2006/8

May 2006

Documents

Statistics Norway

Statistisk sentralbyrå

Statistics Norway
Department of Management Support

Robin Choudhury

User's Guide for a Macroeconomic Model for Malawi

Version 1, May 2006

Abstract

Designing and constructing a system for simulating macroeconomic models can be done in numerous ways. Regardless of how the system has been designed, and what software tool has been chosen to facilitate the simulations, documentation of the system is of great importance. This documentation has two target groups; the technical experts who will work on the system on a frequent basis, and users whose sole intention is to run the model for the sake of analysing the Malawian economy. For the first mentioned group the User's Guide gives detailed information on how to update and extend the database, as well as how to add and modify equations in the model. For the second group the focus is on the process of using the model for forecasting and policy analysis. Some material is intended for both groups, such as the organisation of the system, and how to interact with spreadsheets. It should be stressed however, that this is not a user guide for the TROLL software. As already mentioned, a model system can be developed in many ways and the users of the system should feel free to carry out the various task in their own manner. As the model system is being used and developed further the need for new routines and programmes to handle it will emerge. So equally important as to have this User's Guide, is to make sure it is up to date. Consequently, this guide will be updated as major changes to the system take place.

I would like to thank Torfinn Harding for useful comments to an earlier version of the User's Guide.

Table of contents

List of Figures

List of Tables

List of Boxes

1 Overview

1.1 Introduction

The TROLL software is a tool for both model builders and model users. For the first group, the flexibility of the software is a great benefit. Using TROLL we can, more or less, take into account whatever we want regarding numerical timeseries modelling. But this flexibility has a cost. The way to understand and operate the model system is not an easy one, but as soon as an adequate level is reached the user has a powerful tool. For the second group, the model users, it is possible to do forecasts and policy analysis without having to understand much of the technicalities. This document is summing up the main methods and procedures used to handle the Malawi model and its surroundings. This document relates to the first version of the Malawi model as far as April 2006. The system is going to be revised so changes to some of the routines are inevitable. After major changes this document will be revised accordingly.

This note elaborate on how to use the macroeconomic modelling tool developed to analyse the Malawian economy. The modelling tool consists of all that is needed to use the model: databases for timeseries and coefficients, the equations of the model, and macros and programs to handle various tasks. The purpose of this user's guide is twofold; first of all, it should help users with little or no experience in TROLL to bring the tool into action, but it should also serve as a quick reference, particularly for technical tasks performed infrequently.

Before using the model one should have an overview picture of what the model tool consist of and how it is organised. Chapter 2 gives an outline of the directory and file structure, and a detailed description of their content and purpose. In Chapter 3 we explain the basics. The user will learn how to display equations and data on the screen, and some useful TROLL commands are demonstrated. The principal use of the model tool is to do simulations. In Chapter 4 we enter into details on the forecasting (simulation) process. We start with explaining how to extrapolate and make assumptions about exogenous variables, and then how to simulate the model based on these assumptions. Next, in Chapter 5, we explain how to perform policy analysis to study effects of alternative policies. In Chapter 6 we learn how to edit and install the model. We will describe how to delete, add and modify equations. Databases containing historical timeseries are a central part of the model system. In Chapter 7 we gives a detailed description of the databases. This chapter applies to both end-users who wants to know how to implement their own data, and to technical users whose task is to update or revise the databases. Sometimes we want to oversteer the results from the model. In this chapter we will demonstrate how this can be done. The use of spreadsheets has become increasingly popular. TROLL has got functionality for communicating with spreadsheets. In addition, some programs have been developed especially for this modelling tool to facilitate this. The use of spreadsheets is demonstrated in Chapter 8.

The user's guide aims at putting the most technical parts into the appendices. We will describe how the functions for adjusting exogenous variables are used in Appendix 1. The initial setting will be described in detail in Appendix 2. In Appendix 3 and Appendix 4 we briefly describe the external TROLL library and the variables list respectively.

1.2 Conventions used in this manual

In most chapters in this manual we describe the use of commands or functions by examples from the model system. We have tried to be consistent in using formatted text to highlight the meaning and the context of various keywords, functions etc. In

Table 1 Convention used in this manual

Indicator	Example	Comment
Italics	*lkaccess*;	TROLL commands
`Courier New`	`do prtime(ypct'f(x);`	▪ Example from TROLL's output ▪ Referring to TROLL commands, keywords, etc. ▪ In response to the use of programs, macros, etc
SMALL CAPS	MAIN.FRM	File/path names
UPPER CASE	CPO	Variable names (in text)

There may be mixture of notation i.e. when referring to the database file SIMINPUT.FRM as a file we will type it in SMALL CAPS, while if the simulation program asks for the name of this file we will refer to it as a keyword and write `siminput` in `Courier New`. Similarly, when using the TROLL command *lkaccess;* we will put it in *Italics* as a command we have actually executed, but if we mention the command `lkaccess` in some other context we put it in `Courier New`.

We use flowcharts to illustrate the use of various programs. When we write name of databases and other file names to exemplify, we underline the optional names. When only the first part of the filename is optional (not the extension) as will be the case for databases, we underline only the optional part as in DATABASE.FRM.

