# A Simple CGE Model (*smod_t*): Exercises

2025

## Table of Contents

# 1. Introduction

The single country model (`smod_t.gms`) extends the 123 model to include sectoral detail. This model is a simple single country model that can be calibrated with databases for many regions, with variable numbers of commodities, activities, factors, and households. There are taxes on imports, exports, sales, production, and a direct tax on household income; the available tax instruments can be changed by changing the model code.

The model is relatively simple, but it encompasses the main features of all CGE models. While the model is simple, the organisation of the model code and the methods used to access data and report results is flexible and represents one of the modern ways in which a GAMS based CGE model can be organised.

The information about the structure of the economy is important when explaining simulation results; since each country has a different structure, seemingly similar experiments will produce different results. For background, you will construct a structure table defining characteristics of production and consumption in the economy. In the exercises, you will consider policy shocks that affect all sectors and policy shocks applied to a single sector. You will consider trade and productivity shocks, but the model also allows for international transfers, which you may exploit for subsequent projects.

The exercises emphasize several skills:

1. loading different databases into a CGE model and checking the model is correctly formulated.
2. using a Social Accounting Matrix (SAM) to describe the structure of a country's economy.
3. using loops for policy simulations.
4. using the GDX MERGE (and GDX Diff) utility to compare result files, and
5. changing model code to swap variables and parameters while not changing the variable and equation count.

For Module P5 the SAMs have four commodities and activities: agriculture, natural resources, manufacturing, and services; four factors: land, unskilled labor, skilled labor, and capital; and two households: urban and rural.

## 2. Set up for a single country CGE model (smod_t)

Setup

All the files you will need for this module are in the Practical CGE Library you created in Module P1.

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

1. create the directory for these exercises, i.e., `C:\cgemod\pract\smod\smodt1`,
2. create a New Project with the name `smodt1`,
3. set the Working directory to `C:\cgemod\pract\smod\smodt1`,
4. choose `File>Save`,
5. choose `File>New` in the working directory and `SAVE; new**.gms` will appear in the `smod` project.
6. In Studio press `F6` and in the Model Library Explorer select the `Practical CGE Library` and then select the library file `smod_t` and choose `Load` (or double click of the name), which is SeqNr: 8.
7. The `smod_t.gms` model will now be displayed in the editor window and be listed in the project `smodt1`.
8. You should note the files that have been downloaded to the directory `C:\cgemod\pract\smod\smod_t1`
9. Review the model code and note the contents of each of the subsections; note that there are nearly 2,000 lines of code so COMPILING the model (`Shift+F10`) to generate a reference file might help.
10. Save this file as `smod_t_**.gms`; where ** are wild cards, e.g., your initials.
11. Studio can now be used to work with the open economy model `smod_t_**gms`.

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

**RESIST THE TEMPTATION TO EXPLORE THE CONTENTS OF THE LIBRARY; THIS WAY LIES CONFUSION.**

**NOTHING IS BEING HIDDEN: YOU GET TO SEE AND USE ALL THE FILES.**

Loading data from Excel

1. The data for `smod_t` are loaded via an include file, `smod_t_load.inc` (line 265). Review the content of the file `smod_t.gms` prior to the instruction to include the file `smod_t_load.inc`. Address the following questions
    a. What is the role of the command `$ONEMPTY`? (HINT: check the GAMS documentation.)
    b. What sets are declared?
    c. What parameters are declared?
2. Review the content of the file `smod_t_load.inc`.
    a. How many data files are read in using this file?
    b. What is the meaning of the command `$CALL "GDXXRW ….”`?
    c. What is the meaning of `$GDXIN`?
3. The model is designed so that there is only one data entry point; all sets, parameters, etc., are assigned in a single location/file within the programme.
4. Review `smod_t_mod05.xls,` which contains the data, set information, and parameter values used in `smod_t.gms`
    - ‘Layout’ sheet provides information that is used by `GDXXRW.exe` to convert the data in the Excel workbook into the GDX format ready for reading by GAMS. Note that the sets and parameters in the ‘layout’ sheet have already been defined in the GAMS code.
    - ‘Sets’ sheet provides information on the sectors and factors used, as well as any subsets defined.
    - ‘elastc’ provides information on the elasticities of substitution used in the CES functions (sigma) and in the CET functions (omega)
    - ‘elasty’ provides information on the income elasticities of demand used in the LES functions
    - ‘elastmu’ provides information on the Frisch parameters used in the LES functions.
5. The data are read from the Excel file, `smod_t_mod05.xls,` in the following line of GAMS code:

```
$CALL "GDXXRW i=smod_t_mod05.xlsx o=data_in.gdx INDEX=LAYOUT!A4 trace=3"
```

After being converted to a GDX file, the data are stored in the file `data_in.gdx`; check out the contents of the file `data_in.gdx`. What does the instruction '`trace=3`' do? (HINT: check the GAMS documentation; you may want to explore other values for `trace`.)

