

## Transport Problem Exercises

### Contents

Contents.....	1
1. Introduction .....	2
2. Exploring a GAMS Model .....	3
3. Syntax and Execution Error Exercises .....	8
Syntax Error Exercises .....	8
Execution Error Exercises .....	9
4. Transport Model Exercises.....	11
Changing Unit Transport Costs.....	12
Changing Distances.....	12
A New Market.....	12
Intermediate Markets.....	13

## **1. Introduction**

These exercises aim to develop your understanding of GAMS 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.

## 2. Exploring a GAMS Model

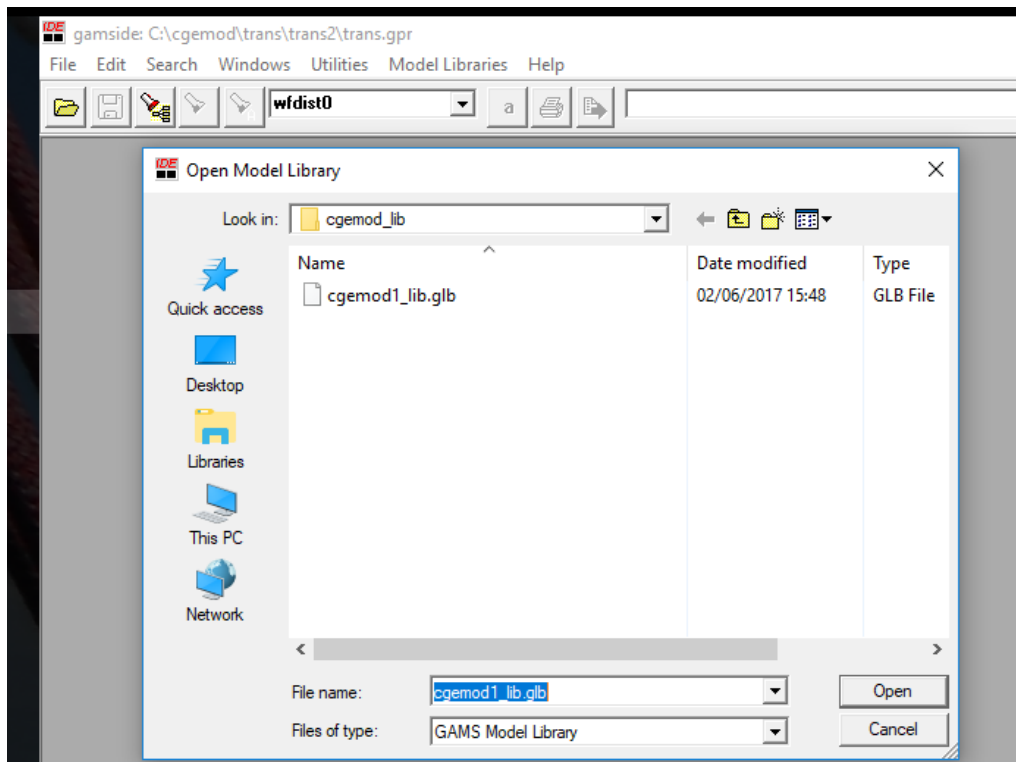
This exercise is designed to help you understand the content of a model and the content of the files generated when that model has been run.

There are several steps that need to be taken now.