2 Directory and file structure

The dialogue between the user, the model system and the TROLL software is carried out via commands on the keyboard, through files and the screen. Typically a program is executed from the keyboard giving some response to the screen prompting for input. Input regarding file names, dates for simulation periods, etc. is usually given from the keyboard, while input of greater quantities of data are usually provided through files. Output from estimations and simulations will similarly be printed to the screen and/or to files. This interactive mode will create a great deal of files. These files can be macros, databases, models and so on. In this chapter we will look at the organisation of these files in the model system. Figure 1 shows the file structure of the Malawi model system. The reason we arrange it in this way is to organize our work. If we have all the files in one directory it soon will become quite chaotic and the user will be confused.

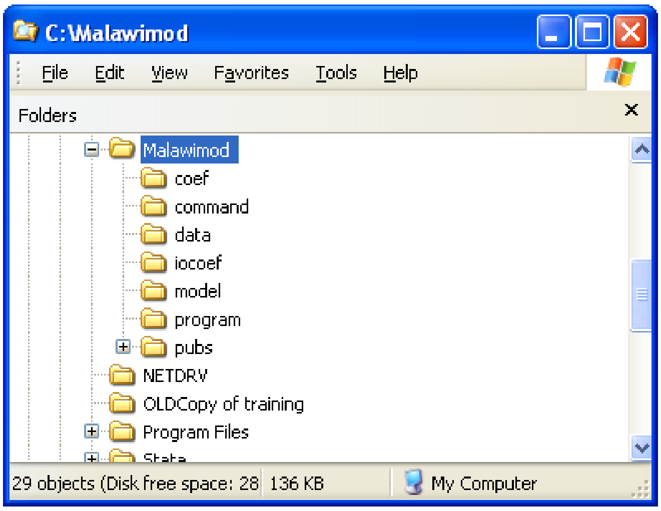
Note that for the time being (April 2006) the COEF, COMMANDS and IOCOEF directories are not in use. Note also that there is another directory called C:\TROLLRESOURCES\LIB used as part of the model system. This directory contains TROLL programs that we have developed. The reason they are not in the PROGRAM sub-directory under MALAWIMOD is that they are more general and can be used by other model systems as well. They are for example used for checking if a database exist or not, for printing warnings or error messages, and others not specific to the Malawi model system.

2.1 The Working Directory

The main directory is called MALAWIMOD and it contains both sub-directories and files. In TROLL this main directory is referred to as “Working Directory”. Throughout this document we will assume that the main directory is C:\MALAWIMOD\ unless otherwise explicitly specified. The Working Directory in TROLL has a special function in the host file system. This means that there is a default access and search to this directory. This feature of the Working Directory makes it convenient to store some macro and program files at the Working Directory for immediate access.

There are several sub-directories, each intended for storing a specific type of files, or files meant for a specific task. The naming of the directories is flexible, and in Appendix 2 we will explain

Figure 1 The file structure of the Malawi model



```
Malawimod
    coef
    command
    data
    iocoef
    model
    program
    pubs
NETDRV
OLDCopy of training
Program Files
Stata
```

29 objects (Disk free space: 28 136 KB My Computer

how these can be changed. The following list (c.f. Table 2) gives a short description of the purpose for the sub-directories:

Table 2 The directory structure of the Malawi model

Directory	Purpose
coef	To store coefficients
command	To store input (batch) files
data	To store databases
iocoef	To store input-output coefficients
model	To store models
program	To store programs
pubs	To store tables and files for creating them

It is important to be aware of that the contents of the various directories will change over time as use of the model will generate some files; both as a result of doing something, but also when the user create files to execute some commands which he wants to save for later use. Anyhow, in Table 3 are a list of some of the files that will be at the Working Directory "C:\MALAWIMOD" and a short description of its purpose:

Table 3 Files at the Working Directory (not exhaustive)

File name	Description
ADDDATA.INP	Install data to the historical time series database (MAIN.FRM).
COLLECT. PRG	Program to collect data for variables in a model.
COLLECT. SRC	Source code for COLLECT. PRG.
DATAFLOW.INP	Example file for the whole process of installing historical data to do a simulation.
ENDDATES.LOG	End dates of variables before they are extrapolated. Created from executing the program AUTOEXTRAP. PRG.
EXOGVALUES.TXT	Text file containing assumptions for future values on exogenous variables.
LASTSIM.INP	Auto generated file from the program SIM. PRG, which opens the databases and specify the model used (with readable access) from the last simulation.
MAKEDATA.INP	Gives access and search to the directories: COMMAND, PROGRAM and TROLLRESOURCES.

