An Introduction to GAMS with GAMS Studio

(General Algebraic Modelling System)

cgemod

Last revised Oct 2025 (GAMS Studio 1.22.2 and GAMS 51.2.0)

Notes

1. This introduction to GAMS with GAMS Studio reflects the approach used by cgemod; other users, including GAMS, may have different opinions.
2. GAMS Studio is evolving (sometimes quite rapidly) so
    a. features added to Studio or GAMS after the versions identified above will not be covered,
    b. some of the processes detailed here may not be found in earlier versions of Studio and/or GAMS,
    c. some of the processes detailed in this introduction may be out of date.
3. An effort has been made to avoid errors, but some are bound to exist. If you find errors, please email, with the details, to scott@cgemod.org.uk .
4. Some important features of GAMS Studio may not be covered in this introduction; if you have suggestions for extensions to this introduction, please email them to scott@cgemod.org.uk . They will all be considered, although they may not be adopted.
5. Features of GAMS, as used by cgemod, are covered in cgemod's open access courses.

The presence of some repetition is intentional.[1]

[1] Although users are encouraged to work through this guide in order, it is very unlikely they will do so.

Table of Contents

1. Introduction

GAMS

The GAMS (General Algebraic Modelling System) suite consists of the base GAMS module and a collection of solvers. GAMS is a high-level programming language amongst whose objectives is to allow programmers to prepare programmes that are transparent. When a programme is run GAMS converts it into code compatible with a specified solver, executes the programme by calling the solver and writes a report file back. GAMS is attractive because

- it can use a range of specialised solvers without requiring the user to know their specific syntax;
- the separation of data and the logic of a problem allow the size of the problem to be increased without increasing the complexity of the representation;
- the programme is its own documentation; and
- it looks after several common programming problems, like other high-level languages e.g., dimensionality.

"GAMS was developed to [overcome a series of mathematical programming problems] by

- providing a high-level language for the compact representation of large and complex models;
- allowing changes to be made in model specifications simply and safely;
- allowing unambiguous statements of algebraic relationships; and
- permitting model descriptions that are independent of solution algorithms" (Brooke *et al*., 1998, p1).

A major, if not the major, use of GAMS is for optimisation models. Examples of such models are linear programming (LP) and Mixed Integer (MIP) models. CGE models can also be formulated as optimisation problems. Among the solvers available are

Solvers

- BDMLP LP
- MINOS NLP
- CONOPT3/4 NLP optimizer
- CPLEX LP and Mixed Integer Programme solver

- KNITRO nonlinear problems with continuous and/or discrete variables
- LINDO nonlinear problems with continuous and/or discrete variables
- SNOPT Sparse Nonlinear Optimizer
- DICOPT Mixed integer Nonlinear Programming
- PATH Mixed complementarity problem solver
- PATHNLP NLP optimiser)

To use GAMS, you need a programme file. All GAMS programme files are prepared as text files and saved as `[filename].gms`. (Note: it is not necessary to limit the filename to 8 characters, or to avoid spaces, but we find it useful to avoid excessively long file names). The easiest way to prepare a GAMS programme is by using a modern text file editor or that can also read GAMS output files. Standard GAMS output files are written (automatically) to disk as `[filename].lst/.log` files where the filename is the same filename for the `[filename].gms` file..

GAMS is available for is available for MS-Windows, LINUX and Mac OS X (see https://www.gams.com/download/).

GAMS has a video that provides a generic overview of GAMS (Informs 2020 Workshop - Part 1: Introduction to GAMS - YouTube), that you may find informative.

Editors

Many programme file editors available (see http://www.mpsge.org/inclib/tools.htm for some suggested EMACS); and no doubt the suggestion that "*Editors are like mother ducks - once you've identified yours...*" is likely to hold.

We adopt the KISS[2] principle in this guide. This means we set up Studio with basic settings that are more than adequate for most users. The Studio settings provide additional functionality for more experienced users: we do not attempt to meet the needs of experienced users.

2 'Keep it simple, stupid'.

2. GAMS with GAMS Studio for MS-Windows

We recommend GAMS's text editor: **GAMS Studio**.[3]

Installing GAMS

The latest version of GAMS is available as a single exe file. Installation instructions from GAMS and some additional instructions are available from the web site (http://www.gams.com). To read the installation instructions a copy of Adobe Acrobat Reader is required (available free from http://www.adobe.com). It is also valuable to have access to file compression software, e.g., WinZip (http://www.winzip.com), RAR (http://www.win-rar.com), both of which can be used for a period without charge; we will assume you are using WinZip. We assume that you are using Windows 11 in 64 bit mode.[4]

The size of models, i.e., variables and equations, and solvers that can be used depend on the license you hold. For CGE analysis we use the PATH and CONOPT solvers, with a preference for the PATH solver for our main models. GAMS also offers a 'demo' license so that new users can test GAMS before purchasing (https://www.gams.com/try_gams/).[5]

Before installing GAMS, you will need a license file. If your institution has a valid GAMS license you should obtain a copy of the license file. If your institution does NOT have a valid license you will need to purchase a license file (https://www.gams.com/buy_gams/) or apply for a demo license ((https://www.gams.com/try_gams/). If purchasing GAMS, you will need the PATH solver for the CGE courses and the PATH and CONOPT solvers for the SAM estimation course.

If you have queries about GAMS licenses, please contact sales@gams.com . We cannot provide licenses.

---

[3] We have verified the Mac OS X version of Studio, but we do not use Apple Macs and therefore cannot be certain that all the functionality of the MS-windows version is available in the Mac OS X version, and vice versa.

[4] GAMS has announced that support for Windows 10, and Mac OS 13, will cease sometime in 2026.

[5] "A free demo license, valid for 5 months and with **size restrictions**, is included with every GAMS distribution".

- You can generate and solve linear models (LP, RMIP, and MIP) that do not exceed 2000 variables and 2000 constraints
- For all other model types the model cannot be larger than 1000 variables and 1000 constraints.
- Etc." (https://www.gams.com/try_gams/)

The installation process is standard for different versions of Windows. In brief the installation process for Windows 11 is as follows

i) The latest version of GAMS is available from GAMS at http://www.gams.com/download/. It is necessary to select the version of GAMS consistent with your operating system: in this document it is assumed that you are using GAMS for Windows.

ii) Read the installation notes provided on the GAMS website (https://www.gams.com/download/).

iii) Copy the file '`windows_x86_64.exe`'to a directory/folder[6], e.g., `TEMP`, on your C drive.

iv) Run the programme '`windows_x86_64.exe`'. It is WISE for you to install GAMS in a directory on the top level of the C Drive (say C:\GAMS); this is **not** the default chosen by the installation software, but it does have some advantages. (It is assumed for all the exercises etc., detailed in this and related documents that GAMS has been installed into the directory C:\GAMS. If not, you will need to adjust the path appropriately.)

v) Follow the installation instructions as they appear on the screen.

vi) When installing GAMS, you will also, by default, install the editor – GAMS Studio.

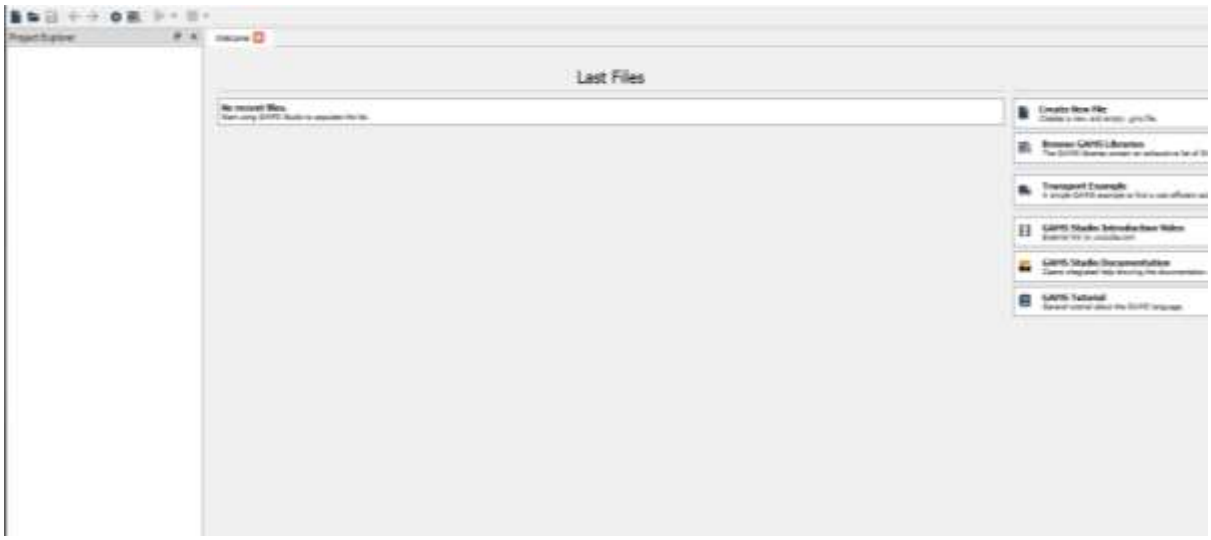
vii) The installation of a Windows version of GAMS will conclude by offering the option to launch GAMS Studio; select this option. Figure 2.1 illustrates how GAMS Studio should appear on your screen.

viii) The final step will be to run a few models to check the installation was successful, this is explained below.

This brief description is NOT a substitute for reading the installation guide or using the Help facility in GAMS Studio.

[6] A 'directory' and a 'folder' are the same; directory is the 'old' DOS name while 'folder' was the name used by Apple in the 1980s. Directory is the name used in this document.

Figure 2.1 GAMS Studio when First Opened



Basic Settings

GAMS Studio provides multiple options for configuring the editor to suit the preferences of individual users. As users gain experience the selected preferences will develop, but when starting it is helpful to have a set of default settings. We assume in this guide that the default settings are those identified in this section.

In Windows Explorer create a working directory. We suggest you call this directory `C:/**/gamstest`; if you plan to do other cgemod courses we suggest `C:/cgemod/gamstest`. The `gamstest` directory will be used to test the installation of GAMS.

In GAMS Studio pen the settings dialogue: use `File>Settings`, or `F7` or the settings cogwheel (⚙). This should open with the 'General' settings tab in view.

1. Select the Workspace tab
    a. Set the Default Workspace to `C:/**/gamstest`; this identifies the user (or the equivalent on your PC).
    b. Deselect all the filters in the Clean-up Workspace section; this clean-up option is useful BUT can have unintended consequences especially for new users.
2. Select the General tab and select the options
    a. '`Open file finds or creates a new project`' is selected.

b. Reduce the 'File count to generate a GSP file' to 0,

c. Set the 'GSP file needs main file' to OFF (no tick),

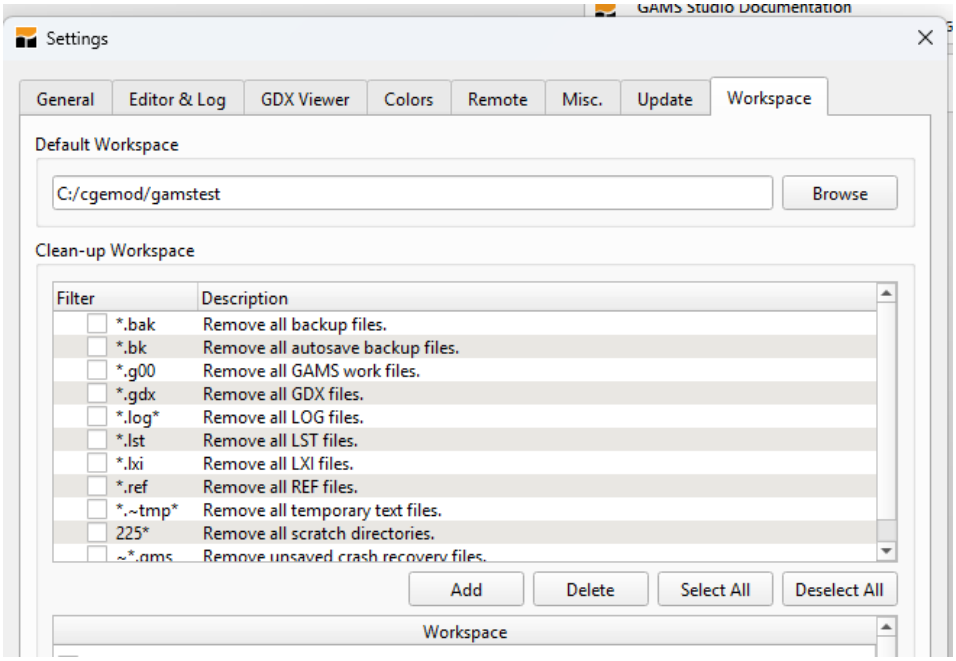
d. Make sure the other settings in this tab are those shown in Figure 2.3.

3. Selected the 'Editor & Log' tab.

   a. Set the font to Courier New and the font size to 12; it helps to use a fixed pitch font. Courier New, and 12 points is a good starting point (the view size can be easily adjusted in Studio without needing to modify the settings).

   b. Make sure the other settings in this tab are those shown in Figure 2.4

Figure 2.2 GAMS Workspace Setting



For the other tabs in the settings tabs the defaults can be left unchanged. At this stage do not worry about understanding the selected settings; they will be more than adequate for some time (and for all the courses offered by cgemod and will be those used in this document and the linked PPT and MP4 files).

We will identify recommended changes as and when they help to illustrate features of GAMS, GAMS Studio and model applications.

As you gain experience you can use the Settings to customise Studio to match your preferences.

Figure 2.3 GAMS Studio General Settings

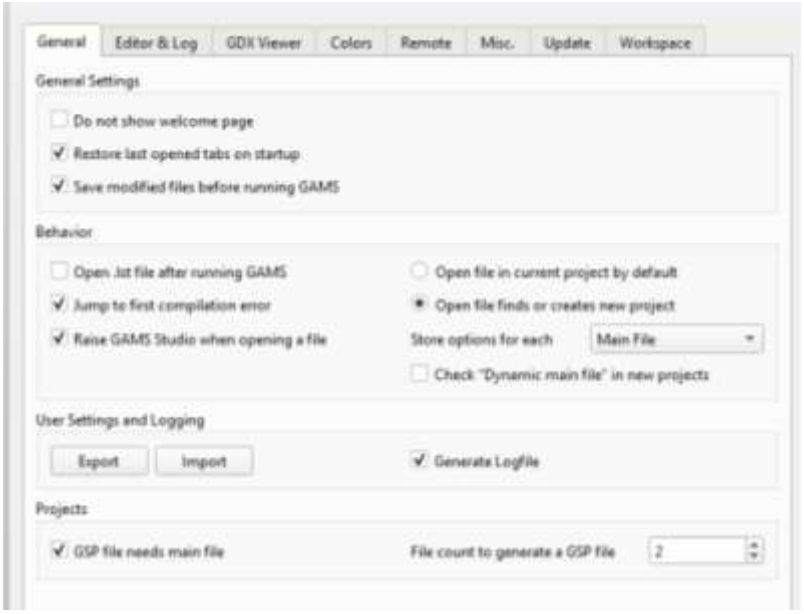
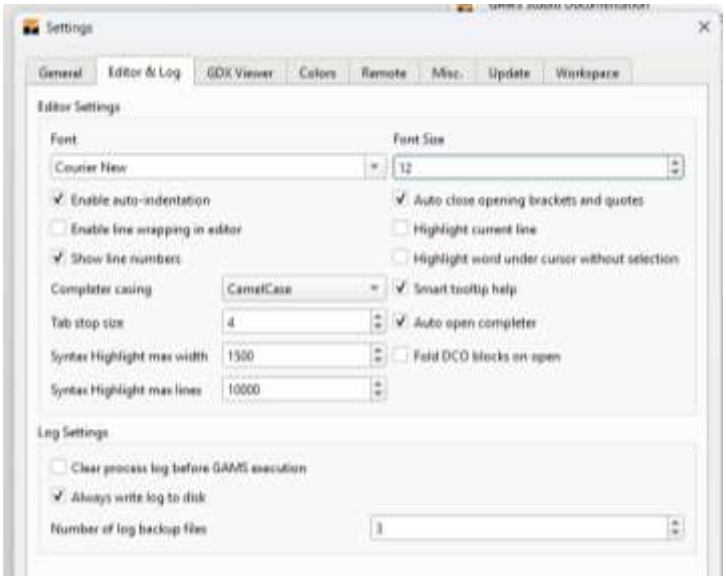


Figure 2.4 GAMS Studio Editor and Log Settings



We will subsequently advise you to change the option 'Open file finds or creates a new project' to the option 'Open file in current project by default'.

3. GAMS Studio

(General Algebraic Modelling System STUDIO)

Using GAMS with GAMS Studio

To use GAMS, you need a programme file. All GAMS programme files are prepared as text files and saved as `[filename].gms`.[7] To prepare a GAMS programme it is necessary to use a text file editor. Historically, it was necessary to use a text file editor to write the programme file and read the resulting (list) file(`[filename].lst`) and to run GAMS from DOS, UNIX, etc. GAMS Studio allows you to do all this from a single Windows based (integrated) interface. This makes things very much easier, but there are one or two features that are initially a little tricky to grasp.

Why advocate using Studio for our courses?[8] The reasons are simple.

1. Studio comes packaged with the GAMS software and therefore only requires the installation of GAMS to give users an editor without needing to install extra programmes.
2. Using a single editor means that all participants are working in the same software environment, which makes it easier to prepare material and provide support.
3. Studio provides all the utilities and routines that are required for cgemod courses.
4. Studio is the future editor for GAMS.
5. Studio can be used on MS-Windows, LINUX and Mac OS X platforms.[9]

More importantly, however, we chose Studio because we like many of the features offered by Studio and therefore, we use Studio for our day-to-day work.

The guidance in this document is intended to supplement the guides available from the GAMS web site (see An Introduction to GAMS Studio - YouTube) or from the link in the 'Getting Started' menu to '**GAMS Studio Introduction Video**'. The GAMS video for Studio

---

7 Notes: (i) It is no longer necessary to limit filenames to 8 characters and to avoid spaces – but it may be good practice to avoid over long filenames and directory names to avoid long paths. (ii) It is also good practice to use multiple directories to avoid difficulties identifying related files.

8 We did not transition to Studio until the software had matured. Early versions of Studio, from 2018 until 2021, did not contain functionality that we consider essential. This has changed and, moreover, the GAMS corporation has been very responsive to feedback, from users, in the evolution of Studio.

9 We have only been able partially to test Studio in Mac OS X and LINUX environments: Mac OS X and LINUX are not our working environments and without using them for long periods we cannot fully test Studio in those environments.

is dated; Studio has evolved appreciably since the video was made. We encourage you to use GAMS's official guides (they may not include all the latest features of GAMS Studio because Studio is an evolving environment).

When you first open Studio, the screen will appear like that in Figure 2.1. (If not select the `View` menu and select `Project Explorer` and `Process Log`.) At the top of the screen is a standard toolbar with dropdown menus, on the left-hand side is a window with the title 'Project Explorer' and in the centre is the 'Welcome' window. The 'Welcome' window has three menus 'Last Files', 'Getting Started' and 'Further Help'. We will explore these components in more depth later in this guide.

The exercises associated with this introduction to GAMS Studio and the pre-course exercises are based on the 'GAMS Tutorial', so while worth viewing it may be better to do so in conjunction with the exercises.

We will start by opening a GAMS programme. For this we will use the Transport Example provided by GAMS and which is used in the exercises that accompany this set of training materials.

A First Programme in GAMS Studio

Before running a first programme in GAMS from Studio it is necessary to do some 'housekeeping'; GAMS needs to know where to find files and where to create files generated when a programme is run. Simplistically, this requires GAMS to know which directory contains the programme files and where files are to be written. This was done when the `Default Workspace` was set to `C:/**/gamstest` (section above).

For now, some considerations about the Workspace will be ignored. Some of the issues ignored here are discussed below where the Project Explorer is considered in more depth.

Now, click on the third item, '`Transport Example`', in the 'Getting Started' menu. The view will change to that illustrated in Figure 3.1 (Figure 3.1 has a few layout adjustments to make it easier to see the content in the figure). There are three important things to note:

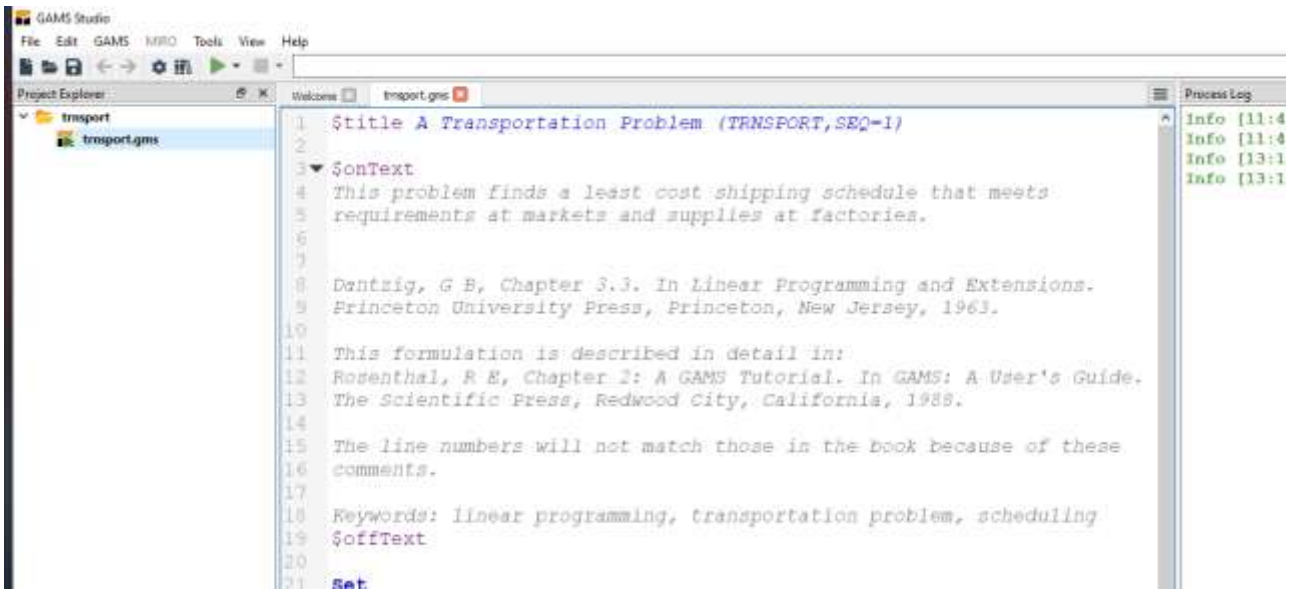
1. The 'Project Explorer' has been populated with a directory/folder symbol '`trnsport`', which identifies that '`trnsport`' is a PROJECT, with a file

'`trnsport.gms`' identified as part of the project. The Project Explorer details the files currently in 'use'.

2. A tab '`trnsport.gms`' has been open in the main part of the screen; the orange symbol indicate that is the file open for editing. This is the window in which files are edited.
3. A 'Process Log' has appeared as a window on the right of the screen. This is where Studio will report on the processes that are and have been implemented.

Running the transport example will allow you to see the basic operation of Studio.

Figure 3.1 A First Programme in Studio



For typical users of Studio there are 8 options for 'running' a programme; these can be accessed in three different ways that produce similar results (see Figure 3.2). To select an option for running the programme click on the small black arrowhead next to the green triangle on the toolbar to open the dropdown menu; a slightly extended menu can be obtained by clicking on the GAMS menu. The 6 standard options are Run (`F9`), Run with GDX Creation (`F10`), Compile (`Shift+F9`) and Compile with GDX Creation (`Shift+F10`). Run with Debugger (`F11`) and Step start with Debugger (`Shift+F11`) are discussed in a later section 15. We do not use the NEOS and GAMS Engine options and therefore do not discuss them.

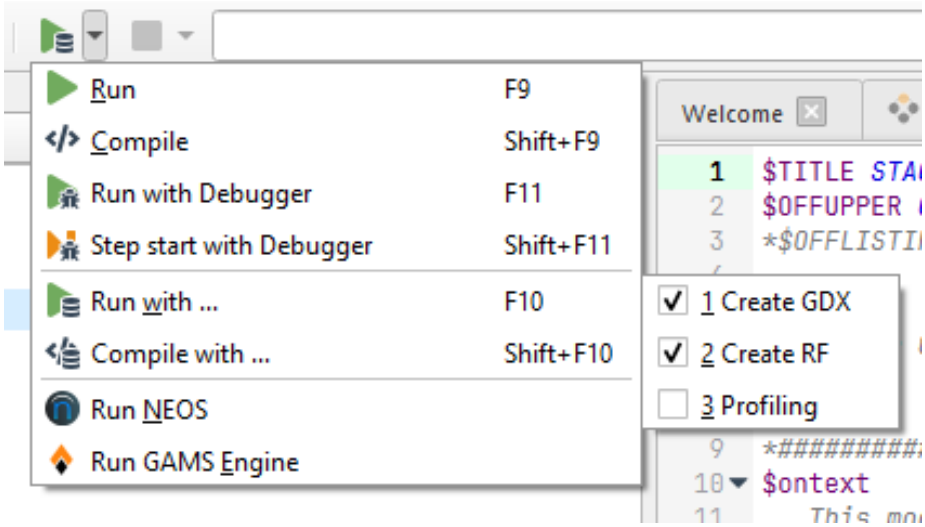
GDX files are discussed in section 9, and reference files are discussed in section 9.

We almost always use `Run with …` (`F10`) with `Create GDX` and `Create RF` **always** selected when running an established model. Less often we use `Compile with ….` (`Shift+F10`) (see section 15 on debugging) with `Create GDX` and `Create RF` selected. `Create GDX` (see section 8) means that a GDX file is automatically generated, while `Create RF` (see section 9) means a reference file is created. We encourage using the keyboard commands identified in the menus.

We rarely use `Run` (`F9`) or `Compile` (`Shift+F9`) since the creation of the GDX file is so valuable and takes no noticeable time, but we do use `Shift+F10`, especially when trying to debug a programme.

Press `F10` (`Run with …`) making sure that `Create GDX` and `Create RF` are selected. The contents of all three windows will change.

Figure 3.2 Running GAMS from Studio



The Project Explorer (Figure 3.3) now identifies four files in the project `trnsport`: the gms file – `trnsport.gms` – that contains the programme, the lst file – `trnsport.lst` – that contains a listing of the output when the programme ran; a gdx file – `trnsport.gdx` – that contains information about the data, variables, equations, results etc., and the reference file – `transport.ref` - that 'works' as index to the project generated by GAMS. Note how the file identified in bold is file/tab open in the editor window.

