

Impact Analysis of CAP Reform on the Main Agricultural Commodities

Report IV

AGMEMOD – GSE Interface Manual

Author: Wietse Dol
Editors: Robert M'barek and Lubica Bartova



EUR 22940 EN/4 - 2008

The mission of the IPTS is to provide customer-driven support to the EU policy-making process by researching science-based responses to policy challenges that have both a socio-economic as well as a scientific/technological dimension.

European Commission
Directorate-General Joint Research Centre
Institute for Prospective Technological Studies

Contact information

Address: Edificio Expo. c/ Inca Garcilaso, s/n. E-41092 Seville (Spain)

E-mail: jrc-ipts-secretariat@ec.europa.eu

Tel.: +34 954488318

Fax: +34 954488300

<http://ipts.jrc.ec.europa.eu>

<http://www.jrc.cec.eu.int>

Legal Notice

Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use which might be made of this publication.

A great deal of additional information on the European Union is available on the Internet. It can be accessed through the Europa server

<http://europa.eu.int>

JRC40457

EUR 22940 EN/4

ISSN 1018-5593

Luxembourg: Office for Official Publications of the European Communities

© European Communities, 2008

Reproduction, for non commercial purposes, is authorised provided the source is acknowledged

Printed in Spain

Impact Analysis of CAP Reform on the Main Agricultural Commodities

Report IV

AGMEMOD – GSE Interface Manual

Author:

Wietse Dol

Agricultural Economics Research Institute, The
Netherlands

Editors:

Robert M'barek and Lubica Bartova

2008

■ Foreword

Quantitative models are important tools for analysing the impact of agricultural policies. One of the modelling approaches used to analyse the impact of the Common Agricultural Policy is AGMEMOD (AGricultural MEmber states MODelling), an econometric, dynamic, partial equilibrium, multi-country, multi-market model. AGMEMOD models provide extensive details of the agricultural sector in individual EU Member States including the new members Bulgaria and Romania, and the EU as a whole.

A study was carried out from November 2005 until June 2007 by the AGMEMOD Partnership under the management of the Agricultural Economics Research Institute (LEI, the Netherlands), in cooperation with the European Commission's Joint Research Centre - Institute for Prospective Technological Studies (JRC-IPTS). The aim was to generate projections for the main agricultural commodity markets for each year from 2005 until 2015.

Detailed documentation on the AGMEMOD modelling approach, along with the outcome of the study, is published in five reports in the JRC-IPTS Scientific and Technical Report Series (Box 1) under the heading "Impact analysis of CAP reform on the main agricultural commodities".

Box 1 Impact analysis of CAP reform on the main agricultural commodities

Report I *AGMEMOD – Summary Report*

This report presents the projections of agricultural commodity markets under the baseline, further CAP reform, enlargement scenarios and exchange rate change sensitivity analyses for the aggregates EU-10, EU-15, EU-25 and EU-27. It summarises the characteristics of the modelling tool used, focusing in particular on the features implemented in this study, and addresses issues that need further attention. (<http://www.jrc.es/publications>)

Report II *AGMEMOD – Member States Results*

This report outlines the results of the baseline projections of agricultural commodity markets, further CAP reform scenario impact analyses and exchange rate change sensitivity analyses for individual EU-27 Member States except Malta and Cyprus. For Bulgaria and Romania enlargement and non-enlargement scenarios are analysed. (<http://www.jrc.es/publications>)

Report III *AGMEMOD – Model Description*

This report describes the modelling techniques used by the AGMEMOD Partnership, with the emphasis on new commodities modelled and policy modelling approaches. (<http://www.jrc.es/publications>)

Report IV *AGMEMOD – GSE Interface Manual*

The Manual gives an overview of the GAMS Simulation Environment (GSE) interface and its application with the AGMEMOD model. (<http://www.jrc.es/publications>)

Report V *Commodity Modelling in an Enlarged Europe – November 2006 Workshop Proceedings*

These proceedings consist of presentations and conclusions of a workshop held in November 2006. The presentation of outcomes of the other models such as FAPRI, ESIM, AGLINK and CAPSIM are included in addition to the AGMEMOD approach. (<http://www.jrc.es/publications>)

Experience has shown that building models and writing the related software can give rise to considerable problems. If software is poorly developed, numerous problems can emerge with the result that it can easily become unreadable for less familiar users. After several revisions and extensions of the initial version this can even be true of the person who wrote the original software. Badly structured and poorly documented software allows very little flexibility and extendibility and cannot be passed on to other developers.

The IPTS has sought a way of making the AGMEMOD model more accessible for end-users and researchers. The GSE interface, which stands for GAMS Simulation Environment, was considered the best tool to make the AGMEMOD model accessible with maximum efficiency and sustainability. One of the main advantages of GSE is that it keeps the original GAMS code of the model intact. The user interface functions are extended separately from development of the model. Not only will project planning become easier, but the quality of the model will also be improved.

We wish to acknowledge the work done by Wietse Dol and Myrna van Leeuwen, researchers at the Agricultural Economics Research Institute (LEI, The Netherlands).

■ Executive summary

This report gives an overview of the GAMS Simulation Environment (GSE) interface and an application of GSE to the AGMEMOD model (AGricultural MEmber states MODelling), addressing directly the needs of model-builders and users in an application-oriented and practical way.

AGMEMOD is an econometric, dynamic, partial equilibrium, multi-country, multi-market modelling system, which provides detailed information on the agricultural sector in each EU Member State and the EU as a whole.

The present report is part of five reports published in the JRC-IPTS technical report series under the heading "Impact analysis of CAP reform on the main agricultural commodities".

First, the report provides a detailed overview on GSE, a professional user interface, which makes models, written in GAMS programming language more accessible and gives greater insight into the relations between input and output.

In this context GSE allows for:

- More transparent (model) links;
- Easier access to and wider use of the results from the model ensuring that corporate knowledge will improve the continuity of the AGMEMOD model;
- A DataViewer to review and analyse data, including a Geographical Information System;
- A version control tool;
- A scenario analyser to compare, print and depict outcomes;
- A link between AGMEMOD and organisations instead of persons.

Model building with GSE provides a good structure, as given standard protocol is employed that can be used for many models. It increases efficiency, transferability, reproducibility, and reviewability.

Then, three quick reference guides are presented for model-builders, -users and new versions. However, these quick reference guides are not a substitute for the manual (see http://www.lei.dlo.nl/nacquit/products/gse/manual/Table_of_Contents.htm).

The first is a reference guide for model-builders and explains how a GAMS model is imported into GSE using the Import2GSE wizard. It shows the basic steps needed to understand GSE and how to get a first (simple) model running under GSE (the TRANSPORT model).

The second is especially for model-users and explains briefly some of the possibilities offered by GSE to change model parameters and run and compare scenarios. It starts with explanations on how to install GSE and on how to install a model in GSE. Furthermore, it shows how to establish and analyse scenarios. A specific data viewer allows viewing, editing and comparing input and outputting values. Output can be generated on paper, HTML, graphics, geographical information system, tables etc.

The third reference guide explains how to make new versions of the model and install them into GSE.

For implementing the model in GAMS, a software within GSE called Gtree is recommended as it organises the model in a tree structure and helps finding errors, improving and maintaining the model. A default tree structure for (almost) all models is recommended, consisting of 'Sets and Elements.set', 'Declarations.gms', 'Import data.inp', 'Model.gms', and 'Output.gms'. Gtree itself hosts several functions like the GDX (GAMS Data eXchange) for data viewing and scenario analysis.

The last chapter of this report gives details on how GSE has been implemented into the AGMEMOD modelling tool.

■ Table of contents

■ Foreword.....	4
■ Executive summary.....	6
■ Table of contents	8
■ List of Tables	9
■ List of Figures.....	9
■ Acronyms	12
1. Overview of the GAMS Simulation Environment (GSE)	13
2. Model-building with GSE tools	14
2.1. Model structure	14
2.2. GAMS Data eXchange (GDX) files	15
2.3. Scenario analysis (GDX DataViewer).....	16
2.4. Model versions (GSE).....	18
3. Quick reference guide to GSE for model-builders	19
3.1. Introduction	19
3.2. Installation of GSE	19
3.3. Your first model in GSE.....	19
3.3.1. Step 1: starting Import2GSE.....	20
3.3.2. Step 2: inserting GSE tags.....	23
3.3.3. Step 3: checking the GSE tags.....	28
3.3.4. Step 4: importing a version	30
3.3.5. Step 5: running additional scenarios	34
3.3.6. Step 6: building a distribution list.....	35
3.3.7. Step 7: You want more	36
4. Quick reference guide to GSE for model-users	38
4.1. Introduction	38
4.2. Installing a model.....	38
4.3. Starting the TRANSPORT model.....	39
4.4. Log on to GSE	41
4.5. GSE main window	43
4.6. Model.....	43
4.7. Scenario.....	44
4.8. Scenario analysis.....	45
4.9. The GSE DataViewer/editor	46
4.10. Steps for creating a new scenario	52
5. Quick reference guide to GSE for new versions of the model	55
5.1. Introduction	55
5.2. Version or scenario?	55
5.3. A new version	55
6. Tips for GAMS models	56
7. GAMS tree	59
7.1. Introduction to Gtree.....	59
7.1.1. File menu.....	61
7.1.2. Actions menu	62
7.1.3. Options menu	64
7.1.4. Switches for Gtree	65
7.1.5. GAMS menu	67
7.1.6. Help menu	68
7.2. Functions of Gtree.....	68
7.2.1. Editor buttons.....	68
7.2.2. Reference file.....	74
7.2.3. Find text in tree	77
7.2.4. Search/replace text in files	79
7.2.5. LST tree.....	80
7.2.6. Additional programs	81
7.2.7. Restore files.....	82
7.2.8. Special Gtree tags	83
7.2.9. Gtree INI file	84

7.2.10.	Code templates	84
7.2.11.	Cleanup files.....	86
7.2.12.	Environment variables	87
7.2.13.	Syntax editor settings	87
7.2.14.	Customise editor options.....	88
7.2.15.	Trees colours.....	88
7.2.16.	GAMS options	89
7.2.17.	GAMS library.....	89
7.2.18.	Solvers.....	91
7.2.19.	GAMS errors.....	92
7.2.20.	Breakpoints in GAMS.....	93
7.2.21.	GAMS dollar commands	93
7.2.22.	Print preview.....	94
7.2.23.	Gtree and GSE	94
7.2.24.	New file in tree template.....	102
7.2.25.	Keyboard shortcuts for Gtree.....	103
7.2.26.	DataViewer	104
7.2.27.	Start parameters.....	106
8.	Implementation of GSE in AGMEMOD.....	108
9.	Documentation.....	110
10.	References	111
11.	Appendices	112
	Appendix 1: the Transport model with GSE tags.....	112
	Appendix 2: GDXOUTPUT versus OUTPUT tags.....	113

■ List of Tables

Table 3.1	Programs used within GSE.....	19
Table 7.1	Choices in the file menu.....	61
Table 7.2	Choices in the actions menu	63
Table 7.3	Choices in the options menu.....	64
Table 7.4	Description of the switches for Gtree	65
Table 7.5	Choices in the GAMS menu.....	67
Table 7.6	Choices in the help menu.....	68
Table 7.7	Descriptions of the editor buttons.....	68
Table 7.8	Description of a reference file sets	74

■ List of Figures

Figure 2.1	Model modules.....	15
Figure 2.2	Reference file selection	16
Figure 2.3	GDX Data Viewer.....	16
Figure 2.4	GDX Data Viewer – select data file.....	17
Figure 2.5	GDX Data Viewer – select GAMS source.....	17
Figure 2.6	GDX Data Viewer – compare GDX files	17
Figure 2.7	GDX Data Viewer – add button.....	18
Figure 3.1	Starting Import2GSE	20
Figure 3.2	Selecting a new model.....	20
Figure 3.3	Model import settings	21
Figure 3.4	Selecting the source path	21
Figure 3.5	Options after importing a model.....	22
Figure 3.6	Buttons after importing a model	23
Figure 3.7	Gtree with the transport model.....	23
Figure 3.8	Inserting a SET tag	24
Figure 3.9	TagEditor of Gtree.....	24
Figure 3.10	Result of a SET tag.....	25
Figure 3.11	INPUT tag options	25

Figure 3.12 Result of the INPUT tag.....	26
Figure 3.13 Creating output parameter x.....	27
Figure 3.14 Creating output parameter z.....	27
Figure 3.15 GDX tag.....	28
Figure 3.16 Checking tags.....	28
Figure 3.17 Forgetting to close an INPUT tag.....	29
Figure 3.18 Pressing Edit in Figure 3.17.....	30
Figure 3.19 Starting GSE.....	30
Figure 3.20 GSE window.....	31
Figure 3.21 Adding a new version of the model.....	31
Figure 3.22 GSE window after inserting a version of the model.....	32
Figure 3.23 Scenario tab sheet.....	32
Figure 3.24 Parameters.....	33
Figure 3.25 Distance in thousands of miles parameter.....	33
Figure 3.26 Rows and columns display.....	34
Figure 3.27 After running GAMS you can view the input and output.....	34
Figure 3.28 Creating a new scenario.....	35
Figure 3.29 Loading the Transport model.....	35
Figure 3.30 Transport model settings.....	36
Figure 3.31 Model distribution options.....	36
Figure 3.32 Distribution list has been created.....	37
Figure 4.1 Installing TRANSPORT.....	39
Figure 4.2 Selecting the installation directory.....	39
Figure 4.3 Creating a new directory.....	39
Figure 4.4 Registration of GSE.....	40
Figure 4.5 Internet registration of GSE.....	41
Figure 4.6 Registration code.....	41
Figure 4.7 Log on to GSE.....	42
Figure 4.8 GSE main window.....	43
Figure 4.9 Scenario tab sheet.....	45
Figure 4.10 Comparing scenarios.....	46
Figure 4.11 Screen with input and output parameters.....	47
Figure 4.12 GSE DataViewer.....	48
Figure 4.13 Selecting elements.....	49
Figure 4.14 Table view.....	50
Figure 4.15 Removing zeros and empty rows/columns.....	51
Figure 4.16 Clicking the right mouse button.....	52
Figure 4.17 Creating a scenario.....	53
Figure 4.18 The new scenario ready to be used.....	54
Figure 7.1 GAMS tree.....	59
Figure 7.2 GAMS tree and settings.....	60
Figure 7.3 File menu.....	61
Figure 7.4 Actions menu.....	62
Figure 7.5 Options menu.....	64
Figure 7.6 Switches for Gtree.....	65
Figure 7.7 GAMS menu.....	67
Figure 7.8 Selection of the programs menu.....	68
Figure 7.9 Help menu.....	68
Figure 7.10 Editor buttons.....	68
Figure 7.11 Pop-up menu.....	70
Figure 7.12 Select as environment file.....	71
Figure 7.13 Print tree.....	71
Figure 7.14 Print all files.....	72
Figure 7.15 Pop-up menu in the editor.....	72
Figure 7.16 Locked file.....	73
Figure 7.17 GAMS comments.....	73
Figure 7.18 Other possibilities for the GAMS comments.....	74
Figure 7.19 Reference tab sheet.....	75
Figure 7.20 Where the set is used in your model.....	75
Figure 7.21 Select names for code completion.....	76

Figure 7.22 Selection of a word for insertion into the code	76
Figure 7.23 Show elements/(sub)set grid	77
Figure 7.24 Search a node.....	78
Figure 7.25 Tree where the file is used	79
Figure 7.26 Find text in files	79
Figure 7.27 Results of the “Find text in file” function.....	80
Figure 7.28 LST tree.....	80
Figure 7.29 Additional programs menu	81
Figure 7.30 Restore backup file.....	82
Figure 7.31 Special Gtree tag	83
Figure 7.32 Special Gtree tag	83
Figure 7.33 GAMS manuals	84
Figure 7.34 Code template window.....	85
Figure 7.35 Cleanup extension.....	86
Figure 7.36 Environmental variables.....	87
Figure 7.37 Syntax editor settings.....	87
Figure 7.38 Customise editor options.....	88
Figure 7.39 Colours in the trees.....	88
Figure 7.40 GAMS options.....	89
Figure 7.41 GAMS model library.....	90
Figure 7.42 GAMS solvers	91
Figure 7.43 GAMS errors.....	92
Figure 7.44 Find GAMS error.....	92
Figure 7.45 Breakpoint in GAMS.....	93
Figure 7.46 GAMS dollar commands.....	93
Figure 7.47 Print preview.....	94
Figure 7.48 Model selection.....	95
Figure 7.49 Selection of the definitions.....	96
Figure 7.50 GSE tag selection.....	96
Figure 7.51 Set tag	97
Figure 7.52 Selection of the parameters.....	97
Figure 7.53 Input tag.....	98
Figure 7.54 Input tag options.....	98
Figure 7.55 Updated model tree.....	99
Figure 7.56 Selection of the GDXOUTPUT tag.....	100
Figure 7.57 GDXOUTPUT reference file of model parameters and variables	100
Figure 7.58 Selection of a parameter/variable.....	101
Figure 7.59 List of parameters/variables	101
Figure 7.60 DataViewer selection.....	104
Figure 7.61 DataViewer as a stand-alone tool.....	104
Figure 7.62 Compare GDX files.....	105
Figure 7.63 CSV/ASCII files.....	107
Figure 8.1 GSE concept in relation to AGMEMOD.....	108
Figure 8.2 Structure of AGMEMOD in the GAMS tree.....	109
Figure 11.1 Inserting OUTPUT tags.....	115

■ Acronyms

AGMEMOD	AGricultural MEMber states MODelling
CAP	Common Agricultural Policy
CEECs	Central and Eastern European Countries
CNDP	Complementary National Direct Payments (top-ups)
EU-10	8 EU Member States of 2004 Enlargement, Malta and Cyprus not included
EU-15	15 EU Member States before 2004 Enlargement
EU-25	23 EU Member States after 2004 Enlargement, Malta and Cyprus not included
EU-27	25 EU Member States after 2007 Enlargement, Malta and Cyprus not included
FAPRI	Food and Agricultural Policy Research Institute, USA
GAMS	General Algebraic Modelling System
GDP	Gross Domestic Product
GSE	GAMS Simulation Environment
JRC-IPTS	Joint Research Centre - Institute for Prospective Technological Studies (Spain)
OECD	Organisation for Economic Co-operation and Development
PSE	Producer Support Estimate
SAPS	Single Area Payment Scheme
SFP	Single Farm Payment
USD	U.S. Dollar
WTO	World Trade Organisation

1. Overview of the GAMS Simulation Environment (GSE)

The GAMS Simulation Environment (GSE) is a professional user interface which makes the GAMS model more accessible and gives greater insight into the relations between input and output. When used for the AGMEMOD modelling tool (AGricultural MEMber states MODelling) it allows:

- More transparent (model) links;
- Easier access to and wider use of the results from the model ensuring that corporate knowledge will improve the continuity of the AGMEMOD model;
- A DataViewer to review and analyse data, including a Geographical Information System;
- A version control tool;
- A scenario analyser to compare, print and depict outcomes;
- A link between AGMEMOD and organisations instead of persons.

Simulation models like AGMEMOD tend to change very rapidly during their lifetime. New versions and new scenarios are developed for each new project, which could endanger the consistency between the conceptual model and the actual computer model. Therefore, both model-builders and IT-scientists have thought about the demands on modern model-building and how it is used in applied research. The main demands are:

- Models should meet the requirements of customers and provide the outcomes in time;
- Models should be part of corporate knowledge (database experts, economists, ICT staff, etc.);
- Researchers other than model-builders must be able to use the model for their research project;
- Results from models should be reliable and their set-up should be clear;
- Models should be flexible to meet the requirements of various research projects, making different versions of the models possible;
- Results from models should be reproducible, both from a scientific point of view and future demands from customers;
- Peers should review models in order to enhance their overall quality;
- Models should be built in such a way that they can be easily connected to other models.

Most of these qualities also apply to the work of the AGMEMOD partnership on development of a projection and simulation tool for the agricultural sector in the EU and its Member States. The modelling teams wish to be able to change the model whenever new policy reforms make this necessary. This implies that several people need to know how the model works, what its assumptions are, how to use it to perform scenario analysis, etc. This means that institutions need to invest in people and in the model, which will only be useful if there are ways to turn the knowledge from the model into corporate (shared) knowledge. However, without good protocols and tools, conversion of knowledge into corporate knowledge is costly and impossible. GSE can be seen as an attempt to “corporate” the knowledge from the AGMEMOD model, by introducing a general approach on how to build GAMS models and user interfaces. Model knowledge should be specified in a mathematical form since this will lead to:

- More general and extendible model structures;
- Higher all-round quality of projections;
- Better understanding of the model for peers and colleagues.

As the user-interface functions are extended separately from development of the model, model-builders can spend more time working on the quality of the model. This means not only much better project planning but also a considerable improvement in the quality of the model. In this way, a package such as GSE will consolidate the AGMEMOD partnership.

2. Model-building with GSE tools

This guide tries to explain that there are several levels of implementing GSE with old and new models. Several examples will be shown to clarify the possibilities opened up by GSE.

2.1. Model structure

Any good model starts with a good structure, i.e. a standard protocol is recommended that can be used for many models: *“by using a standard way of making generic and well-documented models, both the efficiency, in terms of reusability and exchangeability, the creativity and the quality, in terms of attaining scientific and technical standards, of model-making can be increased greatly”*.

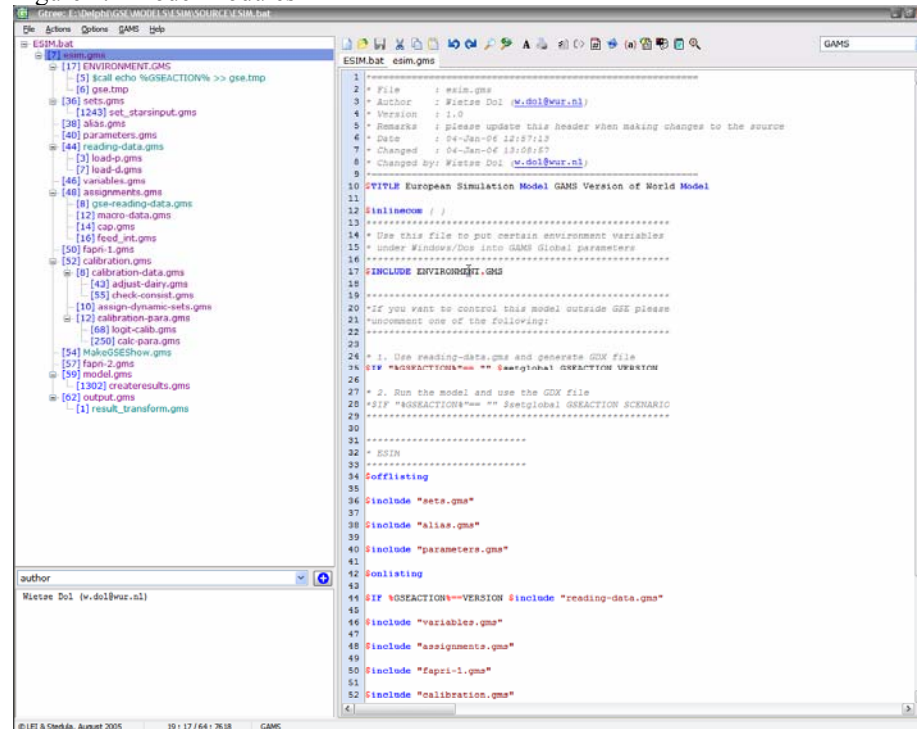
More specifically, this means:

- **Efficiency:** Using a standard modelling procedure and standardised documentation allows easy reuse of the model, greatly reducing the time and money invested when new research questions come up. Also, by working in a standardised way, use and exchanges of modules between model-developers and model-users are increased, which reduces the chance that readily available and reusable modules are re-developed;
- **Transferability:** Creating corporate knowledge by standardised documentation makes it easier to pass on the model to other persons;
- **Reproducibility:** Good design and description of the model-building and actual use of the model in research projects makes reproduction possible, which is scientifically necessary and also increasingly demanded by customers;
- **Reviewability:** Standardised documentation allows easy scientific and technical peer review. Once a model has been reviewed and approved scientifically and technically, this increases trust in a certain model or module.

We recommend Gtree for implementing the model in GAMS. GAMSIDE can be used, but this program does not show the structure in your model. What started as fun became a serious undertaking. As part of GSE, we wrote a small Delphi program we called Gtree (GAMS tree). Gtree will show the structure of your GAMS code in a tree (the result of all the **\$include**, **\$batclude**, **\$libinclude** and **\$sysinclude** statements). By clicking on the tree you can see the contents of the files as well.

The Figure 2.1 shows a model that, instead of writing all the computer code in one file, decided to split the model into different modules (files with a \$include GAMS command). In this model it is immediately clear, for example, where you should look for the parameter definitions (in file parameters.gms), where input data are read (reading-data.gms), etc. Making and maintaining such a tree structure for your model is easy in Gtree and even functions like “find” and “replace” are not just in one file but throughout the whole tree. The structure will be a great help in finding errors and in improving and maintaining your model. Gtree has a host of functions, one of which is the GDX DataViewer.

Figure 2.1 Model modules



2.2. GAMS Data eXchange (GDX) files

GAMS is an extremely powerful programming language for model-builders. The clarity of the code along with the excellent solvers makes it easy to build, improve and extend your model. But GAMS does not meet all the requirements necessary for the entire model life-cycle. The first is that when you start changing your model, GAMS does not offer the possibility of a version control system. You have to do your own bookkeeping and make sure that you copied the right files. The second disadvantage is that GAMS does not make it easy for you to look at the outcome of your model or, better still, to compare several scenarios. GSE will solve these two things for you with minimum effort (just add a few GAMS comments and you will have a working GSE user interface). The third weakness of GAMS is the way input and output data are read/written. This weakness has been solved by GAMS users by writing additional tools/programs that are called within the GAMS code. For instance, there are tools that will read/write data from Access/Excel to GAMS and vice versa. This offers many additional user interfaces for data management, but is still slow for large datasets and is not always simple to implement and maintain. GAMS recognised this weakness and started to propagate use of GDX binary data files in GAMS. Instead of using very difficult GAMS commands (e.g. PUT statements) to generate output, it is very easy to create a GDX file with parameters, variables, equations and sets. Also reading data from GDX within GAMS is simple and, above all, very fast. For example, after running/solving your model the command

```
execute_UNLOAD 'results.gdx'
```

is enough to write all inputs, outputs, parameters, variables, equations and sets to a binary file (results.gdx). Also writing a few variables and parameters to a file is simple:

```
execute_UNLOAD 'results.gdx', x, z, c
```

Here variables x and z and parameter c are written to file results.gdx. Once a GDX file has been created, it is simple to read data from this GDX file into GAMS parameters, etc.:

```

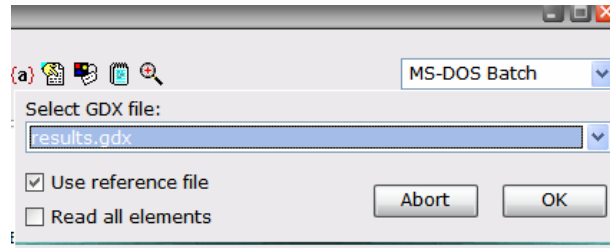
$GDXin 'results.gdx'
$load c
$GDXIN

```

Here the values of parameter c will be read from GDX file results.gdx (at compile time). Several models have used Excel spreadsheets for inputting parameters, after writing some small GAMS code that will write these parameters to a GDX file, after which, instead of reading spreadsheets, inputting the GDX file gives a performance boost of several minutes. So GDX is small and fast, but is this enough to make us happy? The answer is NO. Once you have created a GDX file you need a tool to view the GDX file. GAMSIDE and a separate GAMS tool gdxviewer.exe can be used to view the contents of a GDX file. There is, however, one big drawback when looking at a GDX file: some of the information about, say, a parameter is missing. You can see,

for example, that a parameter has three sets, but you do not know which (you only know the set elements that have values in a GDX file but you do not know which set they belong to).

Figure 2.2 Reference file selection



GAMS has several run-time options that make it possible to extract all the meta-information from your GAMS code and write it to a “reference file”. Using this reference file, it is possible to view a GDX file with all the meta-information available. Within Gtree it is possible to open a GDX file in the GDX DataViewer and to look at the GDX data (the same DataViewer is used with the same options as within GSE when viewing data and comparing scenarios) (Figure 2.2.). Conclusion: creating and reading GDX files is very easy in GAMS. Viewing the content of a GDX file is very easy in Gtree and its GDX DataViewer.

2.3. Scenario analysis (GDX DataViewer)

The previous section concluded that writing a GDX file and viewing it is simple. You just take your model and, after the solve statement, you enter the GAMS command

```
execute_UNLOAD 'results.gdx'
```

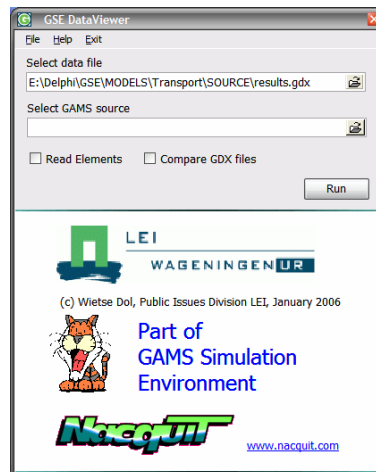
You run this model and all inputs and outputs are written into the GDX file. Now you want to run a second scenario, i.e. you change a few things in the GAMS code and instead of writing file results.gdx you will write file results1.gdx. You run the GAMS code and file results1.gdx is created. You then change your GAMS code and the GDX filename again and run GAMS. You are now running scenarios and would really like to be able to compare the contents of the GDX files (and hence look at the differences between scenarios). This is possible with the GDX DataViewer. Just open the GDX DataViewer (Figure 2.3).

Figure 2.3 GDX Data Viewer



Then select data file results.gdx (Figure 2.4).

Figure 2.4 GDX Data Viewer – select data file



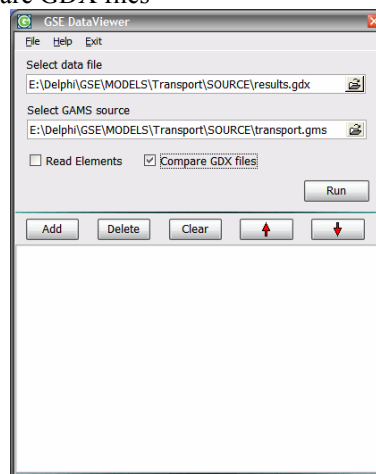
File “Select GAMS source” just points to the GAMS file you ran to generate the GDX file, i.e. this file will be used to create a reference file with all the meta-information in it that is used to show the complete meta-information on the parameters and variables stored in the GDX file (Figure 2.5).

Figure 2.5 GDX Data Viewer – select GAMS source



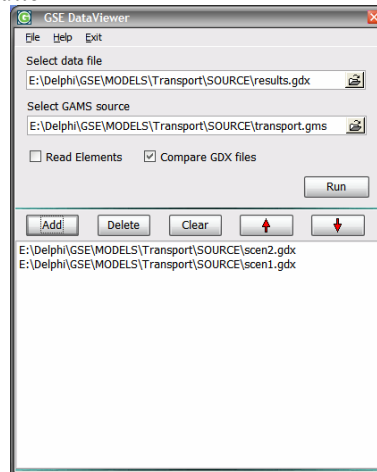
When you press Run the contents of file results.gdx will be shown. However, you would like to compare them with two other scenarios (scen1.gdx and scen2.gdx). Therefore, check the “Compare GDX files” box (Figure 2.6).

Figure 2.6 GDX Data Viewer – compare GDX files



Now press Add and add the two scenarios/GDX files (Figure 2.7).

Figure 2.7 GDX Data Viewer – add button



After this press Run and you will be able to compare and see the three different scenarios in the GSE DataViewer. You can export these data, print them, make graphs, etc., etc. As you can see, running and comparing scenarios is easy.

2.4. Model versions (GSE)

Making GDX files, running GAMS and comparing scenarios is easy but still leaves one big problem: when you open a GDX file you see inputs and outputs but do not know which version of the model (which GAMS code) belongs to this GDX file. Hence to make your research reproducible you need some kind of version control or scenario control system. GSE offers all that. With a few GAMS comments in your source code you can generate a complete graphical user interface for your model. Besides the version and scenario control, GSE offers many more functions. Just have a look at the documentation or at the Transport model example to see the little effort it takes to get your model into GSE.

3. Quick reference guide to GSE for model-builders

3.1. Introduction

You have just installed GSE and are eager to use it. But before you hit the road, please spend a few minutes reading this quick reference guide. GSE offers you a host of functions and if you start using GSE without reading parts of the manual or this quick reference guide you will probably get a lot of fatal errors and other mysterious warnings. These errors/warnings lead to frustration and could possibly discourage you from using GSE for your own models. This quick reference guide is by no means a substitute for the manual. It will just explain some of the basic steps needed to understand GSE and show you how to get your first (simple) model running under GSE. After this you will be ready for the more advanced functions, which you can learn by reading the GSE manual.

3.2. Installation of GSE

Take the GSE CD-ROM and run the SETUP.exe program. If you have downloaded GSE from the internet run the GSESetup.exe program. When finished, under *Programs* you will see the *GSE* folder. In this folder you will find the programs that are used within GSE (Table 3.1).

Table 3.1 Programs used within GSE

Shortcut	Explanation
GSE	The GSE user interface.
GSE user guide	The complete GSE documentation, including:
Gtree	A replacement for GAMSIDE and GSE/GAMS editor.
Import2GSE	A tool to install GAMS models for use within GSE and to perform GSE distributions of your model.
TagEditor	A tool to insert GSE tags into your GAMS code (this function is also available in Gtree and hence this tool is obsolete).
Translate	A tool to translate the GSE user interface into another language (or to change an existing one).
DEMO folder	This folder contains examples of GSE/GAMS.

Tags play an important role within GSE. Tags can be seen as GAMS comments that GSE will use to perform certain tasks, e.g. defining which sets of parameters are used within the GSE user interface.

Before you use GSE for your own models, it is advisable to have a look at the DEMO examples. These will give you an understanding of the user interface of GSE and will give you a good idea of what GSE tags are and what they do. We know that the DEMO examples are simple and that you want more. Be patient and trust us. We have really large, complex models running under GSE. Just start with the simple things and, whenever necessary, have a closer look at the manual when you do not know how to do something in GSE.

3.3. Your first model in GSE

It is possible to build a GSE user interface using your own model and to add all the GSE tags and administration by hand. We started that way and made many mistakes and received large numbers of error messages. No doubt you will make these same errors and probably you are not as persistent as we are to solve them. We therefore decided to build tools that would help you to import a GAMS model into GSE and apply the necessary GSE tags. As a demonstration, we will start with a simple example and explain the steps you need to take, in order to get the GAMS model running in GSE (see the TRANSPORT demo). Imagine you have a working GAMS model (TRANSPORT.GMS) which is stored in directory c:\Program Files\GSE\GAMS SOURCE\TRANSPORT (i.e. GSE stores the original source files of the GSE demos with no GSE tags in a subdirectory of c:\Program Files\GSE\GAMS SOURCE).

Step 0: run GAMS and see if your model runs without GAMS errors.

3.3.1. Step 1: starting Import2GSE

After starting Import2GSE (Figure 3.1) choose option 1 (also available from the toolbar under *Model* and *New*), enter the model name *Transport* and press *New* (Figure 3.2).

Figure 3.1 Starting Import2GSE

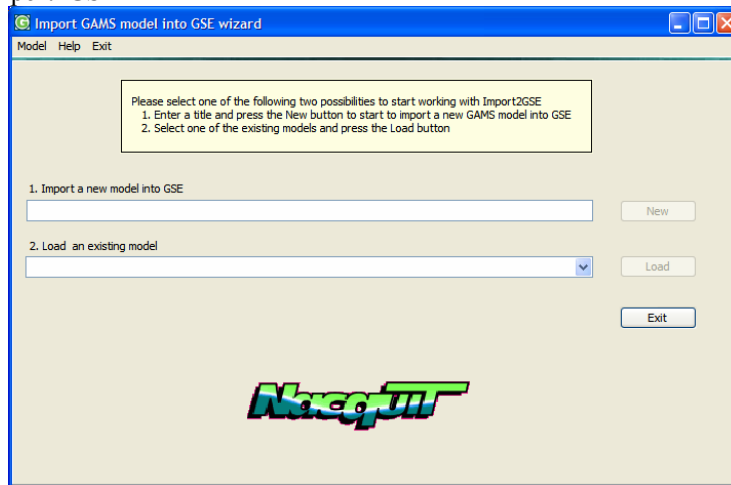
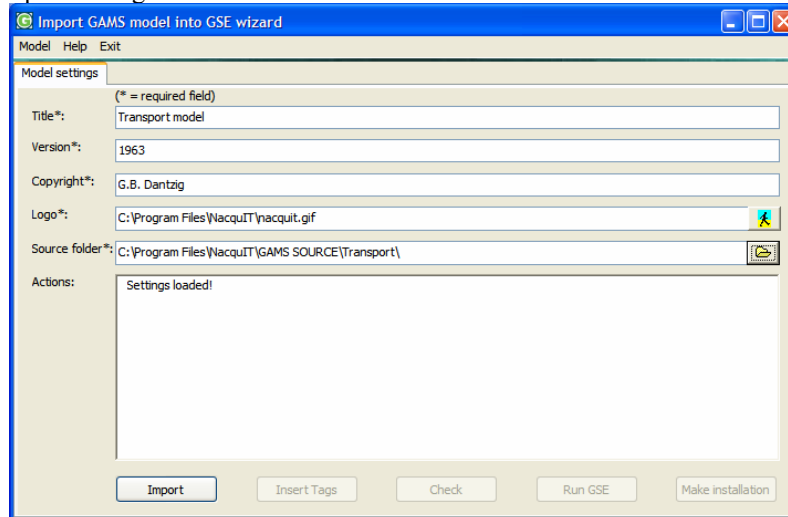


Figure 3.2 Selecting a new model

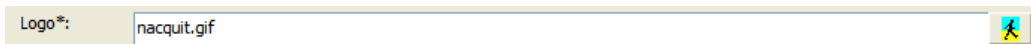



After you press *New* the main Import2GSE window will be shown (Figure 3.3).

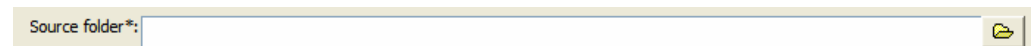
Figure 3.3 Model import settings



Enter the title, version and copyright statement. All these text strings are shown in the startup window when you start GSE.



In the Logo field you can specify which logo will be shown during the startup window. Just type the name of the logo or, even better, press  and use the file explorer to select the graphic file used as a logo.



The last thing you have to do is to specify the path where the original GAMS files are located (in our case C:\Program Files\GSE\GAMS SOURCE\TRANSPORT).


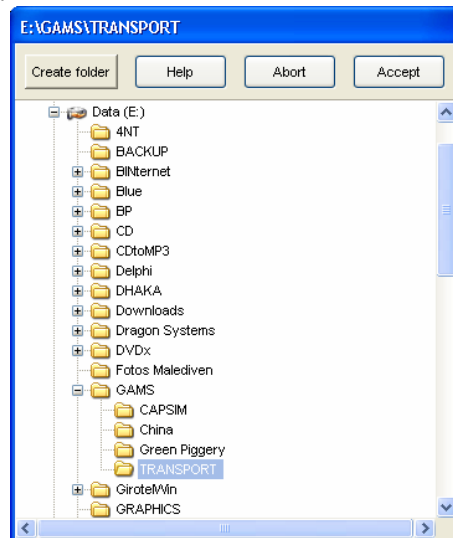
When you press  a path selector is shown (Figure 3.4).

Figure 3.4 Selecting the source path

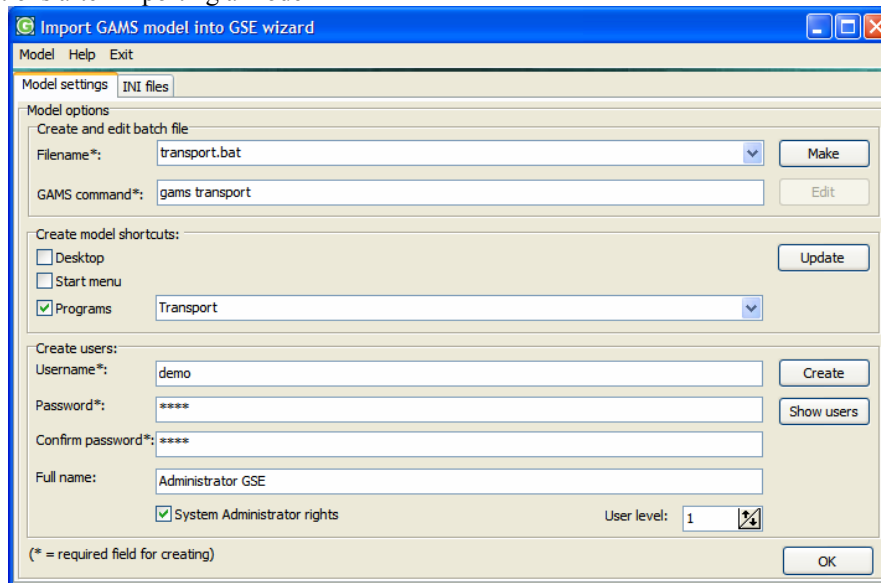


Go to the directory you want (or use *Create folder* to create a new directory) and when you are in the right directory press *Accept*. When you load an existing GSE model (option 2 in Figure 3.2) you see that the source path is \$local\MODELS\TRANSPORT. This means c:\program files\gse\models\TRANSPORT and hence \$local stands for c:\program files\gse (the directory where GSE is installed).