MAKEEQN.INP	Gives access and search to the model directory.
MAKEMODEL.INP	Install the model from this text file (deletes the existing).
MAKETABLES.INP	Generates some tables (under construction).
MODSYS.LOG	Log file from execution of functions to adjust variables.
OPENDB. PRG	Program to open databases.
OPENDB.SRC	Source code for OPENDB. PRG
PROFILE.INP	Input file that runs automatically at start-up.
PROFILE.PRG	Program containing initial settings, called from PROFILE. INP.
PROFILE.SRC	Source code for PROFILE.PRG.
START.INP	Gives readable access to the default databases for timeseries, coefficients, model, macros and programs. Specifies to use the default model.
TROLL.LOG	Log file from a TROLL session (echo of the session window).
VARLIST.PRG	Program to print explanations for symbols in the model.
VARLIST.SRC	Source code for VARLIST.PRG.
VARLIST.TXT	Text file containing symbols in the model and their explanations.

2.2 The data directory

The C:\MALAWIMOD\DATA directory is meant for timeseries databases. It will contain both the historical database and simulations of forecasts and policy shifts that we store. The name of the historical database is preset in the PROFILE.SRC file, and is by default set to MAIN.FRM.

Table 4 Files at the data directory

File name	Description
MAIN.FRM	Historical time series database.
MALMODDB.FRM	Historical time series database for variables used in the model.

Other database files at this directory will have names given by the user during installation of exogenous assumptions and simulations.

2.3 The model directory

At the C:\MALAWIMOD\MODEL directory we store models. Writeable access and search to this directory is obtained by running the file MAKEEQN.INP.

Table 5 Files at the model directory

File name	Description
MALAWIMOD.MOD	The default model file.
TMPMOD.MOD	Model resulting from the calibration routine.

2.4 The program directory

The program directory contains programs developed for handling various tasks related to operating the model system. Table 6 lists only the compiled files. The directory also contains the source code, files with the same name but the extension “src”.

Table 6 Program developed for handling the model system

File name	Description
ADDTOLOG.PRG	Writes a string to the log file.
AUTOEXTRAP.PRG	Extrapolates models variables automatically.

CHDATA.PRG	Manipulates on variables according to information from an external text file.
CHECKDATE.PRG	Reads timeseries from a database. Check for start date and end date and write results to screen.
ENDO2EXOG.PRG	Change symboltype between endogenous and exogenous and store in a temporary model. Simulates the temporary model, stores the variables that have been changed into endogenous in a database.
EQEVAL.PRG	Evaluates the equations in a model by one of three possible choices: evaluation of the right hand side, the left hand side or the residuals.
ESTMOD.PRG	For estimating econometric equation in a model. (Not in use.)
RESCONS.PRG	For calibrating additional constant terms in an equation. (Not in use.)
SHIFTPRG	For doing a policy shift.
SIM. PRG	Simulation program.
TROLL2WKS.PRG	Write TROLL data to a spreadsheet.
WKS2TROLL.PRG	Write spreadsheet data to a TROLL database.
WKSSIM.PRG	Simulate a model using data from a spreadsheet.

2.5 The pubs directory

This directory is used for writing tables based on simulations. By using the MAKETABLES.INP file in the Working Directory, the file INDEX.HTML is created. This file can be opened in a web-browser and contains links to the tables produced.

Table 7 Files in the pubs directory

File name	Description
BOPHTML.INP	Creates the balance of payments table.
FISCALHTML.INP	Creates the fiscal table.
INDEX.HTML	Starting point for browsing the tables.
INDEX.INP	Creates the index.html file.
MACRO1HTML.INP	Creates table with growth rates for macroeconomic figures.
MACRO2HTML.INP	Creates table with levels for macroeconomic figures.
PRICESCOSTSHTML.INP	Creates table with prices and costs.
SPACER.GIF	Used for adjusting the spaces in the tables.
TAB0HTML.INP	Creates table with GDP by expenditure (constant 1994 prices).
TAB1HTML.INP	Creates table with GDP by expenditure (annual percentage change).
TAB2HTML.INP	Create table with GDP by sector of origin (at constant 1994 factor cost).
TABLES	Sub-directory to store the tables.
A SMALL MACROECONOMIC MODEL FOR MALAWI.PDF	Documentation of the first version of the model.
USER'S GUIDE.DOC	This user guide for the model.

3 Basic handling of the model system

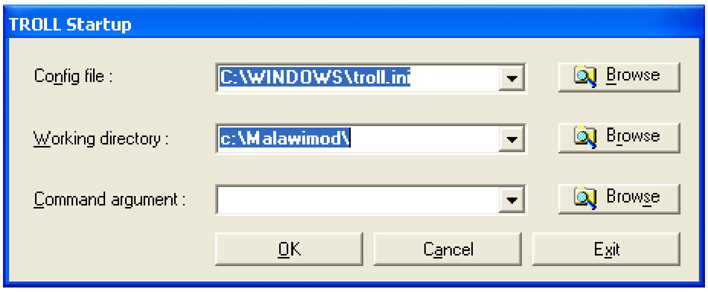
3.1 Starting up

In this chapter we describe how to start TROLL and to specify the Working Directory. We also mention some important initial settings, which will be explained in details in Appendix 2 “Initialisation program”.

There are times when we want to study the model more closely. We may want to see a particular equation or the values of time series. In this chapter we explain how to display this information to the screen, and, in connection to this, we will learn some important TROLL-commands.