The sets and data are loaded from GDX into GAMS:

```
$GDXIN data_in.gdx
$LOADdc sac c a f h g i w r
$LOADdc elastc elasty elastmu
```

'`dc`' instructs GAMS to domain check when loading the data. Domain checking is a very powerful tool that verifies that sets are allocated correctly, e.g., if a parameter is defined over specified sets GAMS checks to ensure all the set labels used to assign the parameter are members of the specified sets.

6. The SAM is read in from the file `SAMR_mod05.gdx`.
7. Review the file `SAMR_mod05.gdx` (HINT: use the GDX viewer facility in GAMS Studio).
    a. How many SAMs are in this GDX file?
    b. What are the accounts used in each SAM?
8. The specific region used in the model is Africa. To see where the region SAM is assigned in the code, go to line 26, `* # Selecting SAM for model` where `SAM(sac,sacp)` is set to equal `SAMR(sac,sacp,"afr")`.
9. Run the model with GDX creation (`F10`). (Make sure that the GAMS Configuration is set to automatically produce a reference file (`smod_**.ref`) and stop after a limited number of errors (`CErr`).)
10. Check out the contents of the files `data_in.gdx` and the file `smod_t_pm.gdx`.

<u>Checking the Model</u>

There are SEVEN checks **all** of which need to be passed. These checks, in the order they should be done are:

1. Check the data in the model are the intended data.
   This can be done by comparing values in the `SAM` reported in the file `data.gdx` with the Africa region of the parameter SAMR in the in file `SAMR_mod05.gdx`. You

only need to check a few values. The values may differ by a factor of 10,000. The data base is scaled, see the parameter `scaling = 10,000`

2. Check that the value for `VAR WALRAS` is zero.

   This check can be done from the `lst` file or `smod_t.gdx`. In the list file search for '`Var Walras`' or use the index for `SolVAR`. In `smod_t.gdx` search for '`Walras`'. If `WALRAS` does not equal zero, then the model fails Walras's law, and some transactions are not accounted for.

3. Check that the basic prices (`PE, PD, PM`) are equal to one.

   This check can be done from the `lst` file or `smod_t.gdx`. In the list file search for '`Var PE`' or use the index for `SolVAR`. In `smod_t.gdx` search for '`PE`'. Do this for each basic price. If values are not one, then the standard price normalisation rule has not been applied correctly.

4. Check that all entries in `ASAM1CHK` are equal to zero

   This checks for differences between the macro-SAM from the database for the model and the macro-SAM calculated from the values of the calibrated parameters and variables. They should be identical and therefore all entries in `ASAM1CHK` equal to zero.

5. Check that all entries in `ASAM2CHK` are equal to one

   This checks the ratio of the entries in the macro-SAM calculated from the values of the calibrated parameters and the macro-SAM calculated from the values of the parameters and variables in the model. They should be identical and therefore all entries in `ASAM2CHK` equal to one.

6. Check the `LHS` values. Search from the beginning of the file `smod_t.lst` for the first string `LHS` in the file; then search from the cursor for the string '***'

   In the output, GAMS reports the equations after all equations are organised so that all endogenous terms are on the `RHS` and hence the `LHS` should equal zero **or a known exogenous value**. Those that do not satisfy the condition are marked with '***'. In the `lst` file GAMS reports the values of the `LHS`: the search for `LHS` finds the first such entry, and second search finds the equations with incorrect `LHS` values.

7. **Numéraire check**: if the numéraire is doubled then all prices should double and quantities remain the same, hence all values should double. **Only do this test after the others have passed**.

Double the numeraire by turning on the numeraire check in the code. Search for '`* For simple numeraire use CPI0 * 2`', delete the '`*`' before '`CPI.FX = CPI0 * 2 ;`' and re run the model. (If the Exchange Rate (`ER`) is fixed, which you not been told to do!!!, it is also necessary to double the exchange rate). Check the values in `ASAM2CHK`; all should be identical and all entries equal to two.

8. Reset the numeraire statement to '`CPI.FX = CPI0 ;`' to do this, restore the '`*`' before the line '`CPI.FX = CPI0 * 2 ;`'

Only after these checks have been passed should you move on to using the model with the new database. Otherwise, you risk spending many hours analysing results that are WRONG/MEANINGLESS.

# 3. Trade Tax Reforms (Run in a Loop)

In this exercise, we will run a series of trade reform simulations in a loop. We will investigate the effects of tariff reductions in all sectors verses tariff reduction in a single sector.

Set up

1. You will continue to work in the subdirectory `C:\cgemod\pract\smod\smodt1`. To access the files needed, in GAMS Studio use `F6` to access the Model Library Explorer and select the `Practical CGE Library` tab. Select the library file `smod_te` and choose `Load` (or double click of the name), which is SeqNr: 9. You will be asked if you want to overwrite any of the files; you should overwrite all files that you are asked about.
2. Open `smod_t_exp1.inc` to edit it, save it as `smod_t_exp1**.inc` where `**` are wild cards, e.g., your initials.

