Transport Problem Exercises with GAMS Studio

This document is subject to revision. GAMS Studio is still a relatively new programme that is still evolving. The next review is planned for September 2023.

Contents

1. Introduction

GAMS, arguably as with all languages, is best learnt by doing. These exercises adopt that principle. These instructions are written assuming you are using these exercises prior to taking a course offered by cgemod.

Aims

The aims of this set of exercises are to develop

1. an understanding of the content and structure of GAMS models;
2. an understanding of the main components of a GAMS model;
3. an ability to identify and correct syntax errors;
4. an ability to identify and correct execution errors; and
5. an ability to modify a simple GAMS programme.

Objectives

At the end of the set of exercises the participant will

1. understand the structure of GAMS models;
2. understand the main components of a GAMS model;
3. be able to identify and correct common syntax errors;
4. appreciate how to correct common execution errors; and
5. be able to modify a simple GAMS programme.

Process

These exercises work by exploring a model; conducting exercises that generate syntax and execution errors; modifying a model; and running some simple experiments.

The basic code is provided as ‘`trans1.gms`’, which is a version of the standard GAMS transport problem used by GAMS as their tutorial. All the exercise assume that the user is working with GAMS Studio as the editor.[1]

A guide to GAMS Studio is available at www.cgemod.org.uk/

[1] Instructions for using GAMS with GAMSIDE, and these exercises setup for GAMSIDE will remain available on www.cgemod.org.uk until, at least late 2023.

2. Set up

Before conducting these exercises, it is necessary to setup your PC to access the files required for the exercises. These exercises, and the associated instructions, will assumed that your PC has been setup inline with the instructions in this section. The reasons for this are simple: first, we can only provide instructions that accurately reflect a setup that is known to us, second, because the setup is known to us, we can anticipate likely difficulties, and third, the setup we advocate has been tested. We anticipate that users of GAMS will, with experience, develop their own ways of working; hence the setup we advocate is designed to provide a basis upon which users can develop their own way of working. We acknowledge that the setup we advocate reflects, to a greater or lesser, extent out way of working.

The instructions for these exercises were developed on Windows based PC. We know that GAMS and GAMS Studio are available for Apple and Linux based PC. Some testing has been conducted for **an** Apple PC, but we are not users of Apple or Linux PCs so we cannot guarantee that the instructions are accurate for Apple OR Linux PCs.

There are two steps that must first be taken

1. install GAMS and GAMS Studio on your PC, instructions for this are available in ‘Intro to GAMS Studio.pdf’ (www.cgemod.org.uk) (we assume that the path for GAMS is C:\GAMS); and
2. open Windows Explorer
   a. create the directory `C:\cgemod` – this is the master directory that will be assumed for all courses offered by cgemod;
   b. create the directory `C:\cgemod\downloads` – this is the directory we will expect you to use to store all the course materials provided for cgemod courses.

3. Exploring a GAMS Model

This exercise is designed to help you understand the structure of a GAMS model and the content of the files generated when that model has been run. This section repeats a lot of material in 'Intro to GAMS Studio.pdf'; this is deliberate.

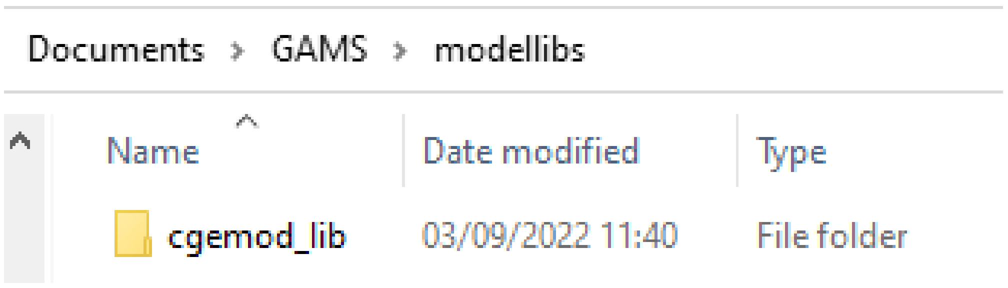
The steps involved are these

1. Create a directory `C:\cgemod\cgemod_lib` to contain the library of computer code you will build during the course. You will add files to this directory regularly as part of a cgemod course.
2. Download the `trans.zip` file from the cgemod site (http://cgemod.org.uk/int_gams.html) and save it in your `downloads` directory.
3. Unzip the contents of the file `trans_lib.zip` into the library directory `C:\cgemod\cgemod_lib`. WinZip may by default unzip these files into a directory, usually called `trans`. If so, you will need to copy these files and paste them into your library directory `cgemod_lib`. The files in this directory must NOT be in sub directories.[1]
4. Create the directory `C:\cgemod\trans` (this is the path that will be assumed in these instructions; do not create the directory on a drive that synchonises in real time with an external drive, e.g., OneDrive, because this can cause inconsistencies between versions of documents
5. Open GAMS Studio.
6. Copy the directory `cgemod_lib`;
7. Open Studio settings - `File > Settings` or `F7` or the `Settings Wheel` on the toolbar and go to the `Misc` tab.
8. Click on the 'Open Location' button, which will open a Windows Explorer window. You need to make sure the directory …`\GAMS\modellibs` is open.
9. Then Paste the `cgemod_lib` directory into the `modellibs` directory.
10. Close the Studio Settings window, remembering to click OK at the bottom of the `Misc` tab.