Now you are ready to import the model into GSE (do all the administrative tasks like creating a database, copying files, etc.) by pressing *Import*. As you become a more experienced user and want more features you can

go to the INI files tab sheet and enter/add/change the GSE INI files for extra functions (see the GSE manual).

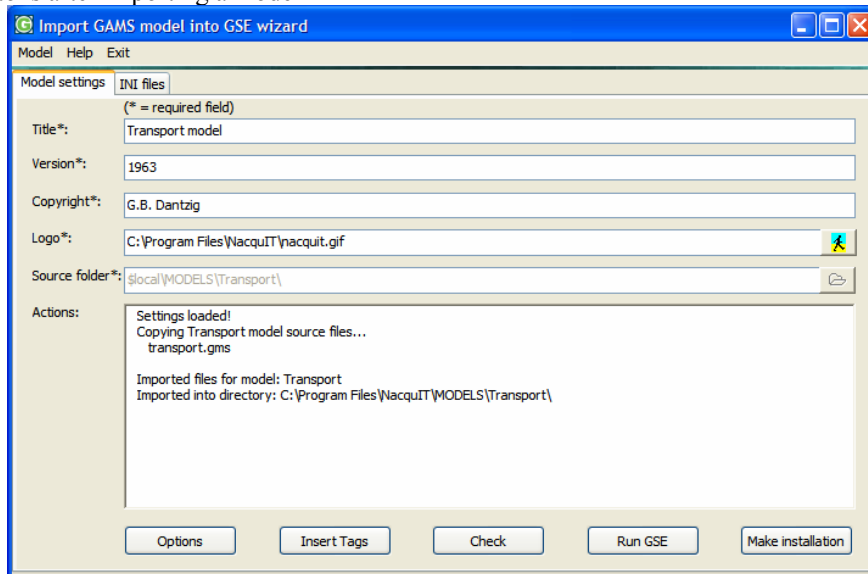
Figure 3.5 Options after importing a model



After importing the model you have several options, as shown in Figure 3.5.

You need to create a batch file (transport.bat) to run the model (gams transport.gms). This is done by entering the Filename and GAMS command and then pressing Make. You can change or add extra batch commands used within GSE by pressing Edit (see the manual for further details). The second option is to create shortcuts for the new model. Just select which shortcut you want and press Update. Once you have imported your model you need to assign users to the GSE interface. These users are allowed to edit and run different versions and scenarios of your model. Do not forget to specify at least one user with system administrator rights. This user can change the database (e.g. add more users or change the database tables). Once you have added a user with administrator rights, you can add more users only if you supply the username and password of a user with administrator rights (this is a security measure since others could use the Import2GSE tool to add users with administrator rights and hence gain control over your model and database). Press OK and you have finished importing the model into GSE. However, you still have to apply the GSE tags to your GAMS model files before you can run GSE and import the first version of the model into GSE. Still, the first step has been taken and, after you have pressed OK, you will see that Import2GSE enables various buttons (Figure 3.6).

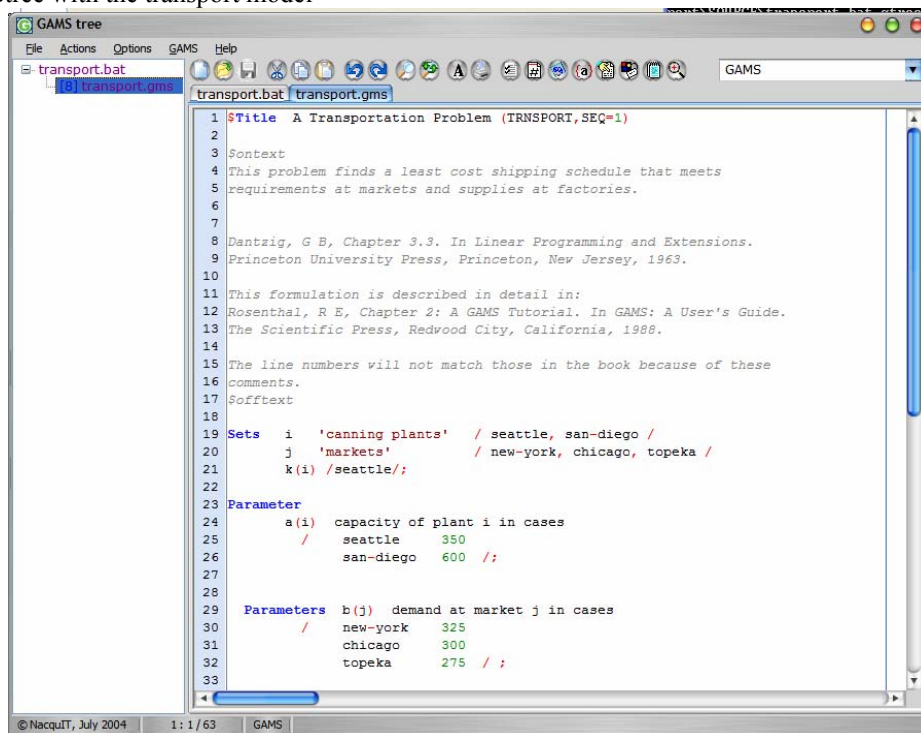
Figure 3.6 Buttons after importing a model



3.3.2. Step 2: inserting GSE tags

Pressing **Insert Tags** (Figure 3.6) will start Gtree (Figure 3.7).

Figure 3.7 Gtree with the transport model



Gtree is an excellent replacement for GAMSIDE and can be used for developing models (versions). The key feature of Gtree is that it shows the structure of your model and allows you easily to add GSE tags.

First select file transport.gms in the tree and then, in the editor, select the three lines:

```

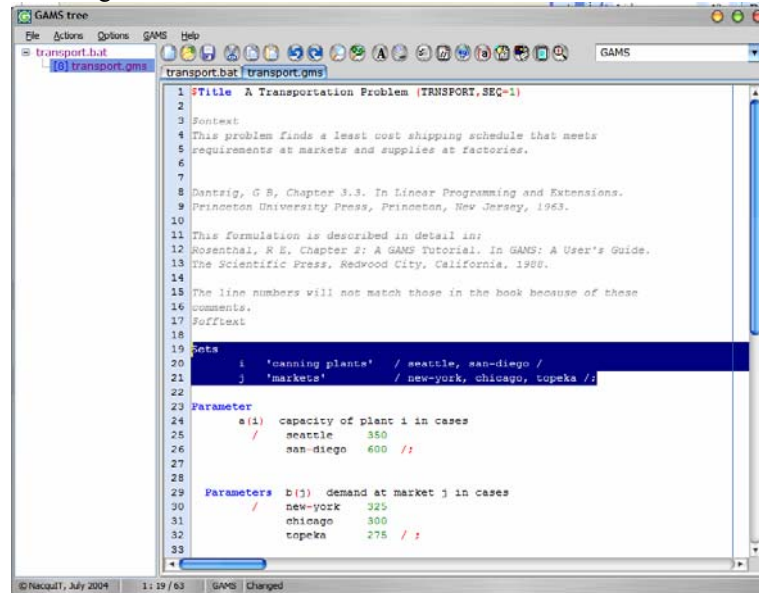
Sets
  i canning plants / seattle, san-diego /
  j markets / new-york, chicago, topeka / ;

```

Next, press Ctrl / (i.e. hold down the Ctrl key and press the forward slash key). The GSE tag window will pop

up (see Figure 3.9). Then press *Insert* (the result is shown in Figure 3.10). In Gtree you will see that a SET tag (lines 22-26) has been added to the tree. Look at the editor (lines 22-26) and note that all GSE tags start with an asterisk in the first column/first character of the line. This asterisk indicates in GAMS that the remainder of the line is a comment. As a result, the Transport model with all the tags inserted can still run under GAMS without causing any errors due to GSE tags.

Figure 3.8 Inserting a SET tag

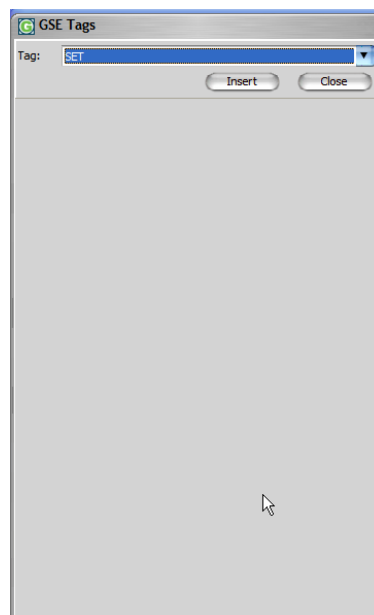


```

1 *Title A Transportation Problem (TRANSPORT,SEQ=1)
2
3 Fonttext
4 This problem finds a least cost shipping schedule that meets
5 requirements at markets and supplies at factories.
6
7
8 Dantzig, G B, Chapter 3.3. In Linear Programming and Extensions.
9 Princeton University Press, Princeton, New Jersey, 1963.
10
11 This formulation is described in detail in:
12 Rosenthal, R E, Chapter 2: A GAMS Tutorial. In GAMS: A User's Guide.
13 The Scientific Press, Redwood City, California, 1988.
14
15 The line numbers will not match those in the book because of these
16 comments.
17 Fonttext
18
19 Data
20 i 'canning plants' / seattle, san-diego /
21 j 'markets' / new-york, chicago, topeka /;
22
23 Parameter
24 a(i) capacity of plant i in cases
25 / seattle 350
26 san-diego 600 /;
27
28
29 Parameters b(j) demand at market j in cases
30 / new-york 325
31 chicago 300
32 topeka 275 /;
33

```

Figure 3.9 TagEditor of Gtree



Inserting a SET tag is not difficult. The second tag which will be inserted is an INPUT tag. The INPUT tag indicates to GSE which parameters should be displayed as input parameters.

Figure 3.10 Result of a SET tag

```

22 * <SET$>
23 Sets
24   i 'canning plants' / seattle, san-diego /
25   j 'markets' / new-york, chicago, topeka /;
26 * </SET$>
27
28 Parameter
29   a(i) capacity of plant i in cases
30   /
31   seattle 350
32   san-diego 600 /;
33
34 Parameters b(j) demand at market j in cases
35 /
36   new-york 325
37   chicago 300
38   topeka 275 /;
39
40 Table d(i,j) distance in thousands of miles
41   seattle new-york chicago topeka
42   san-diego 2.5 1.7 1.8 1.4 ;
43
44 Scalar f freight in dollars per case per thousand miles /90/ ;
45
46 Parameter c(i,j) transport cost in thousands of dollars per case ;
47   c(i,j) = f * d(i,j) / 1000 ;
48
49 Variables
50   x(i,j) shipment quantities in cases
51   z total transportation costs in thousands of dollars ;
52
53 Positive Variable x ;
54

```

In the editor select/mark the lines:

Parameter

```

a(i) capacity of plant i in cases
/ seattle 350
san-diego 600 /;

```

Press Ctrl / and select the INPUT tag from the combo box. Below the combo box you will see the options for the INPUT tag (Figure 3.11). The *Format* field shows you how the data are shown in GSE, and the *Dimension* field is just a text field specifying what the dimensions of the elements of the parameter will be. All other options can be seen as advanced user options and we will skip them for now (see the manual for detailed explanations and examples). Pressing *Insert* will result in Figure 3.12.

Figure 3.11 INPUT tag options

GSE Tags

Tag: INPUT

Insert Close

FORMAT
#,###,##0.###

DIMENSION
number

USERLEVEL
hide

BOUNDS

NONEDIT

INFO

To apply the INPUT tag to more parameters, just select the code in the editor, select the INPUT tag, insert the options and press *Insert*. See Appendix 1 for the file with all the GSE tags inserted. Note that for every input parameter/scalar you have to insert a separate INPUT tag. This looks inefficient, but remember that all the (meta)information you enter for an INPUT tag belongs to one input parameter and this (meta)information can differ between parameters.

Figure 3.12 Result of the INPUT tag

```

28 | <INPUT$>
29 | Parameter
30 |   a(i)  capacity of plant i in cases
31 |       /  seattle  350
32 |         /  san-diego  600 /;
33 | * <FORMAT #,###,##0.###$>
34 | * <DIMENSION number$>
35 | * </INPUT$>
36 |
37 |
38 | Parameters b(j)  demand at market j in cases
39 |       /  new-york  325
40 |         /  chicago  300
41 |         /  topeka  275 /;
42 |
43 | Table d(i,j)  distance in thousands of miles
44 |       /  new-york  chicago  topeka
45 | seattle  2.5      1.7      1.8
46 | san-diego 2.5      1.8      1.4 ;
47 |
48 | Scalar f  freight in dollars per case per thousand miles /90/ ;
49 |
50 | Parameter c(i,j)  transport cost in thousands of dollars per case ;
51 |   c(i,j) = f * d(i,j) / 1000 ;
52 |
53 | Variables
54 |   x(i,j)  shipment quantities in cases
55 |   z      total transportation costs in thousands of dollars ;
56 |
57 | Positive Variable x ;
58 |
59 | Equations
60 |   cost      define objective function

```

You have defined the sets and elements, i.e. the inputs, and are now ready to specify the outputs. This is done with the GDXOUTPUT tag. (Starting from GAMS version 20.6 there is the possibility of saving sets, parameters and equations in a binary GDX file (GAMS Data eXchange).) First go to the end of file TRANSPORT.GMS, insert an empty line and start the TagEditor (by pressing Ctrl /). Select the GDXOUTPUT tag and insert as OUTPUT FILE name the text results.gdx. Then press New and add the symbol x. Do this again for symbol z. Then select the symbol x and enter the data as shown in Figure 3.13. The data for z are displayed in Figure 3.14.

Press Insert to create the GDXOUTPUT GSE tag (Figure 3.15). Note that for output you have to specify if you want output for a GAMS variable or a GAMS parameter/scalar. GAMS treats these two very differently, but just check the PARAMETER or VARIABLE button and the TagEditor knows what to do. Appendix 2 shows you the statement that you should insert if you do not use the GDXOUTPUT tag but use the EXPLICIT OUTPUT and OUTPUT tags instead.

Figure 3.13 Creating output parameter x

The screenshot shows the 'GSE Tags' dialog box with the following configuration:

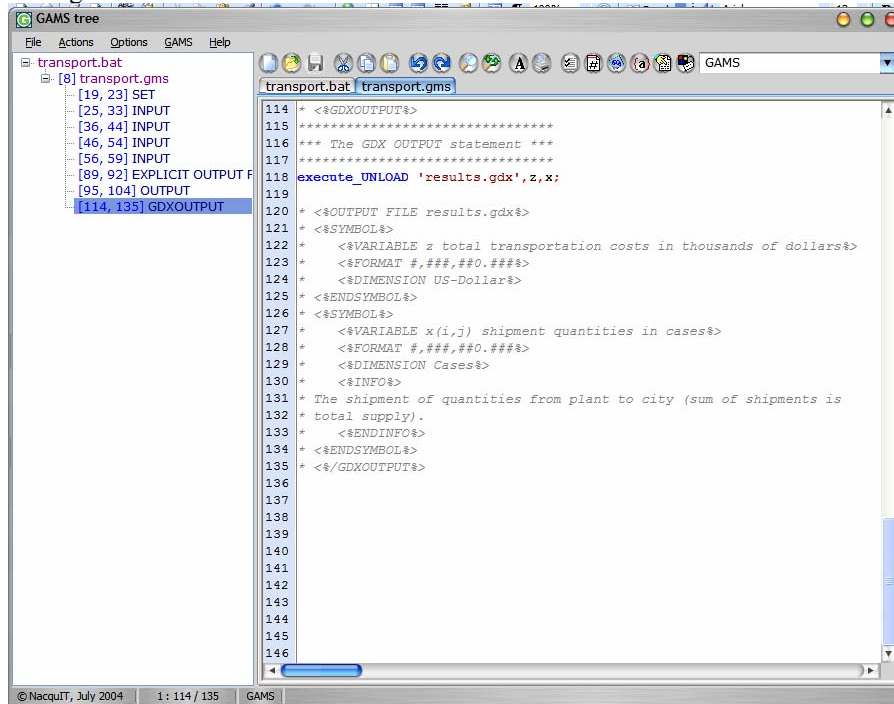
- Tag:** GDOUTPUT
- OUTPUT file name:** results.gdx
- SYMBOL:** x
- Current symbol settings:**
 - GAMS name:** x(i,j)
 - Description:** shipment quantities in cases
 - FORMAT:** #,###,##0.###
 - DIMENSION:** Cases
 - USERLEVEL:** (empty)
 - BOUNDS:** (empty)
 - INFO:** The shipment of quantities from plant to city (sum of shipments is total supply).

Figure 3.14 Creating output parameter z

The screenshot shows the 'GSE Tags' dialog box with the following configuration:

- Tag:** GDOUTPUT
- OUTPUT file name:** results.gdx
- SYMBOL:** z
- Current symbol settings:**
 - GAMS name:** z
 - Description:** total transportation costs in thousands of dollars
 - FORMAT:** #,###,##0.###
 - DIMENSION:** US-Dollar
 - USERLEVEL:** (empty)
 - BOUNDS:** (empty)
 - INFO:** (empty)

Figure 3.15 GDx tag



The complete file with all the tags can be found in Appendix 1. You have to admit that inserting tags is not difficult when using Gtree and its internal TagEditor. Just remember that many tags have options that are interesting for advanced users, but you can skip them for now. As you become a more experienced user you can read the manual and change/update the tags you have already inserted and hence add more functions to GSE. Once you have inserted the tags your version of the Transport model is ready for importing into GSE. As programmers of GSE we thought that using the Gtree TagEditor would generate perfect code. We discovered the hard way, this was that after inserting tags we still had many errors. Consequently, do not import your model into GSE yet, but perform step 3: checking the GSE tags.

3.3.3. Step 3: checking the GSE tags

Checking a version means nothing more than running GSE and importing a new version of the model without writing anything to the GSE database. As GSE parses all the files, it will show you which tags it found. If there is an error, it indicates where (file and line number). You can then go to that file and correct the error. After correcting the error you should restart checking the GSE tags. Repeat this process until there are no errors left and GSE can successfully import the model. Most of the errors are the result of mistyping. Perhaps you have inserted a space or line break where there should not be one. Perhaps you have mixed up two tags or just forgotten to insert the end-tag. Note that GSE is not a complete GAMS parser, so there is a limit to the possibilities that GSE can parse. There are some permissible statements that GAMS will understand but GSE does not. If you do not understand why there is an error look at the examples given in the manual. These indicate what GSE can understand. Note that using the Gtree TagEditor will minimise the risk of typing errors and that most errors are just because GSE will not parse your GAMS statement. Most of the time it is easy to rewrite the statement into something GSE can understand.

You can check the model by clicking on Check in the Import2GSE tool or by using the Actions toolbar option in Gtree and selecting the Check GSE tags option (or pressing the F12 key).

Figure 3.16 shows the checking process and Figure 3.17 shows what happens when you have an error (in this example the user has forgotten the <%/INPUT%> tag for parameter b(j)). If you get an error press Edit and a screen like Figure 3.18 shows where the error occurred (line 36). Insert the forgotten <%/INPUT%> and press Save. In Import2GSE press Check once again to see if there are any more errors. If there are no errors left you can proceed to step 4: importing a version.

Figure 3.16 Checking tags

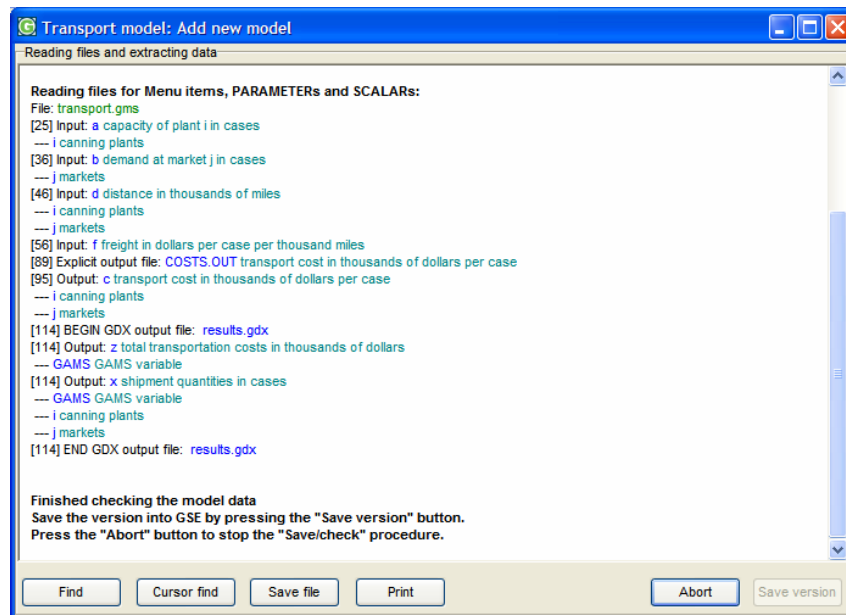


Figure 3.17 Forgetting to close an INPUT tag

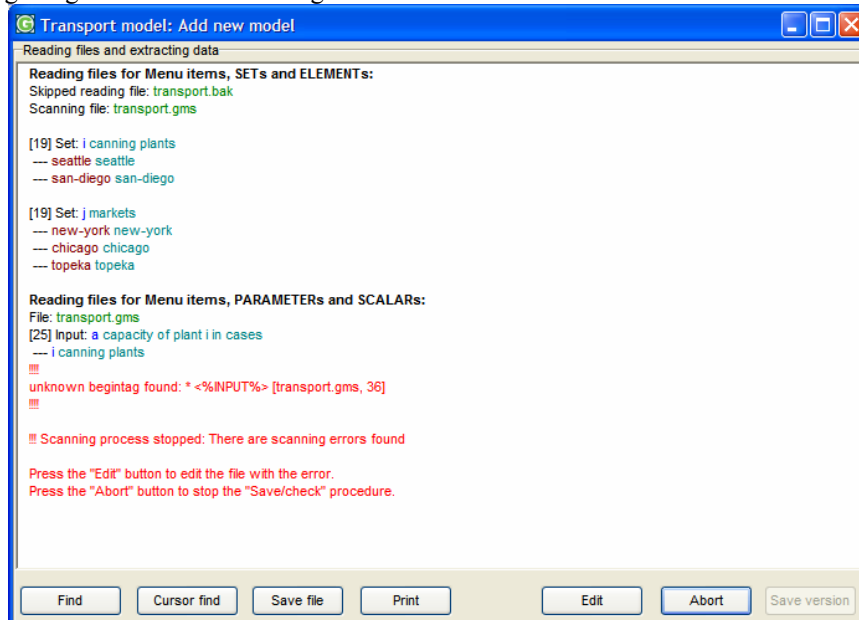
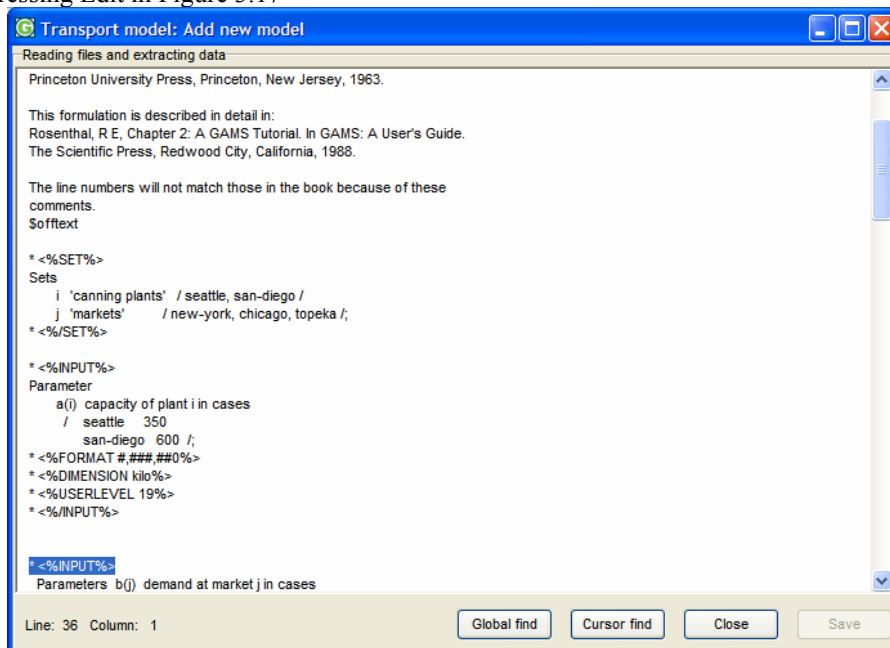


Figure 3.18 Pressing Edit in Figure 3.17



3.3.4. Step 4: importing a version

Congratulations. You have just finished all the preparations for GSE. Now you can insert the version of the model in GSE and start to view/edit the input, run scenarios, look at the output and compare scenarios. Start GSE with the Transport model (once you have created a shortcut icon with the Import2GSE tool, all you have to do is double-click on the icon). You are asked to supply a username and password. Type in one of the users you created with the Import2GSE tool.

Figure 3.19 Starting GSE

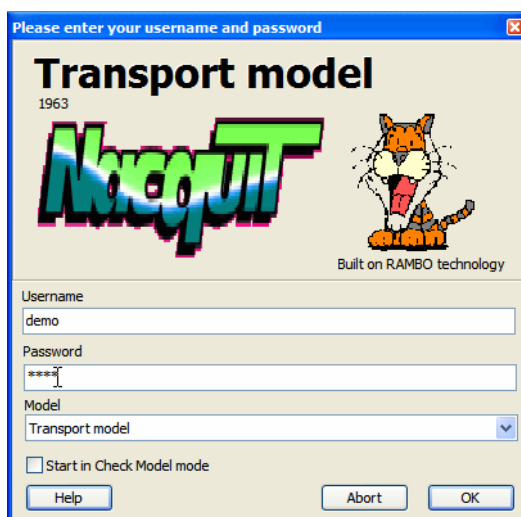
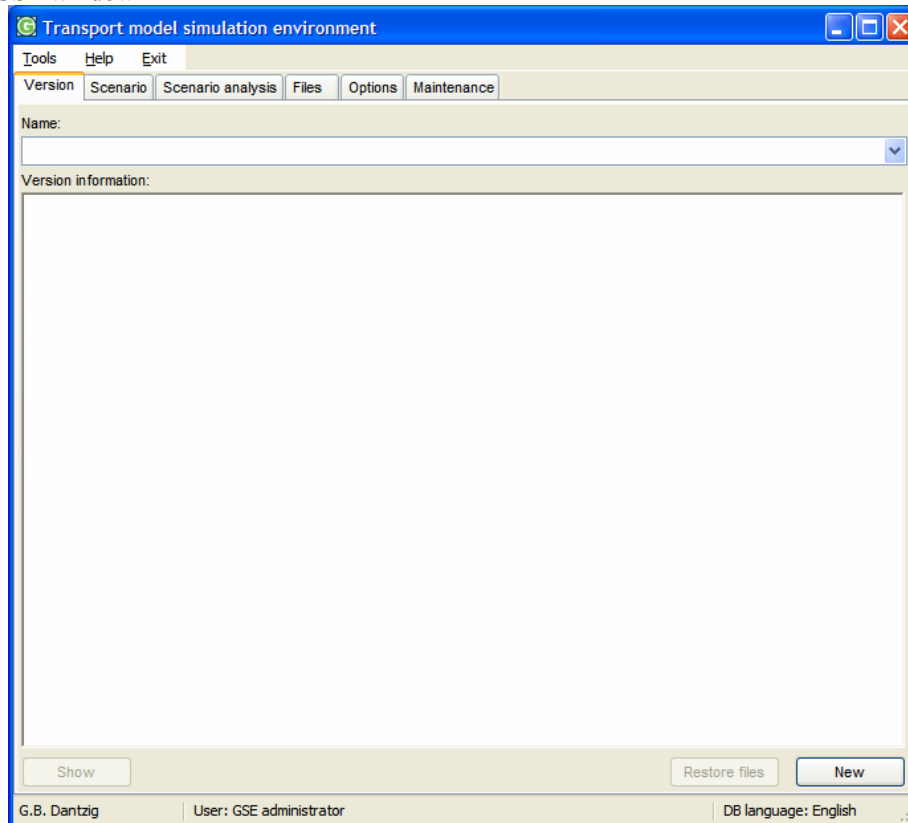


Figure 3.20 GSE window



Now press *New* and enter the *Version description* and the *Start file* (the batch file you created in Import2GSE). In the *Version information* you can type additional information (knowledge) of the model. When you are ready press *Save* and you will see the same process as when checking the GSE tags (step 3), except that now everything is written to the GSE database.

Figure 3.21 Adding a new version of the model

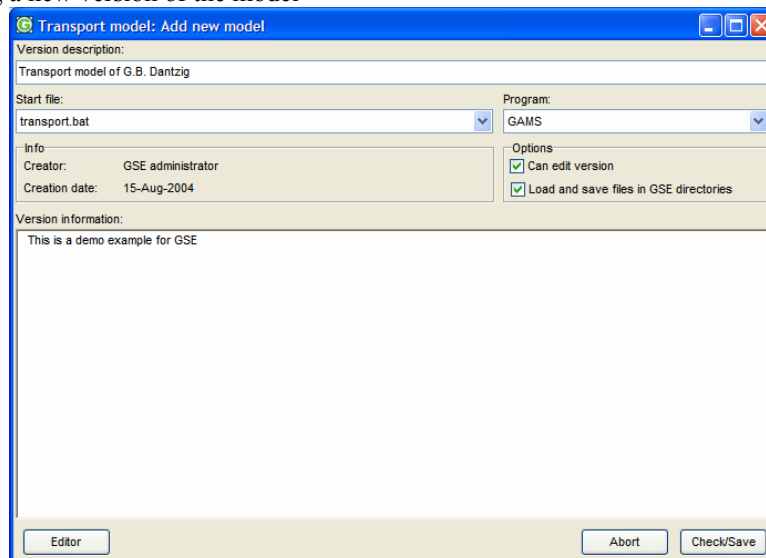
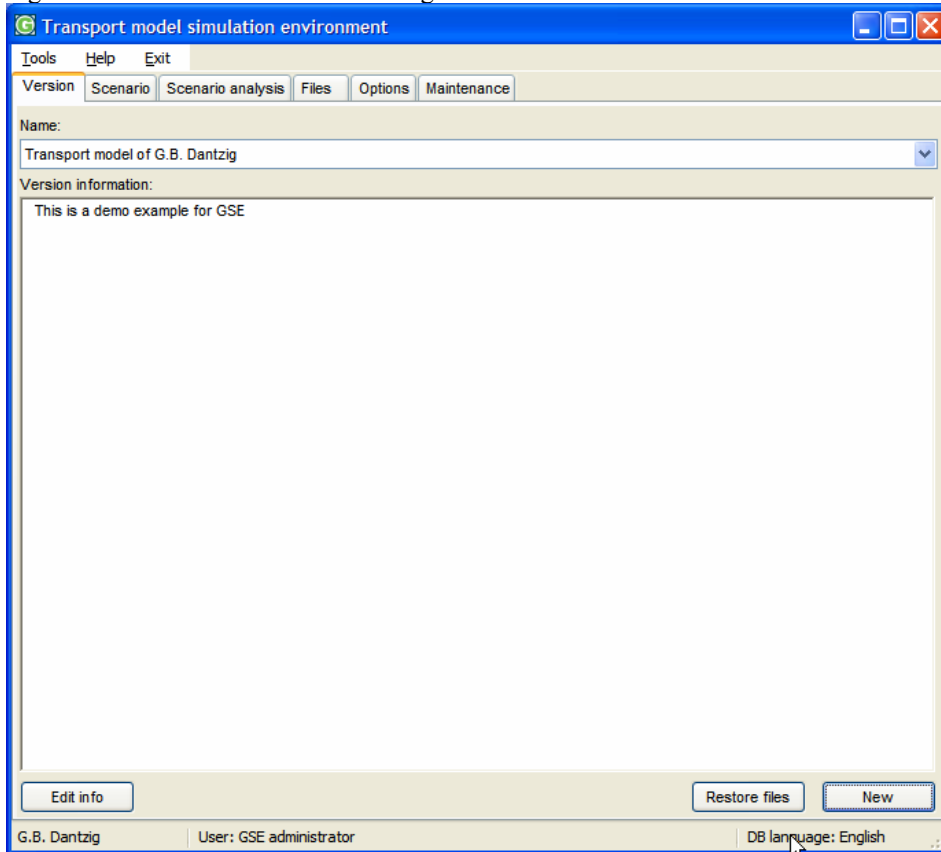
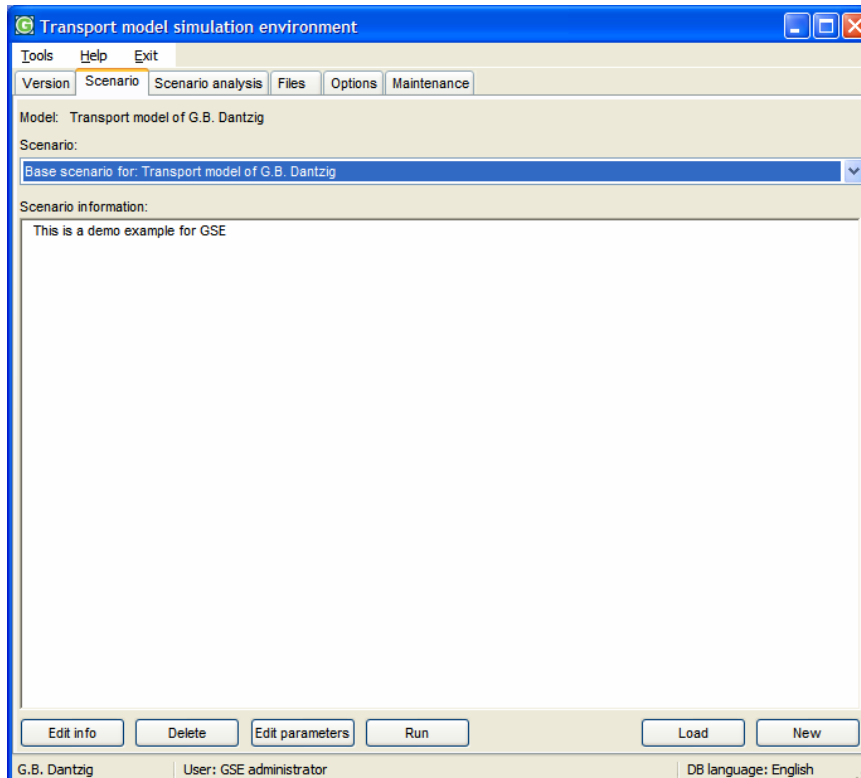


Figure 3.22 GSE window after inserting a version of the model



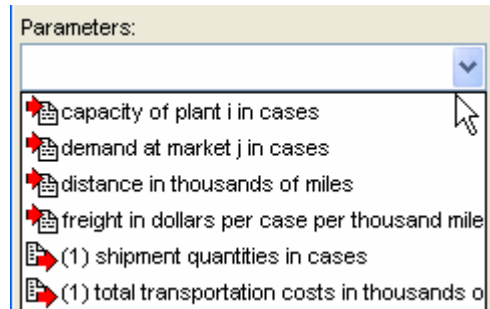
You have not only inserted a new version of the model, but also created a base scenario containing all the input. Go to the Scenario tab sheet (Figure 3.23) and press Edit parameters.

Figure 3.23 Scenario tab sheet



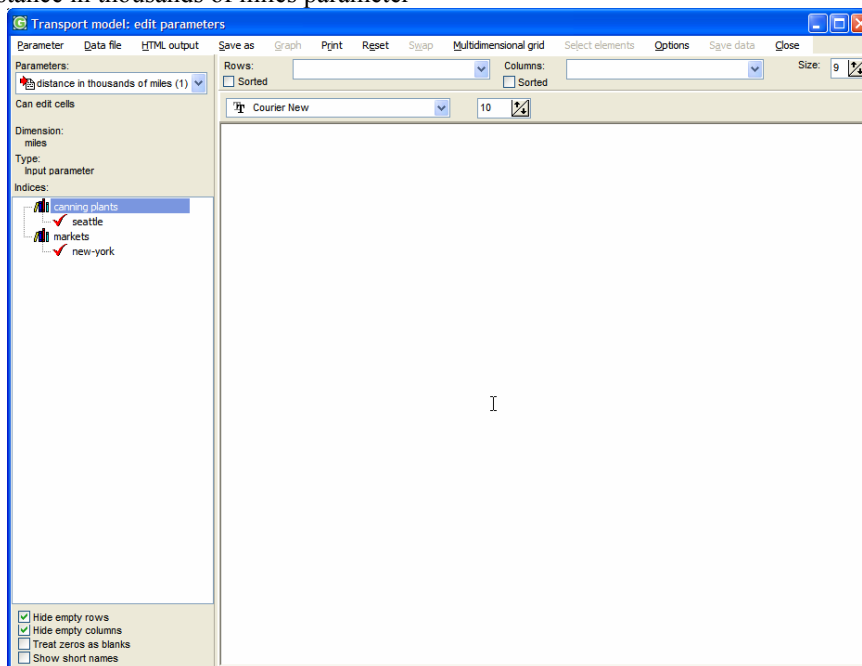
You now are in the DataViewer/editor. By clicking on the *Parameters* combo box you can select one of the parameters.

Figure 3.24 Parameters



Select distance in thousands of miles and you will see Figure 3.25.

Figure 3.25 Distance in thousands of miles parameter



In the viewer you can choose the row and column you want to display (Figure 3.26). See the manual for a detailed description of all the possibilities offered by the viewer.

Figure 3.26 Rows and columns display

	new-york	chicago	topeka
seattle	2.5	1.7	1.8
san-diego	2.5	1.8	1.4

When you are ready to inspect and change the input (if you try to view output you will see that this is empty, since you have not yet run the scenario) press Run in Figure 3.25. GAMS is started and after GAMS has finished (without any errors) you will see Figure 3.27.

Here you have a complete list of all the input and output (including GAMS LST and GDX files). Click on any of them and you can inspect them in the viewer. Once you are satisfied with the scenario press Save scenario and the scenario is saved to the GSE database. After you have saved the scenario you can no longer edit/change the data. You can, however, create a new scenario and then import input from the previous scenario and change it in the new one.

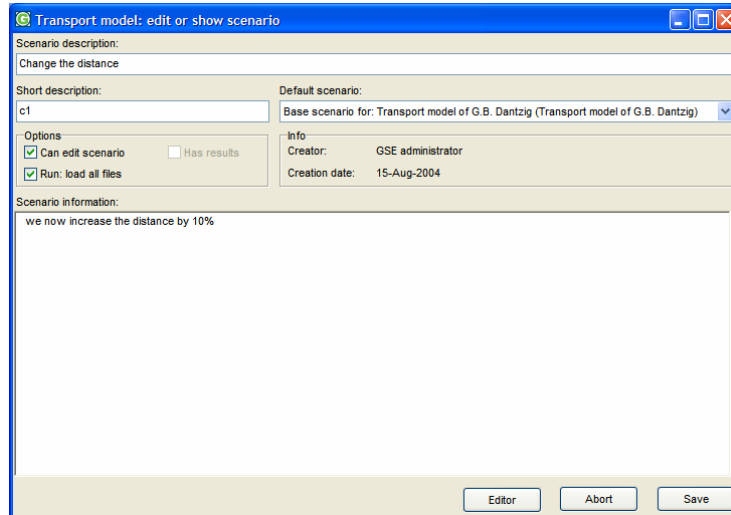
Figure 3.27 After running GAMS you can view the input and output

Input parameters	Output parameters	GAMS lst files	GAMS GDX files
capacity of plant i in cases (a)	shipment quantities in cases (x)	temp.lst	results.gdx
demand at market j in cases (b)	total transportation costs in thousands of dollars (z)	transport.lst	xlink.gdx
distance in thousands of miles (d)	transport cost in thousands of dollars per case (c)		
freight in dollars per case per thousand miles (f)			

3.3.5. Step 5: running additional scenarios

In step 4 you ran the base scenario. Now you want to run some additional scenarios. In the Scenario tab sheet press New and you will get a scenario form (Figure 3.28).

Figure 3.28 Creating a new scenario



To create a new scenario you have to supply a *Scenario description* and a *Short description* (often used for scenario comparison). You can add further details in the *Scenario information* field. Since you do not want to re-enter all the input data you can import input data from another scenario. This is exactly what *Default scenario* does. After you select *Default scenario*, the input data from this scenario are taken as the starting values for the new scenario. If you then go to the DataViewer/editor and change some of the values, only the changed values are stored in the database (hence occupying less space).

Here you can already see the power of GSE. Making scenarios and comparing them is simple. Read the chapter 4 of the manual for more details on how the user interface works.

3.3.6. Step 6: building a distribution list

After you have created several scenarios you will probably want to distribute the model and the scenarios to others. This is simple using the Import2GSE tool. Start Import2GSE, select the Transport model from the *Load an existing model* combo box and press *Load*.