Edit the code to generate a simulation LOOP

We will loop over a (controlling) set called `sim`. You will need to add elements to the set `sim` to control the simulations you will run. We will consider a list of experiments in which no tariffs change and then remove all tariffs in 50% increments; then we will remove tariffs only in agriculture, then only in manufacturing. There will be a total of 4 simulations.

1. Search for, `*## SETS AND PARAMETERS FOR POLICY EXPERIMENTS ####.` Note that the code has the set `SIM` defined and the notation for a set, i.e. '/' at the beginning and end of the listing of elements. The elements `base` and `sim01,` have been declared with descriptions, you will add the set elements `sim02, sim03, and sim04` between '/ and / and later add descriptions. Why replicate the base case?
2. Search for, `* ### Parameters for experiments`. This is where the parameters, `TMSIM(sim)` and `TMSIM2(c,sim)`, used for the simulations, indexed over `sim` have been declared. Note, the GAMS code knows this is a parameter because above the block of text there is the key word 'PARAMETER' which indicates a list of parameters will follow; also, the required ';' to end the command for a parameter list is found below the full list of parameters.

3. Next, you will need to assign values to the parameters `TMSIM(sim)` and `TMSIM2(c,sim)`. Search for `* ### Assigning Parameters for experiments.` Add the following lines:

```
TMSIM(sim)      = TMADJ0  ;
TMSIM2(c,sim)   = tm(c) ;
```

This sets the value for all `TMSIM` as equal to `TMADJ0` and the values for `TMSIM2` as equal to base tariffs, `tm(c);` thereafter the values for specific sims are assigned by OVERWRITING the existing value.

4. Search for, `*## DEFINING THE POLICY EXPERIMENTS ####` - this is where you will load the parameters with experiment values for each simulation, by calibrating `TMSIM` and `TMSIM2` as follows:

```
TMSIM("sim01") = TMADJ0 * 0.50 ;
TMSIM("sim02") = TMADJ0 * 0.0 ;
TMSIM("sim03") = TMADJ0 * 1.00 ;
TMSIM("sim04") = TMADJ0 * 1.00 ;
TMSIM2("cagr","sim03")   = 0.0 ;
TMSIM2("cmanu","sim04")  = 0.0 ;
```

And add `TMSIM, TMSIM2` to the `display` statement. Use the information used in these assignment statements to add descriptions to the elements of `sim` that you added.

5. When designing experiments and simulations you will typically plan out the shocks before declaring the elements in sim and before declaring the parameters to carry the experiments and simulations.
6. Search for `*## POLICY EXPERIMENTS ####` and the `LOOP keyword`. Add the controlling set over which the simulations will run.
7. In the next lines, define the policy shocks:

```
TMADJ.FX  = ??? ;
tm(c)     = ??? ;
```

For each simulation, `TMADJ.FX` will be assigned the level in the parameter `TMSIM(sim)` for that simulation; `tm(c)` will be assigned the level in the parameter `TMSIM2(c,sim)`.

8. Edit the solve statement to identify the model from the core model code:

```
Solve smod_t using MCP ;
```

9. Note that the code includes a line with '`) ;`' this ends the loop over simulations.
10. Completing the model: open `smod_t.gms` and include the experiment file you have just created. Search for `POLICY EXPERIMENTS` and add `$include smod_t_exp1**.inc` where '`**`' are your initials.
    Also add a comment above the `$include` statement that documents what you are doing in this experiment file (memory is fragile, hence making notes is important).
11. Run `smod_t.gms`. NB: in GAMS Studio the run command (F10) implements the file that is identified as the 'main file' in the active project. Make sure that `smod_t_**.gms` is the 'main file', **NOT** the experiment file `smod_t_exp1**.inc`.
12. Verify that the simulations you have programmed were successfully run – search for `TMSIM` and `TMSIM2` in the `lst` file.

An example solution is provided in the file `smod_t_exp1_sol.inc`.

Tabulating results

For ease of access the results for each variable need to be collected for each simulation. In addition, it is often helpful to use the results for the variables, without or without the values for model parameters[1], to derive values, e.g., percentage changes, summary indicators, etc., that can be useful when interpreting model results.

1. The experiment file template `smod_t_exp.inc`, is a simple template to focus on changes needed for the simulation parameters. However, there are no codes for collecting and reporting all the results.
2. The file `smod_t_exp2.inc` contains all the information in `smod_t_exp1_sol.inc` together with codes to report the levels and percentage change results for variables plus some derived values. Open that file and save it as `smod_t_exp2**.inc` where ** are your initials. Review the parameters to store levels and percent changes. Note that each parameter is indexed over `sim`. For

[1] Note that the initial values for all model variables were saved as parameters, e.g., **0(*) where ** is the variable name and (*) is the index on **. This allows the initial values for variables to be used in post simulation calculations.

example, the parameter `levQM(c,sim)` is the level of imports of commodity `c` for each simulation `sim`.