[1] The default settings in WinZip now make you do this as a two-stage process. In WinZip 'classic view' you can avoid the two-step process, but it is fiddly and error prone. It is called progress!!!!

The result should be like that in Figure 3.1. Note how the `modellibs` directory is in GAMS directory in the user's documents directory.

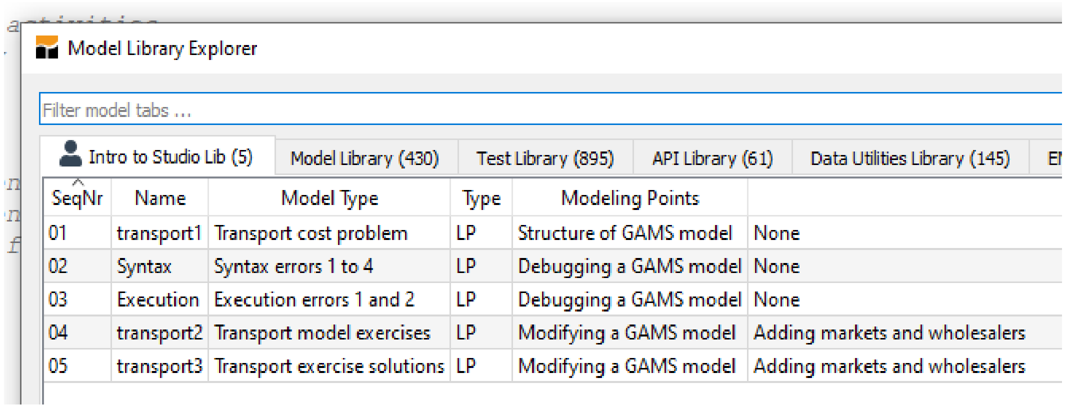
Figure 3.1 GAMS\modellibs Directory


Documents › GAMS › modellibs

Name	Date modified	Type
cgemod_lib	03/09/2022 11:40	File folder

Now reopen the `Model Library Explorer`, `F6`., and expand the window to the right and then click on the tab `Intro to Studio Lib`; you may need to scroll across the tabs the find the `Intro to Studio Lib`. The results should look like Figure 3.2. Note the 'person' figure associated with a User Model Library.

Figure 3.2 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

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 in that our libraries supply multiple files with each download while GAMS libraries 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.

In Studio it is necessary that we establish a new directory for each new project so that a destination directory is specified for the download. To do this carry out the following

1. In Windows Explorer create a directory, say, `C:\cgemod_training`. We suggest doing this on the top level of you working drive (this would be the C drive for us.)
2. In Studio select `File>New Project`. This opens a window – Import Project – that is essentially a Windows Explorer Window, see Figure 3.4.
3. Select the working drive: in the illustration this is SSD (C:).
4. We wish to add a new (sub) directory in the directory `C:\cgemod_training`; so double click on `C:\cgemod_training` and then right-click in the left-hand panel and choose `New>Folder` from the menu. (see Figure 3.4)
5. Give the New Folder a name. In this example we call the folder `C:\cgemod_training\transport`. (Figure 3.5 illustrates the sub-directory that should be created).
6. The Studio panels should appear automatically once the new directory has been created. (Figure 3.6 illustrates how the Project Explorer and Editor panels should appear).
7. In Studio press `F6` and in the Model Library Explorer select the Intro to Studio Lib and then select the library file `transport1`, which is SeqNr: 1, and choose Load (or double click of the name). (Figure 3.7)
8. The `trans.gms` model will now be displayed in the editor window and be listed in the Project Explorer as being in the project Transport. (see Figure 3.8).
9. If you right-click on the project name and select `Open Location`, you will see that 4 files have been downloaded to the directory `C:\cgemod_training\transport`. (See Figure 11.10).
    a. If the file `trans.gms` does NOT open in the editor pane the probable cause is that either you did not establish the library correctly or the file did not load from your working directory. Check first that the file `trans.gms` is in the library (`C:\cgemod\cgemod_lib`); then check if has been downloaded to your working directory (`C:\cgemod_training\transport`); and finally check to see if the file `trans.gms` is in the working directory. As should be evident the likely problem will be associated with the setting up of the directories. If this is the case delete what you have done and start again.

10. Studio can now be used to work with the transport model as done previously in this introduction and in the associated exercises.

This is how each new project can be created from files in a User Model Library. It also demonstrates how multiple files can be provided from a single library entry. This method is used through all the course offered by cgemod.

