Welcome to GAMS[1]

Jesper Jensen
TECA TRAINING ApS
jensen@tecatraining.dk

This version: September 2006

Getting started with GAMS

Welcome to the programming language GAMS. The purpose of this chapter is to get you started using GAMS on your own computer. We will assume that you know how to use standard programs on a computer, e.g. a word processor such as Microsoft Word, but experience with GAMS or programming is not necessary.

The chapter begins with an introduction to the key features of GAMS. We will then show how you install GAMS on your PC, and how you work with GAMS in practice. By the end of the chapter, you will have GAMS installed on your own PC and you will be able to run a small existing GAMS program. This gives you a good starting point for proceeding with the development of economic equilibrium models in GAMS.

If you have already used GAMS, you may find that you are familiar with most of the material in this chapter. You can then either quickly browse the chapter or simply skip it.

We will focus on the aspects of GAMS relevant to modelers interested in economic general equilibrium models. GAMS can be used for many, many other types of applications, but we deliberately skip these details here. Also, this introduction aims to help beginners and therefore ignores details that expert modelers may find useful or important.

Key features of GAMS

GAMS stands for **G**eneral **A**lgebraic **M**odeling **S**ystem. Basically, it is a language that allows you to develop and work with complicated models. The language does require an initial investment of time before you become familiar with the system, but the returns could be large and you will quickly be able to work with your own advanced economic models. You do *not* have to be a specialist in software design, operations research, or mathematics. All it takes is an interest in working with a computer, a solid background in applied economic theory and a commitment to invest time in learning the language.

Here are some of the advantages of using GAMS for economic equilibrium models:

A key feature of GAMS is access to a large set of existing solution algorithms. That is, as a modeler you should only focus on the formulation of a good model and then leave it to state-of-the-art solvers to solve it. This also implies that you can try different solvers without changing the model representation.

Another feature is independence between model formulation and the model data. That is, GAMS allows you to formulate your model without direct reference to a specific data set and you can therefore use the same model code with different data sets or different aggregations of the same data set. Your model may increase dramatically in size with a new data set, but the model formulation remains the same.

Learning GAMS also provides you with the advantage of any other language: It opens a whole new world and allows you to talk to others facing similar challenges. The options are many: You can discuss model code and ask for help, you can read your colleagues' work and you can benefit from ideas embodied in the large set of pre-existing models.[2]

[2] The GAMS installation procedure automatically installs a model library with GAMS models. Also, many research projects based on GAMS models have their own web sites where you can download a copy of their code.

The model representation in GAMS closely follows the way you would write the model using mathematical symbols. Not only does this help you and your colleagues to remember and understand what your model is about. It also allows you to change code in both a simple and safe manner. If you have experiences with programming, I am sure you value a language that reduces the risks of errors and, when errors occur, help you detect and eliminate them.

Flexibility with respect to both computer type and user interface implies that you can use GAMS in a way that suits your habits best. We will focus on GAMS installed on a PC, but the same code can also be run on a large mainframe computer. Also, you can edit your code in one editor, for example, the built-in GAMS-IDE editor, and later switch to another editor, e.g., Emacs (on computers running Unix) or Epsilon (an advanced Emacs-style programming editor for PCs running Windows and Linux).

A final feature of GAMS is that it can be used together with many other programs. For example, many find Microsoft Excel convenient for entering data and for reporting results in nicely formatted tables and figures. This can easily be done with the built-in GDX-utility (GDX stands for **G**AMS **D**ata E**x**change). You can also interface with other programs, such as the plotting program GNUPlot, by using utilities developed and contributed by other GAMS modelers.[3]

The choice is yours, but I strongly believe in the advantages GAMS offers relative to other alternatives such as Microsoft Excel, GEMPACK and Gauss. In any case, I recommend that you choose a programming language that allows you to focus on model development and economic analysis, unless you are really interested in subtle details about solution algorithms. Also, I recommend that you choose a language that both reduces the scope of errors and help you identify and learn from the errors you, like anyone else, will make.

Installation of GAMS

GAMS comes with an easy-to-use installation procedure. The procedure installs the full GAMS system and everything is included in one file with the name **`setup.exe`**.[4]

Perhaps you already have a copy of GAMS on a distribution CD. In this case, the **`setup.exe`** file can be found in the **`\systems\win`** directory.

If you do not have a distribution CD, you can download the most recent GAMS system from the GAMS home page www.gams.com under “Download Current GAMS System”. If you enter your Email address and name, GAMS automatically sends you an Email with a link to the **`setup.exe`** file and a user name and a password you will need to access the file. Follow the link, enter the provided user name and password and save the **`setup.exe`** file on your PC, for example on your desktop.

Now you should be ready to install the program. Simply locate and run (double-click) the **`setup.exe`** file, for example using the Windows Explorer.[5] The setup program first asks you

[3] More details can be found at www.gams.com under ”Contributed Software”.

[4] Any reasonably recent PC will be large and fast enough to run most models. Before you install GAMS, make sure that you have a couple of hundred megabytes of free space on the hard disk for the GAMS system and for the models and data you are going to work on.