Figure 3.29 Loading the Transport model

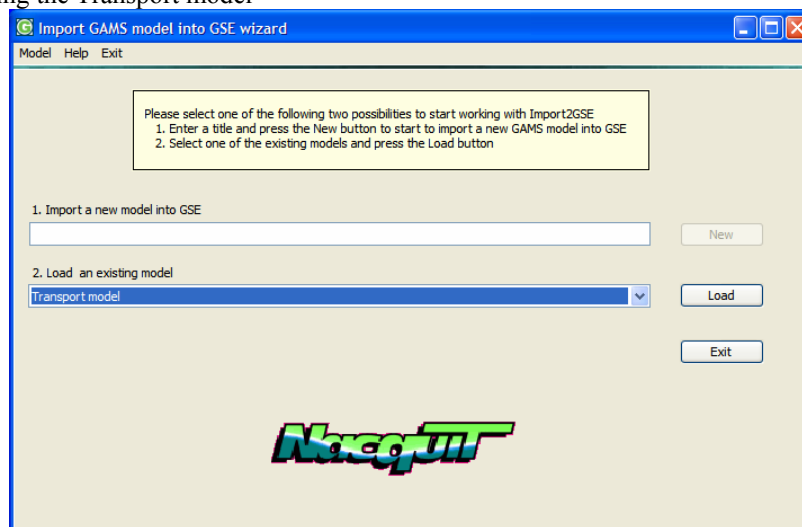
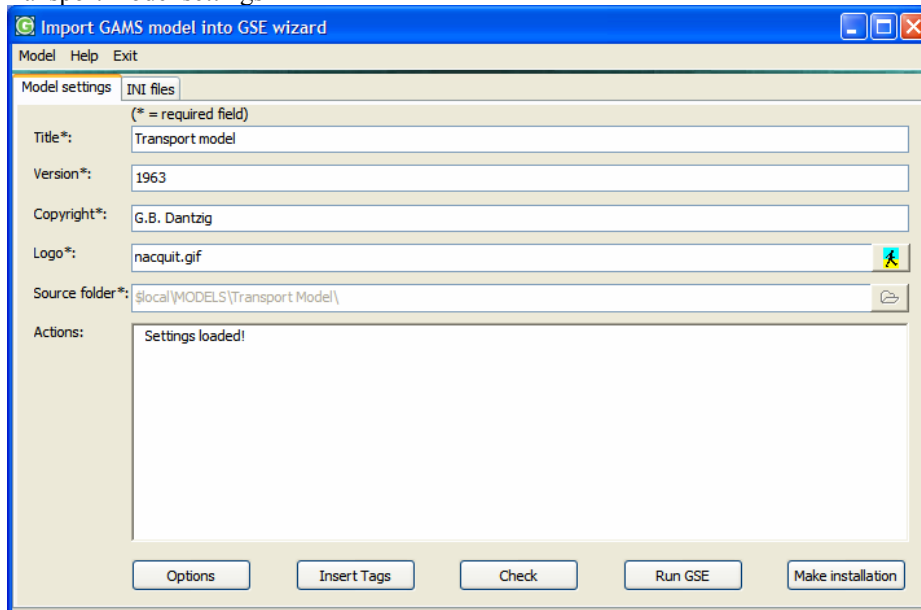
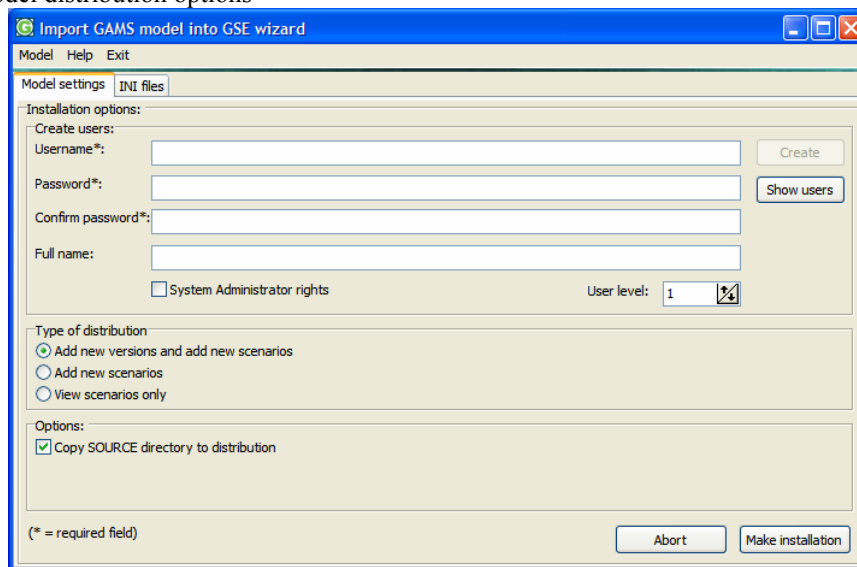


Figure 3.30 Transport model settings



Now press Make installation and you will see the screen in Figure 3.31.

Figure 3.31 Model distribution options



You can add new users and decide which type of authorisation you want, i.e.:

1. Allow people to add new versions and new scenarios.
2. Allow people to add new scenarios, but not new versions.
3. Allow people only to view scenarios already created.

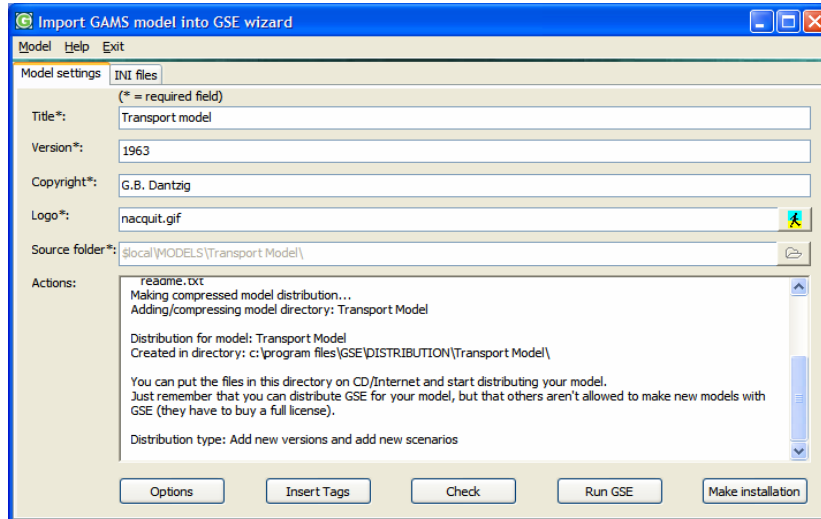
With the option *Copy SOURCE directory to distribution* you can distribute the source code of your GAMS model. After making the selections, just press *Make installation* and after a while you will see that a complete distribution list of your model is created in `c:\program files\GSE\DISTRIBUTION\Transport\`. Just copy the contents of this directory to CD-ROM and you can distribute your model.

3.3.7. Step 7: You want more

You have mastered the basics of GSE but you are not yet a full expert. Read the manual, look at the demos and whenever you have problems or wishes do not hesitate to contact us:

info@nacquit.com
www.nacquit.com

Figure 3.32 Distribution list has been created



4. Quick reference guide to GSE for model-users


4.1. Introduction

Model-builders have the unique talent of describing a world with complex interactions within a mathematical framework. This makes it possible to see what might happen in the real world when certain parameters are changed (e.g. by policy action). Each model-builder wants his or her model to be used to gain a better understanding of the interactions between processes and to calculate the differences between various policy scenarios. Running scenarios and looking at the results often leads to new versions of the models and new scenarios. This means considerable administration for the model-user and, if there is no easy-to-use graphical user interface (GUI), also understanding where to change the software code of the model. GSE makes life easier not only for the model-builder but also for the user. By simply adding comments to the model code, with GSE the model-builder can build a GUI that will help the user to benefit from the model. The user does not need to understand the model code and can run and compare scenarios without detailed knowledge of the model.

GSE offers a host of functions to support from the simplest to very advanced features. Many models do not need all these features, but it is comforting to know: if you need something special within your model, either GSE has implemented it already or it can be implemented by defining new tags.

4.2. Installing a model

One of the features of GSE is that model-builders can make their own model distribution list by using the GSE wizard Import2GSE. This guide assumes that the model-builder has created a distribution list for the TRANSPORT model and that, as a client, you received the TRANSPORT CD-ROM. When you insert the CD-ROM it will automatically start install.exe (if not, you can start this file in the Windows file explorer by double-clicking on it). Figure 4.1 shows the main window of the installation program. In this window you can first

choose where you want to install the model (default C:\Program Files\TRANSPORT). By clicking on the  icon you can change the drive/directory on which it will be installed (Figure 4.2.). It is also possible to create a new directory (by pressing Create folder in Figure 4.2. After pressing this button you can type in the new directory name and a new directory is created. Step two in installing the model is to specify where you want to install shortcuts (i.e. icons that can be clicked to start GSE with the TRANSPORT model). You are ready to install the software. Just for convenience, at the bottom of the installation program you can see the amount of disk space that is available and how much is needed for the model. If you want more information press Readme, otherwise just press Install.

Congratulations. You have installed GSE and the TRANSPORT model. As the final step in the installation process, GSE will start TRANSPORT (because you checked “*Run model after installation*”). The next time you want to start the TRANSPORT model you can use the shortcut icon created on the desktop.

Figure 4.1 Installing TRANSPORT

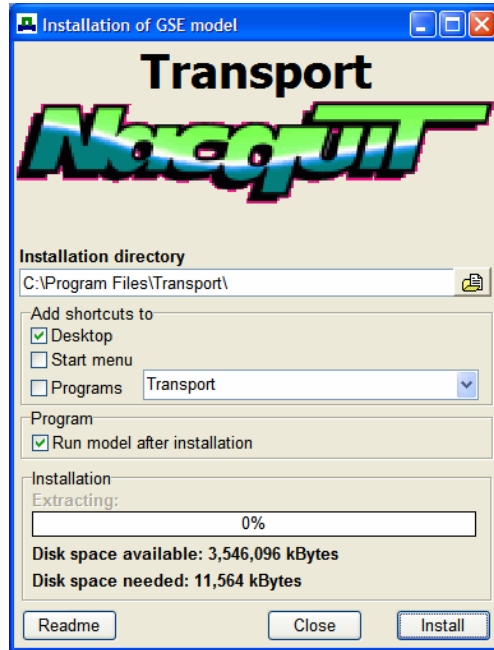


Figure 4.2 Selecting the installation directory

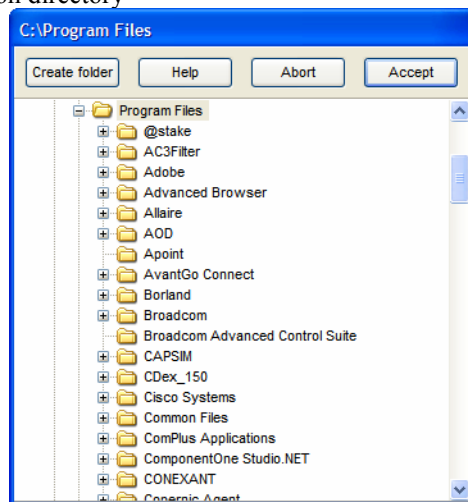
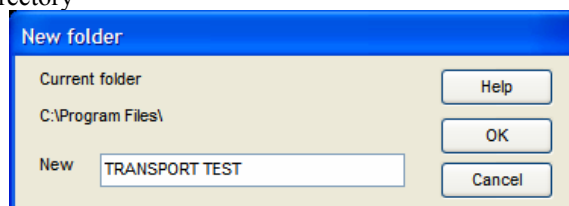


Figure 4.3 Creating a new directory



4.3. Starting the TRANSPORT model

GSE is copyright protected, i.e. when starting GSE you will get a screen like Figure 4.4. You can ignore this screen and press Continue>> to start GSE. This, however, will only work for 30 days. Within this time span you are requested to obtain a registration code. Getting a registration code is not difficult. First press Registration and the window shown on the right of Figure 4.4 is displayed. At the top right-hand corner of the screen you see a red installation code (CB058D in this example). This code is computer-unique, i.e. each computer has its own installation code and the registration code for one computer does not work on another. You can obtain your registration code either by going to the internet or by sending us an e-mail. If you use the e-mail feature, just

type your name and personal ID (which you received from NacquiT when you paid for GSE) in the two boxes and press [Click here](#) to send an e-mail. Your e-mail editor will start by inserting the necessary data. Just type in any other question/request you have and send the mail. We will send you the registration code as soon as possible. Obtaining the registration code by internet is simple and faster than by e-mail. Just press [Click here](#) to register GSE via the internet (Figure 4.5, <http://www.nacquit.com/register/register.exe>). Enter your name and personal ID code and press Register. The result will be a registration code (Figure 4.6) which you can type in (or, better still, copy and paste) in the registration box shown in Figure 4.4. When you press Register, GSE is registered and the registration window will never appear again when you start GSE.

Figure 4.4 Registration of GSE

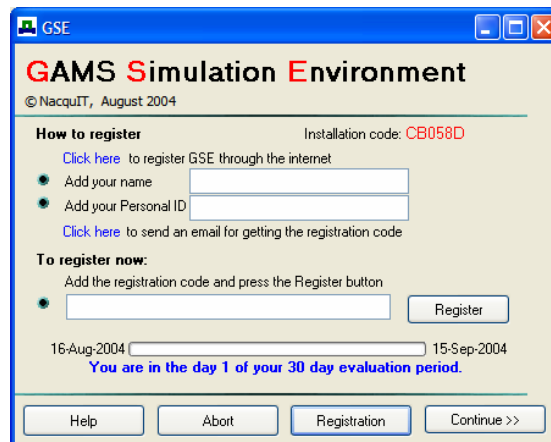
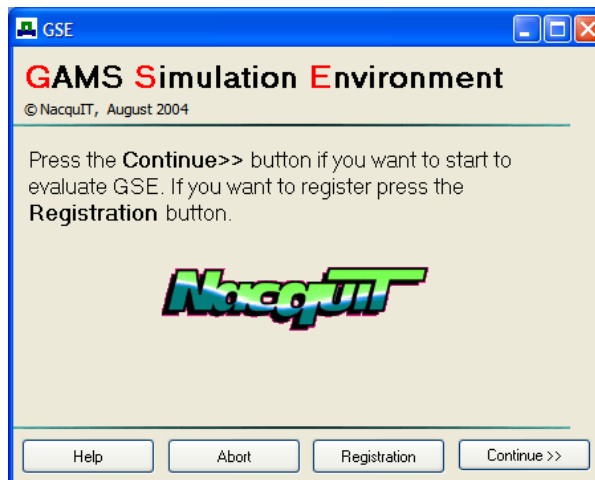
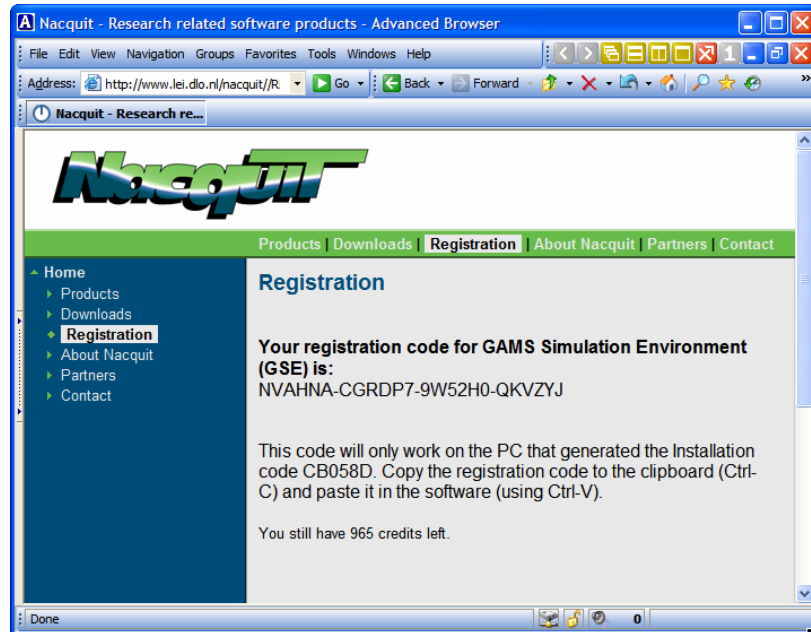


Figure 4.5 Internet registration of GSE



Figure 4.6 Registration code



4.4. Log on to GSE

GSE can be used for many models and by many people simultaneously. It is possible to run GSE on a network or on stand-alone systems. To protect your versions of the model and scenarios, you need a username and password to log on to GSE. When GSE is used on a network, any user can see and compare all the scenarios and models, but only the owner or creator of a version or scenario can delete it. When you start GSE you will see a window like the one in Figure 4.7.

Figure 4.7 Log on to GSE

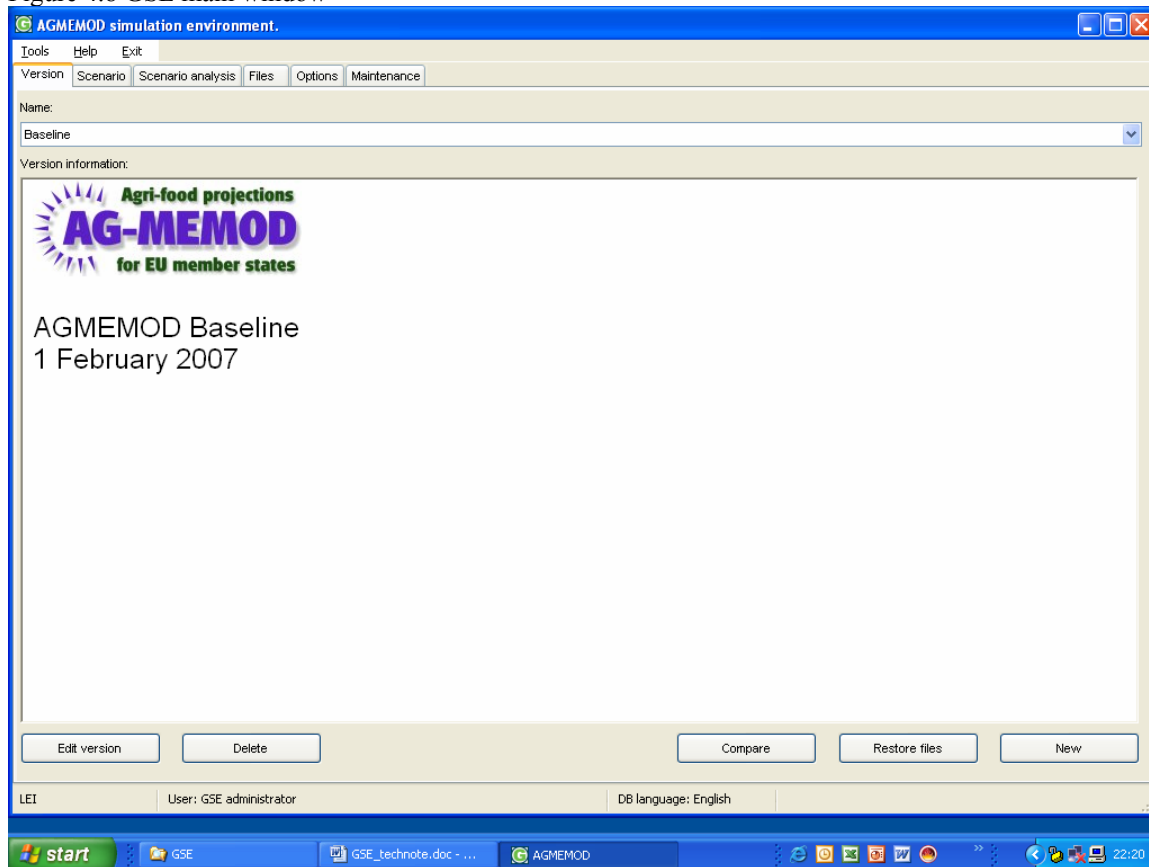


Please use the username and password supplied to you by the model-builder (probably written somewhere on the CD-ROM or case, e.g. for the Transport model you can use the user name **demo** and password **demo**). If you have GSE administrator rights you can add new users to the database yourself (without having to ask the model-builder for more usernames and passwords). Type in your username and password but do not check *Start in Check Model mode* (see the GSE manual for details of when to check this option) and press *OK*. The sections which follow will use the AGMEMOD model to demonstrate certain GSE functions. You can try out this function with the demos supplied by GSE (e.g. the TRANSPORT model).

4.5. GSE main window

After you log on to GSE you will see the main window like the one in Figure 4.8.

Figure 4.8 GSE main window



Note that the GSE window consists of six tab sheets:

1. Version
2. Scenario
3. Scenario analysis
4. Files
5. Options
6. Maintenance

The *Maintenance* tab sheet is available only if you have GSE administrator rights and is used to maintain the GSE database (e.g. add new users or restore deleted scenarios). The *Options* tab sheet is used to select the GUI language, the database language and other settings (for detailed information on these two tab sheets, see the GSE manual). The *Files* tab sheet is used to see and change the GAMS source files, i.e. is useful for a model-builder to build a new version. For you as a model-user tab sheets 1 to 3 are of interest. Before we discuss these sheets in more detail we have to explain what we mean by a version of a model and a scenario. In GSE we assume that you have the same version of the model, as long as you do not change the GAMS code apart from changing the input data. We would call changes to the input data a new scenario, but additions or changes to a model equation would be a new version of the model. Now that you know these definitions it is easy to see if you need to create and load a new version or if you can obtain the output you want from the model by just running a new scenario.

4.6. Model

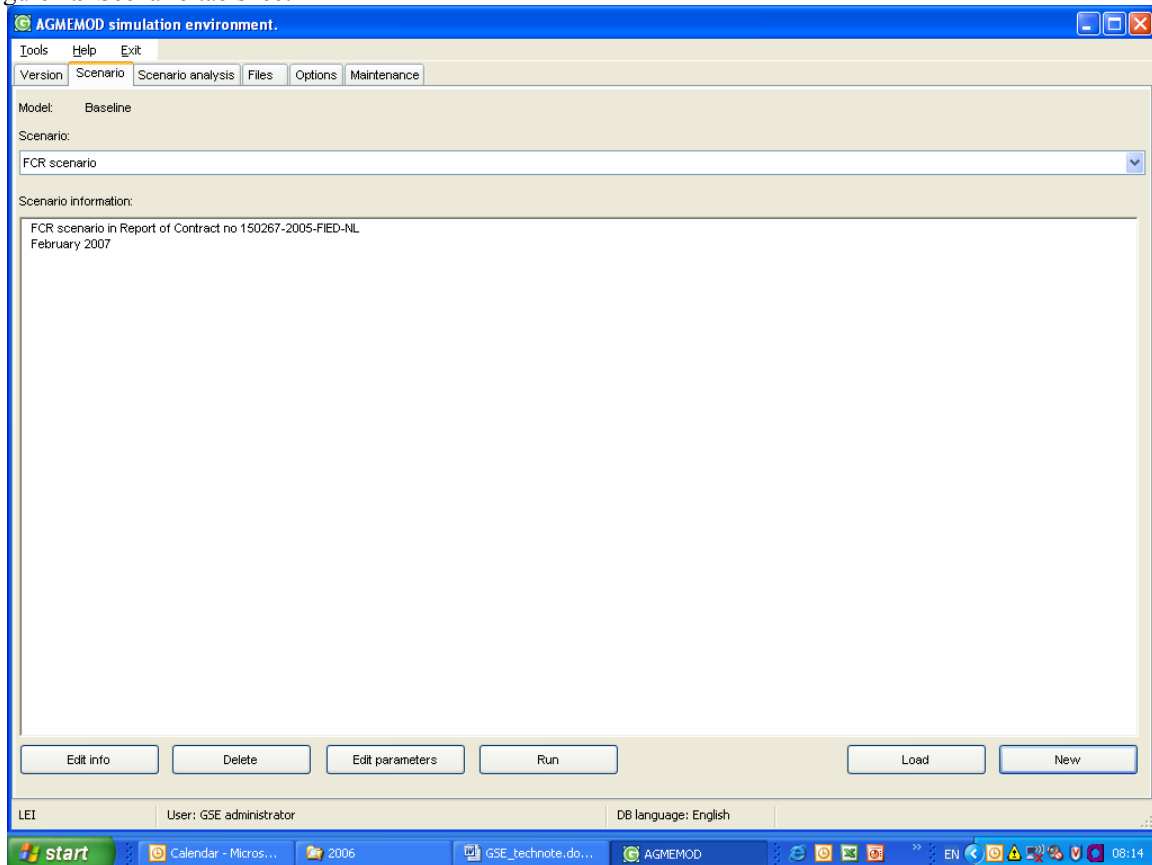
Often when the model-builder sends you the installation CD-ROM he or she will have added one or more versions of the model to the GSE database (usually only one). When you start GSE the window in Figure 4.8 will appear. With the Name combo box you can have a look at which versions are available. When you select a

version the Scenario tab sheet will show you a list of all the scenarios available for that model. Note that for every version the model-builder can add extra information to give the model-user an idea of what is important about this version (and perhaps what changes have been made from the previous version). The buttons on the bottom of the tab sheet are used to change the model version metadata, delete or add a version and restore the files used within a version to the hard disk. You can only delete a version you have created yourself, and hence the work of others is protected. Also note that when you delete a version or scenario the data are not really deleted from the GSE database but are merely inactivated. So if you delete a version or scenario by mistake, a GSE system administrator can restore the deleted data. Note: if you are not the owner of the version of the model, the Edit version button will be changed into Show version, indicating that you cannot change the version information.

4.7. Scenario

After selecting a version of the model the second tab sheet (Scenario) displays a list of all the scenarios built by that version. Whenever a new version is loaded into GSE the current input data are also stored in a Base scenario. Consequently, every version will have at least one (base) scenario available. Again, as with the model version, you can add further information on that scenario to an information field. When creating scenarios it is advisable to use this field to add information on the changes you have made to your input parameters and what you think you will simulate. An example of the Scenario tab sheet is shown in Figure 4.9. At the bottom of the screen there are six buttons. The first will be either Edit info or Show info (depending on whether or not you are the owner of the scenario). With this button you can edit/show the scenario information. With the second (Delete) button you can delete the scenario (a GSE system administrator can undo the delete for you), but, only if you are the owner. The third button will be Edit parameters or Show parameters. It is called Edit parameters if you have not run the GAMS model and are storing the output from the model to GSE. Until you have saved the output, you are allowed to change the scenario input data. Once you have run the model and saved the data Edit parameters will change into Show parameters, indicating that you can only view the input/output data but cannot change them any more. The fourth button is Run and will start the AGMEMOD model (i.e. run GAMS). After you have run the scenario you can view the results (input, output, LST files and GDX files) before deciding whether to save the outcome in the GSE database. After you have saved the scenario data in the GSE database you are no longer allowed to edit and run the scenario and, hence, can only inspect the scenario input and output. You could also decide not to save the scenario and to change some more parameters and run the scenario again. The fifth button will get the AGMEMOD source files from the GSE database and insert the input data of the current scenario into the files. The sixth and last button will create a new scenario (see the section on the steps for creating a new scenario). The most important button in this tab sheet will be Edit/Show parameters. This will start the GSE DataViewer/editor (explained in more detail in the section on the DataViewer/editor). Since you cannot edit a scenario once its output results have been stored, you need to create a new scenario that will import all the input data from the scenario you want to edit. This new scenario can be changed (since it has no output data).

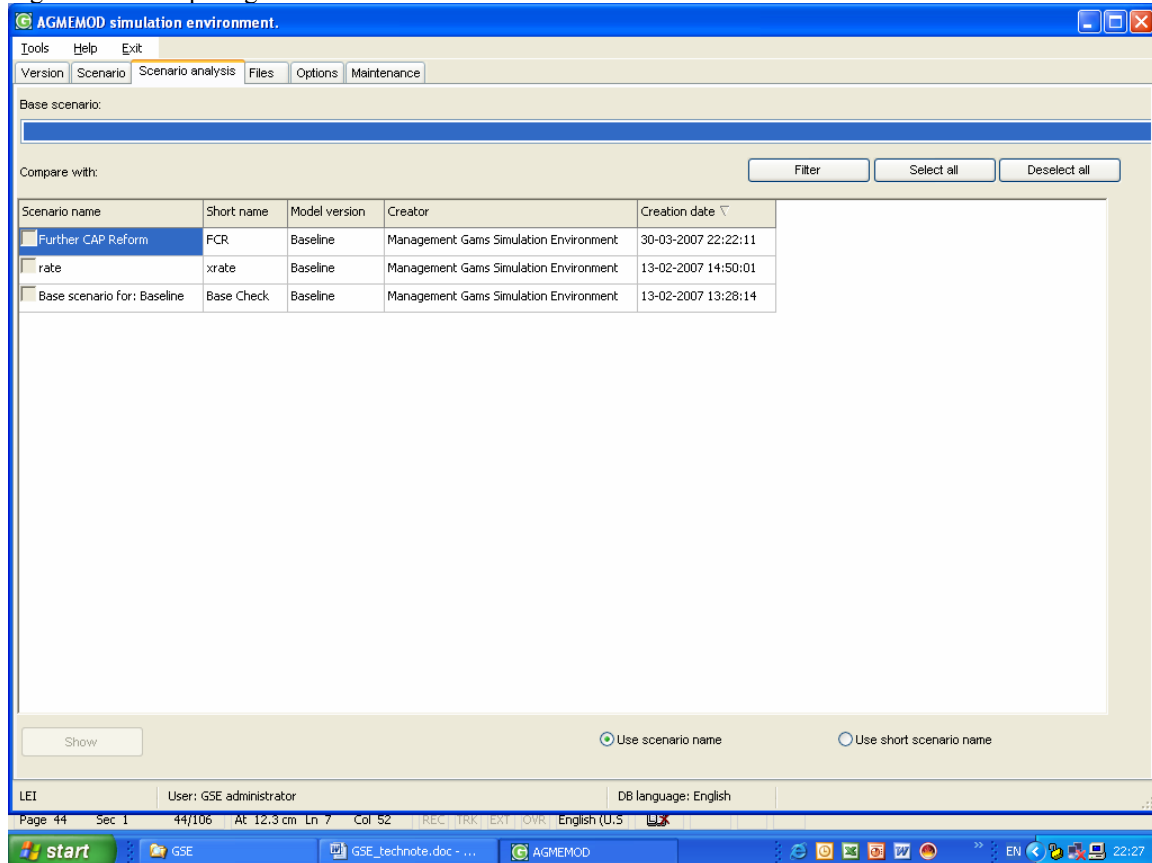
Figure 4.9 Scenario tab sheet



4.8. Scenario analysis

In the *Scenario analysis* tab sheet you can compare scenarios from any version. Once you have selected which scenarios you want to compare you can press *Show*. To help you select the scenarios you can order the scenarios grid by clicking on one of the column headers or by using the filter window (press *Filter*). When comparing scenarios, again the GSE DataViewer/editor will be shown. In this editor you can choose the same input and output parameters as with a normal scenario. When you select a parameter all the parameter data for all the selected scenarios are loaded and the parameter matrix is shown with the extra index (or set in GAMS notation) *Scenario*.






Figure 4.10 Comparing scenarios



4.9. The GSE DataViewer/editor

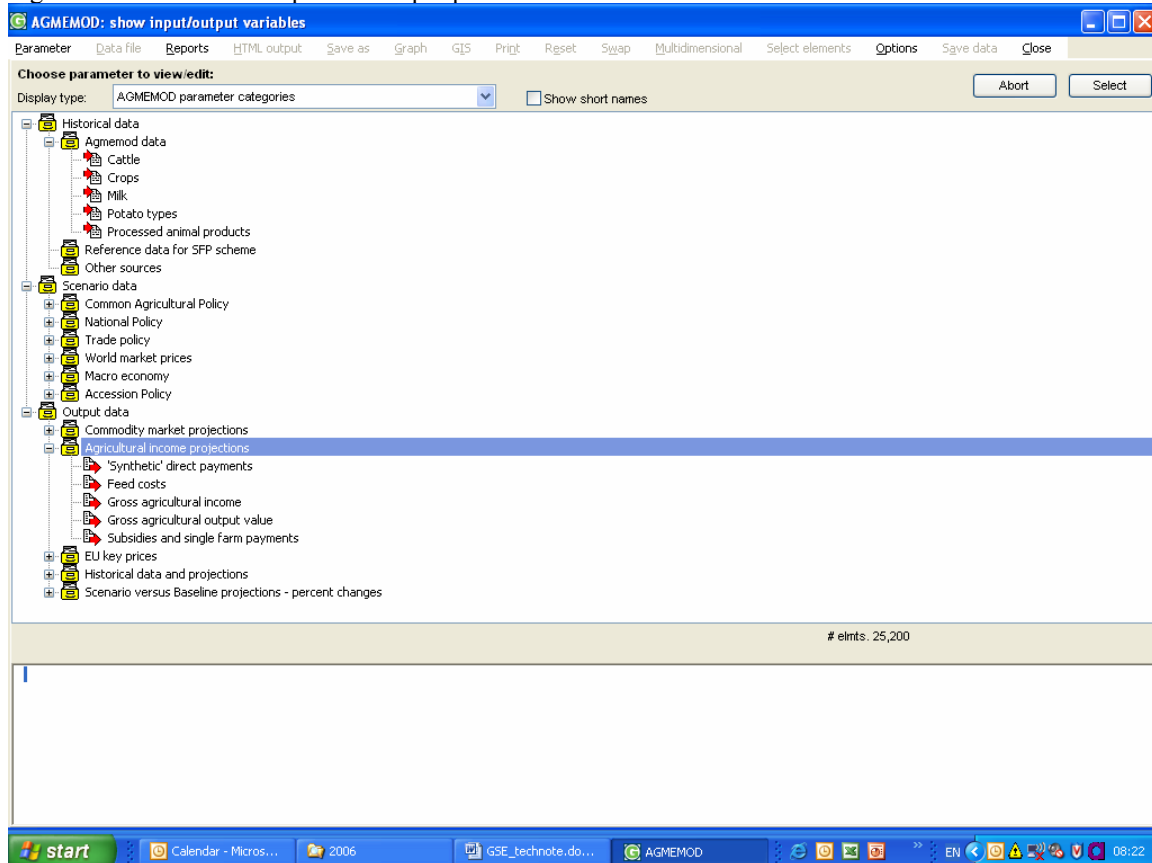
This is one of the most important windows in GSE. Model-builders are interested in how the model works (the scientific theory), IT specialists are interested in how the model is implemented (the mathematics and GAMS in our case) but as a model-user you are interested (only) in the **input** and **output**. The scalars and parameters of the current version of the model need to be presented in such a way that the GSE user can view, edit and compare the values. Since parameters are defined with many GAMS SETs, the outcome of a parameter cannot always be presented well in a two-dimensional world. This window offers you a look at your input and output, generating output on paper, HTML, graphics, etc.

In the top left-hand corner of the screen you can choose one of five types of parameter:

- ▶ Implicit output or Run output with external program defined by the  icon
- ▶ Show menu choices by the  icon
- ▶ Start external data editor by the  icon
- ▶ Input parameters indicated by the  icon
- ▶ Output parameters indicated by the  icon

Do not be frustrated if you do not understand these parameter types. Probably most users will only see/use the input and output parameters. By default the parameters are sorted by parameter type and within each type by alphabet. When you press the parameters combo box you can scroll down the list of all parameters and any additional information on a parameter is shown (Figure 4.11).

Figure 4.11 Screen with input and output parameters



After selecting the parameter, you will see the dimension of the parameter (when supplied using the `<%=DIMENSION%>` tag), the output type and which sets the parameter uses. The example below shows the input parameter *Crop*, with no dimension and the SETs (indices):

1. Countries
2. Crop activities
3. Crops
4. History period
5. Scenario

The value for *Crop* for the set values

Country = France

Crop activities = Area harvested

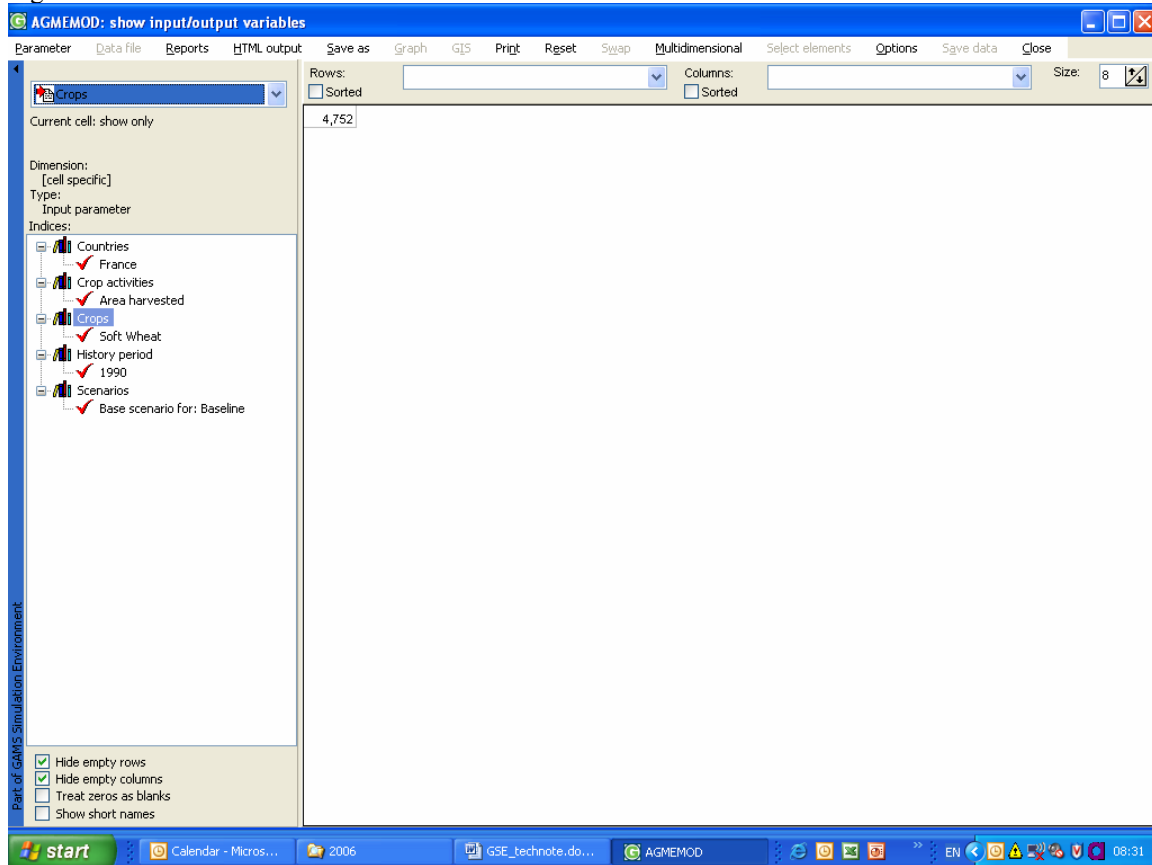
Crops = Soft wheat

History period = 1990

Scenario = Baseline

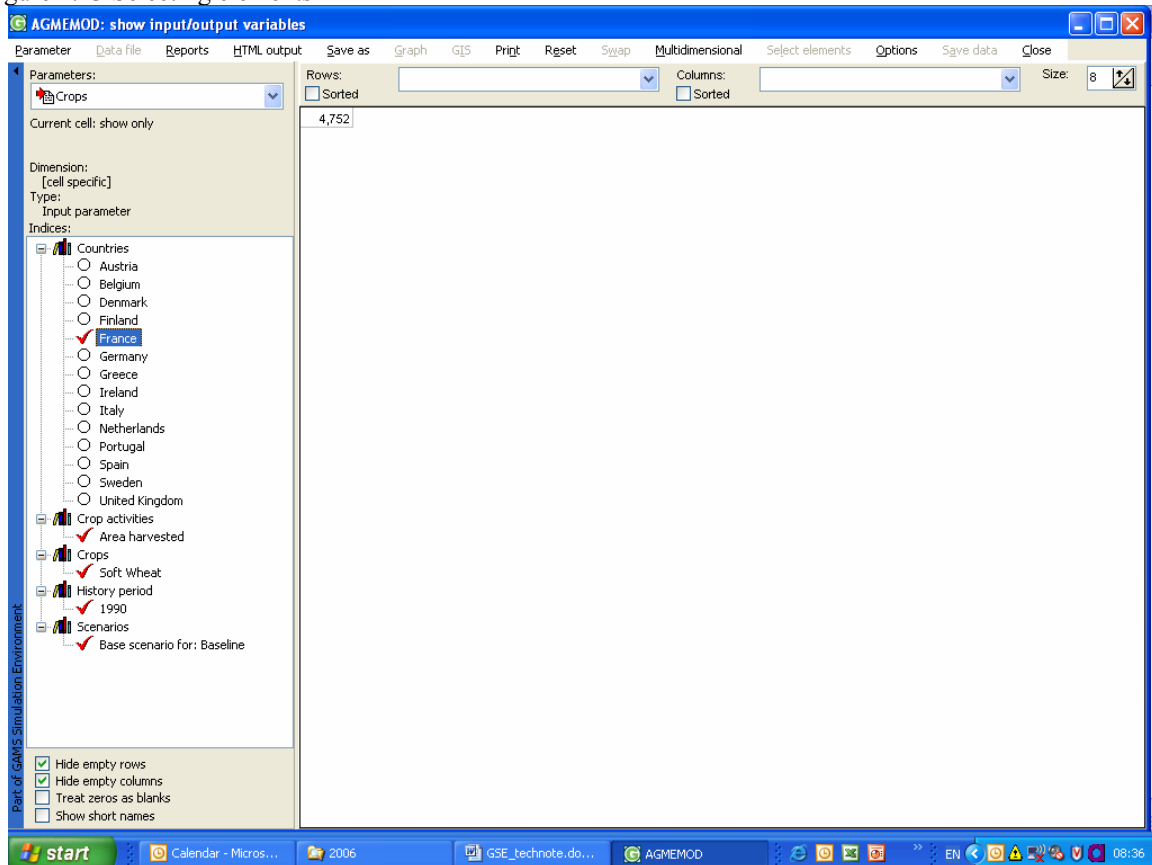
is 4.752. The colour of the value is defined in the INI file, and by default positive values are shown in blue and negative values in red.

Figure 4.12 GSE DataViewer



You can extract a different element from SET by clicking on a SET/index (e.g. countries). This will show you a tree of possible countries (all with a button). Just click on any radio button to select that element and the new value is shown in the grid.

Figure 4.13 Selecting elements



If you want to scroll through the elements in a SET/index you can use the mouse, e.g. by right-clicking on the *Countries* item in the tree *France* will change to *The Netherlands*. The next time you right-click you will get *Finland* and so on. Right-clicking on *Regions* with the mouse and holding down the Control key (Ctrl) will scroll backwards (e.g. from *Finland* to *France*).