It is also a method for establishing a new project in Studio when starting from scratch.

Figure 3.4 Create a New Directory from New Project Menu

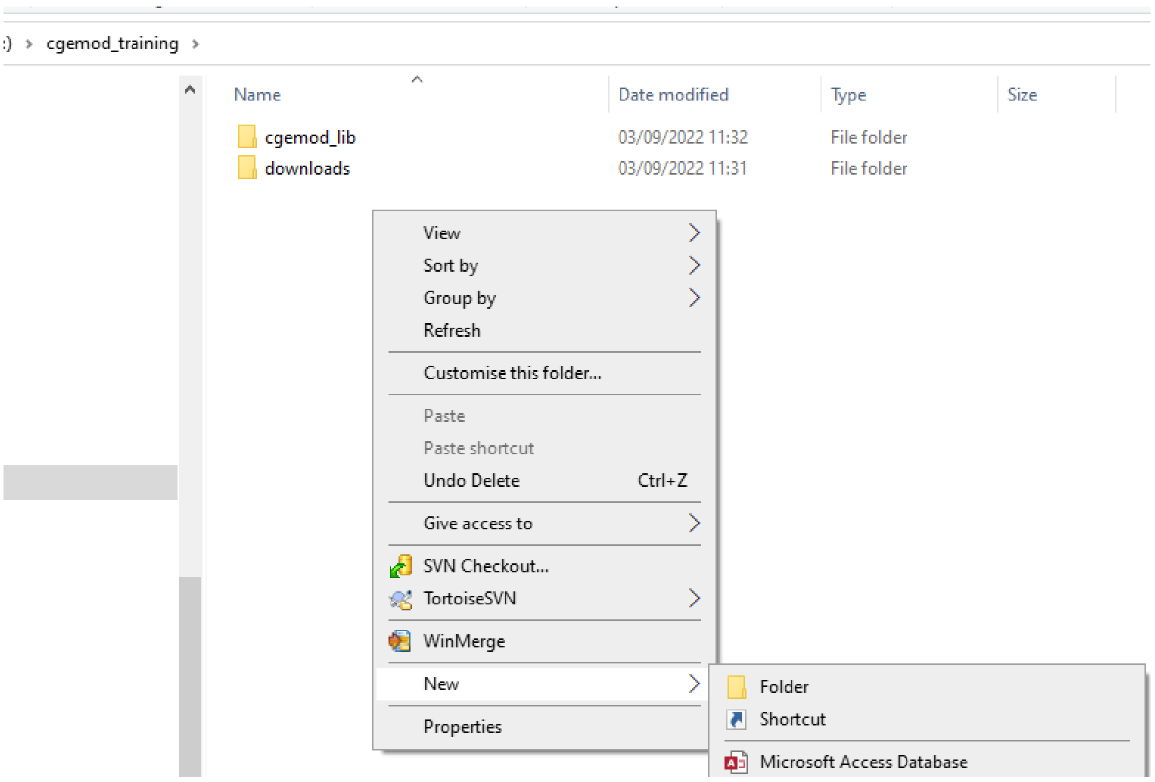


Figure 3.5 cgemod_training/transport Directory

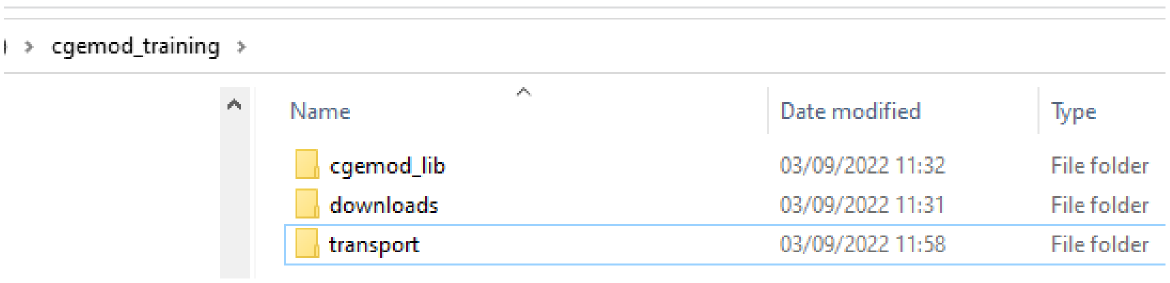
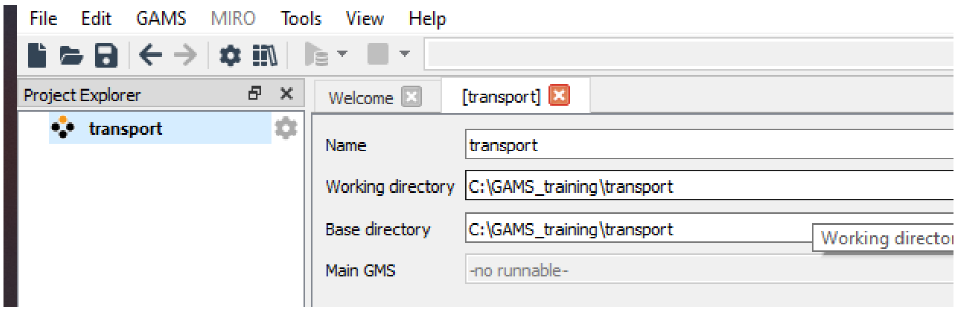
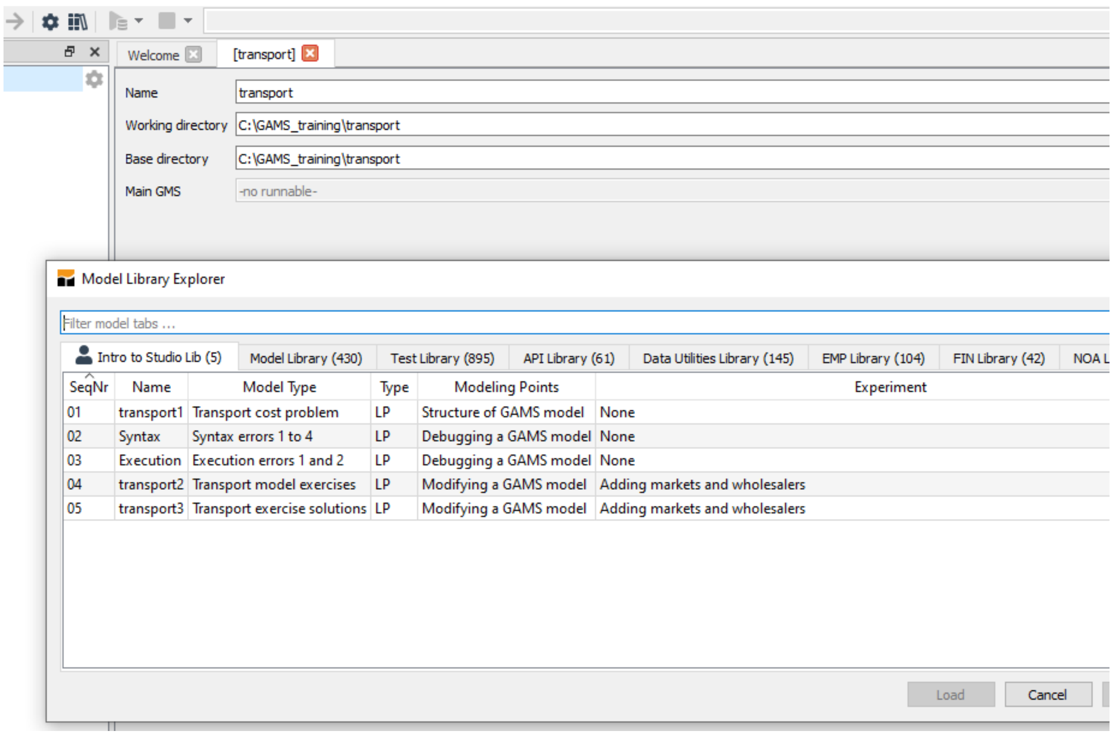