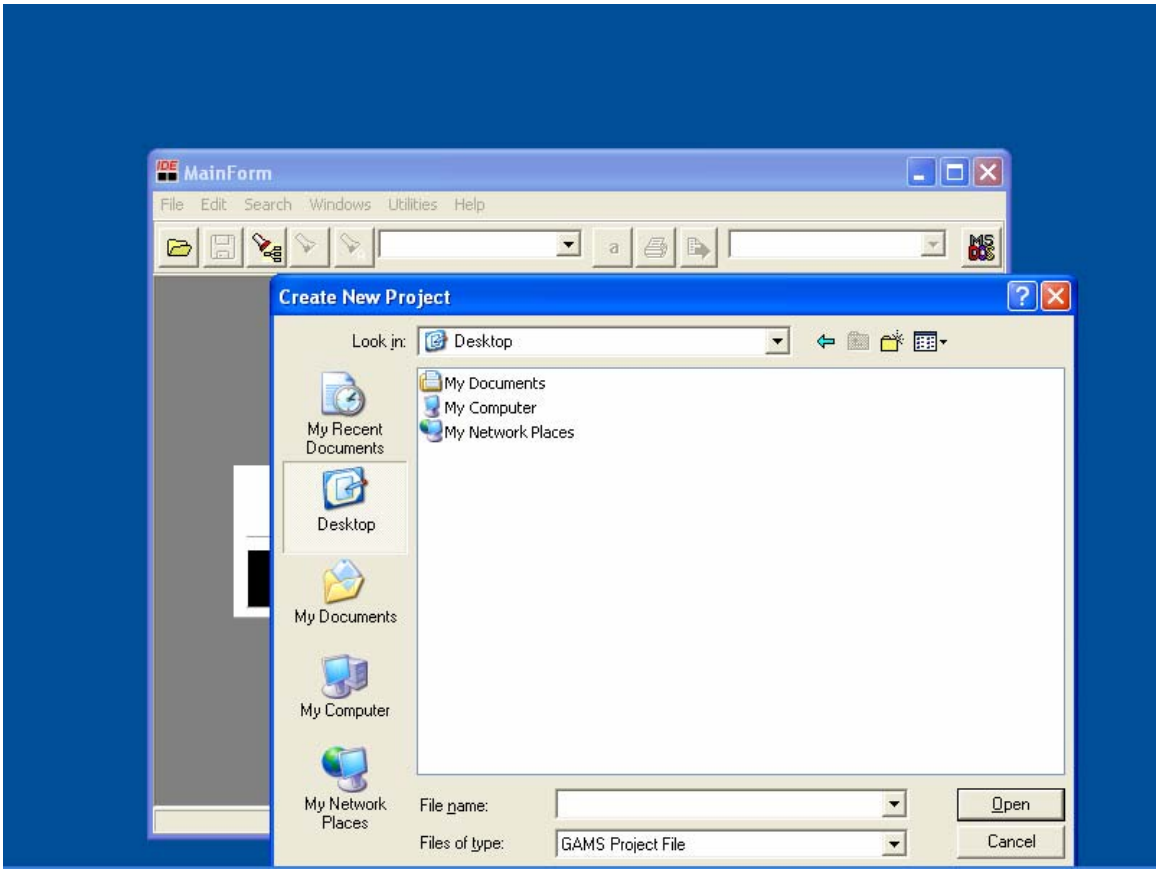
[5] You need to have the rights to install programs on your PC. Contact your system administrator if the computer prevents you from installing GAMS.

for the name and the location of the directory in which GAMS is installed. If GAMS is not already installed on your computer, you can simply accept the default choice. If GAMS is already installed and you want to keep two different versions of GAMS, choose a different name and location for the new version of GAMS.

The installation procedure will also ask you for a license file. If you do not have a license file, choose **`No`**. You can always copy a license file later without having to reinstall the entire system. Even without a license file, you can use GAMS but only to solve small models.[6]

If you have a license file that you want to be part of your GAMS system, answer **`Yes`**. Then browse your files and find the file you want to use. The file name is always **`gamslice.txt`**. If you have bought a GAMS system and have received a floppy disk with a license file, locate your floppy disk drive. If you have a license file on a CD, locate the CD drive and then the directory with the license file. Once found, open the file and the installation program will copy it to the GAMS system.

Choose **`Finish`** at the end to complete the installation of your GAMS system. This should launch the GAMS-IDE interface and your screen should show the following:[7]



[6] This is also called demonstration mode. If you have a license file but it has expired GAMS will also work in demonstration mode only.
[7] IDE stands for **I**ntegrated **D**evelopment **E**nvironment.

If GAMS-IDE does not start automatically, open GAMS-IDE through the icon on your desktop or via the program menu.

Working with GAMS

You are now ready to use GAMS. To work with GAMS, first you need to choose an editor to work with your code. You use the editor to enter your GAMS code and to save a file with a name of your choice and the extension **GMS**. The **GMS** file, also called the input file, is then submitted to the GAMS system, which compiles and executes the GAMS code. GAMS creates an output file with the name you chose for the input file followed by the extension **LST**. This latter file is sometimes called the listing file.

There is generally two ways to do this. Many beginners choose to use GAMS-IDE editor, the Windows-style editor that comes with the GAMS system. When you enter code in GAMS-IDE, it automatically saves a file with the extension **GMS** when you submit the code to the GAMS system. GAMS-IDE also automatically opens the resulting listing file.

The other way is to use a programming editor of your choice. Some would call this the traditional way as this was the only way to use GAMS before GAMS-IDE was developed. Again, you use the editor to enter your GAMS code. After saving the input file, you submit your GAMS code via a DOS-prompt window.[8] You then return to your editor and open the listing file that GAMS has created.[9]

The GAMS-IDE editor

Introduction

Here we will focus on the GAMS-IDE editor. Before we introduce the editor however, we will first do a bit of housekeeping. First, create a project directory on your hard disk. You can do this in Windows Explorer. For example, add a directory at the root of your hard disk **(C:\)** called **Modeling**. Within this directory you can then add subdirectories each time you want to work on a new model. This way you know exactly where your model files are and you separate your GAMS work from the GAMS system files and all other files stored on your computer.

