190
AG	HRR																							6242
AG	HUR																							10441
AG	FIRM																							5249
AG	GVT	2500	4375	1843	137	46			-1693	-293								99						9665
AG	TD																							2308
AG	TM												500	200	1800									2500
AG	TI												684	1400	1554	737								4375
AG	L1								1500	343														1843
AG	L2								137															137
AG	CAP								46															46
AG	LAND																							
AG	ROW												2613	4928	6500	1691								17095
J	AGR												17834	250		210		7417						25711
J	IND												400	12228	3470	20			241	1180				17539
J	SER													150		18532					2653			21335
J	ADM																8255							8255
I	AGR								2715	1402	1327	93										2164	-600	22131
I	FOOD								825	770	1600											6857	200	19156
I	OTHIND								1102	2930	455	2100												13324
I	SER								1948	2983	3797	2086	100					60		20				21190
I	ADM																							8255
X	AGR							7576																7576
X	FOOD							241																241
X	OTHIND							1200																1200
X	SER							2653																2653
OTH	INV							5425																8621
OTH	VSTK																					-400		-400
OTH	TOT	2500	4375	1843	137	46		17095	25711	17539	21335	8255	22131	19156	13324	21190	8255	7576	241	1200	2653	8621	-400	

INV(i) will be the first variable we will check.

We will apply the same procedure as above to find the source of the *infes*.

First, we will check how the variable *INV(i)* is initialized:

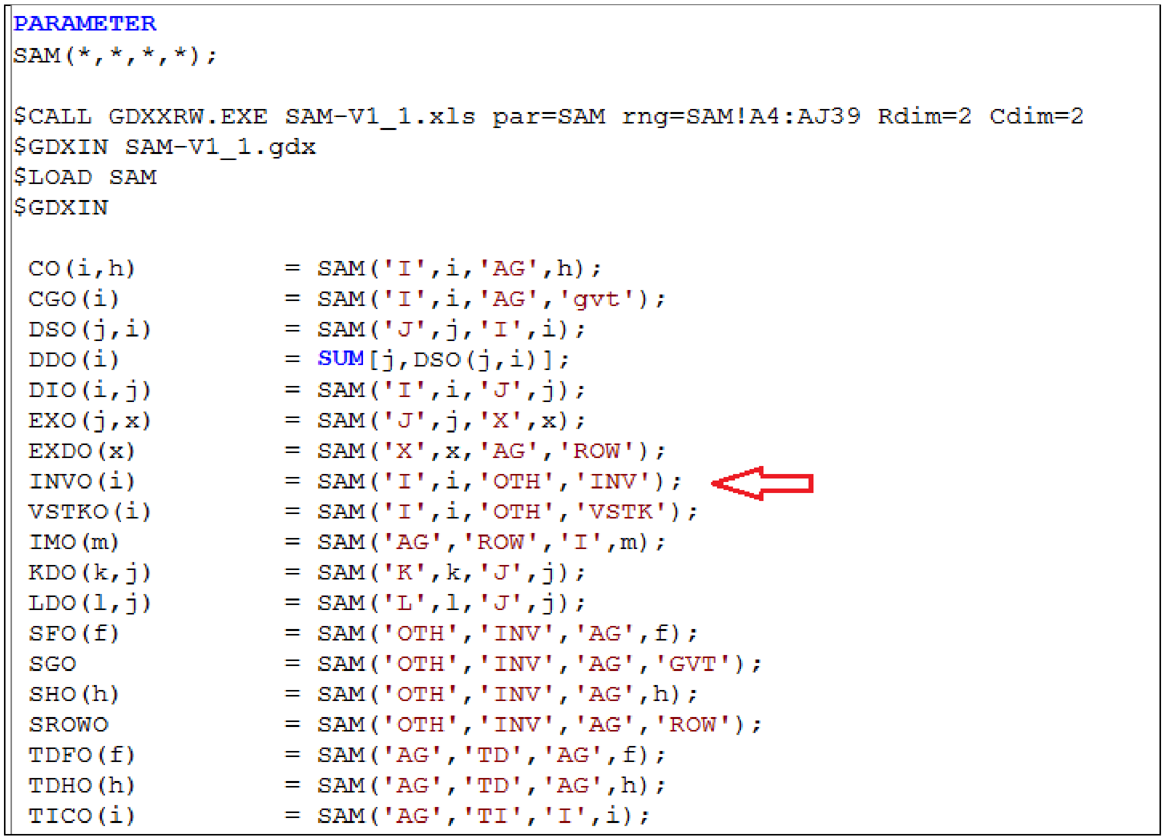
Figure 107 : Initialization of INV(i) in example 16



```
** 6.1 Variable initialisation
**   6.1.1 Volume variables
 C.l(i,h)            = CO(i,h);
 CG.l(i)             = CGO(i);
 CI.l(j)             = CIO(j);
 CMIN.l(i,h)         = CMINO(i,h);
 DD.l(i)             = DDO(i);
 DI.l(i,j)           = DIO(i,j);
 DIT.l(i)            = DITO(i);
 DS.l(j,i)           = DSO(j,i);
 EX.l(j,x)           = EXO(j,x);
 EXD.l(x)            = EXDO(x);
 IM.l(m)             = IMO(m);
 INV.l(i)            = INVO(i);
 KD.l(k,j)           = KDO(k,j);
 KDC.l(j)            = KDCO(j);
 KS.l(k)             = KSO(k);
 LD.l(l,j)           = LDO(l,j);
 LDC.l(j)            = LDCO(j);
 LS.l(l)             = LSO(l);
 MRGN.l(i)           = MRGNO(i);
 Q.l(i)              = QO(i);
 VA.l(j)             = VAO(j);
 VSTK.l(i)           = VSTKO(i);
 XS.l(j,i)           = XSO(j,i);
 XST.l(j)            = XSTO(j);
```

From the figure above, we can see that *INV(i)* is correctly initialized. Let's now go and check how *INV(i)* is assigned in the calibration process.

Figure 108 : Assignment of INVO(i) in example 16



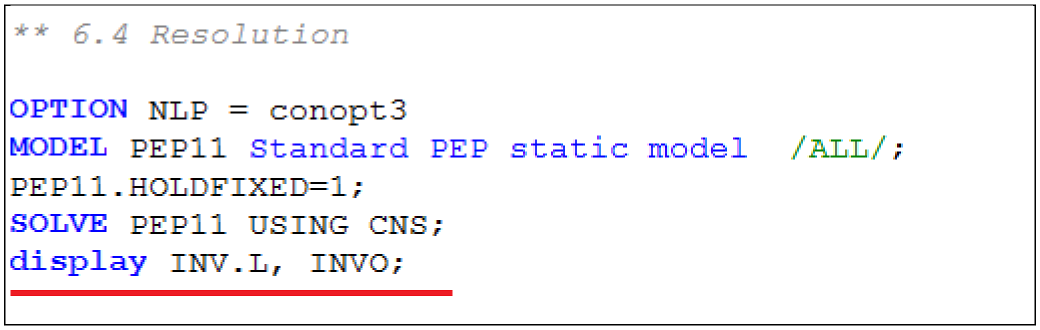
```
PARAMETER
SAM(*,*,*,*);

$CALL GDXXRW.EXE SAM-V1_1.xls par=SAM rng=SAM!A4:AJ39 Rdim=2 Cdim=2
$GDXIN SAM-V1_1.gdx
$LOAD SAM
$GDXIN

 CO(i,h)          = SAM('I',i,'AG',h);
 CGO(i)           = SAM('I',i,'AG','gvt');
 DSO(j,i)         = SAM('J',j,'I',i);
 DDO(i)           = SUM[j,DSO(j,i)];
 DIO(i,j)         = SAM('I',i,'J',j);
 EXO(j,x)         = SAM('J',j,'X',x);
 EXDO(x)          = SAM('X',x,'AG','ROW');
 INVO(i)          = SAM('I',i,'OTH','INV');
 VSTKO(i)         = SAM('I',i,'OTH','VSTK');
 IMO(m)           = SAM('AG','ROW','I',m);
 KDO(k,j)         = SAM('K',k,'J',j);
 LDO(l,j)         = SAM('L',l,'J',j);
 SFO(f)           = SAM('OTH','INV','AG',f);
 SGO              = SAM('OTH','INV','AG','GVT');
 SHO(h)           = SAM('OTH','INV','AG',h);
 SROWO            = SAM('OTH','INV','AG','ROW');
 TDFO(f)          = SAM('AG','TD','AG',f);
 TDHO(h)          = SAM('AG','TD','AG',h);
 TICO(i)          = SAM('AG','TI','I',i);
```

Figure 106 presents the SAM. The assignment of the variable *INVO(i)* is directly read from the SAM. If we pay attention to the line *INVO(i)* (red arrow), we can see that the variable is correctly assigned.

Thus, we need to investigate further for this example.

We go back into the GAMS file and we display the variables *INV(i)* and *INVO(i)*. The two should be equal.