Figure 3.6 New Project View in Studio



Name	transport
Working directory	C:\GAMS_training\transport
Base directory	C:\GAMS_training\transport
Main GMS	-no runnable-

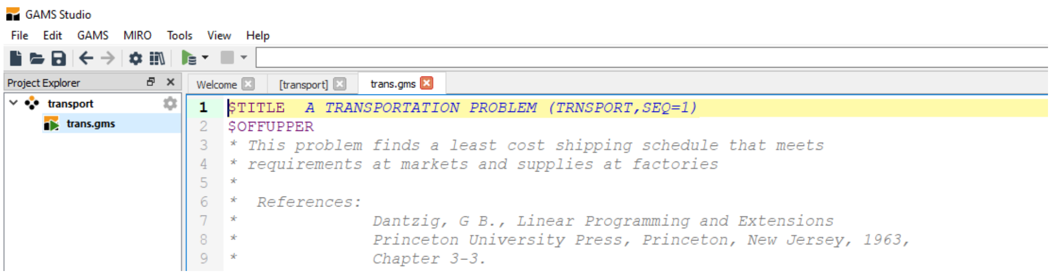
Figure 3.7 User Model Library in Studio



Name	transport
Working directory	C:\GAMS_training\transport
Base directory	C:\GAMS_training\transport
Main GMS	-no runnable-