Second, develop a backup system. And use it. For example, save your GAMS code under a new name every time you have introduced major changes. If there are errors in your new code, you can always get back to your starting point by opening the previous file. Also, make sure that you backup your files on an external media so that you can recover your work in case your computer breaks down or gets stolen. In practice this will also help you clean up your modeling directories as you probably want to delete the sometimes large listing files and any temporary files before you backup one or more modeling directories.

---

[8] GAMS must be added to the system path before you can call GAMS from a DOS-prompt. In Windows XP, choose **System** on the Control Panel and then **Environment Variables** on the **Advanced** tab. This opens a window with system variables at the bottom. Select **Path**, click **Edit** and add first a **;** and then the location of the GAMS system directory to the end of the value field (e.g.: **C:\Program Files\GAMS22.0**).

[9] This may sound more cumbersome, but once you have the keyboard shortcuts on your fingertips, you cannot tell the difference and you can furthermore enjoy the features offered by your personal choice of editor. For example, Epsilon has more advanced editing features and has more options for customization than GAMS-IDE.

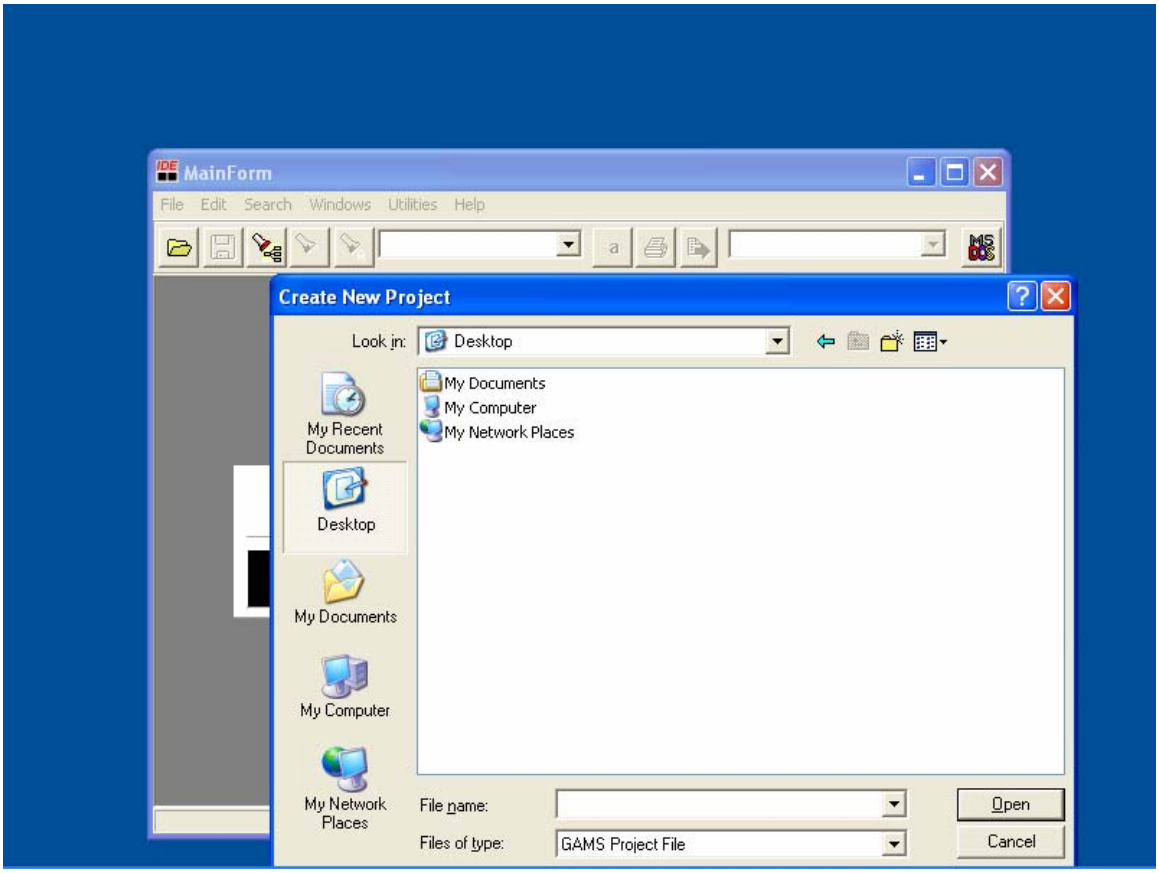
Project files in GAMS-IDE

Now back to the GAMS-IDE editor. A key idea of GAMS-IDE is the project file. This file has two purposes (at least). First, the location of the project file is the starting directory for GAMS. That is, this is where GAMS-IDE stores your files by default and this is where GAMS looks for files when running. For this reason it is (very) important that you always create a new project every time you start working on a new GAMS model. Second, the project file stores any options you may have set for your GAMS runs. We will return to options for GAMS runs below.