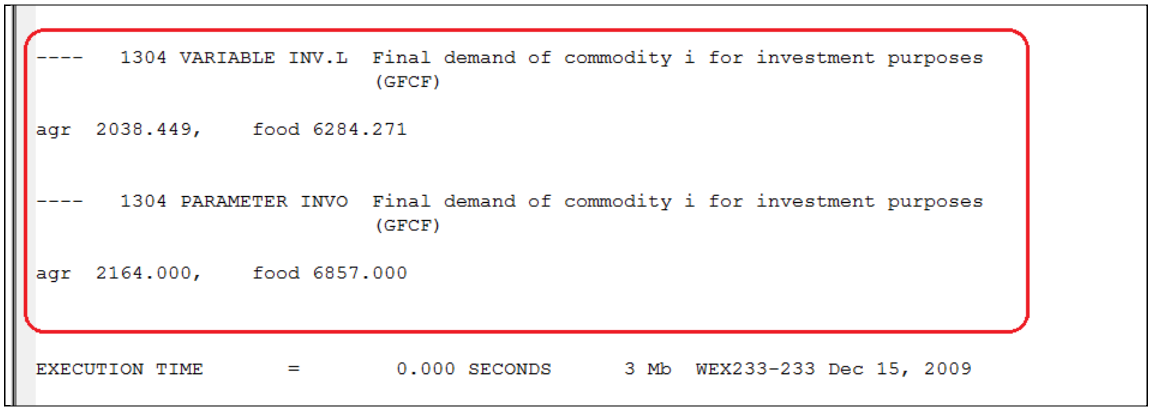
Figure 109 : Display of INV(i) in example 16



```
** 6.4 Resolution

OPTION NLP = conopt3
MODEL PEP11 Standard PEP static model  /ALL/;
PEP11.HOLDFIXED=1;
SOLVE PEP11 USING CNS;
display INV.L, INVO;
```

In the listing file, we find the following results:

Figure 110 : Display of INV(i) and INVO(i) in example 16



```
----    1304 VARIABLE INV.L  Final demand of commodity i for investment purposes
                             (GFCF)

agr  2038.449,     food 6284.271


----    1304 PARAMETER INVO  Final demand of commodity i for investment purposes
                             (GFCF)

agr  2164.000,     food 6857.000


EXECUTION TIME       =        0.000 SECONDS      3 Mb  WEX233-233 Dec 15, 2009
```

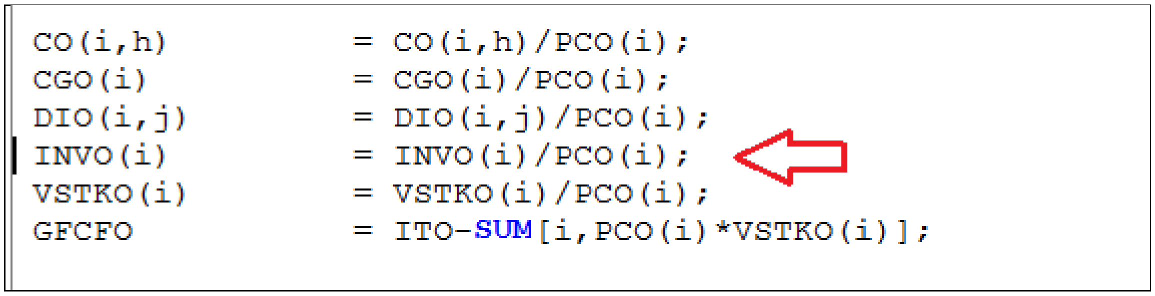
The result is very interesting. The two values are different. *INVO(i)* is exactly equal to the value in the SAM. However, the values we get from the SAM are actual monetary values, meaning that the variables are expressed in a given currency.

This particular variable, *INV(I,)* is extracted from the SAM as a value. We then have to divide it by its price in order to convert it to volume (we want to know how many cows were sold in order to increase the capital stock of a sector the following year (this would be *INV("agr")*).

To compute the volume of *INV(i),* we have to divide the value we extract from the SAM by its price. The price of *INV(i)* is the composite price *PC(i)*.

We then add the following line in the code:

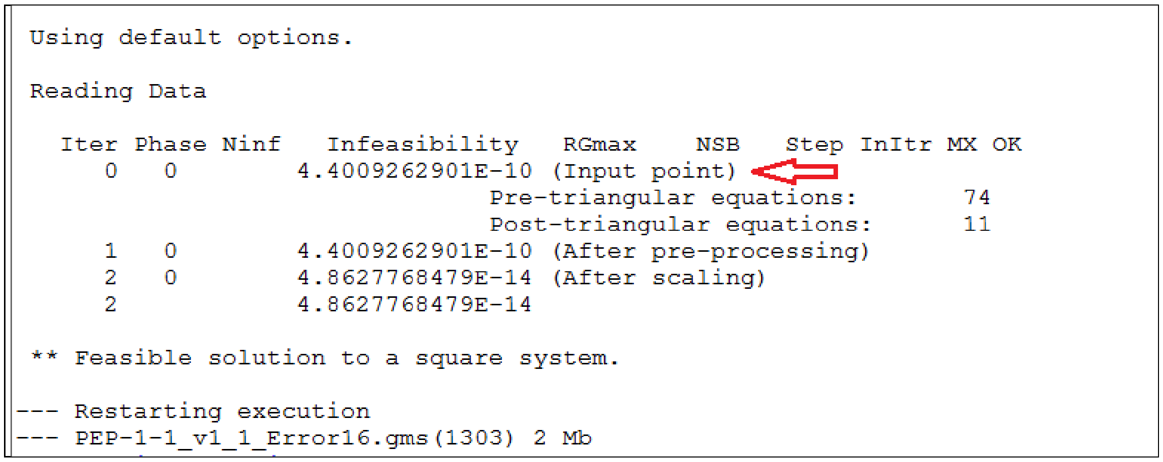
Figure 111 : Obtaining volumes from values in example 16



```
CO(i,h)             = CO(i,h)/PCO(i);
CGO(i)              = CGO(i)/PCO(i);
DIO(i,j)            = DIO(i,j)/PCO(i);
INVO(i)             = INVO(i)/PCO(i);
VSTKO(i)            = VSTKO(i)/PCO(i);
GFCFO               = ITO-SUM[i,PCO(i)*VSTKO(i)];
```

We can then re- run the model and check that it works:

Figure 112 : Process window of example 16 after correction

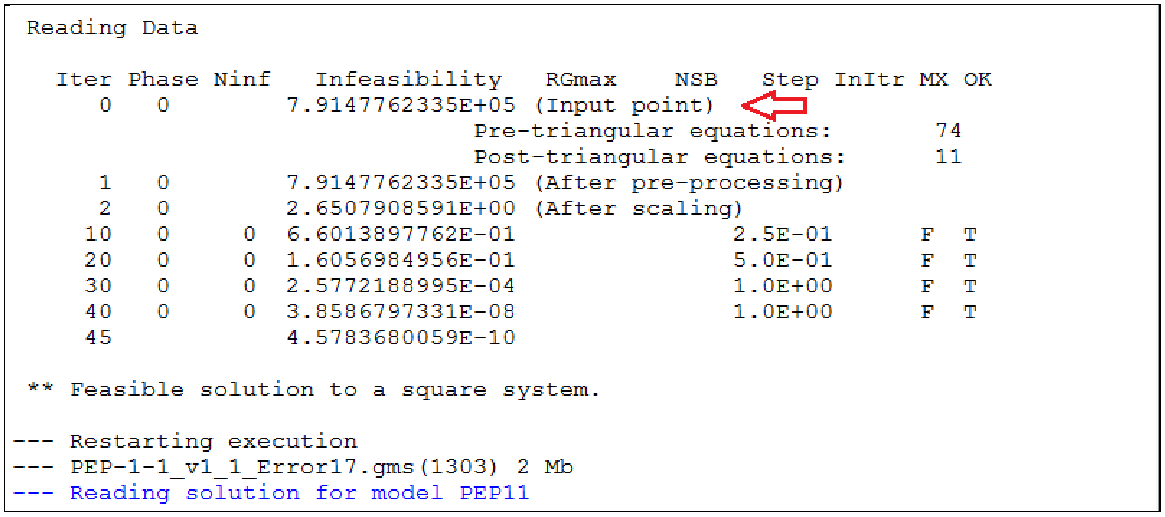


```
Using default options.

Reading Data

  Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
     0   0        4.4009262901E-10 (Input point)
                               Pre-triangular equations:       74
                               Post-triangular equations:      11
     1   0        4.4009262901E-10 (After pre-processing)
     2   0        4.8627768479E-14 (After scaling)
     2            4.8627768479E-14

** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error16.gms(1303) 2 Mb
```

Example 17:

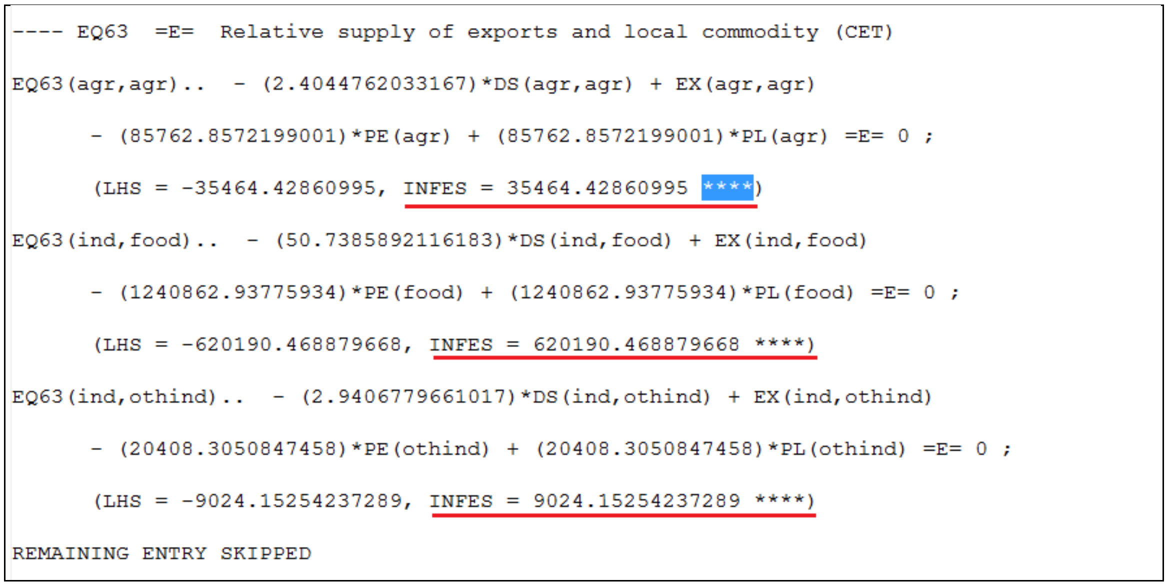
Example 17 is presented as follows:

Figure 113 : Process window for example 17


```
 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0    0        7.9147762335E+05 (Input point)
                                 Pre-triangular equations:      74
                                 Post-triangular equations:     11
      1    0        7.9147762335E+05 (After pre-processing)
      2    0        2.6507908591E+00 (After scaling)
     10    0      0  6.6013897762E-01              2.5E-01      F  T
     20    0      0  1.6056984956E-01              5.0E-01      F  T
     30    0      0  2.5772188995E-04              1.0E+00      F  T
     40    0      0  3.8586797331E-08              1.0E+00      F  T
     45             4.5783680059E-10

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error17.gms(1303) 2 Mb
--- Reading solution for model PEP11
```

