A Basic 1*2*3 Open Economy Model: Exercises

1. Introduction

The closed economy models used (non-linear) Cobb-Douglas functions. Cobb-Douglas functions could be used for trade, but C-D functions are restrictive and are not used in CGE models for trade. These exercises introduce a more sophisticated functional form: Constant Elasticity of Substitution/Transformation functions, that are the dominant functional forms in CGE models. The 123 model is a simple but profound model; its simplicity is achieved by abstracting from having multiple agents – activities, households and factors – and policy instruments (taxes) to focus solely on the modeling of trade using hypothetical data. The next module (O4) restores some of the complexities – policy instruments and factors - and uses 'real world' data.

The exercises in this part of the course introduce you to the modelling of trade in CGE models and CES/CET functions. The basic 123 exercises focus on calibrating CES/CET functions and their properties: they demonstrate how the properties of these functions influence the results from CGE models. It is important to note that model results with CES/CET functions depend not only on the elasticities of substitution/transformation but also on the shares on the arguments in the function. Understanding these properties of the functions will ensure that you can avoid confusing the role of elasticities and shares.[1]

This set of exercises uses the minimalist 123 model developed by Sherman Robinson and various co-authors. The first stage of the exercise is to add in the components of the model code that are missing. The second stage is to carry out two policy experiments that demonstrate how the model responds to (i) changes in elasticities, and (ii) changes in the (exogenous) balance of trade. The appendix contains an additional exercise that demonstrates how the relationships between domestic and world prices depend on the elasticities and trade shares; this provides a means of generating an empirical demonstration that models using the Armington insight produce offer curves that are consistent with standard economic trade theory. This was an important theoretical contribution of the 123 model.

These exercises are influenced by exercises that were originally developed by Sherman Robinson in the early 1990s.

[1] It is common to find modellers increasing elasticities to counter the effects of small shares. This 'habit' can be found even in CGE models that are used extensively for national and international policy advice.

2. Model Coding

READ THROUGH THIS SECTION BEFORE DOING ANYTHING MORE.

Setup

All the files you will need for this module are in the Practical CGE Library you created in module O1.

The first step is to get the correct files into a working directory. So, do the following

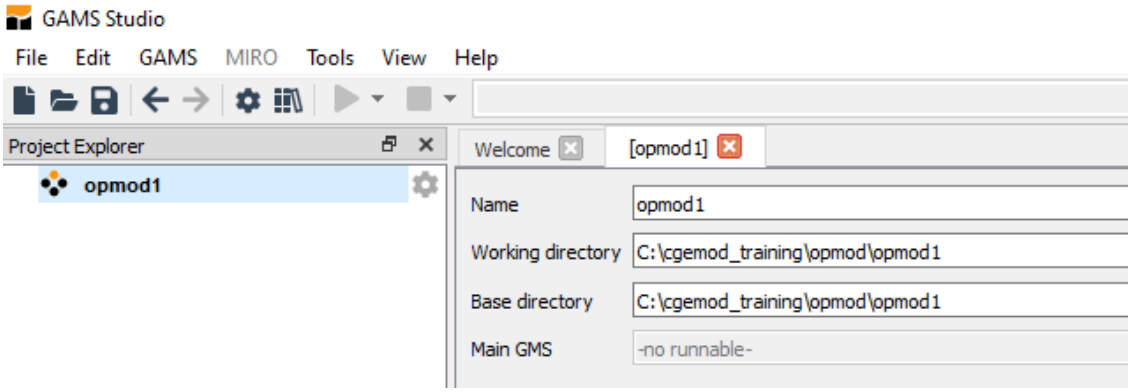
1. Create a separate project directory for the 123 models, (`C:\cgemod_training\opmod`).
2. Open GAMS Studio select `File>New Project` and add a new (sub) directory (`opmod1`) in the directory to give `C:\cgemod_training\opmod\opmod1`.
3. The Studio panels should appear automatically, and the Project Explorer and Editor panels should appear; note how the working directories path is reported as `C:\cgemod_training\opmod\opmod1`. See Figure 3.1.
4. In Studio press `F6` and in the `Model Library Explorer` select the `Practical CGE Library` and then select the library file `opmod1` and choose `Load` (or double click of the name), which is SeqNr: 5.
5. The `123.gms` model will now be displayed in the editor window and be listed in the Project Explorer as being in the project `opmod1`.
6. Save this file as `123_**.gms`; where ** are wild cards, e.g., your initials.
7. You should note the files that have been downloaded to the directory `C:\cgemod_training\opmod\opmod1`. HINT: use `Open Location`.
8. Print a copy of the model and note each of the subsections, including those with nothing in them – there are pre-defined spaces for all the code you will generate.
9. Studio can now be used to work with the open economy model.