Showing only one value is of little interest. By using the *Rows* and *Columns* combo boxes you can choose one or two SETs (indices) and present a table/grid. By default the SETs are displayed in the order in which the elements are defined in the GAMS files (see the section on sets, aliases and elements). If you check the Sorted checkbox you will sort the elements by alphabet.

Figure 4.14 Table view

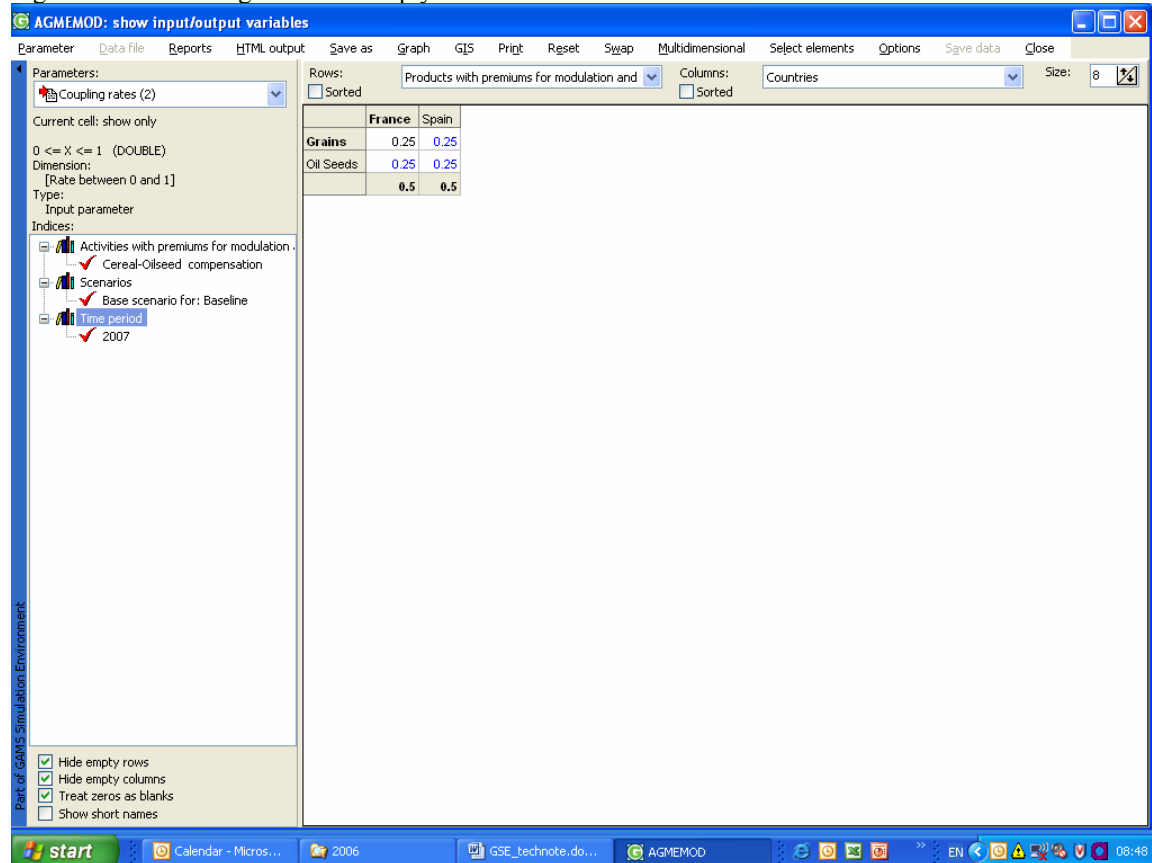
The screenshot shows the AGMEMOD software interface. The main window displays a table with the following data:

	France	Netherlands	Finland	Belgium	Italy	Germany	Spain	United Kingdom	Greece	Portugal	Denmark	Sweden	Ireland	Aus
Grains	0.25	0	0	0	0	0	0.25		0	0	0	0	0	0
Oil Seeds	0.25	0	0	0	0	0	0.25		0	0	0	0	0	0
Durum Wheat	0	0	0	0	0	0	0		0	0	0	0	0	0
Suckler Cows	0	0	0	0	0	0	0		0	0	0	0	0	0
Beef Cattle	0	0	0	0	0	0	0		0	0	0	0	0	0
Ewes	0	0	0	0	0	0	0		0	0	0	0	0	0
	0.5	0	0	0	0	0	0.5		0	0	0	0	0	0

The interface also shows a list of active scenarios on the left, including 'Activities with premiums for modulation', 'Cereal-Oilseed compensation', 'Scenarios', 'Base scenario for: Baseline', and 'Time period 2007'. The bottom status bar shows the time as 08:48.

SETs can contain a large number of elements that are not in use for the current parameter (resulting in empty cells). By checking *Hide empty rows* and *Hide empty columns* the empty rows and columns are removed and only the essential data are presented in a grid. The *Treat zeros as blanks* option will show zero values in the viewer as blanks (and with the *hide empty rows/columns* options on, they can disappear from the viewer, as seen in the example below). With the *Show short names* option you can show the parameters, sets and elements using their description or the original names used in the GAMS model.

Figure 4.15 Removing zeros and empty rows/columns

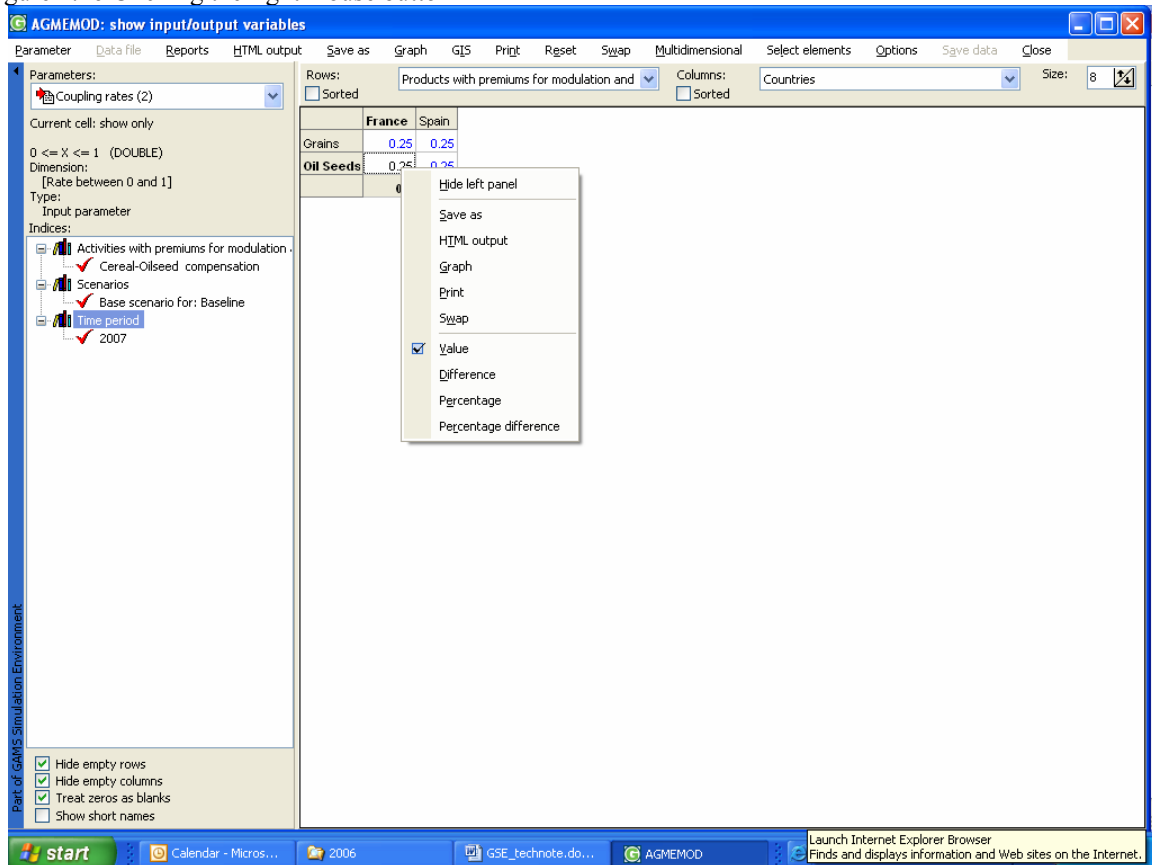


This GSE viewer offers you the following menu options:

- *Parameters*: will allow you easily to select a parameter.
- *HTML output*: will pop up the HTML output window.
- *Graph*: will open the Graph output window.
- *Print*: will show the Print preview window.
- *Save as*: will offer you the choice of saving the grid to Word, Excel, ASCII, HTML or CSV.
- *Reset*: will restore the parameter presentation to its original settings (all SETs in the Indices tree).
- *Swap*: will switch the columns and rows of your grid.
- The options *Multidimensional grid*, *Select elements* and *Options* are explained in more detail in the GSE manual and are useful for complex (multidimensional) data exploration.
- *Save data*: If you are allowed to change data (input parameters only and only when the scenario is not write protected) and have changed some of the values, the *Save data* option will be enabled and you can press this button to save the changes to the database.
- *Close*: will return to the main window.

If you right-click on the data grid the following mouse menu will be shown :

Figure 4.16 Clicking the right mouse button



GSE makes it possible to add comments to every parameter value. This is useful when a scenario analyst changes a certain parameter value and wants to add further information about why this value was changed. By selecting *Edit cell information* you can edit the additional information on a specific cell/parameter value. You can see if a cell has information by looking at whether a value is underlined/hyperlinked. *Hide left panel* will hide the left-hand side of the screen and show you only the data grid. This is useful if you have more columns than are visible on the screen. When you right-click on the grid you can see the left-hand panel again. The *Output*, *Print*, *Save as*, *Graph* and *Swap* options are the same as the corresponding buttons and are especially useful when the left-hand panel is hidden.

By clicking on a row or column header you can select a row/column. By keeping the left mouse button pressed and moving the mouse you can select multiple rows/columns. You can select the whole table by clicking on the top left-hand cell. By pressing the Control key (Ctrl) and then clicking on a cell you can select a cell (or multiple cells by keeping the left mouse button/ctrl key pressed). After selecting cells you can perform certain spreadsheet tasks on the cells selected (multiply, add, divide or change value) and, hence, you do not have to change all the cells separately.

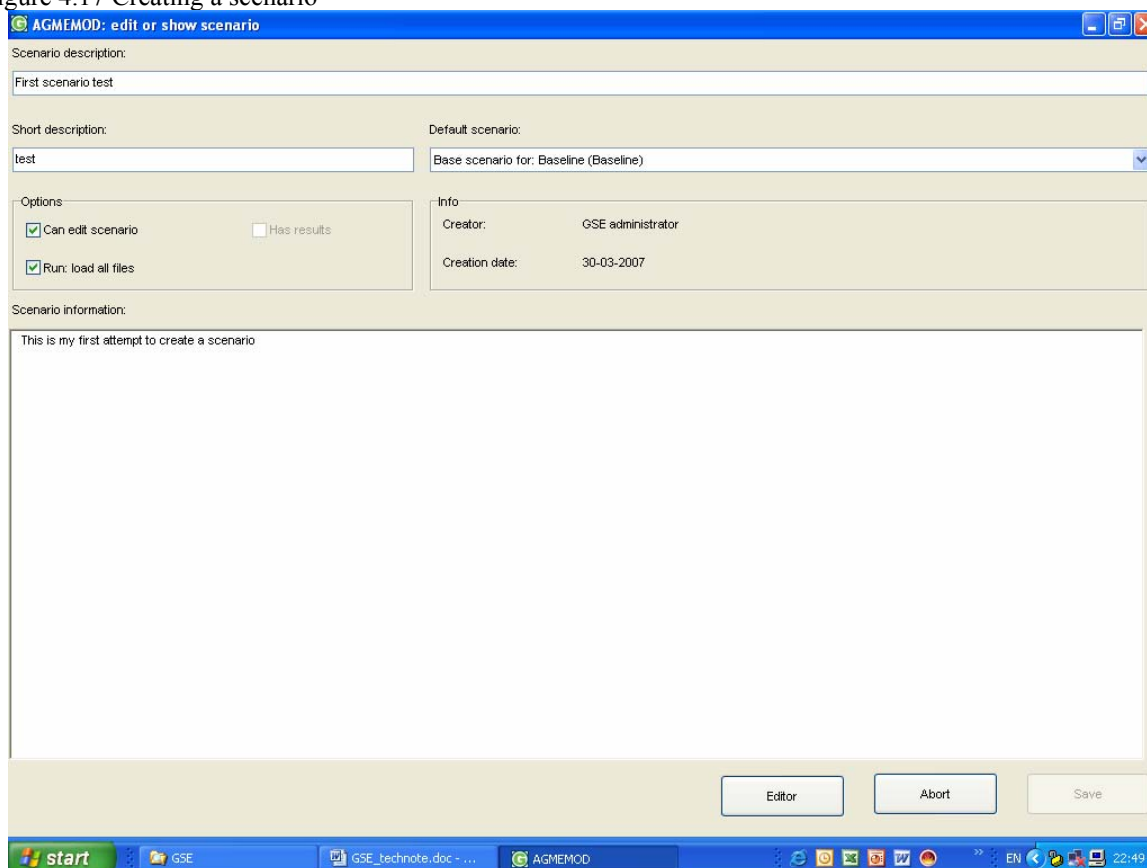
The last three options are useful when comparing scenarios. By default, the values of the data are shown. When comparing scenarios with a column for every scenario value, *Difference* will show the difference between the scenario and the base scenario and *Percentage* the percentage difference between them.

4.10. Steps for creating a new scenario

So you have installed AGMEMOD. Well done! You have discovered that one version of the model and one scenario (the base scenario) are available and that you can only look at the input and output data (since you are not the owner of the version and scenario). What can you do with GSE? How can you run scenarios? Just read the text below and you will be able to start running and comparing scenarios.

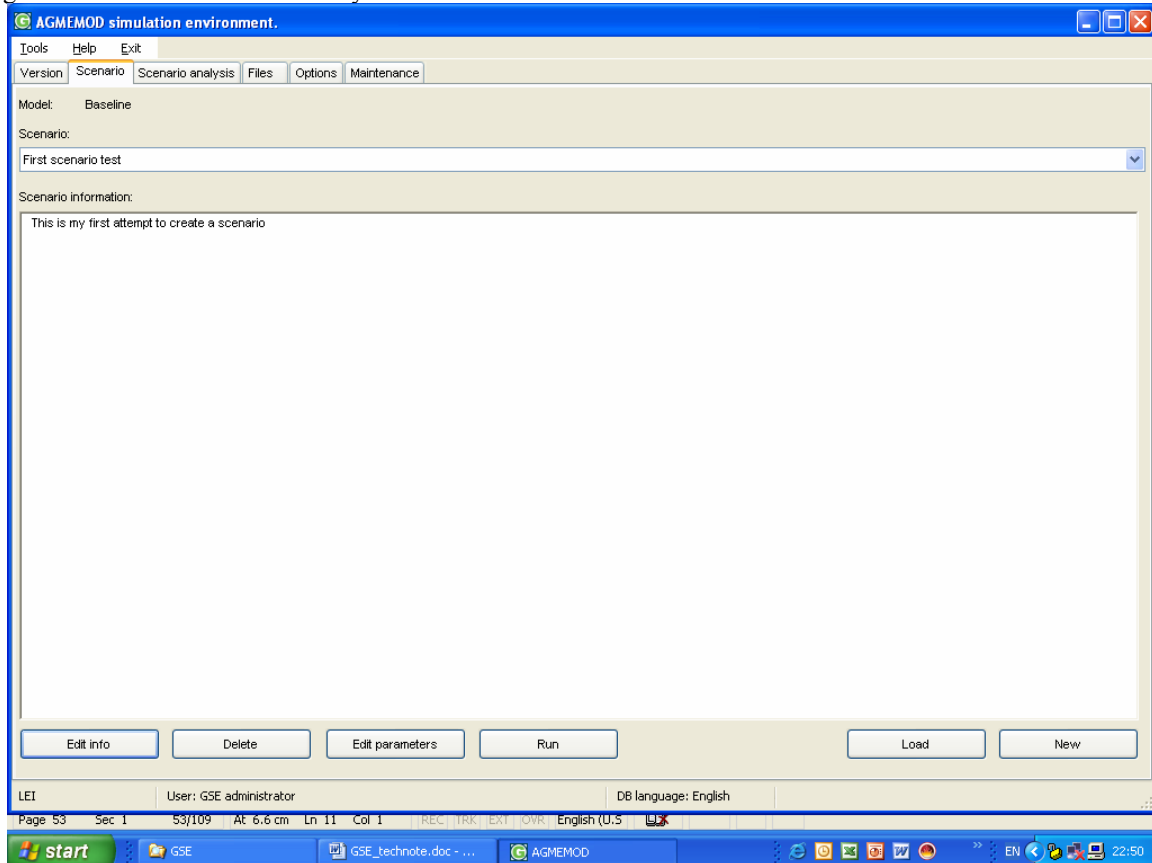
Since you want to create your own scenarios you go to the Scenario tab sheet and press New. The scenario editor will pop up (Figure 4.17).

Figure 4.17 Creating a scenario



In the Scenario description you enter the text shown, for example, in the Scenario tab sheet where you can select between the scenarios. The short description can be used when you compare scenarios (and do not want the description to be too long for certain output). When you create a new scenario you do not want to type in all the required input data. So there is a clever trick in GSE called “scenario inheritance”. In the Default scenario combo box you can select any of the scenarios previously run and saved (even from previous versions of the model). Choosing Base scenario for: AGMEMOD version 1.0 (AGMEMOD version 1.0) simply means that all the data from the Base scenario are taken as the starting point for the new First scenario test scenario. Since you can select any scenario previously saved, you can try to select a Default scenario that will minimise the changes you need to make to get the scenario running. Since GSE cleverly stores only changes in the new scenario, as compared with the default scenario, the GSE database also remains small. Editor can be used to call the GSE editor, so you can add complex Scenario information (e.g. with pictures and colour). Do not forget to use the Scenario information. It will help you later on to understand what you have done with a certain scenario and why you have generated it. Press Save and you have created a new scenario and are back in the Scenario tab sheet of the main GSE window (Figure 4.18).

Figure 4.18 The new scenario ready to be used



GSE has preselected the new scenario for you and now you can use *Edit parameters* to go to the DataViewer/editor and change the parameters you need to change. After you have saved these changes you can press *Run* to start AGMEMOD and when there are no errors you can press *Save* to save the output results. Congratulations. You have created your first scenario and we are sure many more will follow.

Final remarks

You now have a first impression of how to work with GSE. Try out the program and play with the TRANSPORT model. If you would like more functions, have a closer look at the manual or, when you are in GSE, just press the F1 key to obtain help about the current window. If the model is unclear to you or you get lost in the definition of a parameter, etc. just ask your model-builder to add some extra GSE tags to the next version. GSE offers numerous functions that make it possible to transfer knowledge from the model-builder to the users (e.g. by specifying upper and lower bounds, by adding extra information to a parameter or version of a model, by ordering the parameters, etc.). For the latest information and support do not hesitate to contact us:

info@nacquit.com

www.nacquit.com

5. Quick reference guide to GSE for new versions of the model

5.1. Introduction

You have finished reading and trying out the two previous “Quick reference guides”. The first is for model-builders and explains how a GAMS model is imported into GSE using the Import2GSE wizard. The second is especially for model-users and explains briefly some of the possibilities offered by GSE to change model parameters and run and compare scenarios. You have been through the basics of GSE and, by now, should have confidence that GSE can help you with your modelling work. Imagine you have used Import2GSE and imported the TRANSPORT model into GSE. The next step is that, after running a few scenarios, the model-builder will discover that certain points of the model need to be improved. So we want to change the GAMS code. The question now is: how can we do this within the GSE context? This quick reference guide will tell you how to make new versions of the model and install them into GSE.

5.2. Version or scenario?

There are many definitions of the differences between versions of a model and scenarios. GSE uses a very simple one. If you only change input data (parameters and scalars) and run the same mathematical model you have a new scenario. If you want to change not only the values of the parameters but also the actual mathematical model (and, hence, the GAMS code, e.g. by adding or changing equations or methods of calculation) you want a new version of the model.

5.3. A new version

Importing the GAMS TRANSPORT model into GSE with the Import2GSE wizard will take your source model (and other subdirectories) and put it into the `c:\program files\gse\models\TRANSPORT\source` directory. Note that GSE will store all its models (by default) in the `c:\program files\gse\models` directory and creates a separate subdirectory with the model name (TRANSPORT) for all new models and then uses two special subdirectories. The `..\models\TRANSPORT\SOURCE` directory will contain the original GAMS code and, after you have used Gtree (see Quick reference guide for model-builders), this GAMS code will contain GSE tags. When you press *New Version* in GSE and have entered version information press *Save*: all the files that are stored in `..\models\TRANSPORT\SOURCE` are parsed by GSE and stored in the GSE database. When using GSE and running a scenario, all the files and data are taken from the GSE database and stored in a special directory `..\models\TRANSPORT\RUN`. This GAMS code is then executed and the results can be saved as a new scenario. Using this approach means that GSE will use the SOURCE directory only when a new version of the model is loaded and, hence, allows you (the model-builder) to change the GAMS code in this directory and add/change GSE tags. With Gtree this is easy. Before you add your new version to GSE remember to perform the two following steps:

1. Check if the new model runs under GAMS (without any error), e.g. by selecting the “Run project file (F9)” menu item in Gtree.
2. Check if you have entered/changed the new GSE tags correctly by running GSE in the Check-only mode (or select the “Check GSE tags” menu item under Gtree).

Since the files in the SOURCE directory are GAMS files with a few comments (tags) added, you can use any program you like to edit/change the code. You can use GMSIDE, notepad or Gtree. To help you a little, we have added the *Files* tab sheet in GSE. Here you can browse the RUN or SOURCE directories and change the code. Note that it does not matter which program you use, as long as you perform the two checks above before you add a new version of the model to GSE.

6. Tips for GAMS models

□ Divide the model into parts and use the *\$include* statement in GAMS. Parts should be meaningful. Give the file a clear name, since this makes it easier to read the model. You know where to find something but others do not. Sharing knowledge becomes easier when you have a good tree structure. You should use a default tree structure for (almost) all your models, e.g. for Micro Simulation models (MicroWave) the following tree is used:

- 1 Sets and Elements.set
- 2 Declarations.gms
 - 2.1 Stateblock Parameters.par
 - 2.2 Exogenous.par
 - 2.2.1 Counters.par
 - 2.2.2 Constants.par
 - 2.2.3 Standards.par
 - 2.2.4 Trends.par
 - 2.2.5 Policy.par
 - 2.2.6 Model Initials.par
 - 2.3 Endogenous.par
 - 2.3.1 Model Parameters.par
 - 2.3.2 Variables.par
 - 2.3.3 Aggregation Parameters.par
 - 2.4 Equations.equ
 - 2.5 Models.mod
 - 2.6 Other.par
- 3 Import data.inp
 - 3.1 Loop Initialisation.gms
 - 3.2 Data Initialisation.gms
 - 3.3 Initialisation schemes.gms
- 4 MicroWave Model.gms
 - 4.1 Base Period Initialisation.gms
 - 4.2 Transfer from Statblock to Begin Period Model.gms
 - 4.3 Period Model.gms
 - 4.4 Transfer from End Period Model to Stateblock.gms
 - 4.5 Aggregation.gms
- 5 Output.gms

□ Do not use GAMS restart files but use a GDX file instead (in this GDX file save only the parameters, variables and equations you need for a restart). GDX is much faster.

□ Use comments, e.g. specify where your base data come from (are they from a(n) (external) database, did you manipulate them, etc.). When using estimated equations you could give a hyperlink to the documentation and specify useful statistics such as the R-square and T-values, e.g.:

```
=====
* Equation 3:
* In this equation we estimate the Harvested Barley in
* the Netherlands. The exogenous data (V2) come from
* the FADN data of LEI. For documentation on the
* estimated equation see:
* file://CropsEstimatesInTheNetherlands.doc
=====
CropHarvested(P,A,C,T)$ (sameas(C,"NL") and
    sameas(P,"BA") and
    sameas(A,"AHA") and
    (ord(T) GT ST) ) ..
*-----
V2(P,A,C,T) =e= EST(P,A,C,"1")
    * V2(P,"ASH",C,T)
    * V2("G3",A,C,T)
=====
```

□ Use meaningful names for sets, parameters, variables, equations and models, e.g.

Country instead of *C*,

Product instead of *P*.

Note that the names are singular not plural because this makes it easier to read parameters, variables and equations.

□ Use upper- and lower-case protocol for words:

ESTIMATEDPRICEOFBARLEY v. *EstimatedPriceOfBarley*

□ Give elements a *unique* name (to make it clear in equations what you mean) and a long name as well: in an

equation does *PT* stand for *Potato* or *Portugal*? *Netherlands* is better than *NL*. Do not give elements the same name as a set, parameter, variable or equation.

□ Naming conventions for sets, parameters, variables, equations and models can be very useful, e.g. all sets start with an *I*, so we get *iCountry*. For aliases we use the letter *j*, so *jCountry* is an alias for *iCountry*. For parameters, variables and equations you can use a prefix to indicate which group a parameter belongs to, e.g. all starting with *p* will be prices: *pBarleyHarvestedNetherLands* or, better still, *pestBarleyHarvestedNetherlands* will indicate an estimated (*est*) price (*p*) for Barley harvested in the Netherlands.

□ Try to use GDX both as input and as output (it is very fast, e.g. reading AGMEMOD spreadsheets takes 35 seconds, whereas reading the GDX file takes 3 seconds).

□ Use `$SetGlobal` to put all the options you have in your files. Put all `$SetGlobals` in one file, e.g. many people use comments to switch between options.

```
* OPTION PROFILE = 3
* OPTION PROFILETOL = 1
```

Better:

```
* set this global variable to yes if you want debug information in your list file
* set this global variable to no if you do not want debug information
$SetGlobal DebugModel no
```

```
$if "%DebugModel%"=="yes" OPTION PROFILE = 3
$if "%DebugModel%"=="yes" OPTION PROFILETOL = 1
```

□ Use file headers for every file, e.g.:

```
=====
* File      : Settings.gms
* Author   : Foppe Bouma(foppe.bouma@wur.nl)
* Version  : 1.0
* Date     : 12/18/2005 11:17:39 AM
* Changed  : 05-Mar-06 10:26:52
* Changed by: Wietse Dol (w.dol@wur.nl)
=====
* For some tips on Good Modelling Practice see
* file://ModelTips.doc
```

□ Use a good history list.

```
=====
* File      : Settings.gms
* Author   : Foppe Bouma(foppe.bouma@wur.nl)
* Version  : 1.0
* Date     : 12/18/2005 11:17:39 AM
* Changed  : 05-Mar-06 10:26:52
* Changed by: Wietse Dol (w.dol@wur.nl)
=====
* For some tips on Good Modelling Practice see
* file://ModelTips.doc
=====
* Remark by Wietse Dol (w.dol@wur.nl) at 05-Mar-06 10:26:52
* I have added the SetGlobal Debug to make it possible
* to have additional information in the GAMS list file.
* This is helpful during the development of the model.
=====
* Remark by Foppe Bouma(foppe.bouma@wur.nl) at 12/18/2005 11:17:39 AM
* I have added the SetGlobal RunOption
* BaseLine: Running BaseLine from Base data
* from excel, storing them in GDX and saving Solution
* Scenario: Running Scenario from GDX BaseDataFile and loading Solution
=====
```

□ In equations do not use values but use clear names that indicate why this value is there, e.g. in the equation

```
EQWSUFCD(T+1) ..
V2("WS","UFC","DE",T+1)=e=
EST("WS","UFC","DE","1")
+EST("WS","UFC","DE","2") * V2("WS","PFM","DE",T+1)/VMAC("GDPD","DE",T+1)
+EST("WS","UFC","DE","3") * 1000*VMAC("RGDPD","DE",T+1)/VMAC("POP","DE",T+1)
```

```
+EST("WS","UFC","DE","4") * (TREND70(T+1)+1)
+EST("WS","UFC","DE","5") * DUM("D90",T+1)
```

□ It is better to define a scalar :

Scalar BiljonToMiljon /1000/;

```
EQWSUFCDE(T+1) ..
V2("WS","UFC","DE",T+1) =e=
EST("WS","UFC","DE","1")
+EST("WS","UFC","DE","2") * V2("WS","PFM","DE",T+1)/VMAC("GDPD","DE",T+1)
+EST("WS","UFC","DE","3") * BiljonToMiljon*VMAC("RGDPD","DE",T+1)/VMAC("POP","DE",T+1)
+EST("WS","UFC","DE","4") * (TREND70(T+1)+1)
+EST("WS","UFC","DE","5") * DUM("D90",T+1)
```

□ Create a “to do” list. This will make clear what needs to be done before you have the next version of the model.

□ To structure input, output and equations use subsets.

7. GAMS tree

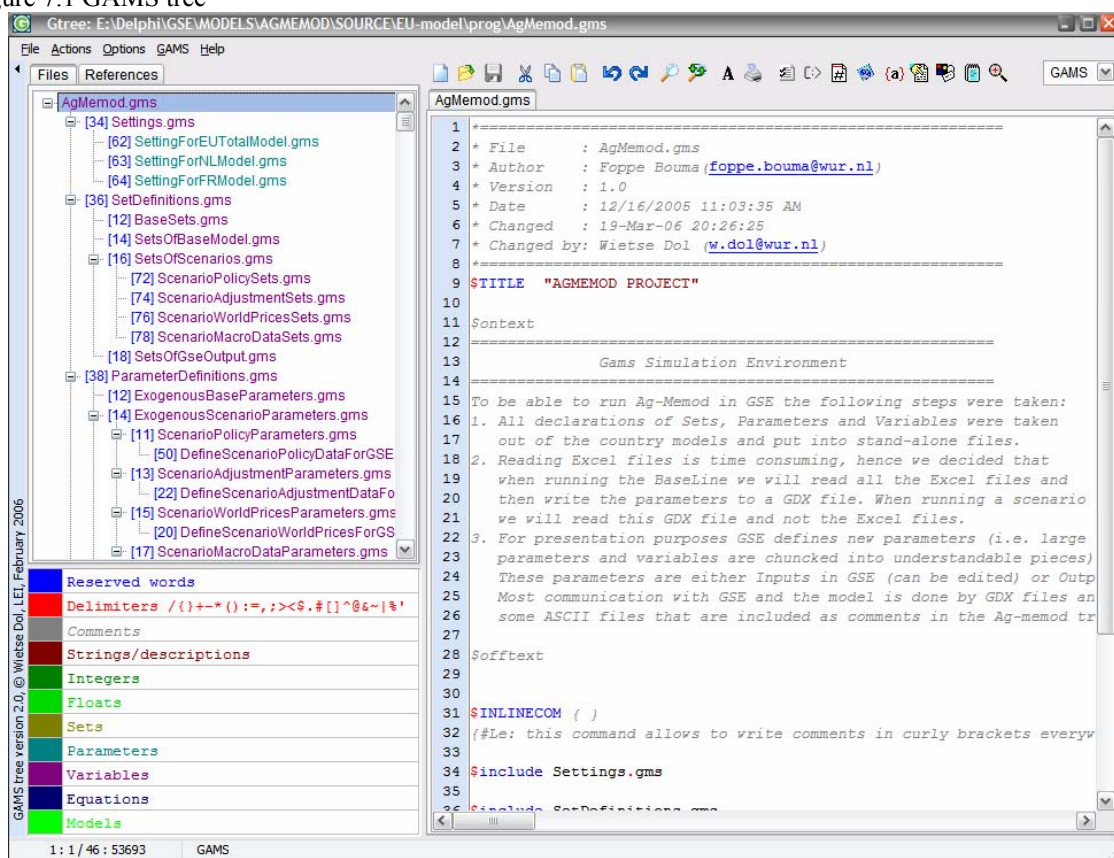
7.1. Introduction to Gtree

Gtree (GAMS tree) is part of GSE and written in the programming language Delphi. Gtree will show, in a tree, the structure of your GAMS code (the result of all the `$include`, `$batclude`, `$libinclude` and `$sysinclude` statements). By just clicking on the tree you will see the contents of the files as well. We think it is useful not only for having a look at other people's work but also to show a good model structure to others. This is what was missing in GAMSIDE. People who had tried to use it came to us with many requests for extensions of the program. By responding to them we turned into a serious competitor to GAMSIDE.

The first time you start Gtree the program will scan your hard disk for GAMS. Be patient, Gtree will do this only the first time. After that, it will remember where you have stored GAMS. Gtree needs to know where GAMS is, because it offers you the option to run GAMS code and see the GAMS documentation, etc. If you do not like this scanning you can edit the Gtree.INI file and change the GAMSdir by hand.

Gtree should be easy to use and many things need no explanation. We will, however, give you information which we think is useful if you are to become an expert in using Gtree. When you load a file into the Gtree editor the file is scanned for `$include`, `$batclude`, `$libinclude` and `$sysinclude` statements and, when this command is available, the file that is `$included`, `$batincluded`, `$libincluded` or `$sysincluded` is also read and scanned for `$include`, `$batclude`, `$libinclude` and `$sysinclude` statements. As a result of this scanning you will get a tree which makes it visible where and how files are related Figure 7.1.

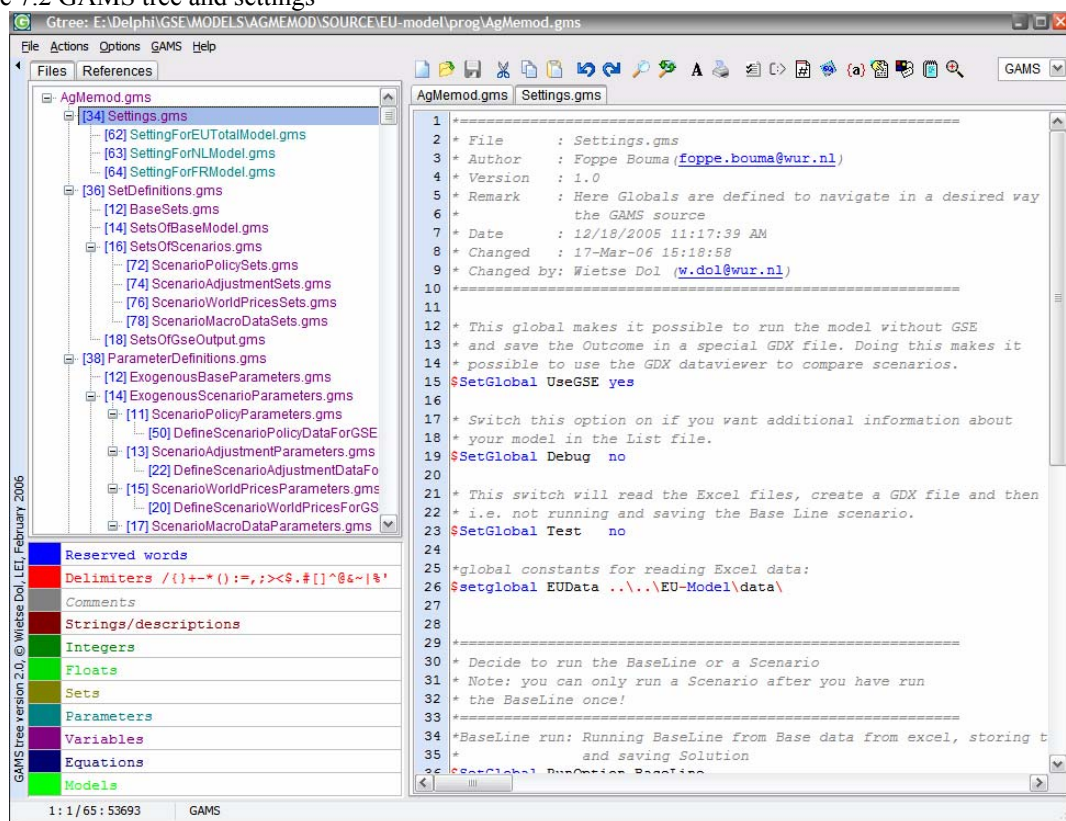
Figure 7.1 GAMS tree



In the example above we can see, for example, which files are included in file Settings.gms and we see the contents of file AGMEMOD.gms in the editor. The [62] before the file SettingForEUTotalModel.gms simply means that on line 62 of file Settings.gms the `$include SettingForEUTotalModel.gms` statement includes this

file. Browse through the tree and you will see that when a file is included, the editor contains two tab sheets on the right-hand side of the window: one with the selected file and one with its parent file (i.e. the one that is included in the selected file). The teal colour of file `SettingForEUTotalModel.gms` indicates that it is not a normal `$include` but a conditional include (in our case you will see the line `$if "%EU_TotalRun%" == "yes" $include SettingForEUTotalModel.gms` on line 62 of the `Settings.gms` file) Figure 7.2.

Figure 7.2 GAMS tree and settings



In the status bar at the bottom of the screen we see the cursor position and file size of the current file (1 : 1 / 65 : 53693 means: column 1 of line 1, the file is 65 lines long and the total model (all files) is 53693 lines long). Then we see that Gtree knows that you are working on files that use GAMS (Gtree is also capable of reading other environments, e.g. SCENTAB). The right-hand side of the status bar is used to show errors/remarks, e.g. when running GAMS.

On the left-hand side of the Gtree window two tab sheets can be seen: *Files* will show you a tree structure of your model (generated from all `$include` commands) and *References* will show the structure of the GAMS model (see reference file).

One very important function in the editor is the syntax colouring, i.e. the colour indicates the function of the word. Just to help you remember the function of the colour a legend is shown in the lower left-hand side of the window (e.g. reserved words are shown in blue).

If you show the tree not much space will be left for the editor. You can collapse/hide the tree by clicking on the panel with the copyright statement. By clicking again the tree will reappear (N.B. you can use Ctrl-M as a keyboard shortcut).

At the top of the screen there is a menu with the options File, Actions, Options, GAMS, About and Help. Their contents and purpose and the keyboard shortcuts can be used to choose from the menu will be explained below.

7.1.1. File menu

The File menu (Figure 7.3) offers the following choices in Table 7.1.

Figure 7.3 File menu

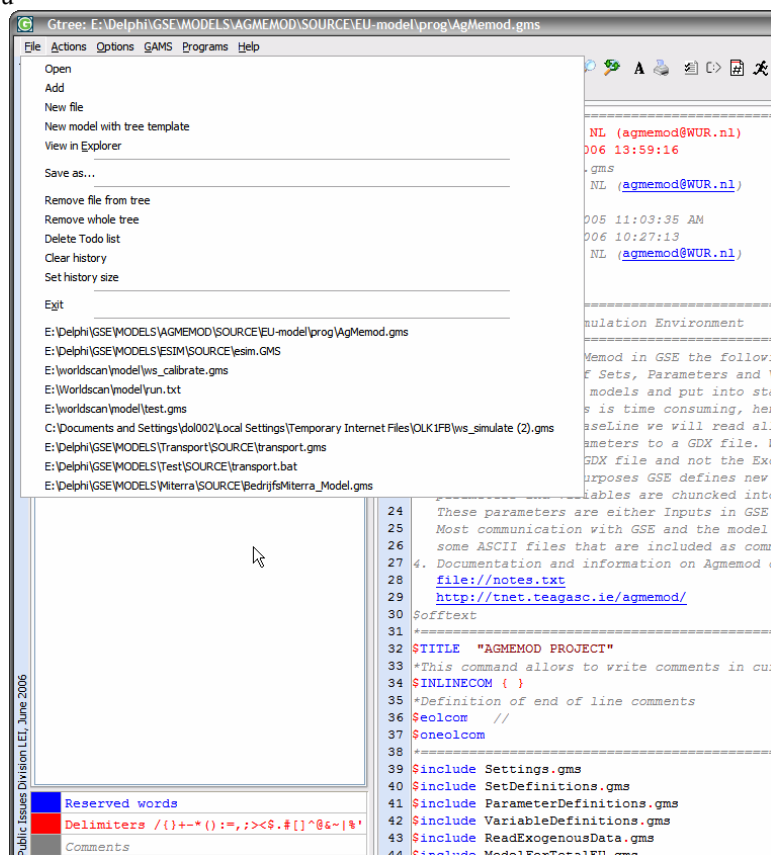


Table 7.1 Choices in the file menu

Open	Empty the left tree pane and ask for a file to be loaded into Gtree, load the file and scan for include statements. The result is a tree like the examples shown above.
Add	This option keeps the files (tree) already loaded and adds another file to the tree.
New file	Creates a new model.
New model with tree template	Creates a new model tree with the aid of a template: see New file with tree template.
View in Explorer	Starts the Windows explorer in the model directory.
Save as...	Saves the current file under a new name.
Remove file from tree	Deletes the file from the tree.
Remove whole tree	Clears the whole tree and removes all the files from the tree.
Delete to do list	Removes the model "to do" list.
Clear history list	Gtree will remember which files you have loaded and keeps a history list. Click on this menu item if you want to clear this history list.
Set history size	By default Gtree will remember the last 10 files you have opened with Gtree. If you want a longer or shorter history list click on this item.
Exit	Closes and quits Gtree
--- History of last loaded files ----	When you click on a file on this list the tree is cleared and the selected file is loaded (as in Load to tree).

7.1.2. Actions menu

The Actions menu (Figure 7.4) offers the following choices (Table 7.2).