But there are seven files in the working directory (Figure 3.4). These are : the `gms`, `lst`, `gdx` and `ref` files. The other three are an index file associated with the `lst` file – `trnsport.lxi`; the log file – `trnsport.log` – records the processes undertaken; and a project file – `transport.gsp`.

GDX files are discussed in section 8, reference files in section 9 and project files in section ??

Figure 3.3 The Transport Problem in Studio – Project Explorer

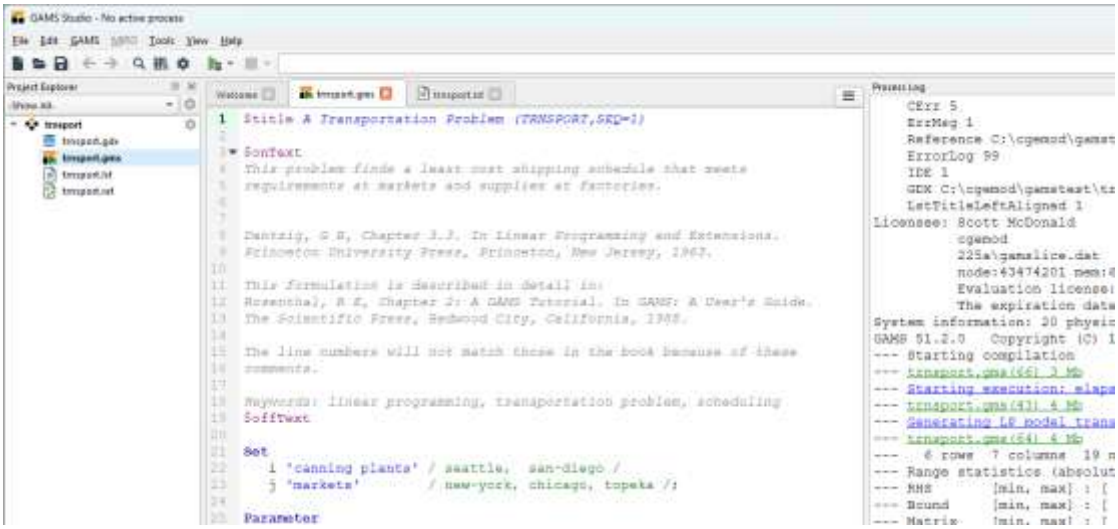
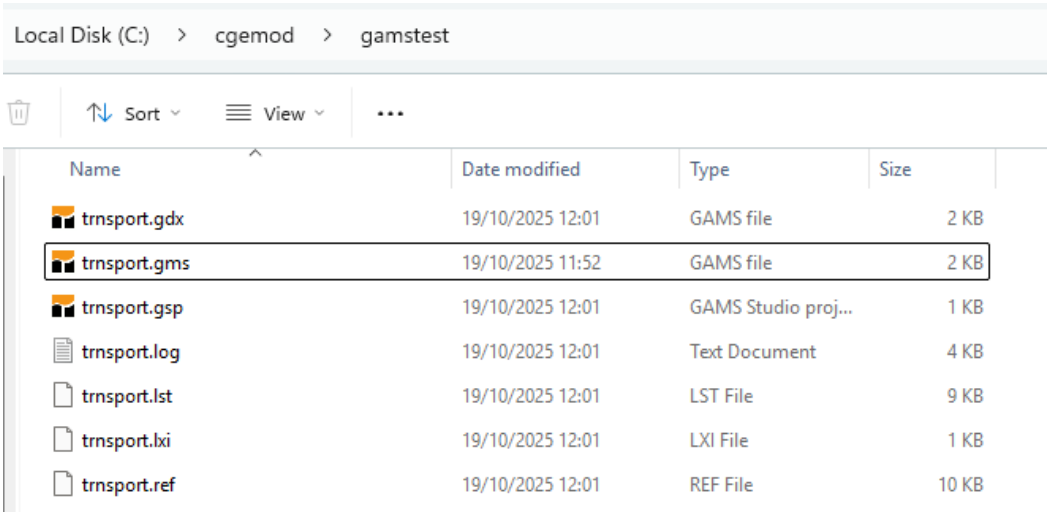


Figure 3.43 Transport Problem files in Windows Explorer



It is important to note that **GREEN** triangle on top of the Studio symbol is associated with the gms file – `trnsport.gms`. The **GREEN** triangle identifies the '**main file**' in each currently open PROJECT. The active PROJECT is identified by the directory that is in **bold**. If the user chooses to run a programme, e.g., by using F10, then the programme that will run will be the **main file**, i.e., the file with the **GREEN** triangle, in the **active** PROJECT, irrespective of the file currently being viewed in the EDITOR window.

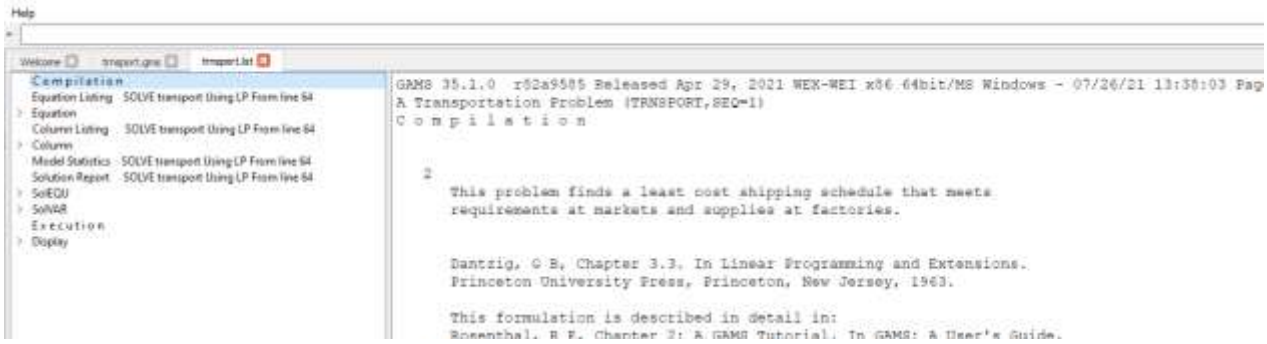
Figure 3.4 The Transport Problem in Studio – Process Log

```
--- Starting compilation
--- trnsport.gms(66) 3 Mb
--- Starting execution: elapsed 0:00:00.043
--- trnsport.gms(43) 4 Mb
--- Generating LP model transport
--- trnsport.gms(64) 4 Mb
---   6 rows  7 columns  19 non-zeroes
--- Executing CPLEX (Solvelink=2): elapsed 0:00:00.187

IBM ILOG CPLEX   35.1.0 r82a9585 Released Apr 29, 2021 WEI x

--- *** This solver runs with a community license.
--- GMO memory 0.50 Mb (peak 0.50 Mb)
--- Dictionary memory 0.00 Mb
--- Cplex 20.1.0.1 link memory 0.00 Mb (peak 0.00 Mb)
--- Starting Cplex

Version identifier: 20.1.0.1 | 2021-04-07 | 3a818710c
CPXPARAM_Advance                                 0
CPXPARAM_Simplex_Display                         2
CPXPARAM_Threads                                 1
CPXPARAM_MIP_Display                             4
CPXPARAM_MIP_Tolerances_AbsMIPGap                0
Tried aggregator 1 time.
LP Presolve eliminated 0 rows and 1 columns.
Reduced LP has 5 rows, 6 columns, and 12 nonzeros.
Presolve time = 0.00 sec. (0.00 ticks)

Iteration      Dual Objective            In Variable
     1             73.125000    x(seattle,new-york) demand(
     2            119.025000     x(seattle,chicago)  demand
     3            153.675000    x(san-diego,topeka)   deman
     4            153.675000  x(san-diego,new-york)  supply

--- LP status (1): optimal.
--- Cplex Time: 0.00sec (det. 0.01 ticks)


Optimal solution found
Objective:          153.675000

--- Reading solution for model transport
--- Executing after solve: elapsed 0:00:00.622
--- trnsport.gms(66) 4 Mb
--- GDX File C:\Users\scott\Documents\GAMS\Studio\workspace\
*** Status: Normal completion
--- Job trnsport.gms Stop 07/26/21 13:38:03 elapsed 0:00:00.
```

Multiple projects can be opened in the Project Explorer. The current **active** project will be the project in bold and moving between projects can be achieved by double clicking on any file in the project that is to become the active project. The contents of the 'Project Explorer' window will be examined in more depth below.

The 'Process Log' window records what has happened when a programme is run. This programme ran to a 'Normal completion' (see the second last line). Clicking on BLUE items in the log opens the l(i)st file – `trnsport.lst` – and moves the cursor to specific places in the file. Items in **GREEN** open the identified file in the editor window, e.g., '`trnsport.gms`' and '`trnsport.gdx`', and where appropriate at the point in the linked file references in the lst file. Click on the third last line of the Process Log – '`GDX File ….`'; this will open the GDX file. Later there will be items in **RED** – syntax errors – that when clicked on will open files in the editor window at the point in the file where the error exists.

Figure 3.5 The Transport Problem in Studio – Editor



Finally, there is the editor window. By default, the editor window will display the `l(i)st` file – `trnsport.lst` – in two parts. The left-hand part is an index file that can be used to navigate the list file, e.g., `SolVAR` links to the report of the solutions for the variables (try double clicking on `SolVar` and then clicking on the arrowhead to open a sub menu and click on one item in the sub menu. As you get to know programmes this will be one of several ways to explore what has happened when running a programme.

You can adjust the layout of the windows in Studio to suit your preferences; this guide will continue with the default configuration.

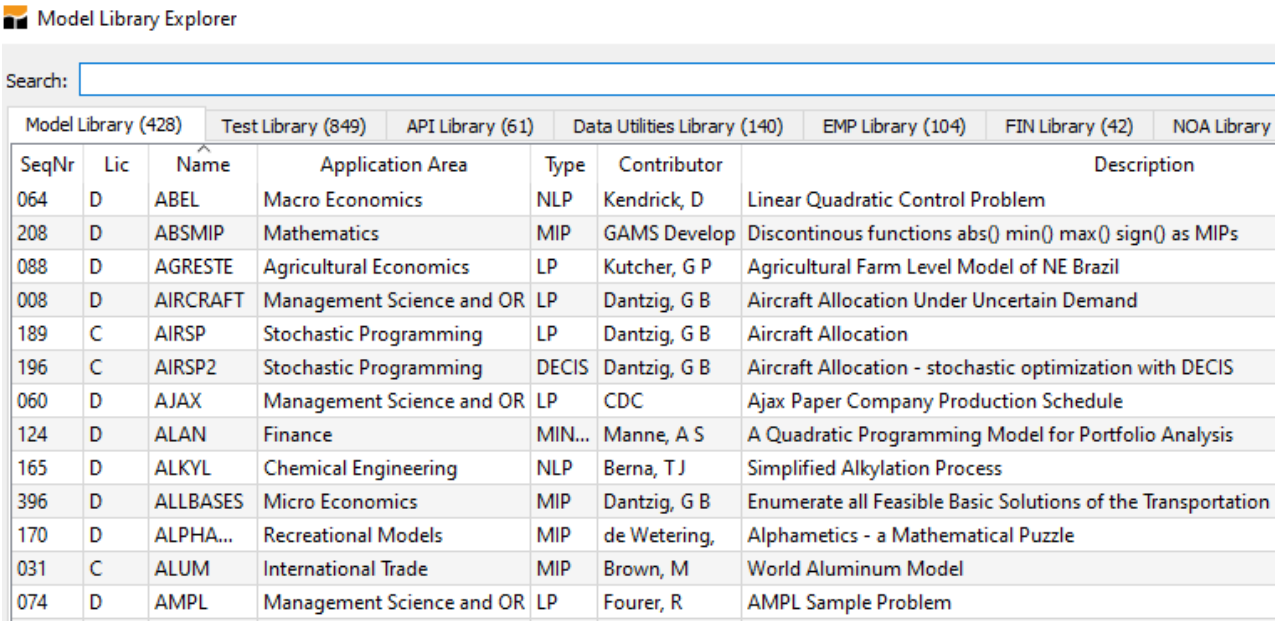
4. Testing a GAMS Installation

To test your GAMS installation, you can run several sample programmes that are provided by GAMS. These programmes are stored in one of the Model Libraries included when GAMS was installed.

In Studio select `GAMS > Model Library Explorer` or press `F6`. The window displayed in Figure 4.1 will appear. Each tab links to a specific library, each of which contains many sample programmes. To test the installation, we will select and run 4 programmes from the `Model Library`. These are

1. `trnsport` (LP :objective value: 153.675).
2. `chenery` (NLP: objective value: 1058.9).
3. `bid` (MIP: optimal solution: 15210109.512).
4. `procsel` (MINLP: optimal solution 1.31164890297).
5. `scarfmcp` (MCP: no objective function).

Figure 4.1 Model Library Explorer



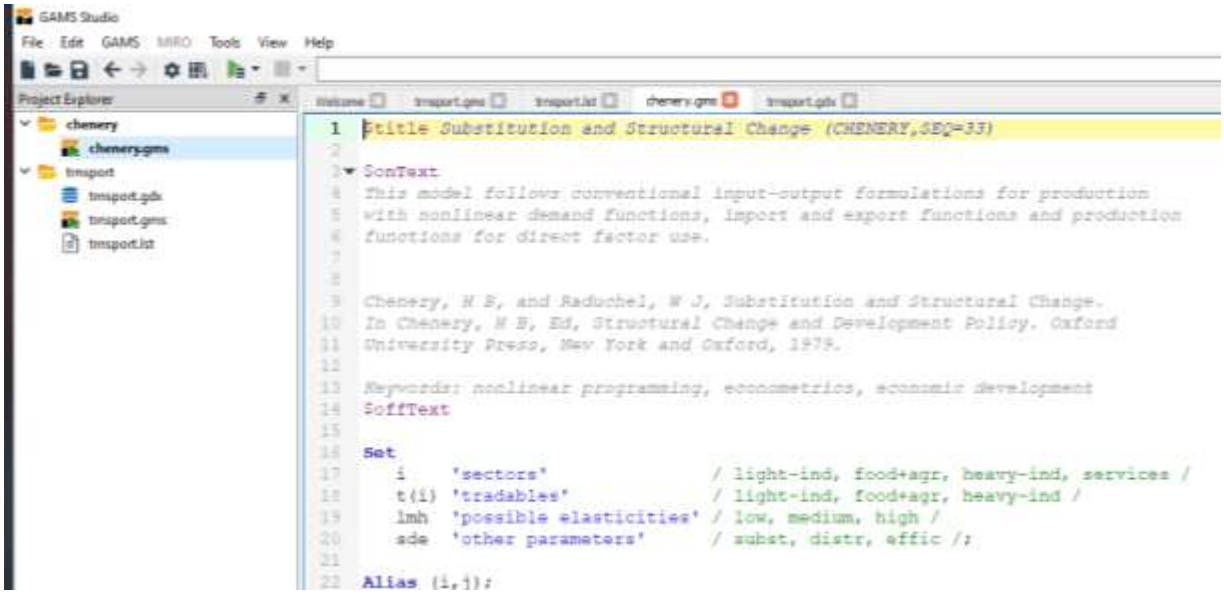
SeqNr	Lic	Name	Application Area	Type	Contributor	Description
064	D	ABEL	Macro Economics	NLP	Kendrick, D	Linear Quadratic Control Problem
208	D	ABSMIP	Mathematics	MIP	GAMS Develop	Discontinous functions abs() min() max() sign() as MIPs
088	D	AGRESTE	Agricultural Economics	LP	Kutcher, G P	Agricultural Farm Level Model of NE Brazil
008	D	AIRCRAFT	Management Science and OR	LP	Dantzig, G B	Aircraft Allocation Under Uncertain Demand
189	C	AIRSP	Stochastic Programming	LP	Dantzig, G B	Aircraft Allocation
196	C	AIRSP2	Stochastic Programming	DECIS	Dantzig, G B	Aircraft Allocation - stochastic optimization with DECIS
060	D	AJAX	Management Science and OR	LP	CDC	Ajax Paper Company Production Schedule
124	D	ALAN	Finance	MIN...	Manne, A S	A Quadratic Programming Model for Portfolio Analysis
165	D	ALKYL	Chemical Engineering	NLP	Berna, T J	Simplified Alkylation Process
396	D	ALLBASES	Micro Economics	MIP	Dantzig, G B	Enumerate all Feasible Basic Solutions of the Transportation
170	D	ALPHA...	Recreational Models	MIP	de Wetering,	Alphametics - a Mathematical Puzzle
031	C	ALUM	International Trade	MIP	Brown, M	World Aluminum Model
074	D	AMPL	Management Science and OR	LP	Fourer, R	AMPL Sample Problem

The first programme – `trnsport` – has already been run so all that you need to do is verify that a 'Normal Completion' has been recorded and that an 'Optimal solution found' with an objective value of 153.67500. These can all be found in the 'Process Log' or the 'Solution Report' in the file `trnsport.lst`.

The selection of '`Open file finds or creates a new project`' in the General tab of the settings dialogue box (`F7`) ensures you continue to work in the `Default Workspace`.

In the Model Library Explorer, select the tab '`Model Library`' and search for `chenery`; clicking on the column labels sorts the columns while using the Search box finds a file if you know its name. Click anywhere on the line for the model `chenery`; the results are shown in Figure 4.2.

Figure 4.2 Model Chenery in Studio



Note the following:

1. The programme – `chenery.gms` – has opened in the editor
2. The project/group `chenery` has been added to the Project Explorer
3. The project `chenery` has been defined as the active project – the project folder is in bold

Run the programme '`chenery.gms`' using `F10` (or the equivalent) and note the changes in Studio. Verify that a 'Normal Completion' has been recorded and that an 'Optimal solution found' with an objective value of 1058.9199. HINT: look in lst file.

Repeat the process for the models '`bid`' (Opt. value 15210109.512), '`procsel`' (Opt. value 1.923099) and '`scarfmcp`' (No opt. value) (NB: `procsel` and `scarfmcp` are in

the main `Model Library` but that the search option searches across all libraries so it is easily found); in each case verify that a 'Normal Completion' has been recorded and that an 'Optimal solution found' with an objective values given above.

Note how for each programme a new project is created. This shows that in Studio multiple projects can be open at the same time; how the Project Explorer viewer and Projects operate with Studio is important to understand.

For the models in cgemod courses we will recommend having the files for each project in a single directory.

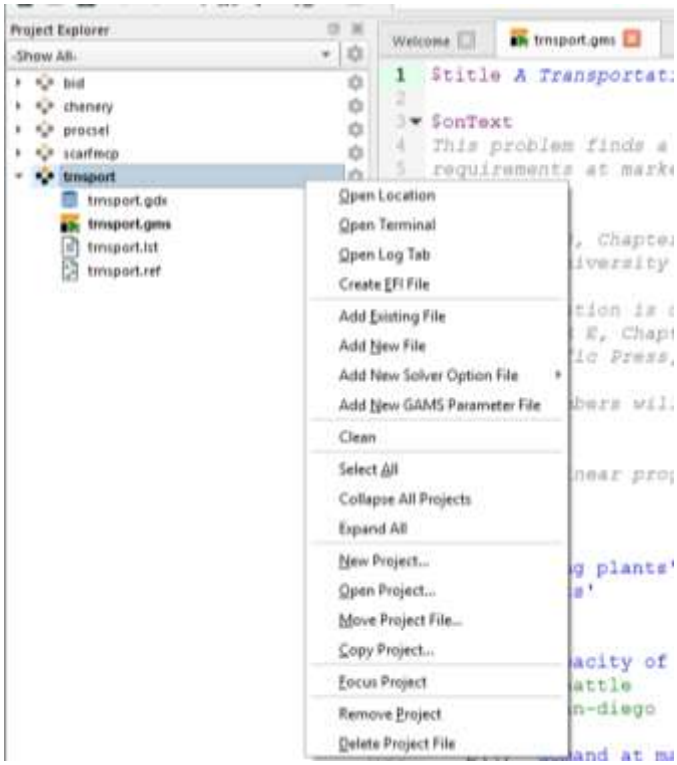
5. Project Explorer, Projects and Project Files

The Project Explorer and the functionality it offers user is a powerful tool for using, controlling and managing projects and associated files. This section only scratches the surface of what can be done, but it will get new users started; indeed, some of the functions will only become useful once users achieve a deeper understanding of GAMS.

5.1 Project Explorer and Projects

There are many options available to the user within the Project Explorer in Studio. These are typically accessed by right clicking on a project name or a file within the project visible in the Project Explorer window; the options available will depend on the item that is right clicked. Some of these options are self-explanatory, e.g., `Open Location`, `Add New File`, `Expand All`; you should choose them and observe the consequences. Some of the options are discussed further below.

Figure 5.1 Right Click on Project Name

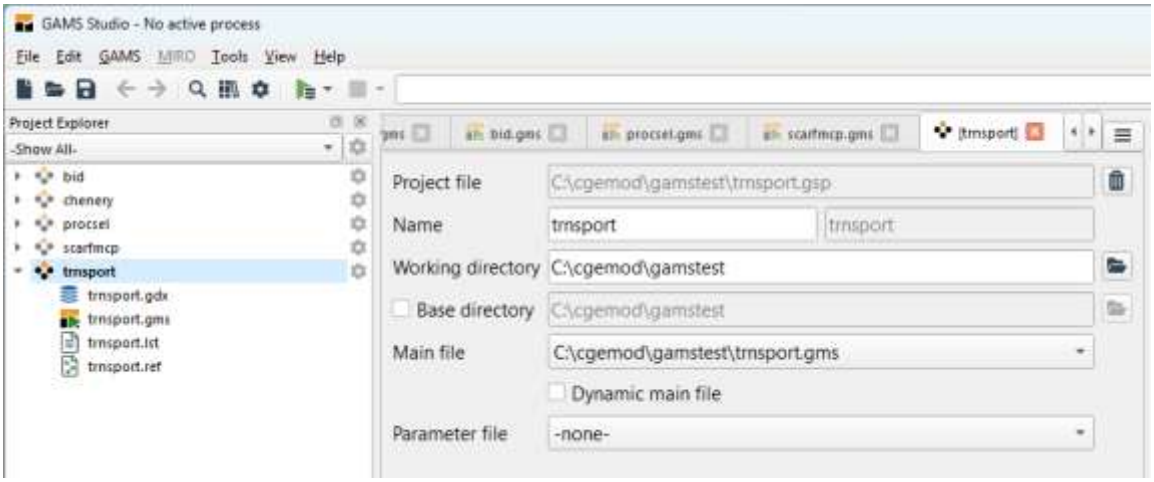


NB: Remove Project removes the project from the Project Explorer, i.e., closes the project, but does not delete the project from your PC.

Details about a project can be found by clicking on the Settings wheel to the right of the project name. This provides details about the project in the edit view (See Figure 5.2). This tells the user the name of the project (GSP) file with its path for the project file, and the name of the current `Main File`. It also identifies the path for the `Working directory` and the `Base directory`; in this case the two directories are the same, which will be the case for most users and applications. They can be different but that is advanced topic that is beyond this introduction.

Until you become an advanced user it is recommended that you leave the 'Base directory' the same as the 'Working directory'.

Figure 5.2 Project Options View



Controlling the Choice of Active Project

The active project is always the project directory whose label is in **bold**. You can change the active project by double clicking on a file in the project you want to be active. If you change the active project, you will also change the file that will be run when pressing `F9/F10`, i.e., the **main** file with the **GREEN** triangle in the active project 'directory'.

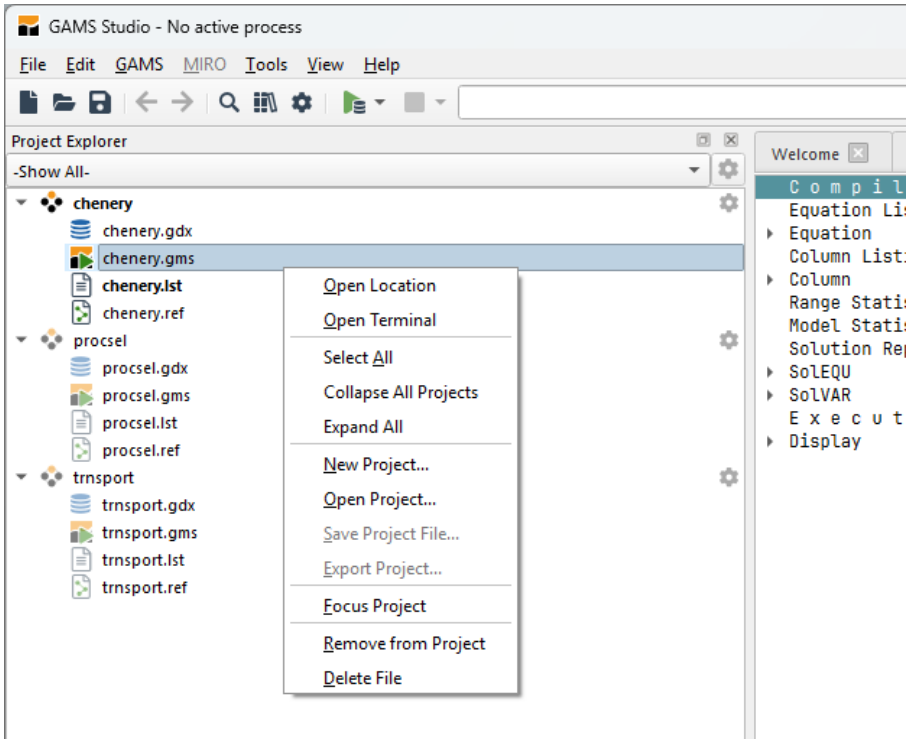
Moving Files between Projects and Changing the Main File

You can move files between projects in the Project Explorer. For instance, assume you want to move the file `procsel.gms` into the project `chenery`. Click on the file '`procsel.gms`' and drag it into the `chenery` directory. You now have two gms files in the `chenery` directory, one is the 'main' file – `chenery.gms`. If you want to make

'`procsel.gms` the 'main' file, right click on the file `procsel.gms` and choose the option – `Set as Main File`. Note how the **GREEN** triangle moves to the file `procsel.gms`.

The main file of the active project will always be the programme that is run when choosing a run option, e.g., F9 or F10 or F11.

Figure 5.3 Right Click on File Name



NB: Remove from Project closes the file Project Explorer; it does not delete the file from your PC.

SUGGESTION: you can have multiple `gms` files in a single project and switch between them using the `Set as Main File` option. While this is viable, and GAMS will allow the user to do so, we discourage the practice especially for new users. We suggest that each project should have one, or at most a few, `gms` files. In the Practical CGE, single country CGE and global CGE courses we have only one `gms` file in each project; in the single country and global CGE courses we have two projects linked to the same directory with a single `gms` file in each project. (This is consistent with our practice when using our advanced models for conducting research and/or consultancy: we only have one `gms` files in

each project but two `gms` files in one directory with two projects. This has some benefits when using command line options, e.g., save and restart.)