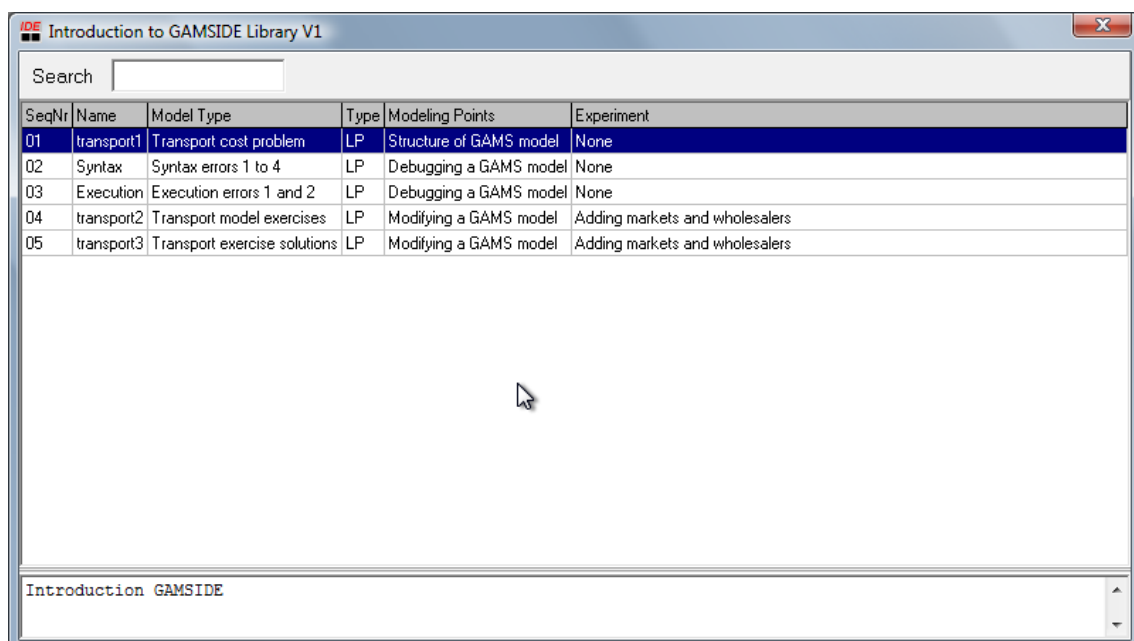
1. In the directory `C:\cgemod\trans` create a `trans2` (the path should read `C:\cgemod\trans\trans2`).
2. Then download the zip file `trans_lib.zip` from the Moodle site (it is one of the files in the folder of downloads for Topic 1.2).
3. We recommend that you create a directory for your downloads in your master directory, i.e., `C:\cgemod\downloads`.
4. Create a directory `C:\cgemod\cgemod_lib` to contain the library you will build during the course. You will add files to this directory each week. (You should have created this directory while working with GAMSIDE in Topic 1.1).
5. 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.<sup>1</sup>
6. Open GAMSIDE and create a project file in the directory `trans2`, call it, say, `trans` (the name is not important).
7. In GAMSIDE select `File>User Model Library` and find the directory `C:\cgemod\cgemod_lib`. The screen should look like the screenshot below. Open the library index file `cgemod1_lib.glb`.

---

<sup>1</sup> 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. It is called progress!!!!

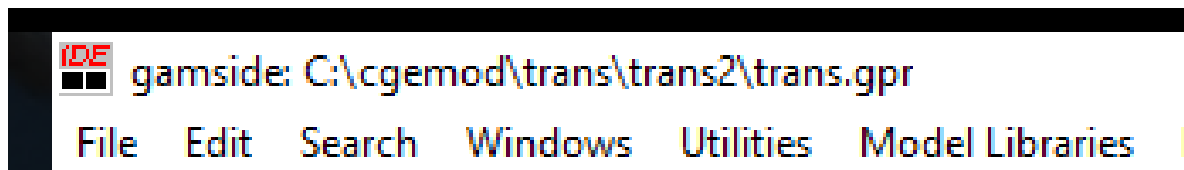


8. When you have selected the index file `cgemod1_lib.glb`, the library window shown in the screenshot below will appear.



9. Double click on the first item in the list (`transport1`). The file `trans.gms` should open automatically in GAMSIDE. If it does not, choose open and select the file `trans.gms` from the directory `C:\cgemod\trans\trans2`).

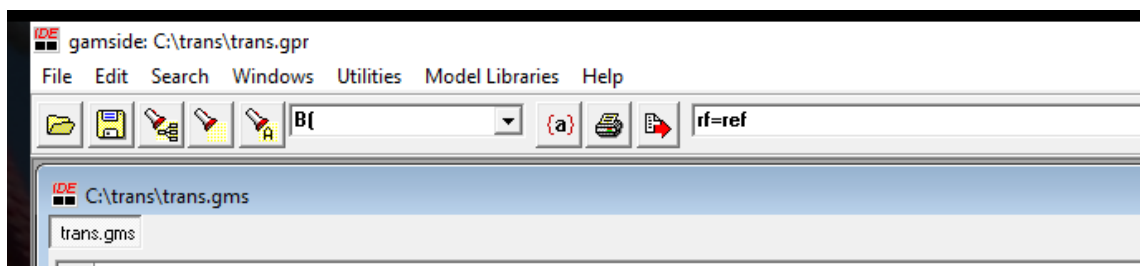
10. If the file `trans.gms` does NOT open automatically in GAMSIDE, and you cannot find it in the directory `C:\cgemod\trans\trans2`, it is probable that you have not created the project file in the directory `C:\cgemod\trans\trans2`, or that you have redirected GAMSIDE to another directory.
11. The easiest way to check this is look in the top right hand corner of GAMSIDE. This will show you the path and the project file name. It should look as in the screenshot below.