This is the first step. Do not move on until you have successfully completed this sequence.

The template (`123.gms`) is complete except for two components, the equation definitions/assignments relating to trade and the parameter calibrations for trade.

Start with the equation assignments, Section 14 in the template. Each of the equation declarations, Section 13, matches an equation in the technical document for this model ('A Basic 1*2*3 Open Economy Model') that is on the Moodle page (P1 – P5, X1 – X4, Y1, C1, E1 – E3).

Figure 3.1 New Project: opmod1



NB: Make a copy of the *.gms file from which you begin this exercise.

The process of revising/updating/modifying should proceed in simple and systematic stages. Complete each stage before moving onto the next. The procedure set out below presumes that all the equations will be added and then the required parameter calibrations will be carried out; it is one way of being systematic.

i) Start with the first equation that requires adding (say `PMDEF`).

ii) Add the equation definition in section 14 of the `*.gms` file by converting the algebraic expression (P1) into GAMS code.

iii) Carry out the same process for each of the declared equations that has not been assigned using the appropriate algebraic expression in the supporting document.

iv) Add the model closure conditions (M1, M2 and M3) to Section 15 of the template.

v) Now calibrate the shift and share parameters for the Armington CES functions. which are assigned in Section 9, sub section 'calibration of shift and share parameters for trade', sub-sub-section 'for imports-domestic composite'.

a) Start with the elasticity parameter *rhoc* . In the data entry section, 8, the elasticity of substitution for the CES function, *sigma*, is assigned, but *rhoc* is a derived parameter. This is done in Section 9, sub-section 'elasticity related parameters', where *rhoc* is assigned as

$$rhoc = \left(\frac{1}{sigma}\right) - 1$$

b) Then move onto the share parameter, *delta*, which is defined as

$$\delta = \frac{P0_M * M0^{(\rho+1)}}{P0_M * M0^{(\rho+1)} + P0_D * D0^{(\rho+1)}}$$

noting that calibrating *delta* requires data on the base prices and quantities and *rhoc*, the elasticity parameter.[2]

c) Then the shift parameter, *ac*, can be defined as

$$ac = \frac{QQ0}{\left(\delta * QM0^{-rhoc} \left(1-\delta\right) * QDD0^{-rhoc}\right)^{\left(-1/rhoc\right)}}.$$

vi) Now calibrate the shift and share parameters for the CET function. The parameters for the CET function, which are assigned in Section 9, sub section 'calibration of shift and share parameters for trade', sub-sub-section 'for exports-domestic composite', is like that for the CES function.

a) The elasticity parameter, *rhot*, is assigned first. In the data entry section, 8, the elasticity of substitution for the CET function, *omega*, was assigned, but *rhot* is a derived parameter: *rhot* is assigned as

$$rhot = \left(\frac{1}{omega}\right) + 1$$

b) The share parameter, *gamma*, is defined as

$$gamma = \frac{1}{\left(1 + \frac{PDS0}{PE0} * \left(\frac{QE0}{QDS0}\right)^{(rhot-1)}\right)}.$$

noting that defining *gamma* requires that *rhot* is defined before *gamma*.

c) Then the shift parameter, *at*, can be defined as

$$at = \frac{QX0}{\left(\gamma * QE0^{rhot} \left(1-\gamma\right) * QDS0^{rhot}\right)^{\left(1/rhot\right)}}.$$

---

[2] In some models the calibration of delta is done in two stages. First defining a parameter *predelta*, $predelta = \left(PM0/PDD0\right).\left(QM0/QDD0\right)^{(1+rhoc)}$ is defined, and then *delta* is defined as $\delta = predelta/\left(1+predelta\right)$.

vii) Check that all the other required parameters, variables and display statements have been included in the template.

viii) Run the model **without** any policy experiments (`F10`) and resolve any compilation and execution errors.

ix) When the model solves correctly move on to the policy experiment.