Open a File from Project Explorer

Files can be opened from Project Explorer by right clicking on the project and selecting the file to be opened, e.g., `Add existing file` and `Add new file`. '`Add existing file`' takes you to the location/directory where the project's files are stored and allows you to open an existing file; note that you will need to select the file type (bottom right of the dialogue box) or type in the file name. **HINT**: in the File name box the wildcard option (*) is useful, e.g., `*.inc, *.gms`, etc., makes only those files with specified file types visible. '`Add new file`' takes you to the location where the project's files are stored and allows you to create a new file of the selected type; note that you will need to select the file type in the bottom dialogue box.

Our preference is to open files using the keyboard shortcuts `Ctrl+O` or `Ctrl+Shift+O`, or `Ctrl+N`, i.e., `File > Open` or `File > Open in a New Project` or `File > New` menus. When a programme is run Studio opens certain files by default whereas others need to be selected. (Space below to the special and useful files.)

Open Terminal

Open terminal opens a 'command prompt' for use when running programmes in command line mode. It is useful but an advanced feature not required for any cgemod course.

Open log tab

This brings the log tab for the selected project to the foreground of the Process Log, NB it does have to be for an active project.

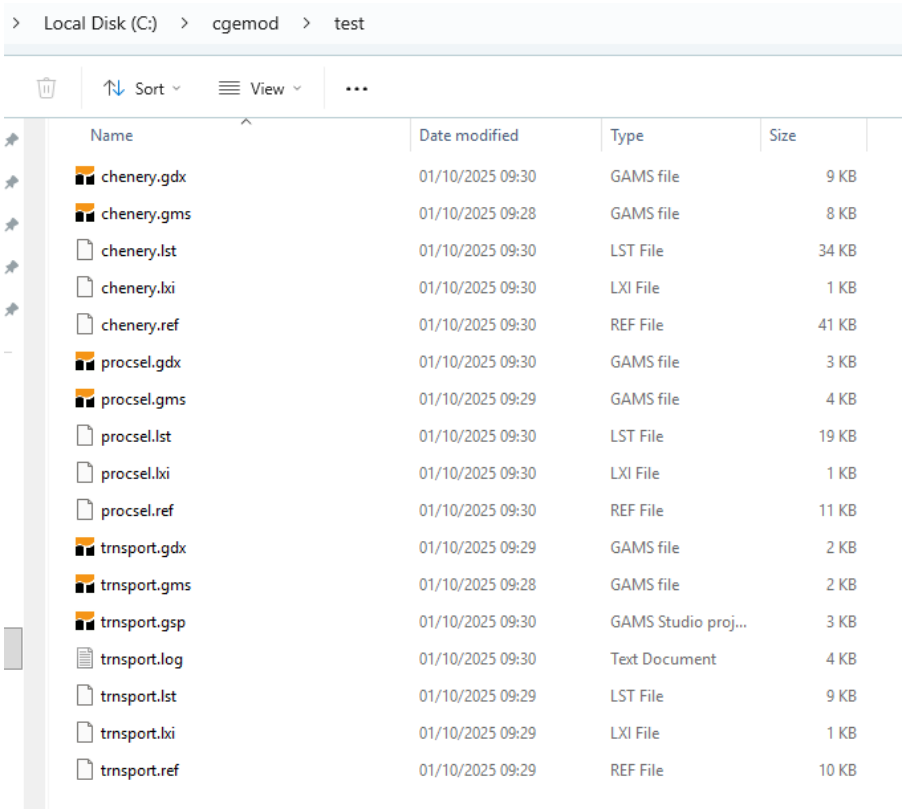
Open Location

The option to open location is very useful. If you right-click on any label in the Project Explorer and select '`Open location`' the window (in windows explorer) in Figure 5.3 will open.

This illustrates something that is useful for this introduction to Studio that we consider an unwise practice when using Studio for real. Note how all the files that were opened when installing and testing GAMS and Studio were saved to a single directory, e.g.,

`C:\cgemod\gamstest` or the Default Workspace directory– use `Open Location` to verify this. The `workspace` directory was created when GAMS was installed and is required as a default.

Figure 5.4 Open Location from Studio



WARNING: It is easy when first using Studio to end up with files relating to a single project in different groups/project directories in the Project Explorer. It is important to keep all the files relating to a project together (see above). Hence, we advocate keeping all files relating to a project in the same project directory in Project Explorer.

This is useful for this introduction to GAMS Studio, but we believe that using a **default workspace directory** for your own work is NOT good practice.

Each project should have its own dedicated directory; to ensure all files relating to that project are in one location. This may be a sub directory.

This reflects the fact that each project can have many files, which means that finding files can become difficult and labourious if they are all stored in a single directory. Indeed, in

our work the files for each project are typically stored in sub-directories within a master directory to aid file management. This is the practice adopted in all cgemod courses other than the 'Introduction to Practical CGE Modelling'.

5.2 Projects

The role of a project file (e.g., `myProject**.gsp`) in Studio has several dimensions in part because a project file must meet the needs of user with different needs. The discussion in this section is aimed at the needs of 'typical' and 'new' users of GAMS. **In many, if not most, circumstances Studio deals automatically with project files.**

In day-to-day applications with GAMS models, especially with how cgemod organises these applications, project files provide a simple method for moving between different parts of the modelling process. In addition, they make it possible to have multiple projects available directly from the `Project Explorer` in Studio. These applications are discussed below under the heading of *Existing GAMS file*.

A common practice is to start working on a new application by adapting one of your existing models or by starting from a model someone has provided. In such cases users can establish a new directory and copy the files into that directory. In such cases GAMS Studio can be left to manage the project file, as described in the *Existing GMS File* section below.

It is different if the user wants to establish a `New Project` without an existing GMS file. This is the case for cgemod courses where the GAMS code and data files are supplied from a USER Model Library.

Existing GMS File

Assume you have been provided with a standalone GMS file, e.g., `trnsport.gms`, and have saved that file in dedicated directory, e.g., `C:\cgemod\trans`. If you open the file `trnsport.gms` in Studio using either `File>Open`, or from Windows Explorer, Studio automatically generates a GSP file called `trnsport.gsp`. However, this may not be immediately visible in Windows Explorer. However, if you run the programme using F10 then the file called `trnsport.gsp` should be visible.

This is because we used the `General tab` of the `Settings` to choose '`GSP file needs main file`' and set the '`File count to generate GSP file`' to 2 (the

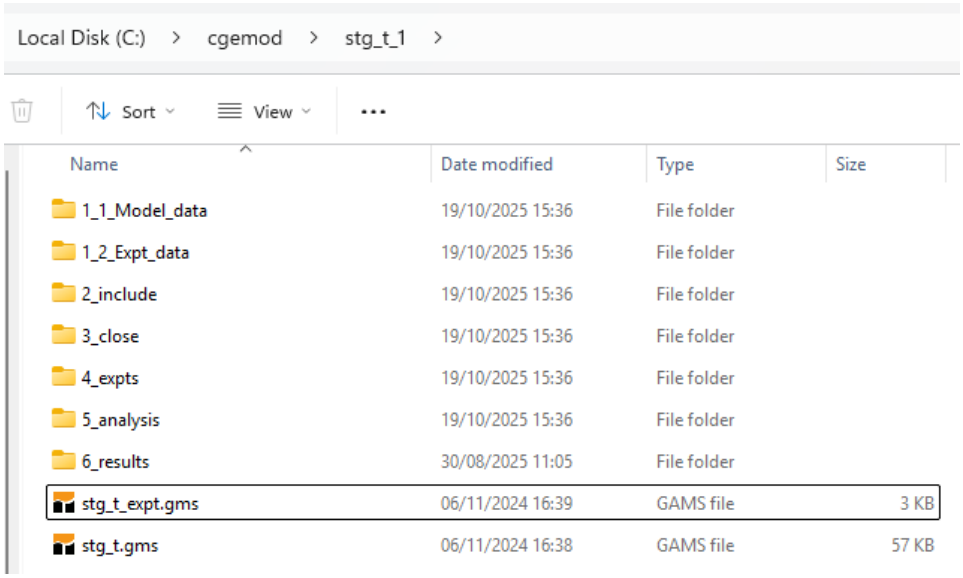
default is 6). This will ensure the GSP file is generated even when working with simple models and makes the project GSP file explicit; when you are more experienced with Studio this will become redundant.

If you choose to save an existing GMS file (`File>Save as` or `Ctrl+Alt+S`) with a different name, the default path will be identified by the project file. But note it will not be the `Main File`, so if you want to run the new GMS file you need to make it the `Main File`.

Two, or more, GMS Files in One Directory

It can be convenient to have more than one GMS file in a directory, see Figure 5.5; this is the case in cgemod's single and global courses and in other cases.

Figure 5.5 Multiple GMS Files in one Directory



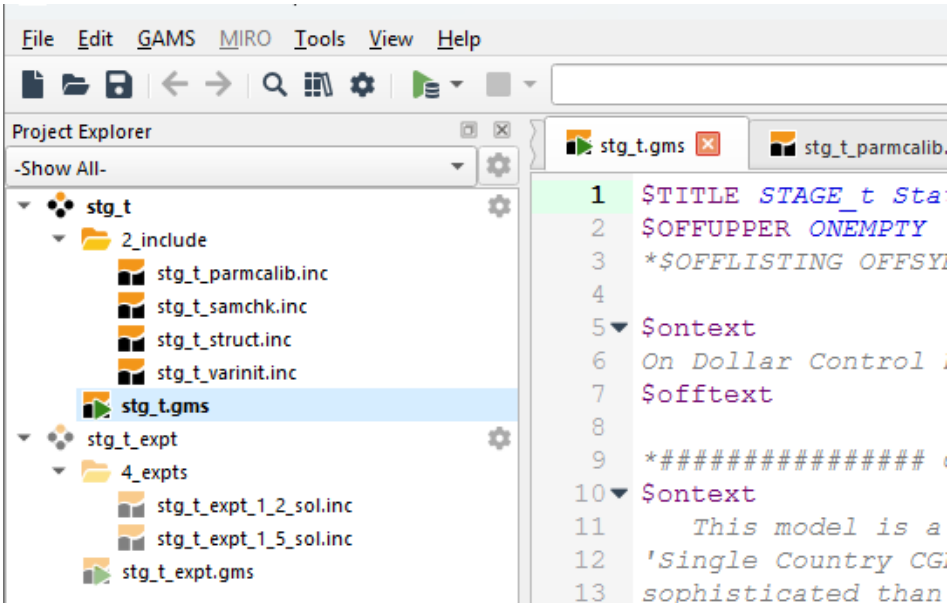
If `File>Open` is used to separately open the two GMS files the default settings in the `General Tab` of the settings dialogue box will open each GMS file in a separate project, see Figure 5.6. Then if `File>Open in Current Project` is used, selected files that relate to each project can be opened; the illustration in Figure 5.6 illustrates cases where various files are stored in sub-directories.

Dynamic Main File

GAMS Studio has an option to have a 'dynamic main file'. This option means that the main file in a project is determined by the GMS file that is open in the editor window, i.e., the selected tab. The dynamic main file option can be switched on for a specific project in the project options, see Figure 5.2, or generically in the Behavior section of the General tab in Studio Settings (`F7`).

In basic/standard operations the dynamic main file option avoids having to change the main file when working with a project with more that one GMS file in the working directory of a projects

Figure 5.6 Two Projects and One Directory in Studio



Accessing Files in a Project

There are various ways files that relate to a specific project can be accessed. Until now the presumption has been that users would use `File>Open ….` , but this is not the only way and may not be efficient.

Accessing Embedded Files from GMS

Assume you are working with a GMS file that has various embedded INCLUDE files, e.g., Figure 5.7, where `stg_t_parmcalib.inc` is an embedded include file in the sub-directory `2_include`, that you want to edit. If you `Ctrl+click` on

`2_include/stg_t_parmcalib.inc` the file will open in the editor window. See Figure 5.7b.

Figure 5.7a Opening Files Embedded in a GMS File 1

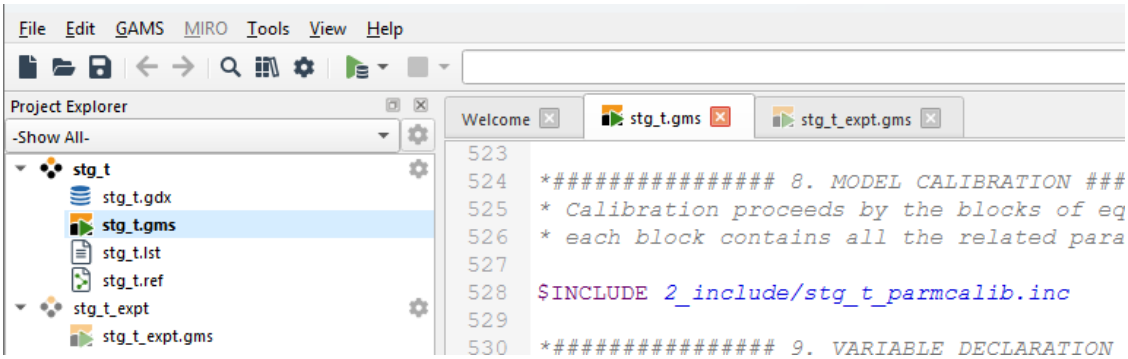


Figure 5.7b File Opened from a GMS File

Note how the file is added to the Project Explorer and opened in the Editor window and the sub directory in which `stg_t_parmcalib.inc` is stored is identified in the Project Explorer.

Accessing Files from the Process Log Window

The various (INCLUDE) files used to compile a programme are identified in **GREEN** in the process log. Clicking on any file of interest opens the file in the editor window and adds the file to the Project Explorer. See Figure 5.8.

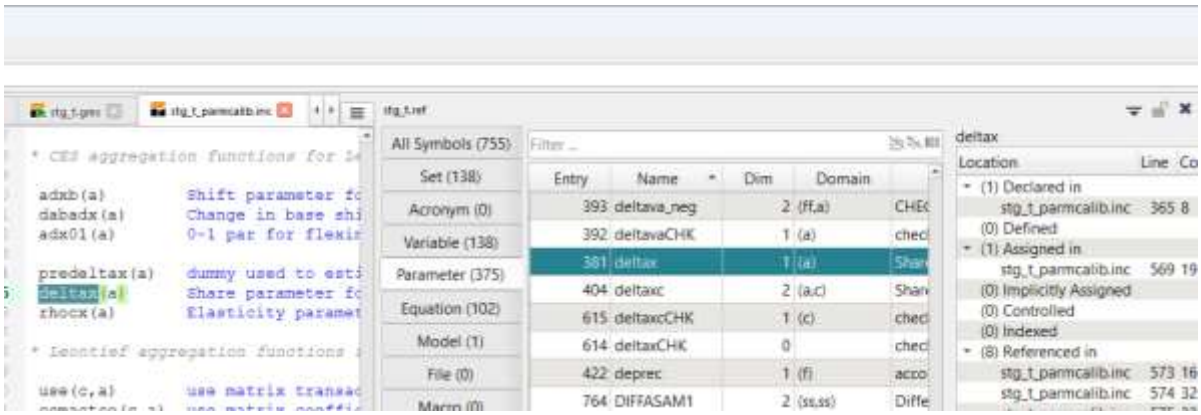
This can be especially useful when trying to modify and/or debug model since the model can be configured to stop in ways that mean the log stops close to where a model failed. See the section 15 on debugging.

Figure 5.8 Opening Files from the Log

Accessing Files from a Reference File

Reference files are discussed in Section 9, but since they provide a method for opening (and navigating) files from the editor window and moving to specific points in the files it is useful to mention them here.

Figure 5.9 Finding where deltax is Declared



Assume you want to track where the parameter `deltax` is used in the model `stg_t.gms` from the reference file. Open the reference file in the editor, select the Parameter filter and then find the entry `deltax`. This opens an index that identifies where `deltax` is referenced in the programme. Figure 5.9 illustrates where `deltax` is declared, i.e., first item in the index.

If you click on indented entries, Studio opens the referenced file, if not already open, and takes you to the line and column identified.

New Project

Sometimes a user wants to start a new project from scratch. The New Project feature in GAMS Studio is not efficient but can be used to create an empty project that can be populated with a new and blank programme file (**.gms) or from a Model Library.[10] All courses offered by cgemod rely on User Model Libraries to provide access to structured course materials, so understanding how to create a New Project is important for the courses. The steps to follow are

1. in Windows Explorer create a directory for the new project, e.g., `C:/cgemod/test`,
2. open the Settings (`F7`)
3. on the General tab make sure that the option '`Open file in current project by default`' is selected,
4. choose `File>New Project` and this opens the project options in the edit viewer,
5. the default name for the project is `newProject`,
6. rename the project name as, for example, `test` in the editor window (note the name in the Project Explorer will not change until the project is saved – see step 7 below),
7. set the Working directory to the new project directory, e.g., `C:/cgemod/test`, using the BROWSE button, (see Figure 5.10a)
8. note the Base directory will default to the same name as the Working directory,
9. choose `File>Save` – **DO NOT USE SAVE AS**[11]
10. now choose `File>New` (see below) that will open a Windows Explorer window with defaults File name, `new***.gms`, (see Figure 10b) – this step together with step 3 above is necessary to ensure that Studio is directed to use the new project's directory,
11. click `SAVE` and `new**.gms` will appear in the test project in the Project Explorer.

This is illustrated in Figures 5.10a and b. Note how the `Main File` identified and that the directory `test` has a single entry in Windows Explorer.

The user can now populate the directory `test` by

1. using a new GMS file (`File>New`),

[10] GAMS are reportedly exploring options to improve the efficiency of the New Project facility.
[11] As of Studio version 1.22.2 using Save As produces an error.

2. copying files from a previous project or an external source,
3. loading files from the `GAMS Model Library (F6)`,
4. loading files from a `USER Model Library (F6)`.

In the latter two ways of populating the project's directory there will be a minimum of **TWO** GMS files; the one downloaded from the library and the one created by `File>New`. The latter will be redundant when working with files from a library.

Figure 5.10a New Project with Project Options in the editor

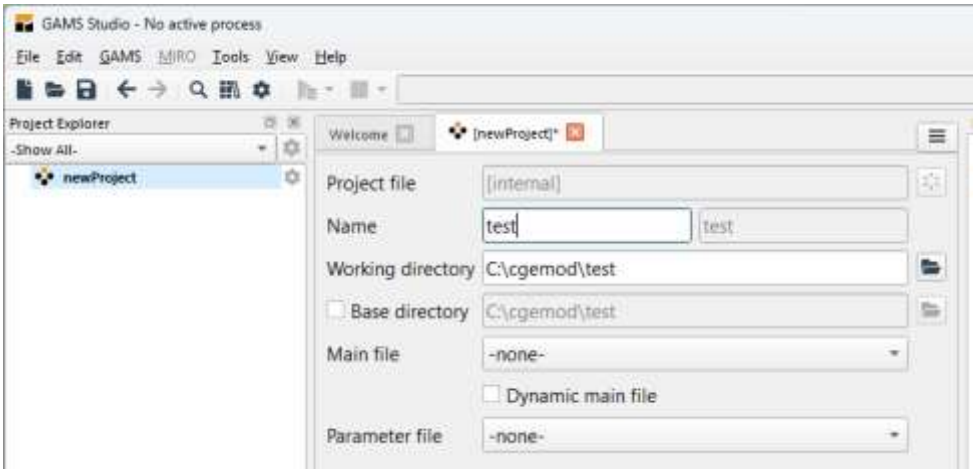
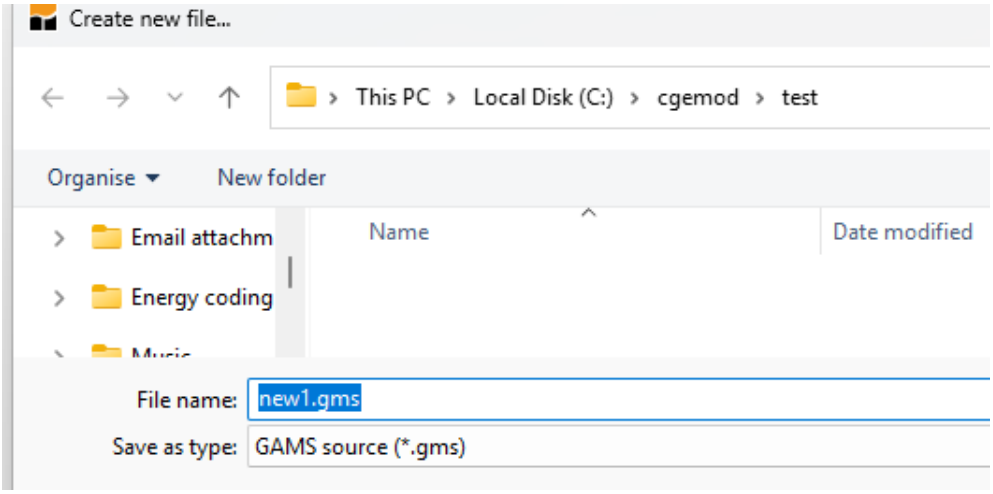


Figure 5.10b New GMS file in New Project



The Model Libraries provided with GAMS use individual GMS files for each library entry. The chosen settings determine how files from model libraries are assigned to projects.

1. Choose F6 and then select the TRANSPORT problem from the Model Library; this will open the file trnsport.gms in the editor view and assign it to a new project - trnsport. The GSP file will become visible when the model is run using F10.
2. Now open settings (F7) and choose the General tab for Settings and choose 'Open file finds or creates a new project'. Choose F6 and then select the ABEL problem from the Model Library; this will open the file abel.gms in the editor view and assign it to the project abel. The GSP file will become visible when the model is run using F10.

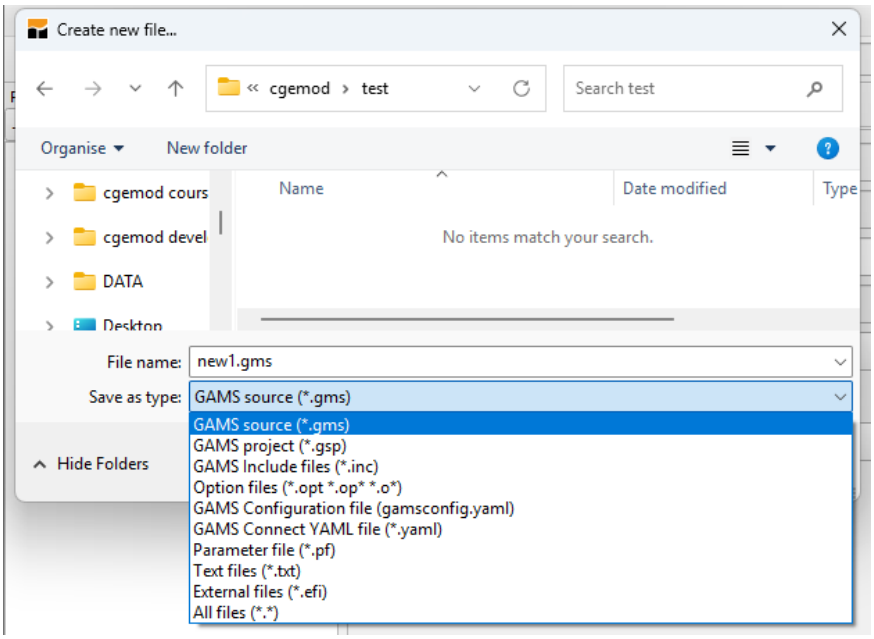
The courses from cgemod use a series of User Model Libraries. The cgemod (User) Model Libraries provide multiple files and/or directories for each component of different modules to ensure that the files for each module are correctly organised in specific directories. The instructions for cgemod courses assume the option Open File in the Current Project by Default is selected.

New Files

The command File>New allows users to create new files and assign them to specific directories. This can be used to add a new file to an existing project, say as part of model development, and to a new project (see Figure 5.11).

Note that the new file can be any of the file types supported by Studio although the default name is new1.gms (see Figure 5.11).

Figure 5.11 New File

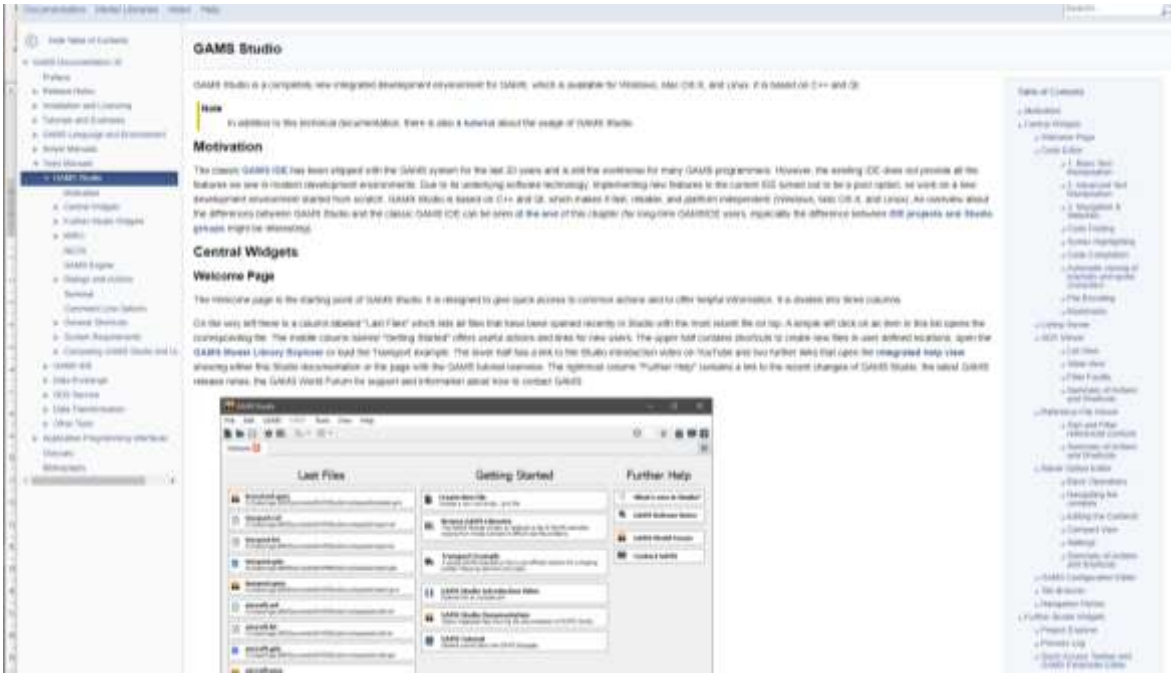


6. Help with Studio

The Help offered by GAMS is comprehensive. You should note that GAMS is a dynamic environment for which updates (releases) are 'frequent' (typically 4 to 6 releases each year), so the documentation may not be perfect.