To start TROLL we double click the TROLL icon on the desktop or click Start -> All Programs -> TROLL for Windows -> TROLL for Windows. The TROLL Startup window, shown in Figure 2, will appear. Now we must choose our Working Directory by typing C:\MALAWIMOD\ (or by using the Browse-button) as in Figure 2.

Figure 2 The TROLL Startup window



Config file : C:\WINDOWS\troll.ini

Working directory : c:\Malawimod\

Command argument :

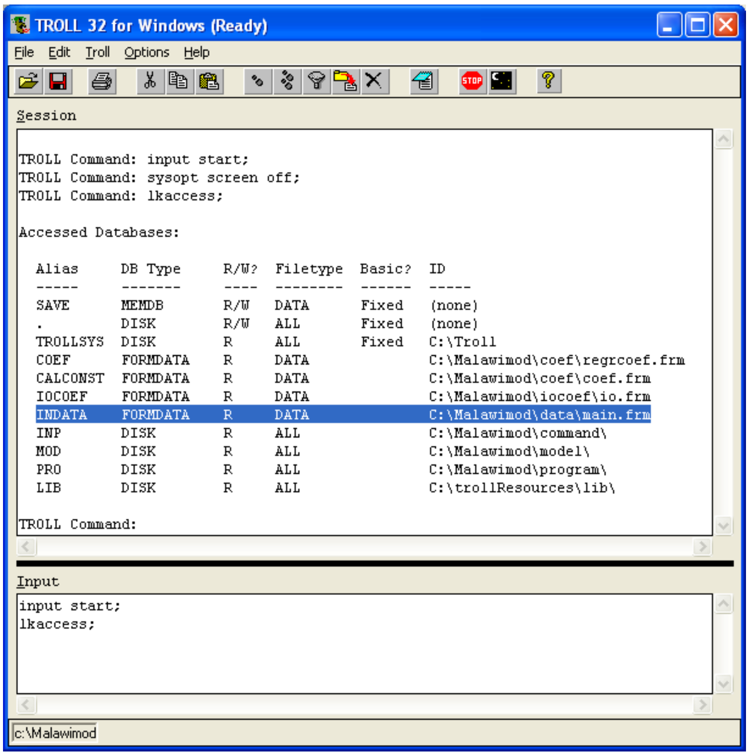
When TROLL is started and C:\MALAWIMOD is chosen as Working Directory, some important things are going on behind the scenes. Every time TROLL is started it will look for a file called PROFILE.INP. This is a feature of TROLL. If this file exists it will be executed. For the Malawi model system we have defined such a file, which in turn, executes a TROLL program called PROFILE.PRG. The reason for this is to initiate some settings. Most of all we want to save the various path names of the host system in TROLL’s memory.

3.2 Displaying model and data

3.2.1 Accessing the model and the database

The first thing to do, after we have started TROLL and defined the Working Directory, is to access the model and the databases, i.e. to tell TROLL which databases and which model we want to make available. In the default settings of the model system we have predefined a default database and a default model to use. They are accessed by typing “*input start;*”. This is illustrated in Figure 3. There we also use the command “*lkaccess*” which produces a list over the accessed databases. From that list we see (amongst others) the default database MAIN.FRM, which contains all the historical timeseries, and the id for the model directory C:\MALAWIMOD\MODEL. Running the START.INP file also builds the search list, which is necessary to tell TROLL in which order we want to look for data. This is because we may have multiple databases with the same variable names.

Figure 3 Accessing the default database and model

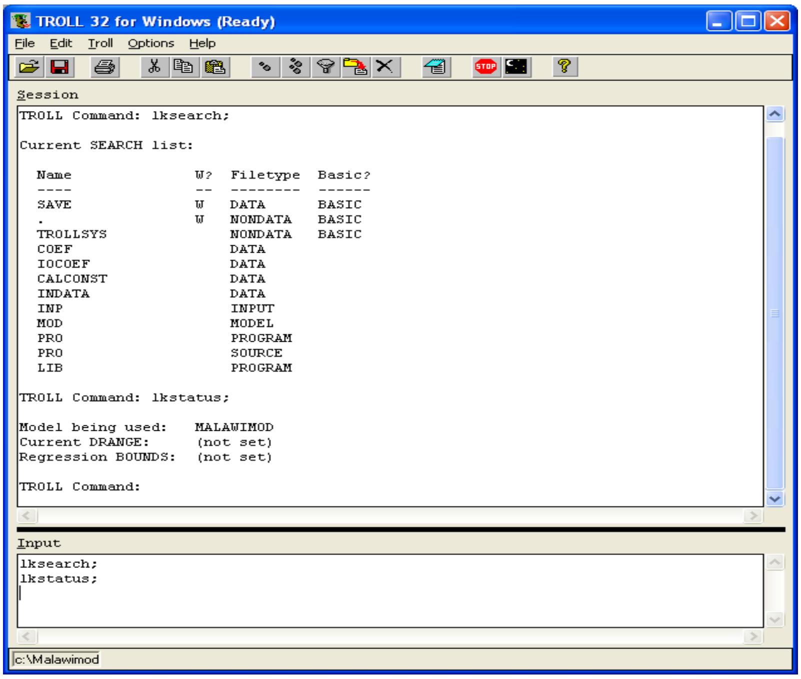


```
TROLL Command: input start;
TROLL Command: sysopt screen off;
TROLL Command: lkaccess;

Accessed Databases:

  Alias     DB Type     R/W?  Filetype  Basic?  ID
  -----     -------     ----  --------  ------  -----
  SAVE      MEMDB       R/W   DATA      Fixed   (none)
  .         DISK        R/W   ALL       Fixed   (none)
  TROLLSYS  DISK        R     ALL       Fixed   C:\Troll
  COEF      FORMDATA    R     DATA              C:\Malawimod\coef\regrcoef.frm
  CALCONST  FORMDATA    R     DATA              C:\Malawimod\coef\coef.frm
  IOCOEF    FORMDATA    R     DATA              C:\Malawimod\iocoef\io.frm
  INDATA    FORMDATA    R     DATA              C:\Malawimod\data\main.frm
  INP       DISK        R     ALL               C:\Malawimod\command\
  MOD       DISK        R     ALL               C:\Malawimod\model\
  PRO       DISK        R     ALL               C:\Malawimod\program\
  LIB       DISK        R     ALL               C:\trollResources\lib\

TROLL Command:
```

```
input start;
lkaccess;
```