Using the Worked Solution

In the directory `C:\cgemod_training\opmod\opmod1` there will be a file `123_sol.gms`; this is our sample solution. One way to use this is in conjunction with the sequential `$STOP` commands.

AFTER SPENDING TIME TRYING, DILIGENTLY, TO SOLVE THE PROBLEMS AT EACH STAGE.

Use a programme for comparing files (our preference is WinMerge). Work out why you got it wrong and then type the changes into your code – **do not copy and paste the code from our sample code**. You could also use `View/Pin right` or `View/Pin below` in Studio, although this option is less efficient in this instance.

Now move on to the next stage of the checking process. Again, spend time trying, diligently, to solve the problems, before resorting to our sample code.

Hints and Suggestions

Some hints and suggestions may prove helpful.

1. Do one step at a time
2. Assigning values to the parameters is one of the trickier tasks – look at how values have been assigned to other parameters and adapt the method to the present needs.
    a. When calibrating elasticity, shift and share parameters for CES and CET it is VERY easy to get parenthesis in the wrong place. GAMSIDE has a useful, and underused, facility for matching parenthesis. To find the parenthesis that is paired/matched with an opening parenthesis: click immediately AFTER an opening parenthesis and then press `Ctrl+B`. The cursory will move to AFTER the matched parenthesis.

3. Remember that GAMS can only use a parameter if it has already been declared and assigned, and that you can only assign something after it has been declared.
4. Use `$ontext` and `$offtext` to shut off part of the code to help with any debugging, e.g., you may want to shut off some parts of the code. This can be used in conjunction with the `$STOP` command.
5. If you have used F10 (GAMS>Run with GDX Creation) the file 123_**.gdx will have been generated, and can be used to check calibration etc.
6. If the GAMS Default Configuration has been set to automatically produce a reference file, the file `r23_**.ref` will have been generated and can be used to help checking for errors.
7. Mistakes are normal. Learning to use the debugging facilities in GAMS will benefit you in the short, medium and long run. 'Experts' may make less mistakes, but one defining characteristic of 'experts' is their ability to debug their code.

Testing Model 2 with a Simple First Experiment

One way to test a model is to run a simple experiment and then check the results make sense.

In the policy experiments section (18) add the lines

```
pwe = pwe * 0.9 ;


Solve mod_123 Using NLP MAXIMIZING QCD;
```

The solution value for *QCD* (the optimand) should be 98.009 and *WALRAS* should be ZERO.

3. Policy Experiment Programming

One method of running policy experiments is to write the policy experiment module as an include file (`*.inc`), and then to call the include file within the main `*.gms` file. Thus, rather than having a separate `*.gms` file for each set of policy experiments, there is a single `*.gms` file for each model and a separate `*.inc` file for each policy experiment. A template for a first policy experiment file will be in your working directory: `123_expt_base.inc`. Before any policy experiments can be implemented it is necessary to add some code to the template.[3] The stages for this are simple.