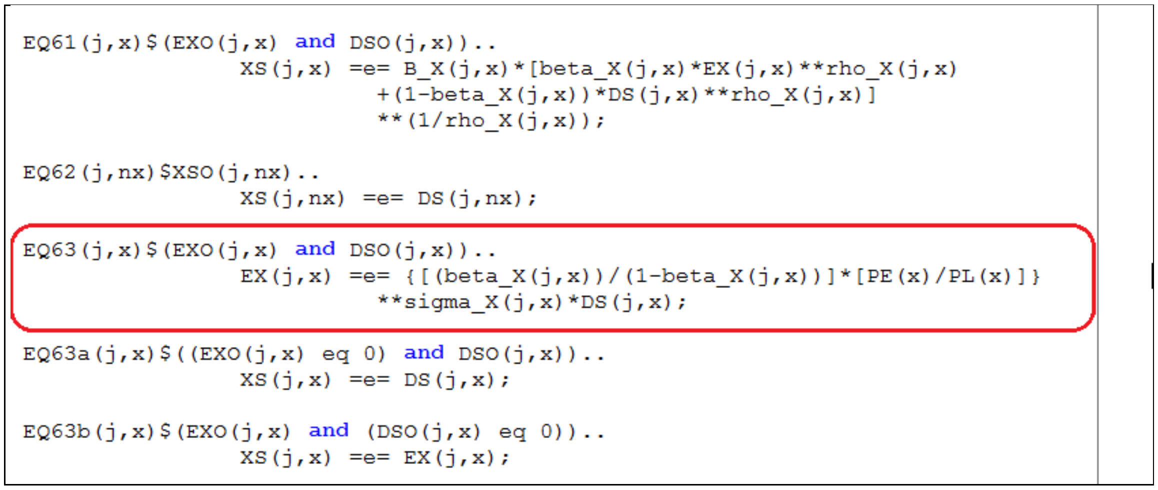
As in the previous examples, we go in the listing file to find the *infes*. As shown in the following figure, we find that there is only one *infes* on equation 63.

Figure 114 : Infes in example 17


```
---- EQ63  =E=  Relative supply of exports and local commodity (CET)

EQ63(agr,agr)..  - (2.4044762033167)*DS(agr,agr) + EX(agr,agr)

      - (85762.8572199001)*PE(agr) + (85762.8572199001)*PL(agr) =E= 0 ;

      (LHS = -35464.42860995, INFES = 35464.42860995 ****)

EQ63(ind,food)..  - (50.7385892116183)*DS(ind,food) + EX(ind,food)

      - (1240862.93775934)*PE(food) + (1240862.93775934)*PL(food) =E= 0 ;

      (LHS = -620190.468879668, INFES = 620190.468879668 ****)

EQ63(ind,othind)..  - (2.9406779661017)*DS(ind,othind) + EX(ind,othind)

      - (20408.3050847458)*PE(othind) + (20408.3050847458)*PL(othind) =E= 0 ;

      (LHS = -9024.15254237289, INFES = 9024.15254237289 ****)

REMAINING ENTRY SKIPPED
```

Because we are only concerned with one equation, we will first go and check how the equation is specified, if it is written correctly, and how the parameters are calibrated.

Figure 115 : Equation 63 in the GAMS code in example 17



```
EQ61(j,x)$(EXO(j,x) and DSO(j,x))..
                XS(j,x) =e= B_X(j,x)*[beta_X(j,x)*EX(j,x)**rho_X(j,x)
                          +(1-beta_X(j,x))*DS(j,x)**rho_X(j,x)]
                          **(1/rho_X(j,x));

EQ62(j,nx)$XSO(j,nx)..
                XS(j,nx) =e= DS(j,nx);

EQ63(j,x)$(EXO(j,x) and DSO(j,x))..
                EX(j,x) =e= {[(beta_X(j,x))/(1-beta_X(j,x))]*[PE(x)/PL(x)]}
                          **sigma_X(j,x)*DS(j,x);

EQ63a(j,x)$((EXO(j,x) eq 0) and DSO(j,x))..
                XS(j,x) =e= DS(j,x);

EQ63b(j,x)$(EXO(j,x) and (DSO(j,x) eq 0))..
                XS(j,x) =e= EX(j,x);
```

Equation 63 is the relative supply of export and local commodity derived from the CET function expressed in equation 61[15].

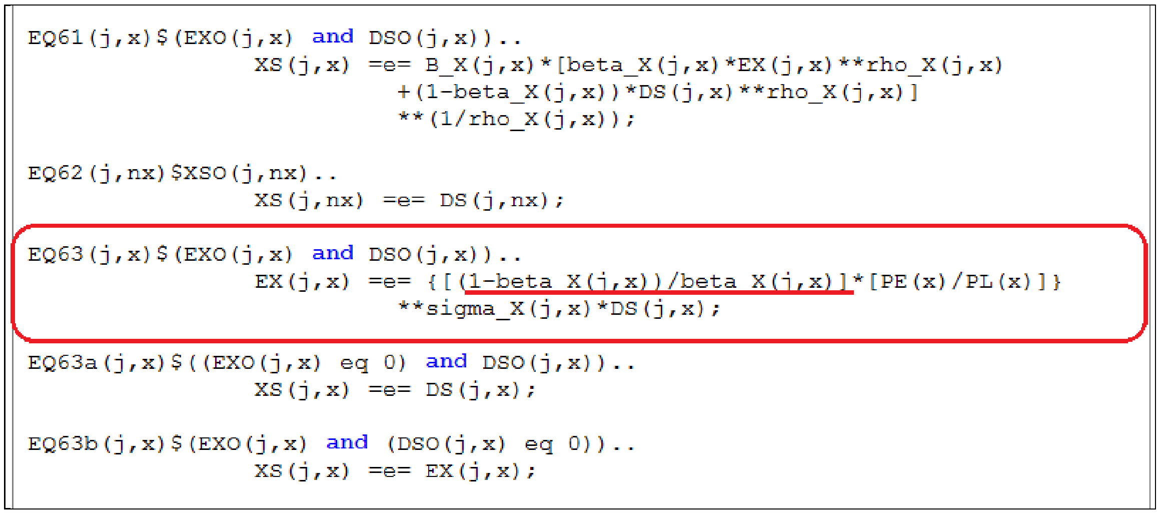
At this point, it's important to check your calculations to ensure that the way equation 63 is written is consistent with the way equation 61 and the *beta_X(j,x)* parameter are calibrated.

When checking on the calculations for the relative supply function, we have a look at figure 114, and find that it is not exactly how we wrote it. Indeed, as the parameter *beta_X(j,x)* refers to the share of exports, the way we specified the equation is not correct.

We then need to change the ratio between the share parameters in equation 63.

[15] For the complete mathematics (derivation of the first-order conditions of revenue maximizing subject to the CET aggregator function defined in equation 61), please refer to Decaluwé et al (2009), pages 87-88

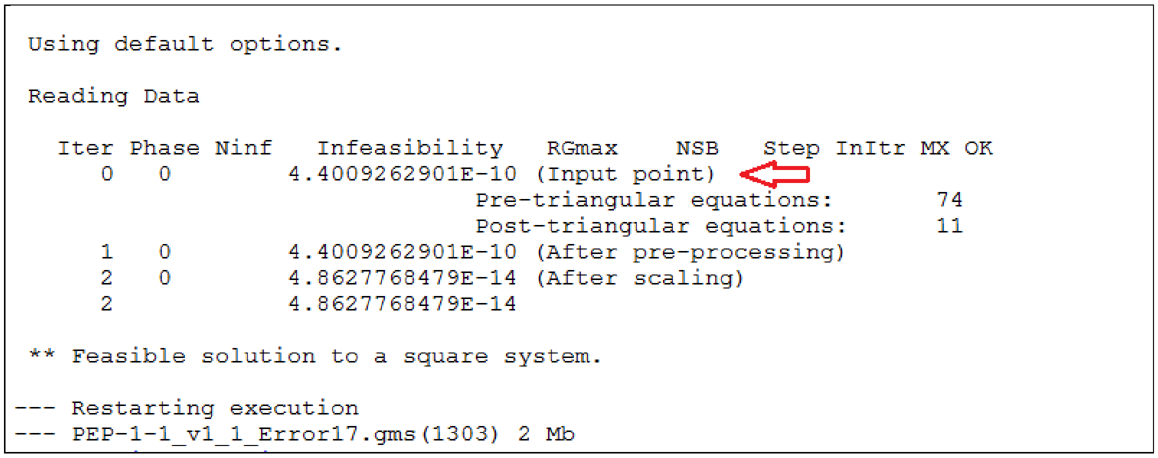
Figure 116 : Correction of equation 63 in example 17



```
EQ61(j,x)$(EXO(j,x) and DSO(j,x))..
                XS(j,x)  =e= B_X(j,x)*[beta_X(j,x)*EX(j,x)**rho_X(j,x)
                                   +(1-beta_X(j,x))*DS(j,x)**rho_X(j,x)]
                                   **(1/rho_X(j,x));

EQ62(j,nx)$XSO(j,nx)..
                XS(j,nx)  =e= DS(j,nx);

EQ63(j,x)$(EXO(j,x) and DSO(j,x))..
                EX(j,x)  =e= {[(1-beta_X(j,x))/beta_X(j,x)]*[PE(x)/PL(x)]}
                                   **sigma_X(j,x)*DS(j,x);

EQ63a(j,x)$((EXO(j,x) eq 0) and DSO(j,x))..
                XS(j,x)  =e= DS(j,x);

EQ63b(j,x)$(EXO(j,x) and (DSO(j,x) eq 0))..
                XS(j,x)  =e= EX(j,x);
```

We then run the model and check that in the process window, the value for the input point is very small.

Figure 117 : Process window after correction of example 17



```
 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        4.4009262901E-10 (Input point)
                                Pre-triangular equations:       74
                                Post-triangular equations:      11
      1   0        4.4009262901E-10 (After pre-processing)
      2   0        4.8627768479E-14 (After scaling)
      2            4.8627768479E-14

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error17.gms(1303) 2 Mb
```

Though this example looks like example 11 in the way that for both, equations were not correctly specified, we found it relevant to introduce a second example implying the same kind of mistake. Actually, in example 17, the mistake is less obvious than in example 11. The idea for both examples, and especially the latter, is that the modeller needs to go back to his or her calculations to check if he or she correctly specified the equations.

4. Specification errors

The last type of error is by far the most difficult for the modeller to solve. Unfortunately, there is no set of rules to follow. However, we can offer some guidelines for solutions based on our experience.

We began with how to fix computation errors (examples 1 to 6). Once these are fixed, if possible, GAMS solves the model (examples 7 to 10). With no compilation errors and a square model with no division by zero, GAMS tries to reproduce the base year, and eventually we would have calibration errors (example 11 to 17).

We insisted on the value of the input point in the process window without a shock.