Use the file `trans.gms` in GAMSIDE and use the GAMS tutorial, which will have been downloaded to the working directory (you may want to print off a copy), 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 GAMSIDE 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, put the command `rf=ref` in the command line box.



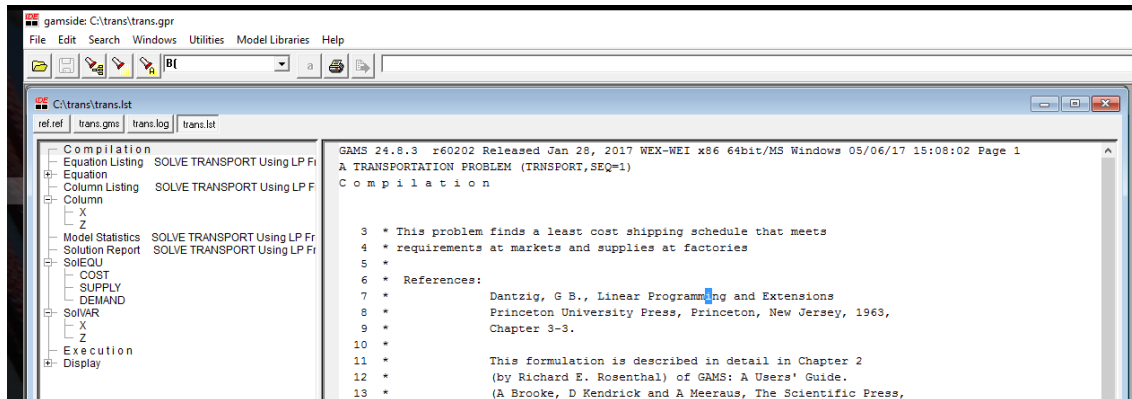
Run the transport model by selecting Run from the File menu, or F9 or the Run GAMS button  on the toolbar. The run command submits the programme file for compilation and, presuming it compiles without error, executes the programme. (If you wish solely to implement the compilation stage choose Compile from the File menu, or Shift F9.)

- The progress of a submitted programme is recorded in an active process window. Information recorded in that window is very useful and provides an easy way to debug a programme file (see below). The information from the process window is recorded as [Filename].log and saved in the project directory.
- The layout of the various windows is a matter of personal choice. One layout that is relatively easy to use is Tile Vertical from the Window menu; once you have set the tile layout it will remain fixed for that project.
- The GAMS output file is returned automatically as a tabbed file in the editor window as 'transport.lst'. (You can choose for this to not happen by changing the settings in the File > Options menu.)

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. 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. The index (\*.lxi) file provides a quick way to navigate the list file; try using the index.