1. Create a copy of the template with a different file name, e.g., `123_expt_pw1.inc`.
2. Section 18a of the template declares and assigns the set, called `sim`, which is used to control the running of the policy experiments. The example in the template is for a change in the world price of imports. This currently has 5 members, one of which replicates the base solution as an aid to subsequent interpretation of the results. Section 18a also declares a parameter to carry the changes that are the subject of the experiments, and a series of parameters that will be used to store the results of the policy experiments. Note how a results parameter has been created for each variable; it may be tempting only to create results parameters for variables that you presume will be of interest, but it is often more efficient in the long run to create parameters for each variable; outputting all the results to GDX and accessing the results in GDX files is so fast a little time setting things up at the beginning pays off. Also, notice how a naming convention has been adopted that relates these parameters to the variables and parameters in the model; this makes reading the files easier.
3. The specification of the policy experiments takes place in Section 18b. This involves the assignment of values to the parameter that carries the changes being simulated, i.e., `pwmSIM(sim)`. A value must be assigned for each member of `sim`. This can be done either in absolute terms, i.e., as a number, or relative terms, i.e., as a proportion of the original value of the parameter. Since it is often common to

[3] It may prove helpful to refer back to the policy experiment sections of previous models.

express the results of policy experiments as percentage changes the latter approach might be preferred.

4. The running of the policy experiments in a loop is initiated in Section 18c. In this section the loop is started, the model **parameter** is set equal to the value for the experiment and the model is resolved. But the results are not stored.
5. It is therefore necessary to assign values to the result parameters; this takes place in Section 18d. After each solve statement, it is necessary to add a series of commands that write the values of the variables to the results parameters. For now, write commands that set the results parameters equal to the LEVELS of the variable, e.g., `resPE(SIM) = PE.L`.
6. After assigning the levels results to the appropriate results parameters, add an `Execute_unload` statement that downloads all the results parameters to a designated gdx file, e.g., `Execute_Unload 'res_expt_pwm.gdx', resPDD, …., resWALRAS` ; (Include all the results parameters.)
7. All the results will be reported in the file `123_**.gdx`, but they will be mixed in with the information about the model. This method has 2 main advantages, (i) it keeps the results of the experiments separate from the information about the model, and (ii) it can keep the results from each set of experiments separate (by using different names for the gdx files).
8. In the `*.gms` file add the following statement to the Policy Experiment section
   `$INCLUDE **.inc`
   where `**.inc` is the name of your policy experiment file.
9. Run the model (`F10`) and policy experiments from the `*.gms` file.

Policy Experiments

The model is set up to conduct two types of policy experiment

1. changes in world prices (*pwm* and *pwe*), which can be modeled using sequences of different world prices;
2. changes in the exogenously determined level of unrequited flows on the capital account of the balance of payments (*KAPWOR*), these can be negative or positive.

A couple of simple experiments involving changes in the world prices of imports and exports will be introduced using this model.

1. Change the world prices of **imports** from 0.8 to 1.2 in steps of 0.1 and download all the results to a GDX file. Save this include file with a descriptive name.
2. Change the world prices of **exports** from 0.8 to 1.2 in steps of 0.1 and download all the results to another GDX file. Save this include file with a descriptive name.

Make sure you can inspect the results in the two GDX files. Do the results make sense? Take some time to evaluate the results.

An example of the include file for the import price changes is included in the directory opmod1: `123_expt_pwm_sol.inc`.

Use the results from the simulations to answer the following questions using tables and graphs:

1. How do the consumer prices change with changes in import prices?
2. How do the consumer prices change with changes in export prices?
3. Are the price changes induced by changes in import prices linear or non-linear?
4. Are the price changes induced by changes in export prices linear or non-linear?
5. How, and why, does household income change with changes in import and export prices?
6. How, and why, does the volume of production change with changes in import and export prices?
7. How, and why, does the value of production change with changes in import and export prices?

4. Armington Insight and Trade Elasticities

The sensitivity of the results from CGE models to changes in the elasticities is an important issue. These experiments address this issue by changing two things in the simulations: the world prices and the trade elasticities. It will be useful if the model is coded so that both changes can be achieved in the same experiment file and the results can be collated. This is achieved by running the experiments in a system of two LOOPS, with one LOOP run within another. This process runs the outer LOOP using the first element of the set controlling the outer LOOP, and then runs the inner LOOP run for each element of its controlling set. When all elements of the inner LOOP's set have been run, the next element of the outer LOOP's controlling set is selected, and the inner LOOP runs again for each element of its controlling set. This is carried on until each element of the outer LOOP has been run with each element of the inner LOOP.