Once the model replicates the base year, we need to run a simulation. As explained in Robichaud et al (2011), in section 5-3, when the modeller runs a shock, he or she wants to check that the model is correctly specified by checking the value of the control variable (in PEP 1-1, its name is *LEON*). This value must be very small.

For this type of error, the modeller needs to go back to his pen and paper.

- First ask yourself if the simulation you are running makes sense given the hypotheses you have chosen. For example, let's say that you assume a Leontief type of function between labour and capital in the production function. Capital is sector specific. You want to simulate a flood of foreign workers in the economy by simulating an increase of labour supply in your model. Given the hypothesis you chose (a very restrictive function between labour and capital and sector specific capital), the model cannot adjust. Thus, the value of the variable *LEON* will be inflated. In this case, you need to think more about the hypotheses you chose given the simulations that you want to run.
- A second explanation could be the magnitude of the shock that is introduced in the simulation. If the magnitude is too big, then the model won't be able to adjust. There again, the value of *LEON* will be too big.
- SAM related problems: you may have problems if you have very extreme values in one specific sector of the SAM. For instance, if 98 % of a commodity available in the economy comes from imports, and a tariff removal shock is applied, this might create a problem since it will be very difficult to replace imports by domestic sales.

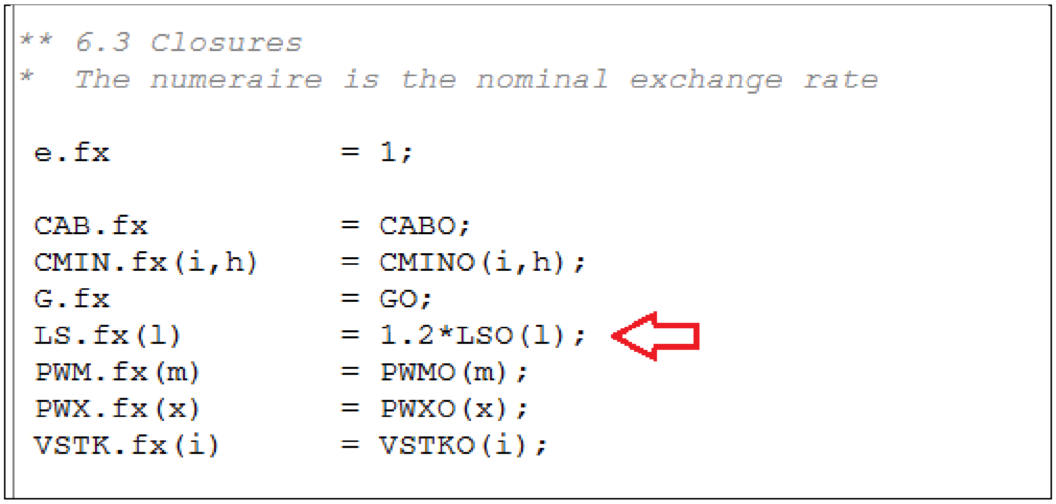
The list above is not exhaustive, of course. Let's assume for the remaining examples that we have already ruled out the above explanations, and we still do not know why the value of *LEON* is so significant.

In the three following cases, we will provide some hints for the modeller to find solutions, but as previously mentioned, solving this type of error can be extremely difficult and time consuming.

Example 18:

For this example, we have introduced an increase of labour supply by 20% as shown in figure 118.

Figure 118 : Increase of labour supply by 20% in example 18

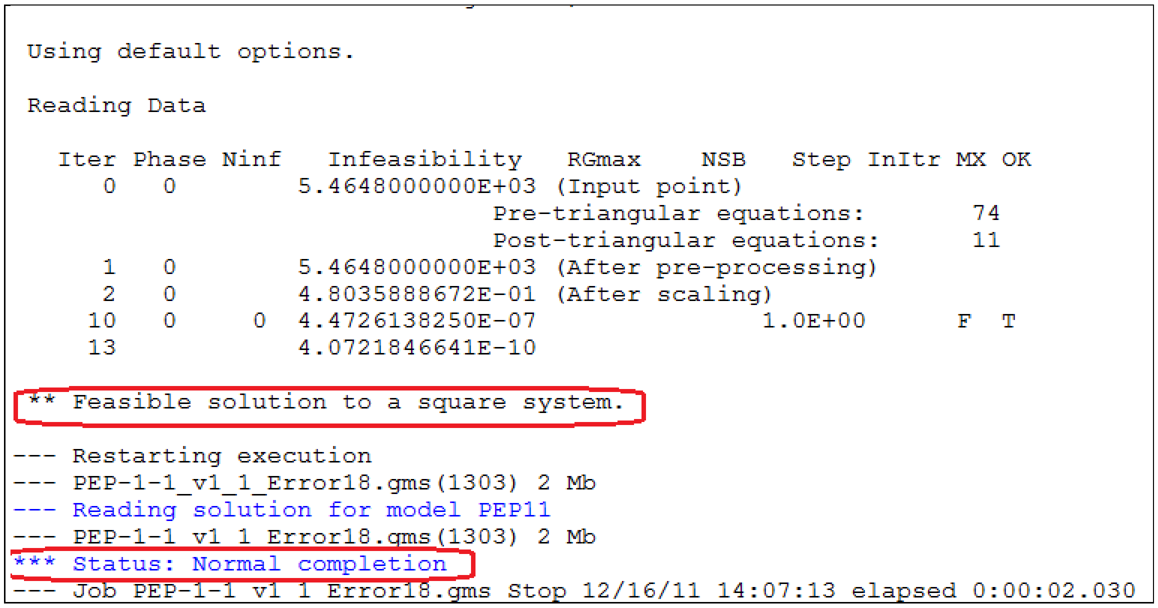


```
** 6.3 Closures
*  The numeraire is the nominal exchange rate

 e.fx                = 1;

 CAB.fx              = CABO;
 CMIN.fx(i,h)        = CMINO(i,h);
 G.fx                = GO;
 LS.fx(l)            = 1.2*LSO(l);
 PWM.fx(m)           = PWMO(m);
 PWX.fx(x)           = PWXO(x);
 VSTK.fx(i)          = VSTKO(i);
```

The process window now shows a large input point. This is normal as a simulation was introduced[16]. The modeller wants to check in the process window that the model is feasible to a square system, and that there is a normal completion.

16 As we explained previously, GAMS replaces the values of each parameter and variables by its benchmark values, as it is the way we initialized the variables. When we introduce a shock, we force a value to be different from its benchmark. In effect, this will create a difference between the left hand side and the right hand side for at least one equation. Thus the input point will report the magnitude of the difference.

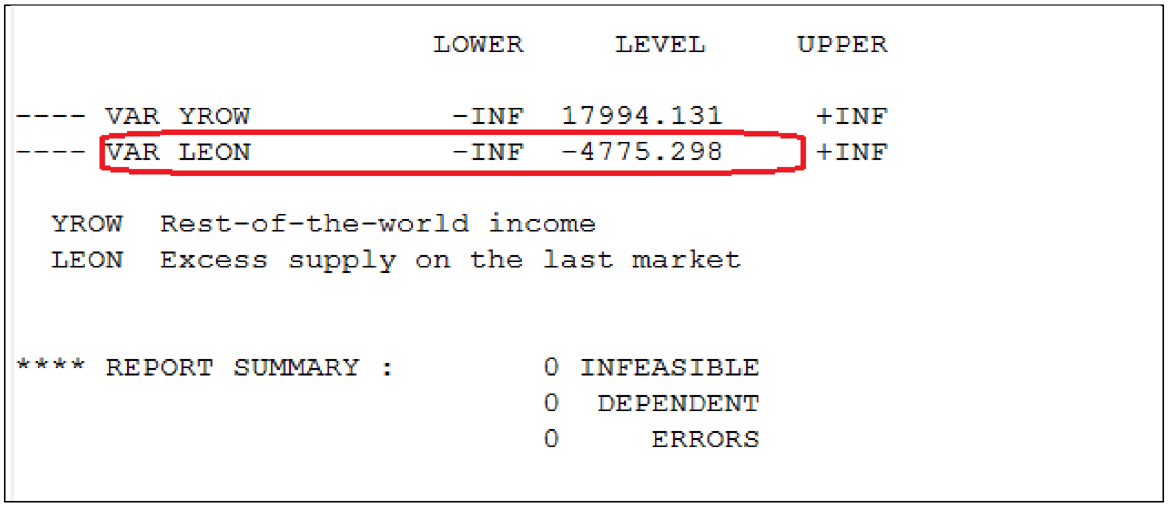
Figure 119 : Process window in example 18



```
 Using default options.

 Reading Data

   Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
      0   0        5.4648000000E+03 (Input point)
                               Pre-triangular equations:       74
                               Post-triangular equations:      11
      1   0        5.4648000000E+03 (After pre-processing)
      2   0        4.8035888672E-01 (After scaling)
     10   0      0 4.4726138250E-07               1.0E+00      F  T
     13            4.0721846641E-10

 ** Feasible solution to a square system.

--- Restarting execution
--- PEP-1-1_v1_1_Error18.gms(1303) 2 Mb
--- Reading solution for model PEP11
--- PEP-1-1_v1_1_Error18.gms(1303) 2 Mb
*** Status: Normal completion
--- Job PEP-1-1_v1_1_Error18.gms Stop 12/16/11 14:07:13 elapsed 0:00:02.030
```

Now we want to go and check the value of *LEON* at the end of the listing file.