To see the search list we type the command “*lksearch*;”. This, in addition to the command “*lkstatus;*” that gives status of the model used among other things, is illustrated in Figure 4. As we can see from Figure 3, the internal name (alias) of the main MAIN.FRM database is “`INDATA`”, which is also on the search list in Figure 4, together with the name of the model being used (`malawimod`).

Figure 4 Displaying the search list and the model used



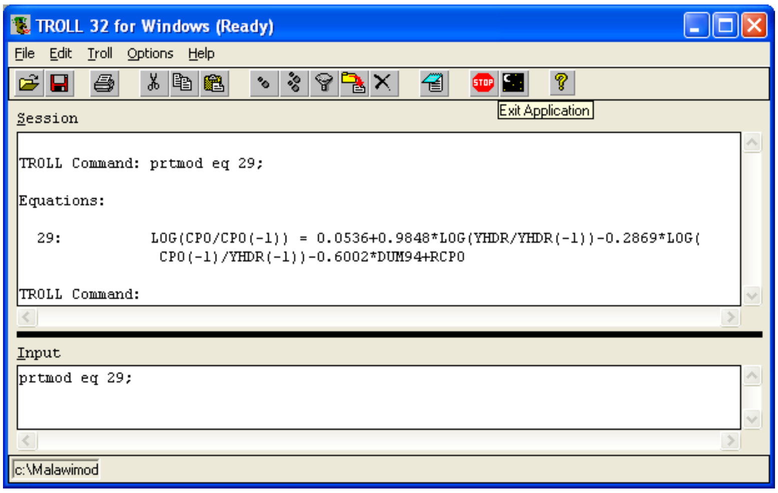
```
TROLL Command: lksearch;

Current SEARCH list:

  Name                W?  Filetype  Basic?
  ----                --  --------  ------
  SAVE                W   DATA      BASIC
  .                   W   NONDATA   BASIC
  TROLLSYS                NONDATA   BASIC
  COEF                    DATA
  IOCOEF                  DATA
  CALCONST                DATA
  INDATA                  DATA
  INP                     INPUT
  MOD                     MODEL
  PRO                     PROGRAM
  PRO                     SOURCE
  LIB                     PROGRAM

TROLL Command: lkstatus;

Model being used:    MALAWIMOD
Current DRANGE:      (not set)
Regression BOUNDS:   (not set)

TROLL Command:
```

```
lksearch;
lkstatus;
```

3.2.2 Printing equations to the screen

Now that we have opened the database for the data and the model, and added them to the search list, we can print them to screen. There are many ways of doing this and we will show only a couple of them. If we want to have a look at the consumption function of the model we can print the equation with the command "*prtmod eq 29;*" which tells TROLL to print equation number 29. This is demonstrated in Figure 5.

Figure 5 Printing an equation

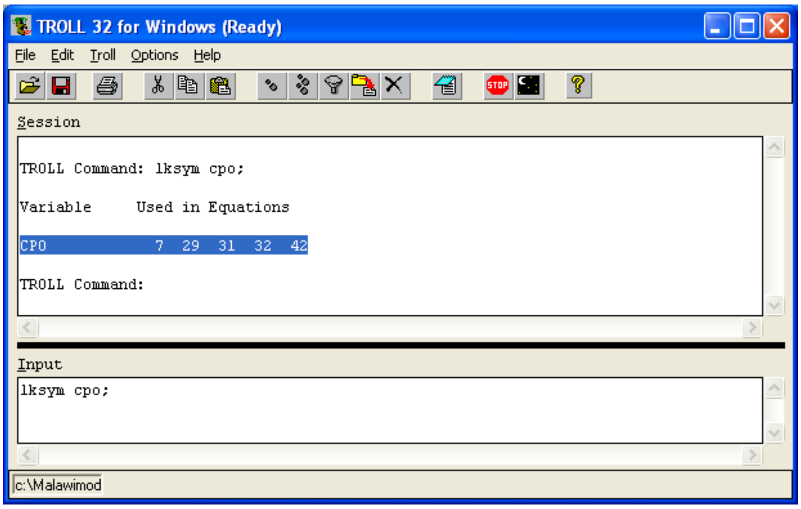


```
TROLL Command: prtmod eq 29;

Equations:

  29:          LOG(CPO/CPO(-1)) = 0.0536+0.9848*LOG(YHDR/YHDR(-1))-0.2869*LOG(
                CPO(-1)/YHDR(-1))-0.6002*DUM94+RCPO

TROLL Command:
```

```
prtmod eq 29;
```