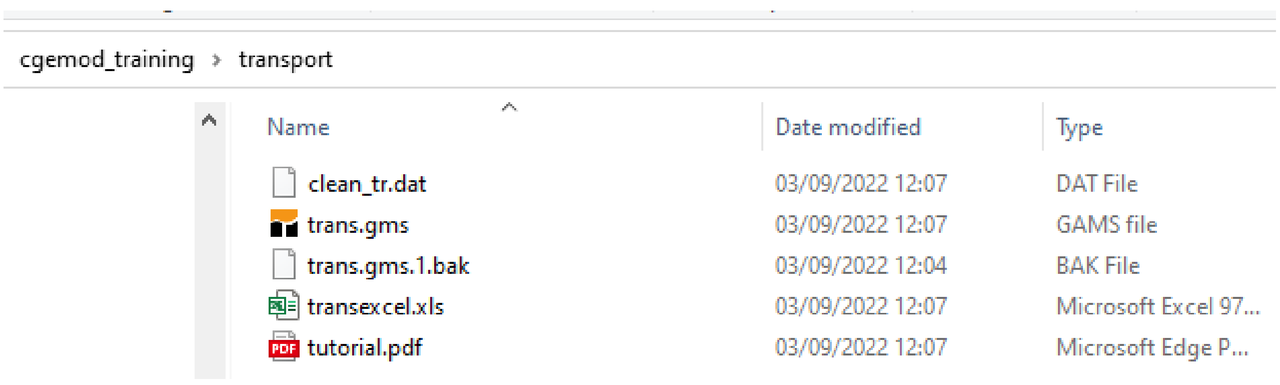
SeqNr	Name	Model Type	Type	Modeling Points	Experiment
01	transport1	Transport cost problem	LP	Structure of GAMS model	None
02	Syntax	Syntax errors 1 to 4	LP	Debugging a GAMS model	None
03	Execution	Execution errors 1 and 2	LP	Debugging a GAMS model	None
04	transport2	Transport model exercises	LP	Modifying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

Figure 3.8 trans.gms in project Transport



```
$TITLE   A TRANSPORTATION PROBLEM (TRNSPORT,SEQ=1)
$OFFUPPER
* This problem finds a least cost shipping schedule that meets
* requirements at markets and supplies at factories
*
*  References:
*             Dantzig, G B., Linear Programming and Extensions
*             Princeton University Press, Princeton, New Jersey, 1963,
*             Chapter 3-3.
```

Figure 3.9 Directory on C:\cgemod_training\Transport



cgemod_training › transport

Name	Date modified	Type
clean_tr.dat	03/09/2022 12:07	DAT File
trans.gms	03/09/2022 12:07	GAMS file
trans.gms.1.bak	03/09/2022 12:04	BAK File
transexcel.xls	03/09/2022 12:07	Microsoft Excel 97...
tutorial.pdf	03/09/2022 12:07	Microsoft Edge P...

Use the file `trans.gms` in GAMS Studio together with the GAMS tutorial, which will have been downloaded to the working directory (you may want to print off a copy), and the (Intro to GAMS Studio.pdf' to work through the code for the transport problem making sure that you understand what each line of the code means. Particular attention should be devoted to the following

1. Keywords (in blue in the standard GAMS Studio colouring for *.gms files).
2. The distinction between declaration and assignment (data entry) statements for parameters.
3. The declaration of variables and the different types and attributes of variables.
4. The declaration and assignment of equations.
5. The documentation facilities available within the code file.
6. The model and solve statements.
7. Display statements.

Once you have familiarized yourself with the code, the model should be run. (Before running the model make sure that GAMS Configuration is set so that a reference file will be automatically created, see Section 10, sub section 'Configuring Studio to Run Commands by Default').

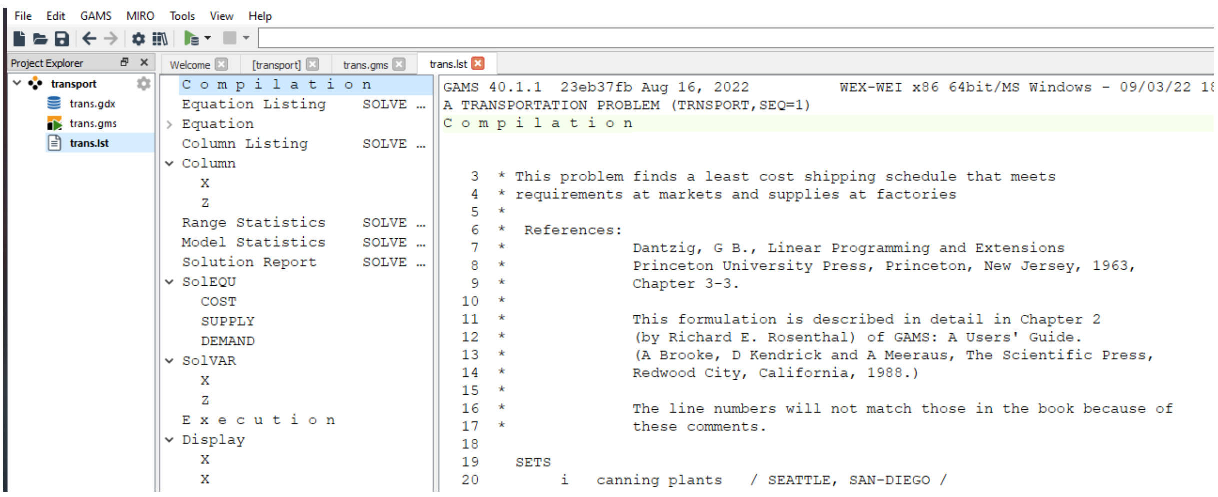
Run the transport model by selecting `F10` or `GAMS>Run with GDX creation`. The run command submits the programme file for compilation and, presuming it compiles without error, executes the programme. (If you wish solely to compilate the model choose `Shift+F10`.)