Figure 120 : Check on variable LEON after simulation in example 18



```
                           LOWER     LEVEL     UPPER

---- VAR YROW               -INF  17994.131     +INF
---- VAR LEON               -INF  -4775.298     +INF

  YROW  Rest-of-the-world income
  LEON  Excess supply on the last market


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

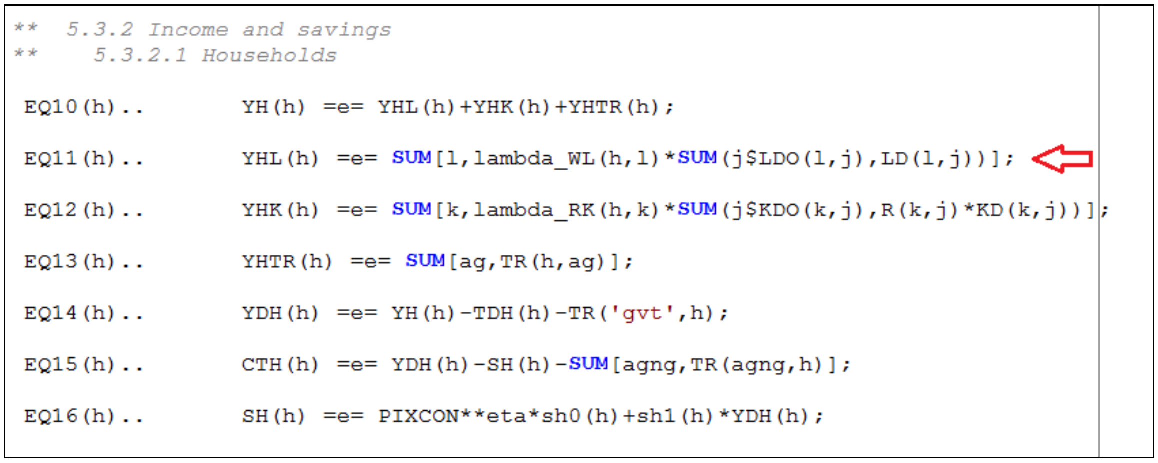
From figure 120, we can see that the value of *LEON* is significant.

Given that the first three hypotheses have already been checked, we need to study the equations very carefully.

The first equations the modeller should have a look at, are the ones that contain a price equal to 1 at the benchmark. Some prices are initialized to one, but if a shock is introduced, then their value is going to change. If the equation is not specified correctly, this is where the problem will appear.

Example 18 will illustrate our statements. We review each equation of example 18.

Figure 121 : Reviewing equations in example 18



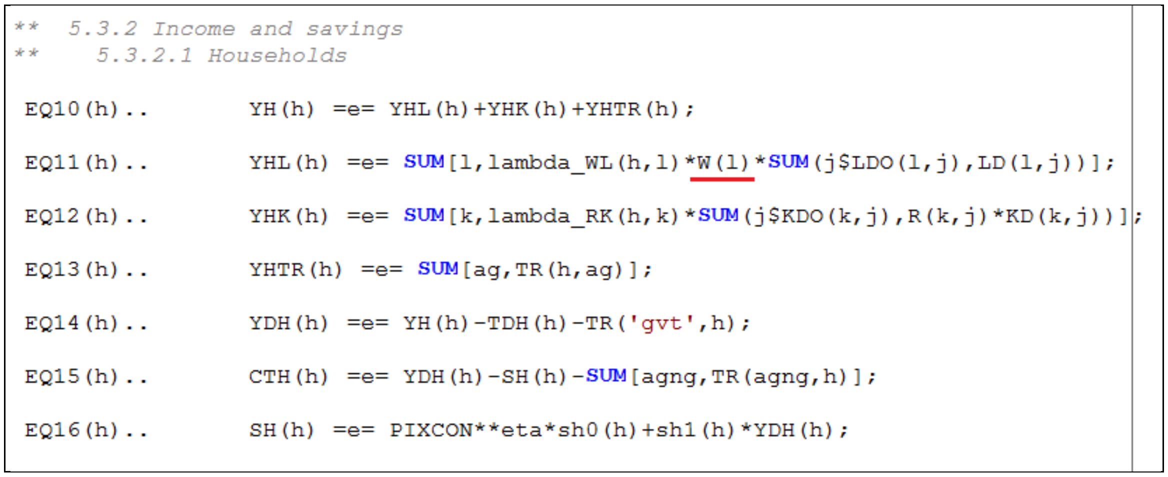
```
**   5.3.2 Income and savings
**     5.3.2.1 Households

 EQ10(h)..          YH(h)  =e= YHL(h)+YHK(h)+YHTR(h);

 EQ11(h)..          YHL(h)  =e= SUM[l,lambda_WL(h,l)*SUM(j$LDO(l,j),LD(l,j))];

 EQ12(h)..          YHK(h)  =e= SUM[k,lambda_RK(h,k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ13(h)..          YHTR(h)  =e= SUM[ag,TR(h,ag)];

 EQ14(h)..          YDH(h)  =e= YH(h)-TDH(h)-TR('gvt',h);

 EQ15(h)..          CTH(h)  =e= YDH(h)-SH(h)-SUM[agng,TR(agng,h)];

 EQ16(h)..          SH(h)  =e= PIXCON**eta*sh0(h)+sh1(h)*YDH(h);
```

If we focus on equation 11, we can see that there is indeed a problem. *YHL(h)* represents households labour income: each household receives a share of the earnings of each type of labour. As it is written in example 18 (and figure 121), there is a price missing on the right hand side of the equation, namely the wage rate.

As the benchmark value is equal to 1 for *W(l)*, this omission was not harmful for replicating the benchmark values. When introducing a shock, prices change and thus equality is not respected anymore.

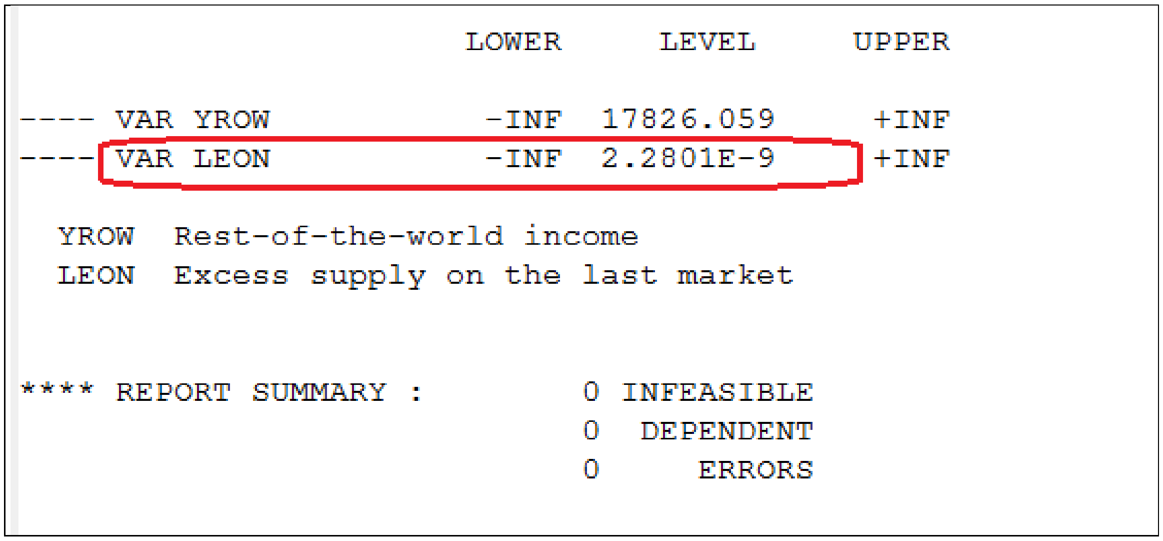
We then need to correct the writing of the equation by introducing the wage rate

Figure 122 : Correction of example 18


```
**  5.3.2 Income and savings
**    5.3.2.1 Households

 EQ10(h)..        YH(h)  =e=  YHL(h)+YHK(h)+YHTR(h);

 EQ11(h)..        YHL(h)  =e=  SUM[l,lambda_WL(h,l)*W(l)*SUM(j$LDO(l,j),LD(l,j))];

 EQ12(h)..        YHK(h)  =e=  SUM[k,lambda_RK(h,k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ13(h)..        YHTR(h)  =e=  SUM[ag,TR(h,ag)];

 EQ14(h)..        YDH(h)  =e=  YH(h)-TDH(h)-TR('gvt',h);

 EQ15(h)..        CTH(h)  =e=  YDH(h)-SH(h)-SUM[agng,TR(agng,h)];

 EQ16(h)..        SH(h)  =e=  PIXCON**eta*sh0(h)+sh1(h)*YDH(h);
```

After this change, we want to run the model again and check the value of *LEON*.

Figure 123 : Check on variable LEON after correction in example 18

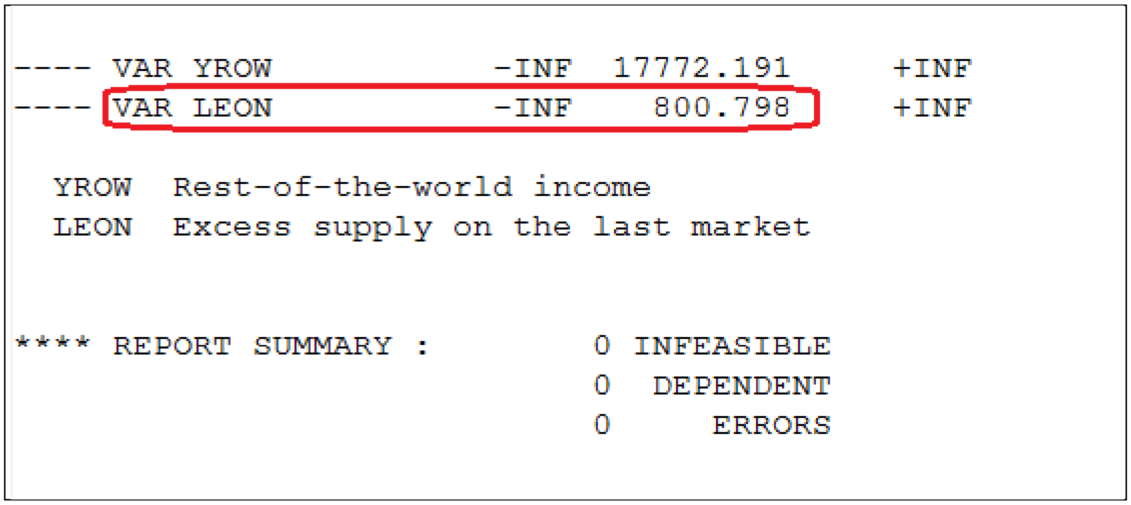

```
                        LOWER     LEVEL     UPPER

---- VAR YROW            -INF   17826.059    +INF
---- VAR LEON            -INF   2.2801E-9    +INF

  YROW  Rest-of-the-world income
  LEON  Excess supply on the last market


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

Here, the value of *LEON* after the shock is very small. Now the modeller can start analyzing the results of the simulation.

Example 19:

The following figure presents the value of *LEON* after a 20% increase in labour supply (as in the previous example).