To access the help facilities for Studio use `Help > Studio Documentation`. Alternatively, you can use `Help > GAMS Documentation`, or `F1`, or `View > Help`, and then Studio documentation can be accessed as part of the full GAMS documentation. The help window will stay open until it is closed. To close the help window, use `View > Help`.

Figure 6.1 Studio Documentation/Help

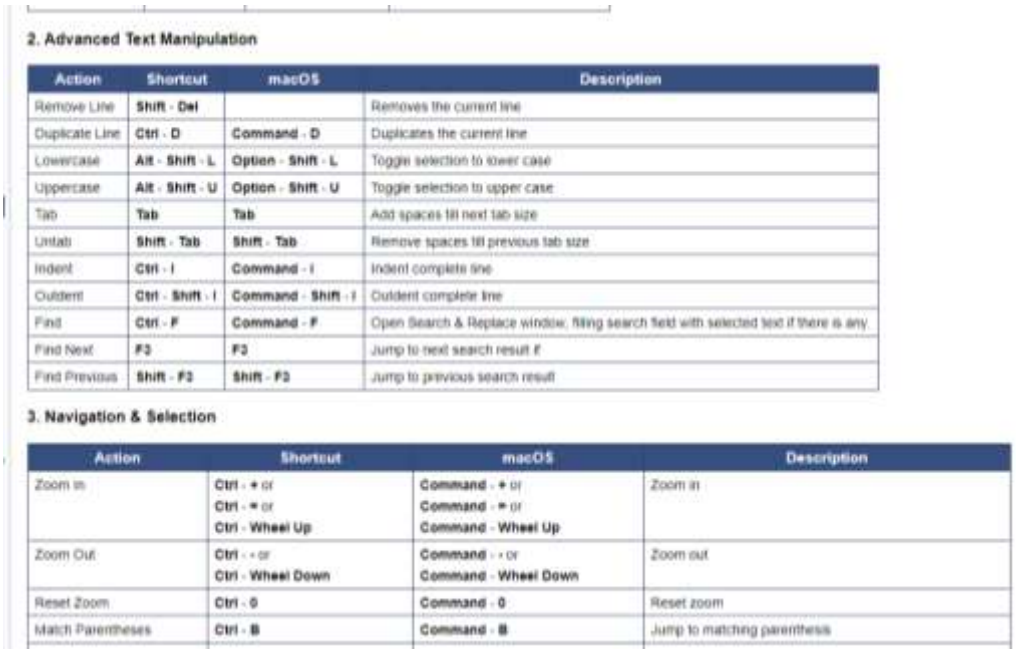


The help systems have indices on the left and right-hand sides. The left-hand side covers the full GAMS documentation, while the right-hand side relates to sub sections.

It is recommended that you familiarize yourself early with several parts of the Studio help. First, the section that details the keyboard short cuts for advanced text manipulation and navigation and selection, Figure 6.2 (basic text manipulation uses keyboard commands that are 'standard'). And second, the section dedicated to 'settings', Figure 6.3; you will in due course wish to customise the settings to mirror your preferences.

NOTE: Studio is a computer programme; hence it will take you time to master its features, so over time you should expect to use the help regularly to enhance your capacity and to keep up to date with changes.[12]

Figure 6.2 Keyboard Shortcuts



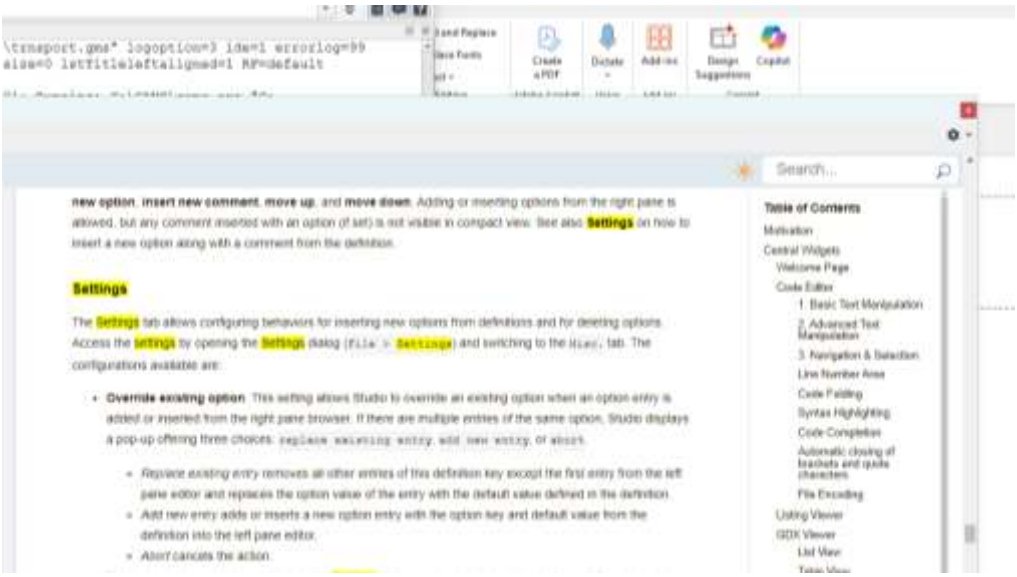
2. Advanced Text Manipulation

Action	Shortcut	macOS	Description
Remove Line	Shift - Del		Removes the current line
Duplicate Line	Ctrl - D	Command - D	Duplicates the current line
Lowercase	Alt - Shift - L	Option - Shift - L	Toggle selection to lower case
Uppercase	Alt - Shift - U	Option - Shift - U	Toggle selection to upper case
Tab	Tab	Tab	Add spaces till next tab size
Untab	Shift - Tab	Shift - Tab	Remove spaces till previous tab size
Indent	Ctrl - I	Command - I	Indent complete line
Outdent	Ctrl - Shift - I	Command - Shift - I	Outdent complete line
Find	Ctrl - F	Command - F	Open Search & Replace window, filling search field with selected text if there is any
Find Next	F3	F3	Jump to next search result if
Find Previous	Shift - F3	Shift - F3	Jump to previous search result

3. Navigation & Selection

Action	Shortcut	macOS	Description
Zoom In	Ctrl - + or Ctrl - = or Ctrl - Wheel Up	Command - + or Command - = or Command - Wheel Up	Zoom in
Zoom Out	Ctrl - - or Ctrl - Wheel Down	Command - - or Command - Wheel Down	Zoom out
Reset Zoom	Ctrl - 0	Command - 0	Reset zoom
Match Parentheses	Ctrl - B	Command - B	Jump to matching parenthesis

Figure 6.3 Settings



[12] For each new release GAMS provides 'release notes' that detail changes. It is wise to read these notes so that you are aware of changes.

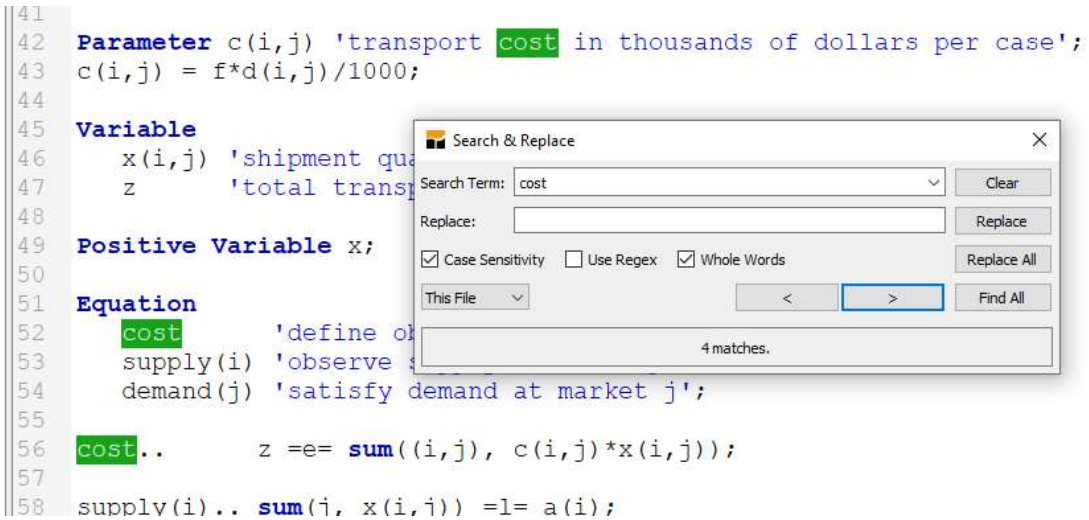
7. Search and Replace

The 'Search (find) and Replace' facility in Studio offers numerous useful features. The search and replace facility can be accessed using `Ctrl+F` or `Edit > Search and Replace`, both of which open the dialogue box illustrated in Figure 7.1. The guidance below is based on a run of the `trnsport.gms` programme with GDX and reference file creation.

Figure 7.1 shows the results of a `Find All` search of the `trnsport.gms` file for the word `cost` with the options `Case Sensitivity` and `Whole Word` selected and the search limited to `This File`. 4 matches were found, each of which is highlighted in GREEN and the references to `cost` can be stepped through using the forward (>) and back (<) arrows.

The range of the search can be controlled by selecting an option in the dropdown box below the case Sensitivity option. In Figure 7.1 it is limited to This File, i.e., the file currently viewed in the editor.

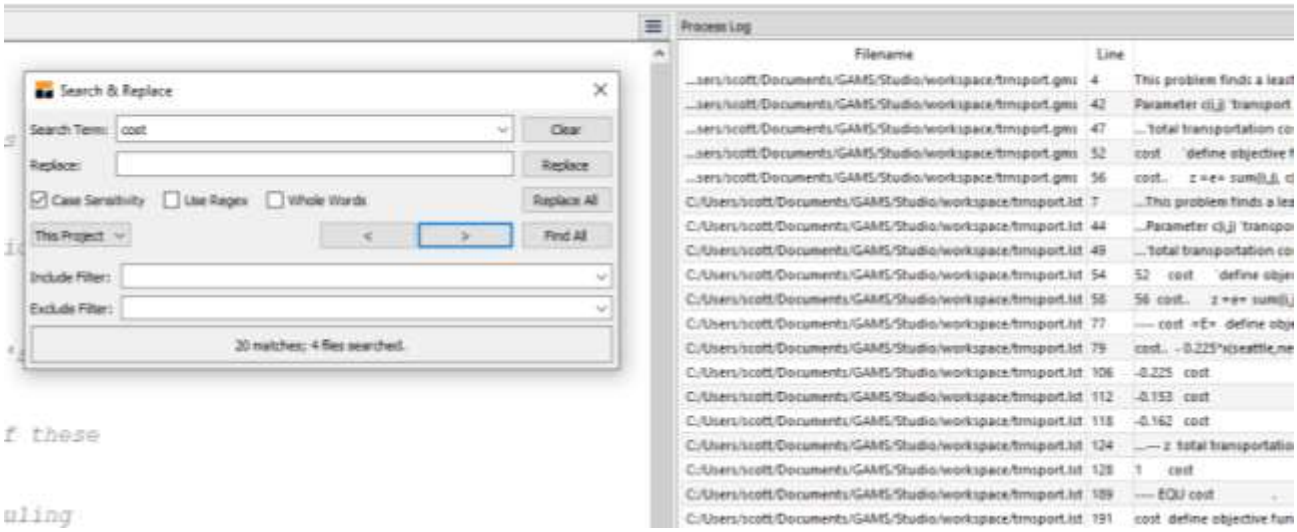
Figure 7.1 Search and Replace



The Search option allows for multiple ranges. Figure 8.2 illustrates the results of a `Find All` search of the `trnsport.gms` file for the word `cost` with the option `Case Sensitivity` selected and the search scope being `This Project.` 30 matches were found in 4 files searched, each of which is highlighted in GREEN and the references to `cost` can be stepped through using the forward (>) and back (<) arrows. It is also possible to search

using the information reported in the Process Log: clicking on any line in the process log will bring the identified file to the front in the editor panel and highlight the reference by converting the highlighting from GREEN to BLUE.

Figure 7.2 Search Results



If something is entered into the `Search Term` and the `Replace` boxes, the `Replace All` and `Replace` buttons become active. According to the scope of the search, replacements can be implemented through one or multiple files.

Selection of Files or Contents (Selection) to be Searched

The ability to select the files that will be searched is useful. The interpretation of the phrases used in the dropdown menu is:

1. "This File" limits the search to the currently active file and is the default. File patterns are ignored for this option.
2. “Selection” limits the search to a selected block of text.
3. "This Project" searches all files that belong to the same group as the currently opened file. A group of files are all children of the same node in the Project Explorer.
4. "Open Tabs" searches all files currently opened in Studio (except GDX files).
5. "All Files" searches all files that appear in the Project Explorer and are searchable.
6. “Folder” searches all files in a designated Folder.

Hints

1. Define the terms to be searched for closely.
    a. Use parentheses if appropriate, e.g., (j) will produce different results to j with whole word selected.
    b. GAMS is case insensitive, but you choose a convention that uses UPPER and lower case to aid readability.
    c. Regex – is an advanced search term interpretation mode and that stands for Regular Expressions. When activated, instead of a single search term users can specify a pattern that matches an array of different words. Further information about regular expressions is available in the Help system.
2. Note that only files that are open in a Project will be searched, i.e., files that are referenced are not searched if they are not open in a Project.
3. Be careful if using Replace All, especially if doing so over multiple files.

8. Opening and Using GDX files

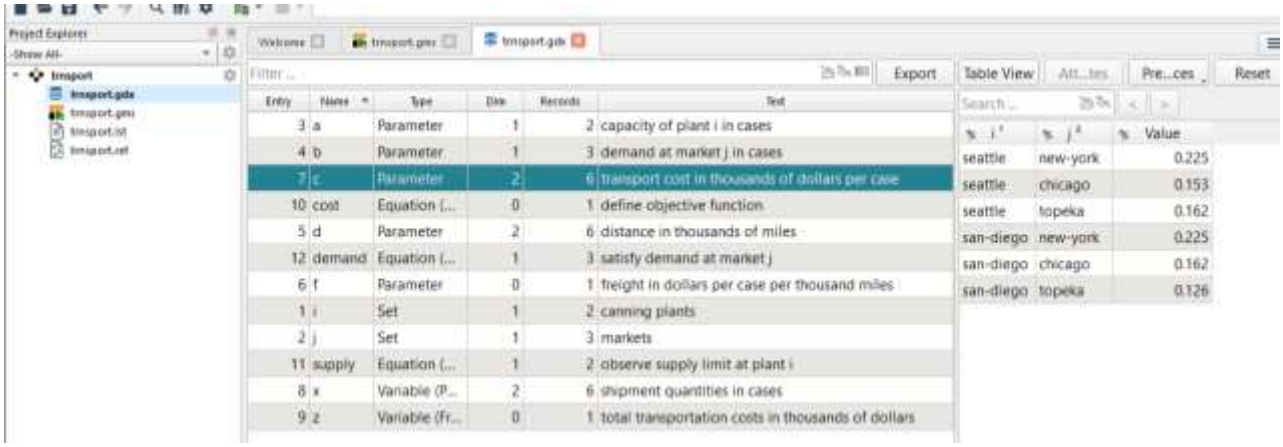
GAMS Data Exchange (GDX) is a series of utilities that have massively impacted on the productivity of the GAMS software; it is difficult to overstate the value of GDX. The GDX utilities are overwhelmingly accessed and used in GAMS programme files and are covered in, and critical for, the cgemod courses; the focus here is on the use of GDX within Studio.

It is important to understand that the standard way of importing, exporting and reporting data relating to a GAMS programme is through GDX, e.g., when data are read in from Excel the data a first transformed into GDX format and then accessed by GAMS. Reading from and writing GDX files is extremely fast. (Inputting data to a GAMS programme in a text file, e.g., as in transport.gms, is nowadays rarely used, unless the amount of data is limited.)

GDXViewer

If a programme `[filename].gms` is run with GDX creation, e.g., using `F10`, then a GDX file will be created called `[filename].gdx`, and be opened automatically in the editor window. The output for the `trnsport.gms` programme – `trnsport.gdx` – is shown in Figure 8.1.

Figure 8.1 GDX in Studio Editor

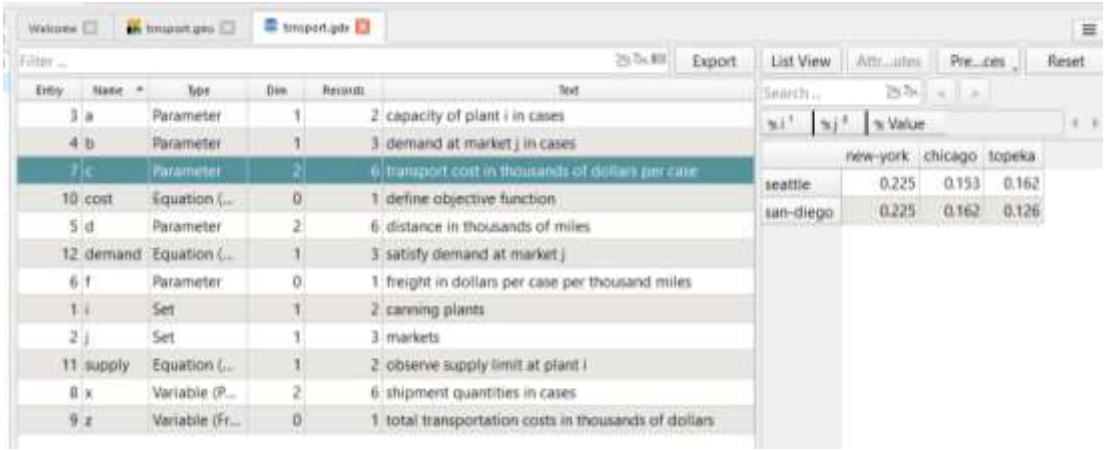


The GDX file contains all the data – sets and parameters – used by the model, the results –variables – and information about the equations. The descriptions are also reported. At this stage the information will have little meaning for you (although by the time you

complete the exercises linked to this guide you should be able to interpret the information); for now, it is enough to see how the information can be manipulated.

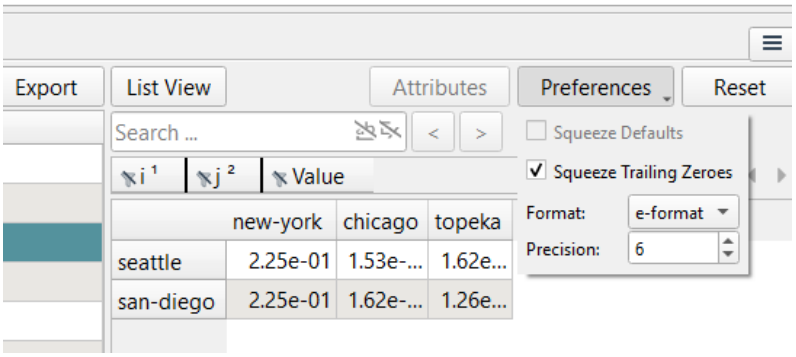
In Figure 8.1 the parameter `c`, entry 7, has been selected and the data are presented in g-format. We want to modify the view in 2 ways; first we want a `Table View`, so click on the `Table View` button, and second, we want to change the numerical format to `e-format`, so click on the `Preferences` button and select `e-format` in the format box. You should achieve the layout shown in Figure 8.3. (Note that lingering the pointer on the format box triggers an information box).

Figure 8.2 Parameter c in GDX List View



You should experiment with the manipulation of the views in the GDX window.

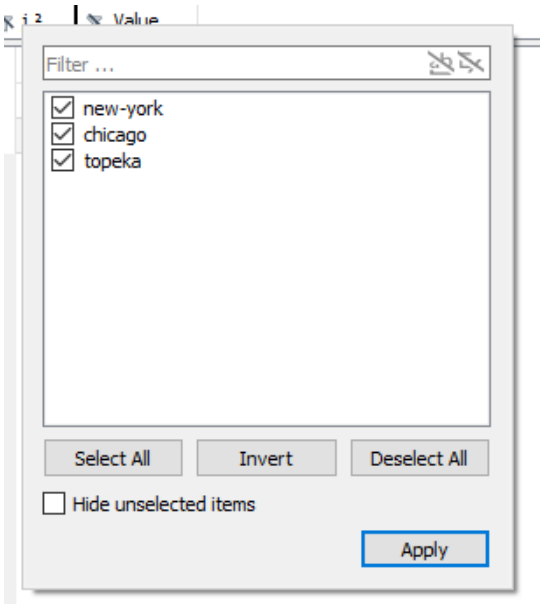
Figure 8.3 Parameter c in GDX Table View and Formatted



Filtering in GDX Viewer

It is often the case that GDX Tables or Lists need to be filtered; most commonly when there are substantial numbers of elements in each index. GDX includes a system for filtering the data that is illustrated in Figure 8.4. To filter on the index/set j right-click on the column index for j (see Figure 8.4) and this will produce the view in Figure 8.4. The filtering can be done in multiple ways: the check boxes for each element can be changed individually (they toggle), all elements can be selected (`Select All`) or deselected (`Deselect All`) or the selection can be inverted (`Invert`), and elements can be found by entering the name in the `Filter ….` Box.

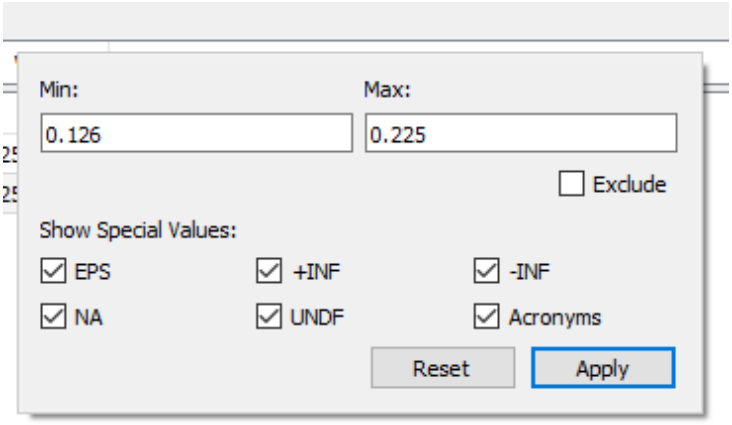
Figure 8.4 Filtering a Parameter the GDX Table or List View



The Values column in Table View or List View also allows for some filtering. If you right-click on the Values column the dialogue box in Figure 8.5 appears. This reports the min(imum) and max(imum) values and allows the user to control the treatment of special values.

The transport model is small so the various components of the GDX file have limited dimensions. This means that the full extent of the advantages of the filtering system are unlikely to be appreciated with this GDX file. But sometime spent experimenting with the filtering options using this small GDX file will be a good investment.

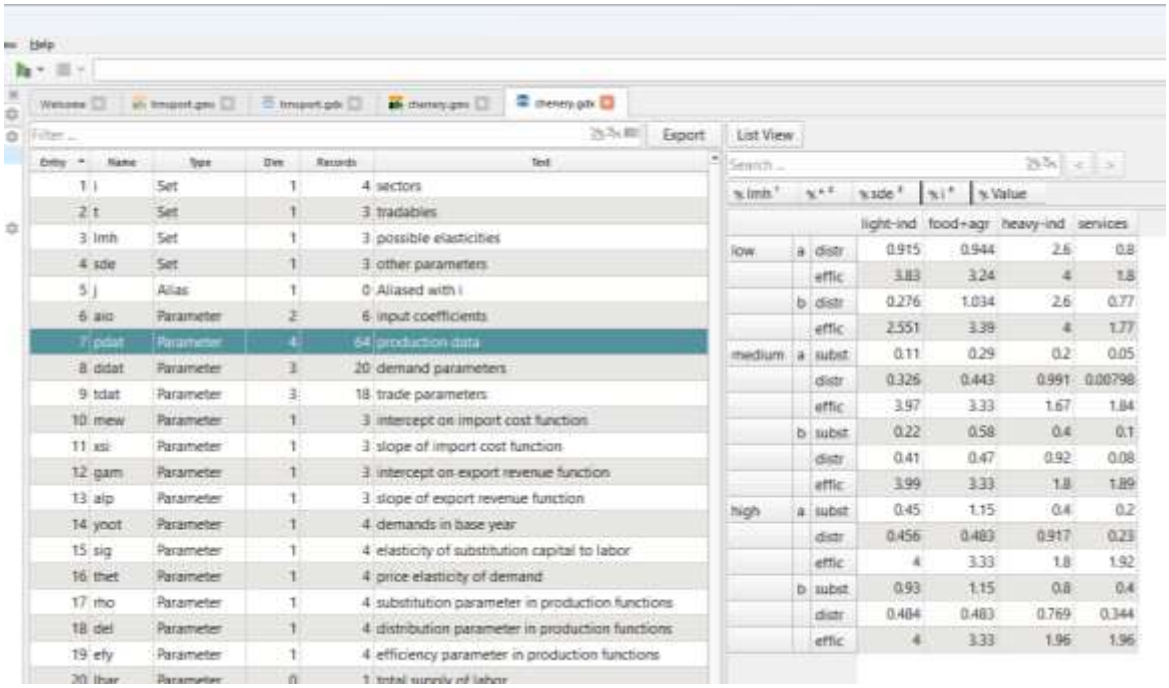
Figure 8.5 Values Filtering



Filtering is especially important when examining data for larger models with components that have more than 2 dimensions. The Chenery model has parameters and variables with 4 dimensions and up to 64 records; Figure 8.6 illustrates a filtering of the `pdat` parameter from the file `Chenery.gdx` in the default Table View. Note how by default there are 3 dimensions in each row and 1 dimension in each column. The dimensions in the rows and columns can be changed by holding the left mouse button on an element and then dragging the elements to the desired row or column. This takes a bit of practice, but the indication that you are succeeding is that a line appears between the indices.

You should experiment with various filtering using the file `chenery.gdx` (you generated this when testing the installation).

Figure 8.6 Filtering a GDX Table or List with More than 2 Dimensions



All variables in GAMS have 5 attributes – Level (*.L), Marginal (*.M), Lower (*.LO), Upper (*.UP), Scale – and so when variables are recorded in GDX the values for all the attributes are recorded. Figure 8.7 illustrates the Table View for the variable `x` in the `chenery.gdx` file, for which only the Level (*.L) values are reported.

Figure 8.7 Variable in GDX Viewer

The Defaults Attributes can be changed in Settings>GDX Viewer tab menu, Figure 8.9; this allows the user to filter by attribute, e.g., the Levels values are often of interest so the other attributes can be filtered out. The applied attributes can be filtered by using the ATTRIBUTES menu that is activated when a variable is being viewed (see Figure 8.8).