In practice, you can think of the project file as a place holder for files and options just like a directory on your hard disk drive. Indeed, the project file is, in some sense, redundant as it just points to your project directory where you have your project files.[10]

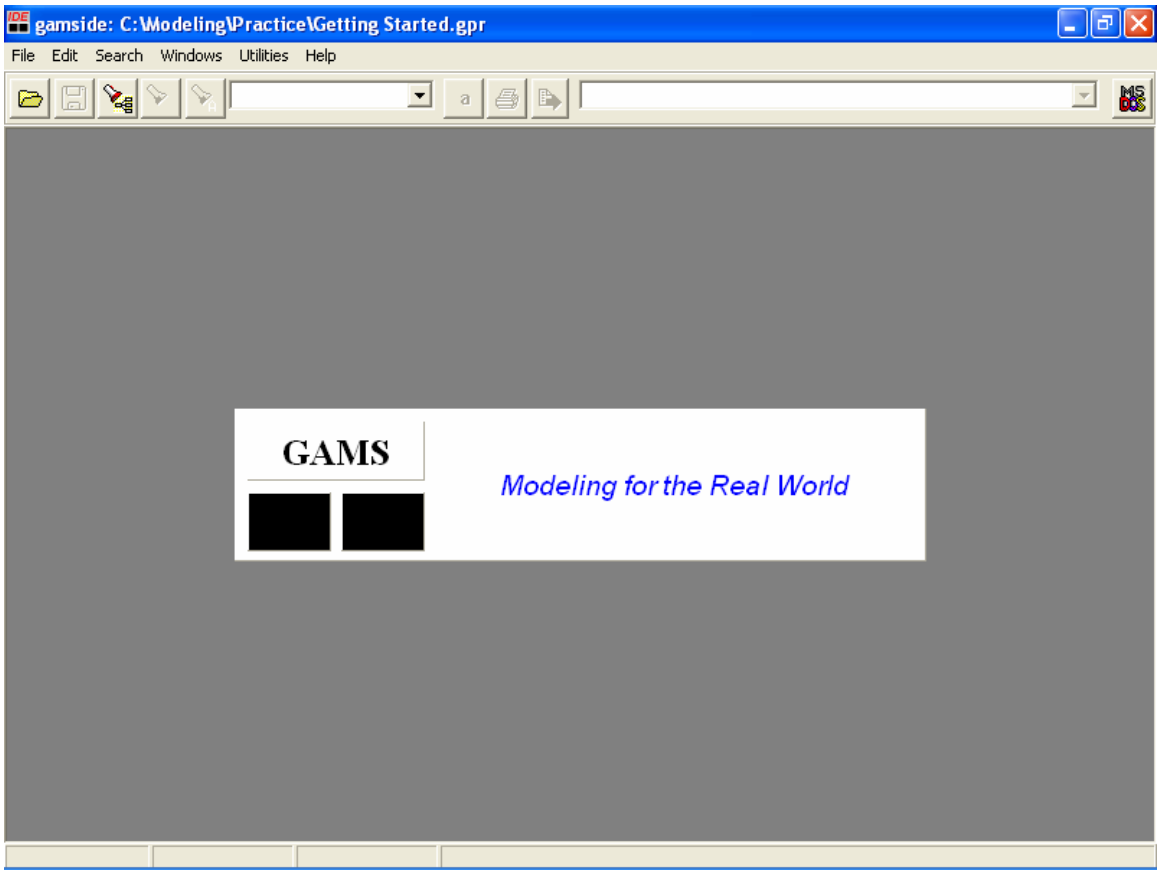
How do you define a new project? The steps are as follows:

1. Create a folder for your new project. For example, create a folder in **`C:\Modeling`** called **`Practice`**, so that your work will be stored in the folder **`C:\Modeling\Practice`**.
2. Open GAMS-IDE (skips this step if GAMS-IDE is already open after installation)
3. Choose **`File | Project | New Project`** and your screen should show the following (skips this step if GAMS-IDE is already open after installation):



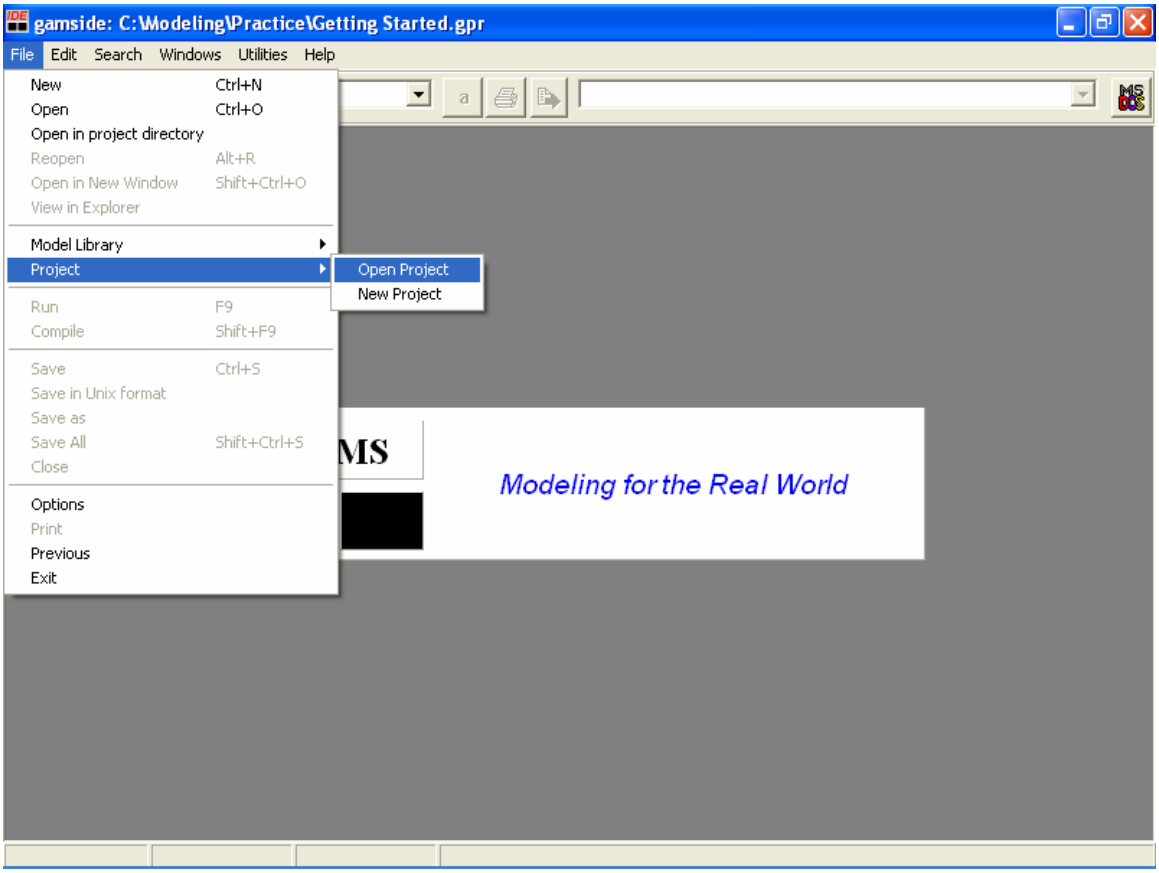
[10] If you get annoyed with this feature of GAMS-IDE, this is not a reason for discarding the idea of using GAMS. Simply get another editor and keep going.