In this example we assumed that we knew the equation number for the consumption function. This is not always the case, in fact, a model change over time as equations are added or deleted, so usually we don't know the equation numbers. But we can, based on our knowledge of the symbols of the variables, look up equations. We know that the symbol for non-smallholders consumption is CPO. By typing the command "*lksym cpo;*" we lists all the equations using that

Figure 6 Looking up equations based on a symbol



```
TROLL Command: lksym cpo;

Variable      Used in Equations

CPO                 7  29  31  32  42

TROLL Command:
```

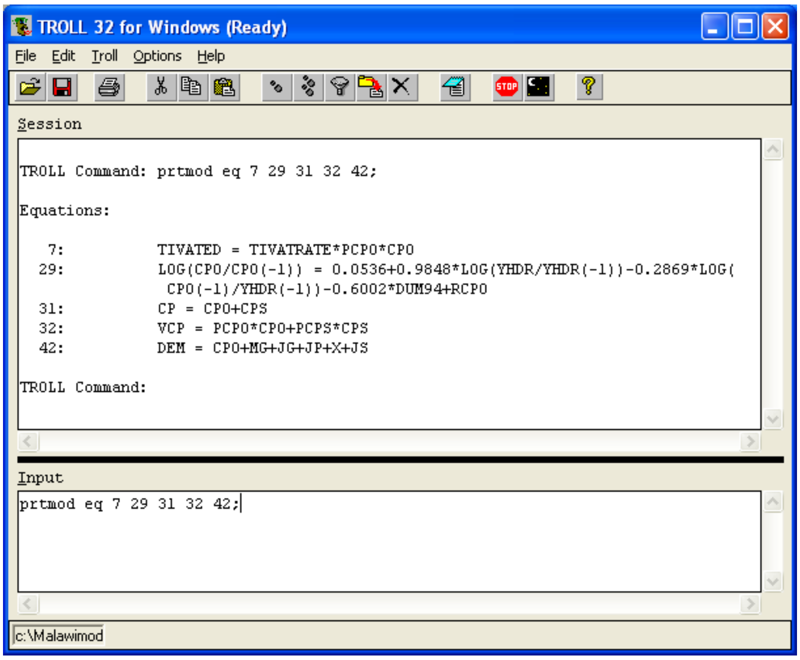
```
lksym cpo;
```

symbol. This is illustrated in Figure 6. As we can see, CPO appears in equations number 7, 29, 31, 32 and 42. To print them we use the `prtmod`-command demonstrated earlier. In this case we type:

```
prtmod eq 7 29 31 32 42;
```

The result from this command is showed in Figure 7.

Figure 7 Printing many equations



```
TROLL Command: prtmod eq 7 29 31 32 42;

Equations:

   7:           TIVATED = TIVATRATE*PCPO*CPO
  29:           LOG(CPO/CPO(-1)) = 0.0536+0.9848*LOG(YHDR/YHDR(-1))-0.2869*LOG(
                 CPO(-1)/YHDR(-1))-0.6002*DUM94+RCPO
  31:           CP = CPO+CPS
  32:           VCP = PCPO*CPO+PCPS*CPS
  42:           DEM = CPO+MG+JG+JP+X+JS

TROLL Command:
```

```
prtmod eq 7 29 31 32 42;
```

Box 1 Identifying the equation number of an econometric variable

TIP

As we saw in the example on how to locate the equations containing CPO, it produced many equations. If we know the variable we are looking for is an econometric one, we can instead look up the residual since it will appear in only one equation. In our example we could have typed ”*lksym rcpo;*”, where RCPO is the name of the residual for CPO. We would have the following:

```
TROLL Command: lksym rcpo;
Variable     Used in Equations
RCPO           29
TROLL Command:
```