Figure 8.8 Filtering a Variable Attributes in the GDX Table or List View

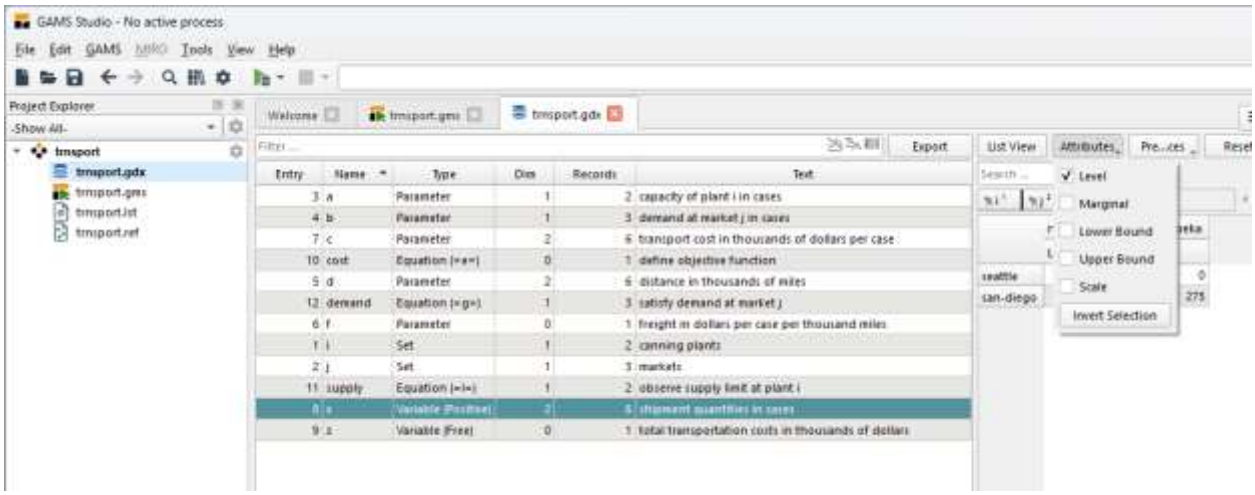


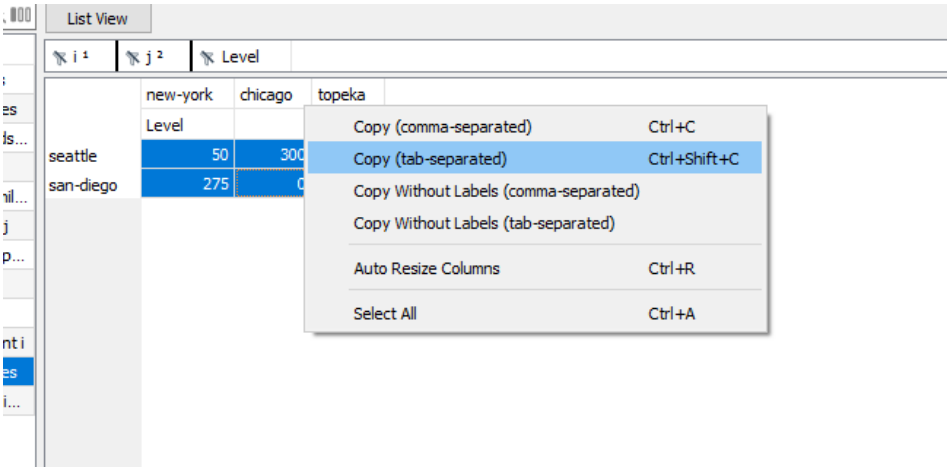
Figure 8.8 Setting Default Variable Attributes in Settings / GDX Viewer Tab

Output Data from GDX Viewer

You can also export information from the GDX viewer. In the `trnsport.gdx` file choose the parameter `c`, select `Table View` and `g-format`, with `Squeeze Trailing Zeros`. You can now select all using `Ctrl+A`, or clicking in the top left cell, or selecting the cells (note the labels are not highlighted but when exporting the labels are carried over).

Right click on the cells, see Figure 8.10, gives you options for exporting, which allow you to copy the filtered and formatted table to other programmes, e.g., Excel, Word, etc. As is common, the options box provides details for the keyboard shortcuts. NB: if exporting to Excel the better option is `Copy (tab-separated)` (`Ctrl+Shift+C`).

Figure 8.10 Copy Options in GDX View



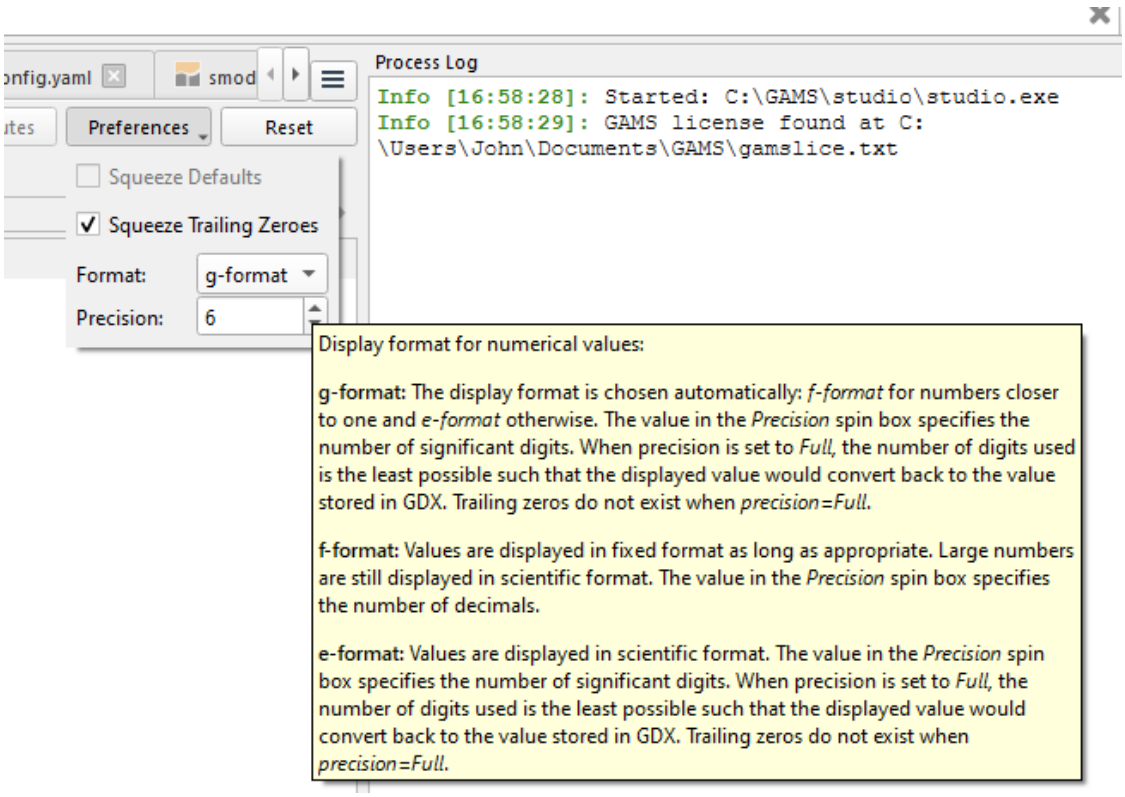
There are other ways in GAMS to output GDX data to other programmes; typically, these involve using the GDX utilities in more sophisticated ways; some of those ways are explored in various courses offered by cgemod.

Format Preferences for Data in GDX

When viewing data in GDX the data are presented in a default setting that can be varied according to need/want. Figure 8.11 illustrates a case where the default format is '`g-format`' with a precision of 6 decimals. Conveniently, details about the different formats can be accessed from within the GDXViewer: select 'Preferences' and then hover the cursor over the down arrow to the right of the format definition and a dialogue box will appear with the descriptions.

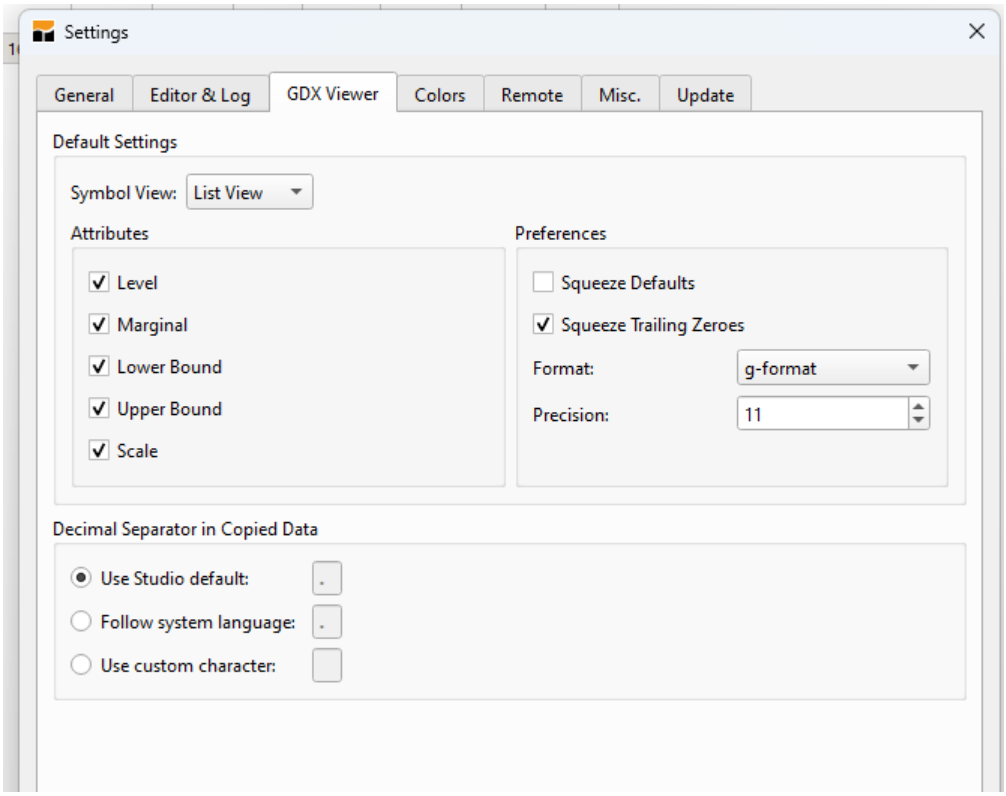
Users can select the format and precision for each symbol presented. In part the format is automatically set as is the precision. For general applications our experience is that the 'g-format' with precision 6 meets most requirements when reporting model results, but 'ideal' settings inevitably vary.

Figure 8.11 Format Preferences



When debugging a model, it can be useful to increase the precision when checking that parameter etc., values are correct. In such cases it can be time efficient to adjust the settings. Choose `File>Settings (or F7 or the settings wheel)>GDX Viewer tab` (see Figure 8.12). It is then possible to assign default settings that accord to user preferences; noting that user preferences may change according to applications.

Figure 8.12 Format Preference Settings



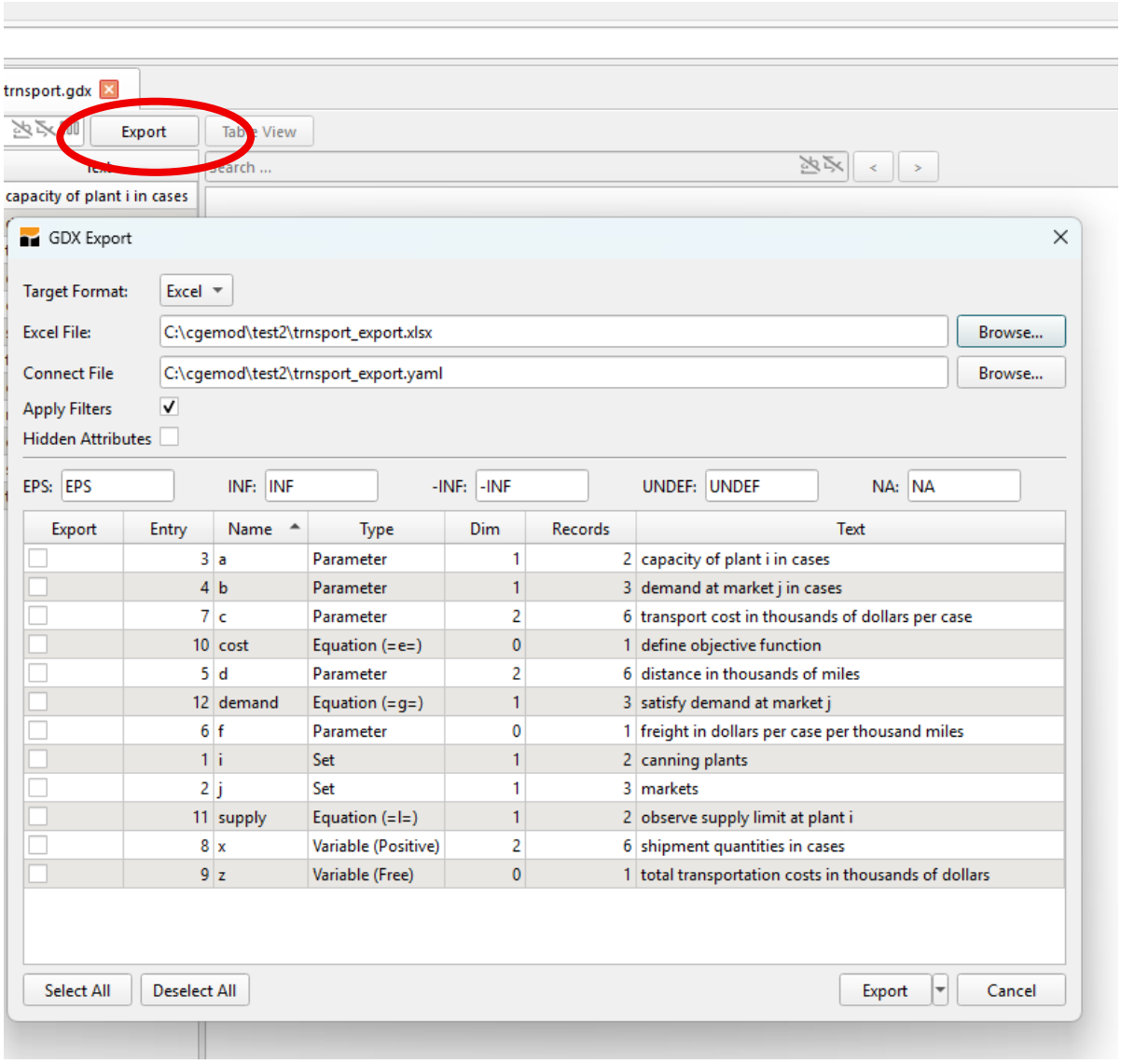
Exporting From GDX Viewer

Studio provides a simple method for selectively exporting data. In the GDX viewer mode select the export button (ringed in red in Figure 8.12) and a dialogue box appears. This utility only exports to Excel using CONNECT by generating a `yaml` file. Assign a name to the Excel and `yaml` files – Note the default path – and then select the items you want to export – in the Export column. The Export radio button implements the process.

The format of the exported data is consistent with using Pivot Tables and Charts in Excel.

NB: the `***.yaml` file can subsequently be added to programme code to automate the export of model results etc., to Excel.

Figure 8.13 Export from GDX Viewer

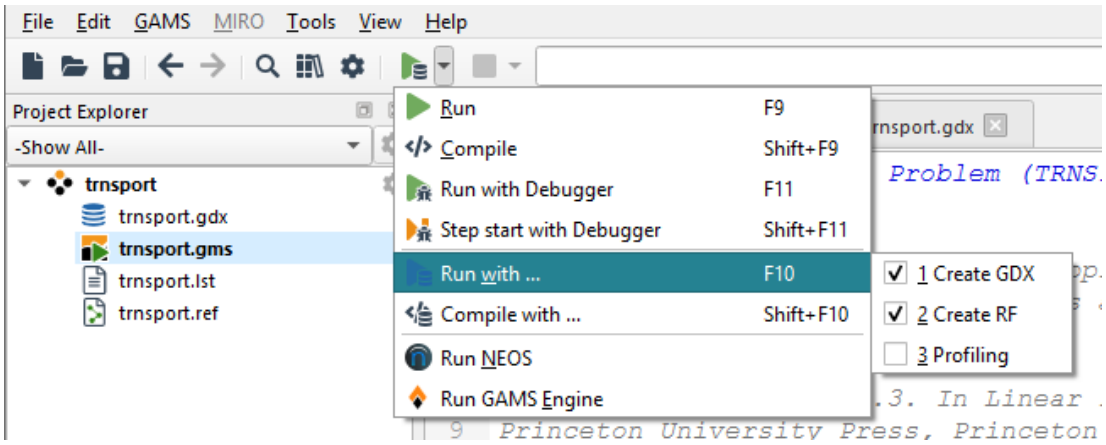


9. Reference files

Reference files are, effectively, indices to GAMS programmes. In a simple programme, such as the `trnsport.gms` programme used in cgemod's open access training materials, an index may not be considered particularly useful, but in larger and more complex programmes that run over multiple files and thousands of lines of code, indices are invaluable.

A reference file can be generated automatically when using the `Run with … (F10)` or `Compile with … (Shift+F10)` by selecting option 2 `Create RF`, see Figure 9.1 `Create GDX` and `Create RF` are the default settings for `Run/Compile with …`.

Figure 9.1 Instruction for Reference File

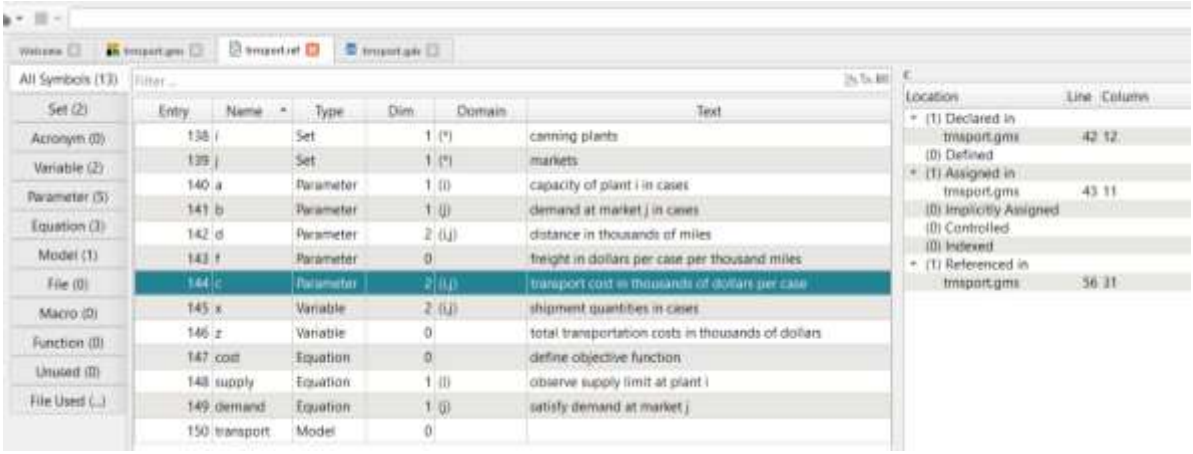


Open the reference file – `trnsport.ref` – in the Editor window. The results should look similar Figure 9.2. The information in the reference file is reported in three frames within the Editor window. The left-hand frame provides a means for filtering between different types of symbols: it opens with `All Symbols` listed. The middle frame lists the symbols with information about their names, types, dim(ensions), domains and descriptions (text). Finally, the right-hand frame contains information about the location of various attributes about a selected symbol. Thus, selections are made in the left and middle frames to acquire the information in the right-hand frame.

The `Parameter c` has been selected for Figure 9.2. Now the right-hand frame can be used to locate where attributes of the parameter `c` occur in the programme. To find where parameter `c` is declared click on the line below '`Declared in`': it will open the file

`trnsport.gms` at the point where `c` is declared. Similarly, you can identify where `c` is assigned and referenced, i.e., each of the attributes with a non-zero number in parentheses

Figure 9.2 Viewing the Reference File and Location Information



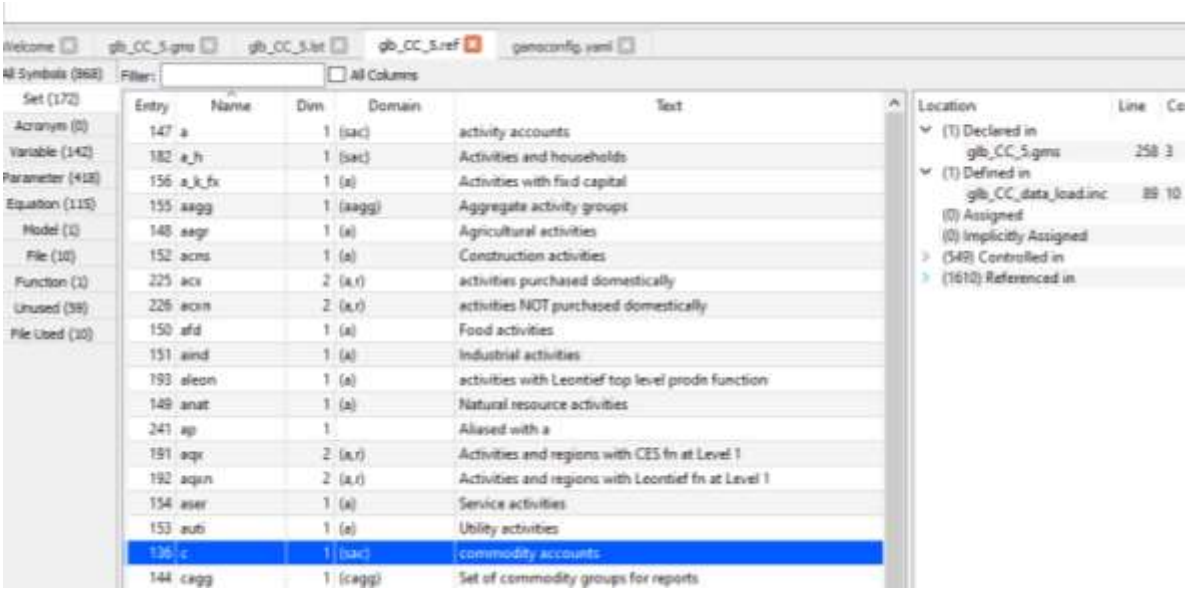
Use the reference file to explore how you can find various components of the programme `trnsport.gms`.

The benefits of the reference file are better seen for a more complex model. The illustration in Figure 9.3 is for a variant of a global model (GLB_3) run without experiments. Note, *inter alia*, how there are now 868 symbols, and 10 Files were used in a programme that ran to over 8,000 lines of code. If the set `c` is selected, it is declared in one file but defined another. Furthermore, it is Controlled in and Reference in several other files. If, for instance, the file in which `c` is Defined is clicked on in the location window, Studio will both open that file and move to the line (89) and column (10) where `c` is defined.

This illustrates how valuable a reference file can be; in fact, we generate a reference file even when we are working with models that we know well. When working with models that are not known, or have not been used for a few weeks, the reference file is a critical tool.

HINT: Using a reference file with the Search and Replace tool is also useful.

Figure 9.3 Reference File for Large Model



10. Configuring Studio

While Studio looks and behaves like a standard Windows programme, and therefore it is not hard to make adjustments, there are some (option) settings that it is helpful to know about from the beginning since they make it much easier to use GAMS.

Settings

All the setting options can be accessed using `File > Settings` or `F7` or the settings wheel (⚙) on the toolbar. The default settings are a good starting point, but some settings may need changing early on to meet your needs.

1. The default settings in the General tab are a good starting point, but we recommend selecting the option 'Open file in current project by default'.

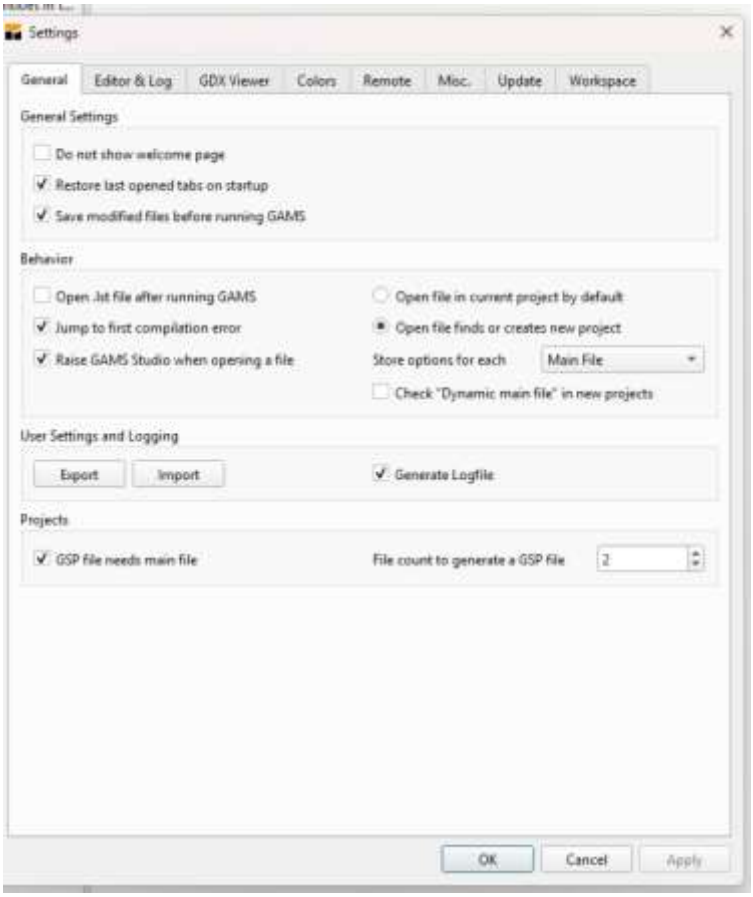
   We recommend this, especially for new users, because it is less error prone. For most users it is more common to be opening a file that is part of the current project, i.e., using `Ctrl+O`, as opposed to opening a new project, i.e., `Ctrl+Shift+O`, requires the user to make a positive decision to open a new project. (see Figure 10.1).
2. In the Editor & Log tab (see Figure 10.2)
   a. use a fixed pitch font, e.g., Courier New, (GAMS programmes can depend on the alignment of entries in columns; this can only be easily assured by using a fixed pitch font)
   b. choose a fontsize that suits you (the choice is inevitably a compromise between having enough information visible on screen and a fontsize that is easy for you to read). In Studio the font size in the editor panel can be increased (Ctrl+ +) or decreased (Ctrl+ –) at any time.
3. The GDX Viewer tab allows customisation of the GDX Viewer, the default settings are a good starting point (3ee Figure 10.3)
4. In the Colors tab the default settings are a good starting point (changes in Studio may change this).
5. The remote tab provides links to MIRO, NEOS and GAMS Engine. These are advanced features that are not used in the cgemod courses.
6. The Misc tab settings are a good starting point.

a. **One bit of information is required: for the User model library click on the 'Open Location' button and make a note of the path identified (this information will be used in the next section).**

7. The Workspace tab has some useful features as you become more skilled with GAMS. For now the important feature is the Default workspace; this is where files will be stored if the correct path has not been assigned (typically from the Project Options).