4. Find the location of your project folder. For example, click on **My Computer** and then click on **C:**, then **Modeling** and finally **Practice**.
5. In the box **File name** enter a project name and then click **Open**. GAMS-IDE then creates a project file with the extension **gpr** and stores this file in your project folder. For example, if you enter the name **Getting Started**, GAMS-IDE puts a file called **Getting Started.gpr** in the directory **C:\Modeling\Practice**.
6. You are now ready to work with a GAMS file. Below we will explain how you create your own file and how you open an existing GAMS file. Note that your project file appears in the title bar of your GAMS-IDE window. If you followed the examples provided above, your screen should look like this:



When you close GAMS-IDE, the project file will store information about the GAMS files you have open and their location. Next time you open the project file, GAMS-IDE will open the same set of files. Assuming that GAMS-IDE is not running, the steps are as follows:

1. Open GAMS-IDE
2. Choose **File | Project | Open Project** as illustrated below.



3. You may see a list of recently opened projects. For example, click on **`C:\Modeling\Practice\Getting Started.gpr`**. If you do not see your project on the list, click on **`Open Project`** and find the location of your project folder **`C:\Modeling\Practice`** and then click on **`Getting Started.gpr`**.
4. You are now ready to work with a GAMS file.

Opening and running GAMS files

You are now ready to both create your own GAMS file, to open an existing file, and to open the GAMS model library. We will use a model from the library to keep the introduction simple. Choose **`File | Model Library | Open GAMS Model Library`**. This opens a window with a list of GAMS models. Choose **`THREEMGE`** and GAMS-IDE opens a tab with the GAMS code and adds the file to the project file.

The library model is now ready both for review and for submitting to GAMS. You run the code in GAMS by clicking the icon with the **`red arrow`** to the right of the printer icon. The first time you run GAMS after installation a window may open asking you if you want to choose your default solvers. You can safely answer **`No`** and skip this step for now. We will return to this issue later in the chapter.

Then two windows open: A DOS-prompt window in which GAMS is running and a process window which shows a log of the steps, GAMS goes through to run the code. When GAMS is done, GAMS-IDE automatically opens the listing file with the output of the run. You can

navigate the listing file directly or via the process window which has links to particular locations in the listing file (more about this later in the chapter).

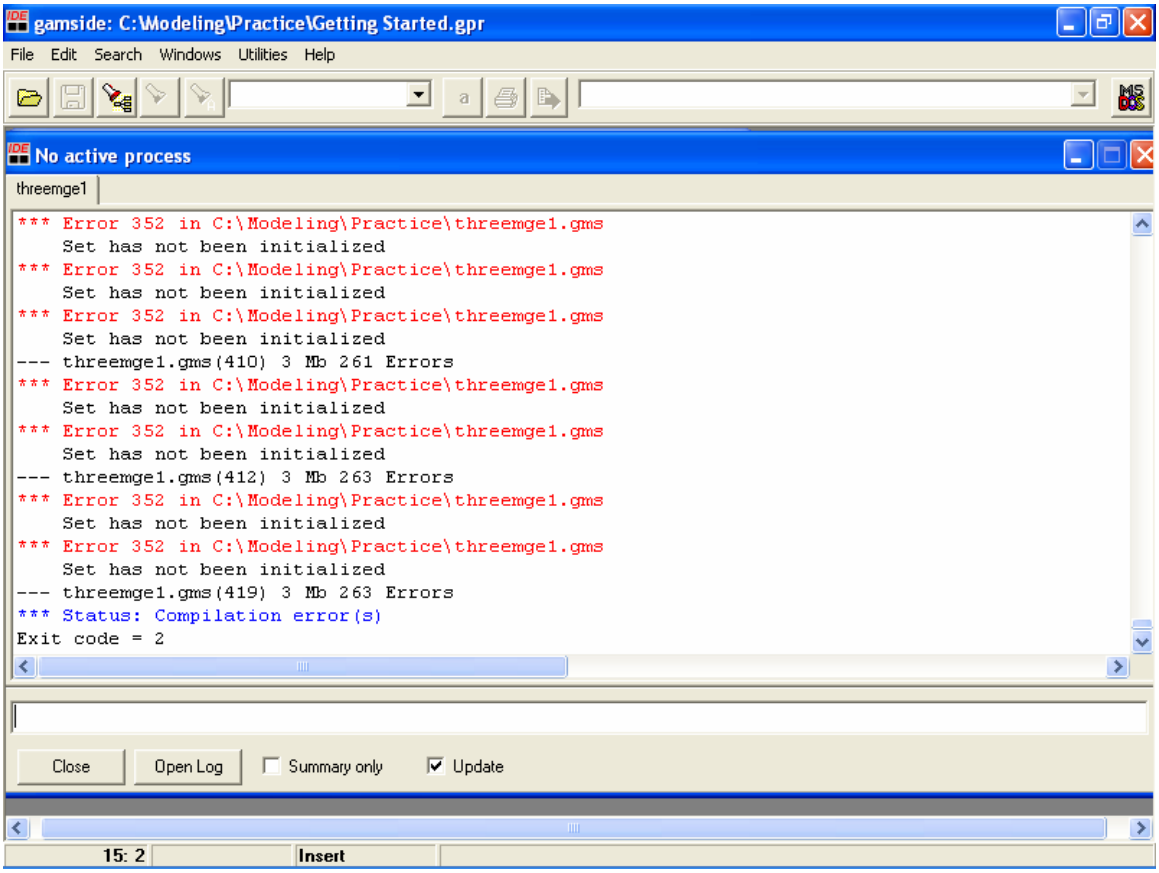
If you want to save your GAMS code under a different name, make sure that you are in the window with the input file (the tab where the file name has the extension **GMS**). Then choose **File | Save as** and enter the file name you want to use.