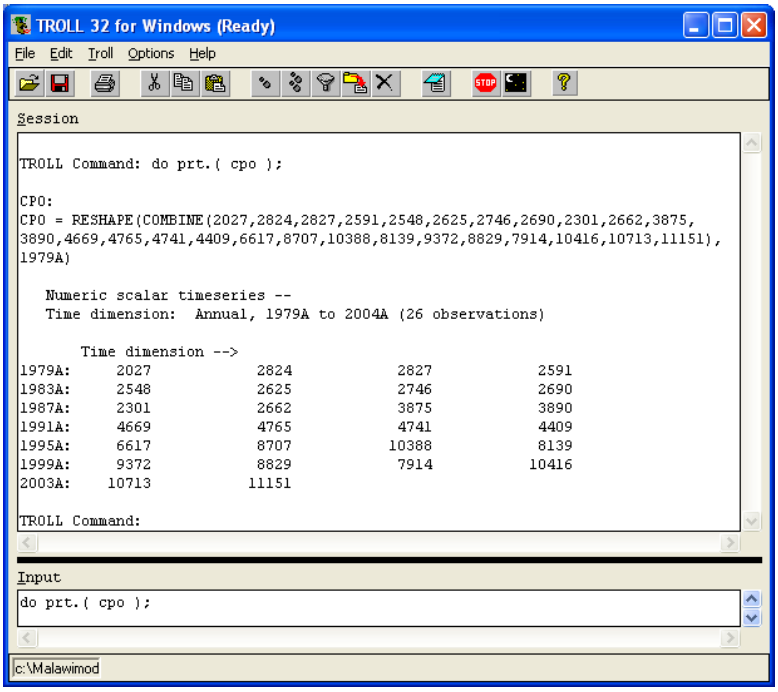
Table 8 Commands for printing equations

Command	Explanation
`prtmod eq 1;`	Prints equation 1
`prtmod eq 3 to 6;`	Prints equations 3, 4, 5 and 6
`prtmod eq 3 to 6 12 20 to 22;`	Prints equations 3, 4, 5, 6, 12, 20, 21, 22
`prtmod all;`	Prints the whole model, including symbolists, comments, functions used in the model, etc.

3.2.3 Printing data to the screen

Provided the “*input start;*” command have been executed, we have access to the default time series database MAIN.FRM. In Figure 8 we have demonstrated one (of many) way to print data to screen. The command “*do prt.(cpo);*” tells TROLL to print the first occurrence

Figure 8 The print command

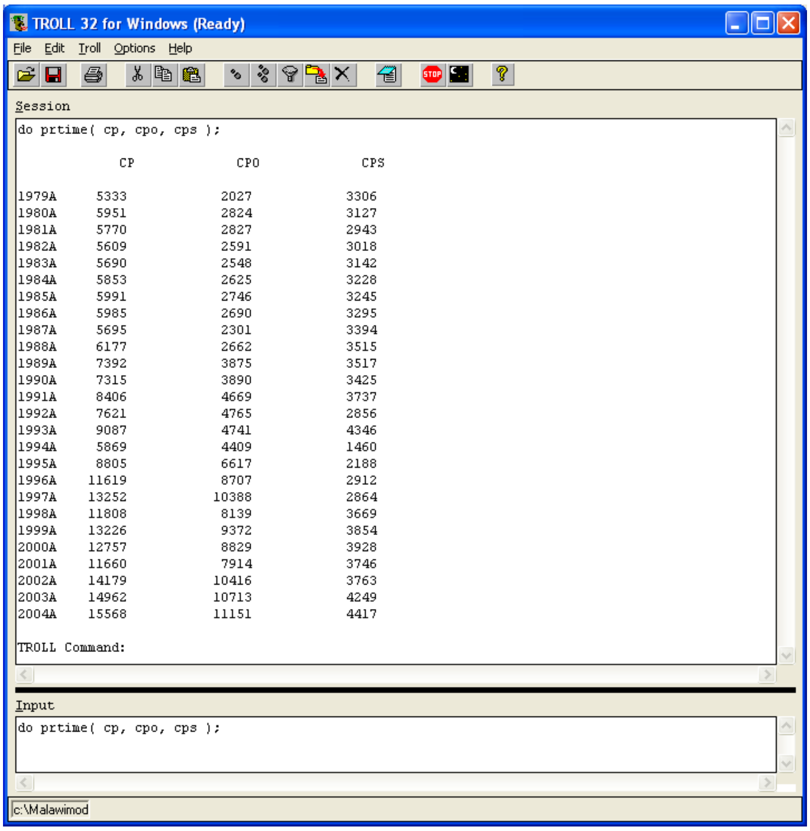


```
TROLL Command: do prt.( cpo );

CPO:
CPO = RESHAPE(COMBINE(2027,2824,2827,2591,2548,2625,2746,2690,2301,2662,3875,
3890,4669,4765,4741,4409,6617,8707,10388,8139,9372,8829,7914,10416,10713,11151),
1979A)

   Numeric scalar timeseries --
   Time dimension:  Annual, 1979A to 2004A (26 observations)

       Time dimension -->
1979A:     2027          2824          2827          2591
1983A:     2548          2625          2746          2690
1987A:     2301          2662          3875          3890
1991A:     4669          4765          4741          4409
1995A:     6617          8707         10388          8139
1999A:     9372          8829          7914         10416
2003A:    10713         11151

TROLL Command:
```

```
do prt.( cpo );
```

of the variable CPO (according to the search list). This command prints the timeseries data in columns, and also contain some metadata[1]. Perhaps more suitable for timeseries is the command "*do prtime (cp, cpo, cps);*, which will print the timeseries CP, CPO and CPS as in Figure 9.