- The progress of a submitted programme is recorded in the `Process Log` pane. Information recorded in this pane is very useful and provides an easy way to debug a programme file (see below). The information from the `Process Log` is recorded as `[Filename].log` and saved in the project directory.
- The layout of the various panes is a matter of personal choice, but for the cgemod courses we assume you adopt the default settings as we do for the illustrations.
- The GAMS output file is returned automatically as a tabbed file in the editor window as '`trnsport.lst`'.

Explore the `Process Log` or open the `trans.log` file and explore its contents. This will report, *inter alia*, details of the model, the iteration steps taken to reach a solution, the fact that an optimal value was found and the value of the objective function.

Now look at the `trans.lst` file and explore its contents. This begins by repeating the model's code and then details the equations, the model statistics, a solve summary, the solution equations (`SolEQN`) and variables (`SolVAR`), and finally reports the levels and marginal values for the variable *X* (see Figure 3.10). Note how there is an index produced for each list file (see below): with the transport problem, the list file is short but soon the list files will start to grow, rapidly.

Figure 3.10 List File for the Transport Model



```
C o m p i l a t i o n
Equation Listing     SOLVE ...
Equation
Column Listing       SOLVE ...
Column
  X
  Z
Range Statistics     SOLVE ...
Model Statistics     SOLVE ...
Solution Report      SOLVE ...
SolEQU
  COST
  SUPPLY
  DEMAND
SolVAR
  X
  Z
E x e c u t i o n
Display
  X
  X
```

```
GAMS 40.1.1  23eb37fb Aug 16, 2022        WEX-WEI x86 64bit/MS Windows - 09/03/22 1
A TRANSPORTATION PROBLEM (TRNSPORT,SEQ=1)
C o m p i l a t i o n


   3  * This problem finds a least cost shipping schedule that meets
   4  * requirements at markets and supplies at factories
   5  *
   6  *  References:
   7  *              Dantzig, G B., Linear Programming and Extensions
   8  *              Princeton University Press, Princeton, New Jersey, 1963,
   9  *              Chapter 3-3.
  10  *
  11  *              This formulation is described in detail in Chapter 2
  12  *              (by Richard E. Rosenthal) of GAMS: A Users' Guide.
  13  *              (A Brooke, D Kendrick and A Meeraus, The Scientific Press,
  14  *              Redwood City, California, 1988.)
  15  *
  16  *              The line numbers will not match those in the book because of
  17  *              these comments.
  18
  19    SETS
  20         i   canning plants   / SEATTLE, SAN-DIEGO /
```

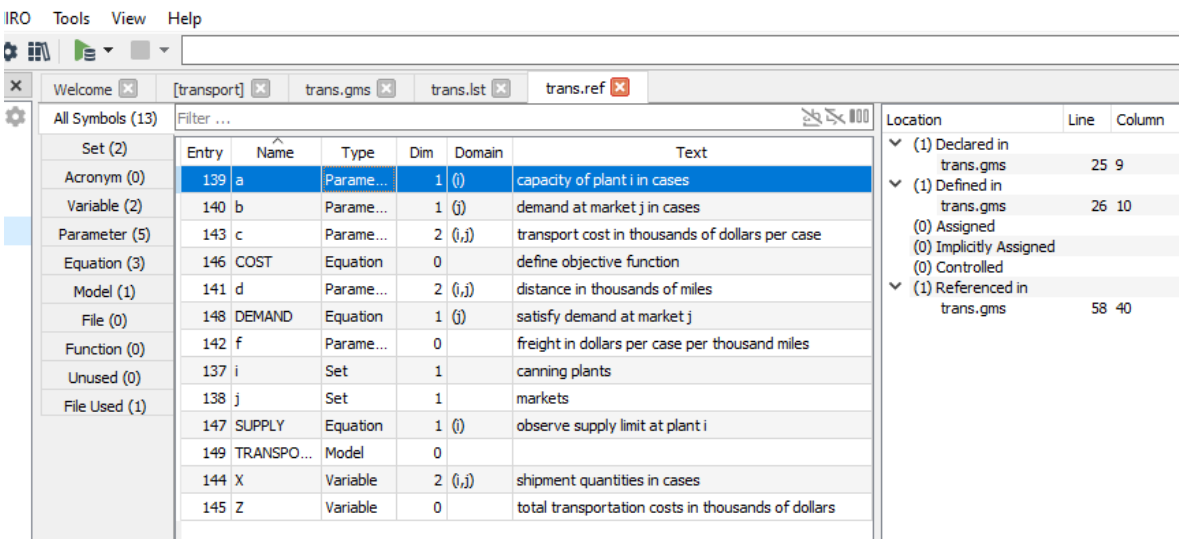
The model file for the transport problem is small and it is easy to navigate it by scrolling through the file in GAMS Studio. This method will some become inefficient, so it is good idea to be able to explore a model's using some form of indexing; a reference file is the best way to do this for a GAMS programme. A reference file allows you to find components of the programme file by double clicking in the certain places so that the programme file comes to the fore with the cursory in the appropriate places.