Figure 7.4 Actions menu

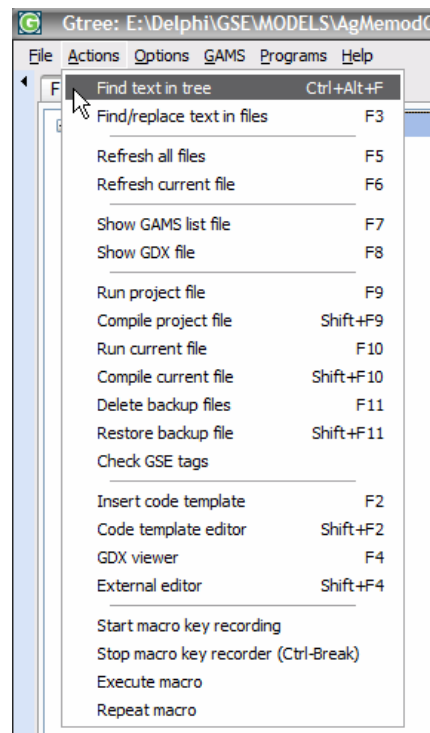


Table 7.2 Choices in the actions menu

Find text in tree (Ctrl-Alt-F)	If you wish to search for a certain filename in the tree, click on this item on the menu. You will then be asked to specify a search text and Gtree will search the tree and present the outcome of the search in a window. See: search a filename.
Find/replace text in files (F3)	If you need to find or replace certain text in one or more files in the tree, click on this item. See: search/replace text in files.
Refresh all files (F5)	If you have made numerous changes to your files (e.g. added or changed \$include statements), click on this item and all the files will be scanned and a new tree built.
Refresh current file (F6)	If you have added/changed \$include statements or GSE tags in the currently selected file, click on this item and the current file will be scanned for \$include and GSE tags and the tree will be rebuilt. Note that this option is much faster than the “Refresh all files” option and is probably the one you will use most of the time. Note that Gtree knows the date of the file and if you have saved a newer version Gtree can automatically scan the updated file (see: Refresh file when selected in tree).
Show GAMS list file (F7)	Shows a list of available GAMS LSTfiles. If there is only one LSTfile, this file will be shown immediately.
Show GDX file (F8)	Shows a list of GAMS GDX files. If there is only one GDX file, this file will be shown immediately.
Run project file (F9)	Runs the files that are at the root of the tree (in our examples file AgMemod.gms). Gtree will have a list of which programs are used for certain file extensions. If no extension/program combination is specified, the windows defaults are taken. See Gtree INI file.
Compile project file (Shift-F9)	Compiles the root of the tree, i.e. runs GAMS and checks the model but does NOT start the solver.
Run current file (F10)	Runs the currently selected file.
Compile current file (Shift-F10)	Compiles the current file.
Delete backup files (F11)	Selects and deletes backup files and other files (see: Cleanup files).
Restore backup file (Shift-F11)	Restores an old version of the current file (see: Restore files).
Check GSE tags	Runs GSE and checks that the GSE tags are correct.
Insert code template (F2)	Types text after entering a few letters (see: Code templates).
Code template editor (Shift-F2)	In this window you can add, change and delete code templates.
GDX viewer (F4)	Starts the GDX DataViewer.
External editor (Shift-F4)	Starts the external editor and opens the currently selected file (see: Gtree INI file).
Start macro key recording (Ctrl-F10)	You can record a series of keystrokes that will perform certain edit tasks and save these keys to file.
Stop macro key recording (Ctrl-Break)	Stops recording and saves the result to a file.
Execute macro (Ctrl-F11)	Selects and runs a key macro file.
Repeat macro (Ctrl-F12)	Repeats the last macro executed.

7.1.3. Options menu

The Options menu (Figure 7.5) offers the following choices (Table 7.3).

Figure 7.5 Options menu

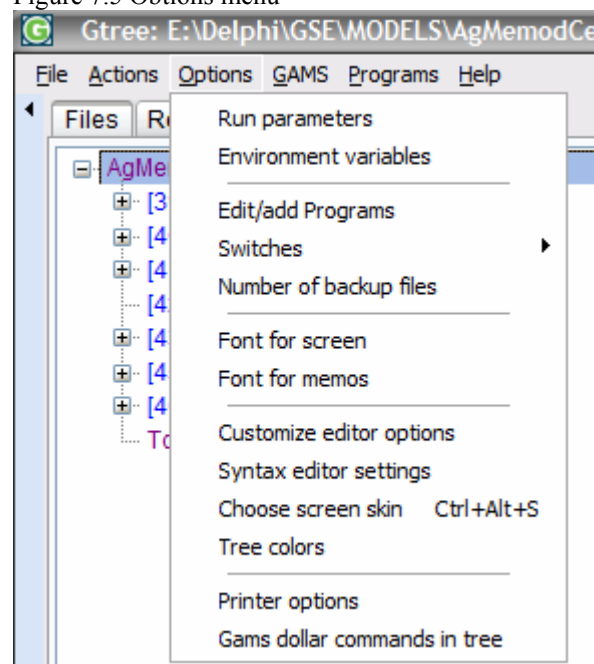


Table 7.3 Choices in the options menu

Run parameters	When you run GAMS or anything else you sometimes need to add parameters, e.g. for GAMS we could add s=save.dat, in which case running GAMS would imply running: gams.exe AGMEMOD.gms s=save.dat Note that after setting run parameters the status bar will show these parameters. In our example in the status bar we would see GAMS: s=save.dat
Environment variables	Shows a window with all the environment variables. Adds, edits and deletes environment variables.
Edit/add programs	In the Gtree menu you can add additional programs you would like to run/call within Gtree (see Additional programs).
Switches	All kinds of Gtree options (see Switches).
Number of backup files	You can specify the maximum number of backups Gtree saves for a single file (see also Restore files).
Font for screen	Selects which font you want to use for the screens.
Font for memos	Selects which font you want to use for the editors.
Colour scheme	By default you can switch between four colour schemes: defaults, classic, ocean and twilights.
Syntax editor settings	Sets the syntax colouring options of the editor.
Customize editor options	Sets the general editor options.
Tree colours	Specifies the tree colours.
Choose screen skin	You will spend a lot of time working in Gtree building your GAMS models. To make life less dull we have made it possible for you to change the look of Gtree to suit your own taste.
Printer options	Sets the printer options used for Gtree.
Gams dollar commands in tree	Creates a list of GAMS dollar commands that will be displayed in the tree.

7.1.4. Switches for Gtree

The switches for Gtree in Options menu (Figure 7.6) are described (Table 7.4).

Figure 7.6 Switches for Gtree

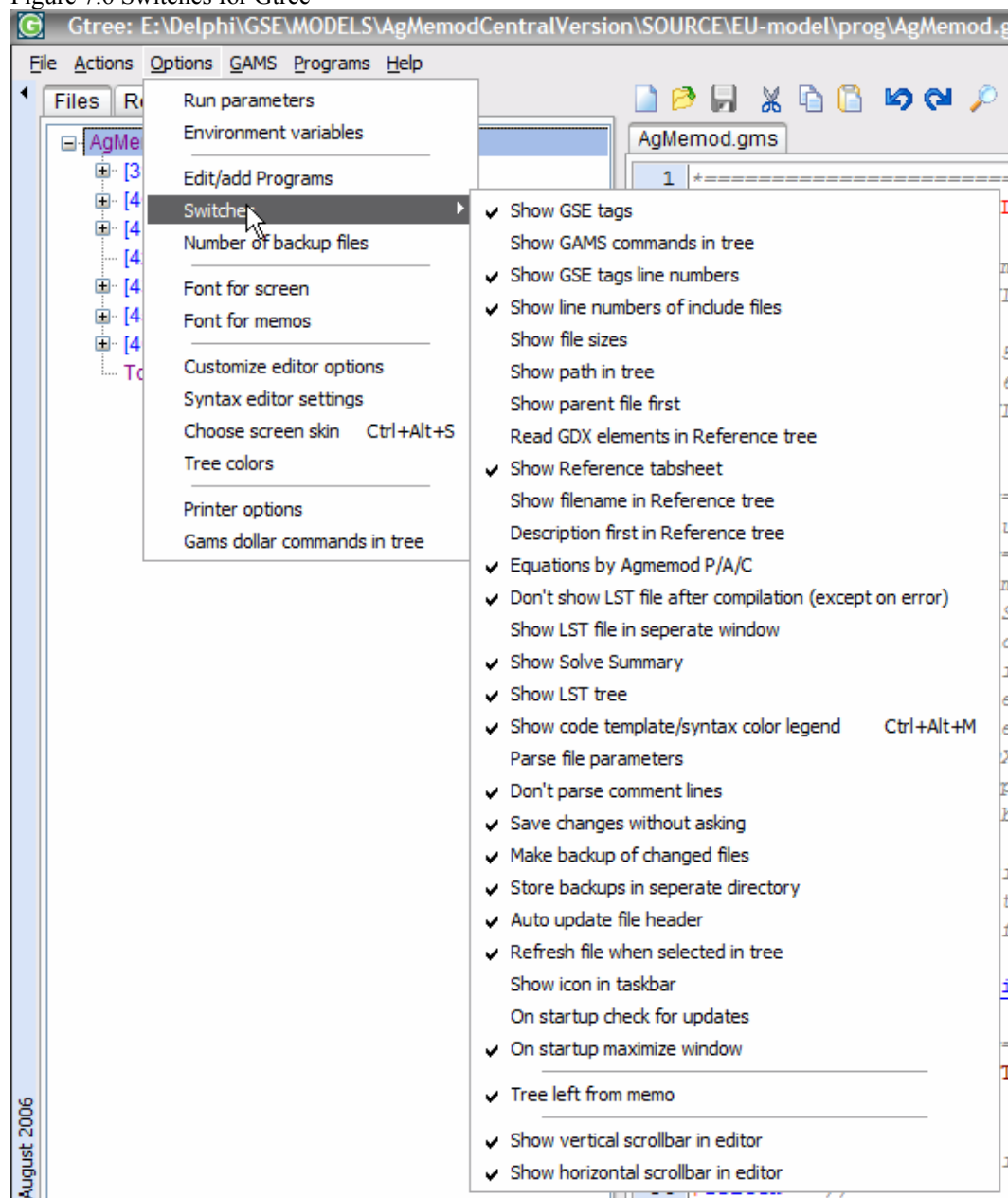


Table 7.4 Description of the switches for Gtree

Show GSE tags	Gtree works for all GAMS files. If you have added GSE tags, you can also show the tags in the tree (see Gtree and GSE). This switch will show/hide GSE tags.
Show GAMS commands in tree	All the GAMS commands on the list are shown in the tree (see GAMS dollar commands).
Show GSE tags line numbers	By default the start and end line numbers are shown in the tree for GSE tags. This switch will show/hide the numbers.
Show line numbers of include files	By default the line number of a file in its parent file is shown (\$include). This switch will toggle and show/hide the line numbers.
Show file sizes	Sometimes it is interesting to know how large a file is. If a file is very big it will take some time to load it into the editor. This

	option will show you the size of a file in the tree (between brackets).
Show path in tree	Shows the complete filename and pathname in the tree.
Show parent file first	By default when you click on a name in the tree, the file is shown in the editor. If the file has a parent file this file is shown on the first tab sheet of the editor. If you select to show the parent file first, both files are available on the tab sheets of the editor, but the parent is shown first. By clicking on the tabs you can switch between the parent and the child. Note that if you click on the parent the line of the \$include statement is marked and displayed on the first line of the editor window.
Read GDX elements in reference tree	By default Gtree will also read the elements that belong to a set. Often sets with a large number of elements are stored in GDX files (instead of ASCII). This option will ignore elements from a GDX file.
Show reference tab sheet	This switch will show or hide the reference tree.
Show filename in reference tree	The filename is shown only the first time reference is made to a file (instead of it being repeated every time, you will see ...). This option will show the complete filename every time.
Description first in reference tree	By default you will see the GAMS name of an item followed by its description. This option will switch to description first and then the GAMS name.
Equations by Agmemod P/A/C	This only works with the Agmemod model and will show the equations in the reference tree sorted by product, activity and country.
Don't show LST file after compilation (except on error)	By default the LST file is shown after each compilation. If you check this option the LST file is shown only when there is an error in the GAMS code/run.
Show LST file in separate window	Check this if you want the LST displayed in a separate window.
Show solve summary	When GAMS has finished running the model, a summary of the GAMS results is shown (telling you if GAMS found an optimal solution).
Show LST tree	LST files can be very long and finding something in them can be time-consuming. A tree of important items can help you navigate easier through the LST file (see LST tree).
Show code template/syntax colour legend Ctrl-Alt-M	By default the colours used for syntax colouring are shown in the file tree. This option makes it possible to hide the colour syntax legend.
Parse file parameters	By default the values of parameters behind a file statement (e.g. \$include or in a batch file) are not parsed. You can switch this option on and parse values of parameters.
Don't parse comment lines	By default \$include statements in comments are ignored when parsing the tree. If you want to have them parsed as well, use this toggle. Some model-builders find it highly instructive to use this option and see what could happen if an asterisk is added or deleted by mistake.
Save changes without asking	Once you have changed the contents of a file in the editor, you are asked if you want to save the changes (e.g. when you select another file in the tree). If you use this switch the changed files are saved automatically without asking you each time if you want to save them.
Make backup of changed files	Every time a file is changed and saved a backup of the file is created.
Store backups in separate directory	Backups are stored in a special subdirectory (and hence make it visible).
Refresh file when selected in tree	When this option is checked and you click on a tree file, the file is rescanned if the file date has been changed since the last scan.
Auto update file header	If you have a file header that was created by the Gtree template/macro head you can automatically change the date the file was last edited and the user who changed the file.

Show icon in taskbar	You can show Gtree as an icon on the taskbar.
Save tree contents	When you quit Gtree a file is created with the complete tree structure in it (with the gtree file extension). The next time you load the file this Gtree file is used and there is therefore no need to scan the files (for large projects this is much faster). This works fine with <i>Refresh file when selected in tree</i> option checked.
On startup check for updates	When Gtree starts, it will check the internet (www.nacquit.com) for updates. If there are updates, they are downloaded and installed automatically.
On startup maximise window	Gtree shows a host of information. It is therefore useful to start Gtree using the maximum screen width and height.
Tree left from memo	By default the tree is shown on the left-hand side of the window and the editor on the right. Using this switch you put the tree on top of the screen and the editor at the bottom.
Show horizontal scrollbar in editor	Uses a horizontal scrollbar in the editor, so you can scroll to the left and right in the editor.
Show vertical scrollbar in editor	Uses a vertical scrollbar in the editor, so you can scroll up and down in the editor.

7.1.5. GAMS menu

The GAMS menu (Figure 7.8) offers the following choices (Figure 7.7).

Figure 7.7 GAMS menu

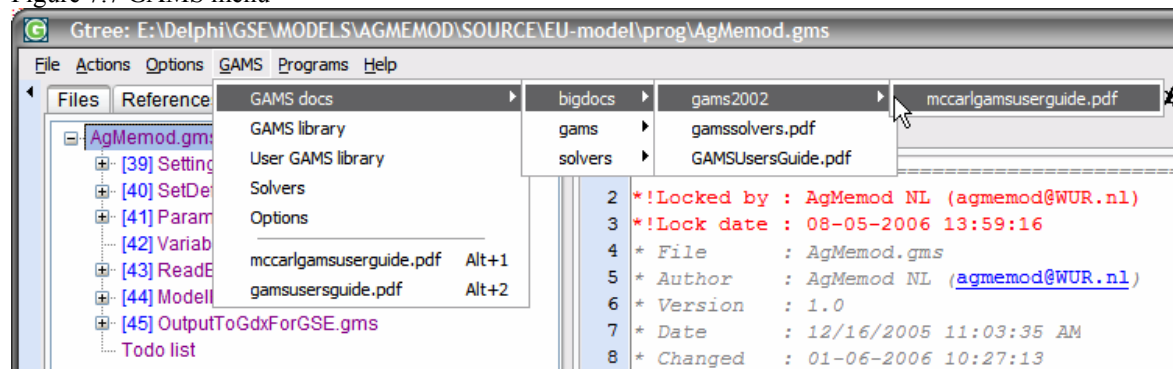


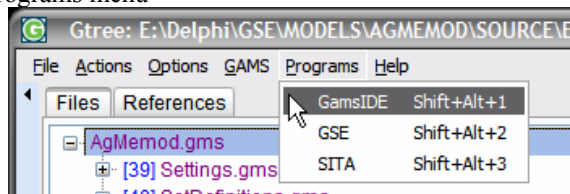
Table 7.5 Choices in the GAMS menu

GAMS docs	Here you will find a complete list of all the available GAMS documentation. Select which document you want to view. Note that the GAMSUserGuide.pdf can be found under bigdocs.
GAMS library	Opens the GAMS library.
User GAMS library	Asks for a user library and shows the Gtree library window.
Solvers	Shows a grid of available Solvers and the default settings for GAMS.
Options	In this window certain GAMS options can be set (see: GAMS options).

In the Gtree.ini file (in the section [GamsDoc]) you can specify a list of documents that are available under a keyboard shortcut (e.g. the Bruce McCarl manual can be started with the Alt-1 key combination).

In the Programs menu you can add programs you would like to start from Gtree (see Additional programs).

Figure 7.8 Selection of the programs menu



7.1.6. Help menu

The **Help menu** (Figure 7.9) offers the following choices (Table 7.6)

Figure 7.9 Help menu

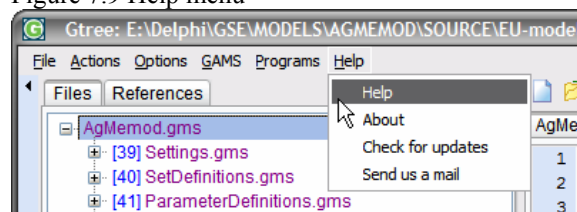


Table 7.6 Choices in the help menu

Help	Shows the help file.
About	Shows information about the program (version number etc.).
Check for updates	Checks the web for updates of Gtree.
Send us a mail	Sends an e-mail with your comments, remarks and/or questions.

7.2. Functions of Gtree

7.2.1. Editor buttons

The editor uses buttons to give you easy access to edit functions. Most of these buttons also have keyboard shortcuts (Figure 7.10; Table 7.7).

Figure 7.10 Editor buttons

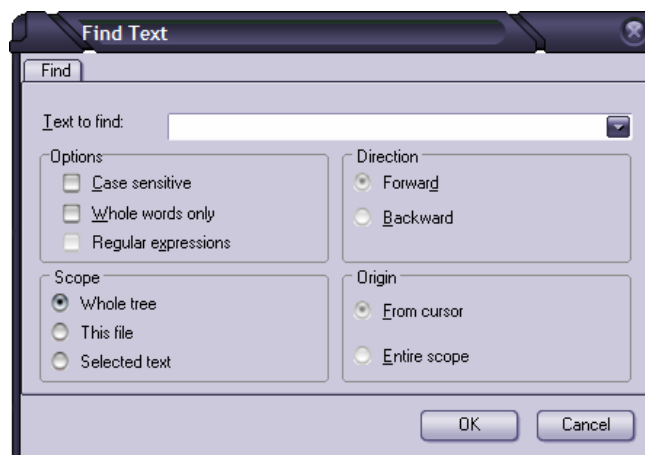


Table 7.7 Descriptions of the editor buttons

1 New (Ctrl-N)	Creates a new file and opens it in the editor.
2 Open (Ctrl-O)	Opens an existing file in the editor.
3 Save (Ctrl-S)	Saves the changes to the current file to disk.
4 Cut (Ctrl-X)	Copies and deletes the selected block to the clipboard.
5 Copy (Ctrl-C)	Copies the selected block to the clipboard.
6 Paste (Ctrl-V)	Pastes the clipboard to the editor.
7 Undo (Ctrl-Z)	Undoes the last change in the editor.
8 Redo (Ctrl-U)	Undoes the previous undo.

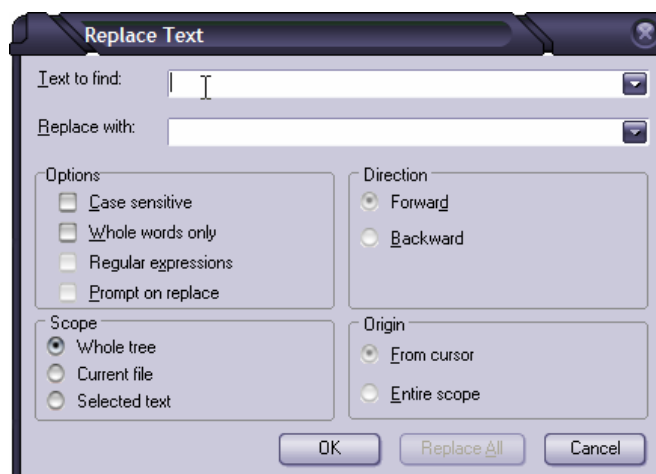
9 Find (Ctrl-F)

Opens a find window that makes it possible to find a certain string in the current file.



10 Replace (Ctrl-R)

Opens a window that makes it possible to find and replace strings in the current file.



11 Font for memo

Selects which font you want to use for the editors.

12 Print (Ctrl-P)

Opens the print preview window.

13 To do list

Opens/creates a "to do" list.

14 Open MS-Dos box

Starts MS-Dos.

15 Line numbers

By default the editor is shown with line numbers. This button will toggle and show or hide the line numbers.

16 Run (F9)

Runs the files that are at the root of the tree (in our examples file AGMEMOD.gms). Gtree will have a list of which programs are used for certain file extensions (if no extension/program combination is specified, the windows defaults are taken, see Gtree INI file).

17 Find matching character

Users entering large complex formulas easily lose track of where brackets start and end. This button marks the beginning and end of matching character sets, to make it possible to see where mistakes were made in the formula (by putting the closing or end character too soon or too late or forgetting the matching character). This matching holds for the characters () [] " ' // and \\. Note that you can start with either the begin or the end character and that Gtree will find the associated end or begin character. Possible keyboard shortcuts are Ctrl-9, Ctrl-0, Ctrl-[Ctrl-], Ctrl-' ,Ctrl-/ and Ctrl-\.

18 Go to line (Ctrl-L)

You can specify in an input box which line number in the current file you want to jump to.

19 Find GAMS error number

If you are interested in what a certain error number (e.g. in a LST

- | | |
|-----------------------------|--|
| 20 Show GAMS list file (F7) | file) means in GAMS, go to the beginning of the error number and click this button. Below the tree you will see the error message associated with that error number. You do not have to scroll to the end of a file to understand what this error number means. |
| 21 Show GDX file (F8) | Shows a list of available GAMS LST files. If there is only one LST file, this file is shown immediately. |
| 22 Close LST file | Shows a list of GAMS GDX files. If there is only one GDX file, this file is shown immediately. |
| 23 Find GAMS error | If you have opened the LST file in a separate window, you can press this button to close the LST file. |
| 24 Go back | If running GAMS returns an error the LST file is loaded and Gtree jumps to the first line in the LST file with an error. Clicking on this button will open the file that contains the error. See: Errors. |
| | When looking at the contents of a file you can find information on a set, parameter or variable by pressing Alt-R (shows the reference information on the word where the cursor is located). After looking at the reference information you can click on this button to go back to the file where you pressed Alt-R. |

On the right-hand side of the buttons you will see a combo box, which shows which syntax colouring files are available and which syntax colouring is being used at the moment (see also the Gtree INI file and Syntax editor settings).

When you right-click with your mouse on the tree you will get the pop-up menu set (Figure 7.11).

Figure 7.11 Pop-up menu

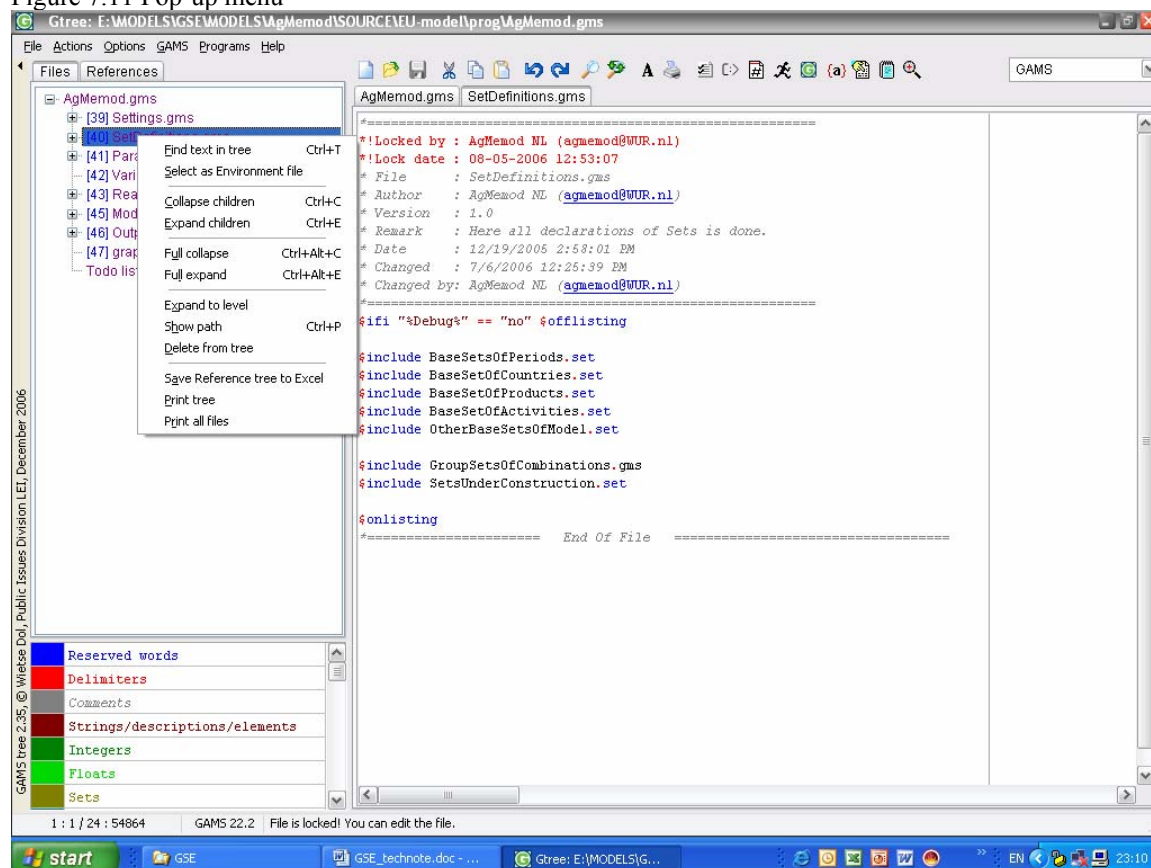
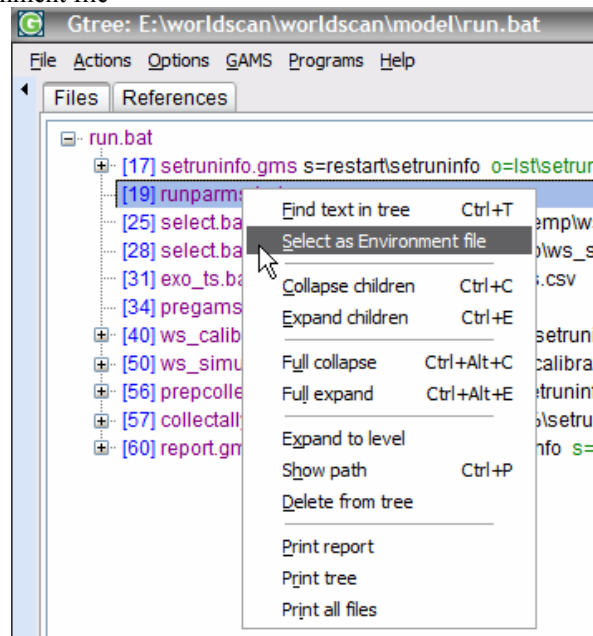


Figure 7.12 Select as environment file



Use this pop-up menu to collapse or expand the tree. Most of the options are clear. Three of them need to be mentioned. The first “Select as Environment file” (Figure 7.12) will take the content of the selected (batch) file and use all the environment declarations (like: set model=agmemod) to define the environment declarations within Gtree (see Environment variables). The second “Print tree” will create a printable version of the complete tree (Figure 7.13):

Figure 7.13 Print tree

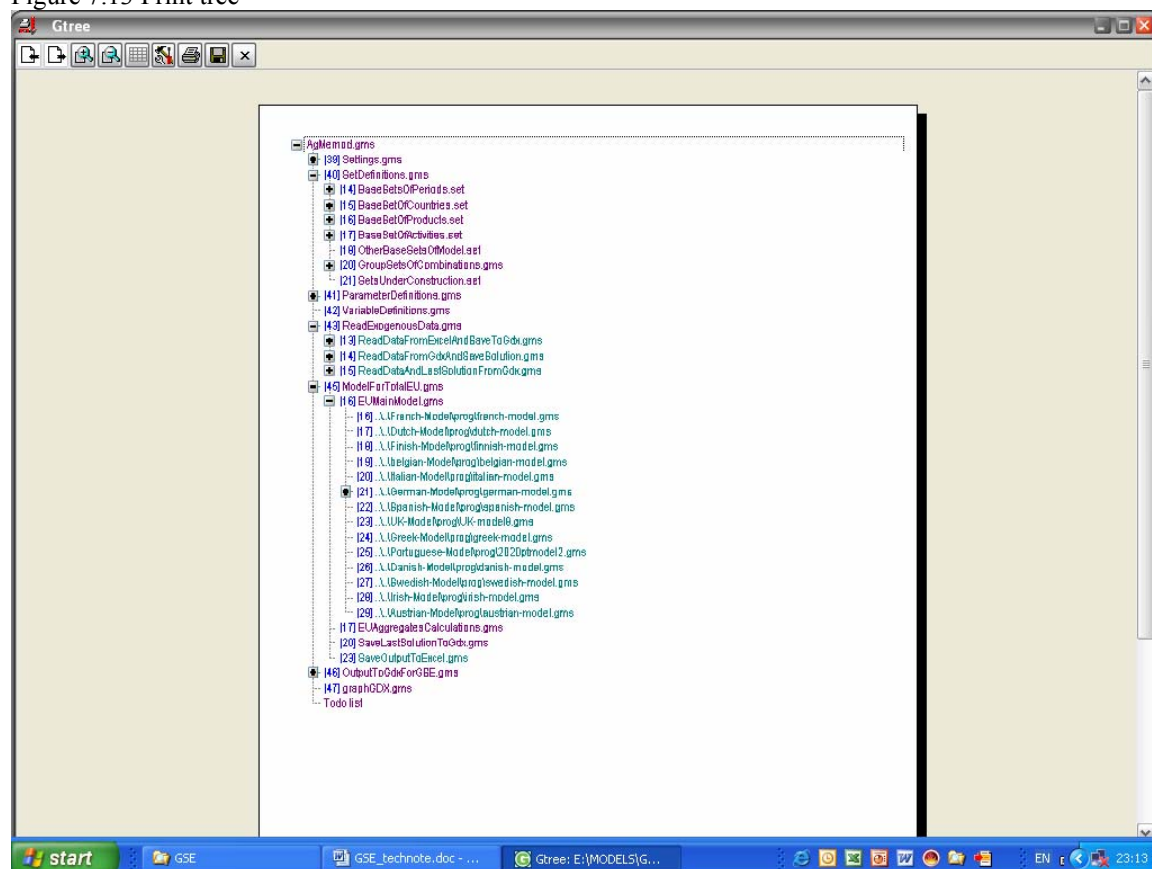
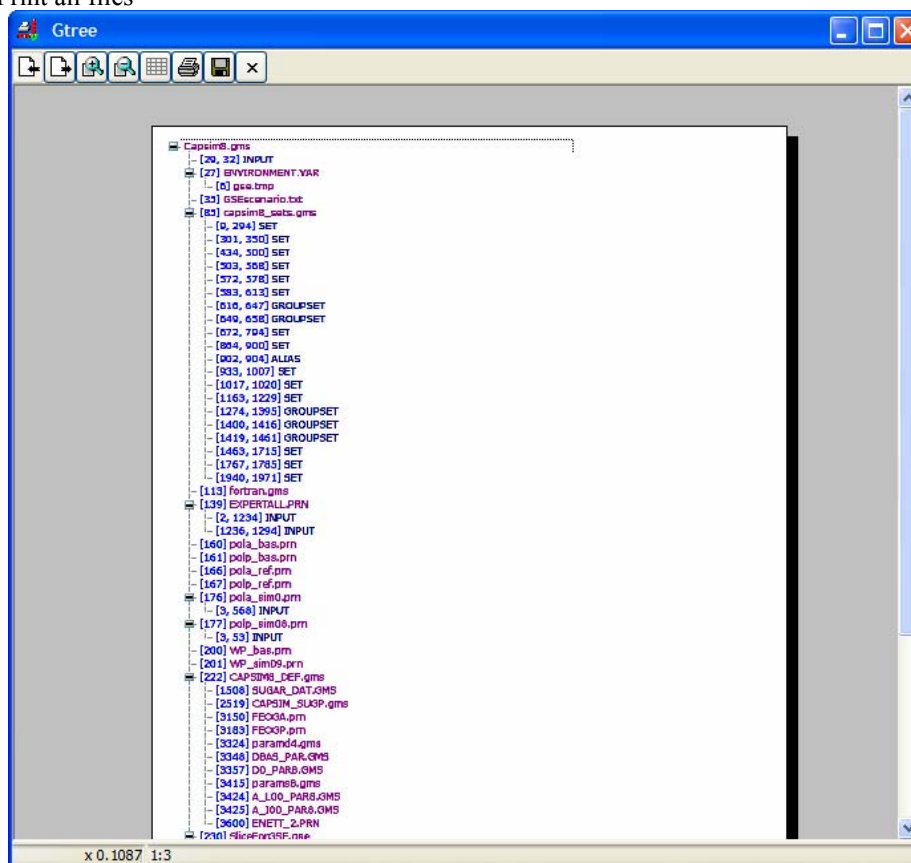


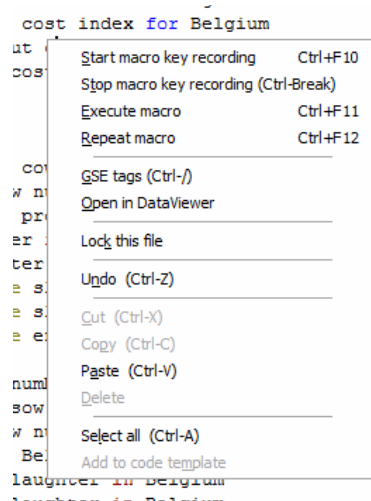
Figure 7.14 Print all files



The “Print all files” option will print all files in the tree (using the printer settings shown in the print preview) (Figure 7.14).

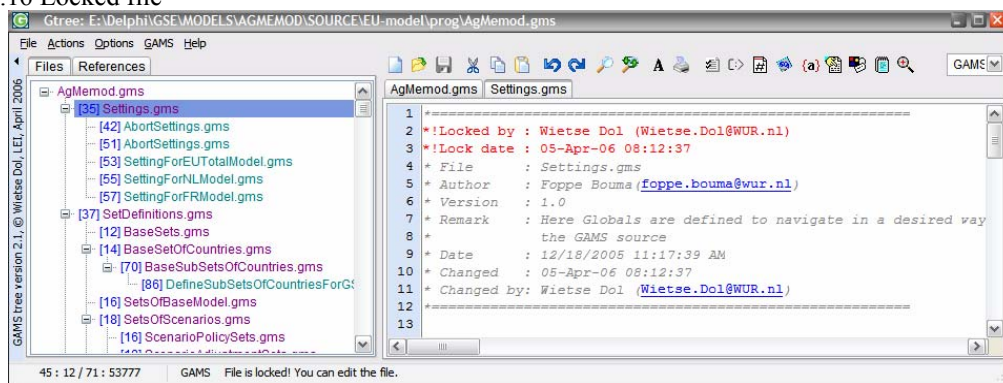
If you are in the editor and right-click your mouse you will see the following pop-up menu (most of the options should be clear after reading this document):

Figure 7.15 Pop-up menu in the editor



The Lock this file option was introduced to allow a group of people to work on a model at the same time. Once a file is locked only the person who locked the file is allowed to edit it; others can only read it. Once the person has finished working on the file he or she will right-click again and choose the menu option Unlock this file. It is clearly visible that the file is locked (Figure 7.16).

Figure 7.16 Locked file



The file lock system also shows another useful feature of Gtree. By default GAMS comments are shown in grey. But if the first character after the star (*) is an exclamation mark (!) the line is shown in red. This draws attention to certain (important) comments (Figure 7.17). To extend the possibilities, you could also the option that if the second character (after the *) is a question mark (?) the line will turn lime green, or that a minus sign (-) makes the comment line blue and an underline () turns the line green (Figure 7.18).

Figure 7.17 GAMS comments

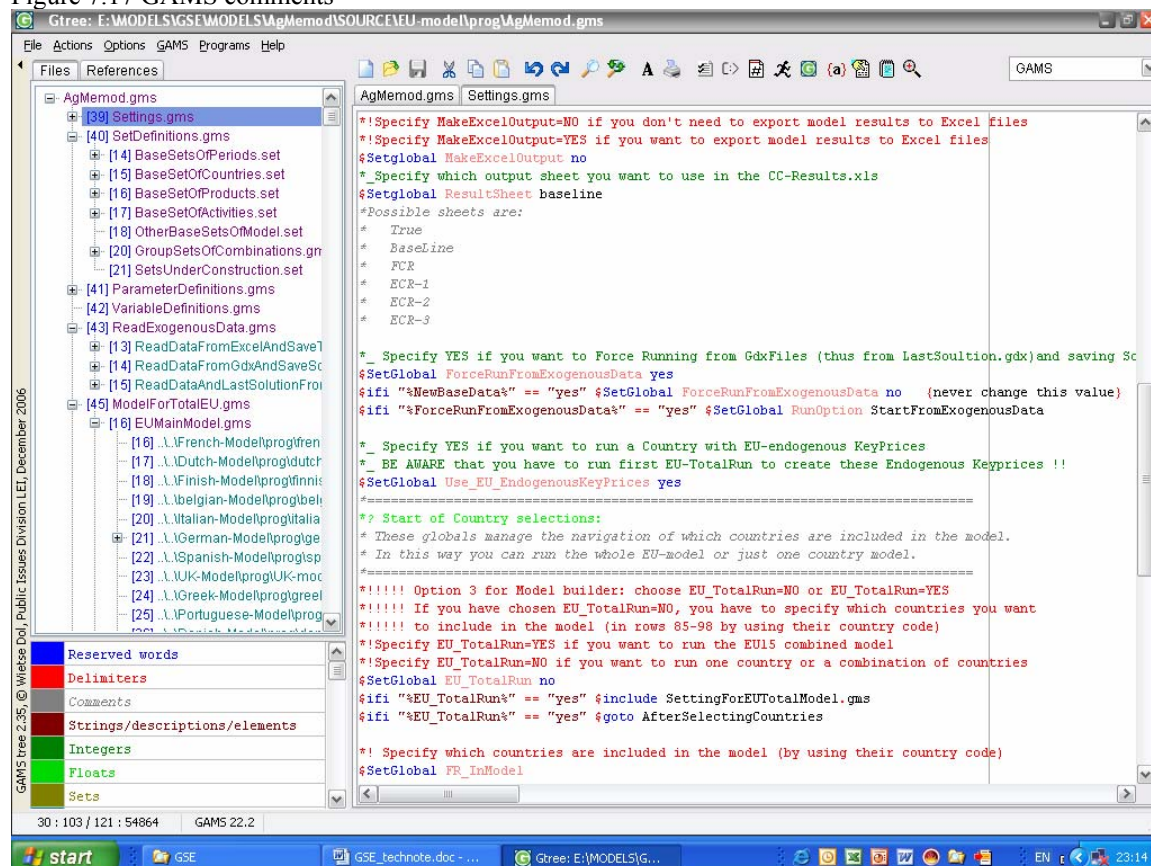
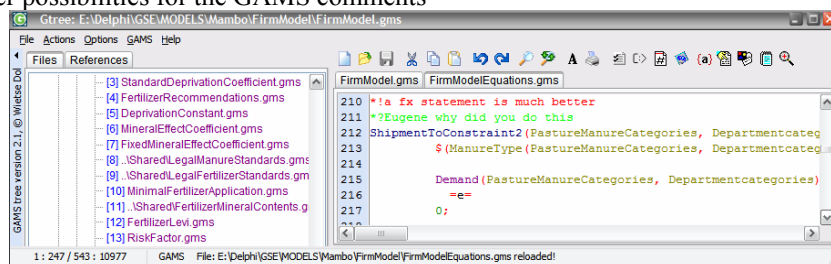


Figure 7.18 Other possibilities for the GAMS comments



7.2.2. Reference file

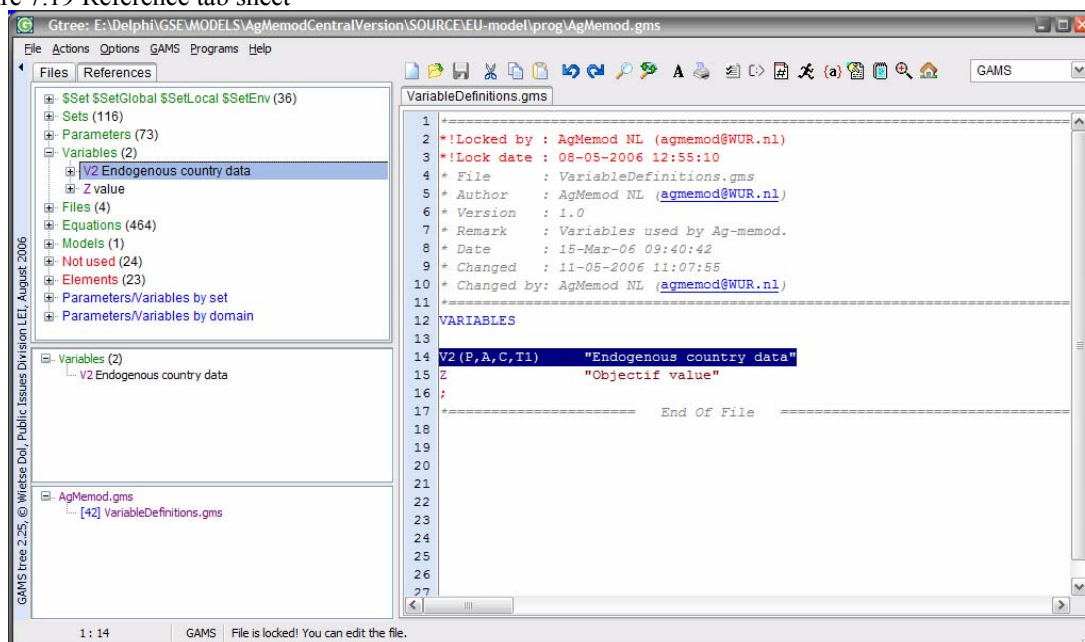
If you build your model with a large number of \$include statements it will be much more readable and it will be easier for you to discover where you have to change your code. GAMS can help you even more with your model. You can run GAMS with your model and create a “reference file”. This file shows where all sets, parameters, variables, files, equations and models are declared, defined, assigned, controlled and referenced. The declarations of all GAMS \$set, \$setglobal, \$setlocal and \$setenv statements can also be found from the GAMS manual:

Table 7.8 Description of a reference file sets

Reference	Description
DECLARED	This is where the identifier is declared. This must be the first appearance of the identifier.
DEFINED	This is the line number where initialisation (a table or a data list between slashes) or symbolic definition (equation) starts for the symbol.
ASSIGNED	This is when values are replaced because the identifier appears on the left of the assignment statement.
IMPL-ASSIGNMENT	This is an “implicit assignment”: an equation or variable will be updated as a result of being referred to implicitly in a solve statement.
CONTROL	This refers to use of a set as the driving index in an assignment, equation, loop or other indexed operation (sum, prod, smin or smax).
REFERENCE	This is a reference: the symbol has been referenced on the right of an assignment, in a display, in an equation or in a model or solve statement.

