Environmental Economics and Natural Resources Group
Hollandseweg 1
6706 KN Wageningen, the Netherlands

READER

GAMS for environmental-economic modelling

- version April 2009 -

Rob Dellink

with contributions by: Juan-Carlos Altamirano-Cabrera, Kelly de Bruin, Ekko van Ierland, Carol Phua, Arjan Ruijs, Erik Schmieman, Judit Szõnyi, Frank Vöhringer and Xueqin Zhu

April 2009

WAGENINGEN UNIVERSITY
SOCIAL SCIENCES

WAGENINGENUR

CONTENTS

EXERCISES

Preface to this version

This is the 2006-version of the GAMS-reader that is available for self-study. Over the last years, this GAMS reader has developed from an internal document just for students at Wageningen University full of bugs, typos and unclarities, to a balanced self-study guide for all kinds of users in Wageningen and around the world. The associated website (go to http://www.enr.wur.nl/uk/ and click on ‘GAMS for environmental economic modelling’) has proven it’s *raison-d’être* and gets a stable 200 – 300 hits per month, and over 50 percent of those come from outside Europe.

The differences with the previous version (edition March 2004) are limited to editorial changes. The basic concept of the reader is (still) that users do not need a background in economics or environmental issues to be able to learn GAMS. Some basic knowledge of economics and environmental economics is useful for the interpretation of the model results, but this is not essential.

Acknowledgements

Many thanks to all colleagues at the Environmental Economics Group of Wageningen University for numerous comments. Also thanks to all MSc students of the “Scenario Studies and the Environment” and “Theories and Models in Environmental Economics” courses, the PhDs that participated in the SENSE course “Introduction to environmental economics” or in the CEEPA Training of trainers workshop, and all other users around the world, who had to cope with an early, unpolished version of the reader. Thanks also to all course coordinators who use this reader in their classes for their interest in our material. Obviously, all responsibilities for errors remain with the authors.

Instructor’s Library

There is an Instructor’s Library available for lecturers who want to use this reader in their teaching. The Instructor’s Library contains the GAMS codes to all exercises. Interested lecturers can contact Rob Dellink to obtain the Instructor’s Library. The Instructor’s Manual as released with previous versions of the reader is no longer maintained.

For more information, see the “GAMS for environmental economic modelling” website on the Internet homepage of the Environmental Economics Group at Wageningen University. Go to http://www.enr.wur.nl/uk/ and click ‘GAMS for environmental economic modelling’.

1. INTRODUCTION

1.1. Introduction

GAMS is a software package used to solve systems of equations. GAMS stands for General Algebraic Modelling System and is constructed by the GAMS Development Corporation. GAMS contains different solvers for different purposes.

Various kinds of economic models can be written down as a system of equations, including systems analysis, non-linear optimisation and equilibrium modelling.

GAMS is widely used across the world among economists.

This reader is written for students of Environmental Economics at Wageningen University. The reader will be used in the course Theories and Models in Environmental Economics (ENR-30306) and for students who want to learn GAMS for their thesis. In principle, the reader is structured such that anyone with access to GAMS can learn how to build models in GAMS, but all examples are chosen within the field of environmental economics.

The reader is structured as follows. Section 2. gives an introduction in the GAMS language and the interface GAMS-IDE that accompanies GAMS. Section 3. presents exercises to learn GAMS. Step by step, the user will learn all major features of GAMS, starting with very simple exercises that experienced Windows-users can quickly go through, via intermediate level exercises that present the building blocks for more advanced models, to the advanced level, where whole GAMS models are constructed using a general economic description. Section 4. presents some exercises that deal with certain model types, like systems modelling. These exercises are specific for the course Theories and Models in Environmental Economics, but other readers may find these exercises a good way to test their understanding of GAMS.

The reader does not provide the answers (model codes) to the exercises. Rather, the main outcomes of the models are represented in Appendix II. You can find some remarks on common error messages in Appendix 1.

1.2. GAMS on the Internet

The homepage of the GAMS corporation (www.gams.com) contains a lot of useful information. From the homepage, a full user guide can be downloaded at www.gams.com/docs/document.htm; the user guide contains the syntax for all GAMS commands and very helpful as a reference when writing GAMS models. Note that the user guide is also available via the Help function in GAMS-IDE.

The introductory chapter to the GAMS User Guide, written by Rick Rosenthal, gives a good overview of how GAMS works. All readers are advised to study this tutorial when starting to learn the GAMS software.

On the homepage of the Environmental Economics Group of Wageningen University, there is a special page with GAMS Tips and Tricks. You can browse through this list of useful hints to increase your programming skills and improve your own models. Don’t worry if you cannot fully understand all hints; some are at a more advanced level or are highly specific.

1.3. GAMS at your home computer

There is a free version of GAMS available for installation on your own computer. This is a limited version of GAMS, which cannot solve large problems, but it may be handy when you are building proto-type models. That is, of course, if you have a PC at hand.

A free copy of the restricted, student version is available for download at http://www.gams.com/download.

This version restricts the size of models that can be run in several ways, though most of the models in this reader can be solved using the demo version.

See the information on the download page for more details on how to get this student version of GAMS. Alternatively, you can borrow the cd-rom from Rob Dellink.

1.4. The "GAMS for environmental economic modelling" website

You can download all the GAMS related materials from the internet site of the Environmental Economics Group, Wageningen: http://www.enr.wur.nl/uk/, click 'GAMS for environmental economic modelling'.

The website contains a free download of this reader, the necessary download files for the exercises and several useful links.

2. GAMS FOR ENVIRONMENTAL-ECONOMIC MODELLING

2.1. For what type of problems can GAMS be used?

GAMS has its origins in economic modelling, but this does not mean that the models you specify have to be in the field of economics. As you can see the subject index of the GAMS model library - http://www.gams.com/modlib/modlib.htm - GAMS can provide a support optimisation program for the several fields. This tutorial mostly covers the field of environmental economics, but GAMS can also be used to analyse for example chess questions (like the maximum number of queens in the game).

Specifying an economic model means that we have to write down one or more equations with some economic relationship. This relationship can be between labour demand and wages, between demand and supply of a good, *et cetera*. The geographical scope of the model can range from writing an economic model for an individual firm to a model that tries to describe the global economy. Many models take the scope of a national economy.

Environmental issues can be included in the GAMS models in many different ways. In environmental-economic models, mostly you write down a ‘standard’ economic model and then add equations for emissions, concentrations, abatement, economic damages from pollution, *et cetera*. But nothing prevents you from writing an environmental model without any economics in it. However, this reader concentrates on environmental economics, so all models we will discuss contain equations describing the economy and equations for environmental issues.

2.2. Working with GAMS-IDE

Most users of GAMS can run the system in the Integrated Development Environment (IDE)[1].

When GAMS-IDE is started, a window will appear with a menu bar along the top and a main Edit Window for GAMS applications. As with most such systems, input and output operations are controlled by the **File** pull down menu, with other menu items used in edit operations, and in running the GAMS system.

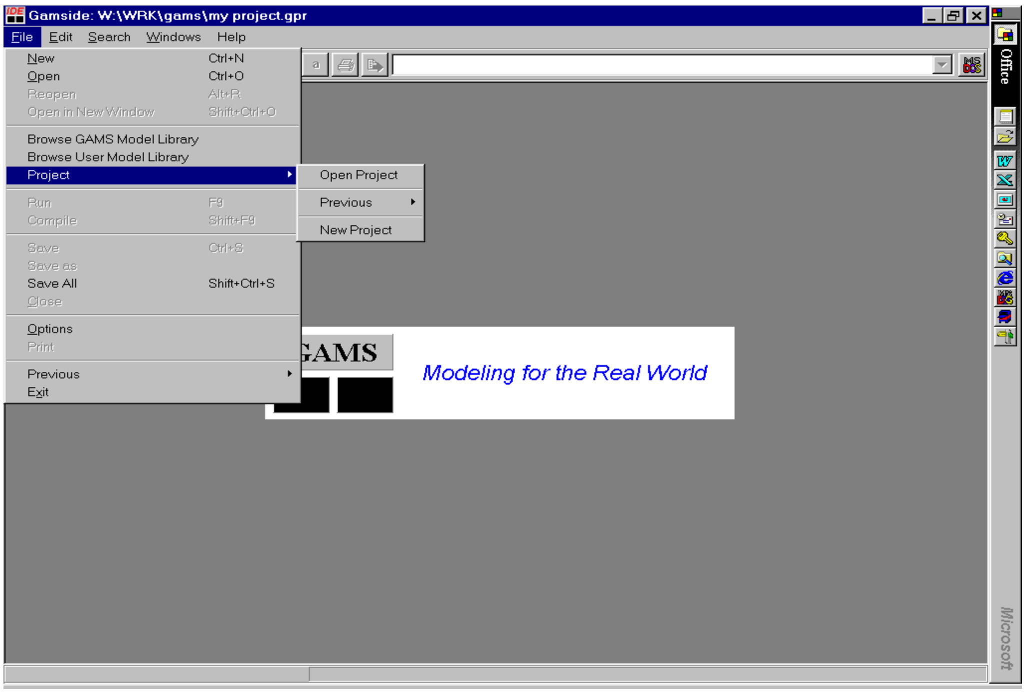
Users should begin each session by selecting a "project". A project is a system file you save but never have to touch. Still, its location is important because the folder (directory) of the current project file is where (`.gms`) input and (`.lst`) output files are saved by default. This allows you to easily keep all the input and output files for any task together in the same directory, and use different directories for different projects. The starting project file (if any) is shown at the top of the main GAMS window; in the picture below, the starting project file is “W:\WRK\GAMS\my project.gpr”. To select another, or create a new one, use the **Project** item on the **File** menu.

The IDE version provides for standard, mouse-driven editing of input files in the main GAMS Edit Window. If the appropriate file is not already displayed, use the **New** or **Open** commands on the **File** menu to activate one. Then create or correct the file with the mouse and tools provided on the **Edit** and **Search** menus. The **Matching Parenthesis** button helps

[1] This section is based on GAMS-IDE version 19.3, as released in May 2000, but also applies to newer versions.

with the many parentheses in GAMS by skipping the cursor to the parenthesis that corresponds to the one it is now positioned in front of. The **Find in file** is also a useful tool, if you work with a complex model.

The GAMS-IDE without any open files looks as follows:



Once a `.gms` file is ready to run, the **Run** item on the main menu bar invokes GAMS. In addition, it automatically causes a `.lst` output to be stored in the current project directory (but not displayed).

The `.lst` output file can be activated using the **Open** command on the **File** menu. However, it is usually easier to first survey an IDE run by examining the separate Process Window, which is automatically displayed. A brief log of the run appears there, and clicking on any of the boldface lines (including run error messages) will activate the entire `.lst` output file and position you on that message. In particular, clicking on **Reading solution for model** will open the `.lst` and position the window at the `SOLVE SUMMARY`.

Syntax errors in GAMS input show in red in the Process Window. Clicking on any such red error message brings up the corresponding `.gms` file in the main GAMS window and positions the cursor at the point where the error was detected.

The use of the IDE is part of the Introductory exercises below. You can learn more about the software if you take a guided tour in the **Help** menu.

In version 19.3 of the IDE and later, different parts of GAMS are presented in different colours. For example, all text that is written as a comment appears in grey, keywords are in blue and set elements are in green (at least in the lines where they are defined).

2.3. The general structure of GAMS programs

The first step in modelling in GAMS is to write an input file. Though it is not strictly necessary, normally a GAMS input file has a file-extension `.gms`. You write the input file, run the model in GAMS and look at the output file for the results (the output file has an extension `.lst`).

The general structure of a simple GAMS input file contains the following elements:

PARAMETERS
: {gives the data or exogenous constants of the model; these values are fixed}

VARIABLES
: {indicates the variables that will be determined (calculated) within the model}

EQUATIONS
: {first, the equations have to be declared, then they are defined}

MODEL
: {the model is given a name}

SOLVE
: {the solution mode is specified, as well as a declaration whether the optimand should be maximised or minimised}

Each of these elements can exist more than once in a single GAMS model.

The restrictions and special meanings of these words are all together called the syntax of a model. GAMS is a computer package and will only understand what you want if you write your model in the correct syntax.

The basic portions of GAMS code are now discussed in more detail.

PARAMETERS

The first step in writing a GAMS model is to provide the constant elements (also called ‘exogenous coefficients’). These are data that are not determined within the model, but they have a fixed value that you have to provide.

There are two stages in specifying parameters, first you must declare them by making a list of all parameters and closing the list with a semi-colon, and then you define the values and close each definition with a semicolon. (You can learn more about the use of semicolon at the end of this chapter.)

Suppose you have a parameter `A` and want to give it a value of `3`. Then, the GAMS code is

```
PARAMETERS
      A     Explanatory text on the meaning of A;
      A = 3;
```

The first line is the ‘code-word’, telling GAMS which part of the model will follow. In this case, the ‘code-word’ is `PARAMETERS`. First give the name of the parameter, then you may write some explanatory text on the meaning (though this is not necessary). In a new section you define the value of the parameter.

Note that there is an alternative way of assigning values to parameters (and scalars). In the declaration, you can directly specify the value between slashes (see Exercise 3.2.3 for more details):

```
PARAMETERS
      A     Explanatory text on the meaning of A      /3/;
```

Constant elements can be specified in several ways: as single parameters called scalars, (if applicable) as vector parameters or as tables (these will be introduced later). GAMS is indifferent to the way the constant elements are specified, but you'll find that all types have their advantages. The following rules of thumb apply. First, tables are the most compact, so if you can use tables, do so. But only put data together into one table that have some cohesion with each other. Second, use parameters for constants with two or more dimensions (the dimensions are given by the indices used) and, finally, scalars are used for single values that do not change (the constants in the narrow sense).

Notice that you don't write a semi-colon after each line, but only after the last line of the block with declarations. You can write a block of variables in the same way (see below).

VARIABLES

The variables are what you are really interested in as a modeller. These are the things that are determined endogenously within the model, and the value of which you cannot calculate beforehand (well, unless you write a very, very simple model). The values of the variables are determined by solving the equations. However, you first have to tell GAMS what the names of the variables are. In this declaration of the variables, you can also provide an explanatory text to the variable, to help you understand what the meaning of the variable is. The GAMS code is:

```
VARIABLES
      X1    Explanatory text on the meaning of X1
      X2    Explanatory text on the meaning of X2
      Y     Explanatory text on the meaning of Y;
```

So just as with the parameters, you first write the 'code-word', then provide the variables line-by-line. You can provide the explanatory text, but this is not necessary; the values of the variables are calculated by the model therefore you do not have to define values here.

Note that if you want to build more complex models, it becomes important to choose good names for your parameters and variables. Using the words "supply" and "demand" instead of just "s" and "d" helps you in later stages to read the model code. Try to keep the names and structure of the code as logical as possible.

EQUATIONS declaration

The treatment of the equations is a little more complicated in GAMS. The thing to remember is that you have to take two steps: first, you declare the equations, and, second, you write the equation itself in the equation definition section. The declaration of the equations is straightforward. You can use any name you want to declare the equations. Most people have some standard way of naming the equations. For example, a useful way of naming the equations is to take the variable that is determined by the equation and put a 'Q' in front of the variable name.

You may find it easier to you write down the equations themselves first, and then just above the equations you write the declarations.

So, in GAMS code, you could have

```
EQUATIONS
      QX1   Explanatory text on the meaning of the equation for X1
      QX2   Explanatory text on the meaning of the equation for X2
      QY    Explanatory text on the meaning of the equation for Y;
```

Equations definition

The core of any model is given by the equations that have to be solved. In GAMS, you can write the equations fairly straightforward. You write them one by one in the following way:

```
QX1.. X1 =L= A;
QX2.. X2 =E= 5;
QY..  Y =E= X1 + X2;
```

In the example above, the first equation determines the value of `X1`. The equation is named `QX1`, and states that the value of `X1` should be less than `A`. The second equation tells us that `X2` should equal `5`. In the third equation, the value of `Y` is determined as the sum of `X1` and `X2`.

You can use three types of equations:

- the left-hand-side should be less than or equal to (`=L=`),
- greater than or equal to (`=G=`) or
- equal to (`=E=`) the right-hand-side.

This is a very simple model that you could calculate by hand. But the structure of the equations is very general, so writing much more complex model will not lead to much more complex GAMS code. For example, you can do multiplication, raise a variable or scalar to some power, *et cetera*. The more complex issues will be dealt with in the exercises in Sections 3.2. to 3.3.

The MODEL statement

The MODEL statement is quite simple: you think of a name to give to the model, for example `TEST`. Any other name can be used, as long as it’s not too long and does not have ‘special’ characters. You have to tell GAMS which equations are part of the model `TEST`. Normally, you want to include all equations, and then the MODEL statement looks like this:

```
MODEL TEST /ALL/;
```

`ALL` refers to that you use all the equations, you can also specify a submodel here by listing all equations you want to include in the model. (Separate the equation names with commas).

The SOLVE statement

The SOLVE statement is to tell GAMS to solve the model. You provide the ‘code-word’ `SOLVE`, the model name, the solution mode, the optimand and whether to maximise or minimise.

For example, if we want to solve model `TEST` by maximising `Y`, using `DNLP` as the solution mode, we write:

```
SOLVE TEST USING DNLP MAXIMIZING Y;
```

The solution mode relates to what type of model you have specified: if your model is linear, use linear programming (`LP`); if it is non-linear, use non-linear programming (`NLP` or `DNLP`). There are more model types, but we will not discuss them in this reader. Throughout this reader, we will use `DNLP` as the solution mode. This solution mode is the most general and works for NLP and LP models as well.

The optimand is the variable that should be maximised or minimised. GAMS will try and find that solution to the model where the value of `Y (the optimand)` is as high (`MAXIMIZE`) or low (`MINIMIZE`) as possible. See Appendix I. for more details.

Use of the semi-colon

When you run the input file, GAMS will read the file you wrote line by line. To tell GAMS that the end of a line has arrived, use a semi-colon (";"). The semi-colon is used to tell GAMS that the end of a command has been reached. In principle, you should end all lines with a semi-colon, except when you declare a list of parameters, variables or equations. Then, this list is regarded as a single block and you should end the block with a semi-colon. For example, if you want to include a second parameter `B`, you could write:

```
PARAMETERS
A     Explanatory text on the meaning of A;
      A = 3;

PARAMETERS
B     Explanatory text on the meaning of B;
      B = 5;
```

But it is more convenient to write it as a list of scalars and use a semi-colon only at the end of the list:

```
PARAMETERS
A     Explanatory text on the meaning of A
B     Explanatory text on the meaning of B;
      A = 3;
      B = 5;
```

The definitions of the equations cannot be treated as a block, so you should write a semi-colon after each equation definition.

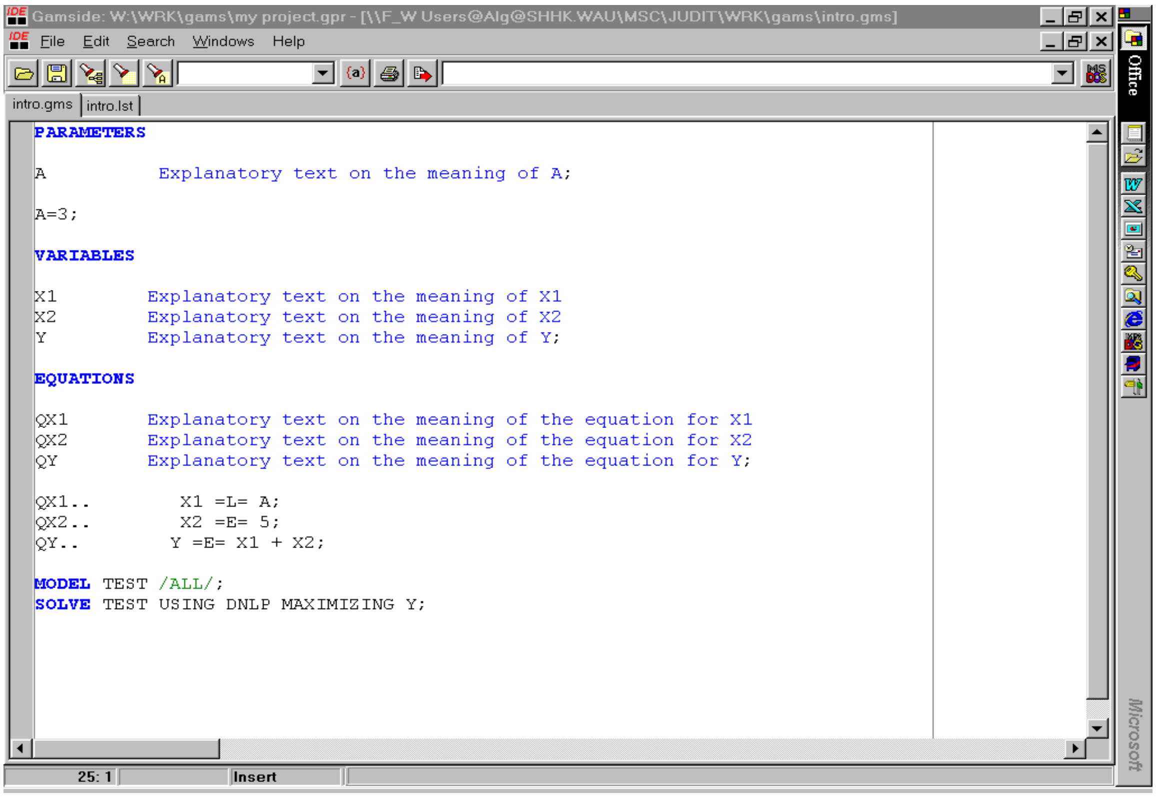
The correct use of the semi-colon will become rapidly clear to you when you start writing your own models, as GAMS will come with an error message if you made a mistake. Still, always be careful in the syntax of your models.

Brief summary:

USE SEMI-COLON	DO NOT USE SEMI-COLON
- after the end of each declaration block (parameters, variables, equations, etc.)	- after each line within a block (for blocks of parameters, variables, etc.)
- after each defined parameter	- after each declared scalar and parameter, only at the end of the block
- after each defined table	- after each declared variable
- after each defined equation	- after each declared equation

The complete code

So, now we have specified a complete model in GAMS code. The full model looks as follows:


Gamside: W:\WRK\gams\my project.gpr - [\\F_W Users@Alg@SHHK.WAU\MSC\JUDIT\WRK\gams\intro.gms]
File
Edit
Search
Windows
Help
intro.gms
intro.lst
25: 1
Insert
Office
Microsoft

```
PARAMETERS

A          Explanatory text on the meaning of A;

A=3;

VARIABLES

X1         Explanatory text on the meaning of X1
X2         Explanatory text on the meaning of X2
Y          Explanatory text on the meaning of Y;

EQUATIONS

QX1        Explanatory text on the meaning of the equation for X1
QX2        Explanatory text on the meaning of the equation for X2
QY         Explanatory text on the meaning of the equation for Y;

QX1..         X1 =L= A;
QX2..         X2 =E= 5;
QY..         Y =E= X1 + X2;

MODEL TEST /ALL/;
SOLVE TEST USING DNLP MAXIMIZING Y;
```

Note that the order in which the portions of GAMS code can be specified is quite flexible. The principle that has to be obeyed is that all elements have to be declared before you can use them; so, for instance, an equation declaration must precede the equation specification.