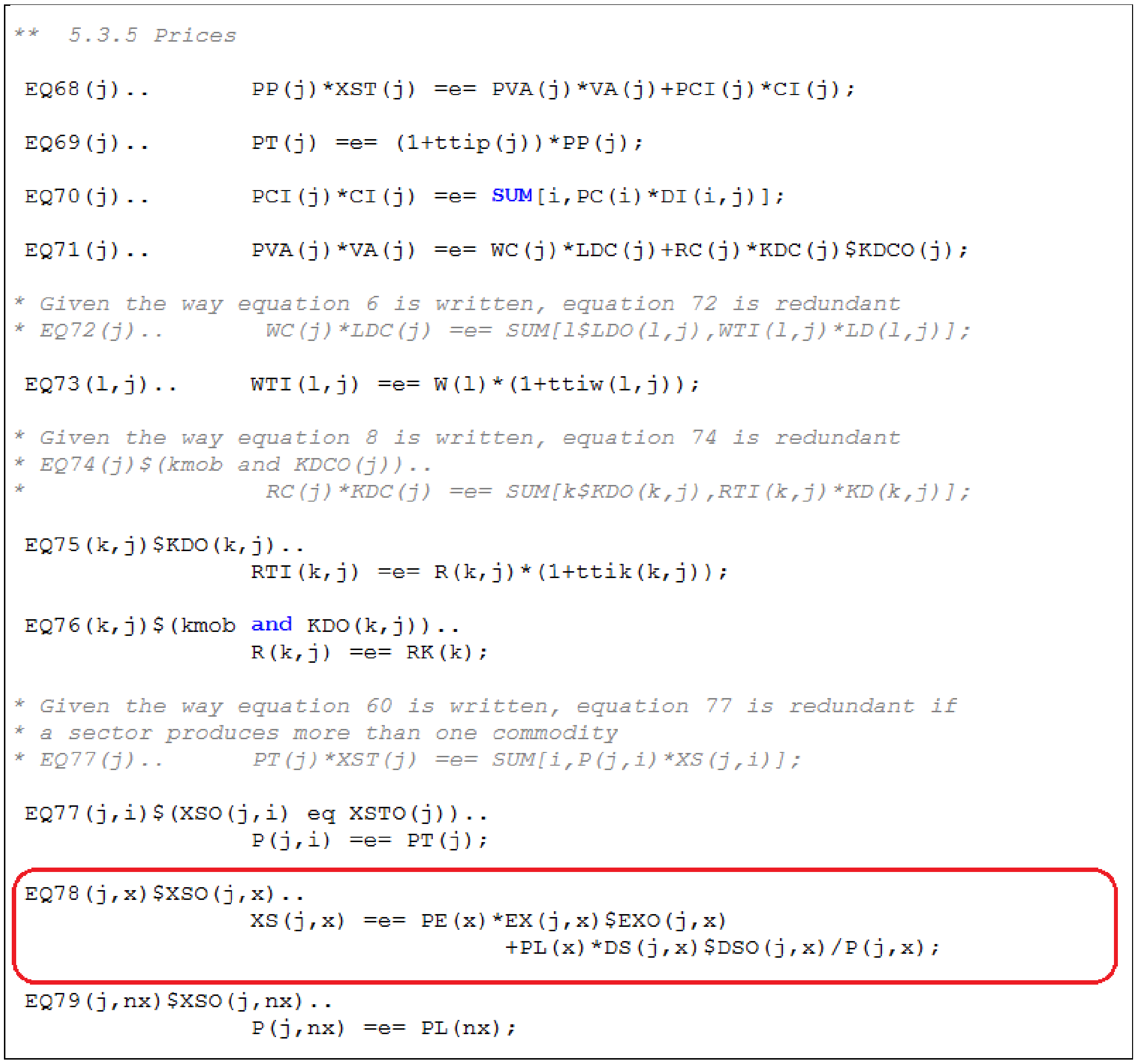
Figure 124 : Checking on LEON after a simulation in example 19



```
---- VAR YROW              -INF   17772.191     +INF
---- VAR LEON              -INF     800.798     +INF

  YROW  Rest-of-the-world income
  LEON  Excess supply on the last market


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

The value of *LEON* is significant. We now need to review the different equations, especially the ones involving prices equal to one at benchmark.

Figure 125 : Price equations in example 19



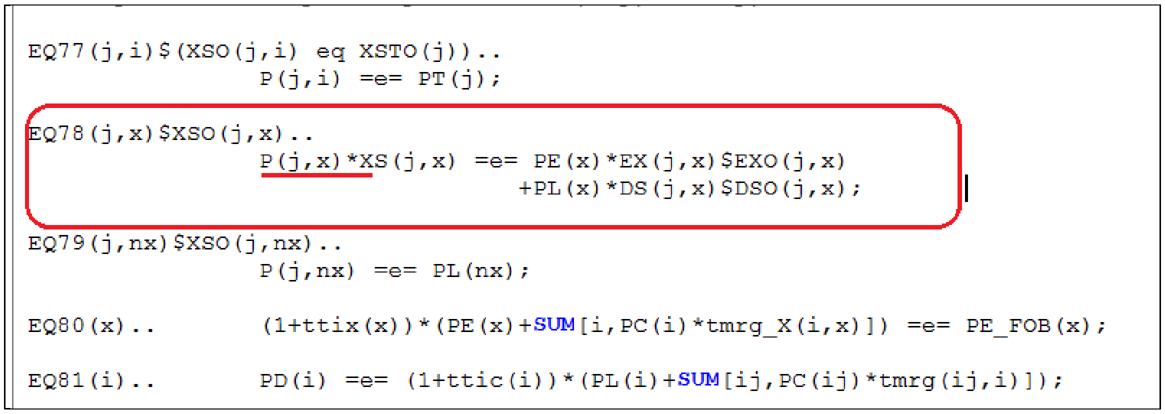
```
**  5.3.5 Prices

 EQ68(j)..        PP(j)*XST(j) =e= PVA(j)*VA(j)+PCI(j)*CI(j);

 EQ69(j)..        PT(j) =e= (1+ttip(j))*PP(j);

 EQ70(j)..        PCI(j)*CI(j) =e= SUM[i,PC(i)*DI(i,j)];

 EQ71(j)..        PVA(j)*VA(j) =e= WC(j)*LDC(j)+RC(j)*KDC(j)$KDCO(j);

* Given the way equation 6 is written, equation 72 is redundant
* EQ72(j)..        WC(j)*LDC(j) =e= SUM[l$LDO(l,j),WTI(l,j)*LD(l,j)];

 EQ73(l,j)..      WTI(l,j) =e= W(l)*(1+ttiw(l,j));

* Given the way equation 8 is written, equation 74 is redundant
* EQ74(j)$(kmob and KDCO(j))..
*                  RC(j)*KDC(j) =e= SUM[k$KDO(k,j),RTI(k,j)*KD(k,j)];

 EQ75(k,j)$KDO(k,j)..
                  RTI(k,j) =e= R(k,j)*(1+ttik(k,j));

 EQ76(k,j)$(kmob and KDO(k,j))..
                  R(k,j) =e= RK(k);

* Given the way equation 60 is written, equation 77 is redundant if
* a sector produces more than one commodity
* EQ77(j)..       PT(j)*XST(j) =e= SUM[i,P(j,i)*XS(j,i)];

 EQ77(j,i)$(XSO(j,i) eq XSTO(j))..
                  P(j,i) =e= PT(j);

 EQ78(j,x)$XSO(j,x)..
                  XS(j,x) =e= PE(x)*EX(j,x)$EXO(j,x)
                                   +PL(x)*DS(j,x)$DSO(j,x)/P(j,x);

 EQ79(j,nx)$XSO(j,nx)..
                  P(j,nx) =e= PL(nx);
```

If we take a closer look at equation 78, we can see that something is not correct in the formulation of the equation. This equation determines the basic price *P(j,x)* obtained by industry *j* for exportable commodity *x*, as a weighted sum of its basic price on the export market and its basic price on the domestic market.

We can tell that equation 78 is written incorrectly because if we look at the right hand side of the equation, we can see that there are missing brackets. In this case, *P(j,x)* is only dividing the second term of the sum (as GAMS respects the mathematical operator priority).

Thus, either we add brackets or we write *P(j,x)* on the left side of equation 78.

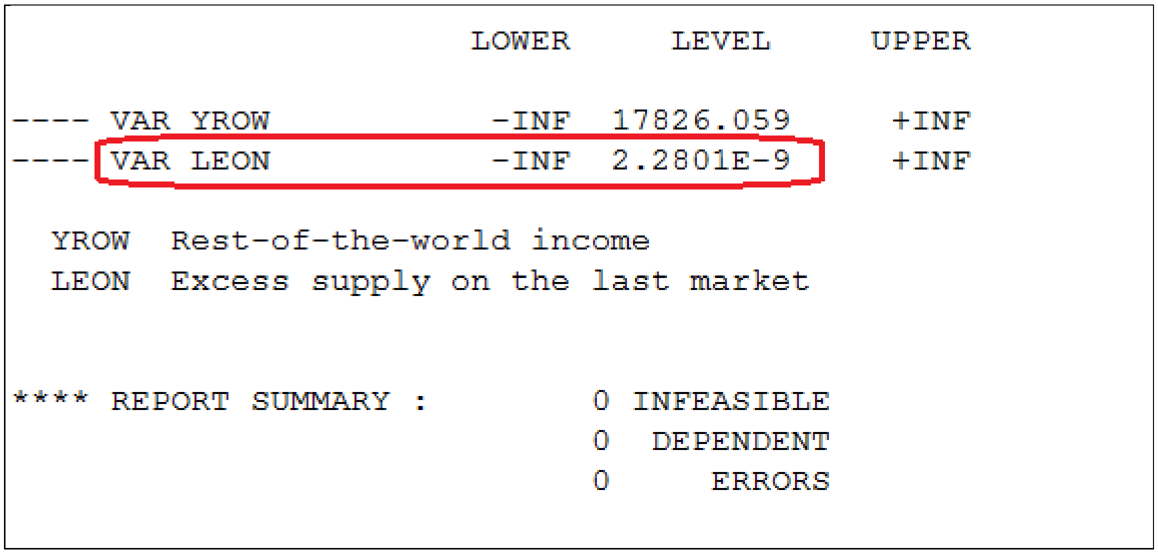
Figure 126 : Correction of equation 78 in example 19



```
EQ77(j,i)$(XSO(j,i) eq XSTO(j))..
                P(j,i) =e= PT(j);

EQ78(j,x)$XSO(j,x)..
                P(j,x)*XS(j,x) =e= PE(x)*EX(j,x)$EXO(j,x)
                                  +PL(x)*DS(j,x)$DSO(j,x);

EQ79(j,nx)$XSO(j,nx)..
                P(j,nx) =e= PL(nx);