Start from the file `123_expt_pwm_sol.inc` that is in the directory `opmod1`. This is our example of a simple experiment file that changes the world prices of imports. Open this file and save it with a new name, e.g., `123_expt_elast_**.inc`. Now we must make some changes.

1. Declare the set `msens` – import elasticity sensitivity - as part of the set declaration and assignment statement in section 19A. Assign three members to this set, `msens1, msens2 and msens3`, describe these as elasticity 1 to 3.
2. Declare the parameter `sigmaMSENS(msens)` - elasticity of substitution between imports and domestic – after the parameter keyword in section 18A. This parameter will be used to provide the elasticity data for the experiment.
3. Change how the parameters for storing the results are declared by adding the set `msens` to each parameter, e.g., `resPDD(sim,msens)`. (Search for and replace "`sim)`" by "`sim,msens)`" for a selected block of text.
4. In section 18B add assignment statements for `sigmaMENS`. Let us assume the experiment will examine 50% increases and decreases in the values of sigma

```
sigmaMSENS("msens1") = sigma * 0.5 ;
sigmaMSENS("msens2") = sigma * 1.0 ;
sigmaMSENS("msens3") = sigma * 1.5;
```

5. In the section 18C add a `LOOP` keyword, indexed on `msens` before the `LOOP` keyword indexed on `sim`.
6. **Now for the tricky part**. The calibrated values of the parameters for CES and CET functions depend on the values of the elasticities. Hence, it is necessary to recalibrate the elasticity, share and shift parameters for the ARMINGTON equation. The following code for the elasticity parameter should be the first code after the `LOOP` keyword that is controlled by `msens`.

```
rhoc        = (1/sigmaMSENS(msens)) - 1 ;
```

7. It is then necessary to recalibrate/reassign the CES function parameters; remember that the shift parameter depends on the elasticity parameter and share parameter depends on the shift and elasticity parameters. Get the necessary code for the model's GMS file and copy it below the assignment statement for `rhoc`.
8. The assignment statement for the results parameters (just after the SOLVE statement) need revising so that they match the indexing in the declared parameters. Add the set `msens` to each parameter.
9. Each `LOOP` statement needs to be closed by "`) ;`". Therefore, an additional `LOOP` closure instruction needs to be added after the instruction of the inner `LOOP`.
10. Finally, make sure the results are written out to an appropriately named GDX file.
11. Add an INCLUDE statement at the end of the model file, e.g., `123_sol.gms` (remember to switch off any other INCLUDE statements)
12. Run the model (`F10`).
13. Check the shocks were as intended (see the display statements for `pwmSIM` and `sigmaSENS`).
14. Check the results make sense.
15. Analyse the results by comparing changes in prices and quantities for different shocks and elasticities.

A worked example (`123_expt_elast_sol.inc`) is in the course library.

1. In GAMS Studio select `GAMS> Model Library Explorer (F6)`. Select the `Practical CGE Library` and then select the library file `opmod1_sol` and choose `Load` (or double click of the name), which is SeqNr: 6.
2. When notified that a file already exists choose to overwrite the file.

3. The `123_sol.gms` will be displayed in the editor window and be listed in the Project Explorer as being in the project `opmod1`.
4. You should note the files that have been downloaded to the directory `C:\cgemod_training\opmod\opmod1`. HINT: use `Open Location`. These will include example solutions for the exercises of this module.

5. Offer Curves

Offer curves are a standard general equilibrium result derived from orthodox trade theory: the 'academic' acceptability of CGE analyses using Armington's insight requires that CGE models will generate appropriately shaped offer curves. This exercise uses the 123 model to demonstrate that CGE models using Armington's insight do generate the appropriate offer curves; this was a fundamental component/objective of de Melo and Robinson's 1989 paper. (If you are uncertain about the concept of Offer curves a short explanation is included in the file '`Offer Curves and CGE Models.pdf`' available among the downloads for Module O3).