If you want to create a new file with GAMS code, you have two choices. One is to open an existing file, for example from the library or a file provided by a colleague, and then you can modify the code to your own use. Also, GAMS-IDE allows for the usual cut and paste options included in other Windows programs. After you are done editing, you probably want to save the file with a new name, as described above, to keep a copy of the original code.

The second choice for creating a new file is to choose **File | New**. This opens a tab with the name **Untitled_1**, which is an empty file where you can enter GAMS code. Again, you probably want to save the file under a different name before running GAMS. By default the file will get the extension **GMS**.

Error messages
Now that you are ready to work with GAMS files, it is also likely that you, like everyone else, will make errors in your code. We will now look at how GAMS reports errors in your code and how you can use GAMS-IDE to help you find and fix the errors.

We will use the library model **THREEMGE** to illustrate. First we will save the file under a new name to keep a copy of original error-free code. Choose **File | Save as** and then a file name, for example **THREEMGE1**. Then delete the letter "E" in the GAMS keyword **SET** (line 15 in the code) and hit the **run** button (the icon with the red arrow). Recall that this automatically saves the GAMS code and opens a process window. In this case the process window looks like:

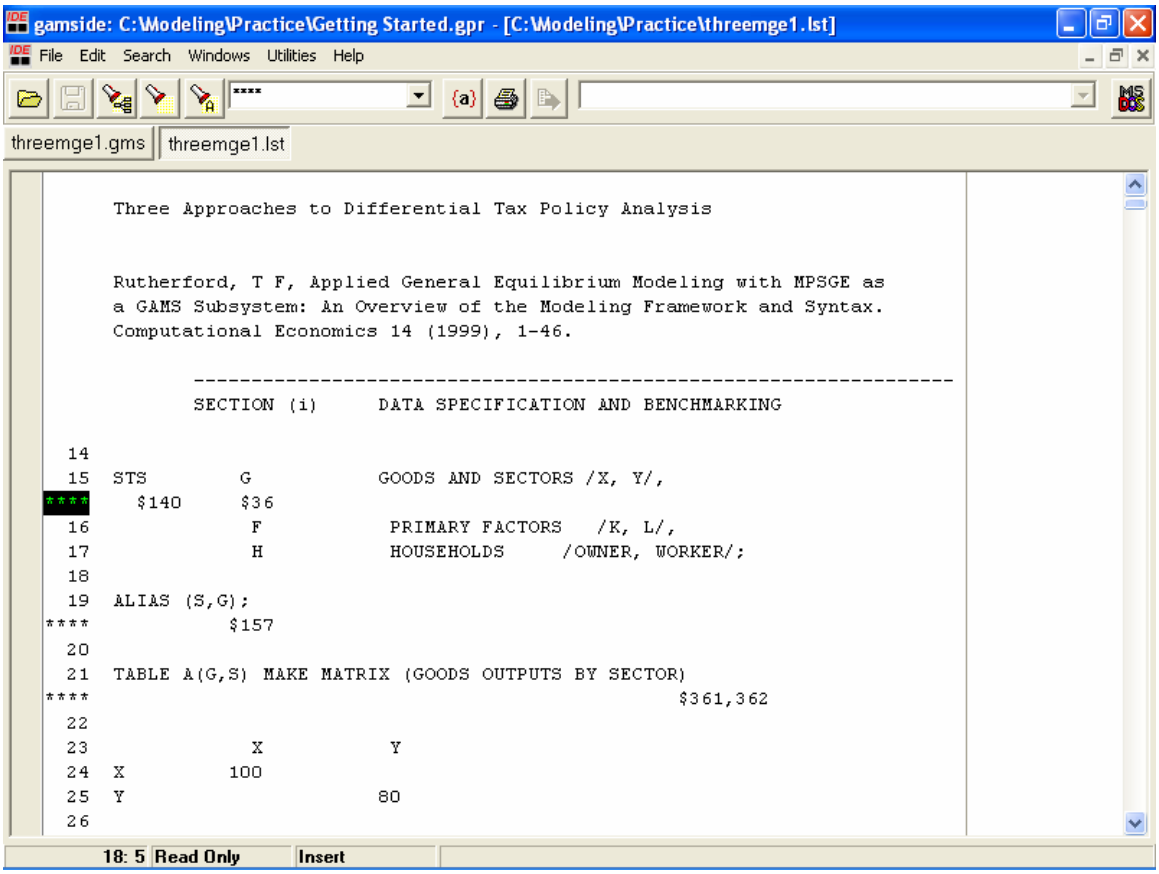


The first thing to note is that more than 200 errors are reported, even as we know that there is only one error in the code. Why? Well, sometimes the first error causes more errors downstream in the code. This immediately suggests that you should always start from the top of the file when fixing code. Also, correct one error at the time unless you know that the errors are unrelated. There are many examples of large models reporting hundreds of errors, all of which are due to one simple spelling error in the top of the code.