True, this model is not very exciting and you will not be amazed that GAMS can actually compute that the optimal value of Y is 8. But this is just the general structure. Using the same syntax, you can specify much more interesting models and solve difficult systems of equations that you cannot calculate by hand.

2.4. Scenarios and sensitivity analysis

Most model simulations do not stop after one solve of the model. Rather, the first solve is used as a reference scenario, that represents the current situation (or, in a dynamic model, the baseline projection represents the most likely development of the variables over time). Then, a so-called counter-factual analysis is done: some parameter values in the model or equation specifications are changed, the changed model is run and the new results are compared to the reference results. This can all be done within one GAMS model file.

The two major types of counter-factual analyses are scenario analysis (sometimes called uncertainty analysis) and sensitivity analysis[2]. The basic difference between these types is that scenario analysis tries to answer questions of the type 'what happens if one or more elements (or equations) in the model change', while sensitivity analysis tries to answer 'what is the consequence of a misspecification of some parameter value'.

In a scenario analysis, several alternative model specifications are compared to each other. These scenarios may differ due to differences in parameter values, but also due to differences in the model equations. In principle, each of the scenarios specified may be equally viable (though they not always are). Often, three or four scenarios are calculated to show the extremes within which the real value will probably lie (*i.e.* the scenarios are used as 'corners of the playing field'). The scenarios specified may be used to do policy recommendations. For example, if we lower the tax rate on labour with 1%, total employment may go up with x%. Of course, these policy recommendations are only valid within the boundaries and assumptions that are made in the model.

A sensitivity analysis has another purpose: an (individual) parameter value is changed to analyse the effects of the value chosen on the model results. This gives a clue on the robustness of the model with respect to the specification of the model. For example, the emissions of phosphor from agriculture in the Netherlands may be estimated at 0.31 grams per guilder of agricultural production per year (1990 data, Statistics Netherlands), which results in total phosphorous emissions from agriculture of 132 million kilograms per year. To investigate how the total emissions will change if the emissions per guilder of production are 1% higher, one can do a sensitivity analysis. In this example, the relationship is linear and the result straightforward: total emissions will also be 1% higher (133.32 million kilograms). But imagine a more complex model where relationships are not all linear. For example, what is the effect of a slight misspecification of the phosphor content of animal feed on total deposition of phosphor in water? Then, the results of a sensitivity analysis cannot be predicted so easily, and you need to simulate the sensitivity analysis in the GAMS program.

2.5. General guidelines model analysis

Models can best be built in a step-by-step manner, or else you will lose track of which element causes the model to malfunction, and you will lose a lot of time debugging. Therefore, it is wise to use the following guidelines.

1. Identify the research questions $\rightarrow$ determine the objectives of the model $\rightarrow$ which mechanisms should be illustrated by the model?

[2] Theoretically, one could think of more types, but this section confines itself to the most common distinction and interpretation of these concepts.

2. How can this be translated into model terms:

   → When using an existing model:

   a. which model is selected as basis (and is this model completely understood)?
   b. in which way does the base model have to be adapted?
   c. in which way does the base model have to be extended?

   → When building a model from scratch:

   a. what are the basic building blocks of the model?
   b. how can these building blocks be specified exactly?

2. Step by step modelling:

   a. split all adaptations and extensions (or building blocks) into the smallest steps possible;

   b. determine for each tiny step which variables, parameters and equations have to be adapted or introduced (be precise);

   c. model this step;

   d. check whether the model solves without problems and that this step has lead to the correct results;

   e. if (and only if) the step is correctly executed and passes the check move to the next step (back to item b); if the model does not work properly, verify whether the model was working properly before this step was implemented; if not, move more steps back, until you have a properly working model; correct the step that created the problems, and move forward again, one tiny step at a time.

3. If necessary, adjust the research questions and model translation.

4. Gather the best available data and insert these in the model → again, use a step-by-step approach, so that you know which steps / pieces of data create problems (n.b. identify chunks of data that can be inserted separately).

5. If (and only if) all mechanisms and their model translation are inserted correctly into the model and the model is working properly with realistic data, the time has come to “dress up the model”, i.e. identify and implement model adaptations and extension that are not necessary for a correct illustration of the mechanisms, such as more details in certain model areas.

3. LEARNING EXERCISES

3.1. Introductory level

The first exercises with GAMS are intended to familiarise you with the basic structure of GAMS and the IDE interface. The goal is that you are able to write, run and read GAMS models and their output.

Exercise 3.1.1. Starting GAMS-IDE

The first exercise should be easy:

Start the GAMS interface program GAMS-IDE by clicking on the GAMS-IDE button (the icon says IDE in red and has two black squares below the name:).

If you cannot find the button or if the program does not start properly, then check if you have access to the disk from which GAMS is run (the Wageningen students should be logged into the network).

Go to the right directory and make a new (or open an existing) project.

If no projects have been created on this machine before, you'll see a window asking you to provide a project or name a new project. Go to your own directory[3] and make a new project (*e.g.* my project.gpr).

If GAMS-IDE opens with a different project with open files, close any files that are open by selecting "File / Close" from then menu, or by clicking the small **x** in the right-hand corner of the window containing the file. Warning: do not press the small **x** in the upper right-hand corner, as this will close GAMS-IDE and not just the file. Now you can make a new project (or open the existing project).

Actually, you can switch between projects without closing your files first. In this case next time you open your project these files are automatically opened.

The next step is to create a new file by selecting "File / New" from the menu. Save this empty file by selecting "File / Save" from the menu, or by pressing the Save button () on the toolbar. Give the file the name "intro".

The program will automatically add the extension ".gms". You could use other names, but for this practical it is useful that you stick to the names suggested here.

Exercise 3.1.2. The basic blocks of writing a model in the IDE

If necessary, open the file "intro.gms" in the project you chose in Exercise 3.1.1. Type the simple model as presented at the end of Section 2.3. (see page 13) Write all the basic blocks of code (parameters, variables, equations, model, solve). Carefully check for typing and other errors.

[3] Note: in some cases there have been problems with using a network drive as the project directory (especially when using Windows 2000). In those cases, create a directory on the hard disk.

Exercise 3.1.3. Reading the process window

Run this GAMS file by selecting “File / Run” from the menu, by pressing the F9-key or by clicking on the button on the toolbar (▶).

The .gms-file is automatically saved before running it.

A process window appears. This window displays the log of the model run: it tells you what GAMS is doing, it shows information on the iterations of the solver and in this window you can read whether GAMS has solved the model correctly. Please take a careful look at the completion status of the model: if the process window informs you (at the bottom) that the model has ‘normal completion’ this does not necessarily mean that an optimal solution was found. So for each solve, carefully check that an optimal solution was in fact found.

Depending on the settings of the project, the process window automatically moves to the last line written (it automatically scrolls down when GAMS moves on), or it stays at the top of the log. In the first case, you can manually scroll back up to look at earlier information, and in the latter case you can manually scroll down. (You can change this setting by selecting “File / Options” from the menu, then checking “Update process window” from the “execute” tab).

Exercise 3.1.4. Reading the listing file

When you run a GAMS file, a so-called listing file is automatically written by GAMS. So when you ran the file intro.gms, GAMS made the file intro.lst in the directory where the project is located (i.e. in the working directory).

Note: do not mix up the log file and the listing file. The log file is what GAMS-IDE shows during the run; the listing file is where GAMS stores the results of your model run.

The .lst output file can be activated using the Open command on the File menu. However, it is usually easier to first survey a run by examining the log file in the separate Process Window, which is automatically displayed. A brief log of the run appears there, and clicking on any of the black-coloured lines (including run error messages) will activate the entire .lst output file and position you on that message. In particular, clicking on Reading solution for model will open the .lst and position the window at the SOLVE SUMMARY. Clicking on a red-coloured line will cause the cursor to jump in the .gms file to the line with the error.

Open the listing file intro.lst.

Read the output file carefully by following each item of the discussion of the general structure of a GAMS output file below.

The general structure of a GAMS output file is[4]:

1. Echo print.

The first part of a list file gives a copy of the GAMS input file that has been run. For the sake of future reference (in case of errors) GAMS puts line numbers on the left hand side of the list file.

[4] Note that parts of the output file may look different if you use a different solver.

2. Error messages. {Only present if there are mistakes in the model!}

If you have made any errors then these are displayed with a dollar sign ($) followed by a number. The error number and an explanation of the error are given after the echo print. If any errors have been made, then these should be corrected before proceeding. Errors are the topic of the next exercises.

3. Reference maps.

The next section of the list file is a pair of reference maps that contain summaries and analyses of the input file for the purposes of debugging and documentation. The first reference map is a cross-reference map such as one finds in most modern compilers. It is an alphabetical, cross-referenced list of all the identifiers (e.g. scalars/parameters, variables, equations) of the model. The second reference map is a list of model identifiers grouped by type and listed with their associated documentary text.

4. Equation listing.

If no errors have been made, the list file will also contain an equation listing. The equation listing allows you to check whether GAMS has indeed generated the model from your input file that you intended.

5. Model Statistics.

This is the last section of output that GAMS produces before invoking the solver and consists of a group of statistics about the model's size.

6. Solve summary.

After the solver executes, GAMS prints out a brief status report. **It is important to <u>always</u> check the Solver Status and the Model Status to verify that GAMS has found the optimal solution**.

7. Solution reports.

This is the part that you are of course most interested in. The solution report of the variables gives the results of the optimisation problem that was solved in GAMS. The report presents lower, level, upper and marginal values, where lower and upper will give any lower and upper bounds that have been imposed. Level gives the optimal value for the variable, i.e. the solution. The value under marginal gives the dual value (shadow price) of the variable; this is the first derivative of the objective value to the level of the variable.

Read the solution report of the file intro. What are the optimal values, lower and upper bounds of the variables?

8. Report summary.

At the end of the solution report, a report summary is given. The desired report summary should be as follows :

```
**** REPORT SUMMARY :   0     NONOPT
                        0     INFEASIBLE
                        0     UNBOUNDED
                        0     ERRORS
```

If you have made any errors, then correct them now.

Exercise 3.1.5. Debugging a compilation error

If you have carefully carried out the exercises above, you have not seen any errors yet. So in this exercise, we will make one intentionally.

Remove the semi-colon after equation `QX1`. Run the model again and see what the process window tells you.

Double-click the red line with the first error in the process window. You automatically jump to the file intro.gms and are near the point where the error is (in this case, you'll jump to the line below the error). Double-click on the error message in the process window, just below the red line. The listing file appears and the error code is shown. More below in the listing file, you'll find an explanation of the error codes, so you can check what type of error you've made.

Fix the error and save the correct file.

(If only you were always so lucky to know what the error is straight away.)

Trying to fix any errors you get when building a model is called *debugging*. This debugging is often the hardest and most time consuming part of the whole process of model building. Through learning-by-doing you'll get better at debugging as you get more experience. So don't feel bad if it takes you forever to fix a simple error when you've just started specifying GAMS model.

Exercise 3.1.6. Debugging a logical error

A second type of errors commonly made can be labelled as 'logical errors'. These errors are not a conflict with the GAMS syntax, but a misformulation of the model itself. These logical errors are sometimes hard to find, especially if you do not fully understand the logic behind the model. And sometimes you do not even get an error, but you get just the wrong results. Therefore, always think hard about the specification of the model and make sure the specification is all right.

In this exercise we'll find out what happens if we want to minimise `Y` instead of maximising it.

Replace '`MAXIMIZE`' with '`MINIMIZE`' in the `SOLVE` statement.
Explain why the model fails to get an optimal solution. Look at the listing file to find out what the model status is.
Repair the error and save the correct file.

Exercise 3.1.7. Building a new model

Now that you have learned to build and debug a GAMS model, you should be able to implement your first environmental-economic model in GAMS. Let's start with a general description of the model:

Due to the existence of open access pastures there exists a tragedy of the commons. In West Africa pasture productivity is threatened by overgrazing, a headtax is envisioned to induce farmers to lower their livestock numbers. If we consider an area of 1000 hectares the present number of cattle is 2000 (without a headtax). The number of cattle supply decreases with 50 for each currency unit of tax imposed. The stocking rate is defined as the number of cattle per hectare. The damage per hectare due to grazing is the square of the stocking rate and valued at cost of 24 currency units. Livestock keeping itself is seen as a social beneficiary activity and valued at 50 currency units per head of cattle. Taxation is seen as undesirable. Hence, the social objective function consists of the positive effects of cattle keeping minus the tax burden and minus the value of damage due to overgrazing.

Implement this model in a new GAMS file called “intro2.gms”.

Note: if the model is solved properly, it will give a MODEL STATUS 2: locally optimal. This does not mean that there are 2 local optima. In non-linear models, like this one, GAMS will never give a model status 1: optimal.
Hints:

- you can use the following parameters: `pastures` (=1000), `cattle0` (=2000), `taxcoef`(=–50), `benefit` (=50), `cost` (=24).
- you need the following variables: `tax`, `damage`, `st_rate`, `cattle`, `obj`. Except for tax, each variable has its own equation based on the description above:
```
eq_cat..  cattle =E= cattle0 + taxcoef*tax;
eq_sr..  st_rate =E= cattle/pastures;
eq_dam.. damage =E= cost*st_rate*st_rate*pastures;
eq_obj.. obj =E= (benefit-tax)*cattle – damage;
```

Fix the tax rate (`tax`) at 20 by typing “`tax.fx=20;`” somewhere between the variable declaration and the solve statement. How does this influence the social objective (`obj`)? What is the value for the social objective if the tax rate is 10? And if the tax rate is 0?

Exercise 3.1.8. Understanding what you have done

This exercise is to test whether you understand what you have been doing so far.

Hint: use the information in the appendix to check your model results.

a. Go back to the file intro (if necessary, open the file again). Replace the defined value for variable `X2` in the equation by a parameter `B` that has the same value (`5`). Make all the necessary changes to the model so that the model works and the same values for `X1` and `Y` result. Check whether GAMS has found an optimal solution. Repair any errors you get underway.

b. As a final test at this level, rewrite the model so that it has an environmental-economic meaning:

Replace equation `QX1` with `QPRD.. PRD =G= 100`;
Replace equation `QY` with `QEMIS.. EMIS =E= CO2+OTHER`;
Add the equation `QCO2.. CO2 =E= coef*PRD`;
Rename the model as `CLIMATE`;
Change the objective from maximizing `Y` to minimizing `EMIS`;
Make sure all parameters and variables are declared and the parameters are given a value (`OTHER=5` and `coef=0.03`) and remove all redundant code.

Run the model and analyse the results.

If you feel confident that you grasp the exercises above, go to the next level. If not, try to play around with the model some more (for example, you can specify parameter `OTHER` as a variable).

3.2. Intermediate level

When you have successfully completed the introductory exercises, you are now ready to learn more of the syntax that is used in GAMS to specify your models. Step by step, elements of the GAMS language will be added, building up from the simplest model to technically more sophisticated models. The goal of the Intermediate exercises is to enable you to specify a model in GAMS based on a well-defined idea of the model characteristics. It is assumed that you will run the model after each exercise and check the solution listing for changes.

Beginner modellers are recommended to use the more sophisticated syntax as much as possible (that is, if you understand the meaning), as this will make things much easier for you when you start building larger models.

Exercise 3.2.1. DISPLAY your results

To make the results more convenient to read, you can use a display statement. In our example, we're not really interested in the value of `PRD` or `OTHER`, but only in the values of `CO2` and `EMIS`. So we add a new line at the end of our code to display the value of variable `EMIS`. You have to specify whether you are interested in the level value of `EMIS` (then use `DISPLAY EMIS.L`), its lower bound (`EMIS.LO`), its upper bound (`EMIS.UP`) or the marginal value (`EMIS.M`). Normally, you are only interested in the level value (`EMIS.L`). Note that these four categories are all represented in the solution listing as discussed above.

Now, if we also want to display the value of parameter `OTHER`, we do not have to specify the `.L` since parameters only have a value, not any bounds or marginal value. You can include more elements in one `DISPLAY` statement by separating them with commas. Combined, this should do the trick:

At the end of the code, add the following line:

```
DISPLAY EMIS.L,OTHER;
```

Save the model under the new name "exercise_321" by selecting File / Save As from the menu and run the model.
Look at the listing file and especially at the results of the display statement. Does it look similar to the results in the appendix?

Save each model with the name of the exercise to identify it easily in the future. At the end of this course you will have a cluster of files in your folder.

Exercise 3.2.2. Commenting out single lines

If you want to comment out a single line, so that GAMS does not read the line, put an asterisk at the beginning of the line.

Between the declaration of the equations and the equation themselves, add the following line:

```
* The equation definition block:
```

Commenting out lines is useful to add some explanations between the GAMS code, but can also be used to remove a part of the code from the model without actually throwing away the text. In this way, you can easily reactivate the line you have commented out if you want to.

Exercise 3.2.3. Using SCALARS

When you define one single parameter you can use scalars. Parameter `OTHER=5` can also be defined with a GAMS code as:

```
SCALARS
OTHER Emissions of other greenhouse gasses      /5/;
```

As you can see, the value of the scalar is given directly in the declaration of the scalar, between slashes. This way of giving values to scalars is convenient (and widely used), but you can also skip the assignment of the value in this statement and assign the value in a separate statement:

```
SCALARS
OTHER Emissions of other greenhouse gasses;
OTHER = 5;
```

Both ways are equivalent. You can also use the direct assignment of values for parameters, though this is often less convenient (and not common).

Note that if you want to change the value of a scalar or parameter later in the model, you cannot use slashes, but have to stick to the separate assignment statement using the equal sign.

Take the climate model and define the emission coefficient `coef` and `other` as scalars.

If you have a single parameter with only a single value, you can choose whether you define it as a scalar or as a parameter.

Exercise 3.2.4. Solving more than one model

The first three exercises at this intermediate level were rather superficial, though in practice they turn out to be very useful. In this exercise, you'll learn how to do so-called counter-factual simulations, *i.e.* you'll learn how to make two different simulations with the same model. This involves solving the model twice and comparing the stored results.

Suppose that we want to know what influence the scalar `OTHER` has on the results for `EMIS`. We store the results of both solves into parameters and can then compare the parameter values (this is necessary since GAMS overwrites the results of the first solve, the variable levels, during the second solve).

This is done as the following:

(i) after the solve statement, declare a new parameter `RES1` and assign it to have the same value as `EMIS.L`;
in GAMS syntax: `RES1 = EMIS.L`;
(ii) change the value of `OTHER` to 7 by adding a new command at the end of your code (after the solve statement!);
in GAMS syntax: `OTHER = 7`;
(iii) give another solve statement;
(iv) declare a new parameter `RES2` and give it the value of `EMIS.L` (which has changed after this second solve; hence, you have to give the value to `RES2` after the second solve statement);
(v) compare `RES1` and `RES2` (use a `DISPLAY` statement);
(vi) analyse the result.

Note that you also learned how to assign a new value to a parameter or scalar; the syntax follows the obvious mathematical notation.

Exercise 3.2.5. Using multidimensional PARAMETERS

We already introduced how to use scalars and single parameters, now we will introduce two dimensional parameters.

Replace the declarations of scalars `RES1` and `RES2` by `PARAMETER RES;`
Store the value of `EMIS` after the first solve in the parameter:
`RES(“1st solve”) = EMIS.L;`

Note: You can declare `RES` only once. In this case declare it where the declaration of `RES1` was before.

The text between brackets is called the *identifier* of the parameter: it tells GAMS which entry in the vector should be used.

Store the value of `EMIS` after the second solve in the same parameter, but using a different identifier:
`RES(“2nd solve”) = EMIS.L;`
Change the display statement to display `RES` and not `RES1` and `RES2`.

The use of the parameter instead of two scalars has the major advantage that the model becomes more compact: fewer lines are needed and items that belong together are stored together. For small models, this is not so much an issue, but for larger models this may increase the readability of the model significantly.

Exercise 3.2.6. SETS and vector specification

In the exercise above, we specified a parameter with two distinct values, using an identifier between brackets. But we did not specify any *domain* within which this parameter identifier should lie. Think of the domain as all identifiers that are allowed for the parameter, *i.e.* it indicates which elements are included in the vector that makes up the parameter.

Such a domain can be specified using the `SET` statement. The `SET` statement gives the index of the parameter, the list of all identifiers that are possible.

In the exercise above, two identifiers are possible for parameter `RES`: “`1ST SOLVE`” and “`2ND SOLVE`”. So parameter `RES` is a vector with 2 values, and the domain is given by the set {“`1ST SOLVE`”, “`2ND SOLVE`”}.

At the beginning of the model code, declare a `SET` called `SOL`, and list all possible elements between slashes:

```
SETS
          SOL       List of all solves
                    /"1st solve", "2nd solve"/
```

Next, in the declaration of the parameter `RES`, tell GAMS that only identifiers that are an element of set `SOL` are allowed, by replacing `RES` with `RES(SOL)`; you do not have to change the commands where the elements of `RES` are given a value.

Now let’s go one step further with the use of sets. In economic models, sets are often used to distinguish production sectors. Up to now, we’ve had only one production value, `PRD`. Suppose that there are three production sectors that each produce. Then the variable `PRD` can represent sectoral production, or in GAMS syntax, `PRD` becomes `PRD(J)`.This influences the model at several places: first, a new set, `J`, has to be introduced, containing three elements; then, the corresponding variables and equations have to be declared and specified for each element of `J`; and finally, the equation specifications are changed.

Note: the following changes are within the existing code; do not add these lines to the end of the file.