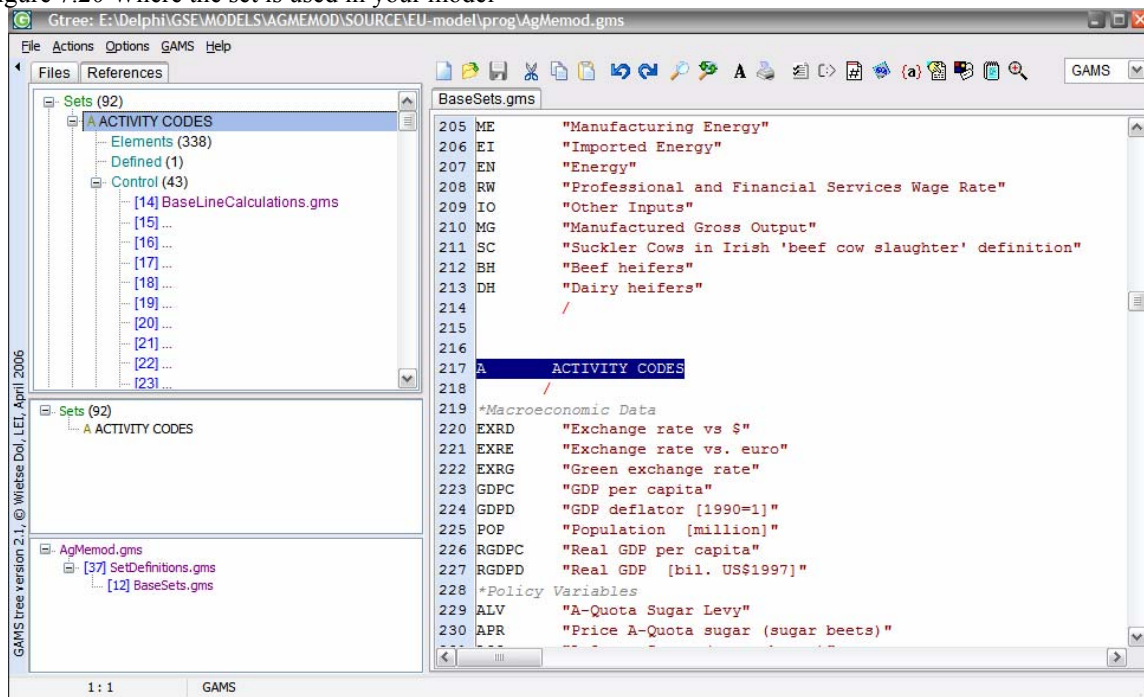
Gtree will do all the work for you: run GAMS, read the reference file and show the outcome in a tree in the *References tab sheet* (Figure 7.19).

Figure 7.19 Reference tab sheet



Just browse through the tree and you will get a very detailed idea of how the model is defined. Sets contains a list of all the sets declared in your model. Click on any Set and you will immediately jump to the file and line where the set is declared. If you want to know where the set is used in your model just start clicking on the Control item, etc., etc. (Figure 7.21).

Figure 7.20 Where the set is used in your model



Gtree will help you even more. After giving you detailed information on sets, parameters, variables, equations and models, two more items can be found in the reference tree:

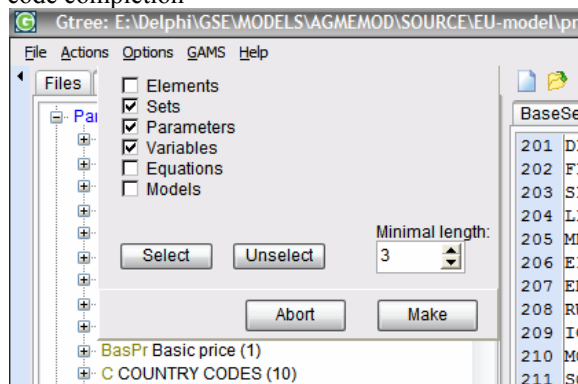
Not used: here you will find a list of everything declared in GAMS but not used in the model (N.B. all displayed in red).

Elements: GAMS requires sets, parameters, variables, equations and models to have unique names. This is not true of elements. Under this item you will find all kinds of useful information on elements (e.g. which element names are also set or parameter names, etc.). The last two items in the tree will give you a good idea which parameters and variables use certain sets or combinations of sets. For example, if you add a new element to a

certain set it is easy to check which parameters and variables use this set and, hence, know which inputs would need additional data (for the new element).

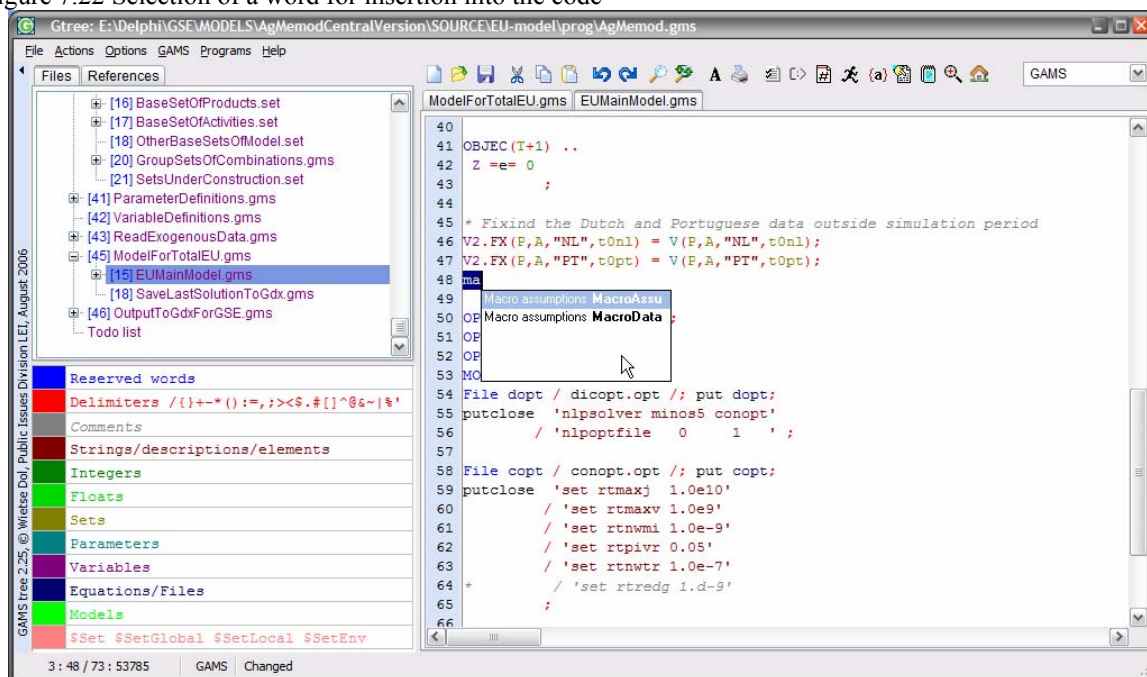
Once you have created a reference tree you can use the results in your model (press F12).

Figure 7.21 Select names for code completion



Use this window to select which names are stored for code completion. Typing your model is time-consuming and typing errors are easily made. Modern programming languages all have the code completion feature: just type a few letters, press Ctrl-J and you will get a list of all words that contain the letters. Select the word that you want from the list and it is inserted into the code (Figure 7.22).

Figure 7.22 Selection of a word for insertion into the code



```

45 * Fixind the Dutch and Portugu
46 V2.FX(P,A,"NL",t0n1) = V(P,A,"
47 V2.FX(P,A,"PT",t0pt) = V(P,A,"
48 MacroAssu(MacroData,C,TE)
49

```

Note that all the set information is also added to the code when running code completion. Every time you press the *Reference* tab sheet, the file for code completion is generated and the selected items are saved. Additional GAMS statements (visible in blue in the editor) are stored in the code completion table.

When creating a reference file, Gtree will also store all names of sets, parameters, etc. so that they are used for syntax colouring (e.g. in the illustration above it is immediately clear that A_Exrate is a set and MacroAssu is a

parameter). This colouring will make your model much more readable and you will see immediately when you make a typing error.

Once you have created the reference tree, when you select a *Set* (which will have one or more subsets) and right-click with your mouse, a pop-up menu will offer you the option: *Show elements/(sub)set grid*. Clicking on this option will open a new window (Figure 7.23).

Figure 7.23 Show elements/(sub)set grid

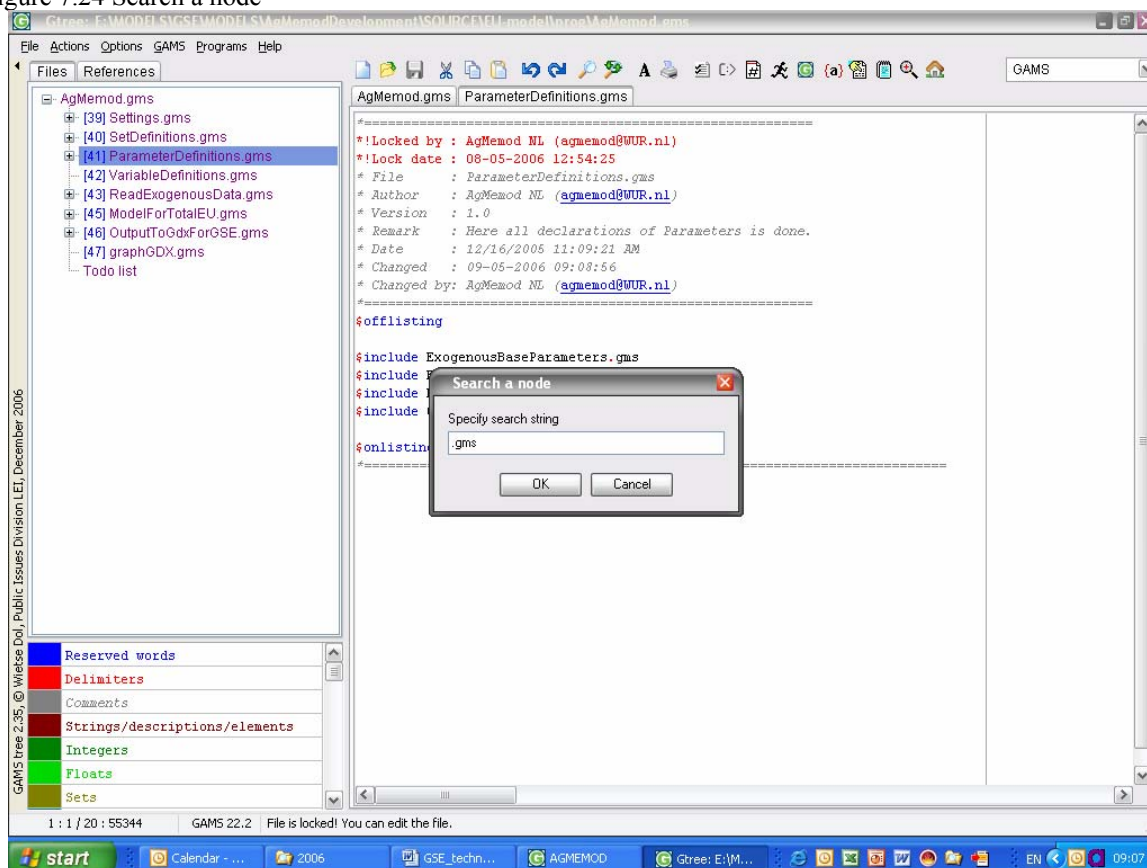
Name	A	A_ExchangeRate	ActCattle	ActExtraCrop	ActMilk	ActProAni	AnPrem	BasPr	COQuota	CostEnv	ExpSub	H
FCO	1											
DPE	1											
CPE	1											
TOP	1											
NIN	1											
FAC	1											
CPR	1											
EPR	1											
CMA	1											
EXRD	1	1										
EXRE	1											
EXRG	1											
GDPC	1											
GDPD	1											
POP	1											
RGDPC	1											
RGDPD	1											
ALY	1											
APR	1											
AQO	1								1			
BLV	1											
BPR	1											
BQO	1								1			
CAI	1											
COM	1											1
MBP	1											
PAI	1											
PCS	1											
PFS	1											
PIN	1											

Here you will see the tree structure of the Set and its subsets, but also a grid giving a clear indication of which elements are used in which (sub)set.

7.2.3. Find text in tree

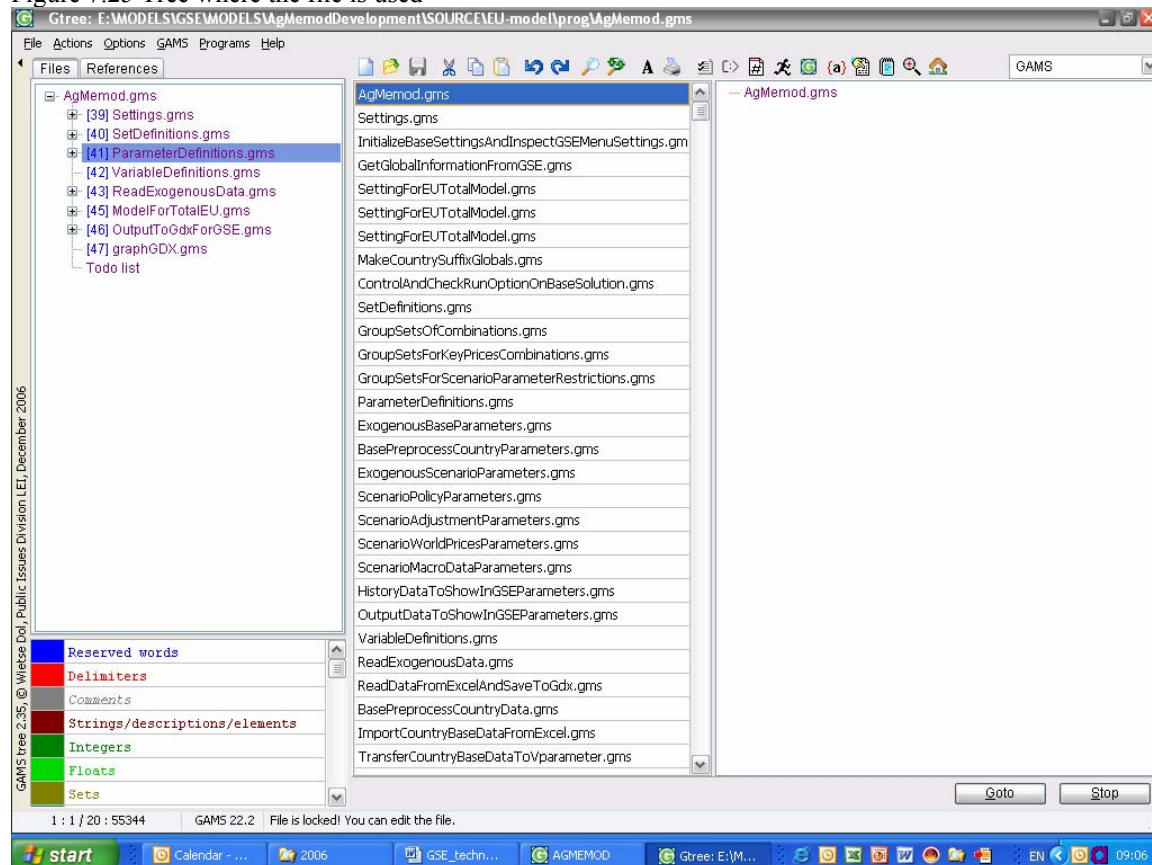
When you are building large models you will probably use a large number of include files. When you are looking for a certain file you can use *Find text in tree* (Ctrl-T). When you click on this item on the menu (or press Ctrl-T) a window will ask you to specify a search string. In this example we are looking for files with the .GMS file extension (Figure 7.24).

Figure 7.24 Search a node



The result of the search is shown below and when you click on a certain file (middle pane) the right-hand pane will show the tree where this file is used (Figure 7.25). If you want to edit this file press *Goto*. If not, select another file or press *Stop*.

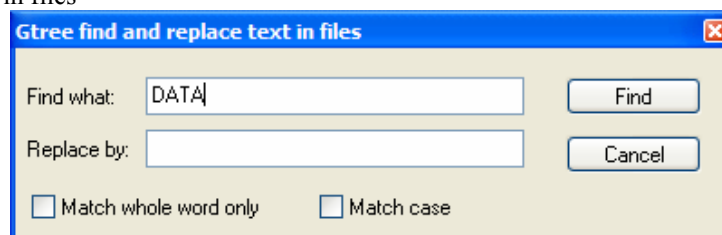
Figure 7.25 Tree where the file is used



7.2.4. Search/replace text in files

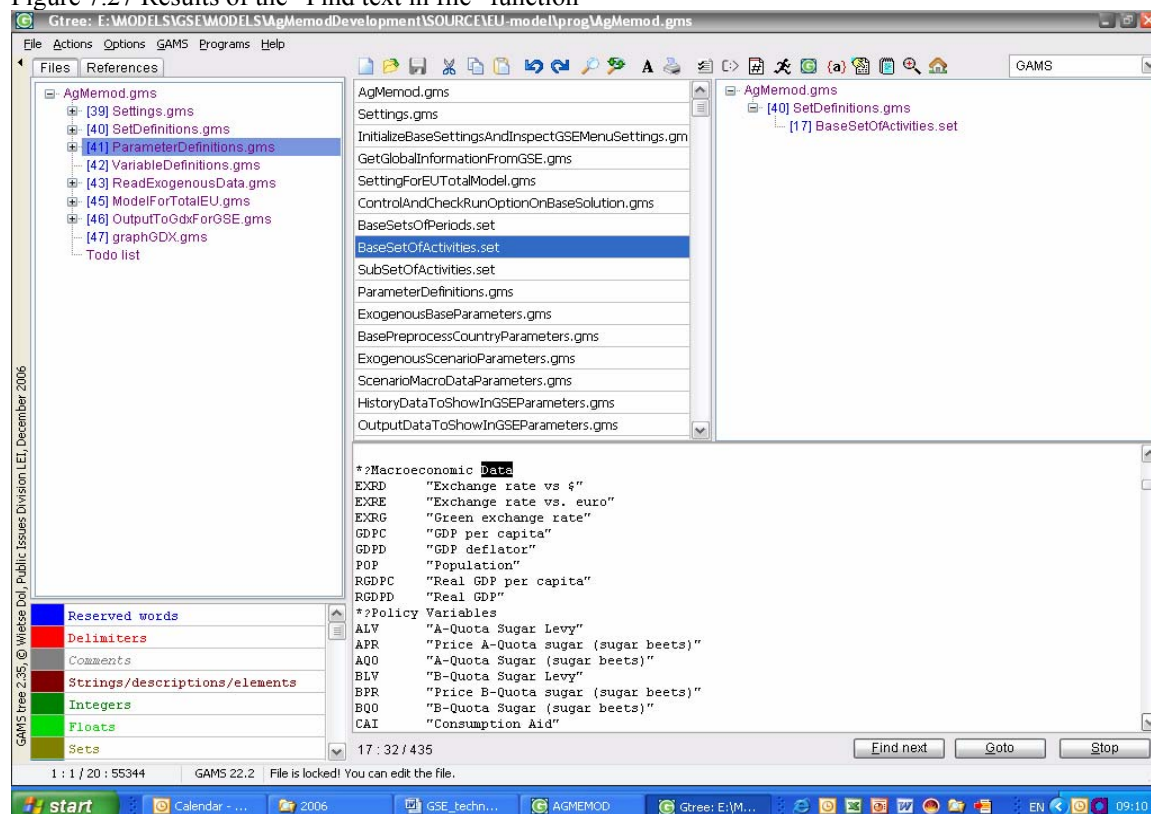
The larger model become, the more files you use and the more difficult it becomes to find specific text. If you are looking for certain text you can use *Find text in files* (F3). When you click on this item (or press the F3 key) a window will ask you to specify a search (or replace) string. In this example we are looking for the text string DATA (Figure 7.26).

Figure 7.26 Find text in files



Below you will find the results (Figure 7.27). In the middle pane you will find a list of the files that contain the search string. On the right you see the tree and know where the file is used. In the bottom right-hand pane you will see the selected file with the search text marked. When you press *Goto* the selected file is opened in the editor. *Find next* will search for the next appearance of the search string. If the search reaches the end of the selected file, the next file from the list is taken and the search is continued. *Stop* will take you back to the Gtree main window. *Replace* will replace the selected text by the replace string. If you want to change the search string in all the files press *Replace all*. Note that since the search is done using the tree structure the first file that is found, e.g. when you are looking for a parameter of a set, will be the file where the definition is given.

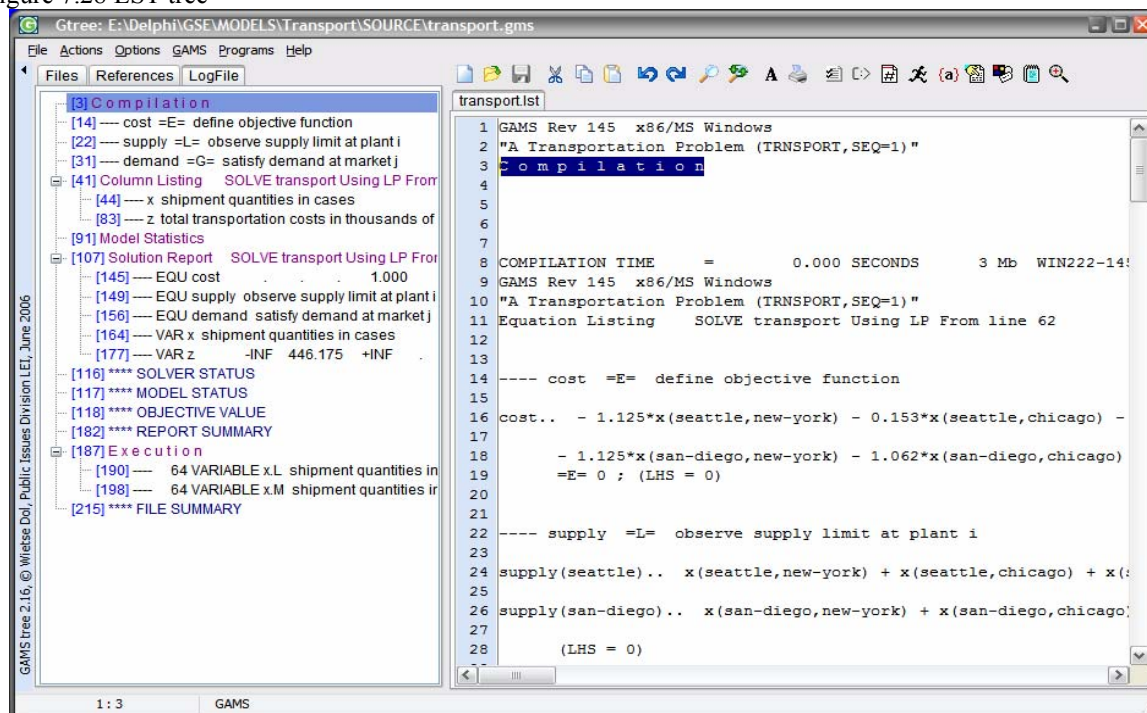
Figure 7.27 Results of the “Find text in file” function



7.2.5. LST tree

The LST tree will help you navigate through or make it easier to find something in the LST file (Figure 7.28). For very long files the tree will take some time to generate, but makes it easier to navigate.

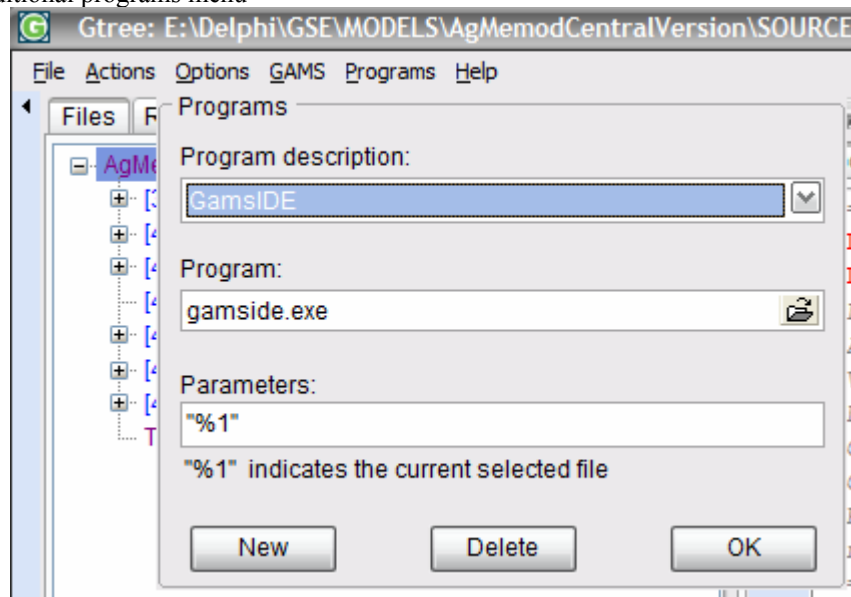
Figure 7.28 LST tree



7.2.6. Additional programs

The menu in Figure 7.29 enables you easily to add, edit and delete programs that can be started from within Gtree (and are visible under the *Programs* menu option in Gtree). Press *New* if you want to create a new program reference in the Gtree menu. Just enter a description and then specify the program you want to start. It is possible to add startup parameters to the program. Just remember that when you use **%1** as a startup parameter, this means that Gtree will add the name of the currently selected file in Gtree as a startup parameter to the program. *Delete* will delete the currently selected program from the menu list. Press OK to close this window and rebuild the Gtree menu options.

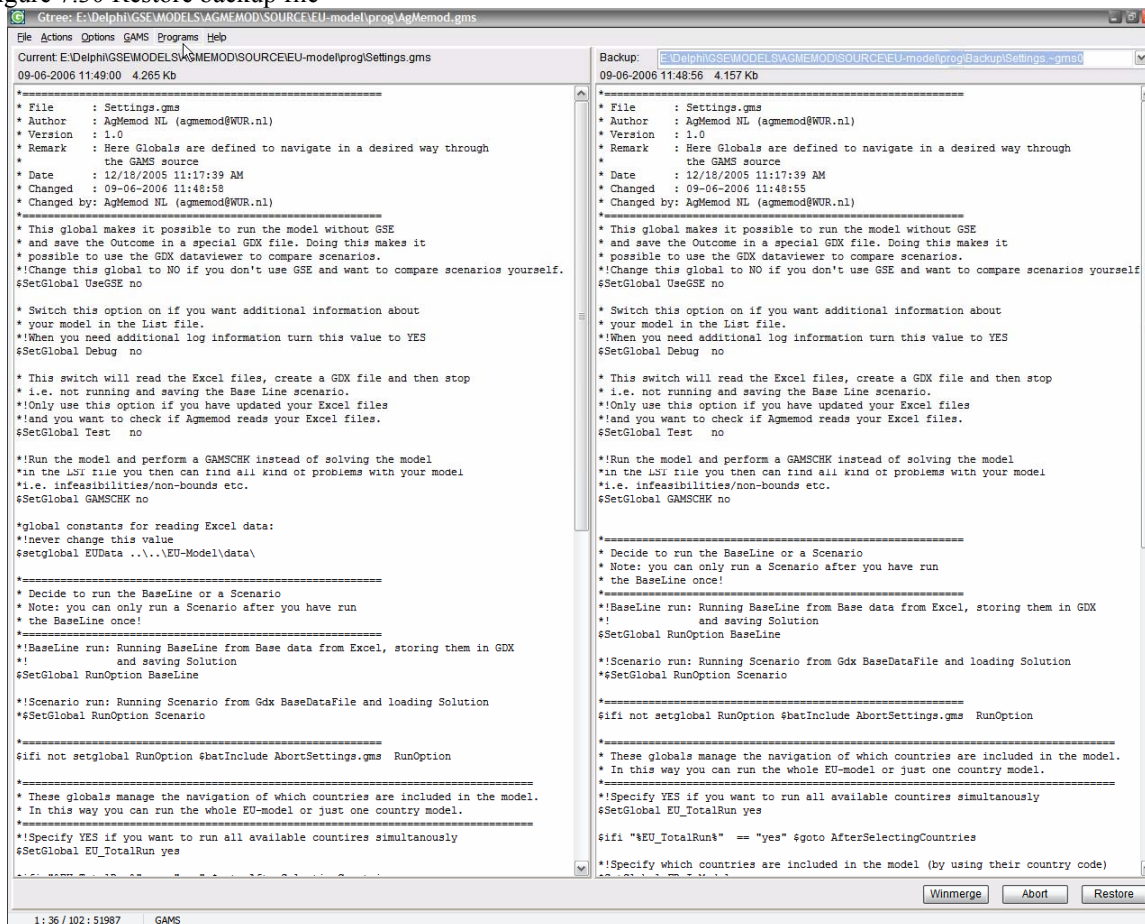
Figure 7.29 Additional programs menu



7.2.7. Restore files

Every time you save a file, a backup file is created (Gtree keeps the last nine files changed). With *Restore backup file* you open a window (Figure 7.30). On the left-hand side the current file is shown and on the right-hand side you can select which backup file you want to see. Press *Winmerge* if you want to start Winmerge and carry out a very detailed file comparison/restore. Press *Abort* to close this window and press *Restore* if you want to restore the currently selected backup to the current file.

Figure 7.30 Restore backup file



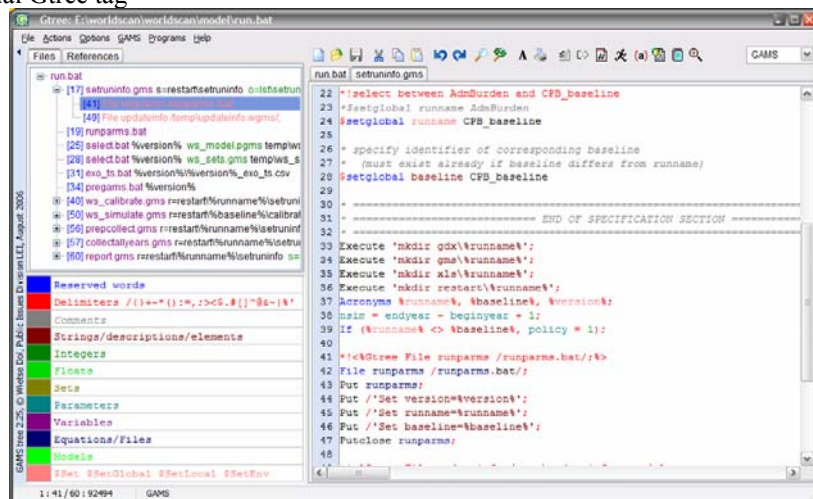
7.2.8. Special Gtree tags

Gtree scans for all \$include, \$batinclude, etc. statements and builds a tree that you can browse. There are two special comments lines (called Gtree tags) that influence the tree content (Figure 7.31). The first is:

```
*!<-%GTREE comment%>
```

This means that the comment is shown in the tree:

Figure 7.31 Special Gtree tag



Here we see that the statement in line 41 is shown in the tree.

The second Gtree tag is like (Figure 7.32):

```
*!<-%GTREECONTENT $include off%>
```

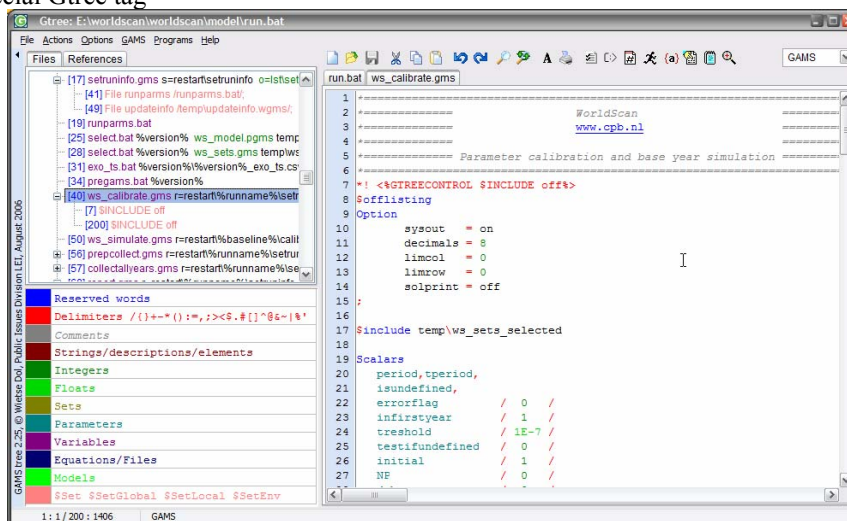
```
*!<-%GTREECONTENT $include on%>
```

```
*!<-%GTREECONTENT $batinclude off%>
```

```
*!<-%GTREECONTENT $batinclude on%>
```

This tag is used to switch on or off the Gtree scanning for include statements.

Figure 7.32 Special Gtree tag



7.2.9. Gtree INI file

Many options which you can select are saved in the GTREE.INI file. The next time you start Gtree all the previous settings are restored. There are three sections in the INI file that are used by Gtree but which you cannot change within the Gtree program. You can change these sections to get a more personalised Gtree. Just to show you the defaults of these three sections:

```
[Settings]
;general settings
CopyRight=NacquIT 2004
HeaderLeft={NAME}
HeaderCenter=
HeaderRight={PAGE}/{NUMPAGES}
FooterLeft={COPYRIGHT}
FooterCenter=
FooterRight={DATE}

[FileExtension]
.GMS=gams.exe
.STI=scentab.exe
.STP=scentab.exe
.BAT=
.CMD=

[Syntax]
1=GAMS.gms
2=Nothing.lst
3=HTML.htm
4=HTML.html
5=objectpascal.pas
6=batch.bat

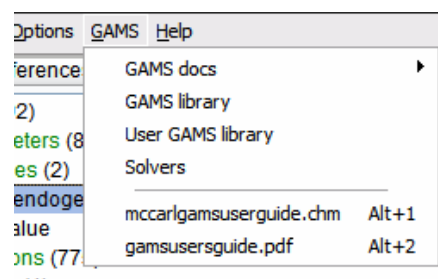
[GAMSdoc]
File1=$GAMSDIR\DOCS\BIGDOCS\GAMS2002\mccarlgamsuserguide.pdf
File2=$GAMSDIR\DOCS\BIGDOCS\gamsusersguide.pdf
```

In the first section (Settings) you will find the header and footer settings for the printer (see print preview). The second section (FileExtension) is an extension of the windows function. The left-hand side of an element indicates a file extension. The right-hand side indicates which program is executed when you *Run* (F9) a file with the given file extension. So .GMS=gams.exe means that files with the .gms extension are run with gams.exe when you press *Run*.

The third section (Syntax) is a list of colour/syntax highlighting used. GAMS.gms simply means that files with the .gms extension will use the syntax colouring defined by the GAMS scheme file (i.e. the file GAMS.SCH found in the SyntaxSchemes directory).

In the last section you can define documents that are shown when clicking on the GAMS menu. In the example here the two important GAMS manuals will be available just by pressing the Alt+1 or Alt+2 keys (Figure 7.33).

Figure 7.33 GAMS manuals

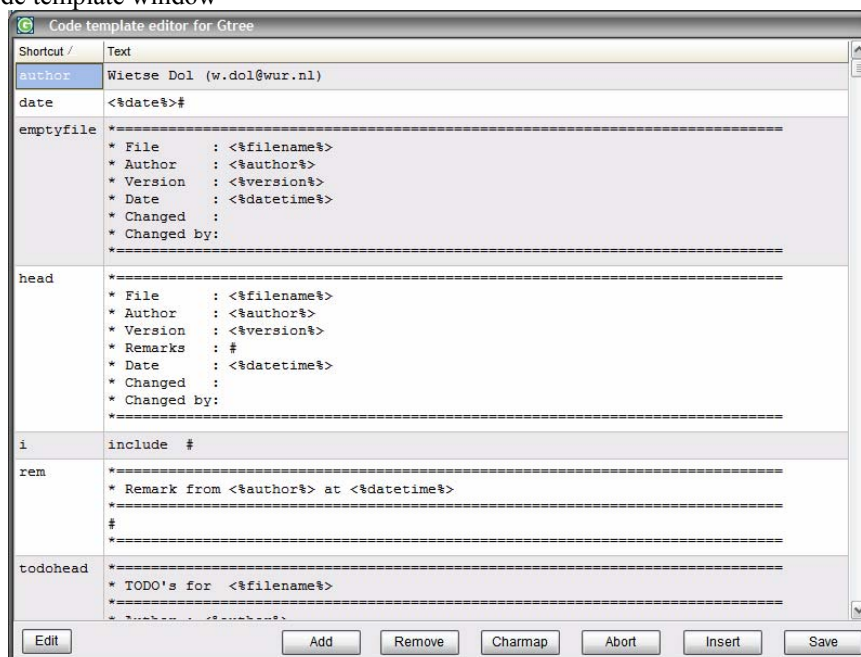


7.2.10. Code templates

Building models and editing text entails a lot of typing. To help you we have implemented certain kinds of code templates. You type a word and then press the F2 key. The word to the left of the cursor is taken and the

program looks at a list to see what text should replace the word. Simple but powerful! Have a look at the *Code template* window (press Shift-F2) (Figure 7.34).

Figure 7.34 Code template window



So by typing `author` in the editor and then pressing F2 “author” is changed into “Wietse Dol (w.dol@wur.nl)”. By typing `$i` and then pressing F2 the letter “i” is changed into “include” (to give you “\$include”). Note from the examples above that the replacement text can be several lines long. If you want to position the cursor somewhere you can use the `#` character. So the word `head` at the beginning of a file will be changed into:

```
=====
* File    : AgMemod.gms
* Author  : Wietse Dol (w.dol@wur.nl)
* Version : 1.0
* Date    : 12/16/2005 11:03:35 AM
* Changed :
* Changed by:
=====
```

and the cursor is at the place where you can enter remarks. This example shows other powerful features. You can insert code templates within a code template, e.g. the `<%author%>` in the `head` example is changed into `NacquIT`. Just to finish the function we have some predefined word, e.g.:

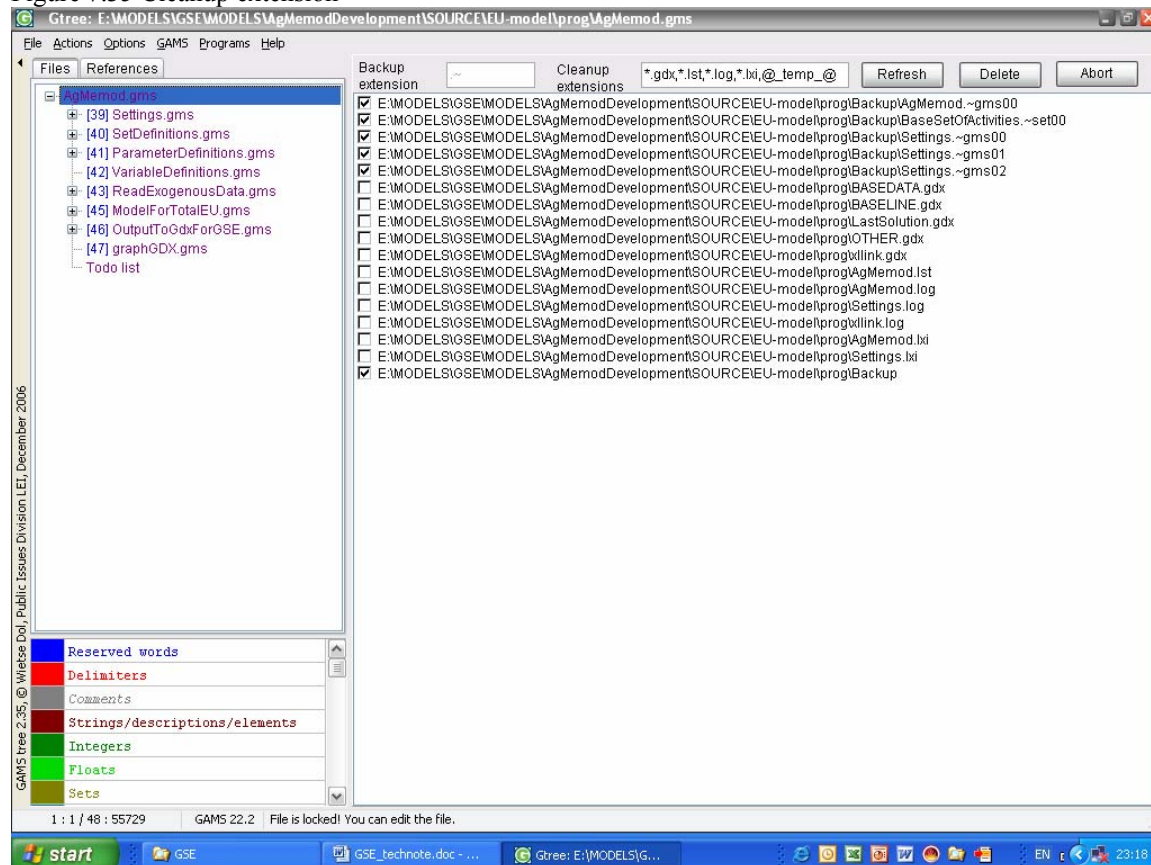
```
<%file%> show the filename including the path,
<%filename%> show the filename without the path,
<%pathname%> show the path of the file,
<%fileext%> show the file extension,
<%date%> show the current date,
<%time%> show the current time,
<%datetime%> show the current date and time.
```

In the code template editor you can use the grid to change the text and words. By pressing *Add* an extra row is added to the grid, so you can add a new code template. To delete a code template click in the grid on the template you want to delete and then press *Remove*. *Abort* will close the code template window without saving any changes made in this window. *Insert* will paste the result of the currently selected code template into the editor, save the changes and close the window. *Save* will save the changes and close the window. If you want multi-line text it can be handy to edit the text not in the grid but in a separate editor. To do so press *Edit* and when you have finished press *Save* to go back to the template grid.