For instance, assume you want to find where the parameter `a` was declared. Open the reference file (`transport.ref`) , click on the Parameters tab and then double click on the cell in the row for `a` and the entry for `Declared in` (see Figure 3.11).

Use the reference file to find the following

1. The declaration and definition statement for the COST equations, and where the COST equation was implemented.
2. Where the set *i* was defined, and used as a control.
3. Where the model was defined, and implemented.

Figure 3.11 Reference File for Transport Model



All Symbols (13)
Set (2)
Acronym (0)
Variable (2)
Parameter (5)
Equation (3)
Model (1)
File (0)
Function (0)
Unused (0)
File Used (1)

Entry	Name	Type	Dim	Domain	Text
139	a	Parame...	1	(i)	capacity of plant i in cases
140	b	Parame...	1	(j)	demand at market j in cases
143	c	Parame...	2	(i,j)	transport cost in thousands of dollars per case
146	COST	Equation	0		define objective function
141	d	Parame...	2	(i,j)	distance in thousands of miles
148	DEMAND	Equation	1	(j)	satisfy demand at market j
142	f	Parame...	0		freight in dollars per case per thousand miles
137	i	Set	1		canning plants
138	j	Set	1		markets
147	SUPPLY	Equation	1	(i)	observe supply limit at plant i
149	TRANSPO...	Model	0		
144	X	Variable	2	(i,j)	shipment quantities in cases
145	Z	Variable	0		total transportation costs in thousands of dollars

Location	Line	Column
(1) Declared in		
trans.gms	25	9
(1) Defined in		
trans.gms	26	10
(0) Assigned		
(0) Implicitly Assigned		
(0) Controlled		
(1) Referenced in		
trans.gms	58	40

3. Syntax and Execution Error Exercises

These exercises are designed to help you to start solving your own programming errors. **Everyone** who programmes a computer makes errors **every time** they write a programme; hence it is essential to develop the skills that allow you to find errors and correct them. Starting to learn how to debug programmes early in the learning experience is very useful, it is also easier than leaving it until later since you will be working with simple programmes. Indeed, the clichés about 'learning from mistakes' could have been devised by reference to computer programming.

Syntax and execution error exercises can be conducted using a single directory and several projects.

Syntax Error Exercises

For these exercises we will need a new directory.

1. In Studio select `File>New Project`. This opens a window – Import Project – that is essentially a Windows Explorer Window, see Figure 3.4.
2. Select the working drive.
3. Add a new (sub) directory in the directory `C:\cgemod_training`; using right-click in the left-hand panel and choose `New>Folder` from the menu and name the new folder `C:\cgemod_training\syntax`.
4. In Studio press `F6` and in the Model Library Explorer select the `Intro to Studio Lib` and then select the library file `syntax`, which is SeqNr: 2, and choose Load (or double click of the name).

There are **four** *.gms files, labeled `trnserr#.gms` (where # is a number from 1 to 4), that are all perturbations of the basic transport linear programme. These four 'files' contains different types of syntax errors that are intended to be progressively more difficult to solve; in each case the solve statement is **not** implemented because of previous errors.

You should run each of the `trnserr#.gms` files in turn and solve the problems each presents before moving onto the next problem file. For each `trnserr#.gms` file there is a solution file, labeled `trnsol#.gms`. The solution file contains corrections for each of the errors and is annotated to provide explanations. Search for the string 'SMcD' in each solution file. You should modify the `trnserr#.gms` file and make sure you can correct the errors making as little use as possible of the solutions provided.

In Studio there are two obvious options for conducting these four syntax error exercises

1. Open each gms file, `trnserr#.gms`, in the same project, making sure that you change the `Main File` for each exercise in turn; or
2. Open each gms file, `trnserr#.gms`, using `File>Open in a New Project`; this will keep all the files associated with each exercise together in the same project although they will all be saved to the directory `C:\cgemod_training\syntax`.

Execution Error Exercises

From within Studio create a New Project with the directory name `C:\cgemod_training\execute` and populate the directory from the library `Intro to Studio Lib` and then select the library file `Execution`, which is SeqNr: 3. (If in doubt about the process consult the 'Intro to GAMS Studio.pdf').