3. Run `smod_t.gms` with the experiment file `smod_t_exp2**.inc`, (it may be good practice to comment out the instruction `$INCLUDE smod_t_exp**.inc` and add the instruction `$INCLUDE smod_t_exp2**.inc`. Remember to add a comment above the `$INCLUDE` statement that documents what you are doing in this experiment file).

*Additional Results*

In addition to reporting the levels and percentage changes for all variables in the model `smod_t.gms` the file `smod_t_exp2.inc` calculates the following

1. EV – this is a measure of the equivalent variation in household welfare (`levEV(h,sim)`)
2. Price indices
    a. world price of exports - `iPWE`
    b. world price of imports - `iPWM`
    c. domestic basic price - `iPD`
    d. world price - `PW`
    e. Exchange rate - `iER`
    f. Real Exchange rate `_RER`
    g. Consumer price - `iCPI`
    h. Producer (domestic) price – `iPPI`

These are simple examples of how a CGE model can be coded to produce useful information that can inform the analysis of results and convey information contained in a report/presentation.

## 4. Trade Tax Reforms with Tax Replacement Closure

Closure rules are important because they determine where the macroeconomic effects of a policy shock, such as a reduction in tariff income, will be felt in the economy. The results from CGE models can be, and often are, sensitive to the choice of macroeconomic closure.

In this exercise, we will investigate the government closure, considering different methods to replace revenue lost when tariffs are removed. Under the default settings, tariff reduction means a decline in government savings and therefore a decline in investment. Alternatively, the government could increase other taxes to make up for lost tariff revenue. We consider tax replacement through an increase in the household income tax with the same tariff scenarios as in the previous exercise.

There are several macro closures the modeler can easily adjust:

1. the foreign exchange market closure;
2. the savings-investment closure; and
3. the government closure.

In the model, the 'default' settings (`smod_t_cl_base.inc`) are:

1. world price of imports and exports are fixed in foreign currency units,
2. fixed current account balance, flexible exchange rate,
3. fixed household savings rate,
4. exogenous tax rates,
5. exogenous real government expenditures, government savings (internal balance) adjusts,
6. investment adjusts (also called savings-driven investment).

**NOTE: the use of these settings as 'defaults' does not imply that they are the 'correct' settings. We are agnostic with respect to the choice of closure associated with different schools of economic thought.**

In addition, it is assumed that the total supply of each factor is fixed, there is full employment, and factors are completely mobile across all sectors (In the next exercise we will consider alternative assumptions about factor market clearing.)

Closure rules are important because they determine where the macroeconomic effects of a policy shock, such as a reduction in tariff income, will be felt in the economy.

In this exercise, we will investigate the government closure, considering different methods to replace revenue lost when tariffs are removed. Under the default settings, tariff reduction means a decline in government savings and therefore a decline in investment. Alternatively, the government could increase other taxes to make up for lost tariff revenue. We consider tax replacement through an increase in the household income tax with the same tariff scenarios as in the previous exercise.

Set up

1. You will continue to work in the subdirectory `C:\cgemod\smod\smod_t1`. All the files needed for the first part of this exercise will already be in the sub directory.

Changing the macroeconomic closure

In the core model file, `smod_t.gms` the model closure conditions are set in an INCLUDE file. This arrangement allows for creating multiple closure settings that can be changed simply by changing the file included.

1. Open the 'base' closure, `smod_t_cl_base.inc` and review the contents. Verify that all tax rates and government expenditure are fixed, and government savings adjusts. Note that the volume of government consumption can adjust so that government expenditure is fixed.
2. Run the model (`F10 or F9`) (see note in red below) with the tariff experiment (`smod_t_exp2**.inc`) as an include file and type '`gdx=KAPGOV_adj`' in the command line. This will generate the file `KAPGOV_adj.gdx` which has all the results for this set of simulations and closure definitions.
3. Check the results to make sure the variables that are now flexible **do** change in the results and that the variables that are now fixed do **not** change in the results.
4. Save a copy of the file `smod_t_cl_base.inc` as `smod_t_tyh_repl.inc`.
5. Edit the closures in `smod_t_tyh_repl.inc`; fix the government savings by removing the '*'before `KAPGOV.FX;` allow household income taxes to adjust by adding an '`*`' before the line of code `TYHADJ.FX = TYHADJ0;` Note that you have kept the variable and equation count – the variable `KAPGOV` is fixed and the variable `TYADJ` is now flexible.
6. Type '`gdx=TYH_adj`' in the command line

7. Run the model with the tariff experiment as an include file. This will generate the file `TYH_adj.gdx` which has all the results for this set of simulations and closure definition.
8. Check the results to make sure the variables that are now flexible **do** change in the results and that the variables that are now fixed do **not** change in the results.