Leave the other settings for now, although you can adjust them later as you become more familiar with Studio.[13]

Figure 10.1 Studio Settings: General tab



[13] We will use these settings during the online course; a common appearance can make it easier for an instructor to provide support across multiple users.

Figure 10.2 Studio Settings: Editor & Log tab

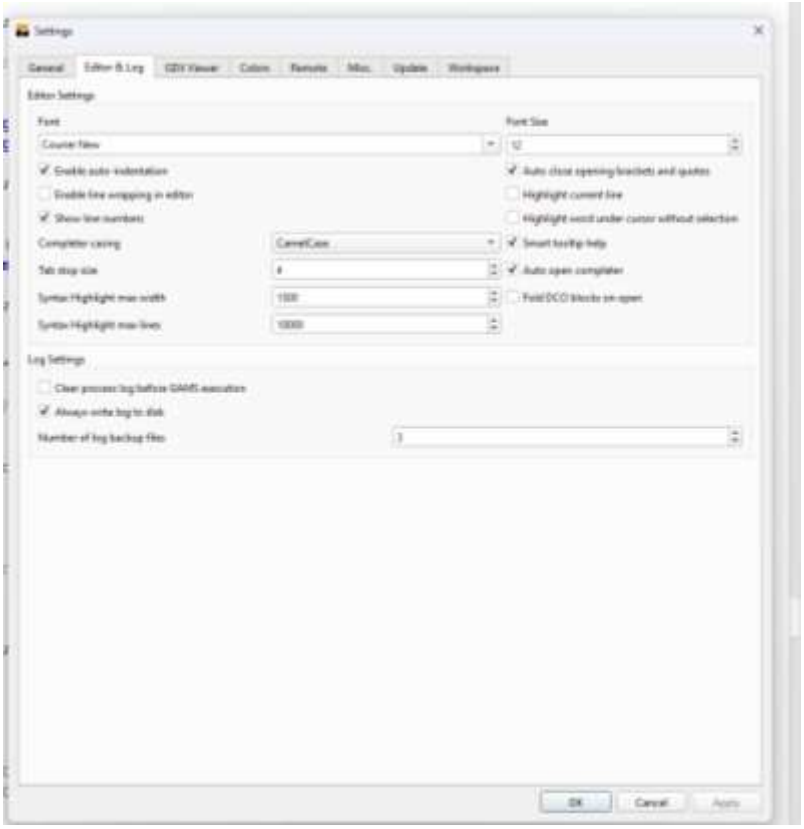


Figure 10.3 Studio Settings: GDX Viewer tab

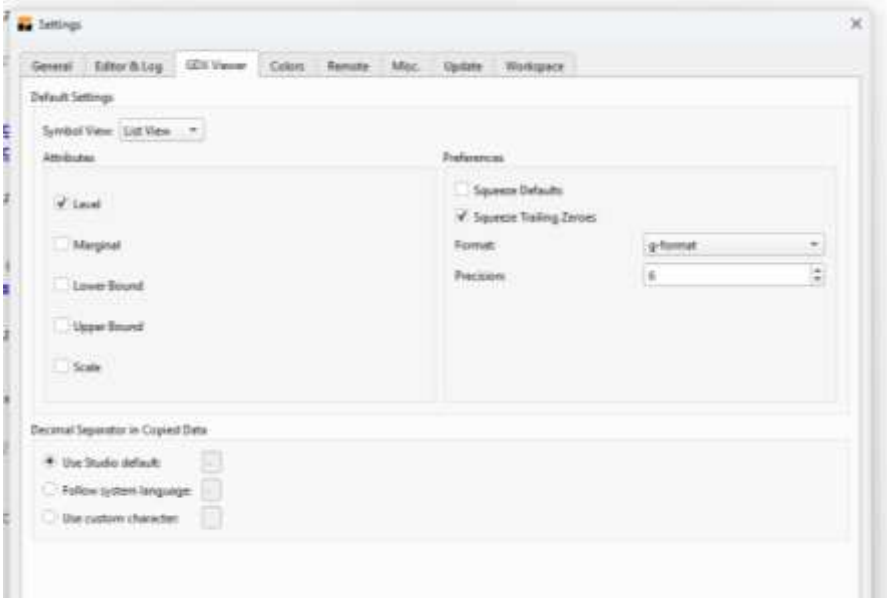
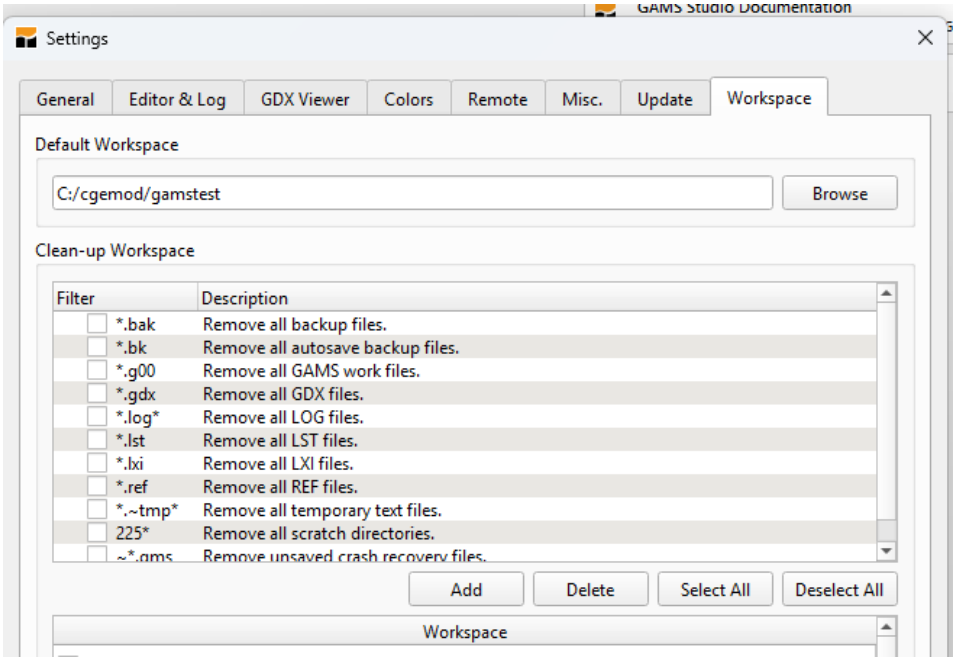


Figure 10.4 Studio Settings: Workspace tab

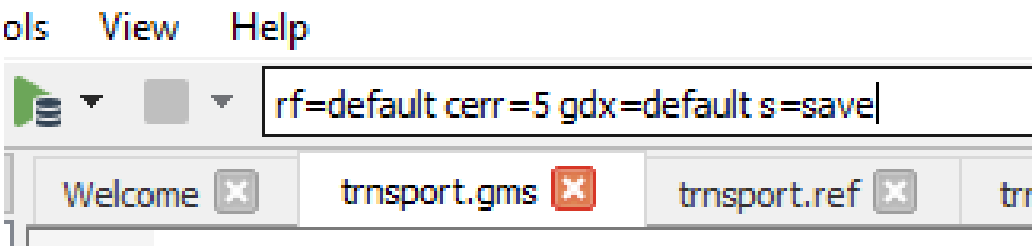


Command Line Options

The command line instructions can be very useful and GAMS Studio has enhanced the usability of the command line. Figure 10.5 illustrates a selection of command line options

1. `rf=default`: instructs GAMS to generate a reference file using the same filename as the programme, i.e., `trnsport.ref`.
2. `cerr=5`: instructs GAMS to stop after encountering 5 cumulative errors (the user can define the number they prefer).
3. `gdx=output`: instructs GAMS to generate a gdx file with the name `output.gdx` – this file would contain the same information as produced for the file `trnsport.gdx` after running with `F10`.
4. `s=save`: instructs GAMS to generate a work file or scratch file with the name save that can be used to sub divide components of a programme – these files are usually deleted at the end of a run (we use this a lot in our day-to-day work and in the cgemod courses because it makes the processes more efficient).

Figure 10.5 Command Line Instructions



Studio provides tools that make command line instructions more efficient.

First, using F10, i.e., run with gdx creation, avoids the need for the command `gdx=output`. And second, other commands can be set up as defaults every time any programme is run – below we will illustrate how to do this for `rf` and `cerr`.

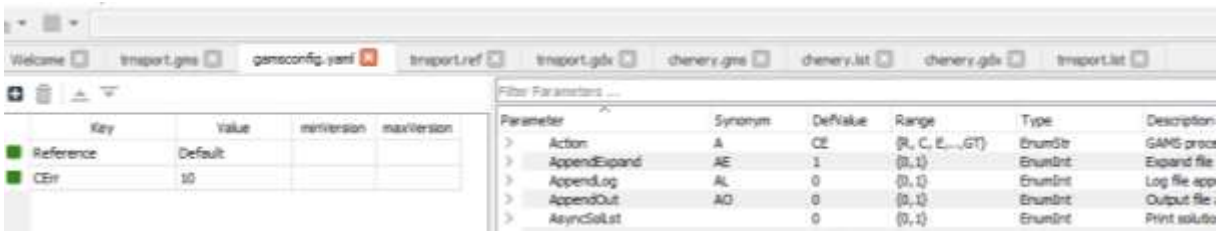
The command line instructions in Studio are project specific unless they have been set up as defaults. We find with the `SAVE` and `RESTART` option that this works best across two projects calling files from a single directory. A common way to work is to run a model that replicates the current situation and then run a separate programme, experiment, that shocks the model to evaluate how changes in model parameters impact upon the results for the model variables. The work file for the model is saved using `s=save`[14] in the command line and the programme with the experiment uses `r(estart)=save` in the command line. Because these command lines are in separate projects they are preserved when files are saved/programmes are run. The use of the save and restart facility is explained in detail in cgemod courses.

Configuring Studio to Run Commands by Default.

Studio provides a method for implementing many instructions by default. Open `GAMS > Default GAMS Configuration`; this will open the configuration editor in the Editor window (see Figure 10.6). The configuration editor has two frames: in the right-hand frame are the available commands, in the left-hand frame are the commands that are implemented by default.

[14] If a user wants to send the work file to a designated directory the instruction can be written to include a path, e.g., `s=sandr\save`, and the restart instruction would need to include the part.

Figure 10.6 Default GAMS Configuration



To instruct GAMS to generate a reference file by default find the line labelled 'reference' (HINT: use the search) and drag the reference parameter into the left-hand frame, See Figure 10.6. The selected parameter – reference – is identified by a **GREEN** box. If the Value box in the left-hand frame is left empty the reference file created every time a GAMS programme is run will have the name `1.ref`. If the value box contains the word default, then the reference file will be named after the programme name, i.e., for `[filename].gms`, as `[filename].ref`. This is a preferred option.

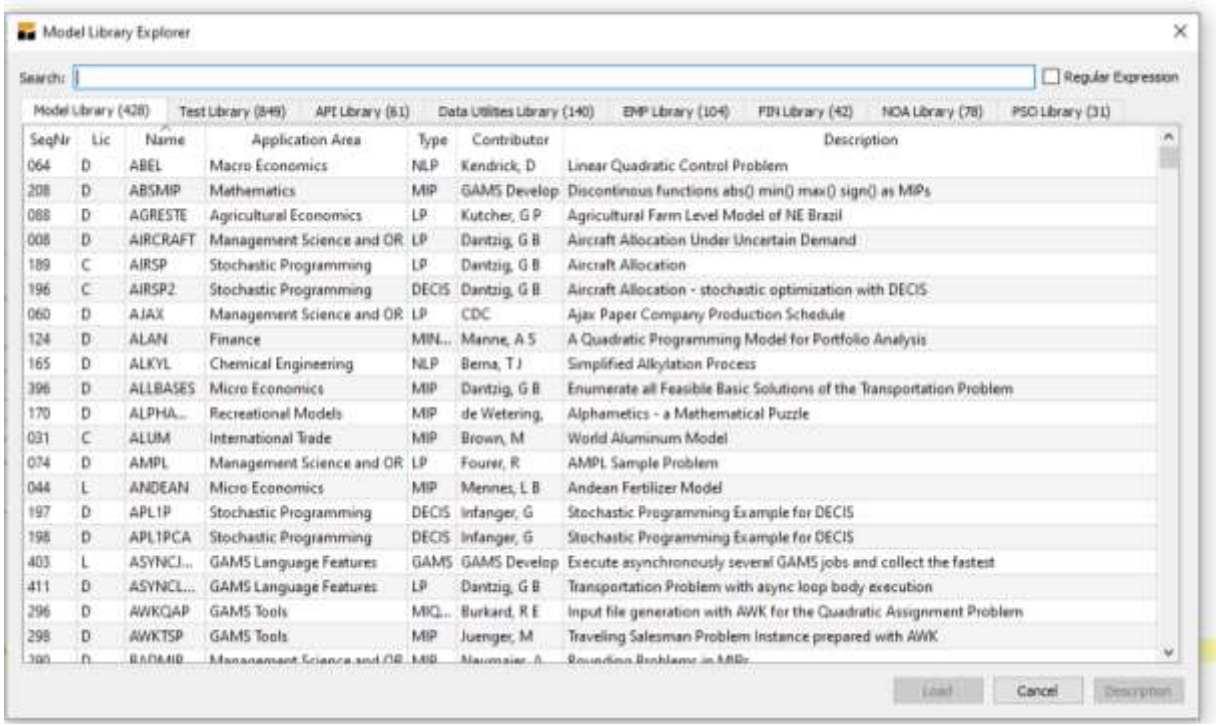
NB: a file `gamsconfig.yaml` will be created and need to be saved (note the asterisk in the tab after its name). Use the location option in the Project Explorer to find where GAMS saves this file.

To instruct GAMS to stop after a specified number of errors file by default find the line labelled 'CEer' and drag the CEer parameter into the left-hand frame, See Figure 10.3. The selected parameter – CEer – is identified by a **GREEN** box. Select the number of cumulative errors after which the programme with stop and enter this number in the Value box in the left-hand frame. Our experience suggests somewhere between 5 and 10 is a practical value.

11. Model Libraries

GAMS comes with several large model libraries. To access the libraries, select `GAMS > Model Library Explorer` or press `F6`. This opens the Model Library Explorer, see Figure 11.1. Note how there are multiple tabs each of which contains a different library. These libraries are very valuable source for code to help you develop your use of GAMS.

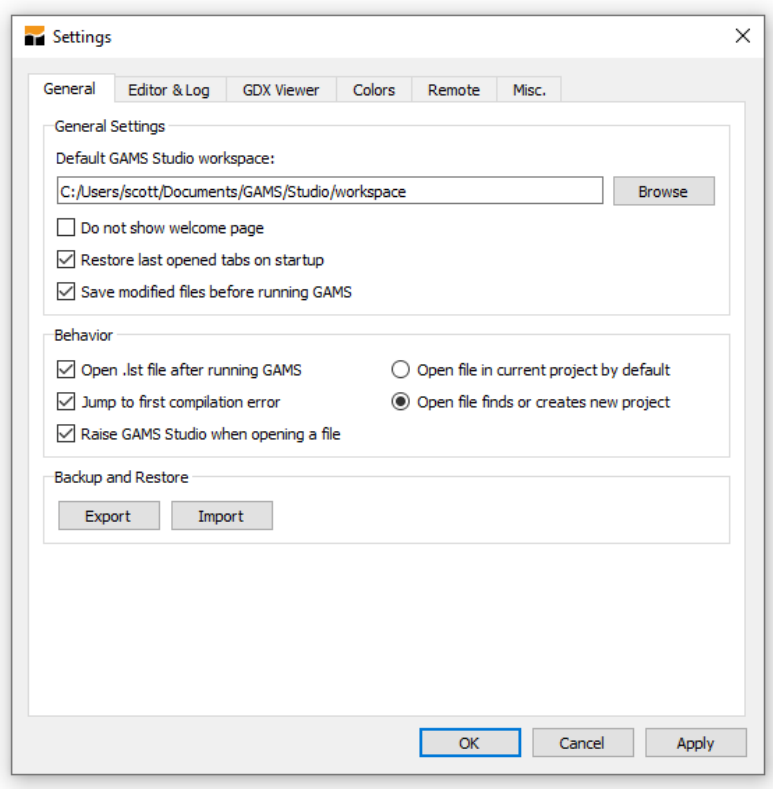
Figure 11.1 Model Library Explorer



If you click on the column title the library will be sorted in the alphanumeric order of the entries in that column. Click on the column `'SeqNr'`, the first entry in that column will now be `'001'` with the name `TRNSPORT`.

You have accessed files from the GAMS libraries, but by default the files were downloaded to the workspace directory (`C:/Users/****/Documents/GAMS/Studio/workspace`). This has been convenient for this introduction to Studio, but far less convenient for day-to-day use. The default destination directory is controlled in the Settings dialogue box (see Figure 11.2). This can be changed to any other directory using the `Browse` button on the General Settings tab. However, this method is limited.

Figure 11.2 GAMS Studio General Settings



Using a GAMS User Model Library

The 'User Model Library' facility in GAMS provides an indexing facility whereby collections of models can be archived and then easily accessed from Studio. All the files contained within a User Model Library are stored in a directory and accessed via the Model Library Explorer. An important task when a user wants to add a 'User Model Library' is to ensure it is saved in a location that the Model Library Explorer will explore. We will illustrate the process using the transport model files setup for this introduction to Studio and the directory structure we advocate for cgemod courses.

We advocate creating a master directory for all the training materials associated with cgemod's courses. So

1. In Windows Explorer create a directory, say, `C:\cgemod`. We suggest doing this on the top level of your working drive (this would be the C drive for us.)
2. Add the sub directory `cgemod_lib` to the master directory, i.e., `C:\cgemod\cgemod_lib`. This sub directory will be used to store all the User

Model Library files for cgemod courses; with each component for each course more files will be added to this directory, and the index will be updated. (It is not strictly necessary to store the files in this way, but mistakes happen, and this method means that correcting certain common mistakes is quicker.)

3. Add the sub directory downloads to the master directory, i.e., `C:\cgemod\downloads`. This sub directory stores zipped archives of model files that are downloaded for cgemod courses.
4. Download the `trans_lib.zip` file from the cgemod site (http://cgemod.org.uk/int_gams.html) and save it in the `download` sub directory.
5. Unzip the contents of the file into the sub directory called `cgemod_lib`.
6. Copy all of the files in the `cgemod_lib` sub directory –just the contents of the directory NOT the directory itself.
7. Open Studio settings - `File > Settings` or `F7` or the `Settings Wheel` on the toolbar and go to the `Misc` tab.
8. Change the path for the User Model library, see Figure 11.3, from `C:\Users\…\Documents\GAMS\modellibs` (the exact path will depend on the settings on your PC) to `C:\cgemod\cgemod_lib\trans_lib`.
9. Close down the Studio Settings window remembering to click OK at the bottom of the Misc tab.

In Studio open the `Model Library Explorer`, `F6`., and expand the window to the right and then click on the tab `Introduction to GAMS Studio;` you may need to scroll across the tabs the find the tab. The results should look like Figure 11.4. Note the 'person' figure associated with a User Model Library.

We, cgemod, use the User Model Libraries to supply collections of files to participants on our courses. The contents of cgemod's User Libraries differ from standard GAMS libraries only in that our libraries supply multiple files with each download while GAMS libraries, typically, only supply individual files. In addition, our libraries are set up with the intention of populating designated directories. Nevertheless, they operate in the same way as standard GAMS libraries.

Figure 11.3 GAMS Studio Misc Settings

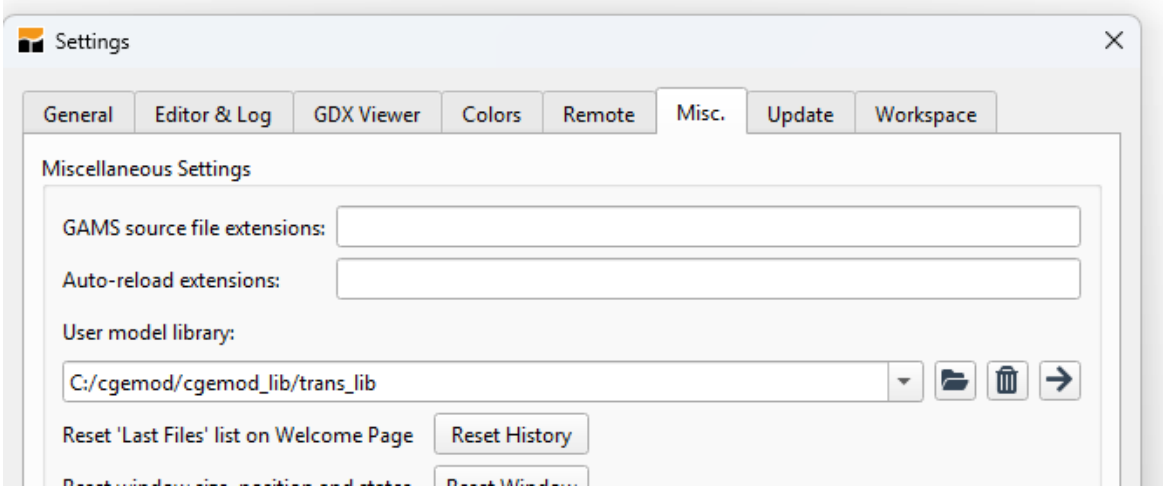
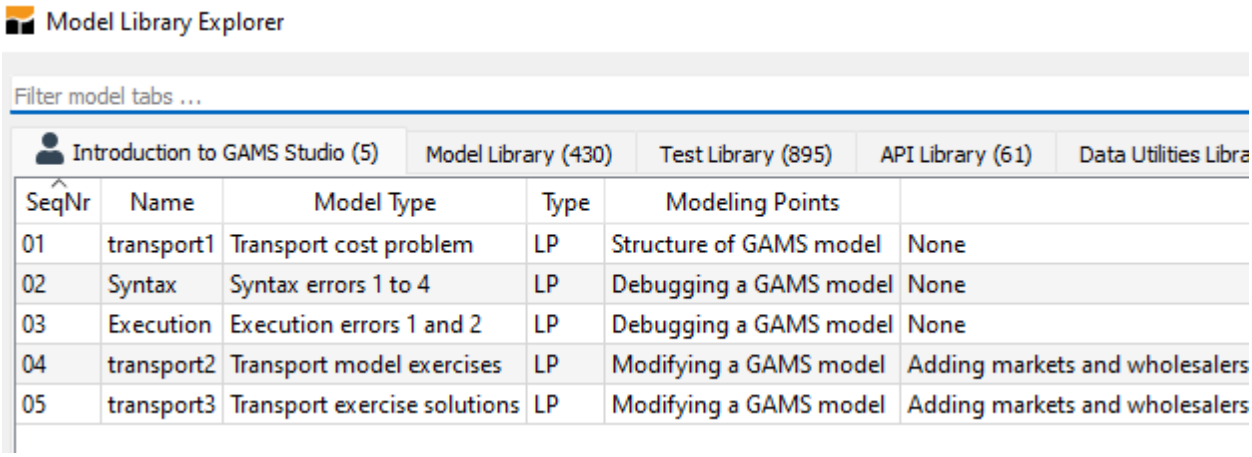


Figure 11.4 Model Library Explorer cw User Model Library



SeqNr	Name	Model Type	Type	Modeling Points	
01	transport1	Transport cost problem	LP	Structure of GAMS model	None
02	Syntax	Syntax errors 1 to 4	LP	Debugging a GAMS model	None
03	Execution	Execution errors 1 and 2	LP	Debugging a GAMS model	None
04	transport2	Transport model exercises	LP	Modifying a GAMS model	Adding markets and wholesalers
05	transport3	Transport exercise solutions	LP	Modifying a GAMS model	Adding markets and wholesalers

An alternative arrangement involves adding the User model library to the default path for model libraries. After step 5 above, i.e., after unzipping the archive from the cgemod web site (http://cgemod.org.uk/int_gams.html)

12. Comparing Files

It is sometimes helpful to compare the content of 2, or more, files. GAMSIDE included a utility, `Diff TextFiles`, that allowed for comparing 2 files, but no equivalent utility is included with Studio. (The developers of GAMS Studio have suggested that a similar utility may at some point be added for Studio.)

Access to comparison software is valuable. They allow easy comparisons between different model versions, which can be helpful when searching for 'changes', 'errors' and 'bugs'. They can also be helpful when developing new versions of existing models.

There are several options for Windows operating systems are available, including

1. MS Word: using `Review>Compare` produces a comparison of two selected documents
2. WinMerge: using `File>Open` than then selecting 2 or 3 documents to compare
3. NotePad++: open two documents and then using `Plugins>Compare`

MS Word and NotePad++ are the easiest to use, while WinMerge has the most features with MS Word the least. MS Word is free if you have MS Office/Office 365, while NotePad++ (https://notepad-plus-plus.org/) and WinMerge (https://winmerge.org/) are open source. As always with open-source programmes be careful when installing them, especially if they come from third-party sites.

The courses offered by cgemod assume the participant has access to comparison software. Our examples assume WinMerge.

13. Studio with Modular Projects

The examples used so far in this introduction are based on simple models where all the model codes are contained in a single `gms` file, and all the files are contained in a single directory. However, many modern models are modular with different components of the model in different files that are selected when the model is compiled; these components may be in multiple sub directories that may, or may not, be included in a 'master' directory. One of the major developments in Studio, compared to GAMSIDE, is the enhanced facilities for organising, searching and using models that are modular. This section provides a brief introduction to using Studio with a modular model that may, or may not, be more complex. The example used for this section is the STAGE_t model as it is used in the (single country) Recursive Dynamic CGE course.

File and Directory Structure

The directory structure and organisation of STAGE_t reflects our view about an appropriate method; this has in part been influenced by the functionality provided by Studio (see Figure 13.1). We suggest that all sub directories and files relating to a specific version of a model should be in a single master directory. Each master directory can support multiple 'projects' – model variants. Each model variant has only a single model file, e.g., `stg_t.gms`, implemented as unique project, and each model variant has one or more experiment file, e.g., `stg_t_expt.gms`, implemented as unique projects. This requires the implicit presumption that the output from the model file is saved as a work file and each experiment file restarts from the work file. The only `gms` files are saved to the top-level of the master directory. Each `gms` file generates a reference file and generic GDX file; these are saved to the top-level of the master directory.