7.2.11. Cleanup files

By default a list of all backup files and a list of temp directories created by GAMS (225a, 225b etc.) is shown (in the model directory and all its subdirectories). A list of files defined in the Cleanup extensions edit box is also shown (Figure 7.35). To change/add/edit the Cleanup extensions, remember that the list should be comma separated and that after changing you should press *Refresh* to refresh the list of files that can be selected for deletion. Press *Delete* to delete the selected files and press *Abort* to close this window and return to the editor.

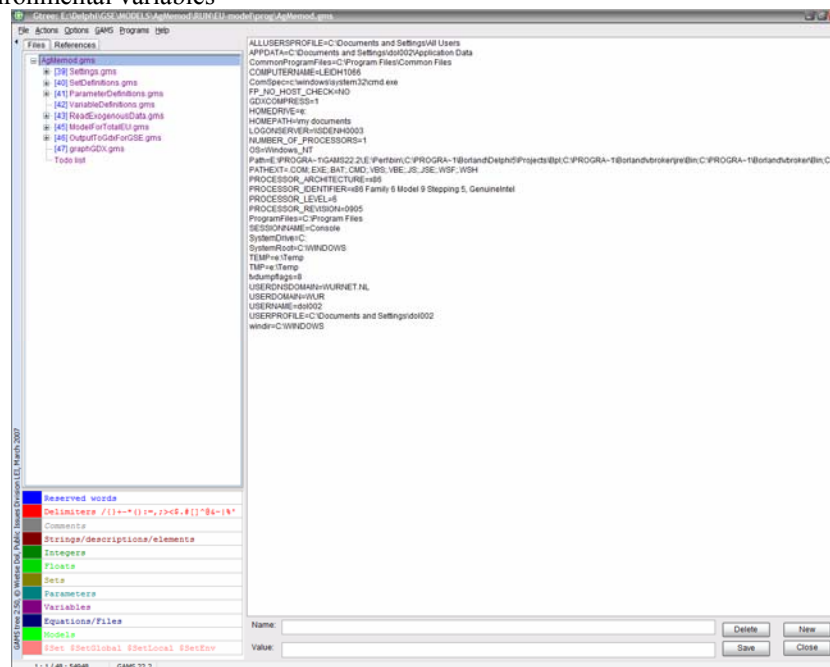
Figure 7.35 Cleanup extension



7.2.12. Environment variables

In the Figure 7.36 you can see a list of all available environment variables (and their values). By clicking on any of them you can edit the value (in the value edit box) and then save the changes by pressing *Save*. If you want a new environment variable press *New*, specify the name and value and then press *Save* again. Pressing *Delete* will delete the selected environment variable.

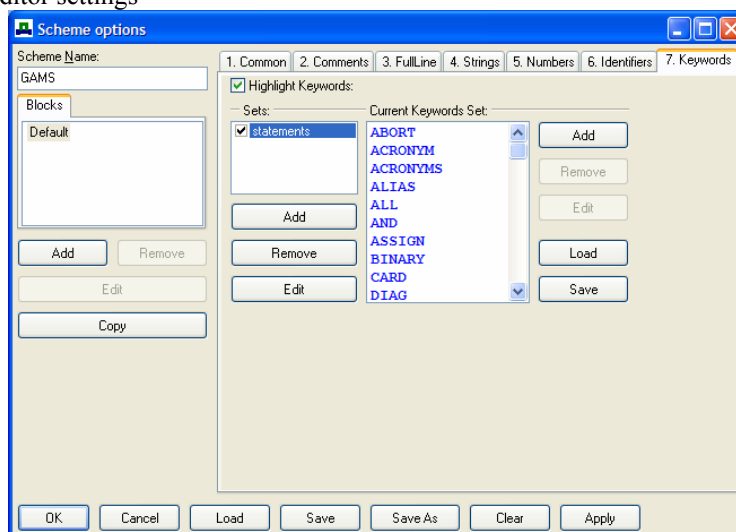
Figure 7.36 Environmental variables



7.2.13. Syntax editor settings

Gtree comes with roughly 35 predefined syntax colourings. If you want more or do not like the current scheme you can change it using this menu item. An example of the GAMS colouring is set out in the Figure 7.37. Try to change the function. Most of the steps are just intuitive.

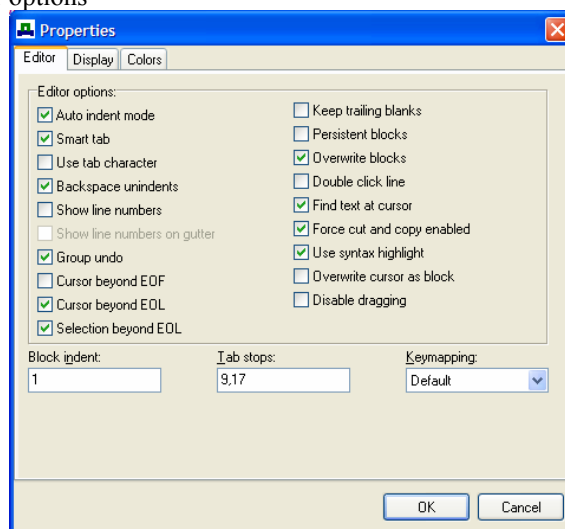
Figure 7.37 Syntax editor settings



7.2.14. Customise editor options

If you want to change the behaviour of the editor press this menu item and change the settings (Figure 7.38).

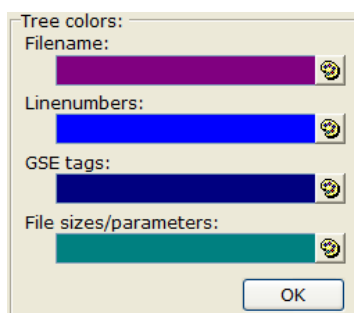
Figure 7.38 Customise editor options



7.2.15. Trees colours

Here you can change the colours used in the trees. Press *tree colours* and select the colour you want (Figure 7.39).

Figure 7.39 Colours in the trees



7.2.16. GAMS options

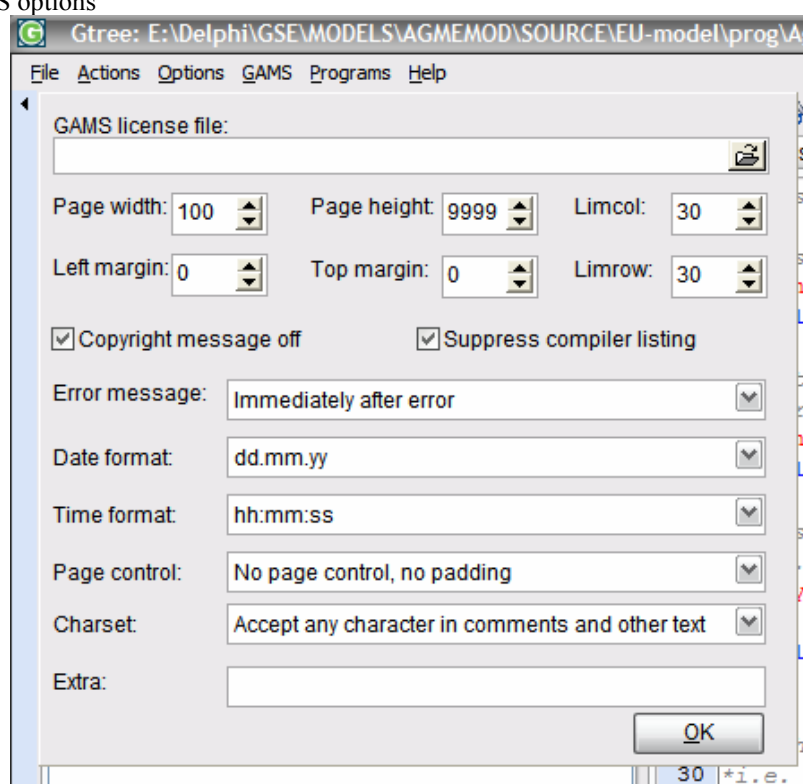
In the window of the

Figure 7.40 you can very easily select a number of important GAMS options, such as:

- Where to read the GAMS licence file;
- The page width and height;
- The left and top margins;
- How to display GAMS errors;
- The date and time format;
- Page control; and
- The character set.

In the *extra* field further options can be added by hand (read the Bruce McCarl manual for a detailed description of all available GAMS options).

Figure 7.40 GAMS options



7.2.17. GAMS library

GAMS is installed with a library containing numerous useful examples. You can use this window to look at the descriptions of these examples and, if you are interested, you can press *Load* to view the currently selected example in the Gtree editor (Figure 7.41). *Close* will close the GAMS library window. Since there are many examples the filter can be used to find any given example. Note that the filter works on the column that is used for sorting (in this example the Name column, as indicated by the triangle). By pressing on a column header you will sort that column (in ascending or descending order, again by clicking on the column header). Note that this is a SQL-type filter, so wild cards like **BA** are allowed.

Figure 7.41 GAMS model library

Name /	Application Area	Type	Contributor	Description
ABEL	Macro Economics	NLP	Kendrick, D	Linear Quadratic Control Problem
ABSMIP	Mathematics	MIP	GAMS Develop	Discontinuous functions abs() min() max() sign() as MIPs
AGRESTE	Agricultural Economics	LP	Kutcher, G P	Agricultural Farm Level Model of NE Brazil
AIRCRAFT	Management Science and OR	LP	Dantzig, G B	Aircraft Allocation Under Uncertain Demand
AIRSP	Stochastic Programming	OSLSE	Dantzig, G B	Aircraft Allocation
AIRSP2	Stochastic Programming	DECIS	Dantzig, G B	Aircraft Allocation - stochastic optimization with DECIS
AJAX	Management Science and OR	LP	CDC	Ajax Paper Company Production Schedule
ALAN	Finance	MINLP	Manne, A S	A Quadratic Programming Model for Portfolio Analysis
ALKYL	Chemical Engineering	NLP	Berna, T	Simplified Alkylation Process
ALPHAMET	Recreational Models	MIP	de Wetering,	Alphametics - a Mathematical Puzzle
ALUM	International Trade	MIP	Brown, M	World Aluminum Model
AMPL	Management Science and OR	LP	Fourer, R	AMPL Sample Problem
ANDEAN	Micro Economics	MIP	Mennes, L B	Andean Fertilizer Model
APL1P	Stochastic Programming	DECIS	Infanger, G	Stochastic Programming Example for DECIS
APL1PCA	Stochastic Programming	DECIS	Infanger, G	Stochastic Programming Example for DECIS
BADMIP	Management Science and OR	MIP	Neumaier, A	Rounding Problems in MIPs
BATCHDES	Chemical Engineering	MINLP	Kocis, G R	Optimal Design for Chemical Batch Processing
BCHFCNET	Management Science and OR	MIP	Bussieck, M	Fixed Cost Network Flow Problem with Cuts using BCH Facility
BCHMKNAP	Management Science and OR	MIP	Beasley, J E	Multi knapsack problem using BCH Facility
BCHOIL	Management Science and OR	MIP	Brimberg, J	Oil Pipeline Design Problem using BCH Facility
BCHTLBAS	Management Science and OR	MINLP	Floudas, C A	Trim Loss Minimization with Heuristic using BCH Facility
BEARING	Engineering	NLP	Coello Coell	Hydrostatic Thrust Bearing Design for a Turbogenerator

Linear Quadratic Control Problem (ABEL,SEQ=64)

Linear Quadratic Riccati Equations are solved as a General Nonlinear Programming Problem instead of the usual Matrix Recursion.

Reference:
Kendrick, D, Caution and Probing in a Macroeconomic Model. Journal of Economic Dynamics and Control 4, 2 (1982).

7.2.18. Solvers

GAMS has many solvers and in this window (Figure 7.42) you will see:

- which solvers are available;
- if you have a full licensed version or a demo version of the solver;
- which kind of problems can be solved with the solver;
- and which solver is used by default in GAMS.

If you want to change the default solver, you can click on the grid to turn an available solver into a selected solver.

Figure 7.42 GAMS solvers

	Liscence	CNS	DNLP	LP	MCP	MINLP	MIP	MIQCP	MPEC	NLP	QCP	RMINLP	RMIP	RMIQCP
AMPL	Full	Not available	Not available	Not available	Not available	Not available	Not available		Not available	Not available		Not available	Not available	
BARON	Demo		Available	Available		Selected	Available	Selected		Available	Available	Available	Available	Available
BDMLP	Full			Available			Selected						Available	
CONOPT	Full	Selected	Selected	Selected						Selected	Selected	Selected	Selected	Selected
CONVERT	Full	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Not available	Not available
CPLEX	Demo			Available			Available	Available			Available		Available	Available
DECISC	Demo			Not available										
DECISM	Demo			Not available										
DICOPT	Demo					Available								
GAMSBAS	Full		Not available	Not available	Not available	Not available	Not available			Not available		Not available	Not available	
GAMSCHK	Full		Not available	Not available	Not available	Not available	Not available			Not available		Not available	Not available	
LGO	Demo		Available	Available						Available	Available	Available	Available	Available
MILES	Full				Selected									
MINOS	Full		Available	Available						Available	Available	Available	Available	Available
MOSEK	Demo		Available	Available			Available	Available		Available	Available	Available	Available	Available
MPECDUMP	Full	Not available	Not available	Not available	Not available	Not available	Not available		Not available	Not available		Not available	Not available	
MPSGE	Demo													
MPSWRITE	Full		Not available	Not available		Not available	Not available			Not available		Not available	Not available	
NLPEC	Full				Available				Selected					
OQNLP	Demo		Available			Available		Available		Available	Available	Available		Available
OSL	Demo			Available			Available						Available	
OSLSE	Demo			Not available						Not available			Not available	
PATH	Demo	Available			Available									
PATHNLP	Demo		Available	Available						Available	Available	Available	Available	Available
SBB	Demo					Available		Available						
SNOPT	Demo		Available	Available						Available	Available	Available	Available	Available
XA	Demo			Available			Available						Available	
XAPAR	Demo			Available			Available						Available	
XPRESS	Demo			Available			Available				Available		Available	Available

7.2.19. GAMS errors

In a perfect world no-one would make any mistakes. After creating our GAMS model in the editor we would press *Run* (or F9) and GAMS would solve our model and show us the LST file. We would read the list file and have a look at the results (e.g. a GDX file). Then perhaps we would change the contents of the model and start all over again. Gtree can do all that, but what if we are not that perfect and made an error in our GAMS code? Just look at what happens if we make a mistake in the Transport problem (Figure 7.43).

Figure 7.43 GAMS errors

```

transport.gms transport.lst
52 50 * a detailed study of consumer behaviour in these cities.
53 51 * The demand can be increased when starting an advertising campaign.
54 52 * <ENDINFO>
55 53 * </INPUT>
56 54 * <INPUT>
57 55 Table d(i,j) distance in thousands of miles
58 56 new-york chicago topeka
59 DGAMS Rev 136 MS Windows
60 A Transportation Problem (TRANSPORT_SEQ=1)
61 C o m p i l a t i o n
62
63
64 57 seattle 2.5 1.7 1.8
65 58 san-diego 2.5 1.8 1.4 ;
66 59 * <FORMAT $,###,##0.###>
67 60 * <DIMENSION 1000 miles>
68 61 * <INFO>
69 62 * The distance is calculated by the distance that is needed to
70 63 * transport a case from the plant to the town center.
71 64 * <ENDINFO>
72 65 * </INPUT>
73 66 * <INPUT>
74 67 Scalar f freight in dollars per case per thousand miles /90/ ;
75 68 * <FORMAT $,###,##0>
76 69 * <DIMENSION $ per 1000 miles>
77 70 * <INFO>
78 71 * This is the average transportation cost when using trucks to transport
79 72 * cases from the production plant to the cities. These costs can change
80 73 * when fuel prices and/or taxes change.
81 74 * <ENDINFO>
82 75 * </INPUT>
83 76
84 77 Parameter c(i,j) transport cost in thousands of dollars per case :
85 78
86 79 c(i,j) = 2*f * d(i,j) / 1000 ;
87 87 **** 6171
88 ..

```

Note that the error is shown on line 87 of the LST file (at the bottom of the screen, since errors are often the result of the previous code). In this example line 79 of the file contains error 171. On the left-hand side of the LST file you see that error 171 means “Domain violation for set”. So, we would immediately see that $d(j,j)$ should become $d(i,j)$. All we have to do is go to line 79 of the file and change the code. Since the model can use many $\$include$ statements it is not immediately clear which file contains the error. To make your life easier, press *Find GAMS error* (the green bug button) or click on the error explanation panel (here the bottom left-hand panel with Error: 171, etc.). As a result, you can edit the file on the error line and when you have corrected the error press F9 to run GAMS again to find the next error (Figure 7.44).

Figure 7.44 Find GAMS error

```

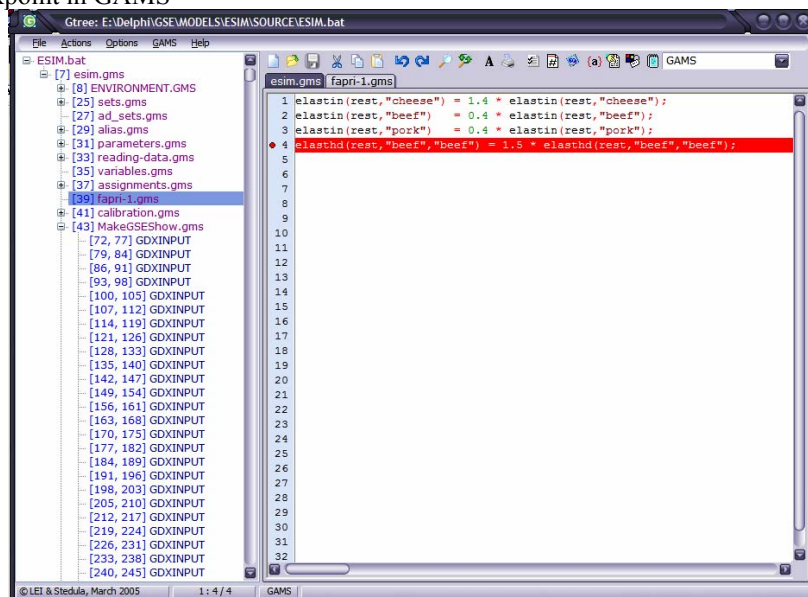
transport.gms
44 chicago 300
45 topeka 275 ;
46 * <FORMAT $,###,##0>
47 * <DIMENSION Cases per day>
48 * <INFO>
49 * The demand is the average predicted demand after
50 * a detailed study of consumer behavior in these cities.
51 * The demand can be increased when starting an advertising campaign.
52 * <ENDINFO>
53 * </INPUT>
54 * <INPUT>
55 Table d(i,j) distance in thousands of miles
56 57 new-york chicago topeka
57 58 seattle 2.5 1.7 1.8
58 59 san-diego 2.5 1.8 1.4 ;
59 * <FORMAT $,###,##0.###>
60 * <DIMENSION 1000 miles>
61 * <INFO>
62 * The distance is calculated by the distance that is needed to
63 * transport a case from the plant to the town center.
64 * <ENDINFO>
65 * </INPUT>
66 * <INPUT>
67 Scalar f freight in dollars per case per thousand miles /90/ ;
68 * <FORMAT $,###,##0>
69 * <DIMENSION $ per 1000 miles>
70 * <INFO>
71 * This is the average transportation cost when using trucks to transport
72 * cases from the production plant to the cities. These costs can change
73 * when fuel prices and/or taxes change.
74 * <ENDINFO>
75 * </INPUT>
76
77 Parameter c(i,j) transport cost in thousands of dollars per case :
78 78
79 c(i,j) = 2*f * d(i,j) / 1000 ;

```

7.2.20. Breakpoints in GAMS

GAMS is famous for clear mathematical notation, for solvers and for the speed of finding a solution, but GAMS certainly is not a modern programming language. Often people just ask: can I stop GAMS from running and inspect the outcome of a certain parameter/variable and then decide whether to continue or to stop running the model? With Gtree this is possible (and simple). Just hold the Ctrl key and click on the number of the line after which you want GAMS to stop (in this example line 4, as shown by the red dot before the line number and the fact that the line is displayed in red) (Figure 7.45). When you run GAMS (press F9) it will stop and show the current status of parameters, variables, sets and equations in the GDXviewer. After you close the GDXviewer, GAMS will ask if you want to stop the GAMS run or not. With this function, both beginners and advanced GAMS users can easily inspect what happens during a GAMS run (of particular interest within LOOPS).

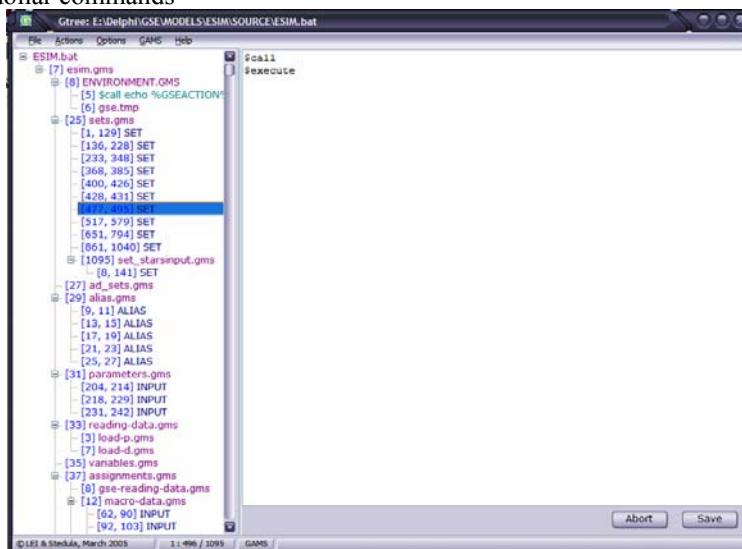
Figure 7.45 Breakpoint in GAMS



7.2.21. GAMS dollar commands

It is sometimes useful to see certain GAMS commands in the tree. Just enter a list of all the GAMS commands you want to see and press *Save*. In the screen (Figure 7.46), \$call and \$execute have been added to the list (as you can see in the tree there is a call statement in line 5 of the file ENVIRONMENT.GMS).

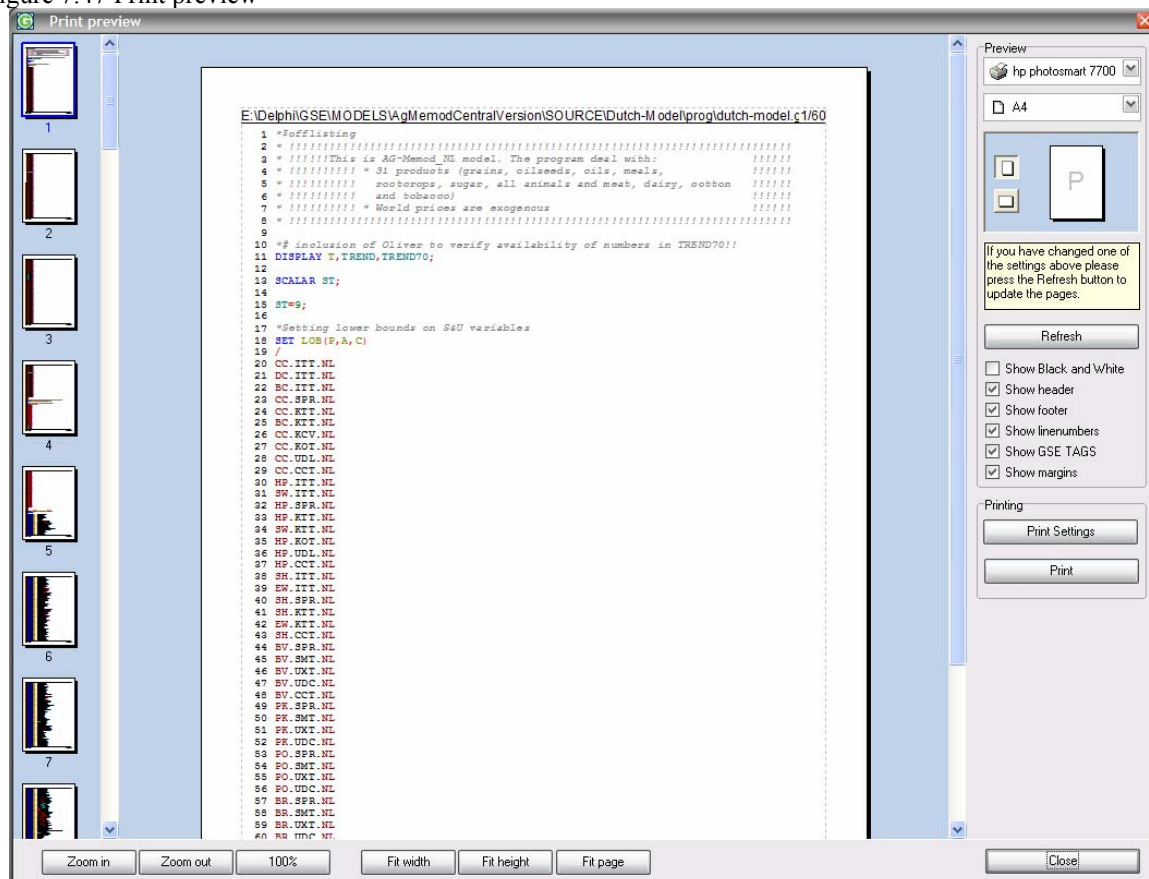
Figure 7.46 GAMS dollar commands



7.2.22. Print preview

If you want to print the contents of your editor this window shows you what will be sent to the printer (Figure 7.47). On the right-hand side you can select the print options and, when you are ready, you can press *Print* to send the outcome to the printer.

Figure 7.47 Print preview



7.2.23. Gtree and GSE

Since Gtree is a good alternative to GAMSIDE and can be used within GSE, combination of GSE and Gtree will give model-builders a powerful tool for making models, running scenarios and changing versions. To get a GAMS model to run in the GSE interface, all you have to do is add a few comments (called “tags”) to your GAMS files. This can be done by hand or using the TagEditor in Gtree. The following tags are supported:

```

SET
ALIAS
GROUPSET
INPUT
GDXINPUT
GDXOUTPUT
GDXSYMBOL
MENUITEM
MENURESULT
MENUORDER
PROGRAM
PARAMETERLIST
IMPLICIT OUTPUT FILE
EXPLICIT OUTPUT FILE
OUTPUT

```


Preparing a model for GSE is very easy with the Import2GSE tool. The most time-consuming part of import is inserting the right GSE tags into your files. These can generate a large number of errors in the form of typos or wrong tag options. Tags range from simple to very complex and, even after reading this documentation, it is not simple to remember all the tags and their options. To simplify insertion of tags we have developed the TagEditor. This should not be thought of as a powerful editor, but simply as a tool for inserting tags in your files. Do not rush to Gtree and the TagEditor immediately. First think for a while which parameters you want the model user to change (inputs) and which outputs you want to present to the model-user. Make a list of these parameters and write down in which files they are located (or find them using *Find*). Also write down which sets, aliases and elements you use. After completing these lists you can start using the TagEditor.

The basic idea of the TagEditor is simple: select the file in which you want to insert a tag, then select the text you want to put a tag around and press Ctrl /. The TagEditor will now be shown and if the tag has options, all the options are shown and you can select or edit them.

After starting Gtree and selecting the Transport model as an example we get Figure 7.48.

Figure 7.48 Model selection

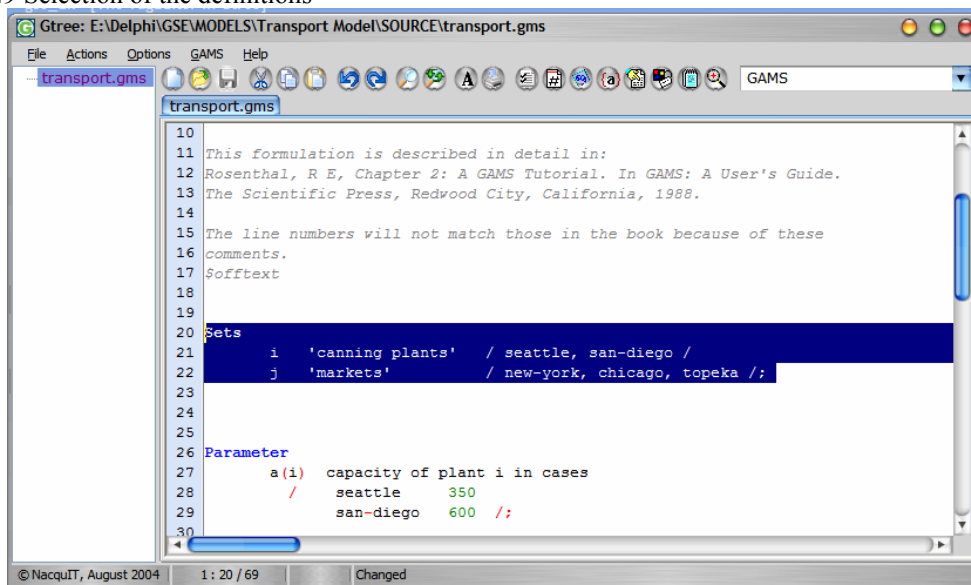
```

1 $Title A Transportation Problem (TRANSPORT,SEQ=1)
2
3 $ontext
4 This problem finds a least cost shipping schedule that meets
5 requirements at markets and supplies at factories.
6
7
8 Dantzig, G B, Chapter 3.3. In Linear Programming and Extensions.
9 Princeton University Press, Princeton, New Jersey, 1963.
10
11 This formulation is described in detail in:
12 Rosenthal, R E, Chapter 2: A GAMS Tutorial. In GAMS: A User's Guide.
13 The Scientific Press, Redwood City, California, 1988.
14
15 The line numbers will not match those in the book because of these
16 comments.
17 $offtext
18
19
20 Sets
21 i 'canning plants' / seattle. san-diego /

```

Now select the two set definitions (Figure 7.49).

Figure 7.49 Selection of the definitions



The screenshot shows the Gtree editor window titled 'Gtree: E:\Delphi\GSE\MODELS\Transport Model\SOURCE\transport.gms'. The editor displays the following GAMS code:

```

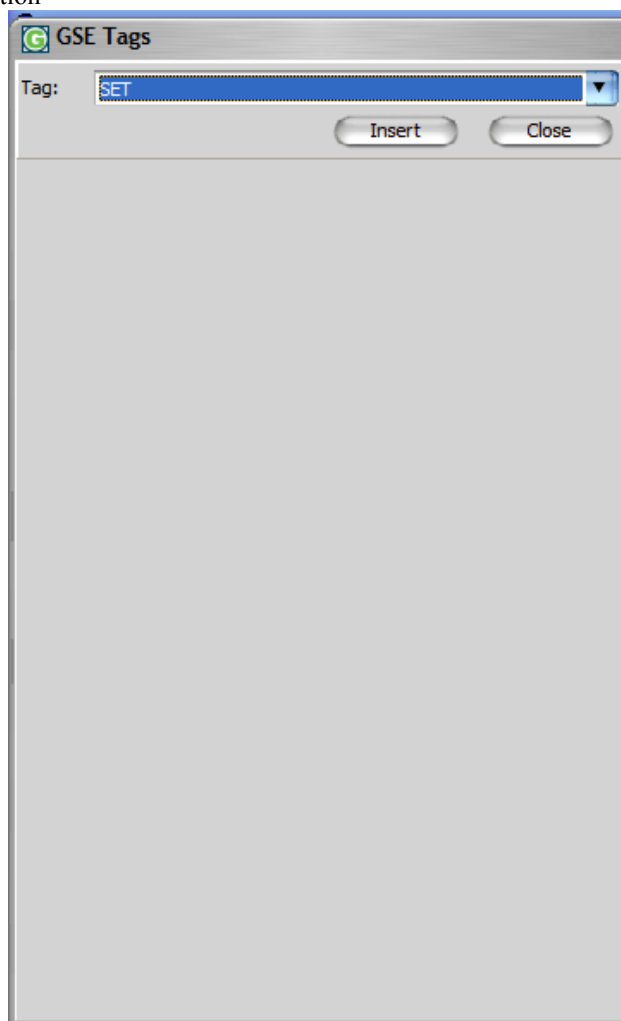
10
11 This formulation is described in detail in:
12 Rosenthal, R E, Chapter 2: A GAMS Tutorial. In GAMS: A User's Guide.
13 The Scientific Press, Redwood City, California, 1988.
14
15 The line numbers will not match those in the book because of these
16 comments.
17 $offtext
18
19
20 Sets
21   i  'canning plants' / seattle, san-diego /
22   j  'markets' / new-york, chicago, topeka /;
23
24
25
26 Parameter
27   a(i) capacity of plant i in cases
28   /   seattle   350
29     san-diego  600 /;
30

```

The lines 21 and 22 are highlighted in blue, indicating they are selected. The status bar at the bottom shows '© NacquiIT, August 2004', '1: 20 / 69', and 'Changed'.

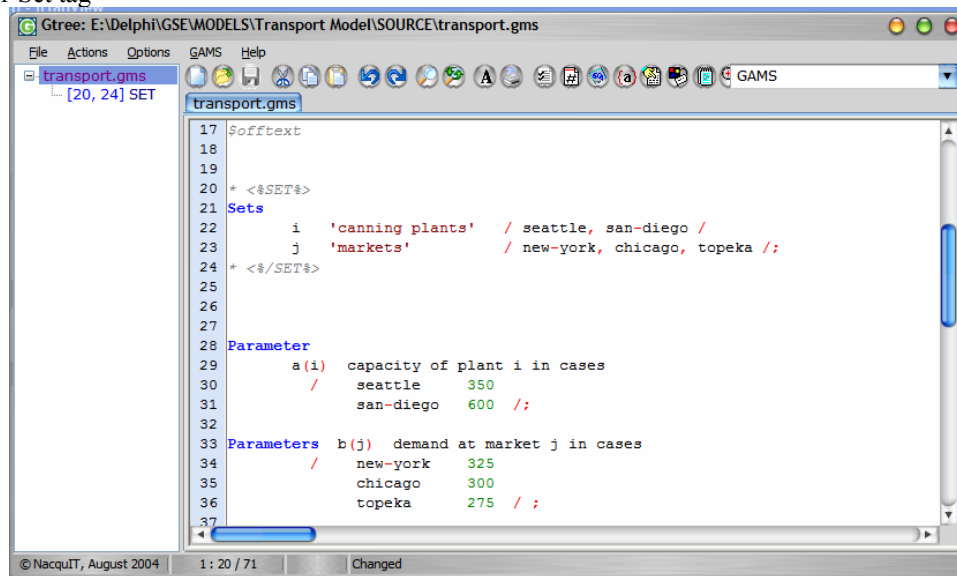
Press Ctrl /

Figure 7.50 GSE tag selection



From the combo box we select the tag we want (in our case the SET tag) (Figure 7.50) and then press *Insert* (Figure 7.51).

Figure 7.51 Set tag



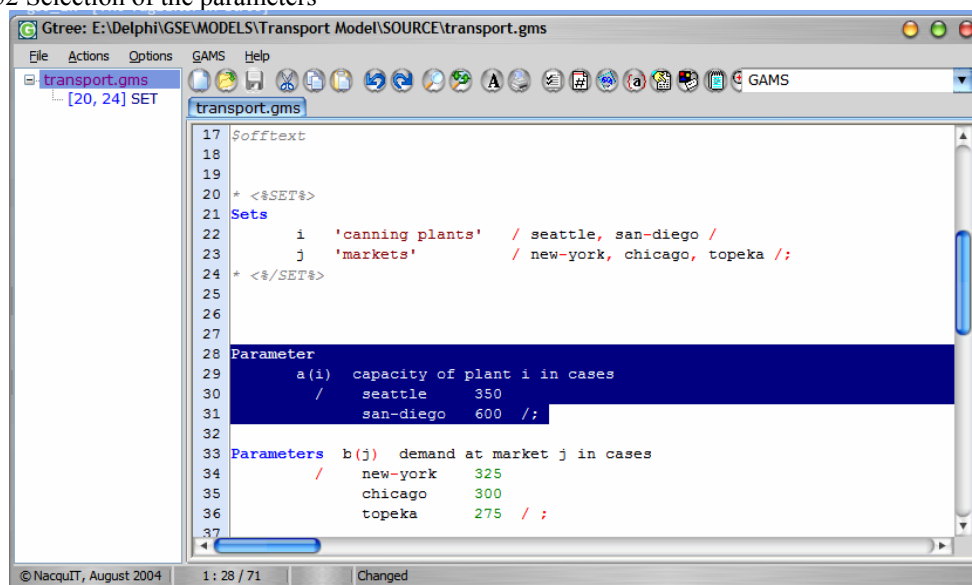
```

17 $offtext
18
19
20 * <%=SET%>
21 Sets
22     i  'canning plants' / seattle, san-diego /
23     j  'markets'       / new-york, chicago, topeka /;
24 * <%=SET%>
25
26
27
28 Parameter
29     a(i)  capacity of plant i in cases
30           / seattle    350
31           / san-diego  600 /;
32
33 Parameters b(j) demand at market j in cases
34           / new-york   325
35           / chicago    300
36           / topeka    275 /;
37

```

Now we select parameter a(i):

Figure 7.52 Selection of the parameters



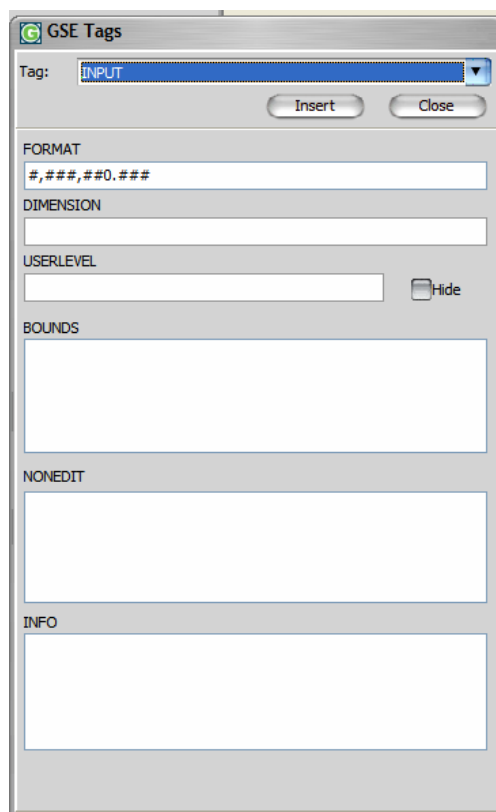
```

17 $offtext
18
19
20 * <%=SET%>
21 Sets
22     i  'canning plants' / seattle, san-diego /
23     j  'markets'       / new-york, chicago, topeka /;
24 * <%=SET%>
25
26
27
28 Parameter
29     a(i)  capacity of plant i in cases
30           / seattle    350
31           / san-diego  600 /;
32
33 Parameters b(j) demand at market j in cases
34           / new-york   325
35           / chicago    300
36           / topeka    275 /;
37

```

and after pressing Ctrl / and selecting the INPUT tag from the combo box we get:

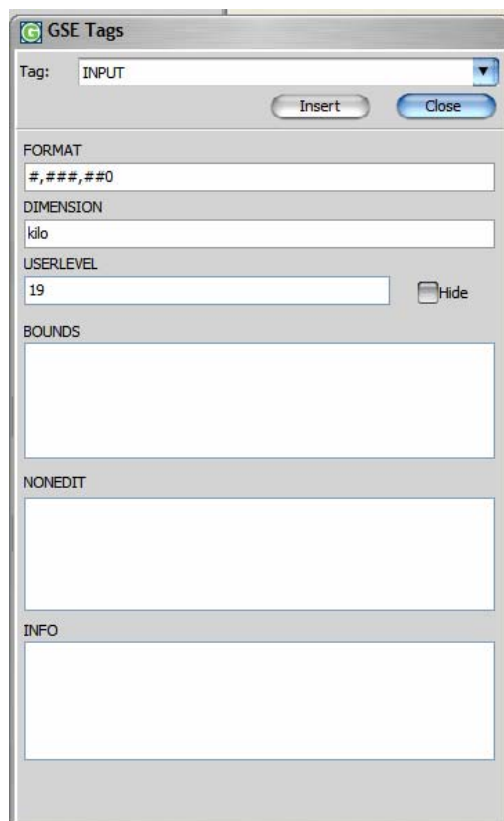
Figure 7.53 Input tag



The screenshot shows the 'GSE Tags' dialog box. At the top, the 'Tag:' dropdown menu is set to 'INPUT'. Below it are 'Insert' and 'Close' buttons. The dialog is divided into several sections: 'FORMAT' with a text field containing '#,###,##0.###'; 'DIMENSION' with an empty text field; 'USERLEVEL' with a text field and a 'Hide' checkbox; 'BOUNDS' with a large empty text area; 'NONEDIT' with a large empty text area; and 'INFO' with a large empty text area.