## Practical CGE Modelling: Transport Problem Exercise

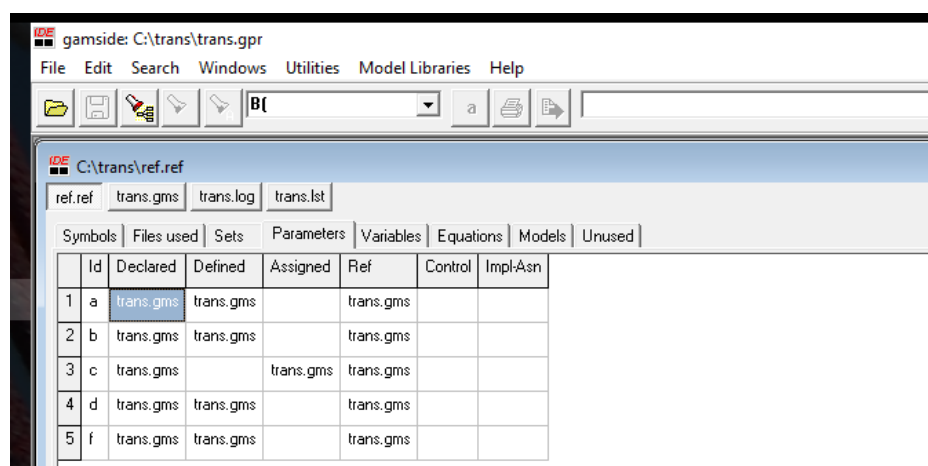


The model file for the transport problem is small and it is easy to navigate it by scrolling through the file in GAMSIDE. This method will some become inefficient so it is good idea to be able to explore a model's using some form of indexing; a reference file is the best way to do this for a GAMS programme. 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 (ref.ref), click on the Parameters tab and then double click on the cell in the row for *a* and column for Declared (see below).

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.



The screenshot shows the GAMSIDE software interface with the reference file (ref.ref) open. The Parameters tab is selected, displaying a table of parameters.

	Id	Declared	Defined	Assigned	Ref	Control	Impl-Asn
1	a	trans.gms	trans.gms		trans.gms		
2	b	trans.gms	trans.gms		trans.gms		
3	c	trans.gms		trans.gms	trans.gms		
4	d	trans.gms	trans.gms		trans.gms		
5	f	trans.gms	trans.gms		trans.gms		

### 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.

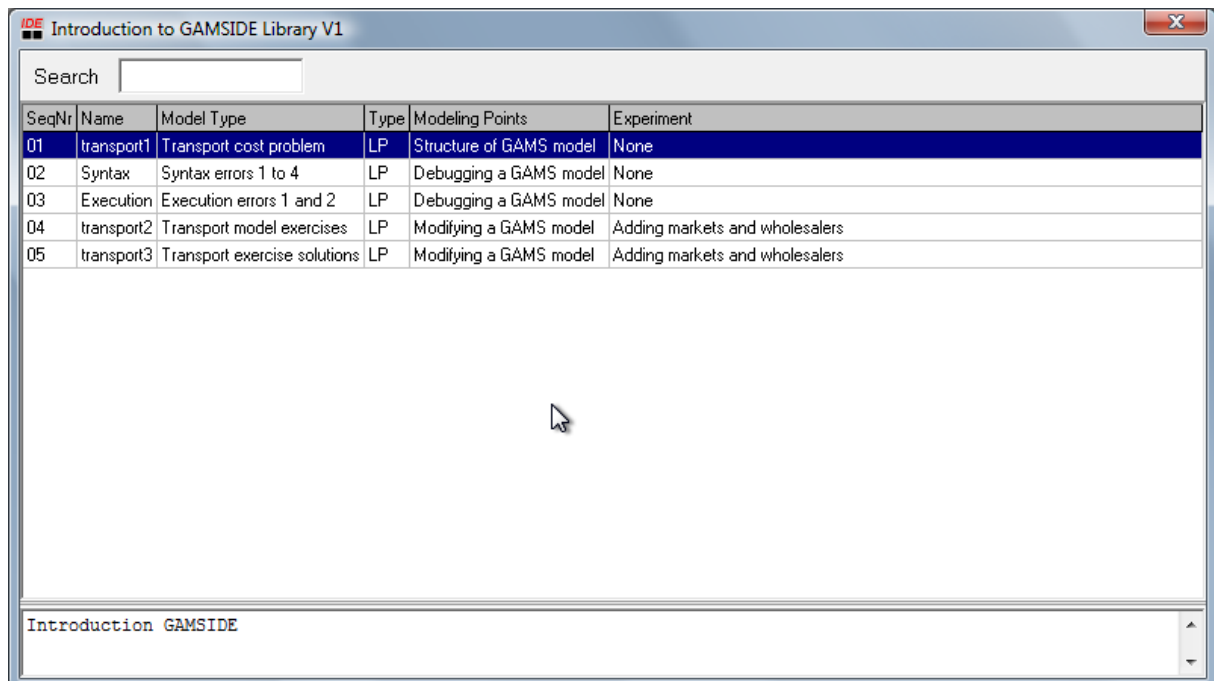
Indeed, the clichés about ‘learning from mistakes’ could have been devised by reference to computer programming.