A second thing to note is that some lines in the GAMS-IDE process window have the color red. Move to the top of the process window and double-click the very first red line. This takes you to the place in the input file (the one with the extension **GMS**) where the error was made. You can now correct the spelling error and rerun the code to check that the code is error free.

In this case it was easy to correct the error. In other cases, errors may not be obvious. Fortunately, GAMS offers more help if you look at the listing file (the output from the GAMS run stored in the file with the extension **LST**). Click on the tab **THREEMGE1.lst** and make sure that you are at the top of the file. The first part of the listing file is an echo print of the input file. If there are errors in the input file, GAMS adds a coded error messages in the line immediately after the lines with errors. The coded error messages consist of ******** in the first column of the line and is followed by **$** (a dollar sign) below the place in the line above where GAMS thinks there is an error. After the dollar sign follows one or more numerical error codes. These error codes are explained after the echo print and indicates what GAMS thinks is the problem.

You can scroll down the listing file to find the first error, but a safer method is to search for the string ********. In GAMS-IDE, you can do this using the search window to the right of the icon with a flashlight highlighting an **A**.



Enter ******** in the window and click on the icon. You are now at the first line with an error. The first error code is 140. Jump to the end of the listing file and read the explanation. When you are ready to correct the error, click on the tab with input code, correct the error and the rerun the code.[11]

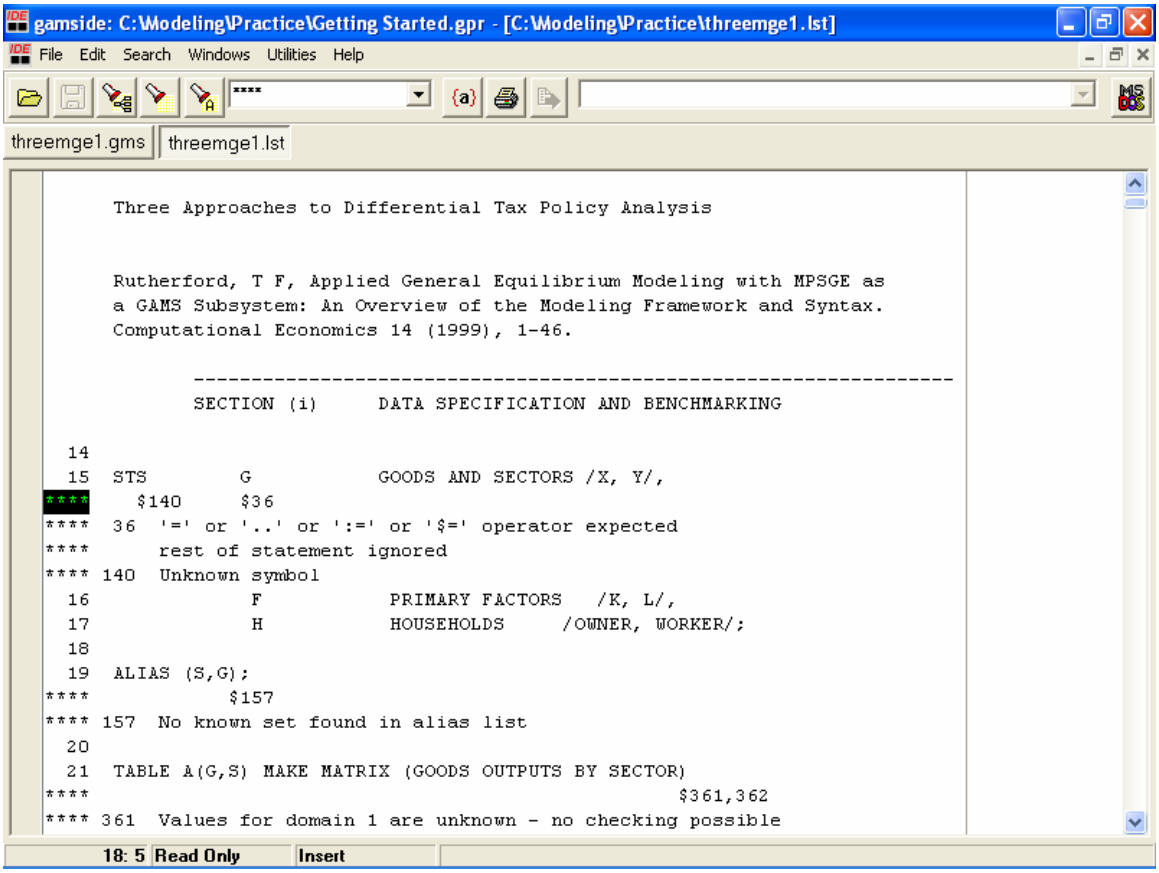
You will make many errors when programming in GAMS. The key is to learn how to find and correct the errors.

Again, the most common error is the missing semicolon. Recall that semicolons are used to separate GAMS statements. Another common error is misspelling of GAMS keywords, such as **PARAMETER** or **EQUATION**. If your editor color codes your GAMS input, this may help avoiding this type of errors in the first place.

[11] Editors other than GAMS-IDE do not provide a process window where errors are highlighted in red. In this case you will go straight to the listing file, examine the error(s) and then correct the error(s) in the input file.