There are **two** *.gms files, labeled `trnserr#.gms` (where # is a numbers 5 and 6), that are perturbations of the basic transport linear programme. These two files contain different types of execution errors - in each case the solve statement **is** implemented but the model does not execute properly because of previous errors that did **not** include syntax errors. Execution errors are typically more difficult to solve than syntax errors.

You should run each of the trnserr#.gms files in turn, and solve the problems each presents, before moving onto the next problem file. When working with this set of problems in Studio, you can adopt either of obvious two options used for the syntax exercises (with suitable changes to file/directory names).

For each `trnserr#.gms` file there is a solution file, labeled `trnsol#.gms`. The solution file contains corrections for each of the errors and is annotated to provide explanations. Search for the string 'SMcD' in each solution file. You should modify the `trnserr#.gms` file and make sure you can correct the errors making as little use as possible of the solutions provided.

4. Transport Model Exercises

Having completed the syntax and execution error exercises it is relatively simple to begin to modify and extend a simple GAMS programme. There are four exercises

1. Changing Unit Transport Costs
2. Changing Distances
3. A New Market
4. Intermediate Markets

The first two exercises involve simply changing data and examining the changes in the results. The aims of these exercise are to

1. improve understanding of a GAMS programme,
2. develop programme organisation skills, and
3. develop an ability to analyse results.

The first three exercises are relatively straightforward and should be regarded as essential parts of the course. The fourth exercise, adding intermediate markets, requires changes to the equations and is therefore more complex. If you cannot complete the fourth exercise in the first week, it is recommended that you return to it later.

Changing Unit Transport Costs

Create a directory called `trans2a` and then get the transport model from the course library (`cgemod_lib`). The files for the transport model are SeqNr 1 with the name `Transport1`. The basic file is identical to the `trans.gms` file in the GAMS Model Library but other files are also downloaded to the directory.

Start from the file '`trans.gms`' and save the file with a new name, e.g., `unitcost.gms`. Then experiment with a series of systematic changes in the per unit transport cost. How do these changes affect deliveries to different markets and total transport costs?

Changing Distances

Using New Project in Studio create a directory `C:\cgemod_training\trans2` and then download the transport model to that directory. The files for the transport model are SeqNr 1 with the name `Transport1`. If GAMS asks if you want to overwrite files with the same name this implies you are trying to download into one of your earlier directories; stop and restart the process.

Start from the file '`trans.gms`' and save the file with a new name, e.g., `distance.gms`. Then experiment with a series of systematic changes in the distances between the plants and the markets. How do these changes affect deliveries to different markets and total transport costs?

HINT: if you have used F10 to run the model the results will all be available in the file `distance.gdx`. The results of interest will be the levels results for the variables X and Z. You can easily copy and paste these into Excel to make the comparisons easy. (NB: the attributes options.)

<u>A New Market</u>

Using New Project in Studio create a directory `C:\cgemod_training\trans3` and then download the transport model to that directory. The files for the transport model are SeqNr 4 with the name `Transport2`. If GAMS asks if you want to overwrite files with the same name this implies you are trying to download into one of your earlier directories; stop and restart the process.

Start from the file '`trans.gms`' and save the file with a new name, e.g., `newmkt.gms`. Then adapt the GAMS code to include another market. Assume the new market is Charlestown and the demand at that market is for 225 units. It is left to you to decide how far Charlestown is from Seattle and San Diego, and by how much to increase production at either or both of Seattle and San Diego (the combined increase in total production capacity must be at least 175 units; it can be more). Experiment with changing various parameters. How do the changes you have made affect the results? (A worked solution is available in Transport3 as `newmkt_sol.gms`.)

Be systematic and collect your results in one file.

<u>Intermediate Markets</u>

Using New Project in Studio create a directory `C:\cgemod_training\trans4` and then download the transport model to that directory. The files for the transport model are SeqNr 5 with the name `Transport3`. If GAMS asks if you want to overwrite files with the same name this implies you are trying to download into one of your earlier directories; stop and restart the process.

Many marketing systems are characterized by intermediate markets, e.g., wholesale markets. This exercise adapts the standard transport problem to introduce two intermediate markets, and hence the production of factories should first be sent to the intermediate markets and then transshipped to the final destinations. The transport costs should be different between the factories and the intermediate markets and between the intermediate markets and the final destinations. It is left to you to determine names for the intermediate markets, the locations of the intermediate markets and hence the distances between the various nodes in the model, the transport costs between nodes, and the quantities supplied by each factory and demanded at each destination.

Start from the file '`trans.gms`' and save the file with a new name, e.g., `wholesale.gms`. Then adapt the GAMS code to include intermediate markets. Note this exercise requires that you modify equations so you should first write out the new model BEFORE starting to code. Experiment with changing various parameters. How do the changes you have made affect the results? (A worked solution is available in Transport3 as `wholesale_sol.gms`.)