- Change the model to include a new set `J` with 3 elements (`1`, `2` and `3`) (at the top of your code);
- declare the existing variable `PRD` and equation `QPRD` as a vector over `J` (add `(J)`);
- include a new parameter `PRD_DATA(J)` to define that sectors `1`, `2` and `3` produce 10, 50 and 40, respectively:

-
  ```
  PARAMETER
  PRD_DATA(J);
  PRD_DATA(“1”) = 10;
  PRD_DATA(“2”) = 50;
  PRD_DATA(“3”) = 40;
  ```
- change the equations `QPRD` such that it uses these sectoral data:
  ```
  QPRD(J).. PRD(J) =G= PRD_DATA(J);
  ```
- Finally, change equation `QCO2` such that all production values contribute to the emissions:
  ```
  QCO2.. CO2 =E= coef*(PRD(“1”)+PRD(“2”)+PRD(“3”));
  ```

Exercise 3.2.7. Summing over an index

The summation of the production values just introduced can be rather cumbersome if the number of sectors becomes large. Therefore, it is often more useful to use the `SUM` command. You can sum any parameter or variable over the corresponding set: for instance, summing variable `PRD` over the set `J` is done by `SUM(J, PRD(J))` $= \sum_j PRD_j$.

Change the equation `QCO2` to include a sum of `PRD` over `J`.

Exercise 3.2.8. Including time: a dynamic specification

In this exercise, we add another set to account for time. All parameters and variables are given an additional index, `T`, so that they are specified for each period of time. As you cannot give an index to a scalar, `COEF` and `OTHER` now have to be specified as parameter. Also, the production variable `PRD` now gets two indexes: `J` and `T`. This is one of the strong points of GAMS: you can write `PRD(J, T)` and GAMS will interpret it as a whole matrix of all possible combinations of `J` and `T`; you do not have to write all elements by hand.

Introduce a new set `T` that has elements 2000 to 2004. This is most easily done by writing `/2000*2004/` as possible elements of `T`. The star indicates that all values between 2000 and 2004 are also included.
Change the `SCALAR` statement into a `PARAMETER` statement and add time to `COEF` and `OTHER`. Give separate commands to specify the values of these new parameters.

This parameter block now looks something like this:

```
PARAMETER
      coef(T)     Emission coefficient CO2
      OTHER(T)    Emissions of other greenhouse gasses;
coef(T)  = 0.03;
OTHER(T) = 5;
```

Add an index T to all parameters, variables and equations.

Do NOT run the model yet, but read on.

Note that
1) parameter RES now contains two indices: RES(T,SOL)
2) you have to write EMIS.L(T) and not EMIS(T).L in the code, and
3) you should never add the index, in this case (T), in the DISPLAY statement.

Now we have a new problem: we can no longer use EMIS as the variable to be minimised, as there is more than one value of EMIS. So, we introduce a new variable and a new equation summing EMIS over time. This new variable can then be the quantity to be minimised.

Add a variable TOTEMIS and an equation QTOTEMIS stating that TOTEMIS equals the sum of EMIS over time. Make TOTEMIS the variable to be minimised in both solves.

Exercise 3.2.9. Using TABLES for data input

You can provide data for a parameter with a separate command for each element, or by using a formula (the easiest formula is to give each element the same value, as in the exercise above). However, if you know the values of the elements you can also write the values in the form of a table. Tables are declared by the TABLE statement and are given a name just like parameters, sets, equations, *et cetera*. Tables are two-dimensional: columns and rows. In the declaration, the first index describes the rows and the second index the columns. You can use a star (*) if either the columns or rows are not made up of a set.

In the following example the table DATA is declared with row elements A and B and the column is the set T:

```
TABLE     Data(*,T)
          2000      2001      2002      2003      2004
A_DATA     3         4         5         6         7
B_DATA     1         1         1         1         1;
```

Now we can get the data of the first row out of the table into the parameter X as follows:

```
X(T) = Data("A_DATA",T);
```

In our model, we suppose we have production values for each of the sectors.

Add the following table:

```
TABLE DATA(J,T)     Input data for PRD
         2000      2001      2002      2003      2004
1          10        11        12        13        14
2          50        52        54        56        58
3          40        42        44        46        48;
```

Put the values from the table in the parameter PRD_DATA(J,T):

```
PRD_DATA(J,T) = DATA(J,T);
```

Analyse the results from the model solutions.

Exercise 3.2.10. Defining POSITIVE VARIABLES

If you know that some variables cannot be negative, you can tell GAMS that they are `POSITIVE VARIABLES` (by using the statement with this name). All variables can be included in this statement, except for the variable that is optimised. The effect of this statement is only that GAMS solves the model quicker; the numerical results will not be different.

Add a command, after the variable declarations but before the model statement, that states that `CO2`, `EMIS` and `PRD` are positive.

Remember that the optimand variable cannot be constrained `(TOTEMIS)`.

Exercise 3.2.11. Providing starting values

Before we go on, first a reminder that when you solve the models, you have to ALWAYS carefully check the solve summary: did you find an optimal solution or were there problems. Don't jump directly to the results, since the results of an error-model are meaningless.

To help GAMS find the optimal solution, you can provide starting values. These starting values should, if your model is well-behaved, have no impact on the outcome itself, but does speed up the iteration process so that GAMS will find the optimal solution faster. For some models, GAMS cannot find the optimal solution at all if you don't provide starting values. If you do not provide starting values, GAMS will implicitly take zero as starting value. The syntax for the starting values is straightforward: before the solve statement you provide a `.L` value to the parameters, for instance `EMIS.L(T) = 10;`.

Provide starting values for `CO2` and `EMIS` of 1 and 6 for each period.
Check the solution listing for any changes.

Exercise 3.2.12. Providing lower and upper bounds

If you have information that the value of a variable cannot get lower (or higher) than some value, you can provide lower (or upper) bounds. This is done in a way similar to providing starting values, only now you don't use the `.L`, but rather the `.LO` for lower bounds and `.UP` for upper bounds.

Lower bounds are very useful if you have an equation that is invalid if a variable becomes zero (for instance `Z =E= X/Y;` if `Y` is zero, then `Z` is undetermined). Then provide a small but positive lower bound on `Y` (like 0.0001).

Include an upper bound of 55 on production of sector 2 and a lower bound of 12 on production of sector 1 for each period. Make sure these bounds apply in the first solve.
Do these bounds influence the solve? Explain how and why.
Remove the bound that causes the model to run incorrectly, run the model again and analyse the results.

Note that these bounds will apply until you change them. So they also apply in the second simulation!

Exercise 3.2.13. Fixing variables

Fixing a variable is the same as providing a lower and an upper bound equal to each other. The shortcut way to do this is to use `.FX` instead of `.LO` and `.UP` and then provide the fixed value. You can ‘unfix’ (release) a variable by providing new bounds; if you want to release the variable completely, provide a lower bound of “`-INF`” and an upper bound of “`+INF`”.

Fix the value of `PRD(“3”,”2000”)` to 45 in the first simulation and release it completely before the second simulation.

Explain how this influences the results of the first solve.

Exercise 3.2.14. Using the LOOP statement

To repeat a number of commands over all elements in a set you can use the `LOOP` statement. Often, a loop is used to do multiple solves. In our case, the loop will then be over set `SOL`. The syntax is

```
LOOP(SOL,
*{add the commands you want for each element in set SOL,            }
*{these could be SOLVE statements, parameter calculations, et cetera }
);
```

The loop makes GAMS first use the first element of the set (in this case “`1st solve`”), go through all the statements given in the loop, and once it reaches the end of the loop, go back to the start of the loop and go through all the statement using the second element of the looped set.

We want to include a loop over all simulations and include the solve statement, the calculation of `RES` (now use `SOL` as the first index, and not “`1ST SOLVE`”) and the new bounds and values for the second solve, so we write directly after the MODEL statement:

```
LOOP(SOL,
        SOLVE CLIMATE USING DNLP MINIMIZING TOTEMIS;
        RES(SOL,T) = EMIS.L(T);
*Prepare for the second solve:
        PRD.LO('3','2000')= -INF;
        PRD.UP('3','2000')= +INF;
        OTHER(T) = 7;
);
DISPLAY RES;
```

Note that you cannot declare a parameter inside a loop, so you’ll have to move that command to above the start of the loop.

Use the LOOP statement to solve over both simulations, calculate parameter RES and change the bounds and values for the second solve.

You can also use the loop statement to calculate the values of a parameter, as we will see later.

Exercise 3.2.15. Using conditional statements ($-operations)

Before we deal with conditional statements, we will first add damages to the model to make it more realistic.

Add a new equation, `QDAM(J,T)` stating that the value of the new variable `DAM(J,T)` equals 0.05.
Change the equations `QPRD` such that production is greater than production data times `(1-DAM(J,T))`.

Explain the effects of this change on the results.

Conditional statements are GAMS' way to say 'only if'. If you write `X=2+3$(Y>1)`, then X will be 2 for values of Y below 1 and X will be 5 for Y-values above 1; the statement is read as 'X equals 2 plus 3 if Y is bigger than 1'. These dollar-operations can be used to conditionally assign values to parameters, but also to equations. For example, the relation between X and Y can also be written as:

```
QX1$(Y<=1).. X =E= 2;
QX2$(Y>1)..  X =E= 5;
```

Note: `Y` has to be a parameter and not a variable for GAMS technical reasons.

In our model, we will specify that damages are larger for higher production values.

Make a copy of the line with the equation for `QDAM(J,T)` and put it just below the original line. Comment out the original line (see Exercise 3.2.2). Add 0.05 to the damages in a sector in a period if the production data (`PRD_DATA(J,T)`) are above 50:

```
QDAM(J,T).. DAM(J,T) =E= 0.05 + 0.05$(PRD_DATA(J,T)>50);
```

Check whether the results of this revised model are the same as before.

Exercise 3.2.16. Using lags and leads in parameters and variables

When you specify a relationship between period T and T-1, this is called a "lag". When you specify a relation between period T and T+1 this is a "lead". A common example of a lead is the build-up of capital stock using investments: capital stock in period T+1 depends on capital stock in period T plus investments in period T: `K(T+1)=K(T)+I(T);`.

Lags and leads can be used for both parameters and variables, though there are some restrictions for use with variables.

Make the parameter `COEF` dependent on the sector by adding an index J:
`CO2(T) =E= SUM(J,COEF(J,T)*PRD(J,T))`
Provide 2000-data for `COEF` (0.01, 0.04 and 0.025 for sectors 1, 2 and 3 respectively) and then use a loop to calculate `COEF(J,T+1)` as `COEF(J,T)*0.99`.

Note that the new specification of the CO2-equation is not equivalent to
`CO2(T) =E= SUM(J,COEF(J,T))*SUM(J,PRD(J,T))`

Next, assume that the conditional statement in the equation for damages depends on the production quantities in the year before instead of the current year.

Change the equation for damages such that the conditional statement depends on the production data in the year before. Analyse the effects on the damages.

```
QDAM(J,T)..         DAM(J,T) =E= 0.05+0.05$(PRD_DATA(J,T-1)>50);
```

Exercise 3.2.17. Raising to a power

Raising a variable to some power can be done by using two stars followed by the power number; if you want to square, the power number is 2.

In the equations for the production quantities, put a square on the damage factor `(1-dam(j,t))`.

Notice that the model has now become non-linear. The model status has now changed from ‘optimal’ to ‘locally optimal’. This does not have to worry you, this status automatically changes when the model becomes non-linear.

Exercise 3.2.18. Using ORD and CARD

The autonomous decrease in the emission coefficient every year with 1% was written as

```
coef(“1”,”2000”) = 0.01;
coef(“2”,”2000”) = 0.04;
coef(“3”,”2000”) = 0.025;
LOOP(T, coef(j,t+1) = coef(j,t)*0.99).
```

The loop is used here because you want to calculate the time-dependent coefficients for every year. This cannot be done by just stating `coef(j,t+1) = coef(j,t)*0.99`. The intuition is simple: when GAMS comes to this line, it only knows the values for the first year (2000). So it can only calculate the values for 2001. Using the loop, you then calculate 2002, 2003 etc. until all periods are calculated.

But, alternatively, you could write:

```
coef(“1”,T) = 0.01*(0.99**(ORD(T)-1));
coef(“2”,T) = 0.04*(0.99**(ORD(T)-1));
coef(“3”,T) = 0.025*(0.99**(ORD(T)-1));
```

The two specifications are mathematically equivalent. The `ORD` that is taken of index T gives the relative position of the active value in the set. So in our example, `ORD(“2000”)` equals 1, `ORD(“2001”)` equals 2, et cetera.

Note: in order to keep your code as simple and neat as possible, perhaps you can rewrite the statements above in vector notation.

Change the autonomous decrease in the emission coefficients by using the `ORD` specification. Check that the values of `coef` have not changed.

Optional:
Another convenient operator is `CARD`. The `CARD` value of a set is the total number of elements in the set. So, in our example, `CARD(T)=5`.

Both operators are often used together. For example, a straight interpolation from 0 to 1 over all elements in T can be modelled as `(ORD(T)-1)/(CARD(T)-1)`.

Check that this is a straight line that starts at 0 and ends at 1 by adding a parameter `INDEX(T)` and displaying index.

Exercise 3.2.19. Using a free variable

Sometimes you want to add a variable to your model for which you do not have an equation, but want GAMS to find the optimal value. In this exercise, we will add such a ‘free variable’. Note that the term ‘free variable’ is not a GAMS code word; free variables are ordinary variables, only without an equation to determine their value.

Suppose it would make sense to weigh the CO2 emissions and the other emissions in the calculation of total emissions (in real life, this doesn’t make sense: you just add different greenhouse gas emissions, without weighing them first; we use this specification for illustration purposes only). What would then the optimal value of the weights be, given that they have to sum to unity? GAMS can answer this question for you if you introduce the weights as free variables.

Add a new variable, `alpha`, and provide a lower bound of 0.2 and an upper bound of 0.8 for `alpha`. `Alpha` can be described as the weight of CO2-emissions.

Change the equation for emissions into:

```
QEMIS(T)..  EMIS(T) =E= CO2(T)*alpha*2 + OTHER(T)*(1-alpha)*2;
```

In this case, if `alpha` is 0.5, both weights equal one and total emissions is just the sum of `CO2` and `OTHER` emissions. If `alpha` exceeds 0.5, more weight is given to `CO2`, and if `alpha` is lower than 0.5 more weight is given to `OTHER`. Note that the lower and upper bound on `alpha` prevent extreme situations where only one of the two sources of emissions matters.

Run the model and check the results. Can you explain the optimal value of `alpha`?

We can now go back to Exercise 3.1.7 and change the parameter tax into a free variable. You can now easily calculate the optimal tax rate in this simple exercise.

Open the model from Exercise 3.1.7 and calculate the optimal tax rate within GAMS.

Exercise 3.2.20. Mapping set elements (optional)

If you want to aggregate a certain data set, you can use a utility in GAMS called “set mapping”. Suppose you have a 5-sector input-output (IO) table (“largedataset”) and want to aggregate this into a 3-sector IO-table (“smalldataset”). To achieve this in GAMS, you can identify a special set:

```
SET MAP(largedataset,smalldataset)
    /light_ind.industry, {more mapped sectors…}/;
```

In `MAP` you state which sectors in the large dataset correspond to which sectors in the small dataset. You use a period (`.`) to connect both elements, *e.g.* `light_ind.industry` means that the sector “light_ind” in the large dataset belongs to sector “industry” in the small dataset.

Once you have established all connections between both datasets, you can sum the elements in the large dataset, conditional on the mapping. For example, if you wish to find the new sectoral consumption levels, you can calculate:

```
smalldataset(i_small, "cons") = SUM(i_large$map(i_large, i_small),
                                largedataset(i_large, "cons"));
```

The full example of using mapping to aggregate a larger dataset into a smaller dataset can be downloaded. The model shows that you can use GAMS to build the new data table.

Download the model “MAPPING.GMS” from the GAMS-homepage of the Environmental Economics Group of Wageningen University.

Add the missing code to calculate the column totals and row totals of the small dataset, using the column and row totals of the large dataset and the mapping.

Hint: you can check whether the calculated totals are correct by looking at the auxiliary parameter `CHECK`. This checks whether subtotals add up to grand totals and row totals equal column totals.

Exercise 3.2.21. Exporting your results to Excel (optional)

With GAMS version 21.0, a new utility has been added to GAMS: GDX. This utility eases the import of raw data from other programs and the export of GAMS-results to other programs.

In this exercise, the results of the model are exported to Excel. You can import data from Excel in a similar way. Check the separate GDX documentation available in GAMS-IDE via Help/docs/gams/gdxutils.pdf.

Two lines are needed to move the results to Excel.

```
execute_UNLOAD '3_2_21.gdx', res;
execute 'GDXXRW.EXE 3_2_21.gdx par=res';
```

The first line creates a new file: 3_2_21.gdx. The file contains the contents of the parameter RES.

The second line converts the just created gdx-file into an Excel file: 3_2_21.xls. You can find these files in your project directory.

Go back to the model of Exercise 3.2.18. If necessary, fix alpha at 0.5.

Add the two lines to the end of your code.

Open Excel and check the results.

Exercise 3.2.22. Understanding what you have done

This exercise is to test whether you understand what you have been doing so far.

Go back to the model of Exercise 3.2.18. If necessary, fix alpha at 0.5.

Add a 3rd simulation to the model, where the value of `OTHER` is 9. Make all the necessary changes so that the model solves all three simulations. Check whether GAMS has found an optimal solution. Repair any errors you get underway.

Hint: `other(T) = other(T)+2`

How about your economic understanding of the model?

For what economic questions could this model be used?
Do you think the relationship between damages and build-up emissions is realistic? Why / why not? How can the model be changed to become more realistic in this point?

If you feel confident that you grasp the exercises above, go to the next level. If not, try to play around with the model some more (especially with the exercises you did not fully understand).

3.3. Advanced level

In the exercises above, a GAMS model was specified, using well-defined, explicit definitions of the model characteristics: the mathematical formula was given and the exercise was to specify this formula in GAMS syntax. This section, the Advanced level, goes one step further and aims at making you familiar with specifying a GAMS model starting from a broad economic description of the model. The mathematical formulae are not always specified and you have to think of the best way to represent the economic relationships in GAMS equations. Furthermore, you will learn how to interpret the results of the GAMS model in economic terms (rather than just in technical terms).

Exercise 3.3.1. Specifying a dynamic optimisation model

In this exercise we will specify a simple dynamic optimisation model.

Equations:

Objective function:	maximize $\sum \{cc(t)*1/(1+r)^{t}\}$
Labour stock:	$L(t) = L0*(1+gL)^{t}$
Capital stock:	K(t) = K(t-1)*(1-delta) + I(t)
Income:	Y(t) = C(t) + I(t)
Production:	$Y(t) = A*K(t)^{0.2}*L(t)^{0.8}$
Consumption per capita:	cc(t) = C(t)/L(t)

Sets:

time	t = 2001 to 2020

Parameters:

annual rate of growth of the labour force:	gL = 0.02
annual rate of depreciation:	delta = 0.10
technology parameter:	A = 1
labour force at t=0:	L0 = 10
capital stock at t=0:	K0 = 20
discount rate:	r = 0.03

Variables:

cc(t)	consumption per capita at time t
L(t)	labour force at time t
K(t)	capital stock at time t
I(t)	investments at time t
Y(t)	industrial output at time t
C(t)	consumption at time t

Specify set `t`, parameters and variables, give starting values to the variables capital and labour.

Notes:

- Starting values are needed to prevent a "division by zero" error (in calculating consumption per capita): `L.L(T) = L0; K.L(T) = K0;`
- declare `C(t)` and `I(t)` as positive variables to prevent very high consumption coupled with highly negative investments, since labour, capital, income and consumption per capita will also be positive, declare them as positive variables as well.

Specify the equations. Remember to name the equations first, use sensible names for the equations. Remember that you can never use the same name twice, so you can not give the same name to two different equations.

Notes:

- In order to calculate a term which has a "to the power t" in it, GAMS needs to be told to take the active value of t. `ORD(t)`, as discussed before, indicates the active value of t. Use `ORD(t)` in the power function.
- The first-period value for capital stock has to be given; one possibility is: `K(t)=E=(K0$(ord(t)=1)+K(t-1))*(1-δ)+I(t);`
- There are two equations with `Y(t)` on the left-hand-side; GAMS does not have a problem with this as long as the equation do not have the same name.

Solve the model and display the optimal investment path I(t).

Note:

- Remember that in order to display the value of a variable you need to specify to GAMS what to display. GAMS can either display the level value (`.L`), the lower or upper bound (`.LO & .UP`) and the marginal value (`.M`). Display the level value of `I(t)`.
- The MINOS solver may provide a different solution to this model than another NLP-solver, Conopt. For small and well-behaving models, they normally provide the same results.

It's time for some basic environmental issues:

Introduce an emission variable: `E(t)=0.5*Y(t)` that is related to the level of output and a damage function which shows that monetary damage is equal to 0.10 times the level of emissions: `D(t)=0.1*E(t)`. Deduct damage from `Y(t)` in the income equation (but not in the production equation). Display `E(t)` and `D(t)`.

Note:

- Deducting damage from national income is easiest done by adding it to the right-hand-side of the equation (i.e. treating damage as a competing category of consumer expenditures). Check the result.

Now introduce an emission reduction function that reduces emissions at increasing variable costs: $CR(t)=0.25*ER(t)^2$ (use the gams code of raising to the power). Deduct these emission reduction costs (`CR`) from the national income and deduct emission reduction (`ER`) from emissions. Calculate optimal path of investment.

Note:

- declare both damages and emission reduction costs as positive variables. Calculate the optimal investment path and the optimal path of emissions and interpret the results.

The last exercise is optional.

Introduce 2 subsets: `T1` for the time period till 2010, and `T2` for the rest.

For example: `T1(T) first period      /2001*2010/;`

Introduce a lower bound on emission reduction (`ER`) of 0.1 for the first period and 0.3 for the second. Analyse the result.

Exercise 3.3.2. Making some more scenarios

This exercise builds upon the model developed in Exercise 3.3.1, without the optional time subsets.

Increase the discount rate from 0.03 to 0.06 and 0.09. Analyse the differences in investment and emissions.

Change the discount rate back to 0.03.

Implement the possibility of technological progress by assuming the following development over time of the technology parameter:
$A(t) = A^*(1+g_A)^{(t-1)}$, with g_A equal to 3%.

Analyse the differences in optimal investment and emission path.

Change the value of g_A to zero.

Implement government policy which limits total emissions to 6.5 in each period.

Note: implement the policy as an upper bound on total emissions.
Analyse the effects for investment and consumption. Calculate the changes in national income.

Exercise 3.3.3. Specifying a model for transboundary pollution (optional)

In this model, a highly stylised description of the economy is given and the emissions and deposition of a transboundary pollutant are specified[5]. Two countries are specified and the effects of unilateral policies of one country are investigated.

Download the model “ACID.GMS” from the GAMS-homepage of the Environmental Economics Group of Wageningen University.