The derivation of offer curves for a 'small' country requires simulating the trade pattern, i.e., export and import volumes, for different offer curves for the Rest of the World. The offer curves for the Rest of the World are defined by a series of rays from the origin in a diagram drawn in *QE* (export) and *QM* (import) space: each ray defines a different set of terms of trade, i.e., the rates of exchange of exports for imports. These terms of trade are set exogenously in the 'small' country case, i.e., the rays are linear with a constant slope.

The process for this demonstration proceeds in two stages; first, the 123 model's data will be adjusted/changed and second, an experiment will be run in which the prices of exports are varied for fixed import prices, i.e., the terms of trade are varied. The exercise starts from the '`123_sol.gms`' and the experiment file '`123_expt_pwm_sol.inc`' that are both in the course library.

1. Open the file `123_sol.gms` and save it with the name `123_offer.gms`.
2. In the data entry section (c line 459) change the values of both imports and exports from 20 to 40 and the domestic supply from 80 to 60. Then below the Table declaration and assignment for the SAM (after the semi colon) add the following line of code '`SAM(sac,sacp) = SAM(sac,sacp) * 0.1 ;`'. This scales the data by dividing through by 10 (it will make it easier to analyses the results without changing the interpretation).
3. Solve the model and check that the variable `WALRAS` equals zero and there are no infeasibilities (search for '`infes`'). ONLY proceed to the next step when you are certain the model is solving correctly.

4. Open the file `123_expt_pwm_sol.inc` and save it with the name `123_expt_offer.inc`.
5. Change the elements in the set `sim` to `sim1` to `sim7` (the labels are not important the choice is for simplicity but having 7 simulations is helpful).
6. Declare the parameter '`pweSIM(sim)  World price of exports`'.
7. Assign the following values to the parameter `pweSIM`, starting with `pweSIM("sim1") = pwe * 0.40`, and increase in steps of 0.2 until `pweSIM("sim7") = pwe * 1.60`.
8. Add the shock statement within the inner `LOOP`

```
pwe        = pwsSIM(sim) ;
```

9. Make sure you delete or comment out any other shock statements.
10. Finally, make sure the results are written out to an appropriately named GDX file.
11. Add an INCLUDE statement at the end of the model file to include the new experiment file you have created (remember to switch off any other INCLUDE statements).
12. Run the model (`F10`).
13. Check the shocks were as intended.
14. Check the results make sense.

Now interpret the results.

1. Make notes of your *a priori* expectations of what will happens to the variables *QM*, *QE*, *QD* and *QM*, and the prices *PX*, *PM*, *PE*, *PD* and *PQ*.
1. Plot out how the quantities of exports (*QE*) and imports (*QM*) change for each value of the parameter `pwe`. There is an Excel workbook – `offer.xlsx` - in the working directory that is set up to make the process straightforward.
2. Are the results consistent with the theory of offer curves for a small country?
3. What happens if the economy is **more** open, e.g., change the values of both imports and exports from 20 to 60 and the domestic supply from 80 to 40?
4. What happens if the economy is **less** open, e.g., change the values of both imports and exports are 20 and the domestic supply is 80?

A worked example of the model is in the file `123_offer_sol.gms`, an example experiment file is `123_expt_offer_sol.inc` and an example for the Excel workbook is `offer_sol.inc`; will have been downloaded from the course library.

Trade Shares and Elasticities

The experiments with different elasticities and degrees of openness (trade shares) of economies demonstrate a very import result. Namely, that the sensitivity of the results to different elasticities depends on **BOTH** the trade shares **and** the elasticities, i.e., there are interactions between the elasticities and shares.

Typically, the functions used in economics operate 'well' when shares are not 'small', although the definitions of 'small' and 'well' are imprecise. When there are 'small' shares in the data the degree of sensitivity to changes in elasticities are muted. One approach to the 'small shares problem' is to increase the associated elasticity, which reduces the curvature of the function and thereby reduces the terms of trade effects. It is arguable that the GTAP databases and model use this method to damp down the terms of trade effects with respect to imports. We are not convinced that this is an appropriate approach: in our more advanced models, i.e., those beyond the scope of this course, we adapt the behaviours in the model rather than shifting the elasticities.