Syntax and execution error exercises can be conducted using a single project directory and file.

#### Syntax Error Exercises

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

Create a directory called `error` and then download the syntax error exercises to that directory. The files for the syntax errors are SeqNr 2 with the name Syntax.





### *Practical CGE Modelling: Transport Problem Exercise*

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.


### Execution Error Exercises

There are **two** \*.gms files, labeled `trnserr#.gms` (where # is a numbers 5 and 6), that are perturbations of the basic transport linear programme.

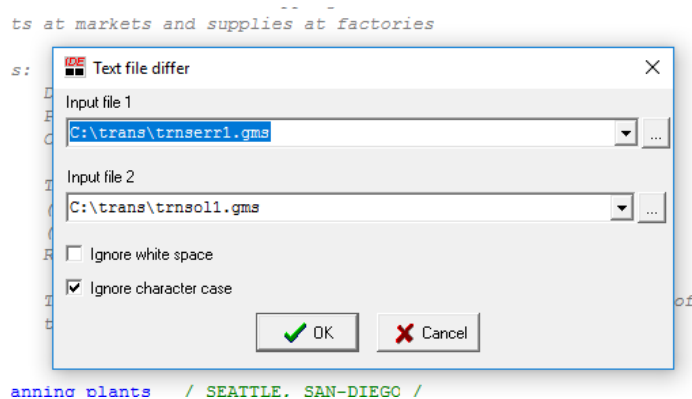
Create a directory called `error2` and then download the execution error exercises to that directory. The files for the execution errors are SeqNr 3 with the name Execution.

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.

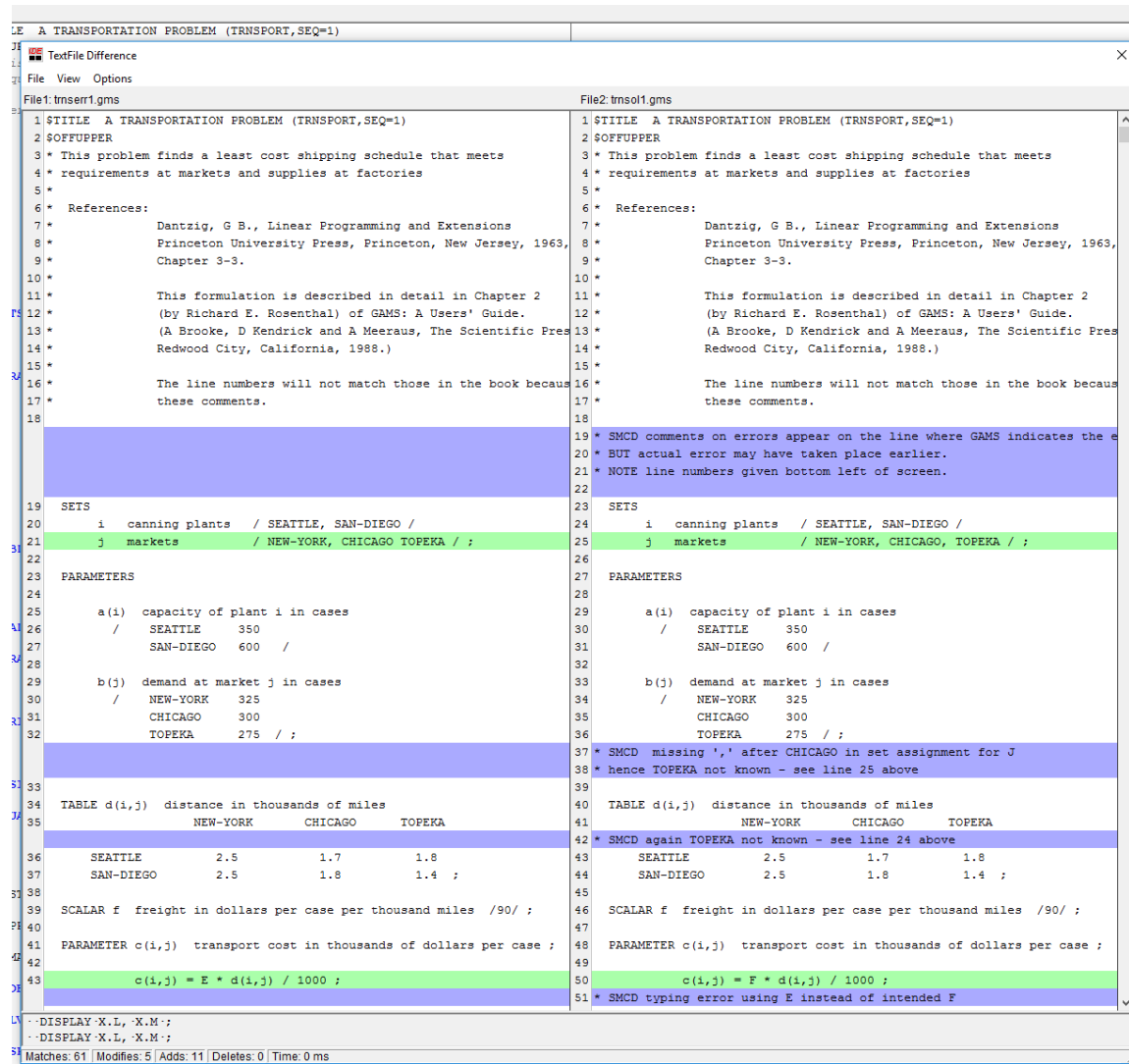
For each `trnserr#.gms` file there is a solution file, labeled `trnsol#.gms`. The solution file contains corrections for each of the errors and is annotated to provide explanations. Search for the string ‘SMcD’ in each solution file. You should modify the `trnserr#.gms` file and make sure you can correct the errors making as little use as possible of the solutions provided. You can also use a utility provided with GAMSIDE called ‘Diff TextFiles’, which is found on the Utilities menu. When ‘Diff TextFiles’ is selected from the Utilities menu a dialogue box opens that allows you to choose two file to compare: choose the files by clicking on the boxes with three dots () on the right-hand side of the dialogue box. When selected click OK.