Add three scenarios to the model:
(i) solve the model without any environmental policy (base scenario);
Add a parameter `REPORT` to store all the important results from the scenario; for example, you can store the level of `FUEL` as follows:

```
Report("FUEL",EMcntry,"No Policy")      = FUEL.L(EMcntry);
```

Report can handle only three dimensions, so reduce the dimensions of `emred` and `emis`.

```
Report("EMIS SO2",EMcntry,"No Policy")  = EMIS.L(EMcntry,"SO2");
```

(ii) solve the model with an upper bound on emissions in NETHERL of 80% of the base level;
(iii) solve the model with an upper bound on acid-deposition in NETHERL of 80% of the base level;
{Hint: remember to remove the upper bound on emissions.}

After each solve statement specify a report that contains the optional values of `fuel, renew, conserv, emis so2, emis nox, emred so2, emred nox, aciddep, ccntry` and `c`. Run the model and analyse the results.

[5] This model is based on a simplification of work by E. Schmieman, Wageningen University.

(iv) change the required emission / deposition reduction from 20% to 50%. Run the model again and compare the results to the previous ones.

Exercise 3.3.4. Investigating international co-operation (optional)

Using the model specified above (Exercise 3.3.3.), the advantages of international co-operation are investigated.

Change the model from Exercise 3.3.3. such that the energy system in Germany is fixed (`FUEL`, `RENEW`, `CONSERV` and `EMRED` are fixed at the level of the base scenario) and emission / deposition reductions (use 20% required reductions) in the Netherlands can only be achieved by changes in the domestic energy system.

How large are the costs of this 'Going-Alone' when compared to the 'Going-Together' specification in Exercise 3.3.3.?

Explain what happens if you increase the required deposition reduction to 50%?

How would you describe an alternative specification where both energy systems are flexible but the objective function is changed from minimising `C` to minimising `Ccntry("NETHERL")`? What would be implicitly assumed in this case for the relationship between both countries?

Exercise 3.3.5. A basic systems model (optional)

Essential elements in a systems model are positive and negative feedback loops. The central feedback loops in the WORLD3-model of Meadows *et al.* (1992) are loops on population and capital as shown in the paper by Meadows *et al*. We start by making a simple system in GAMS that consists of feedback loops on population and capital stock.

Model a feedback loop on capital stock by assuming that investments in year t+1 (`I(T+1)`) are 10% of the capital stock in year t (`K(t)`) and that depreciation in year t+1 (`D(T+1)`) is 5% of the capital stock in year t.

Mathematically this feedback loop can be expressed by three equations:

$$I_{t+1} = 0.1 \cdot K_t; \; D_{t+1} = 0.05 \cdot K_t; \; K_{t+1} = K_t + I_{t+1} - D_{t+1};$$

Introduce the necessary sets, variables and equations to model this system.

Make a 50 year projection of the development of the capital stock volume from 2000 till 2050, *i.e.* solve the model. Assume that the volume of capital stock in 2000 is 100 (fixed value, not just starting value).

Note: GAMS always has to optimise a variable. Introduce an objective variable and maximise it. Does it matter how you define the objective variable in this exercise?

Now we add a feedback loop on the development of population to the model.

Add an equation on population (`P(t)`) by assuming that the number of births in year t+1 (`B(T+1)`) equals 2% of the population in year t. The number of deaths in year t+1 (`DE(T+1)`) equals 1% of the population in year t.

Hint: first write down the (three) mathematical expressions of this feedback loop.

Make a 50 year projection of the development of the world population from 2000 till 2050 (solve the model). Assume that the world population in 2000 is 6000.

Note: in this example, the population is measured in millions of people, as the total number of people in the world is 6 billion.

Now we include the interaction between the feedback loop on capital stock and population.

Add to the model an equation for pollution (`POL(t)`) and edit the equation for births. Assume that a 1% increase of the capital stock increases pollution by 0.25%; furthermore, assume that a 1% increase in pollution lowers the number of births by 0.15%.

Mathematically this is expressed as:

$$Pol_{t+1} = Pol_t + 0.25 \cdot \left(\frac{K_{t+1} - K_t}{K_t} \right) \cdot Pol_t; \qquad B_{t+1} = B_t - 0.15 \cdot \left(\frac{Pol_{t+1} - Pol_t}{Pol_t} \right) \cdot B_t$$

Provide starting values for births and pollution: B(“2000”) = 120 and Pol(“2000”)=20.

Change the coefficient in the birth equation from 0.15 to 0.25. Make new projections on capital stock, population and pollution from 2000 till 2050. Compare the new projections with earlier results. Import your result to excel for the comparison. Which population trend is represented in the model (see Figure 4-2 in ‘Beyond the limit’[6])?

Exercise 3.3.6. Understanding what you have done

This exercise is to test whether you understand what you have been doing so far.

Write a very simple model in GAMS, using any model specification you like (but do not copy the specification of the exercises above). Keep the model small and simple and make sure you understand the economics behind the model.

If you feel confident that you grasp the exercises above, go to the next level. If not, try to play around with the model some more (for example, include a report variable in the model to get a table with all results at the end of your listing file).

[6] See D.H. Meadows et al, 1992, ‘Beyond the Limits, Global Collapse or a Sustainable Future’, Earthscan Publications Limited, London.

4. MODELLING EXERCISES

In the sections above, the basic GAMS modelling rules are explained and practised. In this section, your GAMS skills are tested, by some exercises that specifically deal with the model types as treated in the course Theories and Models in Environmental Economics (ENR-30306).

4.1. Linear optimisation modelling

Exercise 4.1.1. A basic linear optimisation model

GAMS comes with a large library of models for everyone to use. These models can be used as a starting point for your own analysis. In this exercise we will take a simple linear agricultural model and introduce fertiliser use as an environmental issue.

You can open files from the GAMS Model Library by going to FILE, then Model Library, then Open GAMS Model Library and searching for the file you want. In this case, the file you want is "Demo1", a Simple Farm Level Model.

Open the model "DEMO1" from the GAMS Model Library and save it as Lin_opt.gms.

The original model does not consider the cost of using fertilisers. To add more realism to this model, consider the effects of using three fertiliser types (Nitrogen (N), Phosphate (P) and Potassium (K)) on net income. Each crop has different fertiliser requirements. In the table below we can observe the fertiliser needs of the various crops per cropping season, *i.e.* per year.

Crop/fertiliser (kg/ha)	Nitrogen (N)	Phosphate (P)	Potassium (K)
Wheat	10	4	5
Clover	0	0	0
Beans	2	8	2
Onions	30	20	20
Cotton	10	5	9
Maize	12.5	5	7.5
Tomato	50	45	80

The cost of these fertilisers per kilo is

Nitrogen	= \$ 0.19
Phosphate	= \$ 0.23
Potassium	= \$ 0.17

Include the fertiliser requirements for each crop and the price of the fertilisers per kilogram, in the model.

Hint: You can use a set `F` for the different fertilisers, a table `FERTREQ(C,F)` for the data table above, and a parameter `FPRICE(F)` for the fertiliser prices.

If you just add the fertiliser cost, total costs increase. Therefore, we adjust the numbers of `MISCOST(C)` by subtracting the fertiliser cost. Introduce parameter `FERTCOST(C)` for this purpose. The total fertiliser costs can be calculated by multiplying the fertiliser requirements by the fertiliser price and then summing over the different fertilisers. You can check that the total fertiliser costs in dollar per hectare are:

Wheat: 3.67; Clover: 0.00; Beans: 2.56; Onions: 13.70; Cotton: 4.58; Maize: 4.80; and Tomato: 33.45.

Correct `MISCOST` for the fertiliser costs and make sure that total costs are unaffected.

After subtracting the fertiliser costs from `MISCOST`, make sure that fertiliser costs are accounted for in the equation “Cash cost accounting”. Now, we can reduce the impact on the environment by limiting fertiliser use. To get a good reference point, we first simulate a benchmark in which the maximum fertiliser use is so high that it will not be binding.

Add a model equation that limits the use of fertilisers on the farm per year. Start with the following amounts (in kilograms):

Nitrogen	= 9999;
Phosphate	= 9999;
Potassium	= 9999.

Run the model and interpret the results. Add fertiliser use (total use per type) to the report.

Hint: Do not type the maximum numbers directly into the equation, but rather use a parameter `FERTMAX(F)`. Make sure you state the model equation as a weak inequality, not as a strict equation, to give the farmers the opportunity to use less than the maximum amount of fertiliser. (Check what happens if you use a strict equality.)

Next, we want to see how the model reacts if we strictly limit fertiliser use.

Add a second simulation at the end of the code in which you limit fertiliser use to the following amounts (in kilograms):

Nitrogen	= 125;
Phosphate	= 100;
Potassium	= 250.

How does this affect the decisions of the farmer? Check the impact of this fertiliser restriction on farm revenue, crop production, the use of labour, fertiliser use (per crop and per type of fertiliser).

(Optional) Add a new report which clearly shows these variables for the situation without and with a strict limit on fertiliser use.

Run the model and interpret the results.

Hint: Introduce a set for the different simulations and define a reporting parameter over this set.

4.2. Partial equilibrium modelling

Exercise 4.2.1. A basic partial equilibrium model

Download the model “PARTIAL.GMS” from the GAMS-homepage of the Environmental Economics group of Wageningen University.

The original partial equilibrium model has two production sectors; pac (paints and coatings) and ohi (other home improvements). Conventional paints and coatings contain solvents, which are released into the atmosphere during application (drying), thus emitting Volatile Organic Compounds (VOCs). In this exercise we set targets to increase the share of solvent-free paints and coatings to reduce VOC emissions. In order to model the production and use of solvent-free paints and coatings, a new sector has to be included in the model. The benchmark table in the original model has to be altered to reflect the inclusion of the new sector.

In order to disaggregate the pac sector to spc (solvent-containing paints and coatings) and fpc (solvent-free paints and coatings), add a column and a row in the benchmark table, and split pac into spc and fpc. Assume a market share of 40% of solvent-free paints and coatings. Also assume the same technology for the two disaggregated sectors (i.e. split the inputs in proportion to the output shares).

The model has a CES (Constant Elasticity of Substitution) utility function. Now that we have disaggregated the “pac” sector into “spc” and “fpc”, we want to change the utility function to a nested CES function. We want it to reflect the fact that different types of paint are closer substitutes than paints in general and other home improvements.

Replace the existing CES utility function with the following nested CES function:

```
U =e= (theta * (lambda*(C("spc")/CBM("spc"))**delta
               + (1-lambda)*(C("fpc")/CBM("fpc"))**delta)**(rho/delta)
      + (1-theta) * (C("ohi")/CBM("ohi"))**rho)**(1/rho);
```

Remember you have to define the additional parameters and assign values to them:

- lambda is the benchmark share of solvent-containing paints and coatings in the total consumption of paints and coatings, which is 60%.
- delta is the substitution parameter. First, you need to define an elasticity of substitution sigma_low for the lower nest. Assume an elasticity of substitution of 2 between different paint types, i.e. assign the value 2 to sigma_low. Calculate the substitution parameter delta analogically to the calculation of rho for the upper nest.
- theta’s calculation also has to be changed. You have to replace the benchmark consumption of pac by the aggregate benchmark consumption of fpc and spc.

First run a benchmark check, then run a counterfactual scenario by increasing the share of solvent-free paints and coatings in the total consumption of paints and coatings to 70% (initially 40%). As this is a counterfactual, you may not change any parameter values nor the benchmark table! Instead, you need to add a model equation to the equation block. First, write down the equation on a piece of paper. Note that you have to add up the consumption of spc and fpc to refer to the total consumption of paints and coatings. Before you can run the counterfactual, you have to add a new model statement (use a different model name!) and a solve statement at the end of the programme. The model statement should contain all equations of the model, including the new constraint which you have added.

Run a benchmark check and the counterfactual. How have the consumed quantities of the three goods changed due to this increase in the share of solvent free paints and coatings? What is the change in utility? How can this result be explained?

Sensitivity analysis: Play around with different elasticities of substitution (bottom nest between 0.5 and 4, top nest between 0.1 and 0.5) to see how this influences the results.

Implement the sensitivity analysis and interpret the results.

4.3. Game theoretic modelling

Exercise 4.3.1. A model for stability of international climate negotiations

Though game theory does not lend itself easily for constructing GAMS models, you can use insights from game theory, e.g. on cooperative versus non-cooperative behaviour, in GAMS models. The model we will use in this exercise looks at the stability of international climate negotiations.

Download the model "GAME.GMS" from the GAMS-homepage of the Environmental Economics Group of Wageningen University.

Carefully read and check the code and investigate its logic. Take your time for this, don't jump straight to revising the model, first make sure you understand it.

Run the model.

Identify which coalition gives the highest global payoff, and which the lowest. And which gives the highest/lowest abatement level? Can you explain the economic rationale for these results?

Hint: You can use the existing parameter `RESULT` to calculate additional information in GAMS that is helpful in answering these questions, *e.g.* `RESULT(z,"global","payoff")`.

One of the interesting coalition structures is the "Kyoto group", consisting of USA, EU, RoOECD and RoEurope.

How will the payoff for the USA change if they leave this coalition? Are the other regions better off or worse off if the USA leaves?

Hint: First identify the number of the "Kyoto coalition" and the number of the coalition with EU, RoOECD and RoEurope. Then compare the regional payoffs.

Some of the numbers used in this model have a high degree of uncertainty. Therefore, it makes sense to do a sensitivity analysis. Normally you would do this by adding new calculations at the end of the code, but given the large number of calculations, you may opt this time to change the existing code.

Play around with the numbers in the benefit-parameter `s(i)` to find out how sensitive the model is to these values.

4.4. Input-output modelling

Exercise 4.4.1. A basic input-output model

Download the model "IO.GMS" from the GAMS-homepage of the Environmental Economics Group of Wageningen University. Run the model.

Specify the equations for production for the input-output model (use `x=Ax+d`). Include also the remaining equations of the input-output model and calculate the values of x_1, x_2, x_3, `m, w, depr` and `profit` per sector.

Add a dummy objective function:

```
QOBJ.. OBJ =E= SUM(I, X(I));
```

Solve the model.

Note: as you can see, the model contains exactly one equation for each row of the IO-table; in a way, you are calculating the totals for each row.

Interpret the results.

Calculate the impact of an increase in the exports of sector 2 of 10%. Again, interpret the results.

Exercise 4.4.2. An extended input-output model

Use the model as specified in Exercise 4.4.1.

Include in the model the equations for SO_2 and NO_x:

	Agr	Ind	Serv	Cons	Inv	Gvt
SO_2	100	200	250	100	30	50
NO_x	40	50	50	60	20	20

Assume that there's a linear correlation between the changes in production and the amount of sectoral pollution. Increase the consumption of agricultural products by 15%. What is the quantitative change in (sectoral) emission?

Exercise 4.4.3. An input-output model with purification (optional)

Optional: Include the purification sector in the model, where the level of purification is exogenous. You must make some changes in the base IO table in order to fulfil the requirement of equilibrium. Assume that the delivery from industry to the purification industry is 15, from services is 5 (and there's no delivery from agriculture). This will modify the total output values. The delivery from the purification sector to agriculture is 15, to industry it is 20 and to services it is 5. The consumption of purification sector products is 15. The import of the newly added sector is 10, depreciation is 5, and wage contribution is 20. The purification sector reduces the SO_2 emissions with 100 tonnes.

Construct the new input output table and calculate the effects of 20% increase in the delivery from purification to industry? Calculate how much the loss in the purification sector will be.

Hint:
- add purification as a new element of the set j (sectors)
- increase the level of *technical coefficient* in the model

Increase the level of purification to 200. Assume that the deliveries from and to the purification sector double. Change the level of production and the level of sectoral profit and make the required changes in `IOTABLE` for this level of purification.

Run the model and analyse the results.

4.5. Computable general equilibrium modelling

Capturing environmental relationships in a CGE model can be quite difficult. In the exercises below, you will first build a basic CGE model, without any environmental characteristics. Later, we introduce emissions and emission taxes and an exercise to build the model by Dinwiddy and Teal (1988) in GAMS.

Exercise 4.5.1. A basic CGE model

Download the model “CGE.GMS” from the GAMS-homepage of the Environmental Economics Group of Wageningen University. Run the model.

This is a general equilibrium model for the closed economy with one consumer and two producers who produce good A and good B.

Add the missing equations (Cobb-Douglas utility and production functions, market balances and budget constraint: see below).

- the *utility function:* The utility function is given by $U = C_A^{gamma} \cdot C_B^{(1-gamma)}$, where C_A is the consumption of good A.
- the *production functions*: The production function for the producers is of the Cobb-Douglas type ($X = \beta \cdot K^{\alpha} \cdot L^{1-\alpha}$) for each producer (remember to use the right indices!).
- all *market clearance conditions*:
  commodity markets: $C_i = X_i$ (i=A,B)
  factor markets: $K_A + K_B = KS$ and $L_A + L_B = LS$.
- *income balance*: $Y \geq \sum_i P_i \cdot C_i$ (note that the available income is recalculated after each iteration)

In the CGE model, prices are equal to the marginal value of the market clearing equations. After the `SOLVE` statement, but still inside the `LOOP`, the model recalculates the prices using the absolute values of the equation marginals:
`price_parameter_name =ABS(Equation_name.m)`

Fill in the parameter-statements: calculate prices and income. Solve the model and check results.

Optional: To test your GAMS skills, you can add profits to be able to check that they equal zero. Profits are calculated as follows:

- a *profit function for both producers*: $PRF_i = P_i \cdot X_i - r \cdot K_i - w \cdot L_i$

(optional) Add a parameter for sectoral profits, and calculate profits after each solve. Add profits to consumer income.

Exercise 4.5.2. Interpretation of the CGE model

Continue the previous exercise. How many goods and markets are there in this model?

Show that the value of output equals the value of inputs plus profits for both producers.
To gain full understanding of the model, conduct some sensitivity analyses.

The first sensitivity analysis is on the technology parameter beta:

Add (at the end of your code) a sensitivity analysis by changing the technology parameter beta: first from 1 to 1.5 and then to 0.5; explain the results.

Note: this type of sensitivity analysis can best be done by adding different solves at the end of your code and making a parameter that stores the results of all solves. If you just change

the parameter value in the main model code, it is much harder to compare the results for the different parameter values.
The second sensitivity analysis is on the weight of good A in the utility function (gamma):

Conduct sensitivity analysis by changing the weight of good A in the utility function: first from 0.5 to 0.25 and then to 0.75 (the weight of good B changes reversibly); explain the results.

Finally, you can play around with the presentation of the results:

(Optional) Construct an Input-Output table to provide a neatly organised result.

Hints:

- Exercise 3.2.20 provides an example of displaying an input-output table.
- you can change the lay-out of the listing file to get a more neat result from the input-output table if you change the page-width and/or font size. You can do this by going to the File-menu, choosing Options and then changing the font size on tab Editor and changing page width on tab Output.

Exercise 4.5.3. CGE model with environmental issues

Start with the model as specified in the previous exercise.

In this exercise, environmental issues will be added. The environmental side will be kept simple, only emissions will be added (no abatement/purification).

Since we are working in a CGE, we will model the emissions as a market like all other goods. To this end, we assume that polluters will have to pay an (equilibrium) price for emissions to the government. This is equivalent to a system of tradable emission permits, where the government auctions the permits (i.e. the polluters buy the emission permits from the government).

We assume that consumer's pollution is proportional to consumption: for each good, the associated pollution equals $\eta_{C,i} \cdot C_i$ for i=A,B and with $\eta_{C,i}$ constant, similarly, producer's pollution is proportional to production (Q) with fixed coefficient $\eta_{Q,i}$.

Add an equation to calculate total emissions caused by good *i* as the sum of consumption and production pollution for the good, using the pollution coefficients (`EtaQ(G)` and `EtaC(G)`) in the table below.

	good A	good B
Consumption ($\eta_{C,i}$)	0.05	0.1
production ($\eta_{Q,i}$)	1.0	0.5

Note that in the GAMS model, good *i* is indicated with index G.

Add a parameter (`PERMITS`) so that we can control the environmental policy. To start with, assume that `PERMITS` equals 1000.
Add another equation stating that the sum of emissions over all goods should be less than the number of permits.

The consumer expenditures on emission permits equal $\sum_{i=A,B} \eta_{C,i} \cdot C_i \cdot p_E$ (where p_E is the price of the emission permits) and reduce the income available for consumption.

Add the consumer's expenditures on emission permits to the other expenditures in the income balance equation.

Next, we have to recalculate the price of emission permits after each iteration and add the total revenues from the sale of the permits to the consumer's income.

After each iteration, add a line to calculate the price of emission permits as the absolute value of the marginal of the equation stating that total emissions should be lower than the number of permits. Add `PE` to the normalisation procedure.

Change the normalisation to use `PE` if wages equal zero (in calculation of parameter `SCALE`).

Furthermore, add the total revenues from sale of the permits, equal to `PE*PERMITS`, to the calculation of consumer's income.

This completes all the changes to the model itself. However, we also want to change the simulations done with the model.

Remove the sensitivity analysis as modelled in the previous exercise. After the base calculation, add two simulations, first one where the number of permits are reduced to 470 and then one where the number of permits is reduced even further to 370. Solve all simulations and check the results.

Make sure that the results are stored correctly in the `RESULTS` parameter.

Exercise 4.5.4. Specifying an open economy (optional)

Now that we have an environmental-economic model for a closed economy, we can extend it to include international trade. So in this exercise, exports and imports are added to the model. International trade adds to the pollution in the country (think for example of emissions from transport).

Open the model from the previous exercise.

Add a parameter `EtaX(G)` to store the pollution associated with international trade. For good A and B, EtaX equals 0.05 and 0.1, respectively.

Add two positive variables: `X(G)` and `M(G)` to account for exports and imports.

We have to change the commodity balance equation (EEQ) to include international trade as follows: `X(G) =E= Q(G) – C(G) + M(G);`

Next, we have to add international trade as a source of emissions. Total emissions from international trade in good G equal `EtaX(G)*(X(G)+M(G))`, so both exports and imports cause pollution.

Change the commodity balance and pollution equations.

Next, we assume that a good can only be exported or imported (we prevent transit of goods) in a new equation: `X(G)*M(G) =E= 0;`

Finally, we add an equation stating that the balance of payments should be in equilibrium, or in other words, the total value of exports has to be equal to the total value of imports:

```
SUM(G,p(G)*(X(G)-M(G)))=E=0;
```

Add the equations above to the model.

Recalculate the different scenarios and compare the results with the previous exercise. To aid this, adjust the `RESULTS` parameter to include exports and imports.