6. Dutch Disease

The implications of 'Dutch Disease' have been subject to extensive scrutiny using CGE models. The issue of concern is how an economy responds to an unrequited inflow of funds; these could be in the form of a resource boom, e.g., the Nijmegen gas fields in Holland, diamonds in Botswana, oil in Cameroon. The nature of the effects can be easily illustrated using the basic 123 model.

Analyses of the implications of Dutch Disease can be achieved by examining the impacts on the 123 model's economy of an increase in the exogenous balance of trade (*KAPWOR*). These experiments require that the balance of trade[4] is **fixed** exogenously and that the exogenously determined balance of trade is changed.

Switch off the policy experiment include file for the world price experiments. Run the model to ensure there are no errors – remember to check `WALRAS` and do a numéraire check.

Now use the model to empirically determine the following:

i) The effect of changes in the exogenously determined balance of trade on prices and quantities.

ii) How do the effects vary for different elasticities of substitution?

These experiments can be done by simply modifying the experiment file used for the world price experiments. We will assume, for exposition purposes, that you are starting from our worked example of the world price experiment file used for sensitivity analyses (`123_expt_elast_sol.inc`). Change the file name (for example to `123_expt_dutch_**.inc`) where ** are your initials.

1. Declare the parameter `KAPWORSIM(sim)` – external balance for simulations – after the parameter keyword in section 18A. This parameter will be used to provide the trade balance shocks for the experiment.
2. In section 18b – Defining the Policy experiments – assign values to `KAPWORSIM` for each member of the set sim. Using

```
KAPWORSIM("base")   = KAPWOR0 + 0.0 ;
KAPWORSIM("sim2")   = KAPWOR0 + 5.0 ;
```

[4] In this model and data system the trade balance and current account balance are synonymous. This is not the case in reality.

```
KAPWORSIM("sim3")   = KAPWOR0 + 10.0 ;
KAPWORSIM("sim4")   = KAPWOR0 + 15 ;
KAPWORSIM("sim5")   = KAPWOR0 + 20 ;
```

3. Add the shock statement within the inner `LOOP`

```
KAPWOR.FX       = KAPWORSIM(SIM) ;
```

4. Make sure you delete or comment out the shock statement for the changes in world prices.
5. Finally, make sure the results are written out to an appropriately named GDX file.
6. Add an INCLUDE statement at the end of the model file, e.g., `123_sol.gms` (remember to switch off any other INCLUDE statements).
7. Run the model (`F10`).
8. Check the shocks were as intended (see the display statements for `KAPWORSIM` and `sigmaSENS`).
9. Check the results make sense.

Now it is time to interpret the results:

1. Make notes of your *a priori* expectations of what will happens to the variables *QM*, *QE*, *QD* and *QM*, and the prices *PX*, *PM*, *PE*, *PD* and *PQ*.
2. Plot out how the quantity and price variables change as the inflow of funds increase for the base case elasticities, i.e., the results for '`msens2`'.
3. Explain the changes in *QCD*?
4. What happens to the variables *QE* and *QD*? Are the changes in these variables consistent with the changes in the prices of *PE* and *PD*?
5. What happens to the exchange rate? Does it appreciate or depreciate? Is the appreciation or depreciation consistent with your expectations? (NOTE: the exchange rate is defined in the American way, not the European way).
6. What has happened to the real exchange rate?

A worked example of this experiment file `123_expt_dutch_sol.inc` will have been downloaded from the course library as part of '`opmod1_sol`'.

7. Three-level LOOPs

It is straightforward to add multiple LOOPs in an experiment file; however, discretion is important. Too many LOOPs and it can become difficult to work out which results matter and which do not. So, the aim is to get enough to help your analyses but not too many as to confuse yourself.

An example of a three-level system of LOOPs for the world price experiments is in the experiment file `123_expt_elast_sol3.inc`, which will have been downloaded from the course library as part of '`opmod1_sol`'. This example considers the implications for the results arising from differences in BOTH the import (Armington) and export (CET) elasticities.

Examine this file carefully so that you understand how the third LOOP is introduced and how it is coded.