## Practical CGE Modelling: Transport Problem Exercise



This will open a window in which Input file 1 is in the left-hand pane and Input file 2 is in the right-hand pane; typically, you will need to resize this window. The differences between the two files are highlighted. If the differences are comments and/or blank rows the rows are highlighted in purple. If there are differences in rows of code, then the rows are highlighted in green.

## Practical CGE Modelling: Transport Problem Exercise



```

1 $TITLE A TRANSPORTATION PROBLEM (TRANSPORT,SEQ=1)
2 $OFFUPPER
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
17 * these comments.
18
19 SETS
20 i canning plants / SEATTLE, SAN-DIEGO /
21 j markets / NEW-YORK, CHICAGO TOPEKA / ;
22
23 PARAMETERS
24
25 a(i) capacity of plant i in cases
26 / SEATTLE 350
27 SAN-DIEGO 600 /
28
29 b(j) demand at market j in cases
30 / NEW-YORK 325
31 CHICAGO 300
32 TOPEKA 275 / ;
33
34 TABLE d(i,j) distance in thousands of miles
35 NEW-YORK CHICAGO TOPEKA
36 SEATTLE 2.5 1.7 1.8
37 SAN-DIEGO 2.5 1.8 1.4 ;
38
39 SCALAR f freight in dollars per case per thousand miles /90/ ;
40
41 PARAMETER c(i,j) transport cost in thousands of dollars per case ;
42
43 c(i,j) = E * d(i,j) / 1000 ;
44
45 ..DISPLAY X.L, X.M ;
46 ..DISPLAY X.L, X.M ;
47
48 Matches: 61 | Modifies: 5 | Adds: 11 | Deletes: 0 | Time: 0 ms

```

## 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 (i) improve understanding of a GAMS programme, (ii) develop programme organisation skills, and (iii) develop the 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

Create a directory called `trans2b` 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 you should do so. **NB:** provided you renamed files before modifying them you will not lose your work.

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?

### A New Market

Create a directory called `trans3` and then download the transport model to that directory. The files for the transport model are SeqNr 4 with the name `Transport2`.

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

### *Practical CGE Modelling: Transport Problem Exercise*

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`.)

### Intermediate Markets

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`.)