Also, recall that the first error often causes more, related errors. Unless you are sure that two errors are unrelated it is therefore often best to correct the error that appears first, then rerun GAMS before working on the subsequent errors.

Setting GAMS options
You can avoid having to jump to the end of the listing file to find the explanation of the error code. This is done by setting the option **`errmsg`**. Choose **`File | Options`** and then the tax **`Execute`**. Tick **`Use following additional parameters`** and add **`errmsg=1`** in the parameter window. GAMS will now add the explanation of the error code(s) immediately after they occur.



Now that we have introduced options in GAMS, we will also mention a few of the other settings that you can control. The first tab **`Editor`** contains a number of options for the editor in GAMS-IDE. If you like to use tabs when programming to help format your input code nicely and you want to avoid that the tabs are replaced with spaces when GAMS is run, you should make sure that **`Insert Tab`** is selected in the window for **`Tab key action`**.

The fourth tab in the options window is **`Solvers`**. The rows are the list of solvers available in the GAMS system and the columns are the problem types that GAMS can solve. The table tells you what solvers are available for what problem types (indicated by a small square) and what the current selection or default is (indicated by an X). Also, the first column tells you if you have a full license to the solvers or if you have a demonstration license.

No defaults have been selected by the installation procedure. For economic equilibrium models most of the problems will be formulated as a MCP (Mixed Complementarity Problem) or a NLP (Non-Linear Programming Problem). PATH is a state-of-the-art MCP-solver, so this is a good choice for a default. If you do not have a license, another good alternative is MILES. CONOPT is a state-of-the-art-solver for NLP problems. Make your choices by clicking the appropriate cells and then click **OK**.

More about GAMS outputs
As already explained, GAMS creates an output file with the results of the run. This file is often called the listing file and has the extension **lst**. The file contains a lot of information. Sometimes you will probably find that there is too much and in other cases you will probably want more when you really need it.

A typical listing file contains the following outputs in the following order:

- An echo print of the input file
- A reference map
- A listing of the equations included in the code
- Model statistics
- A status report
- A solution report

The first section of the GAMS output is an echo print, or a copy, of your input file. GAMS adds line numbers to your code for future reference. Also, if there are compilation errors in your GAMS model, they will appear in this section.

Next follows a reference map. This is not important to beginners, so it suffices here to say that it is an organized presentation of all the entities in the model, including cross-references. For example, the map provides an answer to question: What parameters are in the model and where are they declared, assigned values and referenced?

The listing of equations is probably more useful. This section shows how GAMS interprets your model and includes the current values of sets and parameters installed into your algebraic model. Therefore, this is the section to look for an answer to the question: Did GAMS generate the model you intended?

The model statistics tells you how big your model is. For example, here you can read how many equations and how many variables there is in your model.

The status report is where it gets really interesting. Here you will find an answer to the question: Did your model solve? You should be cheering if it says **NORMAL COMPLETION** in **SOLVER STATUS** and **OPTIMAL** in **MODEL STATUS**. If there is a report of **ITERATION INTERUPT** or **INFEASIBLE** then there is more work to do.

The final default output is a solution report. Here you will find a listing of your results. At the end of the listing file you may also find the results of any display of GAMS symbols you may have added after the solve statement.

Manuals and documentation
There are many sources of information if you want to learn more about both GAMS and how to use the GAMS-IDE editor. We will only mention a few. The first is the GAMS manual which is

included in the GAMS system that you have installed on your computer. The file format is Adobe Acrobat's Portable Document Format (pdf) so you need to have the free Adobe Acrobat Reader program installed on your computer to access the file.[12] You can access the file via GAMS-IDE: Choose **`Help | GAMS Users Guide`** and the GAMS manual will open. If you are sharing a printer with your colleagues, check with them before hit the print button: The full document is several hundred pages. Also, you will probably mainly be using the manual as a reference to look up details, in which case the search facility in the electronic version is convenient.

The help menu point also offers access to documentation of GAMS-IDE. Here you can read more about all the features not covered here, including options for editing and searching your GAMS code. You can also find technical documents with documentation of the solution algorithms available in GAMS.

Finally, if you are interested in more details, visit the GAMS homepage www.gams.com. Here you will find more info about the GAMS system, software contributed by other GAMS modelers, and a link to a mailing list where you can post messages to other GAMS modelers and search for messages discussing issues you are interested in.

The format of your GAMS code

We complete this chapter by offering a few hints on the format of your GAMS code. There are only a few formatting requirements, so most of the following is largely a matter of personal style.

GAMS is very relaxed about style and punctuation. For example, GAMS ignores blanks and case. Tabs are also ignored, except when data is entered using tables where tab stops are assumed to have a length of 8 characters.

GAMS distinguishes between explanatory text and comments. Explanatory text is optional and is a descriptive text that can be added when you declare sets, set elements, parameters, variables, equations, and models. It is a good habit always to add explanatory text. The text does not have to be in quotes. This is only required if the text contains special characters, GAMS keywords, commas or other characters which GAMS can interpret as a syntax error.

Comments are added in lines beginning with a **`*`** in the first column and several lines may be inserted as long as the first line begins with a **`*`**. You can add a longer comment without a **`*`** in the first column if you instead delimit the text with **`$ontext`** in the line before the comment begins and **`$offtext`** in the line after the comment ends.

Comments are ignored by the GAMS compiler and therefore only appear in the GAMS input file. Explanatory text is carried along with the name of the associated GAMS symbol and a display of the symbol in the output file reproduces the text. This is helpful, particularly in large models with lots of output.

[12] The Adobe Acrobat Reader is available from www.adobe.com.