Can you explain the magnitude of exports and imports in all scenarios?

Exercise 4.5.5. Specifying a multi-country model (optional)

We are now in a position to add the final step to this CGE model: adding a second country. We assume there are two countries which both have their own capital and labour supply, and which both produce and consume the same goods. They can trade in the produced goods, but not in capital, labour or emissions.

We change the scenarios to a base scenario, a regional environmental policy scenario (each country has an upper limit on country emissions of 235) and a global environmental policy (regional emissions are unconstrained, but global emissions should be lower than 470).

Since the changes to be made to the model from the previous exercise are a bit tedious, and there are a few technical difficulties (in particular how to model both regional and global environmental policies), you may download the multi-country environmental CGE-model from the “GAMS for environmental economic modelling” website (see page 5).

Download the model “CGE_AUGMENTED.GMS” from the GAMS-homepage of the Environmental Economics Group of Wageningen University. Run the model.

Compare the multi-country model with the open-economy model. Can you explain the differences?

Check how the different scenarios are formulated (the equations themselves are changed!). If you want, you can add more scenarios to the model.

Interesting scenarios could include import taxes to simulate protectionist policies (protecting domestic production), export and production subsidies, and sensitivity analysis on the parameter values (*e.g.* on the pollution coefficients, on the technology parameters or on consumer preferences).

This basic environmental-economic CGE-model can be enhanced in several ways. For example, one could develop a dynamic version of the model, add pollution abatement (mitigation), add differential taxes on supply and demand, specify capital as mobile across countries, specify rationing on the labour market (minimum wages and unemployment), *et cetera*. A detailed description of how these extensions to the model can be specified is beyond the scope of this reader.

Exercise 4.5.6. The Dinwiddy and Teal model (optional)

The CGE-model that you will develop in this exercise uses a national accounting matrix to describe the initial situation. The formulation of the model itself is based on the model as described in Dinwiddy and Teal (1988)[7].

The accounting matrix, taken from the national accounts statistics, reads as follows:

	X1	X2	C	Endow	Total volume	Price	Total value
X1	0	0	0.824		0.824	1.942	1.6
X2	0	0	0.653		0.653	2.449	1.6
K	0.267	0.533		0.8	0.8	1.5	1.2
L	1.2	0.8		2	2	1	2
PRF	0	0			0		0
Total (value)	1.6	1.6	3.2	3.2			

where the column for 'Endow' gives the total labour and capital supply, as owned by the consumers.

You can try to start the model from scratch (a good way of testing your GAMS skills) or download the base Dinwiddy&Teal model without equations from the GAMS-homepage of the Environmental Economics Group of Wageningen University. If you start the model from scratch, define the table and the input parameters. In the table above, the Endow column is exogenously given, and the wages are fixed to 1.

The model should contain the following equations:

- the *utility function:* The utility function is given by $U = C1^{0.5} \cdot C2^{0.5}$, where C1 is the consumption of good 1 (0.824 as can be seen from the table).
- the *production functions*: The production function for the producers is of the Cobb-Douglas type ($X = L^{\alpha} \cdot K^{1-\alpha}$), where the share of labour in production costs (α) equals 0.75 for sector X1 and 0.5 for sector X2.
- all *market clearance conditions*:

commodity markets: $C_i = X_i$

factor markets: $K_1 + K_2 = K^*$ and $L_1 + L_2 = L^*$.

- a *profit function for both producers* (to be able to check that they equal zero):

$$PRF_i = P_i \cdot X_i - P_K \cdot K_i - P_L \cdot L_i$$

- *income balance* (include profits as part of household income):

$$\sum_i P_i \cdot X_i = \sum_i P_K \cdot K_i + \sum_i P_L \cdot L_i + \sum_i PRF_i$$

Specify the model and solve for maximum utility.

Add new lines to the end of the model where you free the value of `W` and fix `R` at 1. Give new starting values for `W`, `P1` and `P2` (use for example `W.L = W.L/1.5` and the same for the prices). Show that the relative prices resulting from this model are not different from the relative prices in the original model and that the quantities have not changed.

[7] C.L. Dinwiddy and F.J. Teal, 1988. 'The two-sector general equilibrium model: a new approach; Allan, Oxford.

As a final part of this exercise, we will introduce some basic environmental modelling:

Introduce two new variables that calculate the emissions from production and add two equations to the model to monitor the emissions. Assume fixed coefficients: for producer 1, assume emissions of 100 per unit of production; for producer 2, assume a coefficient of 50.

Introduce a scalar `ESUM` and calculate it as the sum of all emissions after each solve. What is the value of `ESUM`?

Note: the emissions are not a real part of the model, as they do not affect any of the other variables; the emissions are just *monitored* in the model; hence, there are no markets and equilibrium prices for the emissions.

Exercise 4.5.7. The basic CGE model in MPSGE (optional)

Students who are interested in learning the special GAMS subsystem MPSGE (which is specifically designed for CGE modelling) for their thesis can contact Xueqin Zhu. If you want to do CGE models in your thesis, MPSGE is highly recommended. Apart from the MPSGE-code for the current exercise, there are 8 small exemplary models in MPSGE-code available at the “GAMS for environmental economic modelling” website.

Download the model “CGE_MPSGE.GMS” from the GAMS-homepage of the Environmental Economics Group of Wageningen University.

Read the code and compare it carefully with the code of the model in Exercise 4.5.1. Try to see the resemblances.

It is probably necessary to read some more material on MPSGE before you can actually work with it. Some references are given on the “GAMS for environmental economic modelling” website.

4.6. Neo-classical growth modelling

Exercise 4.6.1. A basic growth model (a simplified DICE model)

The DICE model, which describes the interaction between economic growth and climate change, was originally specified by W. Nordhaus[8]. The DICE model itself is used in the next exercise, here we will construct a simplified version of the model.

Consider a description of an economy that can be characterised by the following rules:
1. Use positive variables when appropriate!
2. The model starts in 1980 and ends in 2010.
3. Specify economic activity (represented by variable `Y`) as a function that grows with 3% per year (use parameter `gy` as the annual growth rate); in the first year, economic activity is exactly 100.
4. Carbon emission, `sigma(T)`, per unit of economic activity equals 0.5 in 1980 and grows at the same speed as economic activity.
   {Hint: use `ORD(T)` to calculate the values of sigma for all periods.}
5. Total carbon emissions in period t, `E(T)`, is the product of `sigma` and `Y`, multiplied by `(1-MU(T))`, where `MU(T)` is the abatement ratio (or control rate).

[8] See W.D. Nordhaus, 1994, ‘Managing the global commons’, MIT Press, Cambridge.

6. Marginal abatement costs are increasing in abatement levels as follows: `AC(T)=0.5*MU(T)**2`.
7. Marginal damage costs are defined as `DC(T)=0.1*(sigma(T)*(1-MU(T)))**2`.
8. Consumption is determined in an equation as economic activity minus total abatement costs (calculated as marginal abatement costs times economic activity) and minus total damage costs (calculated as marginal damage costs times economic activity).
9. The economy maximises total utility, summed over all periods, where utility in period t is the present value of consumption, discounted with a discount rate of 4%. {Hint: the present value of consumption in the second period equals `C("2")/1.04`.}
10. Note that there is no equation to determine `MU(T)`. This free variable is calculated by GAMS. Fix `MU(T)` at 0 for 1980 and give a lower bound of 0.001 and an upper bound of 0.999 for all other periods.

Write down the model in GAMS notation, run the model, if necessary debug any errors and give an interpretation of the results.

Exercise 4.6.2. An extended growth model (the DICE model)

Download the model "AD-DICE.GMS" from the BlackBoard site[9]. Run the model.

- What control variables does this model have? How are they denoted?
- Explain the difference between the variables Y and Q.
- Explain the logic behind the mitigation scenarios.
- Run the mitigation-adaptation scenarios: Which control is more "important" in this model ? Can mitigation(adaptation) compensate for low levels of adaptation(mitigation)?
- Change the Social rate of time preference (B_PRSTP) to 0.1. What are the effects on mitigation? And adaptation? Can you explain these effects?

[9] The original code for the DICE model can be found at the homepage of W. Nordhaus: http://www.econ.yale.edu/~nordhaus/homepage/dicemodels.htm

APPENDIX I: COMMON ERROR MESSAGES IN GAMS

If your model does not solve properly, there are several things you can do. First step is to carefully read the error messages that GAMS provides. These should give some clues to what the problem is and where it is located in your source file. The second step is to carefully review your code. And again. Unfortunately, error messages in GAMS can sometimes be a bit cryptic. Therefore this section gives some hints that may help you find out what the most common error messages in GAMS mean.

Error code 129

You model does not run. GAMS gives an error code 129 and stops completely. This error is caused by a very specific problem with Windows NT on the Wageningen UR-network and GAMS. You are trying to run a model that is in a project that is saved on the W-disk. This is not possible. Save your model, open a new project (go to file, project, new project) on the C-disk (for example: c:\mydocuments\models\gams.gpr) and open your gams file again.

Exit code 109

Error: too many process directories exist, remove or rename directory(’s) 225?
Gams gives exit code 109 and stops completely. Temporary files (stored in subdirectories 225a, 225b, *et cetera*) have not been deleted automatically and you have to do it manually. Go to the directory where your project file is stored and delete all directories that start with 225.

Error 002 Identifier expected

This error message means that you are using some gams code where GAMS expects you to use a name. There are two possible causes for this error message:

- You can get these error messages when you use a name for a set, parameter, variable or equation, that also has a meaning in gams code. For example you want to name one of your variables “system”. System however is a specified GAMS code so you will get this error message. In GAMS-IDE it is easy to check if a word is a specified gams-code or not. Whenever a word turns blue, this means that the word is GAMS code, do not use this word as a name.
- You can get this error message when you do not give a name when GAMS expects you to. For example if you type the word parameter, GAMS expects you to at least declare one parameter. If you do not declare a parameter you get this error message.

Error 036 '=' or '..' or ':=' or '$=' operator expected
-or-
Error 037 '=l=' or '=e=' or '=g=' operator expected

Remember when giving a value to a parameter you use “=” For example:

```
Parameter   INCOME;
INCOME=100;
```

When writing an equation you have to specify whether the lefthandside is equal to (`=E=`), greater than (`=G=`) or less than (`=L=`) the righthandside.

Error 066/141 The symbol shown has not been defined or assigned. A wild shot: You may have spurious commas in the explanatory text of a declaration. Check symbol reference list.

This error message means you are using a parameter in an equation that has not been given a value. For example suppose you are using equation in which you say that the use of capital (variable) must be lower then the total stock of capital (parameter). If you do not give a value to the parameter stock capital then GAMS will not be able to calculate the value of the variable capital and will give an error message.

Check the listing file for information about which parameter does not have a value.

Error 071 The symbol shown has been declared as an equation, but no symbolic equation (..) was found. Hint - look for commas in the documentation text for the equations. Use quotes around the text or eliminate the commas.

This error message means that you have declared an equation but that you have not used this equation. Delete the declared equation that you are not using.

Error 096 Blank needed between identifier and text
(-or- illegal character in identifier)
(-or- check for missing ';' on previous line)

This error code can mean three things.

1. Most commonly it will mean that you forgot a semicolon (;). Check the line just above the error message to make sure that there is a semicolon there.
2. It can also mean that you have no space between the name of a set, parameter, variable or equation and the explanatory text.

   For example, the following example is correct:

   ```
   VARIABLES
      Y     income;
   ```

   However the following example will give an error code:

   ```
   VARIABLES
      Yincome;
   ```

3. Finally this error message can mean that you have used some illegal characters in names for sets, parameters, variables and equations. Names can only contain letters or numbers. Each name should at least contain 1 letter. Do not use names longer then 10 characters (though later versions of GAMS can handle longer names than earlier versions). Tip: try to give names that refer to the meaning of the variable (or parameter); it makes the reading of the model code much easier.

   Examples of legal names are: a, a15, revenue

   Examples of illegal names are: 15, $casg, muchtoolong, milk&meat

Error 140 Unknown symbol

This error code means that you are using a set, parameter, variables or equation without declaring it first. You always have to define a name first before you can use it. Check if you have defined the name. If you did define the name, check if you have made any writing errors when using the name.

Error 148 Dimension different - The symbol is referenced with more/less indices as declared

You are using a parameter or a variable differently then you have declared it. If you have a variable with an index, you have to always use the index when using the variable. For example if you declare a certain variable Y as Y(t) which means that variable Y has the index t than every time you use Y you have the write Y(t).

So the following example is correct:

```
Variable    Y(t)  income;
Equation    QI(t) income equation;
QI(t)..     Y(t)=E=P*X;
```

The following example is not correct:

```
Variable    Y(t)  income;
Equation    QI    income equation;
QI..        Y=E=P*X;
```

Error 142 No suffix allowed here - suffix ignored

When you get this error message you are probably using a suffix after the name of a parameter. Suffix’s are: .L (level), .UP (upper bound) .LO (lower bound) .M (marginal value). If you want to display the value of a parameter just write the name:

```
DISPLAY Y;
```

instead of :

```
DISPLAY Y.L;
```

if Y is a parameter

Error 143 A suffix is missing

When referring to a variable outside an equation you always have to put a suffix after the variable name. Suffix’s are: .L (level), .UP (upper bound) .LO (lower bound) .M (marginal value). So if you want to display the value of a variable, remember to put a suffix after the variables name:

```
DISPLAY Y.L;
```

instead of :

```
DISPLAY Y;
```

if Y is a variable

Error 149 Uncontrolled set entered as constant

This error message means that the index of the equation name is not equal to the index of variables used in that equation.

Take for example the following equation:

```
QY..       Y(t)=E=P*X(t);
```

Income equals the value of consumption in all time periods t. However you have to specify to GAMS for which time period t this equation holds. If you want this equation to hold for all time period t, you have to add an index to the equation name:

```
QY(t)..    Y(t)=E=P*X(t);
```

Error 195 Symbol redefined with a different type

This error message means that you are using the same name for two different identifiers. Never use the same name twice. Not even equations can have the same names! Remember that GAMS does not see the difference between lower and upper case letters. For example "y" and "Y" are the same names for GAMS.

Error 409 Unrecognizable item - skip to find a new statement, looking for a ';' or a key word to get started again

This is a very general error message; there are several ways of getting this error message:

- You have made a writing mistake in one of the gams codes. GAMS expects a gams code and does not see one and gives this error message.
- You have used an index in the display statement. You can never use an index in the display statement.

Exec Error 0 Division by zero

If you get execution error 0 then GAMS tries to divide by zero. A solution to this problem is to give a small lower bound (e.g. 0.00001) for any variable that you use in the lower part of a division.

Sometimes you can get around this error by restructuring the equation (`A*B=E=C` instead of `A=E=C/B`).

Exec Error 10 Illegal arguments in ** operation

You can use the "to the power to" operation in GAMS but this can easily cause problems. Always use small lower bound (e.g. 0.00001) for variables you use in ** operations (if you are sure that they will have positive values anyway). GAMS sometimes has problems with performing a ** operation with negative numbers.

Rearranging you equations may also help. For example, if `(A-B)**2 =E= D;` does not work, write two equations: `A-B =E= C;` and `C**2 =E= D;`.

**** Warning ** The variance of the derivatives in the initial point is large. A better initial point, a better scaling, or better bounds on the variables will probably help the optimization.**

This is actually not an error message, but a warning. Basically, it says that your model is not well-behaved. Your model may still solve properly, but chances are high that the solver fails to find the optimal solution (and you do know how to check whether GAMS found the optimal solution, right!?).

The warning is quite informative and lists different options you have to improve the working of your model: provide better starting values, lower and upper bounds and scaling. You can find exercises on how to use these features in Section 3.2. of this reader.

In general, when you get warnings like this, it is a good point to start with evaluating your model formulation. After carefully reviewing your code, re-ordering the code in a more logical way, and avoiding unnecessary complications in equations, you can move to the next step: carefully review your code again. You are bound to find things you missed the first time.

The warning may, however, also affect your choice of solver. For most model types, there are different solvers you can use to find the optimal solution. In many instances, especially for smaller models, all major solvers do a good (or even excellent) job in finding the optimum quickly. You can check for which solvers you have a full license in the options menu of GAMS-IDE.

One of the leading experts on this issue is Erwin Kalvelagen of GAMS Development Corporation. His advice on this issue is, as always, excellent:

"In general, CONOPT is one of the most reliable solvers for very large [non-linear, RD] models, but it is always possible another solver does better, so trying […] is certainly a good idea. Here are some more hints:

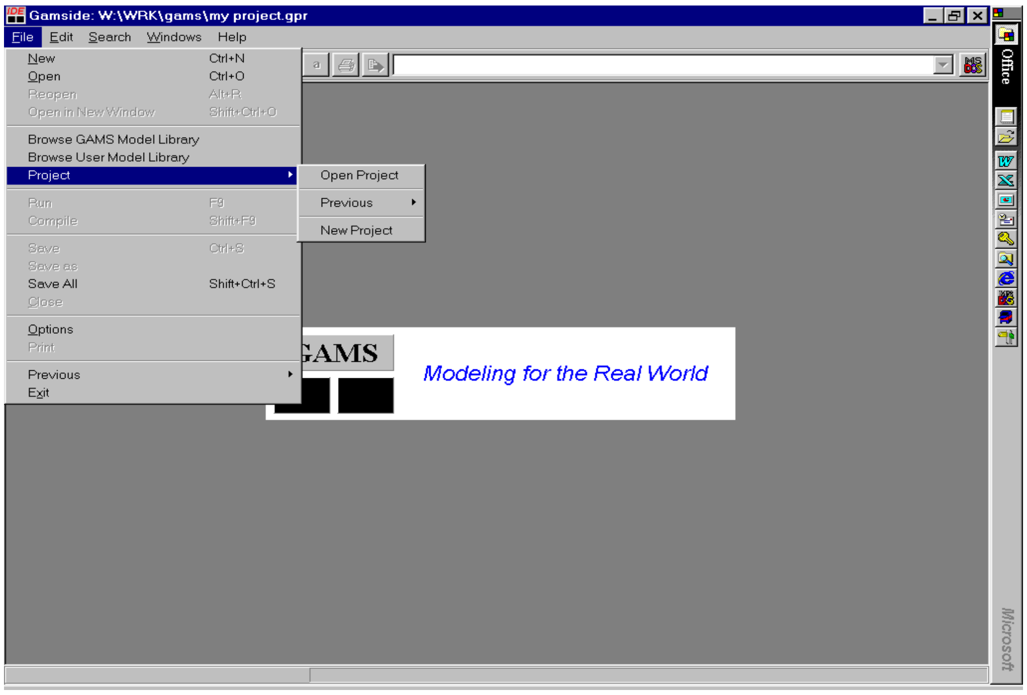
- use a smaller instance to provide a starting point for a large instance.
- use smart aggregation […] to reduce the size, solve that model and use this as a starting point for the disaggregated version.
- use the point that the default solver gives you as a starting point for another solver.
- even if other solvers cannot solve the model their behaviour and messages may give some insight in the origin of the problem.
- look at the model formulation.
- look again at the model formulation."

APPENDIX II: SOLUTIONS TO THE EXERCISES

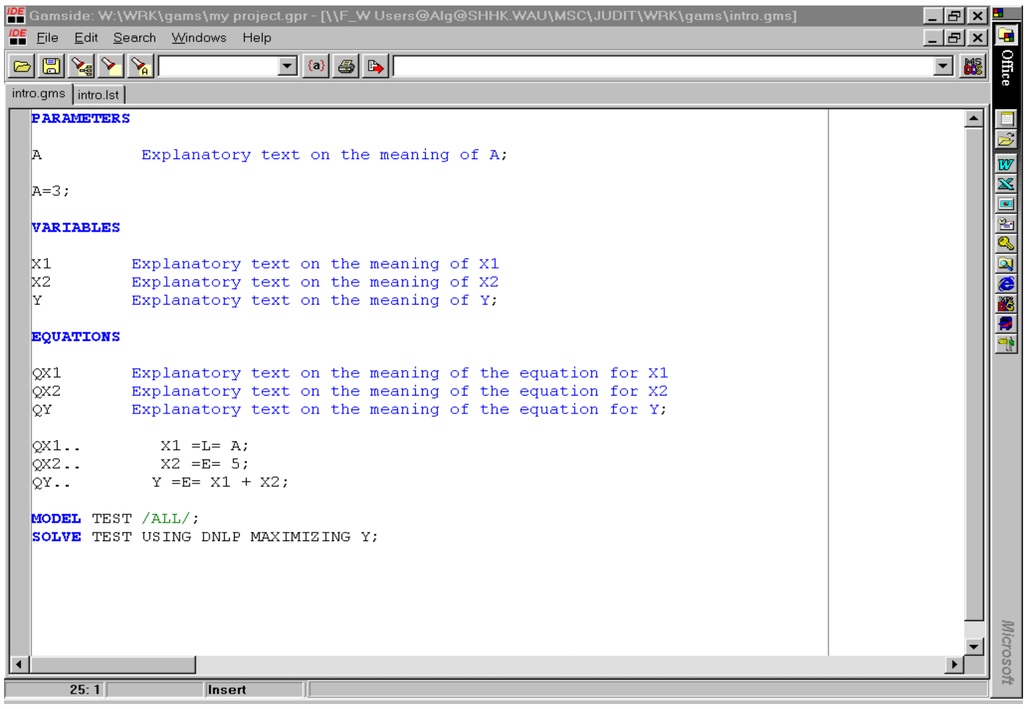
II.1. Results for Introductory level

Results from Exercise 3.1.1. Starting GAMS-IDE

Your desktop should look similar to this:



Results from Exercise 3.1.2. The basic blocks of writing a model in the IDE

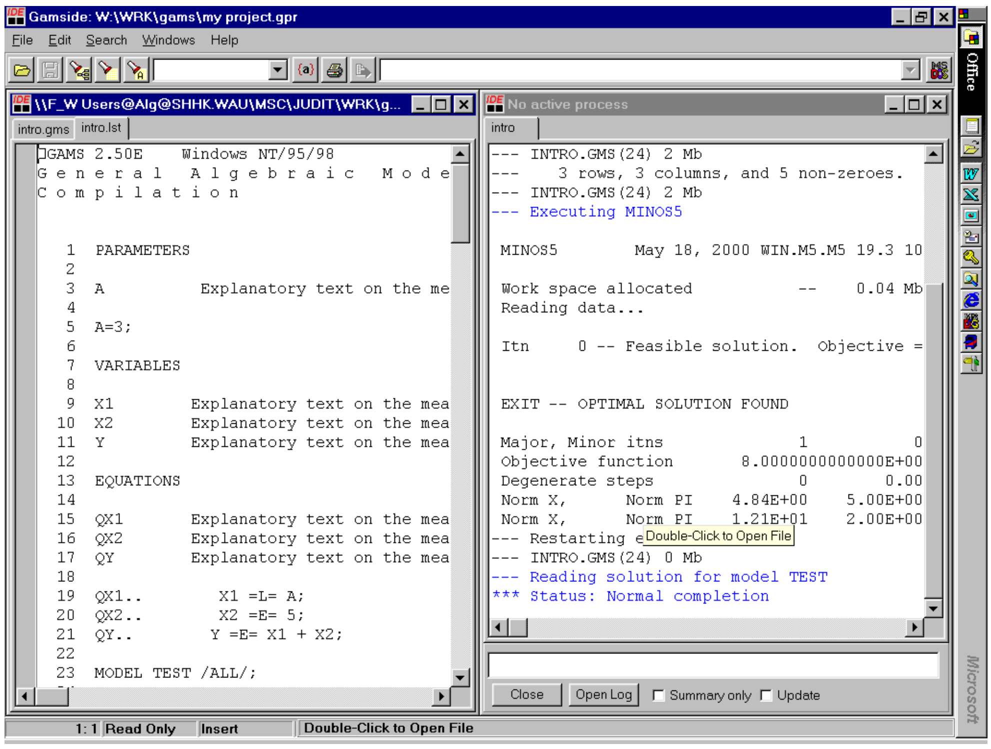


Gamside: W:\WRK\gams\my project.gpr - [\\F_W Users@Alg@SHHK.WAU\MSC\JUDIT\WRK\gams\intro.gms]
File Edit Search Windows Help
intro.gms
intro.lst
25: 1
Insert
Office
Microsoft

```
PARAMETERS

A          Explanatory text on the meaning of A;

A=3;

VARIABLES

X1        Explanatory text on the meaning of X1
X2        Explanatory text on the meaning of X2
Y         Explanatory text on the meaning of Y;

EQUATIONS

QX1       Explanatory text on the meaning of the equation for X1
QX2       Explanatory text on the meaning of the equation for X2
QY        Explanatory text on the meaning of the equation for Y;

QX1..        X1 =L= A;
QX2..        X2 =E= 5;
QY..        Y =E= X1 + X2;

MODEL TEST /ALL/;
SOLVE TEST USING DNLP MAXIMIZING Y;
```

Results from Exercise 3.1.3. Reading the process window

After running the model, the GAMS-IDE should look similar to this[10]:


Gamside: W:\WRK\gams\my project.gpr
File Edit Search Windows Help
\\F_W Users@Alg@SHHK.WAU\MSC\JUDIT\WRK\g...
intro.gms
intro.lst
No active process
intro
Close
Open Log
Summary only
Update
1: 1
Read Only
Insert
Double-Click to Open File
Office
Microsoft

```
GAMS 2.50E    Windows NT/95/98
G e n e r a l   A l g e b r a i c   M o d e
C o m p i l a t i o n


   1  PARAMETERS
   2
   3  A           Explanatory text on the me
   4
   5  A=3;
   6
   7  VARIABLES
   8
   9  X1          Explanatory text on the mea
  10  X2          Explanatory text on the mea
  11  Y           Explanatory text on the mea
  12
  13  EQUATIONS
  14
  15  QX1         Explanatory text on the mea
  16  QX2         Explanatory text on the mea
  17  QY          Explanatory text on the mea
  18
  19  QX1..          X1 =L= A;
  20  QX2..          X2 =E= 5;
  21  QY..          Y =E= X1 + X2;
  22
  23  MODEL TEST /ALL/;
```

```
--- INTRO.GMS(24) 2 Mb
---    3 rows, 3 columns, and 5 non-zeroes.
--- INTRO.GMS(24) 2 Mb
--- Executing MINOS5

 MINOS5       May 18, 2000 WIN.M5.M5 19.3 10

 Work space allocated           --    0.04 Mb
 Reading data...

 Itn     0 -- Feasible solution.  Objective =


 EXIT -- OPTIMAL SOLUTION FOUND


 Major, Minor itns              1           0
 Objective function      8.0000000000000E+00
 Degenerate steps               0        0.00
 Norm X,       Norm PI    4.84E+00    5.00E+00
 Norm X,       Norm PI    1.21E+01    2.00E+00
--- Restarting e
--- INTRO.GMS(24) 0 Mb
--- Reading solution for model TEST
*** Status: Normal completion
```

Results from Exercise 3.1.4. Reading the listing file

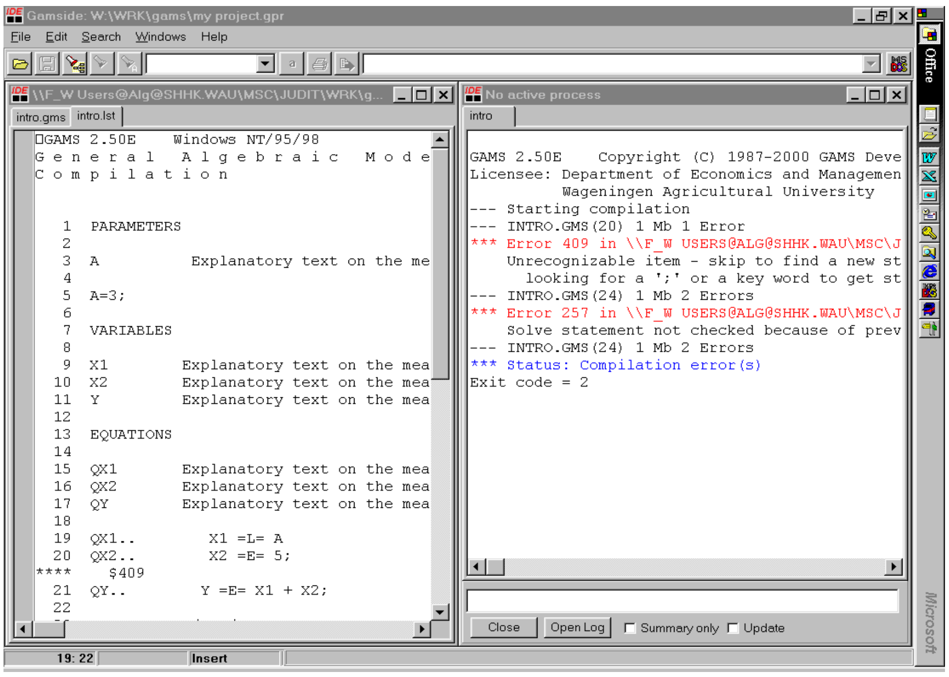
In our simple example, the report on the variables should be as follows :

```
                  LOWER            LEVEL         UPPER          MARGINAL
---- VAR X1       -INF             3.0000        +INF              .
---- VAR X2       -INF             5.0000        +INF              .
---- VAR Y        -INF             8.0000        +INF              .
  X1          Explanatory text on the meaning of X1
  X2          Explanatory text on the meaning of X2
  Y           Explanatory text on the meaning of Y
```

Therefore, the optimal values for variables `X1`, `X2` and `Y` are `3`, `5` and `8` respectively. Note that none of the variables has a lower or upper bound. INF means infinity, so the variables can take infinitely large and small values. The marginal value of all variables is zero, which confirms that the solution found is optimal.

[10] If you use a different solver than Minos, the process window will look different, but the same essential information is displayed.

Results from Exercise 3.1.5. Debugging a compilation error


Gamside: W:\WRK\gams\my project.gpr
File Edit Search Windows Help
\\F_W Users@Alg@SHHK.WAU\MSC\JUDIT\WRK\g...
intro.gms intro.lst
No active process
intro
Close Open Log Summary only Update
19: 22 Insert
Office
Microsoft

```
GAMS 2.50E    Windows NT/95/98
G e n e r a l   A l g e b r a i c   M o d e
C o m p i l a t i o n


   1  PARAMETERS
   2
   3  A            Explanatory text on the me
   4
   5  A=3;
   6
   7  VARIABLES
   8
   9  X1          Explanatory text on the mea
  10  X2          Explanatory text on the mea
  11  Y           Explanatory text on the mea
  12
  13  EQUATIONS
  14
  15  QX1         Explanatory text on the mea
  16  QX2         Explanatory text on the mea
  17  QY          Explanatory text on the mea
  18
  19  QX1..          X1 =L= A
  20  QX2..          X2 =E= 5;
****    $409
  21  QY..          Y =E= X1 + X2;
  22
```

```
GAMS 2.50E    Copyright (C) 1987-2000 GAMS Deve
Licensee: Department of Economics and Managemen
          Wageningen Agricultural University
--- Starting compilation
--- INTRO.GMS(20) 1 Mb 1 Error
*** Error 409 in \\F_W USERS@ALG@SHHK.WAU\MSC\J
    Unrecognizable item - skip to find a new st
      looking for a ';' or a key word to get st
--- INTRO.GMS(24) 1 Mb 2 Errors
*** Error 257 in \\F_W USERS@ALG@SHHK.WAU\MSC\J
    Solve statement not checked because of prev
--- INTRO.GMS(24) 1 Mb 2 Errors
*** Status: Compilation error(s)
Exit code = 2
```

Removing the semi-colon from equation `QX1` results in the following window:
At the end of the listing file the list of errors are explained:

`“409 Unrecognizable item - skip to find a new statement; looking for a ';' or a key word to get started again”` is displayed.

Results from Exercise 3.1.6. Debugging a logical error

The problem is `UNBOUNDED`, because you can keep lowering the value of `Y` by lowering `X1`. Remember that `X1` should be less than `A`, not equal to. So `X1` can get any value below `3`. Consequently, `Y` can get any value below `8`. The minimum is then minus infinity; GAMS reports this as unbounded.

The listing file gives the following solve summary:

```
**** SOLVER STATUS      1 NORMAL COMPLETION
**** MODEL STATUS       3 UNBOUNDED
```

Results from Exercise 3.1.7. Building a new model

The results for the variables when the tax rate (`tax`) equals 30 is:

```
              LOWER          LEVEL          UPPER         MARGINAL
VAR DAMAGE    -INF          6000.0000       +INF             .
VAR ST_RATE   -INF             0.5000       +INF             .
VAR CATTLE    -INF           500.0000       +INF             .
VAR OBJ       -INF          4000.0000       +INF             .
```

For `tax=20`, `obj` equals `6000`; for `tax=10`, `obj` equals `6000` and for `tax`=0, `obj` equals 4000; the optimal value of `tax` is 15, when `obj` equals `6250`.

Make sure that you use the explanatory text.

Results from Exercise 3.1.8. Understanding what you have done

The values of `X1` and `Y` should be identical to the earlier exercises: `3` and `8`, respectively.

Did you think of removing the equation declaration and the equation definition for `QX2`?

The full model for emissions can look like this:

```
PARAMETERS
      coef  Emission coefficient CO2
      OTHER Emissions of other greenhouse gasses;
      Coef=0.03;
      OTHER=5;
VARIABLES

      CO2   Emissions of CO2
      PRD   Production quantity
      EMIS  Total emissions of greenhouse gasses;
EQUATIONS

      QPRD  Equation for economic production
      QCO2  Equation for CO2 emissions
      QEMIS Equation for total climate emissions;

QPRD..      PRD =G= 100;
QCO2..      CO2 =E= coef*PRD;
QEMIS..     EMIS =E= CO2+OTHER;

MODEL CLIMATE /ALL/;
SOLVE CLIMATE USING DNLP MINIMIZING EMIS;


With solution:
                        LOWER     LEVEL     UPPER    MARGINAL
---- VAR EMIS           -INF      8.000     +INF       .
---- VAR CO2            -INF      3.000     +INF       .
---- VAR PRD            -INF    100.000     +INF       .
```

II.2. Results for Intermediate level

Results from Exercise 3.2.1. DISPLAY your results

At the end of the listing file, the following lines have been added by GAMS:

```
 VARIABLE  EMIS.L  = 8.000 Total emissions of greenhouse gasses
 PARAMETER OTHER   = 5.000 Emissions of other greenhouse gasses
```

Results from Exercise 3.2.2. Commenting out a single line

The line with the comment does not change the results.

Results from Exercise 3.2.3. Using SCALARS

```
SCALARS
coef    emission coefficient   /0.03/
OTHER   other emissions  /5/;
```

The result does not change.

Results from Exercise 3.2.4. Solving more than one model

As the value of `OTHER` has increased from 5 to 7, we expect that the value of `RES2` equals `RES1`+2. The display statement gives:

```
     PARAMETER RES1                 =        8.000
     PARAMETER RES2                 =       10.000
```

The other values in the model have not changed (`PRD` and CO2 are unaffected).

Remember to check *for each solve* whether GAMS has found an optimal solution!

Results from Exercise 3.2.5. Using multidimensional PARAMETERS

The display statement changes to:

```
----     31 PARAMETER RES
1st solve  8.000,    2nd solve 10.000
```

Results from Exercise 3.2.6. SETS and vector specification

In the solution listing, the solution for `CO2` is not influenced, but `PRD` is changed:

```
              LOWER          LEVEL         UPPER         MARGINAL
---- VAR CO2  -INF           3.0000        +INF              .
  CO2         Emissions of CO2

---- VAR PRD         Production quantity
         LOWER          LEVEL         UPPER         MARGINAL
1        -INF           10.0000       +INF              .
2        -INF           50.0000       +INF              .
3        -INF           40.0000       +INF              .
```

Results from Exercise 3.2.7. Summing over an index

No changes in the results.

Results from Exercise 3.2.8. Including time: a dynamic specification

For each year 2000 to 2004 the same values result in the model:

```
----     45 PARAMETER RES
                  2000        2001        2002        2003        2004
1st solve        8.000       8.000       8.000       8.000       8.000
2nd solve       10.000      10.000      10.000      10.000      10.000
```

The new variable `TOTEMIS` equals 40 in the first solve and 50 in the second solve.

Results from Exercise 3.2.9. Using TABLES for data input

The values of variable `PRD` changes, as does the parameter `RES` (and for instance also `CO2` and `CONCENT`).

```
---- VAR PRD         Production quantity
              LOWER          LEVEL          UPPER         MARGINAL
1.2000        -INF           10.0000        +INF             .
1.2001        -INF           11.0000        +INF             .
1.2002        -INF           12.0000        +INF             .
1.2003        -INF           13.0000        +INF             .
1.2004        -INF           14.0000        +INF             .
2.2000        -INF           50.0000        +INF             .
2.2001        -INF           52.0000        +INF             .
2.2002        -INF           54.0000        +INF             .
2.2003        -INF           56.0000        +INF             .
2.2004        -INF           58.0000        +INF             .
3.2000        -INF           40.0000        +INF             .
3.2001        -INF           42.0000        +INF             .
3.2002        -INF           44.0000        +INF             .
3.2003        -INF           46.0000        +INF             .
3.2004        -INF           48.0000        +INF             .

----     55 PARAMETER RES
                 2000        2001        2002        2003        2004
1st solve       8.000       8.150       8.300       8.450       8.600
2nd solve      10.000      10.150      10.300      10.450      10.600
```

Results from Exercise 3.2.10. Defining POSITIVE VARIABLES

The model results do not change, but you can see the difference in the solution for `PRD`:

```
---- VAR PRD         Production quantity
              LOWER          LEVEL          UPPER         MARGINAL
1.2000          .            10.0000        +INF             .
1.2001          .            11.0000        +INF             .
1.2002          .            12.0000        +INF             .
1.2003          .            13.0000        +INF             .
1.2004          .            14.0000        +INF             .
2.2000          .            50.0000        +INF             .
2.2001          .            52.0000        +INF             .
2.2002          .            54.0000        +INF             .
2.2003          .            56.0000        +INF             .
2.2004          .            58.0000        +INF             .
3.2000          .            40.0000        +INF             .
3.2001          .            42.0000        +INF             .
3.2002          .            44.0000        +INF             .
3.2003          .            46.0000        +INF             .
3.2004          .            48.0000        +INF             .
```

The lower bounds are no longer "`-INF`", but a dot is given. This dot indicates zero.

Results from Exercise 3.2.11. Providing starting values

If your model behaves normally, the starting values do not influence the results. So in this simple example, nothing changes.

Results from Exercise 3.2.12. Providing lower and upper bounds

Note that the lower and upper bound go on `PRD` (the variable) and not on `PRD_DATA` (the parameter); you cannot put bounds on parameters, as their values are already fixed.

If you put the upper value on `PRD("2",T)` at a level below the value in `PRD_DATA("2",T)`, then the model becomes infeasible. The model does not make any sense anymore: the value cannot be smaller than 55 and larger than `PRD_DATA("2",T)` at the same time. So the upper bound on `PRD("2",T)` should be removed.

The solution listing for `PRD` for the first simulation contains:

```
---- VAR PRD         Production quantity
              LOWER          LEVEL          UPPER         MARGINAL
1.2000        12.0000        12.0000        +INF            0.0300
1.2001        12.0000        12.0000        +INF            0.0300
1.2002        12.0000        12.0000        +INF              .
1.2003        12.0000        13.0000        +INF              .
1.2004        12.0000        14.0000        +INF              .
2.2000          .            50.0000        +INF              .
2.2001          .            52.0000        +INF              .
2.2002          .            54.0000        +INF              .
2.2003          .            56.0000        +INF              .
2.2004          .            58.0000        +INF              .
3.2000          .            40.0000        +INF              .
3.2001          .            42.0000        +INF              .
3.2002          .            44.0000        +INF              .
3.2003          .            46.0000        +INF              .
3.2004          .            48.0000        +INF              .
```

And the display statement at the end delivers:

```
----     63 PARAMETER RES
                2000         2001         2002         2003         2004
1st solve      8.060        8.180        8.300        8.450        8.600
2nd solve     10.060       10.180       10.300       10.450       10.600
```

So the lower bound on `PRD("1",T)` affects the results for 2000 and 2001. Annual emissions, as reported in parameter `RES` are 0.06 and 0.03 higher than without the bounds.

Results from Exercise 3.2.13. Fixing variables

Fixing the level of `PRD("3","2000")` to 45 in the first solve will affect production in 2000 and consequently also emissions (and concentration).

```
---- VAR PRD         Production quantity
              LOWER          LEVEL          UPPER         MARGINAL
1.2000        12.0000        12.0000        +INF            0.0300
1.2001        12.0000        12.0000        +INF            0.0300
1.2002        12.0000        12.0000        +INF              .
1.2003        12.0000        13.0000        +INF              .
1.2004        12.0000        14.0000        +INF              .
2.2000          .            50.0000        +INF              .
```

```
2.2001          .           52.0000         +INF              .
2.2002          .           54.0000         +INF              .
2.2003          .           56.0000         +INF              .
2.2004          .           58.0000         +INF              .
3.2000       45.0000        45.0000       45.0000          0.0300
3.2001          .           42.0000         +INF              .
3.2002          .           44.0000         +INF              .
3.2003          .           46.0000         +INF              .
3.2004          .           48.0000         +INF              .

----     66 PARAMETER RES
                2000        2001        2002        2003        2004
1st solve      8.210       8.180       8.300       8.450       8.600
2nd solve     10.060      10.180      10.300      10.450      10.600
```

Results from Exercise 3.2.14. Using the LOOP statement

Nothing changes, you just wrote the code in a different way.

Results from Exercise 3.2.15. Using conditional statements ($-operations)

From the solution listing of the first solve, it shows:

```
---- VAR DAM Damages from climate change in terms of production losses
              LOWER          LEVEL          UPPER         MARGINAL
1.2000        -INF            0.0500        +INF              .
1.2001        -INF            0.0500        +INF              .
1.2002        -INF            0.0500        +INF              .
1.2003        -INF            0.0500        +INF              .
1.2004        -INF            0.0500        +INF              .
2.2000        -INF            0.0500        +INF              .
2.2001        -INF            0.0500        +INF              .
2.2002        -INF            0.0500        +INF              .
2.2003        -INF            0.0500        +INF              .
2.2004        -INF            0.0500        +INF              .
3.2000        -INF            0.0500        +INF              .
3.2001        -INF            0.0500        +INF              .
3.2002        -INF            0.0500        +INF              .
3.2003        -INF            0.0500        +INF              .
3.2004        -INF            0.0500        +INF              .
```

The display statement shows:

```
----     68 PARAMETER RES
                2000        2001        2002        2003        2004
1st solve      8.135       8.039       8.153       8.277       8.420
2nd solve      9.925      10.039      10.153      10.277      10.420
```

As a result of the damages, production goes down; this will lead to lower emissions and concentrations. Emissions are not exactly 5 percent lower than in the previous exercise, as the bounds on the variables also influence the results.

Using the conditional statement on the damage function, damages will be:

```
---- VAR DAM Damages from climate change in terms of production losses
              LOWER          LEVEL          UPPER         MARGINAL
1.2000        -INF            0.0500        +INF              .
1.2001        -INF            0.0500        +INF              .
1.2002        -INF            0.0500        +INF              .
1.2003        -INF            0.0500        +INF              .
```

```
1.2004        -INF            0.0500        +INF              .
2.2000        -INF            0.0500        +INF              .
2.2001        -INF            0.1000        +INF              .
2.2002        -INF            0.1000        +INF              .
2.2003        -INF            0.1000        +INF              .
2.2004        -INF            0.1000        +INF              .
3.2000        -INF            0.0500        +INF              .
3.2001        -INF            0.0500        +INF              .
3.2002        -INF            0.0500        +INF              .
3.2003        -INF            0.0500        +INF              .
3.2004        -INF            0.0500        +INF              .

----     68 PARAMETER RES
                2000        2001        2002        2003        2004
1st solve      8.135       7.961       8.072       8.193       8.333
2nd solve      9.925       9.961      10.072      10.193      10.333
```

The higher damages will strengthen the effects on emissions.

Results from Exercise 3.2.16. Using lags and leads in parameters and variables

Sector-specific emission coefficients:

```
----     71 PARAMETER RES
                2000        2001        2002        2003        2004
1st solve      8.145       7.960       8.047       8.136       8.229
2nd solve      9.970       9.960      10.047      10.136      10.229
```

Remember that the sector specific coef is also part of the equation `CO2`. `COEF` should be moved inside the `SUM`.