**A design feature of GAMS Studio is that the last command line instruction supersedes previous instructions. The command from GAMS Configuration that generates a GDX file named `[model].gdx` is implemented before an instruction in the command line – in these exercises `gdx=KAPGOV_adj` and `gdx=TYH_adj`. Thus if you use F10 with a `gdx=****` instruction in the command line the file `[model].gdx` will not be created.**

GDXMERGE and GDXDIFF

To compare results from models with different macroeconomic closures, you could look at the parameter of interest in each `GDX` file generated when the model runs, export results to Excel and compare. It is labourious, and error prone, to compare results by 'eyeball'. There are features in GAMS that allow users to compare results systematically with less effort and less likelihood of error.

Set up

1. You will continue to work in the subdirectory `C:\cgemod\pract\smod\smodt1`. To access the files needed, in GAMS Studio use `F6` to access the Model Library Explorer and select the `Practical CGE Library` tab. Select the library file `compare` and choose `Load` (or double click of the name), which is SeqNr: 10. You will be asked if you want to overwrite any of the files; you should overwrite all files that you are asked about.
2. GAMS Studio will display the file `compare.gms` in the editor window, if it is open close the file.

A more efficient method is to let GAMS do much of the work by using the GDXMERGE and/or GDXDIFF utilities.

1. Open the file `compare.gms` in a **new project** (use `File>Open in New Project`, or `Ctrl+Shift+O`), noting that the programme `compare.gms` will have access to all files in the working sub-directory, `C:\....\smod\smod_t1`.
2. Review the code. It is a simple file that
   a. merges the two `gdx` files, `KAPGOV_adj.gdx` and `TYH_adj.gdx`. See the line

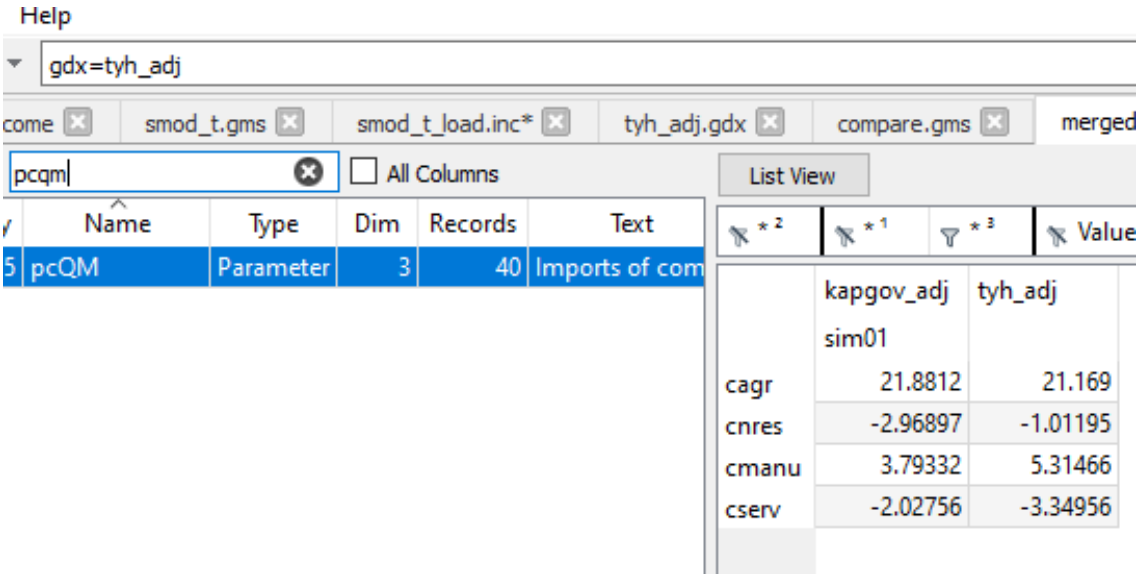
   ```
   $CALL gdxmerge KAPGOV_adj.gdx TYH_adj.gdx
   ```

   This generates a file, '`MERGED.GDX`' in which each parameter has an additional index, which has the names of the `gdx` files (in this case, `KAPGOV_adj` and `TYH_adj`).

   b. Reports the difference between the two `gdx` files, `KAPGOV_adj.gdx` and `TYH_adj.gdx`. See the line

   ```
   $call gdxdiff KAPGOV_adj.gdx TYH_adj.gdx clos_dif.gdx
   ```

   This generates a file, '`CLOS_DIF.GDX`' in which each parameter has an additional index, which identifies the files used in the order they were called (in this case, `dif1` and `dif2`).
3. Open the file `MERGED.GDX` and check that the intended changes in import duty rates have been implemented.