EQ80(x)..       (1+ttix(x))*(PE(x)+SUM[i,PC(i)*tmrg_X(i,x)]) =e= PE_FOB(x);

EQ81(i)..       PD(i) =e= (1+ttic(i))*(PL(i)+SUM[ij,PC(ij)*tmrg(ij,i)]);
```

Then we can run example 19 again and check the value of *LEON*

Figure 127 : Checking of LEON after correction of example 19



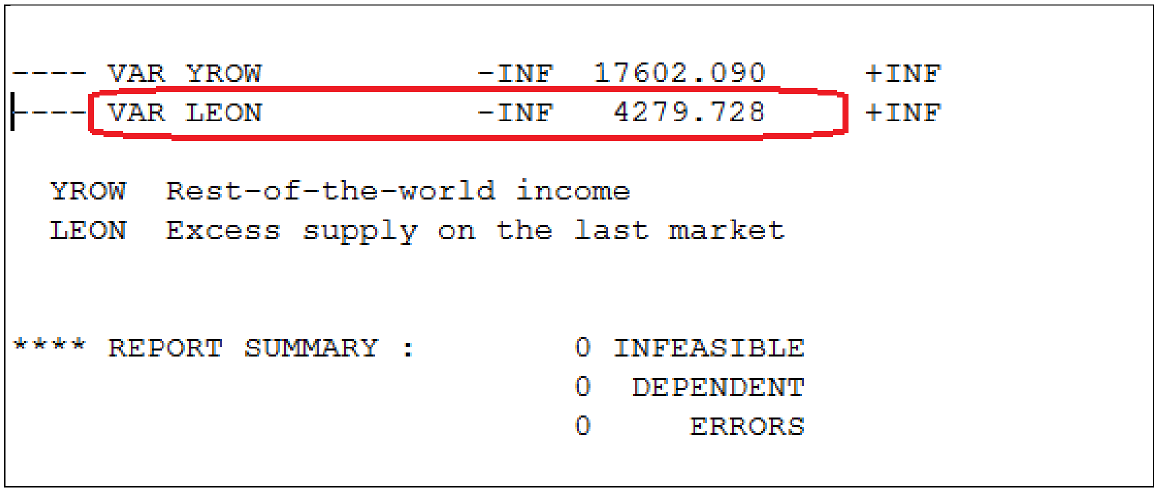
```
                           LOWER     LEVEL     UPPER

---- VAR YROW               -INF   17826.059    +INF
---- VAR LEON               -INF   2.2801E-9    +INF

  YROW  Rest-of-the-world income
  LEON  Excess supply on the last market


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

In conclusion, a missing pair of brackets, or brackets that close or open at the wrong place, can lead to an incorrect specification of the model.

Example 20:

Figure 128 reproduces the value of *LEON*, after an introduction of a 20% increase of labour supply.

Figure 128 : Value of LEON in the listing file in example 20

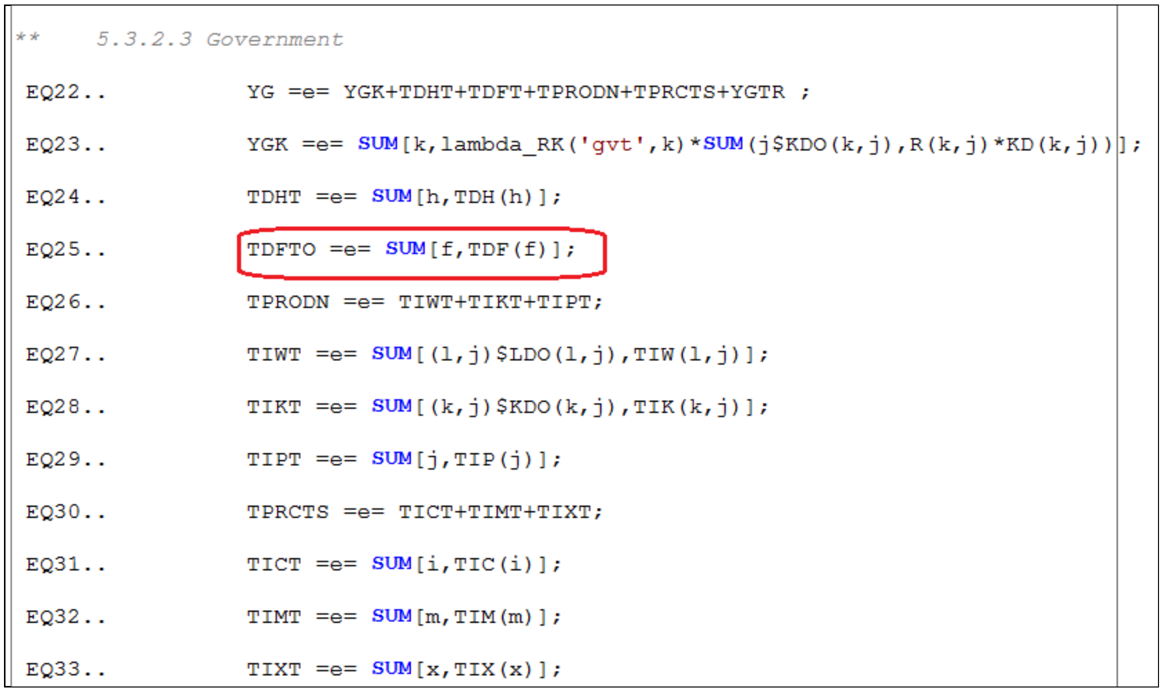


```
---- VAR YROW            -INF   17602.090      +INF
---- VAR LEON            -INF    4279.728      +INF

  YROW  Rest-of-the-world income
  LEON  Excess supply on the last market


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

As for the two previous examples, we will start reviewing the equations, focusing on the ones including prices whose values are equal to one at the benchmark.

This time, we do not find any mistakes with this particular type of equation. Consequently, we need to review all the equations.

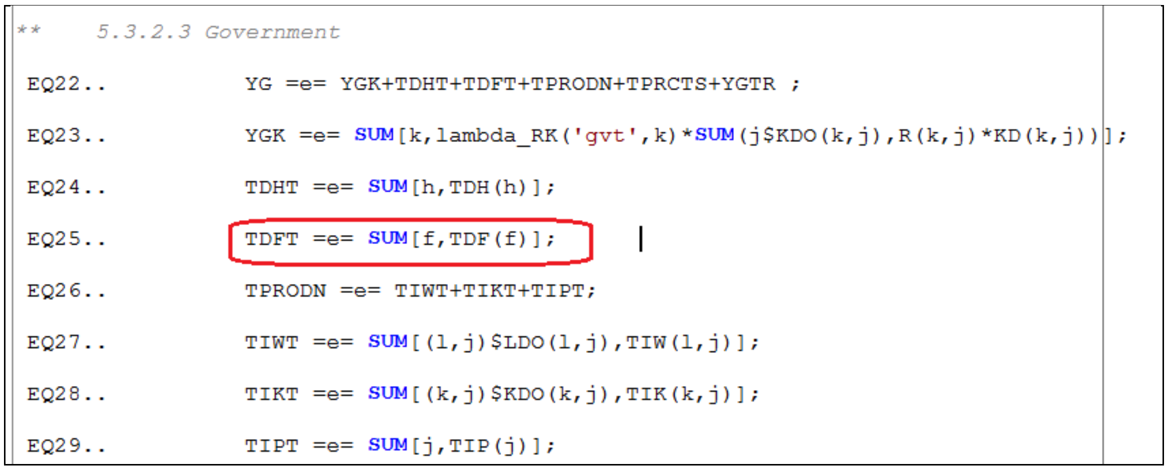
Figure 129 : Equations in example 20



```
**     5.3.2.3 Government

 EQ22..              YG =e= YGK+TDHT+TDFT+TPRODN+TPRCTS+YGTR ;

 EQ23..              YGK =e= SUM[k,lambda_RK('gvt',k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ24..              TDHT =e= SUM[h,TDH(h)];

 EQ25..              TDFTO =e= SUM[f,TDF(f)];

 EQ26..              TPRODN =e= TIWT+TIKT+TIPT;

 EQ27..              TIWT =e= SUM[(l,j)$LDO(l,j),TIW(l,j)];

 EQ28..              TIKT =e= SUM[(k,j)$KDO(k,j),TIK(k,j)];

 EQ29..              TIPT =e= SUM[j,TIP(j)];

 EQ30..              TPRCTS =e= TICT+TIMT+TIXT;

 EQ31..              TICT =e= SUM[i,TIC(i)];

 EQ32..              TIMT =e= SUM[m,TIM(m)];

 EQ33..              TIXT =e= SUM[x,TIX(x)];
```

When reviewing all of the equations, we find a problem with equation 25. This equation defines total direct taxes paid by firms. The problem here is that the left hand side of the equation, *TDFTO*, refers to the value of the variable at its benchmark value, but *TDFTO* will not change when a shock is introduced. Here, it should be the variable *TDFT* instead of the parameter *TDFTO*.

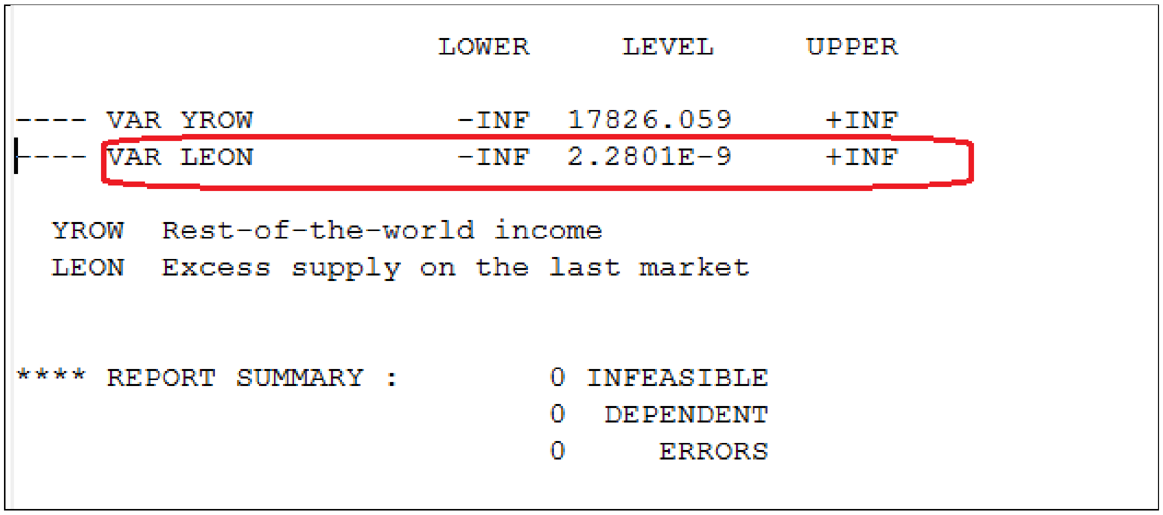
Figure 130 : Correction of equation 25 in example 20