Changing the conditional statement for damages leads to:

```
---- VAR DAM Damages from climate change in terms of production losses
              LOWER          LEVEL          UPPER         MARGINAL
1.2000        -INF            0.0500        +INF              .
1.2001        -INF            0.0500        +INF              .
1.2002        -INF            0.0500        +INF              .
1.2003        -INF            0.0500        +INF              .
1.2004        -INF            0.0500        +INF              .
2.2000        -INF            0.0500        +INF              .
2.2001        -INF            0.0500        +INF              .
2.2002        -INF            0.1000        +INF              .
2.2003        -INF            0.1000        +INF              .
2.2004        -INF            0.1000        +INF              .
3.2000        -INF            0.0500        +INF              .
3.2001        -INF            0.0500        +INF              .
3.2002        -INF            0.0500        +INF              .
3.2003        -INF            0.0500        +INF              .
3.2004        -INF            0.0500        +INF              .

----     71 PARAMETER RES
                2000        2001        2002        2003        2004
1st solve      8.145       8.063       8.047       8.136       8.229
2nd solve      9.970      10.063      10.047      10.136      10.229
```

Results from Exercise 3.2.17. Raising to a power

The model status:

```
L O O P S         SOL        1st solve
               S O L V E      S U M M A R Y
     MODEL   CLIMATE             OBJECTIVE  CONCENT
     TYPE    DNLP                DIRECTION  MINIMIZE
     SOLVER  MINOS5              FROM LINE  62

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE               39.6216

 RESOURCE USAGE, LIMIT          0.109     1000.000
 ITERATION COUNT, LIMIT        15        10000
 EVALUATION ERRORS              0            0
```

The results:

```
----     71 PARAMETER RES
                 2000        2001        2002        2003        2004
1st solve       8.050       7.915       7.805       7.884       7.967
2nd solve       9.827       9.915       9.805       9.884       9.967
```

Results from Exercise 3.2.18. Using ORD and CARD

```
----     28 PARAMETER INDEX         Straight line from zero to one
2001 0.250,    2002 0.500,    2003 0.750,    2004 1.000
```

Note that the solution listing does not display zero values; hence `INDEX("2000")` is not displayed.

Results from Exercise 3.2.19. Using a free variable

In all cases, the optimal value for `alpha` is 0.8, which is equal to the upper bound. This means that GAMS gives the lowest possible weight to `OTHER` emissions. As these exceed the values of `CO2` emissions, this was to be expected: GAMS tries to find the lowest combined emissions and adjusts the value of `alpha` to that purpose.

Solving Exercise 3.1.7. with free variable:

```
LOWER     LEVEL     UPPER    MARGINAL
---- VAR TAX            -INF      15.000     +INF       .
---- VAR DAMAGE         -INF  37500.000     +INF       .
---- VAR ST_RATE        -INF       1.250     +INF       .
---- VAR CATTLE         -INF   1250.000     +INF       EPS
---- VAR OBJ            -INF   6250.000     +INF       .
```

The optimal tax rate is 15.

Results from Exercise 3.2.20. Mapping set elements

```
----  63 PARAMETER check  Auxiliary parameter to check matching totals
( ALL       0.000 )
```

```
----  63 PARAMETER smalldataset  The aggregated dataset (usable input)
                  agri    industry    services        cons     Col_tot
agri             5.000                              45.000      50.000
industry        15.000     190.000      30.000     115.000     350.000
services        10.000      40.000     370.000     280.000     700.000
primary_in      20.000     120.000     300.000                 440.000
Row_tot         50.000     350.000     700.000     440.000
```

Results from Exercise 3.2.21. Exporting your results to Excel (optional)

The Excel file contains the following data:

```
                   2000        2001        2002        2003        2004
1st solve          8.05    7.915377    7.805389    7.883995    7.966849
2nd solve        9.8275    9.915377    9.805389    9.883995    9.966849
```

Results from Exercise 3.2.22. Understanding what you have done

Did you remember to change the set `SOL`? The command `OTHER(T)=7` in the loop should be changed to `OTHER(T)=OTHER(T)+2`.

```
----     70 PARAMETER RES
                  2000        2001        2002        2003        2004
1st solve        8.050       7.915       7.805       7.884       7.967
2nd solve        9.827       9.915       9.805       9.884       9.967
3rd solve       11.827      11.915      11.805      11.884      11.967
```

The model focuses on the build-up of greenhouse gasses stemming from production. One could use the model to analyse the relationship between production quantities per sector and the building-up of greenhouse gas emissions. The inclusion of damages makes this model more realistic, as higher production values lead to higher damages, which in turn will destroy some produced goods. The relationship between damages and concentrations is that damages reduce production and hence reduce emissions of CO2. Consequently, concentrations of greenhouse gasses also go down. But there are a few problems: in this model, production plays only a negative role as it causes emissions, but in reality, production will increase consumption and hence utility and is thus a ‘good’ thing. Then, if production is ‘good’, damages are ‘bad’, while they are only ‘good’ in this model. But either good or bad, damages will decrease emissions.

You can make the model more realistic by including emission reductions through abatement, by including and maximising consumption subject to a present concentration maximum, *et cetera*.

II.3. Results for Advanced level

Results from Exercise 3.3.1. Specifying a dynamic optimisation model

The optimal path of investment:

```
----     51 VARIABLE  I.L      investment

2003 1.512,    2004 1.931,    2005 1.970,    2006 2.009,
2007 2.049,    2008 2.090     2009 2.132,    2010 2.175,
2011 2.218,    2012 2.262,    2013 1.205
```

In the first period the present value consumption is very high, so this is favoured over investments. However, to sustain long-run high levels of consumption investment are needed to ensure the capital stock level. After a certain period (in this case 2013) investing in new capital becomes less and less attractive, since all remaining capital is thrown away after the model horizon. Hence investments fall back to zero to maximise consumption.

Display emission and damage:

```
----     55 VARIABLE  E.L           emission variable
2001 5.714,    2002 5.684,    2003 5.693,    2004 5.807,
2005 5.923,    2006 6.042     2007 6.163,    2008 6.286,
2009 6.412,    2010 6.540,    2011 6.671,    2012 6.804
2013 6.860,    2014 6.824,    2015 6.789,    2016 6.753,
2017 6.718,    2018 6.683     2019 6.648,    2020 6.614
```

```
----     55 VARIABLE  D.L           damage
2001 0.571,    2002 0.568,    2003 0.569,    2004 0.581,
2005 0.592,    2006 0.604,    2007 0.616,    2008 0.629,
2009 0.641,    2010 0.654,    2011 0.667,    2012 0.680,
2013 0.686,    2014 0.682,    2015 0.679,    2016 0.675,
2017 0.672,    2018 0.668,    2019 0.665,    2020 0.661
```

The overall investment path is also influenced by the new variables:

```
----     55 VARIABLE  I.L           investment
2003 0.512,    2004 1.811,    2005 1.848,    2006 1.884,
2007 1.922,    2008 1.960,    2009 2.000,    2010 2.040,
2011 2.080,    2012 2.122,    2013 1.130
```

Adding emission reduction costs:
```
ER(T)= 0.20       and   CR(T)= 0.01 for each year.
```

```
----     57 VARIABLE  I.L           investment
2003 0.512,    2004 1.811,    2005 1.847,    2006 1.884,
2007 1.922,    2008 1.960,    2009 2.000,    2010 2.040,
2011 2.080,    2012 2.122,    2013 1.130
```

Having two subsets:
```
ER(2001-2010)=0.2       CR(2001-2010)=0.01
ER(2011-2020)=0.3       CR(2011-2020)=0.022
```

Results from Exercise 3.3.2. Making some more scenarios

When changing the discount rate, the optimal investment path becomes:

For `R = 0.03`:

```
----     55 VARIABLE  I.L           investment
2003 0.512,    2004 1.811,    2005 1.847,    2006 1.884,    2007 1.922,
2008 1.960,    2009 2.000,    2010 2.040,    2011 2.080,    2012 2.122,
2013 1.130,

----     55 VARIABLE  E.L           emission variable
2001 5.514,    2002 5.484,    2003 5.493,    2004 5.607,    2005 5.723
2006 5.842,    2007 5.963,    2008 6.086,    2009 6.212,    2010 6.340
2011 6.471,    2012 6.604,    2013 6.660,    2014 6.624,    2015 6.589
2016 6.553,    2017 6.518,    2018 6.483,    2019 6.448,    2020 6.414
```

For `R = 0.06`:

```
VARIABLE  I.L           investment
2005 1.106,    2006 1.550,    2007 1.581,    2008 1.612,    2009 1.645,
2010 1.677,    2011 1.711,    2012 1.745,    2013 1.780,    2014 0.410

55 VARIABLE  E.L           emission variable
2001 5.514,    2002 5.484,    2003 5.454,    2004 5.425,    2005 5.496
2006 5.610,    2007 5.726,    2008 5.845,    2009 5.966,    2010 6.089
2011 6.215,    2012 6.343,    2013 6.474,    2014 6.479,    2015 6.444
2016 6.409,    2017 6.375,    2018 6.341,    2019 6.307,    2020 6.273
```

For `R = 0.09`:

```
----     55 VARIABLE  I.L           investment
2006 0.596,    2007 1.347,    2008 1.374,    2009 1.401,    2010 1.429
2011 1.458,    2012 1.487,    2013 1.517,    2014 1.203

----     55 VARIABLE  E.L           emission variable
2001 5.514,    2002 5.484,    2003 5.454,    2004 5.425,    2005 5.395
2006 5.427,    2007 5.540,    2008 5.654,    2009 5.772,    2010 5.891
2011 6.013,    2012 6.137,    2013 6.264,    2014 6.358,    2015 6.324
2016 6.290,    2017 6.256,    2018 6.222,    2019 6.189,    2020 6.156
```

Including technological progress:

```
----     58 VARIABLE  I.L           investment
2003 1.670,    2004 2.574,    2005 2.724,    2006 2.883,    2007 3.052
2008 3.230,    2009 3.418,    2010 3.618,    2011 3.829,    2012 4.053
2013 4.289,    2014 1.984

----     58 VARIABLE  E.L           emission variable
2001  5.514,    2002  5.654,    2003  5.930,    2004  6.288,    2005  6.667
2006  7.068,    2007  7.492,    2008  7.941,    2009  8.417,    2010  8.920
2011  9.452,    2012 10.016,    2013 10.612,    2014 11.044,    2015 11.321
2016 11.605,    2017 11.895,    2018 12.193,    2019 12.498,    2020 12.811
```

Both emission and investment show significant increase.

Put an upper bound on emission:

```
----     63 VARIABLE  E.L           emission
2001 5.514,    2002 5.484,    2003 5.493,    2004 5.607,    2005 5.723
2006 5.842,    2007 5.963,    2008 6.086,    2009 6.212,    2010 6.340
2011 6.471,    2012 6.500,    2013 6.500,    2014 6.500,    2015 6.500
2016 6.500,    2017 6.497,    2018 6.462,    2019 6.427,    2020 6.392
```

Emission level is limited by the upper bound from the year 2012.

Investment and consumption per capita are also lower in the second decade.

Results from Exercise 3.3.3. Specifying a model for transboundary pollution (optional)

For 20% reduction in emissions or deposition, the report parameter gives:

```
   156 PARAMETER REPORT    Reporting the results of the scenarios
                    No Policy  Emis Poli~  Depo Poli~
FUEL     .NETHERL       3.410       2.731       1.908
FUEL     .GERMANY       9.174       9.174       9.174
RENEW    .NETHERL 1.000000E-6 1.000000E-6       0.138
RENEW    .GERMANY 1.000000E-6 1.000000E-6 1.000000E-6
CONSERV  .NETHERL 1.000000E-6       0.679       1.364
CONSERV  .GERMANY       6.116       6.116       6.116
EMIS SO2 .NETHERL       0.225       0.180       0.120
EMIS SO2 .GERMANY       3.518       3.518       3.168
EMIS NOx .NETHERL       0.633       0.506       0.305
EMIS NOx .GERMANY       2.027       2.027       1.917
EMRED SO2.NETHERL 1.00000E-10       0.003       0.089
EMRED SO2.GERMANY 1.00000E-10 1.00000E-10       0.914
EMRED NOx.NETHERL 1.00000E-10       0.003       0.265
EMRED NOx.GERMANY 1.00000E-10 1.00000E-10       0.495
ACIDDEP  .NETHERL      16.122      15.232      12.898
ACIDDEP  .GERMANY       3.300       3.273       2.955
CCNTRY   .NETHERL      54.560      54.594      55.564
CCNTRY   .GERMANY     243.723     243.723     245.352
C        .ALL         298.283     298.317     300.916
```

For 50% reduction in emissions or deposition, `DISPLAY REPORT` gives:

```
   156 PARAMETER REPORT    Reporting the results of the scenarios
                    No Policy  Emis Poli~  Depo Poli~
FUEL     .NETHERL       3.410       1.879       1.705
FUEL     .GERMANY       9.174       9.174       7.645
RENEW    .NETHERL 1.000000E-6       0.167       0.341
RENEW    .GERMANY 1.000000E-6 1.000000E-6       1.529
CONSERV  .NETHERL 1.000000E-6       1.364       1.364
CONSERV  .GERMANY       6.116       6.116       6.116
EMIS SO2 .NETHERL       0.225       0.113       0.093
EMIS SO2 .GERMANY       3.518       3.518       1.623
EMIS NOx .NETHERL       0.633       0.316       0.224
EMIS NOx .GERMANY       2.027       2.027       1.506
EMRED SO2.NETHERL 1.00000E-10       0.174       0.291
EMRED SO2.GERMANY 1.00000E-10 1.00000E-10       3.414
EMRED NOx.NETHERL 1.00000E-10       0.174       0.495
EMRED NOx.GERMANY 1.00000E-10 1.00000E-10       0.828
ACIDDEP  .NETHERL      16.122      13.896       8.061
ACIDDEP  .GERMANY       3.300       3.232       1.754
CCNTRY   .NETHERL      54.560      55.711      58.456
CCNTRY   .GERMANY     243.723     243.723     269.049
C        .ALL         298.283     299.433     327.505
```

Results from Exercise 3.3.4. Investigating international co-operation (optional)
The exercise presents a non-co-operative strategy. Germany follows no emission and deposition policy, their cost is fixed on the base level. The result shows that the cost of The Netherlands increased significantly.

Using 20% reduction, the report gives:

```
   158 PARAMETER REPORT    Reporting the results of the scenarios
                    No Policy  Emis Poli~  Depo Poli~
FUEL     .NETHERL       3.410       2.731       1.705
FUEL     .GERMANY       9.174       9.174       9.174
RENEW    .NETHERL 1.000000E-6 1.000000E-6       0.341
RENEW    .GERMANY 1.000000E-6 1.000000E-6 1.000000E-6
CONSERV  .NETHERL 1.000000E-6       0.679       1.364
CONSERV  .GERMANY       6.116       6.116       6.116
EMIS SO2 .NETHERL       0.225       0.180       0.066
EMIS SO2 .GERMANY       3.518       3.518       3.518
EMIS NOx .NETHERL       0.633       0.506       0.169
EMIS NOx .GERMANY       2.027       2.027       2.027
EMRED SO2.NETHERL 1.00000E-10       0.003       0.713
EMRED SO2.GERMANY 1.00000E-10 1.00000E-10 1.00000E-10
EMRED NOx.NETHERL 1.00000E-10       0.003       0.794
EMRED NOx.GERMANY 1.00000E-10 1.00000E-10 1.00000E-10
ACIDDEP  .NETHERL      16.122      15.232      12.898
ACIDDEP  .GERMANY       3.300       3.273       3.201
CCNTRY   .NETHERL      54.560      54.594      66.848
CCNTRY   .GERMANY     243.723     243.723     243.723
C        .ALL         298.283     298.317     310.570
```

For 50% required reduction in deposition in the Netherlands, the model becomes infeasible. The reason is that the transboundary pollution from Germany is so large that this deposition target can never be met.

Minimising just the Dutch cost implies that Germany either does not care about making costs, or does not have the power to influence the results. In essence, in this simulation it is assumed that the Netherlands dominate Germany.

The results of such an exercise would be that all environmental targets are reached by reducing transboundary pollution from Germany to the Netherlands: emission reductions in the Netherlands are very low, while they are very high in Germany. Hence, costs are low in the Netherlands and high in Germany.

Results from Exercise 3.3.5. A basic systems model

The model without pollution:

```
----      53 VARIABLE K.L             Capital stock
2000  100.000,    2001  105.000,    2002  110.250,    2003  115.762,    2004  121.551
2005  127.628,    2006  134.010,    2007  140.710,    2008  147.746,    2009  155.133
2010  162.889,    2011  171.034,    2012  179.586,    2013  188.565,    2014  197.993
2015  207.893,    2016  218.287,    2017  229.202,    2018  240.662,    2019  252.695
2020  265.330,    2021  278.596,    2022  292.526,    2023  307.152,    2024  322.510
2025  338.635,    2026  355.567,    2027  373.346,    2028  392.013,    2029  411.614
2030  432.194,    2031  453.804,    2032  476.494,    2033  500.319,    2034  525.335
2035  551.602,    2036  579.182,    2037  608.141,    2038  638.548,    2039  670.475
2040  703.999,    2041  739.199,    2042  776.159,    2043  814.967,    2044  855.715
2045  898.501,    2046  943.426,    2047  990.597,    2048 1040.127,    2049 1092.133
2050 1146.740
```

```
----     53 VARIABLE P.L             Population
2000 6000.000,    2001 6060.000,    2002 6120.600,    2003 6181.806,    2004 6243.624
2005 6306.060,    2006 6369.121,    2007 6432.812,    2008 6497.140,    2009 6562.112
2010 6627.733,    2011 6694.010,    2012 6760.950,    2013 6828.560,    2014 6896.845
2015 6965.814,    2016 7035.472,    2017 7105.827,    2018 7176.885,    2019 7248.654
2020 7321.140,    2021 7394.352,    2022 7468.295,    2023 7542.978,    2024 7618.408
2025 7694.592,    2026 7771.538,    2027 7849.253,    2028 7927.746,    2029 8007.023
2030 8087.093,    2031 8167.964,    2032 8249.644,    2033 8332.141,    2034 8415.462
2035 8499.617,    2036 8584.613,    2037 8670.459,    2038 8757.163,    2039 8844.735
2040 8933.182,    2041 9022.514,    2042 9112.739,    2043 9203.867,    2044 9295.905
2045 9388.864,    2046 9482.753,    2047 9577.581,    2048 9673.356,    2049 9770.090
2050 9867.791
```

With pollution feedback:

```
----     53 VARIABLE  P.L  population
2000 6000.000,    2001 6059.775,    2002 6118.728,    2003 6176.867
2004 6234.201,    2005 6290.738,    2006 6346.487,    2007 6401.456
2008 6455.653,    2009 6509.086,    2010 6561.764,    2011 6613.695
2012 6664.886,    2013 6715.344,    2014 6765.079,    2015 6814.097
2016 6862.406,    2017 6910.014,    2018 6956.928,    2019 7003.155
2020 7048.703,    2021 7093.578,    2022 7137.789,    2023 7181.341
2024 7224.243,    2025 7266.500,    2026 7308.120,    2027 7349.110
2028 7389.475,    2029 7429.224,    2030 7468.362,    2031 7506.896
2032 7544.833,    2033 7582.178,    2034 7618.938,    2035 7655.120
2036 7690.729,    2037 7725.771,    2038 7760.254,    2039 7794.182
2040 7827.561,    2041 7860.398,    2042 7892.699,    2043 7924.468
2044 7955.712,    2045 7986.437,    2046 8016.647,    2047 8046.349
2048 8075.548,    2049 8104.249,    2050 8132.459
```

```
----     53 VARIABLE  POL.L  pollution
2000 20.000,    2001 20.250,    2002 20.503,    2003 20.759,    2004 21.019
2005 21.282,    2006 21.548,    2007 21.817,    2008 22.090,    2009 22.366
2010 22.645,    2011 22.928,    2012 23.215,    2013 23.505,    2014 23.799
2015 24.097,    2016 24.398,    2017 24.703,    2018 25.012,    2019 25.324
2020 25.641,    2021 25.961,    2022 26.286,    2023 26.614,    2024 26.947
2025 27.284,    2026 27.625,    2027 27.970,    2028 28.320,    2029 28.674
2030 29.032,    2031 29.395,    2032 29.763,    2033 30.135,    2034 30.511
2035 30.893,    2036 31.279,    2037 31.670,    2038 32.066,    2039 32.467
2040 32.872,    2041 33.283,    2042 33.699,    2043 34.121,    2044 34.547
2045 34.979,    2046 35.416,    2047 35.859,    2048 36.307,    2049 36.761
2050 37.220
```

The amount of capital stock is the same as before, since it is not influenced by the feedback.

Results from Exercise 3.3.6. Understanding what you have done

No results for this exercise can be given.