4. Now look at the parameter `pcQM`, the percent change in quantity imported by sector and simulation. Results for each closure assumption are shown below for `sim01`.
5. Rename the file `MERGED.GDX` to `MERGED_clos**.gdx` where `**` are your initials.

6. Review how the differences are reported when using GDXDIFF.

Changing the macroeconomic closure: investment driven savings with tax replacement

When using the closed economy model, you developed a so-called 'Keynesian' closure where the investment_savings account was equilibrated by changes in savings behaviour to meet exogenously determined investment decisions.

For this exercise modify the file in `smod_t_tyh_repl.inc`; so that the only change is that investment is exogenous. Use the solution from the closed economy model as a guide. Save the file with a new name and rerun the experiments with `gdx=SAV_adj.gdx`.

Then extend the `$CALL gdxmerge` and the `$CALL gdxdiff` so that the THREE GDX files are compared.

*Analysis*

Analyse the results. The shock is a trade shock where import duties decline, so it is appropriate to use the price trees as a guide and start where the shock is introduced into the system. You should emphasise explanation not simple reporting of the results.

Among others you should report on:

a) Import prices (*PM*) and quantities (*QM*)
b) Export prices (*PE*) and quantities (*QE*)
c) Domestic prices (*PD* and *PQS*) and quantities (*QD* and *QQ*)
d) Wage rates (*WF***WFDIST*)
e) The exchange rate
f) Internal and external balances
g) Household consumption (*HEXP* and *QCD*) and welfare (*EV*)
h) Production structure (*QX* and *QXC*)
i) etc.

Write a short explanation for why the results differ for each closure. Do any closures borrow from the future, incur unrequited debts, etc.?

# 5. Trade Policy and Factor Market Clearing

Assumptions about factor market clearing are important to your analysis of trade results. The default assumption in `smod_t.gms` is full employment and long run – all factors are mobile across all activities. Trade reform may have a different effect on an economy in the short run, when some factor, such as capital, is activity specific. Likewise, the results from trade reform may be very different in the short run, i.e., one or more factors is immobile, or when there is surplus labour in the economy (commonly referred to as an unemployment clearing condition). In this exercise, you will change the factor market clearing conditions and analyze the impacts of trade reform. Results will be compared using the GDX MERGE utility. The default macroeconomic closure assumption is that tax rates and government spending are fixed, and government savings adjusts.

Set up

1. You will continue to work in the subdirectory `C:\cgemod\smod\smod_t1`. No additional files are needed from the library.

Edit code to change factor market clearing

Long run factor market clearing conditions are set as the 'default' setting, i.e., it is assumed that there is full employment, and all factors are fully mobile across all activities. You have already run the trade liberalisation scenario with the default closure and have the files `KAPGOV_adj.gdx` and `TYH_adj.gdx` that store the results.

1. Open the core model, `smod_t.gms` and restore the default marcoeconomic closure assumptions (change the closure INCLUDE file).
2. Search for *## FACTOR MARKET CLEARING and review the settings for `Basic Factor Market Clearing Conditions` and the section for `* Alternative Factor Market Clearing Conditions`.
3. In Module P2 sets of 'generalized factor market clearing equations'; were introduced. The 'alternative factor market clearing conditions' follow the same arrangement but are set up for FOUR rather than two factors.
4. Review the 'alternative factor market clearing conditions' and interpret the meaning of the base setting.

# NB: When changing closure settings remember the key rule:

# FIX ONE, FLEX ONE.

Short run Factor Market Clearing

This is a classic short run setting where capital is assumed to be activity specific, i.e., capital cannot be reallocated (is immobile) between activities. In the short run, capital is activity specific in all activities but in fixed supply.

1. Open the 'base' closure, `smod_t_cl_base.inc` and review the contents. Verify that all tax rates and government expenditure are fixed, and government savings adjusts. Note that the volume of government consumption can adjust so that government expenditure is fixed.
2. Save a copy of the file `smod_t_cl_base.inc` as `smod_t_shortk.inc`.
3. Open the new closure file, `smod_t_short.inc`, fix factor demand for capital in each activity, 'switch' on the code
   ```
   FD.FX('fcap',a) = FD0('fcap',a);
   ```
   This fixes 4 variables: the demand for capital in each of the 4 activities.
4. Four variables that had been held constant must be made flexible to keep the count between the number of variables and equations in the system. The activity specific factor productivity variable will adjust. 'Switch' on the code:
   ```
   WFDIST.LO('fcap',a) = - inf;
   WFDIST.UP('fcap',a) = + inf ;
   ```
   Be sure to use an upper case letter "O" for `WFDIST.LO('fcap',a)`, NOT a zero "0".
5. For 'tidiness'[1] 'switch' off the code
   ```
   WFDIST.FX("fcap",a)    WFDIST0("fcap",a) ;
   ```
6. Note that the equation for factor market clearing for the factor '`fcap`' is redundant because the supply is fixed at the base level and the quantity demanded in each sector is also fixed at the base level. So 'switch' off the code
   ```
   FS.FX("fcap")      = FS0("fcap") ;
   ```

[1] It is 'good' practice to make sure that when fixing and/or flexing variables to ensure that the previous instructions are turned off or on. GAMS will impose the last instruction it encounters when compiling the model, hence actively controlling the properties of variables is a risk averse strategy.