```
**     5.3.2.3 Government

 EQ22..              YG =e= YGK+TDHT+TDFT+TPRODN+TPRCTS+YGTR ;

 EQ23..              YGK =e= SUM[k,lambda_RK('gvt',k)*SUM(j$KDO(k,j),R(k,j)*KD(k,j))];

 EQ24..              TDHT =e= SUM[h,TDH(h)];

 EQ25..              TDFT =e= SUM[f,TDF(f)];

 EQ26..              TPRODN =e= TIWT+TIKT+TIPT;

 EQ27..              TIWT =e= SUM[(l,j)$LDO(l,j),TIW(l,j)];

 EQ28..              TIKT =e= SUM[(k,j)$KDO(k,j),TIK(k,j)];

 EQ29..              TIPT =e= SUM[j,TIP(j)];
```

We can then check the value of *LEON* after the simulation:

Figure 131 : Check on variable LEON after correction in example 20



```
                              LOWER      LEVEL      UPPER

---- VAR YROW                  -INF   17826.059      +INF
---- VAR LEON                  -INF   2.2801E-9      +INF

  YROW  Rest-of-the-world income
  LEON  Excess supply on the last market


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

All in all, this last type of error can take a long time to find, especially if the model has several equations. As previously mentioned, the examples we presented here are not in any way a comprehensive list of all possible scenarios resulting from this type of error.

To avoid this type of error, we strongly recommend that the modeller starts from a model that works (i.e. with a very small control variable when running a simulation), and make one change at a time. After each addition, run a simulation and look for the value of the control variable. This way if the value of this variable is not correct, the modeller will know that it comes from his new addition.

References:

Decaluwé, B., A. Lemelin, H.Maisonnave and V.Robichaud (2010) The pep standard computable general equilibrium model single-country, static version pep-1-1, second revised edition october 2009 (minor corrections, july 2010) http://www.pep-net.org/programs/mpia/pep-standard-cge-models/pep-1-1-single-country-static-version/

McCarl, B.A. A.Meeraus, P.van der Eijk, M. Bussieck, S. Dirkse and P. Steacy (2009) McCarl Expanded GAMS User Guide, Version 23.3, GAMS Development Corporation, Washington DC, USA

Robichaud, V., A. Lemelin, H.Maisonnave and B.Decaluwé (2011), PEP1-1, a user guide, AGRODEP and Poverty and Economic Policy Research network

Rosenthal, R.E (2008), GAMS, A User's Guide, GAMS Development Corporation, Washington DC, USA

Table of figures:

Figure 1 : Process window of example 1 3
Figure 2 : Process window of example 1 (2) 4
Figure 3 : Process window of example 1 (3) 4
Figure 4 : GAMS file error 1 5
Figure 5 : Checking for parenthesis (F8 shortcut) 6
Figure 6 : Checking for parenthesis (2) 6
Figure 7 : Brackets in example 1: 7
Figure 8 : Adding the missing bracket 7
Figure 9 : Running a GAMS code: 8
Figure 10 : Process window of example 1 9
Figure 11 : Process window of example 2 10
Figure 12 : Location of the error in example 2 11
Figure 13 : Correction of example 2 11
Figure 14 : Process window of example 2 12
Figure 15 : Process window for example 3 13
Figure 16 : Error 3 in the GAMS file 13
Figure 17 : Set declaration: 14
Figure 18 : Correction of example 3: 15
Figure 19 : Process window of example 3 15
Figure 20 : Process window of example 4 16
Figure 21 : Error in example 4 16
Figure 22 : Set declaration in PEP 1-1 17
Figure 23 : Correction of error 4 17
Figure 24 : Process window after correction of example 4 18
Figure 25 : Example 5 19
Figure 26 : Location of the error in the GAMS file (example 5) 19
Figure 27 : Social Accounting Matrix used in PEP-1-1 20
Figure 28 : Correction of example 5 21
Figure 29 : Checking example 5 solution 21
Figure 30 : Process window of example 6 22
Figure 31 : Location of error 6 in the GAMS code 22
Figure 32 : Assignment of a value to a variable using the SAM 23
Figure 33 : Assignment of a value to a variable by giving it a value directly 23
Figure 34 : Assignment of a value to a variable by calibrating it 23
Figure 35 : Looking for PLO (i) in the GAMS code 24
Figure 36 : Correction of example 6 24
Figure 37 : Verify example 6 solution 25
Figure 38 : Process window of example 7 26
Figure 39 : Listing file of example 7 27

Figure 40 : Assigning the value of KDO(k,j) in example 7 27
Figure 41 : First half of the SAM used for PEP 1-1.......... 28
Figure 42 : Example 7 corrected.......... 28
Figure 43 : Process window of example 7 after correction.......... 29
Figure 44 : Process window of example 8 30
Figure 44 : Closure rules in GAMS in example 8.......... 31
Figure 46 : Correction of example 8 32
Figure 47 : Process window of example 8 after correction.......... 32
Figure 48 : Process window of example 9 33
Figure 49 : Listing file for example 9 (1) 34
Figure 50 : Listing file for example 9 (2) 34
Figure 51 : Listing file for example 9 (3) 35
Figure 52 : Listing file for example 9 (4) 35
Figure 53 : Display of XSTO(j) in example 9 35
Figure 54 : Listing window of example 9 36
Figure 55 : Display of XSTO in the listing file (example 9) 36
Figure 56 : Display of XSTO(j) and j in example 9 37
Figure 57 : Display of XSTO and j in the listing file (example 9) 37
Figure 58 : Set definition in example 9.......... 38
Figure 59 : Correction of example 9 38
Figure 59 : Process window after correction of example 9.......... 39
Figure 61 : Process window of example 10 40
Figure 62 : Closure rules in GAMS in example 10.......... 41
Figure 63 : Equilibrium equations in example 10 42
Figure 64 : Set definition in PEP 1-1 : 43
Figure 65 : Declaration of equation 89 in example 10 43
Figure 66 : Declaration of equation 89 after correction in example 10.......... 43
Figure 67 : Equation 89 in example 10 44
Figure 68 : Equation 89 after correction in example 10.......... 44
Figure 69 : Process window after correction of example 10.......... 45
Figure 70 : Process window of example 11 47
Figure 71 : Listing file of example 11.......... 48
Figure 72 : Listing file of example 11 (2).......... 49
Figure 73 : Looking for an Infes in the listing file 49
Figure 74 : Looking for infes in example 11.......... 50
Figure 75 : Listing file of example 11.......... 51
Figure 76 : Equation 1 in example 11 51
Figure 77 : Correction of example 11 52
Figure 78 : Process window of example 11 52
Figure 79 : Process window of example 12 53
Figure 80 : Infes in example 11.......... 53
Figure 81 : Writing of equation 55 in example 12.......... 54

Figure 82 : Calibration of the parameter gamma_INV(i) in example 12 54
Figure 83 : Correction of the calibration of parameter gamma_INV(i) : 55
Figure 84 : Process window of example 12 after correction 55
Figure 85 : Process window of example 13 56
Figure 86 : Listing file for example 13 57
Figure 86 : Equations 45 and 46 in example 13 57
Figure 88 : Variable initialization in example 13 58
Figure 89 : Initialization of YROW in example 13 61
Figure 90 : Process window of example 13 after correction 61
Figure 91 : Process window of example 14 62
Figure 92 : Listing file in example 14 62
Figure 93 : Equations in example 14 63
Figure 94 : Initialization in example 14 64
Figure 95 : Assignment of variables in example 14 65
Figure 96 : partial reproduction of the SAM: 66
Figure 97 : Assignment of SROWO after correction in example 14 67
Figure 98 : Process window after correction in example 14 68
Figure 99 : Process window in example 15 69
Figure 100 : Infes found in the listing file example 15 69
Figure 101 : Initialization of YG in example 15 70
Figure 102 : Correction of initial value for YG in example 15 71
Figure 103 : Process window of example 15 after correction 71
Figure 104 : Process window of example 16 72
Figure 105 : Listing file in example 16 72
Figure 106 : SAM of PEP 1-1 73
Figure 107 : Initialization of INV(i) in example 16 74
Figure 108 : Assignment of INVO(i) in example 16 75
Figure 109 : Display of INV(i) in example 16 75
Figure 110 : Display of INV(i) and INVO(i) in example 16 76
Figure 111 : Obtaining volumes from values in example 16 76
Figure 112 : Process window of example 16 after correction 77
Figure 113 : Process window for example 17 78
Figure 114 : Infes in example 17 78
Figure 115 : Equation 63 in the GAMS code in example 17 79
Figure 116 : Correction of equation 63 in example 17 80
Figure 117 : Process window after correction of example 17 80
Figure 118 : Increase of labour supply by 20% in example 18 82
Figure 118 : Process window in example 18 83
Figure 120 : Check on variable LEON after simulation in example 18 83
Figure 121 : Reviewing equations in example 18 84
Figure 122 : Correction of example 18 85
Figure 123 : Check on variable LEON after correction in example 18 85

Figure 124 : Checking on LEON after a simulation in example 19.. 86
Figure 125 : Price equations in example 19 87
Figure 126 : Correction of equation 78 in example 19... 88
Figure 127 : Checking of LEON after correction of example 19 88
Figure 128 : Value of LEON in the listing file in example 20... 89
Figure 129 : Equations in example 20 90
Figure 130 : Correction of equation 25 in example 20... 90
Figure 131 : Check on variable LEON after correction in example 20.. 91

Table of contents

1. Compilation errors 3

Example 1: 3

Example 2: 10

Example 3: 13

Example 4: 16

Example 5: 19

Example 6: 22

2. Execution errors: 26

Example 7: 26

Example 8: 30

Example 9: 33

Example 10: 40

3. Calibration errors 46

Example 11: 47

Example 12: 53

Example 13: 56

Example 14: 62

Example 15: 69

Example 16: 72

Example 17: 78

4. Specification errors 81

Example 18: 82

Example 19: 86

Example 20: 89