Now we can enter the tag options:

Figure 7.54 Input tag options



The screenshot shows the 'GSE Tags' dialog box with the following options entered: 'Tag:' is 'INPUT'; 'FORMAT' is '#,###,##0'; 'DIMENSION' is 'kilo'; 'USERLEVEL' is '19'; and the 'Hide' checkbox is checked. The 'Insert' and 'Close' buttons are visible at the top right.

and press *Insert*.

Figure 7.55 Updated model tree

```

26
27
28 * <%=INPUT%>
29 Parameter
30     a(i)  capacity of plant i in cases
31         /  seattle    350
32           san-diego  600  /;
33 * <%=FORMAT #,###,##0%>
34 * <%=DIMENSION kilo%>
35 * <%=USERLEVEL 19%>
36 * <%=INPUT%>
37
38 Parameters b(j)  demand at market j in cases
39         /  new-york    325
40           chicago     300
41           topeka      275  /;
42
43
44 Table d(i,j)  distance in thousands of miles
45         new-york    chicago    topeka
46 seattle    2.5        1.7        1.8

```

You see how easy it is! After inserting the tag, the tree is automatically updated and we have a SET tag on lines 20-24 and an input tag on lines 28-36. If you want to, you can change a tag already inserted by hand, but it is easier to place the cursor anywhere between the begin and end tags and press Ctrl / Again the TagEditor is started but now all the options already used/inserted are shown and you can start to edit/add/change the tag options.

Inserting tags is not difficult (if you want to know what all the tag options mean, just start the help file and look in the Technical Manual). One tag needs special attention: the GDXOUTPUT tag. Any mistakes you make in the tags will be recognised by GSE when you try to insert a version of the model into GSE, i.e. after parsing your model GSE will show you where you made a mistake. In the GDXOUTPUT tag you specify which parameters and variables you save as outputs and want to show to the user. GSE cannot check if you misidentify a variable as a parameter (and vice versa). Hence you could possibly successfully insert a version of the model into GSE and, after running a scenario, get an error when you save the results. This is undesirable, and the TagEditor can therefore help you to avoid making this mistake.

Figure 7.56 Selection of the GDXOUTPUT tag

The screenshot shows the 'GSE Tags' dialog box. At the top, the 'Tag:' dropdown menu is set to 'GDXOUTPUT'. Below this are 'Insert' and 'Close' buttons. The 'OUTPUT file name' field contains 'gse.gdx'. The 'PARAMETER/VARIABLE' dropdown is currently empty. Below these are 'New', 'Delete', and 'GAMS' buttons. A section titled 'Current settings:' contains radio buttons for 'PARAMETER' and 'VARIABLE', with 'PARAMETER' selected. Below this are fields for 'GAMS name', 'Description', 'FORMAT' (with the value '#,###,#0.###'), and 'DIMENSION'. There is also a 'USERLEVEL' field and a 'Hide' checkbox. At the bottom, there are 'BOUNDS' and 'INFO' sections, both currently empty.

After selecting the GDXOUTPUT tag in the TagEditor (Figure 7.56), press *GAMS*. Gtree will run GAMS and make a reference file, parse this file and show you all the parameters and variables that are defined in your model (Figure 7.57).

Figure 7.57 GDXOUTPUT reference file of model parameters and variables

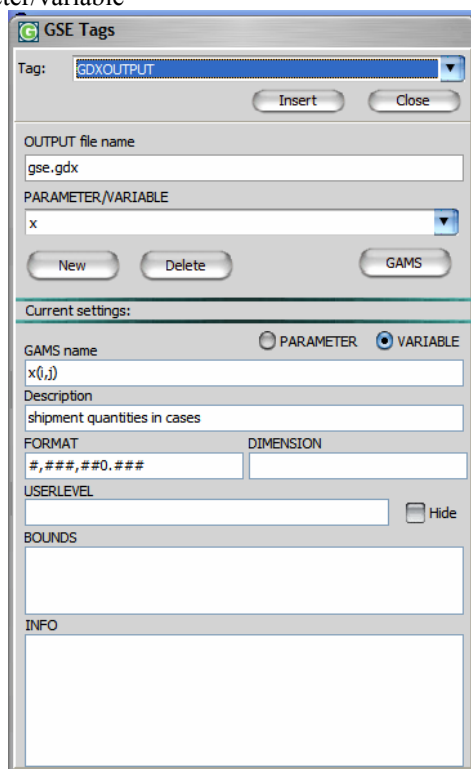
The screenshot shows the 'GSE Tags' dialog box with the 'GDXOUTPUT' tag selected. The main area displays a list of parameters and variables with checkboxes for selection. The list is as follows:

<input type="checkbox"/>	a: capacity of plant i in cases	parameter
<input type="checkbox"/>	b: demand at market j in cases	parameter
<input type="checkbox"/>	c: transport cost in thousands of dollars per case	parameter
<input type="checkbox"/>	d: distance in thousands of miles	parameter
<input type="checkbox"/>	f: freight in dollars per case per thousand miles	parameter
<input checked="" type="checkbox"/>	x: shipment quantities in cases	variable
<input checked="" type="checkbox"/>	z: total transportation costs in thousands of dollars	variable

At the bottom of the dialog, it shows '# PARAMETERS = 5' and '# VARIABLES = 2'. There are checkboxes for 'Show name' and 'Show description', both of which are checked. 'Abort' and 'OK' buttons are at the bottom right.

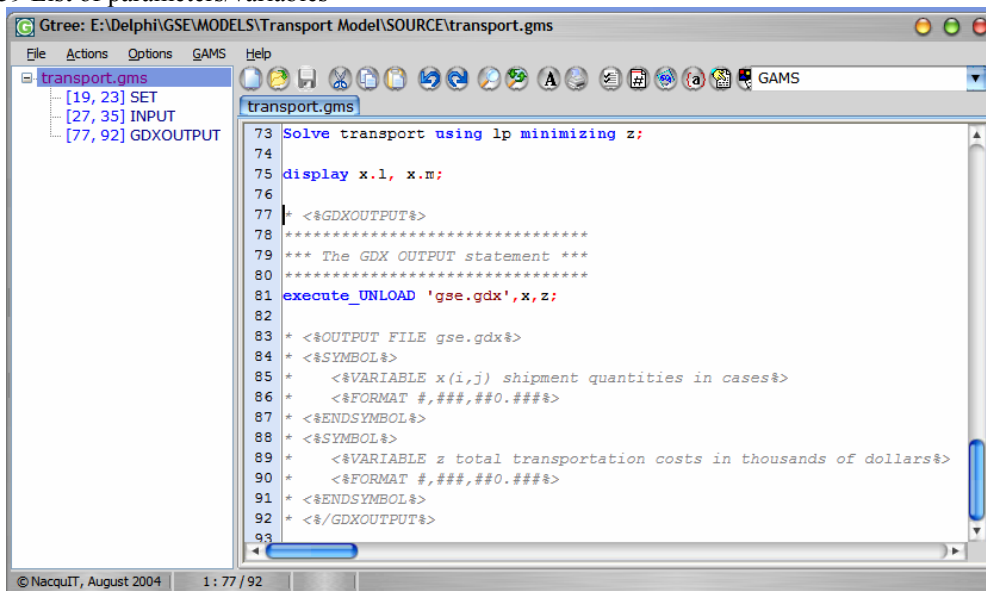
Just check which variables/parameters you want to show/put in a GDX file and click on *OK*.

Figure 7.58 Selection of a parameter/variable



The PARAMETER/VARIABLE combo box offers a list of all the outputs and when one of them is selected, the options (meta-information) of this output parameter/variable are shown in the *Current settings* (Figure 7.58). Just change/add the options for all your outputs and after pressing *Insert* you get Figure 7.59.

Figure 7.59 List of parameters/variables



So inserting output parameters/variables is reduced to selecting them from a list (Figure 7.59).

7.2.24. *New file in tree template*

You can select a file in which a template of a model structure you want to create is declared. Suppose that for a new micro-economic model we always want to use the same tree structure and that we create a file called `MicroWave.template`. The content of the file looks like this:

- 1 Sets and Elements.set
- 2 Declarations.gms
 - 2.1 Stateblock Parameters.par
 - 2.2 Exogenous.par
 - 2.2.1 Counters.par
 - 2.2.2 Constants.par
 - 2.2.3 Standards.par
 - 2.2.4 Trends.par
 - 2.2.5 Policy.par
 - 2.2.6 Model Initials.par
 - 2.3 Endogenous.par
 - 2.3.1 Model Parameters.par
 - 2.3.2 Variables.par
 - 2.3.3 Aggregation Parameters.par
 - 2.4 Equations.equ
 - 2.5 Models.mod
 - 2.6 Other.par
- 3 Import data.inp
 - 3.1 Loop Initialisation.gms
 - 3.2 Data Initialisation.gms
 - 3.3 Initialisation schemes.gms
- 4 MicroWave Model.gms
 - 4.1 Base Period Initialisation.gms
 - 4.2 Transfer from Statblock to Begin Period Model.gms
 - 4.3 Period Model.gms
 - 4.4 Transfer from End Period Model to Stateblock.gms
 - 4.5 Aggregation.gms
- 5 Output.gms

When this file is selected, a complete structure of all these files is created and shown in Gtree.

7.2.25. Keyboard shortcuts for Gtree

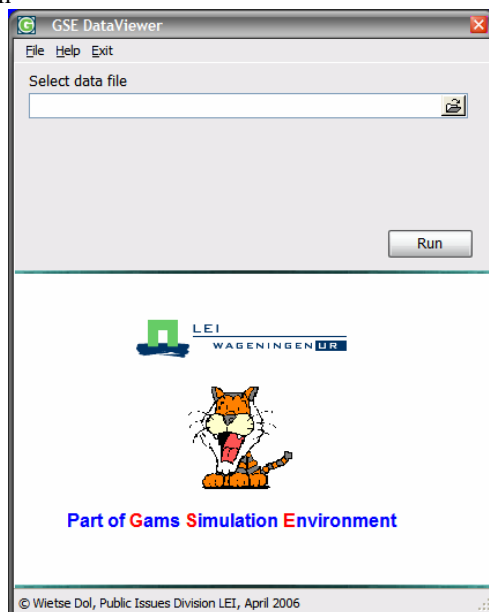
Key	Explanation
Alt-R	Search the reference tree for the selected word or the word at the current cursor
Ctrl-Alt-R	Go back to line/file from last Alt-R
Alt-I	Include file (add \$include command in current editor)
F1	Help
F2	Insert code template
Shift-F2	Code template editor
F3	Find and replace
F4	Start external GDX viewer
Shift-F4	Start external editor
F5	Refresh all files
F6	Refresh current file
F7	Show GAMS LST file
F8	Show GDX file
F9	Run project file
Shift-F9	Compile project file
Alt-F9	Jump to GAMS error
F10	Run current file
Shift-F10	Compile current file
F11	Delete backup files
Shift-F11	Restore backup file
F12	Create code template/syntax colouring from model tree
Ctrl-B	Start last GDX file
Ctrl-E	Exit
Ctrl-/	Start GSE tag editor
Ctrl-S	Save files
Ctrl-Z	Undo
Ctrl-U	Redo
Ctrl-G	Go to line
Ctrl-F	Find text in documents
Ctrl-Alt-F	Find text in tree
Ctrl-R	Replace
Ctrl-L	Next for find/replace
Ctrl-Y	Delete line
Ctrl-(Ctrl- Ctrl-{ Ctrl-“ Ctrl-‘	Find matching character
Ctrl-N	New file
Ctrl-O	Open file
Ctrl-P	Print
Ctrl-M	Hide/show tree panel
Ctrl-Alt-M	Hide/show legend panel
Ctrl-X	Cut
Ctrl-C	Copy
Ctrl-V	Paste
Ctrl-Alt-S	Select skin for windows
Ctrl-K 1	Define bookmark 1. There are nine bookmarks (1 .. 9)
Ctrl-1	Go to bookmark 1
Alt + Left mouse	Select block
Ctrl-Shift-L	Copy current line
Ctrl-U	Put selected block in uppercase
Ctrl-L	Put selected block in lowercase
Ctrl-T	Switch characters
Alt-T	Switch words
Ctrl-D	Open MS-dos box
Ctrl-J	Code completion for sets, parameters, variables, etc.
Alt-0, Alt-1, Alt-2 etc.	Collapse/expand a tree at level 0, 1, 2, etc.
Ctrl-↑ Ctrl-↓ Alt-↑ Alt-↓	Previous, Next *! Comment; Previous, Next *? comment
Ctrl “line number”	Mark line as breakpoint
Ctrl-I	Find the word “infes” in the current file
Ctrl-F10	Start macro key recording
Ctrl-Break	Stop macro key recording
Ctrl-F11	Execute macro
Ctrl-F12	Repeat macro

7.2.26. DataViewer

The DataViewer is part of GSE and Gtree but can also be used as a stand-alone tool for viewing data (Figure 7.60). The current DataViewer supports the following formats:

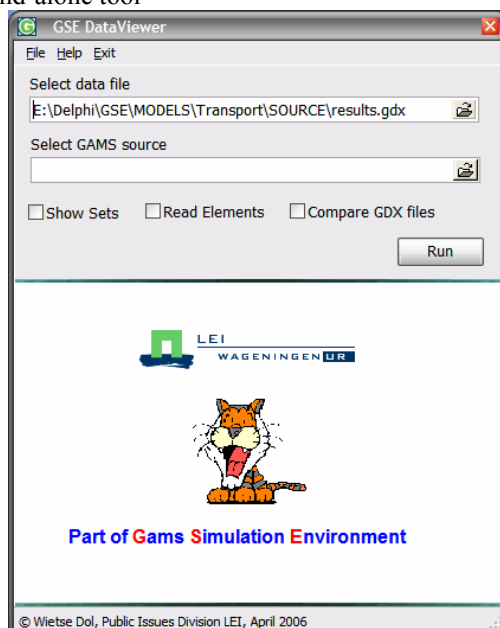
- GDX files;
- Access databases;
- XML files;
- CSV files.

Figure 7.60 DataViewer selection



To use the DataViewer as a stand-alone tool, first select the datafile you want to view and then press *Run* to read the data and show the result in the GSE Viewer. If you are using a GDX file you will have additional options, as shown in the next figure.

Figure 7.61 DataViewer as a stand-alone tool



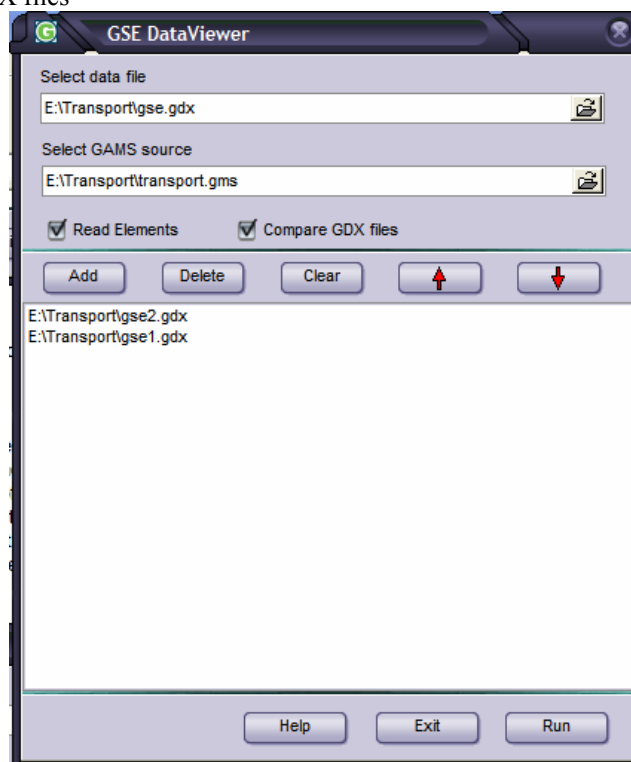
GDX files are a very easy way to save output from a GAMS program/model and are also a very fast and compressed way to store GAMS input data. GDX files lack certain meta-information to show you their contents

clearly. By combining the GDX file with its GAMS code you can display the GDX file in the GSE Viewer with all its meta-information (i.e. the DataViewer will create and parse a GAMS reference file from the GAMS source). To specify which GAMS file is used for the model/meta-information, use the *Select GAMS source* edit box. When you specify a GAMS source this file is executed by GAMS and a reference file (called gse.ref) is created. This reference file is used to locate where sets, parameters, variables and other meta-information are defined. *If you specify a file with the file extension .REF as the GAMS source the DataViewer will not run GAMS but will use the contents of the specified reference file immediately.*

You also have three additional checkboxes. The first – *Show Sets* – will not only show the parameters, variables and equations in the output, but also add all the defined sets to the output. The second – *Read Elements* – will ensure that all elements, sets and subsets that are defined in the GAMS source are read. The default is that only sets that are directly used in parameters, variables and equations are read and that elements are not read.

The GSE viewer is extremely powerful and GSE is the ideal tool for running and comparing scenarios. The only disadvantage of GSE is that model-builders do not like to save/store all their preliminary versions in GSE and use GSE to run and compare scenarios. This is too time-consuming and makes the GSE database unnecessarily big and complex. To help model-builders, you can use the DataViewer: run your scenario/model and save your inputs/outputs into a GDX file. When running different scenarios, change the filename of this GDX file. After selecting the GDX file of the base scenario in the DataViewer check *Compare GDX files* and the DataViewer window will change (Figure 7.62)

Figure 7.62 Compare GDX files



Now you can start and *Add* additional GDX files to the list, *delete* a selected GDX file from the list, *clear* the whole list and/or change the order of the GDX files. Press *Run* and the DataViewer will load and compare all the GDX files/scenarios.

7.2.27. Start parameters

When using the DataViewer you can start the program with parameters. This makes it possible to automate viewing of your datafiles. The syntax is:

```
dataviewer.exe "datafile" ["gams source file"] [/INIFILE="y\x.z"] [/PARFILE="y\x.z"] [/GDX="gdx1,gdx2,..."]
[/REF] [/ELEMENTS] [/SETS]
```

The parameters between brackets [] are optional. Do not type those brackets when using the DataViewer with startup parameters. Also note that the quotes around the parameters are optional, i.e. if you specify a parameter that contains no spaces (in the filename) you can skip those quotes. However, if you have a space in your filename you need to use the quotes.

datafile Specify here the filename that you would otherwise specify in the *Select data file* field.

gams source file Specify here the filename that you would otherwise specify in the *Select GAMS source* field. If you leave out the path the DataViewer will assume the same path as for the datafile.

/INIFILE By default the DataViewer.ini is read for all kinds of startup settings. You can use this parameter if you want to use not this default INI file but another one. The y\x.z is any full path + filename. If you leave out the path the DataViewer will assume the same path as for the datafile.

/PARFILE You can start the DataViewer and specify a parameter list file that has to be used to present the data selection.

/GDX Here you specify a comma-separated list of.gdx files you want to compare with the datafile.

/REF If you use this option the DataViewer will not run GAMS and generate a reference file, but instead will use the reference file already specified at *GAMS source file*.

/ELEMENTS This has the same effect as checking *Read Elements*.

/SETS This has the same effect as checking *Show Sets*.

N.B. When distributing your GDX files with the DataViewer you can use the MakeLink.exe program to create a shortcut on your desktop. All the shortcut start parameter settings are specified in the MakeLink.INI file.

Figure 7.63 CSV/ASCII files

File: E:\Delphi\GSE\delimited_data.txt

```

Sales Variable=20,Q1=10,Q2=10,Q3=10,Q4=10,Total=10
Seasonality,0.9,1.1,0.8,1.2,
"Units Sold","3,592","4,390","3,192","4,789","15,962"
"Sales Revenue",0,0,0,0,0
"Cost of Sales",0,0,0,0,0
"Gross Margin",0,0,0,0,0
Salesforce,"8,000","8,000","9,000","9,000","34,000"
Advertising,"10,000","10,000","10,000","10,000","40,000"
"Corp Overhead",0,0,0,0,0
"Total Costs","18,000","18,000","19,000","19,000","74,000"
"Prod. Profit","-18,000","-18,000","-19,000","-19,000","-74,000"
    
```

File columns are: Fixed Size Delimited

Delimiter options: Tab Semicolon Comma Space Other

Schema options: Use first line as schema

Sales Variable	Q1	Q2	Q3	Q4	Total
Seasonality	0.9	1.1	0.8	1.2	
Units Sold	3,592	4,390	3,192	4,789	15,962
Sales Revenue	0	0	0	0	0
Cost of Sales	0	0	0	0	0
Gross Margin	0	0	0	0	0
Salesforce	8,000	8,000	9,000	9,000	34,000
Advertising	10,000	10,000	10,000	10,000	40,000
Corp Overhead	0	0	0	0	0
Total Costs	18,000	18,000	19,000	19,000	74,000
Prod. Profit	-18,000	-18,000	-19,000	-19,000	-74,000

Index/set:

- Record number
- Schema
- Sales Variable
- Q1
- Q2
- Q3
- Q4
- Total

Name: delimited_data Description:

Buttons: Apply, Load..., Save, Abort, Run

8. Implementation of GSE in AGMEMOD

Using GSE means that the mathematical formulation of the model must be put in GAMS code in the implementation phase of the model-building process. The model-building protocol has been followed, from context analysis, conceptualisation, information analysis and mathematical modelling to GAMS implementation.

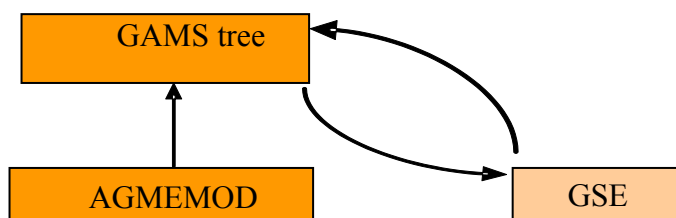
Short description of AGMEMOD

The system has been largely econometrically estimated at individual Member State level and produces results for the EU as a whole. The country models contain operators' behavioural responses to changes in prices, policy instruments and other exogenous variables on the agricultural market. Commodity prices adjust so as to clear all markets considered. For each commodity modelled and in each country, the system generates the main domestic market variables, such as production, demand for food and feed, prices, trade and stocks. Agricultural income is calculated at sector level.

There was no need to build the AGMEMOD model from scratch: for the old Member States AGMEMOD models were already available in GAMS code (GAMS-IDE), whereas the models for eight new Member States of 2004 and 2007 EU enlargements have been put into GAMS as part of this study. So far, each country model has been migrated from Excel to GAMS, equation by equation. That means that the EU-15 model could be seen as a sequence of the complete set of country equations without considering adjustment of (parts of) the commodity models to a more generic structure. Over the longer term, however, this is expected to become a severe problem when the EU-25 combined model is developed.

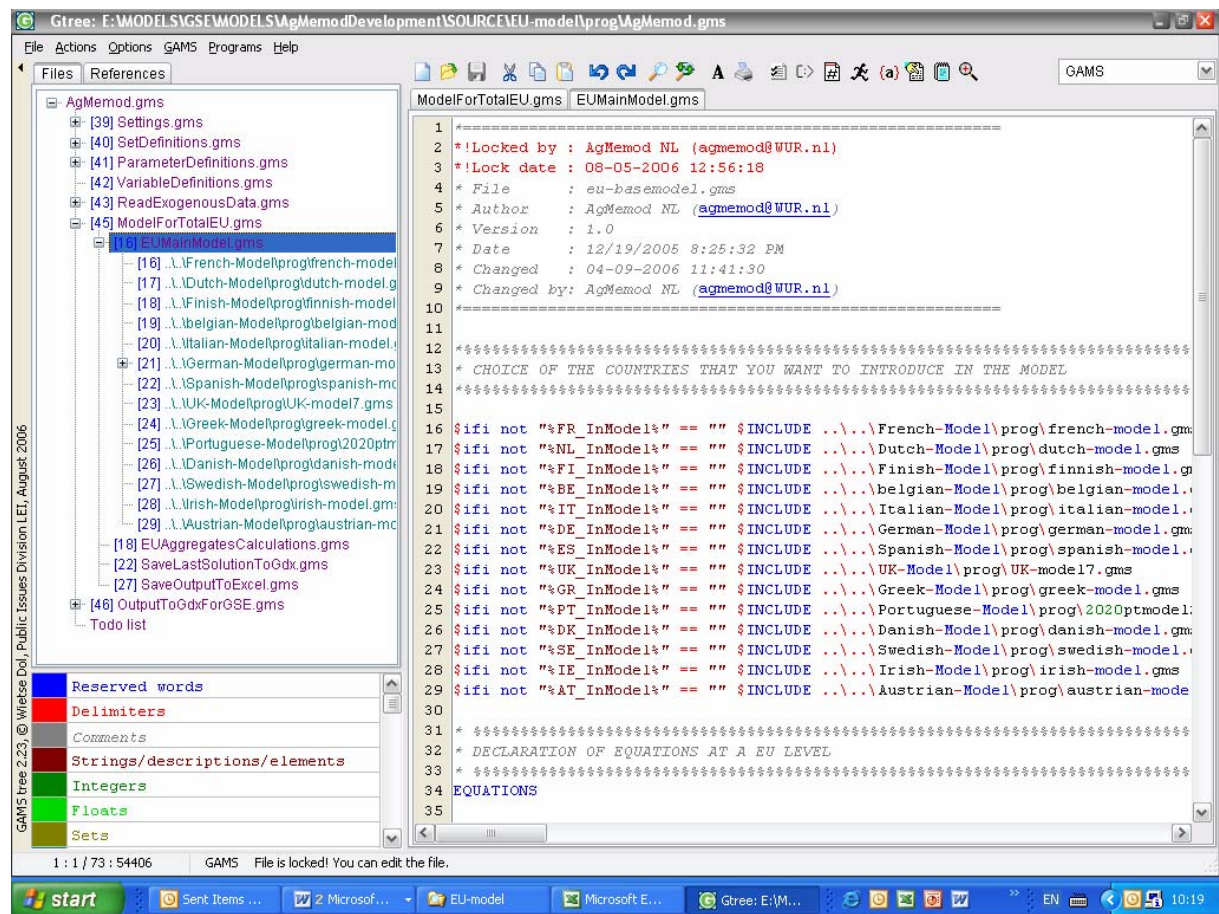
To make AGMEMOD more transparent and accessible, a restructure of the technical program code of the model was needed. A two-step procedure has been followed here (see Figure 8.1). First, the model was restructured using the *Gtree* tool, which stands for *GAMS tree* and can be considered an alternative to GAMS-IDE (Dol, 2006). The left-hand column in Figure 8.2 shows the breakdown of AGMEMOD into sub-files for settings, parameter and variable definitions, data reading, model calculations and output savings.

Figure 8.1 GSE concept in relation to AGMEMOD



Second, the *Gtree* version of the model was prepared to implement it in the user-friendly GSE tool. In practice, this will enable AGMEMOD users to run (several) scenarios, to save their outcomes, to examine scenario inputs and outputs and to examine the scenario outcomes in a GSE environment. An explanation of the toolbar and the various function buttons can be found in the GSE quick reference guide (Dol, 2006).

Figure 8.2 Structure of AGMEMOD in the GAMS tree



9. Documentation

- *Manual MetaWave*, Wietse Dol (2005), <http://www.nacquit.com/downloads/metawave.doc>
- *Manual Gtree*, Wietse Dol (2005), <http://www.nacquit.com/downloads/gtree.doc>
- *Manual GDX DataViewer*, Wietse Dol (2005), <http://www.nacquit.com/downloads/gdxviewer.doc>
- *Manual GSE*, Wietse Dol, Foppe Bouma and Barbara van der Hout (2005), <http://www.nacquit.com/downloads/gse.doc>
- *MicroWave Manual*, MicroWave consortium (2005), <http://www.nacquit.com/downloads/MicroWave.doc>
- *The MicroWave GAMS model tree*, MicroWave consortium (2005), <http://www.nacquit.com/downloads/MicroWaveGAMS.doc>
- *MicroWave: a generic framework for microsimulation-based ex ante policy evaluation*, Jacques Wolfert, Jan Lepoutre, Wietse Dol, Steven Van Passel, Hennie van der Veen and Foppe Bouma (2005), <http://www.nacquit.com/downloads/MicroWaveParma.pdf>

You can also download demo versions of all the software from the Nacquit website:

<http://www.nacquit.com/downloads/gsesetup.exe> (includes Gtree and the GDX DataViewer)

<http://www.nacquit.com/downloads/gtreesetup.exe> (includes the GDX DataViewer)

<http://www.nacquit.com/downloads/metawavesetup.exe>

For problems and questions you can always send an e-mail to: info@nacquit.com

10. References

Agricultural Economics Research Institute (2005). "Impact analysis of CAP reform on the main agricultural commodities." Report 1a on Intended Research Techniques. First deliverable of contract No 150267-2005-FIED-NL.

Agricultural Economics Research Institute (2006). "Impact analysis of CAP reform on the main agricultural commodities." Report 1b on Developed Research Techniques. Second deliverable of contract No 150267-2005-FIED-NL.

Agricultural Economics Research Institute (2006). "Impact analysis of CAP reform on the main agricultural commodities." Report 2 on Assumptions and Baseline Results of Country and EU 25/27 Aggregate Models. Third deliverable of contract No 150267-2005-FIED-NL.

Agricultural Economics Research Institute (2006). "Impact analysis of CAP reform on the main agricultural commodities." Report 3 on Scenarios to be Analysed. Fourth deliverable of contract No 150267-2005-FIED-NL.

Agricultural Economics Research Institute (2006). "Impact analysis of CAP reform on the main agricultural commodities." Report 4 on Baseline and Scenario results of country and EU 25/27 aggregate models. Fifth deliverable of contract No 150267-2005-FIED-NL.

Agricultural Economics Research Institute (2007). "Impact analysis of CAP reform on the main agricultural commodities." Final Report. Sixth deliverable of contract No 150267-2005-FIED-NL.

11. Appendices

Appendix 1: the Transport model with GSE tags

\$Title A Transportation Problem (TRANSPORT,SEQ=1)
\$Ontext

This problem finds the lowest cost shipping schedule that meets the requirements of the markets and supplies at factories.

Dantzig, G. B., Chapter 3.3. In Linear Programming and Extensions. Princeton University Press, Princeton, New Jersey, 1963.

This formulation is described in detail in:
Rosenthal, R. E., Chapter 2: A GAMS Tutorial. In GAMS: A User's Guide. The Scientific Press, Redwood City, California, 1988.

The line numbers will not match those in the book because of these comments.

\$Offtext

```
* <%SET%>
Sets
  i  canning plants / seattle, san-diego /
  j  markets       / new-york, chicago, topeka / ;
* <%SET%>
* <%INPUT%>
Parameters

  a(i) capacity of plant i in cases
      / seattle  350
      san-diego 600 /
* <%INPUT%>
* <%INPUT%>
  b(j) demand at market j in cases
      / new-york  325
      chicago   300
      topeka    275 / ;
* <%INPUT%>
* <%INPUT%>
Table d(i,j) distance in thousands of miles
      new-york  chicago  topeka
seattle    2.5    1.7    1.8
san-diego  2.5    1.8    1.4 ;
* <%INPUT%>
* <%INPUT%>
Scalar f freight in dollars per case per thousand miles /90/ ;
* <%INPUT%>
Parameter c(i,j) transport cost in thousands of dollars per case ;

      c(i,j) = f * d(i,j) / 1000 ;

Variables
  x(i,j) shipment quantities in cases
  z      total transportation costs in thousands of dollars ;

Positive Variable x ;

Equations
  cost    define objective function
  supply(i) observe supply limit at plant i
  demand(j) satisfy demand at market j ;

cost ..   z =e= sum((i,j), c(i,j)*x(i,j)) ;
supply(i) .. sum(j, x(i,j)) =l= a(i) ;
demand(j) .. sum(i, x(i,j)) =g= b(j) ;

Model transport /all/ ;
```


Solve transport using lp minimising z ;

Display x.l, x.m ;

```
* <%GDXOUTPUT%>
*****
*** The GDX OUTPUT statement ***
*****
execute_UNLOAD 'results.gdx',z,x;

* <%OUTPUT FILE results.gdx%>
* <%SYMBOL%>
* <%VARIABLE z total transportation costs in thousands of dollars%>
* <%FORMAT #,###,##0.###%>
* <%DIMENSION US-Dollar%>
* <%ENDSYMBOL%>
* <%SYMBOL%>
* <%VARIABLE x(i,j) shipment quantities in cases%>
* <%FORMAT #,###,##0.###%>
* <%DIMENSION Cases%>
* <%INFO%>
* The shipment of quantities from plant to city (sum of shipments is
* total supply).
* <%ENDINFO%>
* <%ENDSYMBOL%>
* <%/GDXOUTPUT%>
```

Appendix 2: GDXOUTPUT versus OUTPUT tags

In the code above it is possible to change the GDXOUTPUT part:

```
* <%GDXOUTPUT%>
*****
*** The GDX OUTPUT statement ***
*****
execute_UNLOAD 'results.gdx',z,x;

* <%OUTPUT FILE results.gdx%>
* <%SYMBOL%>
* <%VARIABLE z total transportation costs in thousands of dollars%>
* <%FORMAT #,###,##0.###%>
* <%DIMENSION US-Dollar%>
* <%ENDSYMBOL%>
* <%SYMBOL%>
* <%VARIABLE x(i,j) shipment quantities in cases%>
* <%FORMAT #,###,##0.###%>
* <%DIMENSION Cases%>
* <%INFO%>
* The shipment of quantities from plant to city (sum of shipments is
* total supply).
* <%ENDINFO%>
* <%ENDSYMBOL%>
* <%/GDXOUTPUT%>
```

with two EXPLICIT OUTPUT and two OUTPUT statements plus some GAMS loopings to print variables x and z to a file. Select the OUTPUT tag in the TagEditor and add the data as shown in

Figure 11.1. You have to insert two OUTPUT tags: the first for variable x and the second for z.

```

*****
*** The GSE OUTPUT statement ***
*****
* <%EXPLICIT OUTPUT FILE%>
* <%OUTPUT FILE LABEL shipment quantities in cases%>
* <%FILE x.out%>
* <%/EXPLICIT OUTPUT FILE%>

FILE expl1/x.out/;
put expl1 'x(i,j) shipment quantities in cases/';
put expl1 '/';
loop (i, loop (j,
  put "level   ." put i.tl; put '!'; put j.tl; put ' '; PUT x.l(i,j):10:3; put /;
  put "lo     ." put i.tl; put '!'; put j.tl; put ' '; PUT x.lo(i,j):10:3; put /;
  put "up     ." put i.tl; put '!'; put j.tl; put ' '; PUT x.up(i,j):10:3; put /;
  put "marginal ." put i.tl; put '!'; put j.tl; put ' '; PUT x.m(i,j):10:3; put /;
  ));
put expl1'/';
putclose expl1;

* <%OUTPUT%>
* <%OUTPUT FILE expl1%>
* <%VARIABLE x(i,j) shipment quantities in cases%>
* <%FORMAT #,###,##0,###%>
* <%DIMENSION Units%>
* <%/OUTPUT%>

*****
*** The GSE OUTPUT statement ***
*****
* <%EXPLICIT OUTPUT FILE%>
* <%OUTPUT FILE LABEL total transportation costs in thousands of dollars%>
* <%FILE z.out%>
* <%/EXPLICIT OUTPUT FILE%>

FILE expl2/z.out/;
put expl2 'z total transportation costs in thousands of dollars/';
put expl2 '/';
put "level   ";PUT z.l:10:3; put /;
put "lo     ";PUT z.lo:10:3; put /;
put "up     ";PUT z.up:10:3; put /;
put "marginal ";PUT z.m:10:3; put /;
put expl2'/';
putclose expl2;

* <%OUTPUT%>
* <%OUTPUT FILE expl2%>
* <%VARIABLE z total transportation costs in thousands of dollars%>
* <%FORMAT #,###,##0,###%>
* <%DIMENSION US-Dollar%>
* <%/OUTPUT%>

```

As you can see, the GDXOUTPUT tag is much easier to implement, but is only available for GAMS versions 20.6 and higher. With GDXOUTPUT you can write multiple outputs (variables, scalars and parameters) to one file, whereas with the OUTPUT tag you need to create one file for every output parameter/variable separately. Since GDX is binary output, the output file is also much smaller with GDX than with the OUTPUT tag.

Figure 11.1 Inserting OUTPUT tags

The screenshot shows a dialog box titled "GSE Tags" with a dropdown menu set to "OUTPUT". Below the dropdown are "Insert" and "Close" buttons. The dialog is divided into several sections:

- OUTPUT FILE name:** A text field containing "COSTS.OUT".
- GAMS PUT FILE name:** An empty text field.
- GAMS name:** Radio buttons for "PARAMETER" (selected) and "VARIABLE". Below is a text field containing "c(i,j)".
- Description:** A text field containing "transport cost in thousands of dollars per case".
- FORMAT:** A text field containing "#,##0.###".
- DIMENSION:** A text field containing "1000 US-Dollar per case".
- USERLEVEL:** A text field with a "hide" checkbox to its right.
- BOUNDS:** An empty text area.
- INFO:** A text area containing the text: "The Total transportation costs equals average transportation costs (f) multiplied by the average distance (d)."

■ AGMEMOD Partnership

- *Agricultural Economics Research Institute (LEI), The Hague, The Netherlands:* Hans van Meijl, Myrna van Leeuwen, Andrzej Tabeau
- *Bundesforschungsanstalt für Landwirtschaft (FAL), Braunschweig, Germany:* Petra Salamon, Oliver von Ledebur
- *Centre of Agricultural Economics, INRA-ESR, Rennes, France:* Frédéric Chantreuil, Fabrice Levert
- *Teagasc-Rural Economy Research Centre (RERC), Athenry, Co. Galway, Ireland:* Trevor Donnellan, Kevin Hanrahan
- *Latvian State Institute of Agrarian Economics (LSIAE), Riga, Latvia:* Danute Jasjko, Guna Salputra, Ludmilla Fadejeva
- *University of Ljubljana, Biotechnical Faculty (LJUB), Ljubljana, Slovenia:* Emil Erjavec, Stane Kavcic, Darja Regoršek
- *Universität für Bodenkultur Wien (BOKU), Wien, Austria:* Martin Kniepert
- *Université Catholique de Louvain (UCL), Louvain-La-Neuve, Belgium:* Bruno Henry de Frahan, Olivier Harmignie
- *Institute of Agriculture Economics (IEABG), Sofia, Bulgaria:* Nedka Ivanova, Mariya Peneva
- *Research Institute of Agriculture Economics (VUZE), Prague, Czech Republic:* Ivan Foltyn, Jan Kubát
- *Food and Resource Economic Institute (FØI), Frederiksberg C, Denmark:* Jorgen Dejgaard Jensen
- *Institute of Economics and Social Sciences of Estonian Agricultural University (EAU), Tartu, Estonia:* Mati Sepp
- *MTT Agrifood Research Finland (MTT), Helsinki, Finland:* Jyrki Niemi, Lauri Kettunen
- *Department of Economics, University of Athens (NKUA), Athens, Greece:* Elias Mantzouneas
- *Corvinus University of Budapest (CUB), Budapest, Hungary:* Tibor Ferenczi
- *Polytechnic University of Marche-Ancona (UNIVPM), Ancona, Italy:* Roberto Esposti, Antonello Lobianco
- *Lithuanian Institute of Agrarian Economics (LAEI), Vilnius, Lithuania:* Irena Krisciukaitiene, Salomeja Andrekiene, Andrej Jedik, Willi Meyers, Aiste Galnaityte
- *Warsaw School of Economics (WSE), Warsaw, Poland:* Sylwia Krawczyńska, Katarzyna Kowalska
- *Institute of Agricultural Economics (IEARO), Bucharest, Romania:* Camelia Serbanescu, Cristian Kevorchian
- *Slovak Agricultural University (SAU), Nitra, Slovak Republic:* Lubica Bartova, Pavel Ciaian, Jan Pokrivcak
- *Unidad de Economía Agraria, Centro de Investigación y Tecnología Agroalimentaria de Aragón, (CITA), Zaragoza, Spain:* Azucena Gracia
- *Queen's University of Belfast (QUB), Belfast, UK:* Zi Ping Wu, Philip Kostov

European Commission

EUR EUR 22940 EN/4 – Joint Research Centre, Institute for Prospective Technological Studies

Title: Impact Analysis of CAP Reform on the Main Agricultural Commodities. Report IV AGMEMOD - AGMEMOD – GSE Interface Manual

Author: Wietse Dol.

Editors: Robert M'barek and Lubica Bartova

Luxembourg: Office for Official Publications of the European Communities
2008

EUR - Scientific and Technical Research series; ISSN 1018-5593

Abstract

This report is based on a study carried out by the AGMEMOD Partnership under the management of the Agricultural Economics Research Institute (LEI, in the Netherlands), in cooperation with the Joint Research Centre – Institute for Prospective Technological Studies (JRC-IPTS) to generate projections for the main agricultural commodity markets for each year from 2005 until 2015.

The report gives an overview of the GSE interface and how GSE is applied with the AGMEMOD model, directly addressing the needs of model-builders and users in an application-oriented and practical way.

Detailed documentation on the AGMEMOD modelling approach, along with the outcome of the study, is published in five reports in the JRC-IPTS Scientific and Technical Report Series under the heading "Impact analysis of Common Agricultural Policy reform on the main agricultural commodities".

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