The model and experiment `gms` files each call in multiple INCLUDE files at compile time; all these files are labelled `[filename].inc` and saved in sub directories. It is not a requirement of GAMS to label nested files as `*.inc` files, e.g., all the nested INCLUDE files could be labelled `*.gms` or even `*.txt`, but is our preference to always label INCLUDE files as `*.inc` for added clarity. Data are supplied to the models and experiments as Excel or GDX files; all Excel files are converted to GDX files by the model and experiments.

Assume that the intention is to work with the `stg_t_dyn.gms` model. Figure 13.2 illustrates how Studio might appear after the `stg_t_dyn.gms` model has been run (`F10`) and the reference file has been opened and the `gms` file scrolled to line 442. A problem at this point could be defined as a lack of information about the contents of the various, and apparently unidentified, INCLUDE files.

Figure 13.1 STAGE_t in Windows Explorer

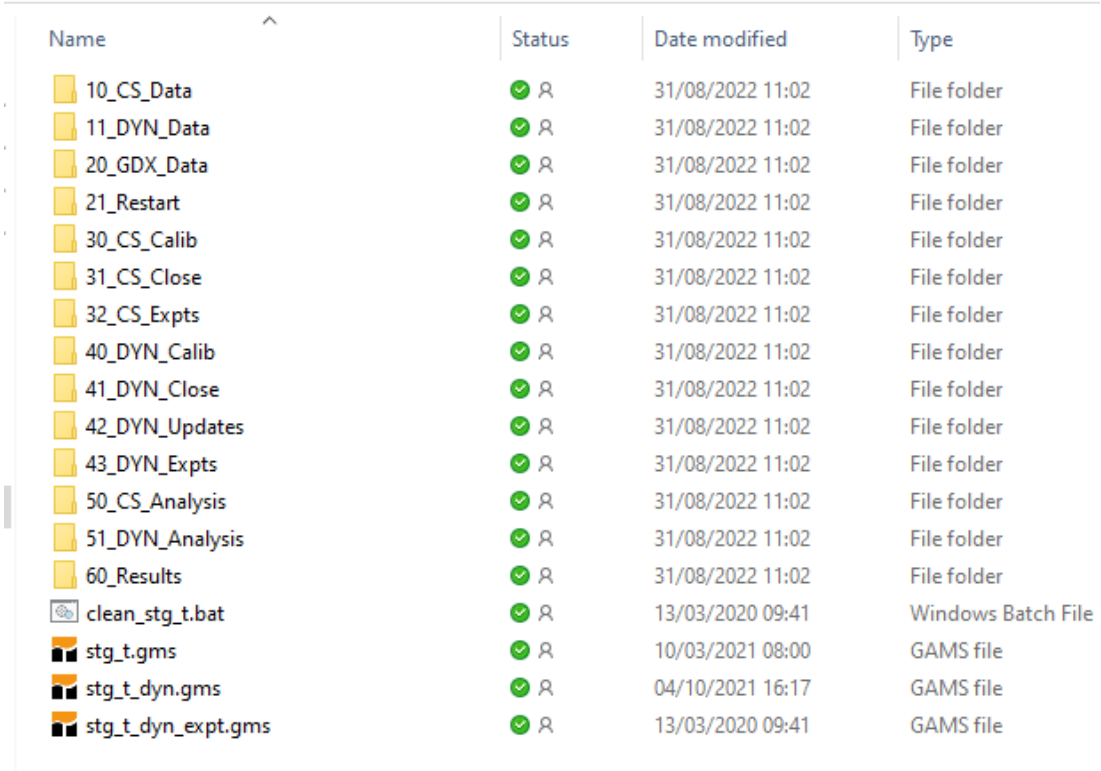
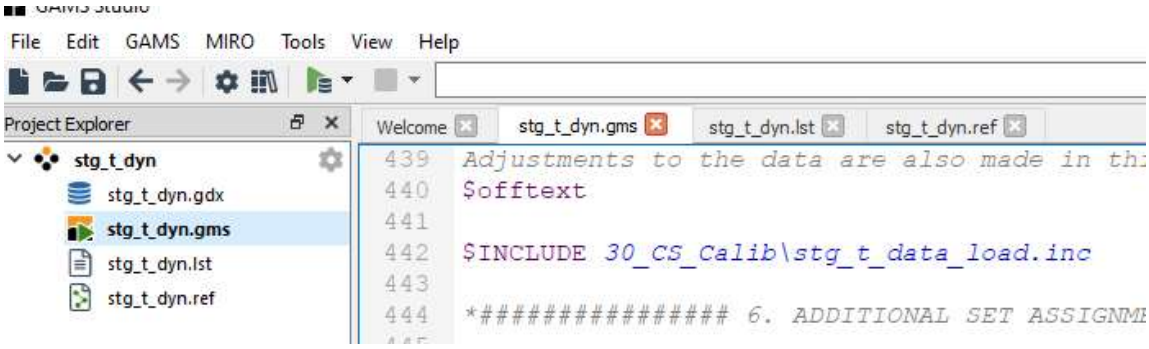
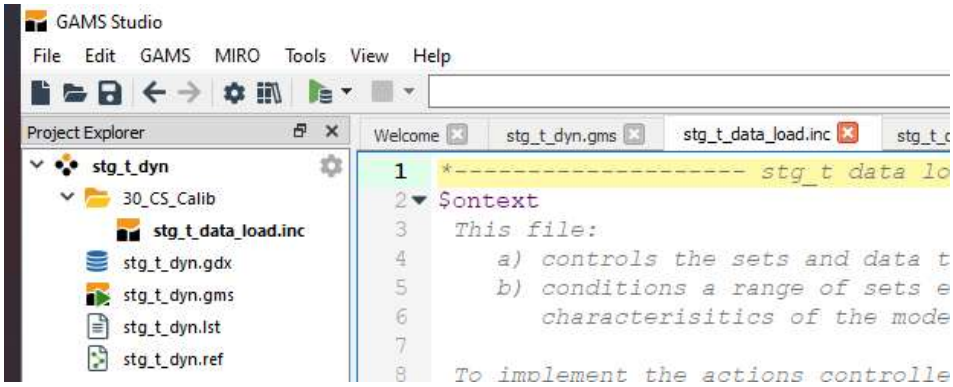


Figure 13.2 STAGE_t in Studio



To overcome this problem, it can be useful to be able to have all, or a selection, of the INCLUDE files open in Studio. Studio provides multiple ways to achieve this; each of which has advantages in different situations. The most obvious option is `Ctrl+O` (if the setting '`Open file in current project by default`' is chosen) or `Ctrl+Shift+O` (if the setting '`Open the file finds or creates a new project`' is chosen). Figure 13.3a illustrates the outcome if the file `stg_t_data_load.inc` from the sub directory `30_CS_Calib` has been opened. Note how the Project Explorer shows that file `stg_t_data_load.inc` is from the sub directory `30_CS_Calib`. The same outcome could be achieved by right-clicking on the project name and selecting `Add Existing File` from the popup menu.

Figure 13.3a STAGE_t in Studio – Opening & Finding Files



Another way to open a file it to Ctrl+click on an $INCLUDE command in any file. This is illustrated in Figure 13.2; if the user chooses to Ctrl+click on file details, i.e., the text in BLUE, in line 442, the file directory `30_CS_Calib\stg_t_data_load.inc` is opened producing the same outcome illustrated in Figure 13.3a.

When first starting to explore a modular model, it can be helpful to be able to get an overview of the model structure and the various nested files. One way to achieve that is compile the model (`Shift+F10`) [15]with the option to generate a reference file. If the reference file is opened and the symbols are filtered for `File Used`, this informs the user that 9 files, including the master `*.gms` file were used, see Figure 13.3b. The locations of the

[15] Choosing to compile rather than run the model may be necessary if there is syntax or execution problems with the model.

files used are listed in the right-hand pane in the order in which they were accessed. Moreover, if the user clicks on any of the file locations the respective file is opened in Studio, e.g., clicking on the second file in the file locations list opens the file `30_CS_Calib\stg_t_data_load.inc` as illustrated in Figure 13.3a.

One simple option would be to open all the files used in turn. This is in addition to using the reference file to select individual symbols and open the files in which they were used, see Figure 9.4 above.

Figure 13.3b STAGE_t in Studio – Opening & Finding Files

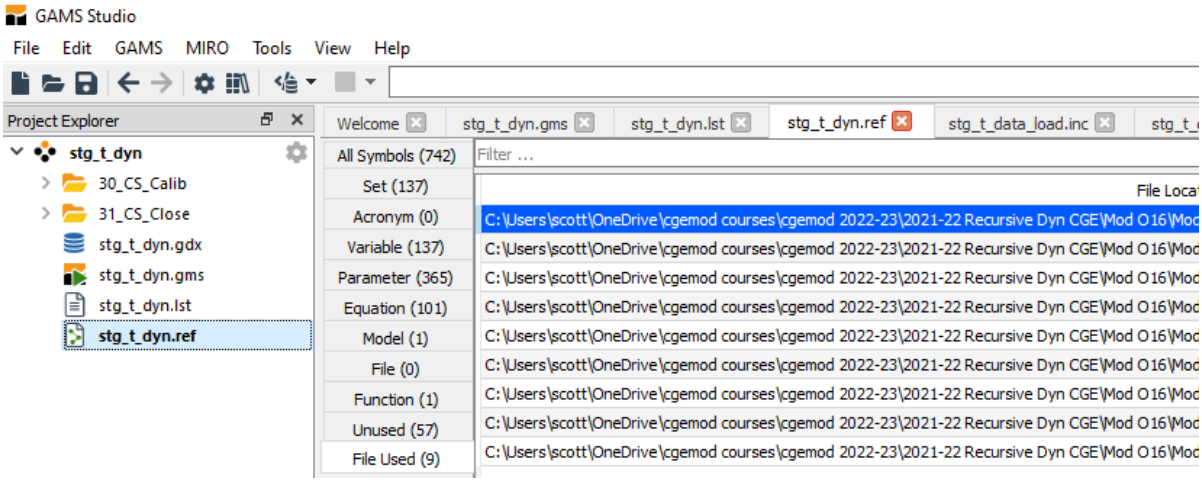


Figure 13.3c STAGE_t in Studio – Opening & Finding Files

Alternatively, when the programme is compiled the process log identifies all the included files in GREEN. If the user clicks on an INCLUDE file that file will be opened if currently closed or brought to the front in the editor pane. Note how clicking on the reference to the *.gms file, in GREEN, immediately before each INCLUDE file is referenced will open the *.gms file on the line where the INCLUDE file is cited, e.g., `stg_t_dyn.gms(521)` where 521 identifies the line number.

Figure 13.3d STAGE_t in Studio – Open Files & Modular Structure

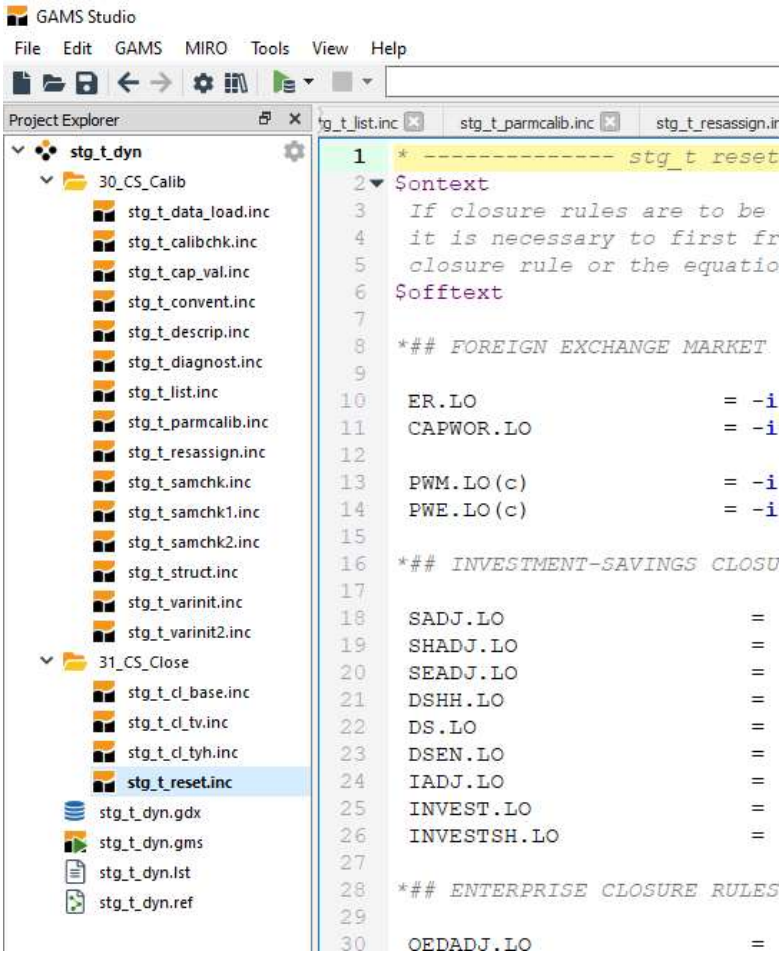


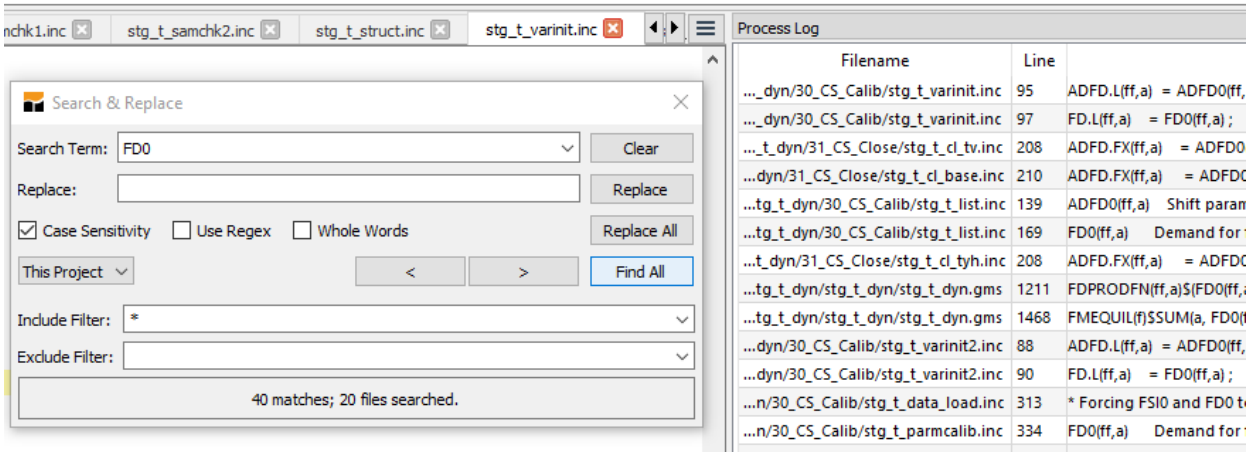
Figure 13.3d illustrates a situation where all the INCLUDE files in 2 sub directories (`30_CS_Calib` and `30_CS_Close`) have been opened in Studio. Clearly since only 9 files have been used there must be unused files that have been opened; just because a file exists does not mean it is used. Also note that if a block of files are opened at the same time that they are in alphabetic order and not the order in which they are/were accessed, but if the files are opened individually they listed in the order in which they were opened. Having all the

contents of sub directories listed may involve clutter; the sub directories can be collapsed by clicking on the down arrow by each sub directory. If all sub directories are collapsed Studio would look like the case illustrated in Figure 13.3b. Right clicking in the Project Explorer offers the options to `Collapse all projects` or `Expand All`.

Search & Replace in Modular Models

The previous discussion of the Search & Replace options, in section 8, primarily focused on searching for the contents individual files. In modular models it can be useful to search across multiple files. Studio offers 2 extremely useful Search & Replace search options that are especially useful when working with modular models.

Figure 13.4a Search and Replace in STAGE_t – 'This Project' filter

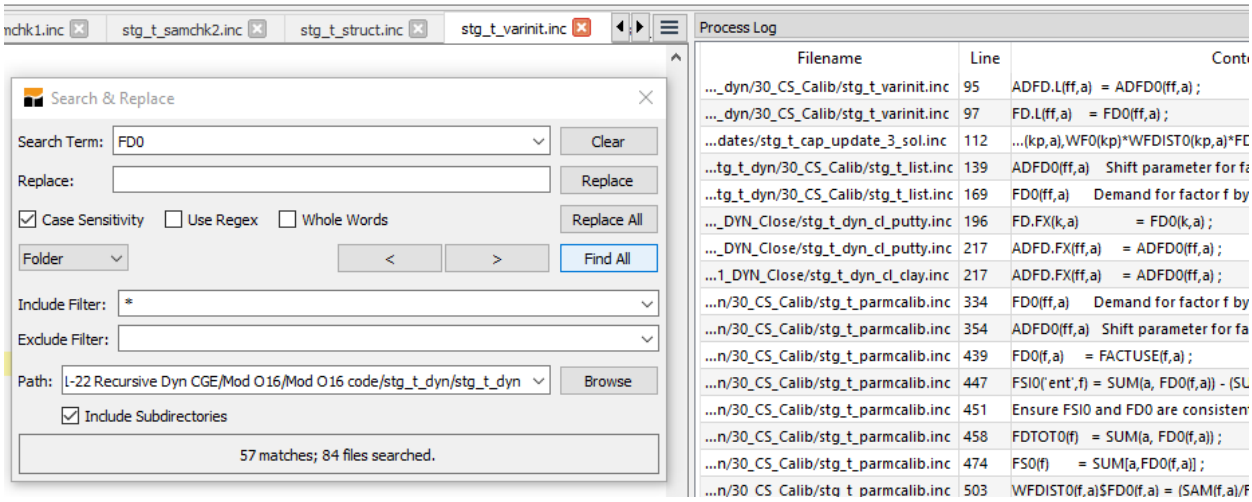


The first option is to define the Search over `This Project` (see the dropdown menu). The illustration in Figure 13.4a shows the results from a search for `FD0` across the open files illustrated in Figure 13.3d, except for `stg_t_dyn.lst` that was closed prior to the search. This reports `10 matches; 20 files searched`, and the `Process Log` provides links to each occurrence of the terms `FD0`, or the arrows < or > can be used to step through the matches. This illustrates several things to be considered: first, the definition of `This Project` is those files currently open in the `Project Explorer`, second, the search is over the files that are open NOT the files used in the model, and third references to `FD0` in files used by the model are not identified. These second and third considerations

could be partially overcome by including the `*lst` file in the search since the `*.lst` file will include contents from all the files used.

Another option is to extend the search over all files included in the folder, option `Folder`, that hosts the project. The results from this search are illustrated in Figure 13.4b, again the *.lst file was excluded. There are now `57 matches; 84 files searched`, with each reference listed in the `Process Log`. Again the arrows < or > can be used to step through the matches.

Figure 13.7b Search and Replace in STAGE_t – 'Folder' filter



The Folder search option is flexible, e.g., the specific folder to be searched can be identified by the `Path` that can be found using `Browse`.

14. Additional Features of GAMS Studio

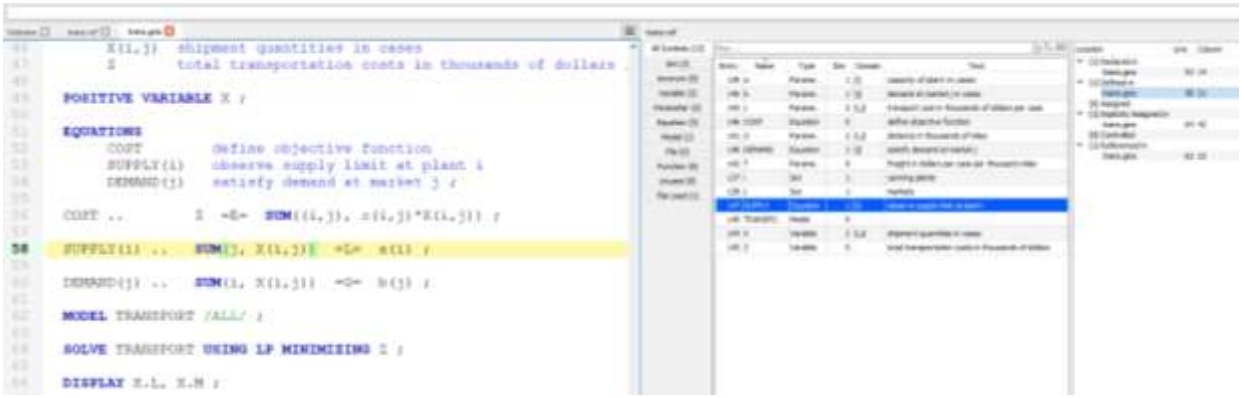
There are other editing features of GAMS Studi that are useful. This section only covers those features that are used by cgemod and that we can explain.

PIN View

"Studio provides a Pin View to carry a clone of a selected tab. This allows user to have two file contents visible at the same time" (GAMS Studio documentation, Sept 2022). The key advantage of the PIN view options is the facility that PIN offers to work interactively across (related) documents.

Suppose the user wants to use a reference file interactively with the main model file. The reference file can be pinned using – `VIEW>PIN Right` or `PIN Below`; this will pin a clone of the reference file to the `right` or `below` the editor pane and the editor pane can used as normal. For the transport model `PIN Right` is illustrated in Figure 14.1a and `PIN Below` is illustrated in Figure 14.1b.

Figure 14.1a PIN Right View: trans.gms and trans.ref



Once the reference file is pinned the user can work interactively to use the reference file to track symbols through the GAMS model.

Similarly, a GDX file can be pinned, and the user can then use the model file to identify how the various symbols (equations, variable, parameters, sets) are specified.

The PIN view is particularly useful when working with a modular model like that described above (section 13). If the reference file is pinned then when selecting the attribute

(declared, assigned, referenced) of a symbol will open the pertinent file in the editor window See Figure 14.2. In this illustration a parameter, *ac*, was selected and the attribute chosen was '`Assigned in`': this opened the file `stg_t_parmcalib.inc` at line 180.

Figure 14.1b PIN Below View: trans.gms and trans.ref

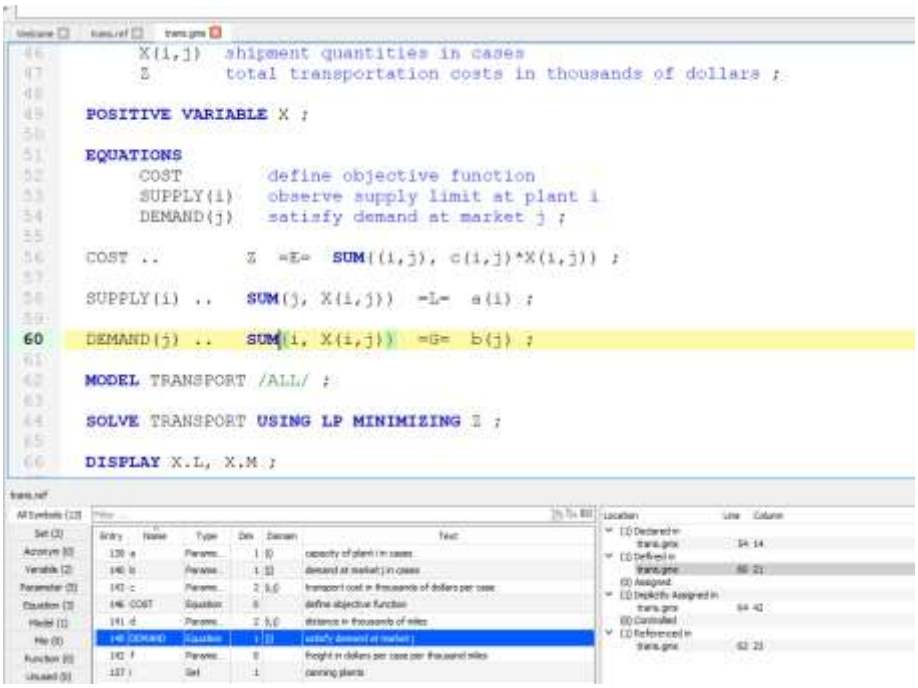
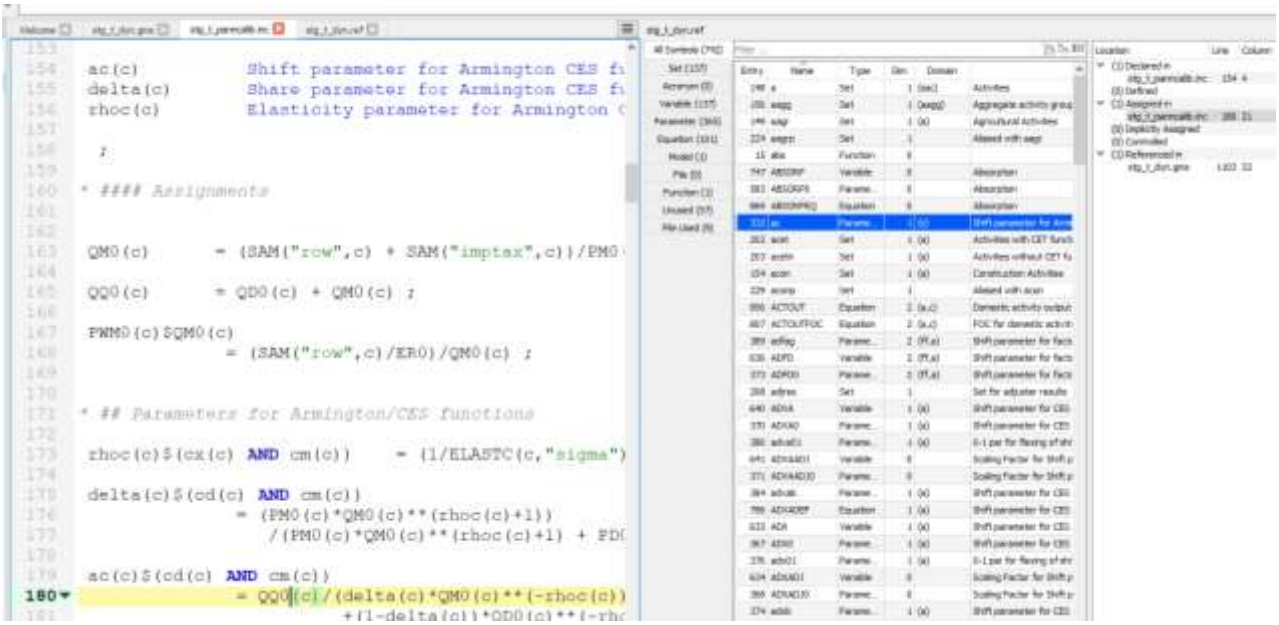
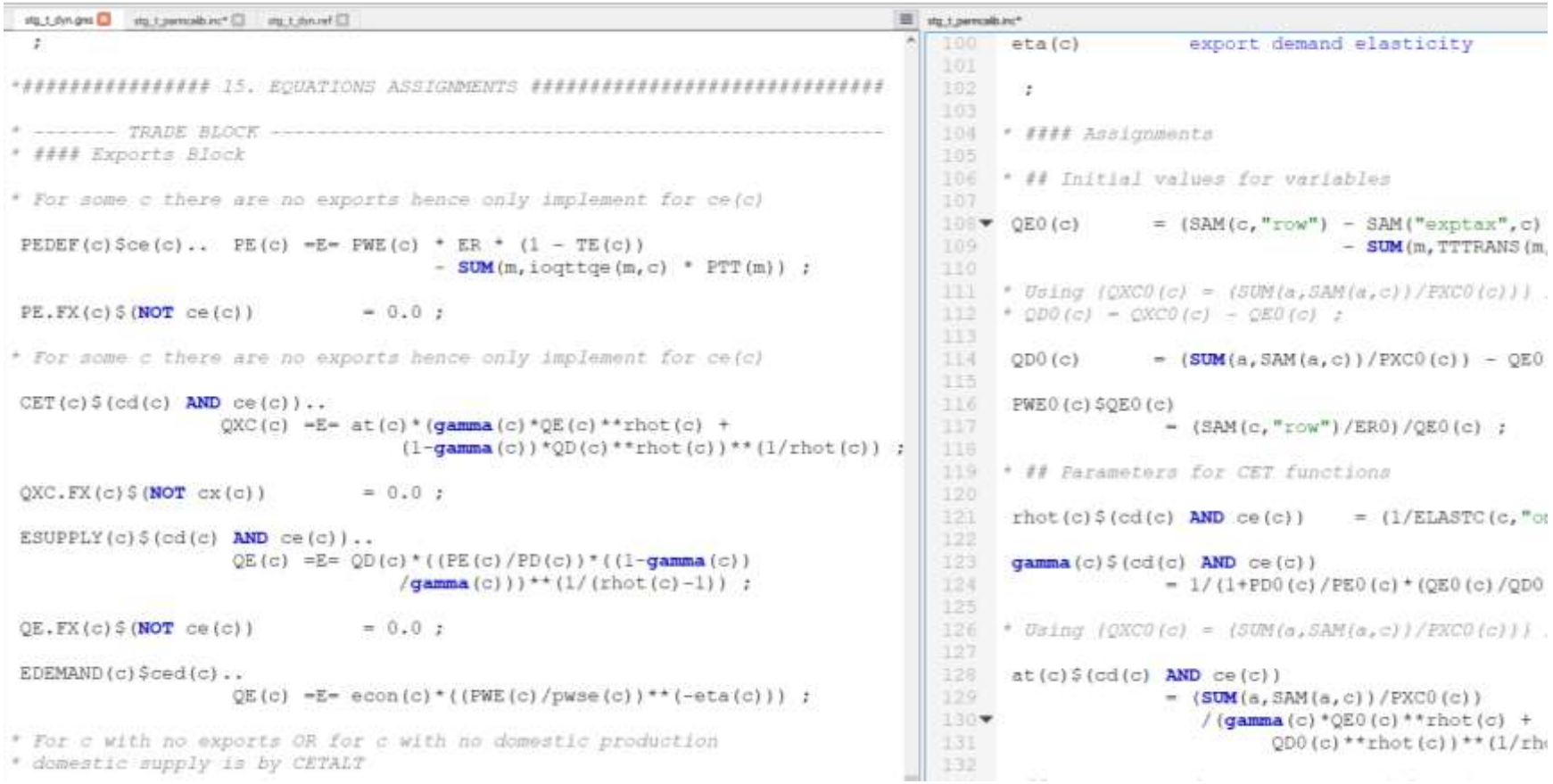