7. There is no need for the variable `WF('fcap')`, because the change in factor returns will be captured as differences in the returns by activity, i.e., `WFDIST('fcap',a)`. 'Switch' on the following code:
   `WF.FX('fcap') = WF0('fcap');`
8. Run the model, writing `gdx=short_run` in the command line.

Now consider the case in which land is fixed, but capital is mobile across activities.

1. Save a copy of the file `smod_t_short.inc` as `smod_t_shortn.inc`.
2. Use the same process to make land activity specific as used to make capital activity specific.
3. Run the model, writing `gdx=fland_aagr` in the command line.

*A Challenge*

Should the short run factor market clearing condition for land be used in **all** experiments with **this** dataset? If so why, if not why?

Surplus Labour Market Clearing

**A misunderstanding of this factor market clearing condition has been seen in many applications of global CGE models.**

Consider the case where there is 'surplus' unskilled labour; this is often described as unemployment although that is an imprecise description. In a simple representation of surplus unskilled labour, the supply of the unskilled labour is unconstrained and there is an infinitely elastic supply of unskilled labour at a constant price:[1] hence the wage rate for unskilled labour is constant.

1. Save a copy of the file `smod_t_short.inc` as `smod_t_surp_usklb.inc`
2. In the file `smod_t_surp_usflb.inc` turn off the short run factor market clearing condition you created for a short run closure with capital or land as the fixed factor, i.e., return the factor market clearing conditions to those whereby all factors are fully employed and fully mobile.
3. Allow the supply of unskilled labour (`FS('fusklb')`) to be flexible.
4. Fix the payment to unskilled labour (`WF('fusklb')`).

[1] This is similar in spirit to the classic model by Sir Arthur Lewis (Lewis, 1954).

5. For tidiness 'switch' off the code:

```
WF.LO('fcap',a) = - inf;
WF.UP('fcap',a) = + inf ;
```

6. Run the model, writing `gdx=surp_fusklb.gdx` in the command line.

Comparison of the Results

1. To compare the results across the three different assumptions about factor market clearing, use the `GDX MERGE` utility. Open the file `compare.gms` and save it as `compare2.gms`.
2. Edit the call statement to refer to the four `gdx` files to compare: `KAPGOV_adj.gdx short_run.gdx fland_aagr.gdx unemp_fusklb.gdx`. Note that there are only spaces between the names in the call statement.
3. Run `compare2.gdx` (make sure `compare2.gms` is the Main File before running the model). Review the merged file, `MERGE.GDX`.
4. Save `MERGE.GDX` as `MERGE_factor**.GDX` where `**` are your initials.

*Analysis*

Analyse the results. The shock is a trade shock where import duties decline, so it is appropriate to use the price tress as a guide and start where the shock is introduced into the system. You should emphasise explanation not simple reporting of the results.

Among others you should report on:

j) Import prices (*PM*) and quantities (*QM*)
k) Export prices (*PE*) and quantities (*QE*)
l) Domestic prices (*PD* and *PQS*) and quantities (*QD* and *QQ*)
m) Wage rates (*WF***WFDIST*)
n) The exchange rate
o) Internal and external balances
p) Household consumption (*HEXP* and *QCD*) and welfare (*EV*)
q) Production structure (*QX* and *QXC*)
r) etc.

Write a short explanation for why the results differ for each closure. Do any closures borrow from the future, incur unrequited debts, what does the short run factor market clearing

condition for land do, etc. How sensitive are the results to macroeconomic closures and factor market clearing conditions?

Sample solutions for the macroeconomic closures and factor market clearing conditions are provided:

- `smod_t_cl_tyh_repl_sol.inc`,
- `smod_t_cl_tyh_repl_keyn_sol.inc`,
- `smod_t_cl_tyh_repl_shortk_sol.inc`,
- `smod_t_cl_tyh_repl_shortn_sol.inc`,
- `smod_t_cl_tyh_repl_surp_sol.inc`

# 6. The Stone-Geary Utility Function

The Stone-Geary utility functions is used in many CGE for the determination of consumption patterns by households; it is particularly popular with models that focus on low-income/developing economies where low-income households are deemed to require subsistence consumption quantities. It is not the only utility function used in CGE models; other choices include CES, Constant Demand Elasticity (CDE), An

Implicit Direct Additive Demand System (AIDADS), CD, nested LES/CES, nested CES, etc., some of which have increased in popularity as models have been elaborated.