II.4. Results for Modelling exercises

Results from Exercise 4.1.1. A basic linear optimisation model

```
----    199 PARAMETER croprep  crop report summary
                onions      cotton      tomato       total
BM.landuse       2.000       2.000       2.000       6.000
BM.output        6.000       3.000       6.000      15.000
BM.revenue     750.000    1050.000     720.000    2520.000
CF.landuse       1.214       2.786       1.214       5.214
CF.output        3.643       4.179       3.643      11.464
CF.revenue     455.357    1462.500     437.143    2355.000

----    199 PARAMETER labrep  labor report summary(days)
              demand      family   temporary    hire-out
BM.jan        10.320      10.320                  14.680
BM.feb        10.000      10.000                  15.000
BM.mar        20.000      20.000                   5.000
BM.apr        49.160      25.000      24.160
BM.may        22.840      22.840                   2.160
BM.jun         4.000       4.000                  21.000
BM.jul        37.000      25.000      12.000
BM.aug        34.000      25.000       9.000
BM.sep        26.000      25.000       1.000
BM.oct        66.000      25.000      41.000
BM.nov        58.320      25.000      33.320
BM.dec         9.360       9.360                  15.640
BM.total     347.000     226.520     120.480      73.480
CF.jan         6.266       6.266                  18.734
CF.feb         6.071       6.071                  18.929
CF.mar        20.000      20.000                   5.000
CF.apr        37.704      25.000      12.704
CF.may        28.010      25.000       3.010
CF.jun         5.571       5.571                  19.429
CF.jul        24.821      24.821                   0.179
CF.aug        23.786      23.786                   1.214
CF.sep        17.357      17.357                   7.643
CF.oct        80.929      25.000      55.929
CF.nov        54.266      25.000      29.266
CF.dec         5.683       5.683                  19.317
CF.total     310.464     209.556     100.909      90.444

----    199 PARAMETER fertrep  fertilizer report summary (kg)
          onions      cotton      tomato       total
BM.N      60.000      20.000     100.000     180.000
BM.P      40.000      10.000      90.000     140.000
BM.K      40.000      18.000     160.000     218.000
CF.N      36.429      27.857      60.714     125.000
CF.P      24.286      13.929      54.643      92.857
CF.K      24.286      25.071      97.143     146.500
```

Results from Exercise 4.2.1. A basic partial equilibrium model

Benchmark values:

```
---- VAR C  consumption of the good produced by sector i

       LOWER     LEVEL     UPPER    MARGINAL
spc 1.0000E-7     6.000     +INF       .
fpc 1.0000E-7     4.000     +INF       .
ohi 1.0000E-7   130.000     +INF       .

                       LOWER     LEVEL     UPPER    MARGINAL
---- VAR U          1.0000E-7     1.000     +INF       .
```

Counterfactual:

```
---- VAR C  consumption of the good produced by sector i
       LOWER     LEVEL     UPPER    MARGINAL
spc 1.0000E-7     3.220     +INF       .
fpc 1.0000E-7     7.512     +INF       .
ohi 1.0000E-7   129.268     +INF       .

                       LOWER     LEVEL     UPPER    MARGINAL
---- VAR U          1.0000E-7     0.993     +INF       .
```

Results from Exercise 4.3.1. A model for stability of international climate negotiations

```
----    240 PARAMETER RESULT
        abat level     payoff         TAC          TB         MAC          MB
Empty coalition:
1 .USA      16.235     14.542       1.238      15.780       0.196       0.196
1 .EU        6.623     15.892       0.562      16.454       0.205       0.205
1 .RoOECD    5.366     14.033       0.397      14.430       0.180       0.180
1 .RoEurope  8.106      5.419       0.197       5.616       0.070       0.070
1 .Asia     36.306      7.707       1.838       9.545       0.119       0.119
1 .RoWorld   7.708      7.566       0.332       7.898       0.098       0.098
1 .global   80.344     65.159
…
Kyoto coalition:
xx.USA      32.314     17.305       7.702      25.007       0.651       0.196
xx.EU       13.630     22.655       3.422      26.076       0.651       0.205
xx.RoOECD   11.856     19.927       2.941      22.868       0.651       0.180
xx.RoEurope 25.511      3.284       5.616       8.900       0.651       0.070
xx.Asia     36.306     13.289       1.838      15.126       0.119       0.119
xx.RoWorld   7.708     12.184       0.332      12.516       0.098       0.098
xx.global  127.325     88.643
…
Global coalition:
64.USA      37.870     38.277      11.906      50.183       0.868       0.196
64.EU       16.140     47.008       5.321      52.330       0.868       0.205
64.RoOECD   14.003     41.325       4.566      45.891       0.868       0.180
64.RoEurope 29.521      9.211       8.650      17.861       0.868       0.070
64.Asia    126.289    -11.014      41.369      30.355       0.868       0.119
64.RoWorld  31.692     14.602      10.515      25.117       0.868       0.098
64.global  255.516    139.410
```

Results from Exercise 4.4.1. A basic input-output model

Result of the base scenario reproduce the original IO table:

```
----     85 PARAMETER RESULT        Resulting IO-table

        AGRI     INDU     SERV     CONS     INV      GOVT     EXPORT   TOTAL
AGRI     10       30       40       100      0        40       180      400
INDU     20       60       50       300      200      30       340      1000
SERV     40       60       10       200      0        20       170      500
IMPO     60       120      20       400      100      100      0        800
DEPR     10       20       10                                           40
WAGES    200      600      300                                          1100
PROFIT   60       110      70                                           240
TOTAL    400      1000     500      1000     300      190      690
```

Note that GAMS will sometimes change the order of the rows.

Increase industrial export with 10%

```
           AGRI       INDU       SERV    CONS   INV    GOVT     EXPORT     TOTAL

AGRI     10.033     31.095     40.189    100           40       180      401.317
INDU     20.066     62.190     50.237    300    200    30       374     1036.492
SERV     40.132     62.190     10.047    200           20       170      502.369
IMPO     60.198    124.379     20.095    400    100   100                804.671
DEPR     10.033     20.730     10.047                                     40.81
WAGES   200.659    621.895    301.421                                   1123.975
PROFIT   60.198    114.014     70.332                                    244.543
TOTAL   401.317   1036.492    502.369   1000    300   190       724
```

Results from Exercise 4.4.2. An extended input-output model

Increase the consumption of agriculture products by 15%

```
----    110 PARAMETER RESULT        Resulting IO-table

           AGRI      INDU       SERV    CONS    INV    GOVT   EXPORT   TOTAL

AGRI     10.389    30.030     40.132    115            40      180     415.551
INDU     20.778    60.060     50.165    300    200     30      340    1001.003
SERV     41.555    60.060     10.033    200            20      170     501.648
DEPR     10.389    20.020     10.033                                    40.442
IMPORT   62.333   120.120     20.066    400    100    100              802.519
WAGES   207.775   600.602    300.989                                  1109.366
PROFIT   62.333   110.110     70.231                                   242.674
TOTAL   415.551  1001.003    501.648   1015    300    190      690

SO2     103.888   200.201    250.824    100     30     50              734.912
NOx      41.555    50.050     50.165     60     20     20              241.770
```

Results from Exercise 4.4.3. An input-output model with purification (optional)

The modified IO table is:

```
TABLE IODATA(*,*) Input-output table in billions of US dollars
AGRI     INDU     SERV     PURI     CONS     INV      GOVT     EXPORT   TOTAL
AGRI     10       30        40         0       100               40       180       400
INDU     20       60        50        15       300      200      30       340      1015
SERV     40       60        10         5       200               20       170       505
PURI     15       20         5                  15                                   55
IMPO     60       120       20        10       400      100     100                  810
DEPR     10       20        10         5                                             44
WAGES    200      600       300       20                                           1120
PROFIT   45       105       70                                                     220
TOTAL    400      1015      505       55       1015     300      190      690
SO2      100      200       250     -100       100       30       50                 630
NOx      40       50        50                  60       20       20                 240
```

Effect of 20% increase in delivery from purification to industry

```
       AGRI     INDU     SERV    PURI     CONS     INV      GOVT   EXPORT   TOTAL
SO2    100.019  200.241  250.226 -107.339  100      30       50              623.146
NOx     40.007   50.060   50.045            60      20       20              240.113
```

7.339 tonnes SO2 increase in purification activity

Increase the purification activity to 200:

```
----     90 PARAMETER RESULT        Resulting IO-table

          AGRI      INDU      SERV    PURI  CONS   INV   GOVT   EXPORT   TOTAL
AGRI     10.000    30.000    40.000          100          40      180     400
IND      20.000    60.000    50.000    30    300   200    30      340    1030
SER      40.000    60.000    10.000    10    200          20      170     510
PUR      30.000    40.000    10.000           30                          110
IMPORT   60.000   120.000    20.000    20    400   100   100              820
DEPR     10.000    20.000    10.000    10                                  50
WAGES   200.000   600.000   300.000    40                                1140
PROFIT   30.000   100.000    70.000                                       200
TOTAL   400.000  1030.000   510.000   110   1030   300   190      690

SO2     100.000   200.000   250.000  -200    100    30    50              530
NOx      40.000    50.000    50.000           60    20    20              240
```

Results from Exercise 4.5.1. A basic CGE model

The equations are not given in the solution. (That would be too easy!)

Prices before scaling (remember that these are not useful for interpretation):

```
PARAMETER w = 0.198  price of labour
PARAMETER R = 0.329  price of capital
PARAMETER scale = 0.198  only relative prices matter so scale all
                         prices
```

The result of the CGE model:

```
PARAMETER RESULT        parameter for storing results
                                 A           B      total
1. base.commodity price       2.663       2.404
1. base.Capital price                                1.667
1. base.Wage                                         1.000
1. base.income                                        2000
1. base.production          375.496     415.887
1. base.consumption         375.496     415.887
1. base.Utility                                    395.176
1. base.K                   360.000     240.000
1. base.L                   400.000     600.000
```

(optional) The displayed profit converges to zero (due to numerical accuracy the reported numbers will not be exactly zero, but should be very close to zero.

Results from Exercise 4.5.2. Interpretation of the CGE model.

The number of market clearing and resource balance equations shows the number of markets. In this model you find a market clearing equation for QA, for QB, a capital balance and labour balance. (Four in total).

The total value of outputs is equal to the income {Q(G)*P(G)}; the total value of inputs is equal to the cost of production {K(G)*r+L(G)*w}. The value is equal to 1000 for both products.

Changing technology parameter (beta=1.5)

```
PARAMETER RESULT  parameter for storing results
                                     A           B       total
1. hightech.commodity price       1.775       1.603
1. hightech.Capital price                                1.667
1. hightech.Wage                                         1.000
1. hightech.income                                    2000.000
1. hightech.production          563.244     623.830
1. hightech.consumption         563.244     623.830
1. hightech.Utility                                    592.764
1. hightech.K                   360.000     240.000
1. hightech.L                   400.000     600.000
```

A higher technological parameter results in higher production with the same amount of capital and labour. It also contributes to lower commodity prices. The utility also increases due to higher consumption. The allocation of resources over the sectors does not change.

The results for beta=0.5 can be explained as the reverse:

```
                                     A           B       total
1. hightech.commodity price       5.326       4.809
1. hightech.Capital price                                1.667
1. hightech.Wage                                         1.000
1. hightech.income                                    2000.000
1. hightech.production          187.748     207.943
1. hightech.consumption         187.748     207.943
1. hightech.Utility                                    197.588
1. hightech.K                   360.000     240.000
1. hightech.L                   400.000     600.000
```

If gamma=0.25 in the utility function: U =E= C('A')**gamma*C('B')**(1-gamma)

The results:

```
                                  A           B       total
1. bpref.commodity price       2.361       2.219
1. bpref.Capital price                                1.364
1. bpref.Wage                                         1.000
1. bpref.income                                    1818.182
1. bpref.production          192.519     614.517
1. bpref.consumption         192.519     614.517
1. bpref.Utility                                    459.747
1. bpref.K                   200.000     400.000
1. bpref.L                   181.818     818.182
```

By decreasing the consumer's preference for product A, the production/consumption of product A will decline, meanwhile the demand for the substitution product B will increase. This also influences prices, primarily because the production of good A is less labour intensive than the production of good B.

Input-output table (gamma=0.25)

```
----    152 PARAMETER IO  Input-output table

                       A           B       Cons  TotalVolume      Price  TotalValue

Base.A                                  192.519     192.519       2.361     454.545
Base.B                                  614.517     614.517       2.219    1363.636
Base.Capital     200.000     400.000                600.000       1.364     818.182
Base.Labour      181.818     818.182               1000.000       1.000    1000.000
Base.Profits        {~0}        {~0}
Base.TotalValue  454.545    1363.636   1818.182
```

Results from Exercise 4.5.3. CGE model with environmental issues.

NOTE: With GAMS build 20.4, this exercise works when using the MINOS solver, but not with Conopt.

If   `PERMITS=1000,`

```
                                 A          B
1. base.commodity price       2.663      2.404
1. base.Capital price                                1.667
1. base.Wage                                         1.000
1. base.income                                    2000.000
1. base.production          375.496     415.887
1. base.consumption         375.496     415.887
1. base.Utility                                    395.176
1. base.K                   360.000     240.000
1. base.L                   400.000     600.000
1. base.EMISSION            394.271     249.532    643.803
```

If   `PERMITS=470,`

```
                                 A          B
2. permit470.commodity price  1.000       0.500
2. permit470.Capital price                             EPS
2. permit470.Wage                                      EPS
2. permit470.Emission permit price                   1.000
2. permit470.income                                470.000
2. permit470.production     223.810     391.667
```

```
2. permit470.consumption    223.810     391.667
2. permit470.Utility                               296.072
2. permit470.K              108.467     491.533
2. permit470.L              663.265     336.635
2. permit470.EMISSION       235.000     235.000    470.000
```

If PERMITS=370,

```
                                  A          B
3. permit370.commodity price  1.000       0.500
3. permit370.Capital price                             EPS
3. permit370.Wage                                      EPS
3. permit370.Emission permit price                   1.000
3. permit370.income                                370.000
3. permit370.production     176.190     308.333
3. permit370.consumption    176.190     308.333
3. permit370.Utility                               233.078
3. permit370.K               64.990     535.010
3. permit370.L              786.470     213.530
3. permit370.EMISSION       185.000     185.000    370.000
```

Results from Exercise 4.5.4. Specifying an open economy (optional).

```
                                       A           B      total
1. free trade.commodity price      1.022       2.045
1. free trade.Capital price                                1.111
1. free trade.Wage                                         1.000
1. free trade.income                                    1666.671
1. free trade.production           0.010     815.183
1. free trade.consumption        815.188     407.594
1. free trade.Utility                                    576.425
1. free trade.net export        -815.178     407.589
1. free trade.K                    0.010     599.990
1. free trade.L                    0.010     999.990
1. free trade.EMISSION            81.528     489.110     570.638

2. envir policy.commodity price    1.000       0.500
2. envir policy.Capital price                                EPS
2. envir policy.Wage                                         EPS
2. envir policy.Emission permit price                      1.000
2. envir policy.income                                   470.000
2. envir policy.production       223.810     391.667
2. envir policy.consumption      223.810     391.667
2. envir policy.Utility                                  296.072
2. envir policy.net export             0           0
2. envir policy.K                108.467     491.533
2. envir policy.L                663.265     336.635
2. envir policy.EMISSION         235.000     235.000     470.000

3. stricter policy.commodity price 1.000       0.500
3. stricter policy.Capital price                               EPS
3. stricter policy.Wage                                        EPS
3. stricter policy.Emission permit price                     1.000
3. stricter policy.income                                  370.000
3. stricter policy.production    176.190     308.333
3. stricter policy.consumption   176.190     308.333
3. stricter policy.Utility                                 233.078
```

```
3. stricter policy.net export          0           0
3. stricter policy.K              64.990     535.010
3. stricter policy.L             786.470     213.530
3. stricter policy.EMISSION      185.000     185.000     370.000
```

In the free trade scenario only product B is produced in the country, and product A is imported, due to comparative cost advantage.

In the environmental policy scenarios there is a decrease in foreign trade, it is no longer advantageous to trade the goods, because the associated transport cost (pollution) is higher than associated cost of domestic production.

Results from Exercise 4.5.5. Specifying a multi-country model (optional).

The model can be downloaded from the GAMS page of the Environmental Economics Group. The scenario results are not available.

Results from Exercise 4.5.6. The Dinwiddy and Teal model (optional).

```
----  105 VARIABLE  U.L     = 0.734  utility of the consumer
          VARIABLE  C1.L    = 0.824  total consumption of good 1
          VARIABLE  C2.L    = 0.653  total consumption of good 2
          VARIABLE  X1.L    = 0.824  total production of good 1
          VARIABLE  X2.L    = 0.653  total production of good 2
          VARIABLE  K1.L    = 0.267  demand for capital in sector 1
          VARIABLE  K2.L    = 0.533  demand for capital in sector 2
          VARIABLE  L1.L    = 1.200  demand for labour in sector 1
          VARIABLE  L2.L    = 0.800  demand for labour in sector 2
          VARIABLE  PRF1.L  = 0.000  profits of sector 1
          VARIABLE  PRF2.L  = 0.000  profits of sector 2
----  106 VARIABLE  W.L     = 1.000  wage rate (price of labour)
          VARIABLE  R.L     = 1.500  rental price of capital
          VARIABLE  P1.L    = 1.942  price of good 1
          VARIABLE  P2.L    = 2.449  price of good 2
----  107 PARAMETER ESUM  = 115.051     pollution
```

Fix r=1

```
----  127 VARIABLE  W.L   = 0.667 wage rate (price of labour)
          VARIABLE  R.L   = 1.000 rental price of capital
          VARIABLE  P1.L  = 1.295 price of good 1
          VARIABLE  P2.L  = 1.633 price of good 2
```

All the other variables are unchanged, therefore the total pollution parameter also remains the same.

Results from Exercise 4.5.7. The basic CGE model in MPSGE (optional).

Results are not available for this exercise.

Results from Exercise 4.6.1. A basic growth model (a simplified DICE model)

```
----      53 VARIABLE  Y.L            economic activity
1980 100.000,    1981 103.000,    1982 106.090,    1983 109.273,    1984 112.551
1985 115.927,    1986 119.405,    1987 122.987,    1988 126.677,    1989 130.477
1990 134.392,    1991 138.423,    1992 142.576,    1993 146.853,    1994 151.259
1995 155.797,    1996 160.471,    1997 165.285,    1998 170.243,    1999 175.351
2000 180.611,    2001 186.029,    2002 191.610,    2003 197.359,    2004 203.279
2005 209.378,    2006 215.659,    2007 222.129,    2008 228.793,    2009 235.657
2010 242.726

----      53 VARIABLE  E.L           total carbon emission
1980  50.000,    1981  50.373,    1982  53.277,    1983  56.339,    1984  59.566
1985  62.965,    1986  66.544,    1987  70.312,    1988  74.276,    1989  78.444
1990  82.826,    1991  87.429,    1992  92.262,    1993  97.334,    1994 102.653
1995 108.228,    1996 114.067,    1997 120.179,    1998 126.572,    1999 133.253
2000 140.230,    2001 147.510,    2002 155.100,    2003 163.006,    2004 171.234
2005 179.787,    2006 188.670,    2007 197.886,    2008 207.438,    2009 217.325
2010 227.549

----      53 VARIABLE  AC.L          marginal abatement cost
1981 0.001,    1982 0.001,    1983 0.002,    1984 0.002,    1985 0.002,    1986 0.002
1987 0.002,    1988 0.003,    1989 0.003,    1990 0.003,    1991 0.004,    1992 0.004
1993 0.005,    1994 0.005,    1995 0.006,    1996 0.007,    1997 0.007,    1998 0.008
1999 0.009,    2000 0.010,    2001 0.011,    2002 0.012,    2003 0.013,    2004 0.015
2005 0.016,    2006 0.018,    2007 0.020,    2008 0.022,    2009 0.024,    2010 0.026

----      53 VARIABLE  DC.L          marginal damage cost
1980 0.025,    1981 0.024,    1982 0.025,    1983 0.027,    1984 0.028,    1985 0.030
1986 0.031,    1987 0.033,    1988 0.034,    1989 0.036,    1990 0.038,    1991 0.040
1992 0.042,    1993 0.044,    1994 0.046,    1995 0.048,    1996 0.051,    1997 0.053
1998 0.055,    1999 0.058,    2000 0.060,    2001 0.063,    2002 0.066,    2003 0.068
2004 0.071,    2005 0.074,    2006 0.077,    2007 0.079,    2008 0.082,    2009 0.085
2010 0.088

----      53 VARIABLE  C.L           consumption
1980  97.500,    1981 100.406,    1982 103.264,    1983 106.195,    1984 109.199
1985 112.278,    1986 115.432,    1987 118.664,    1988 121.972,    1989 125.360
1990 128.826,    1991 132.372,    1992 135.999,    1993 139.706,    1994 143.495
1995 147.366,    1996 151.318,    1997 155.353,    1998 159.469,    1999 163.668
2000 167.948,    2001 172.309,    2002 176.751,    2003 181.273,    2004 185.875
2005 190.556,    2006 195.315,    2007 200.151,    2008 205.063,    2009 210.049
2010 215.110

----      53 VARIABLE  U.L           utility
1980 97.500,    1981 96.544,    1982 95.473,    1983 94.407,    1984 93.344
1985 92.284,    1986 91.228,    1987 90.175,    1988 89.124,    1989 88.076
1990 87.030,    1991 85.987,    1992 84.945,    1993 83.904,    1994 82.865
1995 81.827,    1996 80.790,    1997 79.754,    1998 78.719,    1999 77.684
2000 76.649,    2001 75.615,    2002 74.581,    2003 73.547,    2004 72.514
2005 71.481,    2006 70.448,    2007 69.416,    2008 68.384,    2009 67.353
2010 66.322

----      53 VARIABLE  MU.L          abatement ratio
1981 0.050,    1982 0.053,    1983 0.056,    1984 0.060,    1985 0.063,    1986 0.067
1987 0.070,    1988 0.074,    1989 0.078,    1990 0.083,    1991 0.087,    1992 0.092
1993 0.097,    1994 0.103,    1995 0.108,    1996 0.114,    1997 0.120,    1998 0.127
1999 0.133,    2000 0.140,    2001 0.148,    2002 0.155,    2003 0.163,    2004 0.171
2005 0.180,    2006 0.189,    2007 0.198,    2008 0.207,    2009 0.217,    2010 0.228
```

Results from Exercise 4.6.2. An extended growth model (the DICE model)

Four scenarios:

- In the *base scenario* there is no abatement and hence no abatement cost, and the temperature costs are also assumed to be zero. This represents a Business-as-Usual scenario, where climate issues have no impact on the economy; or, put differently, where the economy does not react to climate issues.

- In the *market scen*ario there are temperature costs, but the abatement cost is still fixed to zero. This scenario reflects the case where climate issues do influence the economy, but the economy does not react.
- In the *opt_cont scenario* an optimal abatement rate is calculated. This optimal point is where marginal abatement costs equal marginal temperature costs. The climate influences the economy and the economy reacts by investing in abatement.
- In the *concent scenario* an upper bound is placed on CO2 concentration. This scenario reflects a situation where the policy makers set a pre-defined goal for the climate issue; this goal is not necessarily optimal; nonetheless, the response by the economy is set at a cost-effective (i.e. least cost) level.

Y(T) is the output and its value is the same in the four scenarios. Q(T) is the level of the gross output, and its value is $Q_T = \frac{Y_T}{(1 - abcost_T) * (1 - tecost_T)}$

Consequently, Q(T) can be interpreted as the total production value, while Y(T) is the amount of produced goods available to consumers. This latter quantity is smaller since some of the produced goods have to be used for abatement and some are lost due to damages.

In the base case:	abcost = 0 and tecost = 0,	therefore	$Q_T = Y_T$
Market scenario:	abcost = 0 and tecost > 0,	therefore	$Q_T > Y_T$
Opt_cont scenario:	abcost > 0 and tecost > 0,	therefore	$Q_T > Y_T$
Concent scenario:	abcost > 0 and tecost > 0,	therefore	$Q_T > Y_T$

Abcost and Tecost are expressed in marginal terms, as they are expressed per unit of production.

The ‘technical’ objective of the model is to maximise the total present value of the utility. However, the ‘economic’ objective, i.e. the purpose of the exercise, is to determine the optimal level of abatement (MU). This control ratio is the ‘free variable’ in the model, whose value cannot be determined beforehand but which results from equating marginal damage and abatement costs.

If the discount rate decreases the capital accumulation and output increases. The CO_2 concentration also increases.

More weight is placed on future consumption if you use a low discount rate. However, consumption in later periods can become irrelevant in present value terms, and the model may come with unrealistic values for consumption and investment.