Figure 14.2 Reference File for a Modular Model: PIN Right View



There are circumstances in which it can be helpful to view two files that are related. Figure 14.3 illustrates the use of the PIN view to simultaneously view the equations in a model file (`stg_t_dyn.gms`) and the file where parameters in the equations are calibration `stg_t_parmcalib.inc`. Both files can be edited and saved as usual.

Figure 14.3 Viewing a Model and Parameter Calibration file: PIN Right View

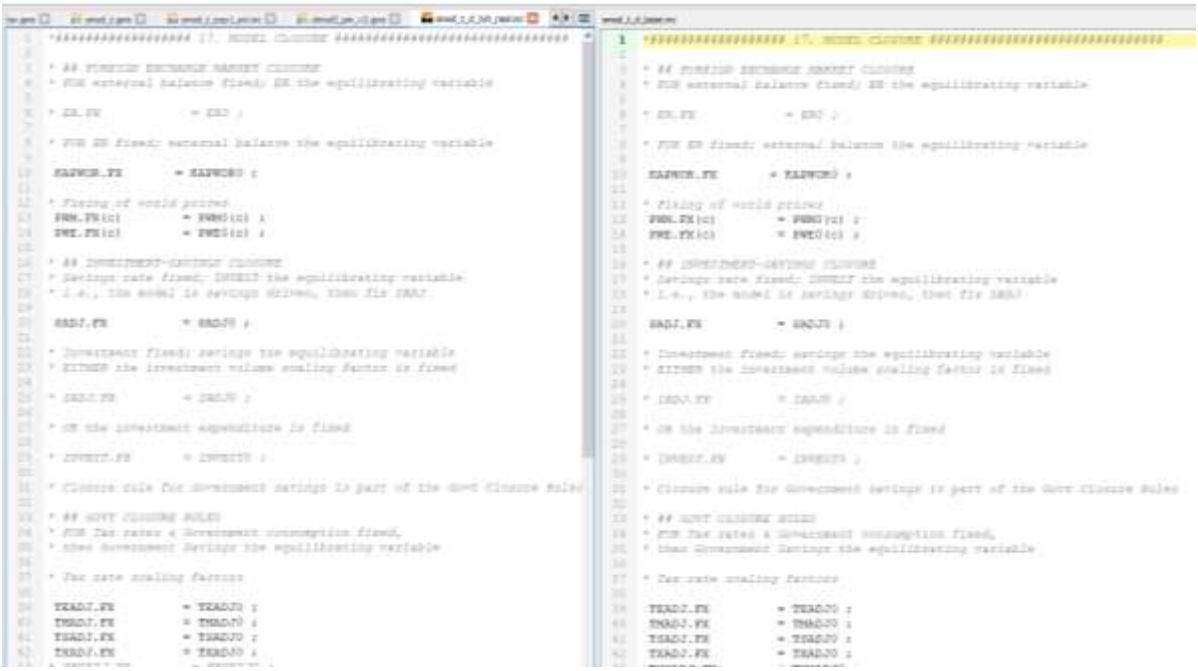


```
  ;

*################ 15. EQUATIONS ASSIGNMENTS ##############################

* ------- TRADE BLOCK ----------------------------------------------------
* #### Exports Block

* For some c there are no exports hence only implement for ce(c)

PEDEF(c)$ce(c)..  PE(c) =E= PWE(c) * ER * (1 - TE(c))
                            - SUM(m,ioqttqe(m,c) * PTT(m)) ;

PE.FX(c)$(NOT ce(c))          = 0.0 ;

* For some c there are no exports hence only implement for ce(c)

CET(c)$(cd(c) AND ce(c))..
                  QXC(c) =E= at(c)*(gamma(c)*QE(c)**rhot(c) +
                              (1-gamma(c))*QD(c)**rhot(c))**(1/rhot(c)) ;

QXC.FX(c)$(NOT cx(c))         = 0.0 ;

ESUPPLY(c)$(cd(c) AND ce(c))..
                  QE(c) =E= QD(c)*((PE(c)/PD(c))*((1-gamma(c))
                              /gamma(c)))**(1/(rhot(c)-1)) ;

QE.FX(c)$(NOT ce(c))          = 0.0 ;

EDEMAND(c)$ced(c)..
                  QE(c) =E= econ(c)*((PWE(c)/pwse(c))**(-eta(c))) ;

* For c with no exports OR for c with no domestic production
* domestic supply is by CETALT
```

```
100  eta(c)          export demand elasticity
101
102   ;
103
104  * #### Assignments
105
106  * ## Initial values for variables
107
108  QE0(c)       = (SAM(c,"row") - SAM("exptax",c)
109                               - SUM(m,TTTRANS(m,
110
111  * Using (QXC0(c) = (SUM(a,SAM(a,c))/PXC0(c))) .
112  * QD0(c) = QXC0(c) - QE0(c) ;
113
114  QD0(c)       = (SUM(a,SAM(a,c))/PXC0(c)) - QE0
115
116  PWE0(c)$QE0(c)
117                = (SAM(c,"row")/ER0)/QE0(c) ;
118
119  * ## Parameters for CET functions
120
121  rhot(c)$(cd(c) AND ce(c))     = (1/ELASTC(c,"o
122
123  gamma(c)$(cd(c) AND ce(c))
124                = 1/(1+PD0(c)/PE0(c)*(QE0(c)/QD0
125
126  * Using (QXC0(c) = (SUM(a,SAM(a,c))/PXC0(c))) .
127
128  at(c)$(cd(c) AND ce(c))
129                = (SUM(a,SAM(a,c))/PXC0(c))
130                   /(gamma(c)*QE0(c)**rhot(c) +
131                         QD0(c)**rhot(c))**(1/rh
132
```

Another use of the PIN view options is to compare two 'small' files when there may be benefits. NB: for 'large' files we recommend using specialist comparison software, see section 12 above. The illustration in Figure 14.4 has two files that define different selections of endogenous and exogenous variables: in this instance two different closure files for a CGE model.

Figure 14.4 Viewing Alternative INCLUDE file: PIN Right View



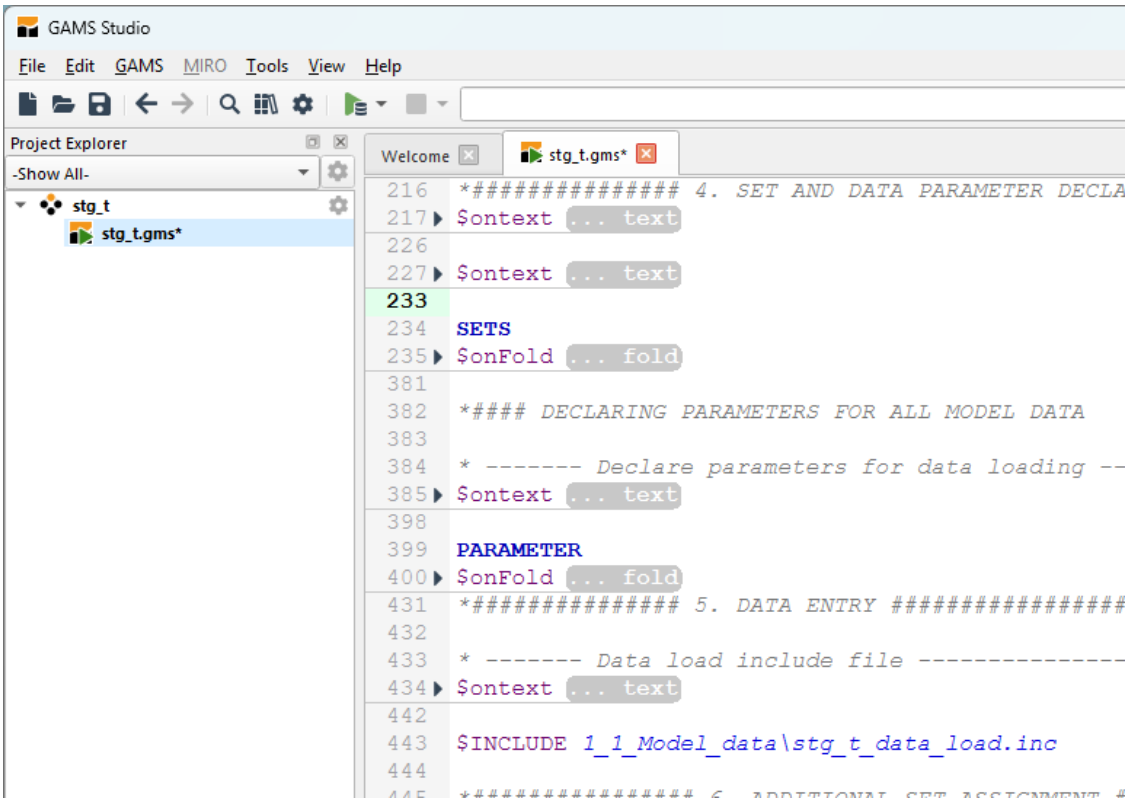
Because these files have identical structures and the process is one of exchanging one exogenous variable for another, the PIN view may be helpful.

Folding/Unfolding Text

GAMS model files can become long and then navigating through them can be labourious, even when using a reference file (section 9) and/or the search functions (section 13). One drawback of the reference file and search functions is that the file structure is not as transparent as users might like, especially if the user is learning a model. GAMS Studio now provides 2 simple options for compressing the content of the programme file, without loss of information, that can make the structure of the contents more transparent.

Folding is implemented from `View>Fold all Text blocks` (`Alt+O`). One way this may impact of the information in the editor window is illustrated in Figure 14.5 (based on STAGE_t model used in the single country course). Folding is implemented by using `$onFold` and `$offFold` in the file where the user wishes to fold the text and works in the same way as `$onText` and `$offText`. When the instruction to fold applies to both the `$onText`/`$offText` and `$onFold` `$offFold` commands. To unfold the text use `View>UnFold all Text blocks` (`Alt+Shift+O`).

Figure 14.5 Folded GAMS (gms) Model File

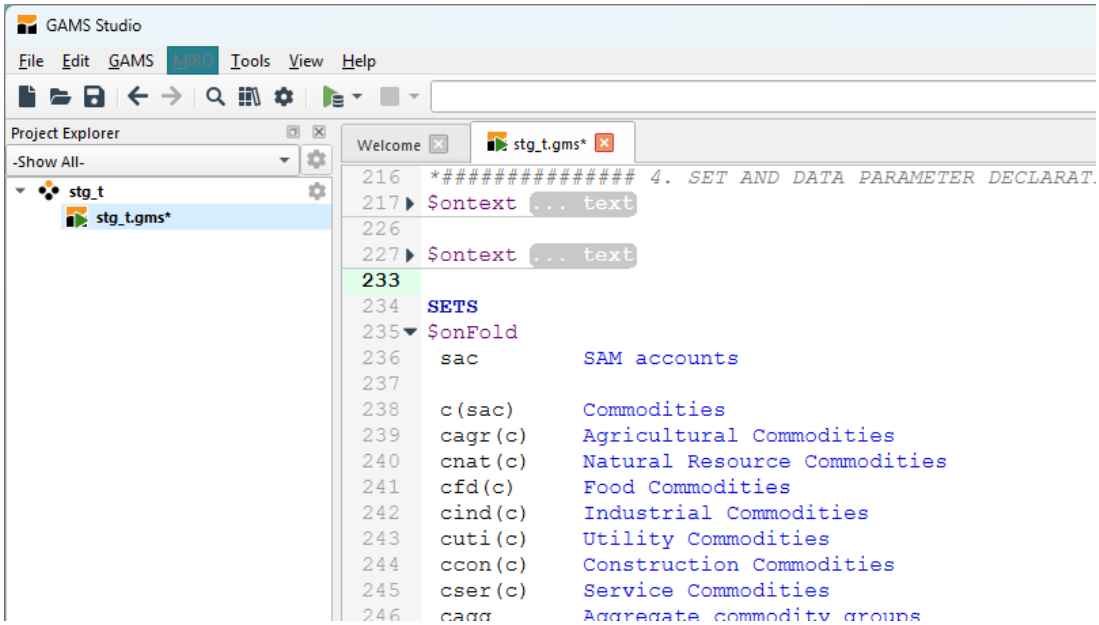


Note how in Figure 14.5 the `$INCLUDE` command is not folded since these commands are important for understanding the structure of the programme.

The little back arrowheads to the right of the line numbers for the `$onText`/`$onFold` commands allow the user to toggle between folded and unfolded text at specific places in the file. Figure 14.6 illustrates a case where a specific text block has been unfolded. Note how the black arrowhead now points downward.

Note also how the key word 'SETS' was not included within the `$onFold` `$offFold` command. This effectively indexes the code that has been folded.

Figure 14.6 Unfolded Text in a Model File



'Navigation' in GAMS Studio

Some of these navigation 'tips' are mentioned elsewhere but are collected here, despite some repetition.

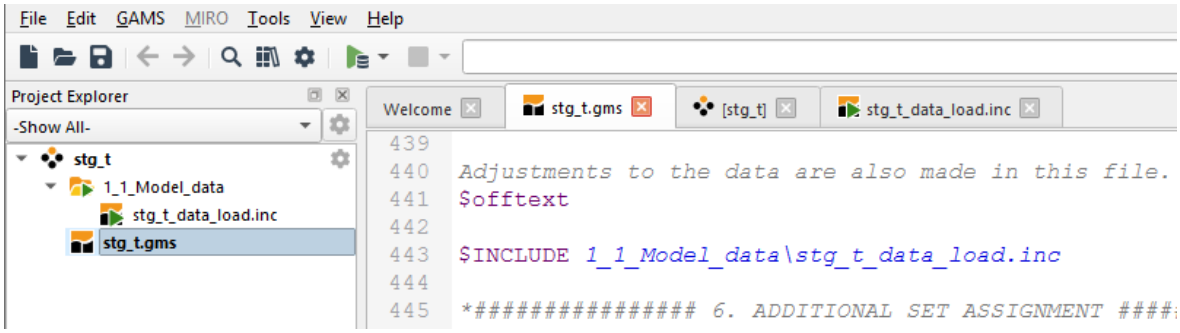
Opening Files from within Files

When GAMS compiles a programme file all the files that are referenced in the **main** file are 'included' within the compiled file, i.e., there are links in the **main** file to other files that are embedded/'included' in other files. When editing a programme, it may be necessary to edit the content of those files that are 'included'. One obvious way to access such files is `File>Open…` , but GMAS Studio provides a shortcut.

How this works is illustrated in Figure 14.7. The include file `stg_t_data_load.inc` in the sub directory `1_1_Model_data` is embedded in the programme file `stg_t.gms`. Instead of using `File>Open..` the file can simply be opened

by using `Ctrl+click` on the file name, which adds the file to the Project Explorer. A useful feature of this is that Project Explorer shows the path to the included file.

Figure 14.7 Accessing Files embedded in Other Files

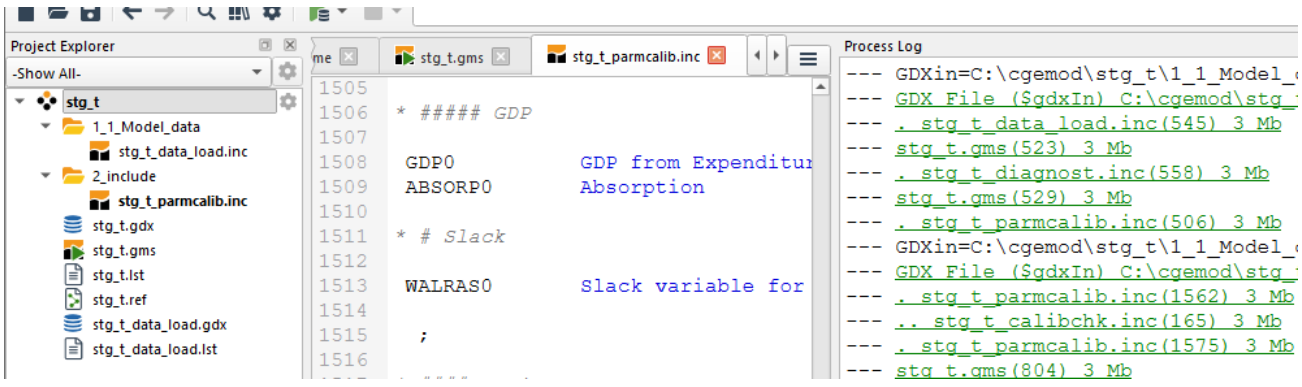


Navigating from the Process Log

The process log records, in GREEN, all files used to compile a model. Clicking on the file names opens the file in the editor window, adds the file to the Project Explorer with the path to the included file.

This is very useful when debugging a model because it, in effect, can identify the embedded file whose content stopped the model.

Figure 14.8 Accessing Files from Process Log

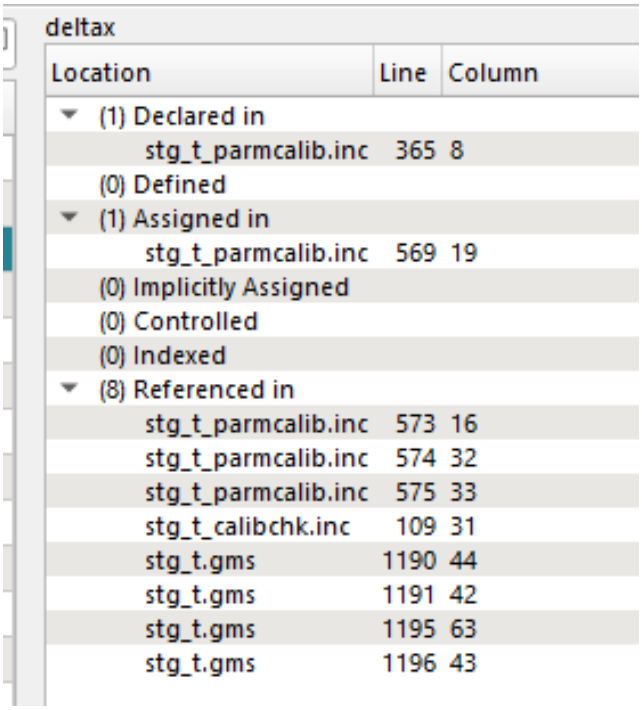


Navigating Using the Reference File

Reference files are one of the most useful features of GAMS (see section 9). The information available from a reference file allows users to identify where elements of a model are declared, defined, assigned, controlled, index and referenced. When a reference file is filter for a specific element in a model a view like that in Figure 14.9 will appear in the editor

window. Clicking on the indented entries for the parameter `deltax`, will open the respective file, e.g., `stg_t_parmcalib.inc`, at the point where the element is referred to, e.g., where `delatax` is declared in line 365 and column 8 of the file.

Figure 14.9 Accessing References within Model Code



15. Notes on Debugging a GAMS Programme in Studio

Computer programmes nearly always have numerous errors. GAMS Studio provides easy ways to find where the errors occur, especially compilation errors. The examples below are from the Syntax files in the examples provided for an introduction to Studio.

1. Compilation error messages appear in **RED** in the `Process Log` pane. `Click` on these messages and the editor window for the `[Filename].gms` file is chosen and the cursor moves to where the error was noticed. Clicking on the exclamation mark in the red circle by each error in the gms file shuts the view to where error is reported in the *.`lst` file.

- In the `[Filename].lst` file you will find information about the type of error. The error is marked by **** and a $#, where # is a number, on the line below where the error was noticed. More information on the meanings of the $# codes is given where the error is in the list file (you can also search for the string '`Error Messages`'). The information given is (usually) helpful.
- A brief description of the nature of each error is reported in the `*.lst` file.

Compilation Errors

Compilation errors are (essentially) syntax errors. In the listing file GAMS will provide useful suggestions about the likely cause of the error message. The listing file also contains markers that make it easy to find the compilation errors: search for the text string "****".

Some general principles may be helpful

- start from the top of the programme and work down – this one reason for selecting the CErr option and setting it to a low(ish) number;
- solve each error as it appears - do not skip onto the next error without a good **programming** reason;
- do not make too many changes at a time - as you become more familiar with GAMS and compilation errors the number of errors corrected at each stage will increase, but when starting out it is easy to compound errors;
- if substantial changes are made to the code save the input file with a revised name before running the file.

Common syntax errors include the following

- failing to end an operation with a ";" - GAMS often identifies this type of error as occurring on the next line of code or at the next keyword,
- assigning or using a parameter or variable before it has been declared,
- using a parameter or variable before it has been declared or assigned,
- spelling mistakes,
- the "*" used for comments and/or to comment out lines of code is NOT the first item in the line of code (NB: a space is an item in a line of code),
- set operations trying to use sets that are already under control - this is where the aliases become essential.

The solutions to most syntax problems are relatively simple. Controlling sets is one type of syntax error that can prove somewhat less tractable.

Execution Errors

Execution errors, in **BLUE**, are trickier. They can often arise because the model has been incorrectly specified. There are some ways to help find execution errors.

- DISPLAY statements for parameters and initial values for variables can be used to check the values returned by the programme.
- Outputting all model information to GDX will later prove useful, i.e., using F10 in Studio.
- The listing of values for variables from the model should be consistent with the values from the database.
- Generating a reference file is useful (see above to achieve this as a default).
- If the model is consistent the left-hand and right-hand sides of the equations should equate, or at least contain no "significant" discrepancies. This can be checked in the equation listing (controlled by the "limrow" and "limcol" options linked to the solve statement). Searching for the text string "***" is the quick way to find errors. A discrepancy is indicated by the statement "LHS = <value>", where the error is only likely to be important if the value is greater than about 1.0E-5. This indicates that the problem is probably associated with the definitions of the parameters and variables in that equation.

As with compilation errors it is unwise to try and do too much at once. If substantial changes are made to the code, it is wise to do so in a new version of the input file.

Using $STOP

We recommend setting `CErr` to less than 10^{16} (see section 10) when programming a new model, using or adapting an existing model, and when developing experiment files. This provides a default that ensures that errors are not compounded and therefore saves time.

However, when programming a new model, using or adapting an existing model, and calibrating a model using the `$STOP` option can be extremely useful. This option allows the user to stop the programme at user defined part of the model,[17] e.g.,

1. after loading data,
2. after assigning values to parameters,
3. after initialising variables.

The objective is to stop the model before errors are compounded, e.g., if assigned parameter values are used to initialise variables error in parameter values conflict with the reasons for initialising variables.[18]

Using the Debugger

GAMS Studio recently introduced a DEBUGGER that can be triggered by using '`RUN with Debugger`' (`F11`) or '`Step start with Debugger`' (`Shift+F11`). Preliminary experiments with the DEBUGGER suggest this will be a useful option, especially through its integration with the PIN VIEW (see section 14).

This section will be expanded when we have more experience using the DEBUGGER.

16 We typically set `CErr` to 5.

17 The code structure used for our models was, in part, influenced by the using $STOP to aid debugging.

18 This is problematic when equation code is used to initialise variable.