This short exercise is solely concerned with the calibration of a LES function. From the day-to-day user's perspective calibration of a LES only involves making changes in two worksheets in the Excel workbook: `hoelast` (income elasticities of demand) and `frischelast` (Frisch parameter - elasticity of the marginal utility of income with respect to income). But it is useful to appreciate the calibration steps in the model code.

A problem is that a complete demand system requires that expenditure on commodities must exhaust the funds available for consumption (`HEXP` in `smod_t`). Hence, a complete demand system requires that the 'Weighted sum of the expenditures elasticities must equal one, where the weights are average budget shares. When normalised, such elasticities will satisfy Engel aggregation'. But how can the user ensure that the income elasticities of demand are consistent?

This is achieved in `smod_t` when calibrating a LES function. If the elasticities of demand ($ELASTY_{c,h}$) that are read into the model are summed using average budget shares ($alphah_{c,h}$) and the resultant aggregates ($sumelast_h$) are normalised the resultant elasticities ($yhelast_{c,h}$) are consistent.

```
sumelast(h) = SUM(c, alphah(c,h)*ELASTY(c,h)) ;

yhelast(c,h)$sumelast(h) = ELASTY(c,h)/sumelast(h) ;
```

The normalised elasticities can then be combined with the average budget shares to estimate the marginal budget shares ($beta_{c,h}$). The subsistence quantities ($qcdconst_{c,h}$) can then be determined using the marginal and average budget shares and Frisch parameter ($ELASTMU_h$).

```
beta(c,h)      = yhelast(c,h)*alphah(c,h) ;
```

```
qcdconst(c,h)$(PQD0(c) and frisch(h) and beta(c,h))
    = [HEXP0(h)/PQD0(c)]*(alphah(c,h)+(beta(c,h)/frisch(h)));
```

Understanding some of the properties of this utility function can serve to increase awareness of the implications of different functions. The following exercises are designed with this in mind.

1. Compare the values of $ELASTY_{c,h}$ and $yhelast_{c,h}$, paying special attention to the **relative** magnitudes of the initial and the normalised values.
2. In the Excel workbook (`smod_t_mod05_data.xlsx`) make a copy of the elasticities in the worksheets `hoelast` and `frischelast` and save them so that you retain the values for reuse. Now for the rich household, set the Frisch parameter equal to (minus) 1 and all the income elasticities for that household equal to one.
    a. Rerun the model
    b. Compare the values for of $ELASTY_{c,h}$ and $yhelast_{c,h}$
    c. Compare the values for $alphah_{c,h}$ and $beta_{c,h}$ for the rich household. Why are the values the same?
    d. What do you expect to be the values of $qcdconst_{c,'h\_rich}$'; what are the values and why?
3. Now make the Frisch value for the poor household more negative.
    a. Rerun the model.
    b. How do you expect this change to affect the values of $qcdconst_{c,'h\_poor}$'; why do you expect the result and what are the values and why?
4. Use the copied values for `hoelast` and `frischelast` that were in the original database.
5. Use the `compare.gms` programme to compare the results.

Use the trade policy simulations with your preferred macroeconomic closures and factor market clearing conditions to assess the impact of different elasticities on the results. You should focus on explaining **why** you get the results not the values in the results.

The exercises in Module P6 will allow to experiment with different values for `hoelast` and `frischelast`. These values will become more important during the second part of the course.

# 7. International Transfers

The values for all international transfers in the database supplied are all zero and the only transactions between each country and the rest of the word are for trade in goods & services and current account balance, which is identical to the trade balance. This reflects the fact that the source data – the Global Trade Analysis Project (GTAP) – assumes all (bilateral) international transfers, except trade in goods and services, are zero.

This model allows you to shock/change net remittances to households and net grants/aid to government. No detailed instruction for running simulations is provided; but such shocks are legitimate and can be used for the projects in Module P6.

Some hints about such shocks are, however, appropriate.

1. The size of the shocks should be realistic – scaling the shocks relative to GDP or government or household incomes may be wise.
2. The magnitudes for remittances need to be set for each household; would you expect poor or rich households to have the larger absolute or relative remittances?
3. A positive change in international transfers indicates that the country is a net recipient.
4. A negative change in international transfers indicates that the country is a net donor.
5. Some countries may be simultaneously net donors and net recipients.

Feel free to experiment with different patterns and/or magnitudes of international transfers. You will learn more about how CGE models might behave by experimenting. **NB**: you do not have to conduct such experiments during the course, they can be done after the course.

Use the trade policy simulations with your preferred macroeconomic closures and factor market clearing conditions to assess the impact of different elasticities on the results. You should focus on explaining **why** you get the results not the values in the results. Use the `compare.gms` programme to compare the results.