Figure 9 Printing timeseries



```
do prtime( cp, cpo, cps );

              CP              CPO              CPS

1979A      5333              2027             3306
1980A      5951              2824             3127
1981A      5770              2827             2943
1982A      5609              2591             3018
1983A      5690              2548             3142
1984A      5853              2625             3228
1985A      5991              2746             3245
1986A      5985              2690             3295
1987A      5695              2301             3394
1988A      6177              2662             3515
1989A      7392              3875             3517
1990A      7315              3890             3425
1991A      8406              4669             3737
1992A      7621              4765             2856
1993A      9087              4741             4346
1994A      5869              4409             1460
1995A      8805              6617             2188
1996A     11619              8707             2912
1997A     13252             10388             2864
1998A     11808              8139             3669
1999A     13226              9372             3854
2000A     12757              8829             3928
2001A     11660              7914             3746
2002A     14179             10416             3763
2003A     14962             10713             4249
2004A     15568             11151             4417

TROLL Command:
```

```
do prtime( cp, cpo, cps );
```

A somewhat more advanced printing option is the `prtdset` command. This macro gives us a lot more alternatives. We can print the values and the growth rates of timeseries and we can print the difference (in level and/or in per cent) between variables from different databases (i.e. before and after a policy shift).

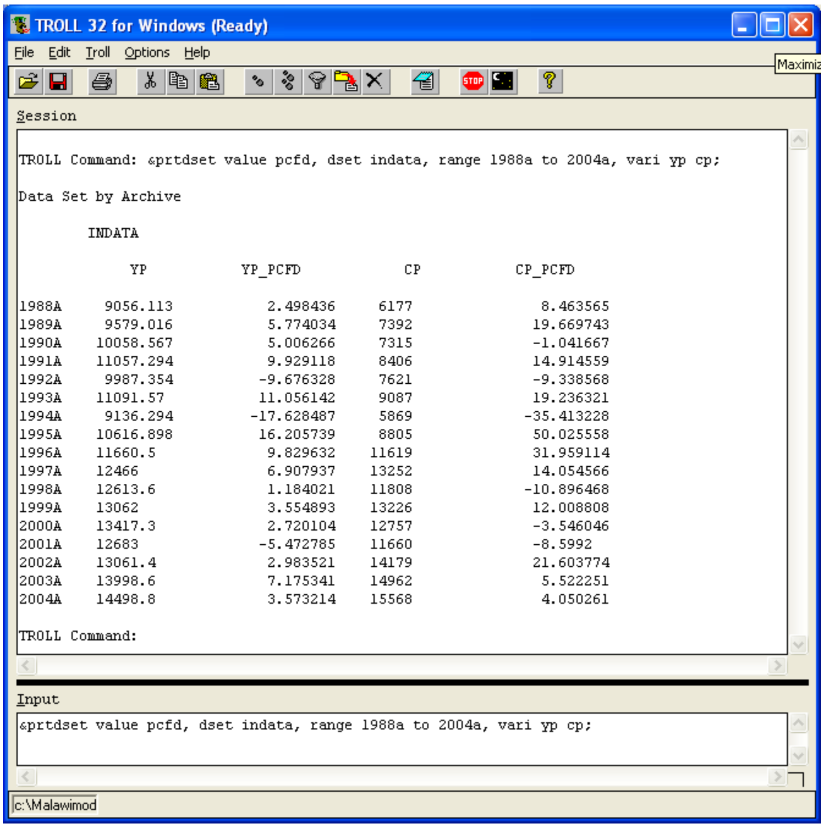
By typing

```
&prtdset value pcfd, dset indata, range 1988a to 2004a, vari yp cp;
```

we have the results as in Figure 10. Here we have specified to print the value (`value`) and the growth rates (`pcfd`) from the dataset (dset) `indata`, for the period (`range`) 1988 to 2004, of two variables (`vari`).

[1] Information about how the timeseries data has been created. In this example we see that using the functions reshape and combine together with the numbers has created the time series.

Figure 10 The prtdset command



```
TROLL Command: &prtdset value pcfd, dset indata, range 1988a to 2004a, vari yp cp;

Data Set by Archive

        INDATA

             YP            YP_PCFD          CP           CP_PCFD

1988A     9056.113          2.498436     6177             8.463565
1989A     9579.016          5.774034     7392            19.669743
1990A    10058.567          5.006266     7315            -1.041667
1991A    11057.294          9.929118     8406            14.914559
1992A     9987.354         -9.676328     7621            -9.338568
1993A    11091.57          11.056142     9087            19.236321
1994A     9136.294        -17.628487     5869           -35.413228
1995A    10616.898         16.205739     8805            50.025558
1996A    11660.5            9.829632    11619            31.959114
1997A    12466              6.907937    13252            14.054566
1998A    12613.6            1.184021    11808           -10.896468
1999A    13062              3.554893    13226            12.008808
2000A    13417.3            2.720104    12757            -3.546046
2001A    12683             -5.472785    11660            -8.5992
2002A    13061.4            2.983521    14179            21.603774
2003A    13998.6            7.175341    14962             5.522251
2004A    14498.8            3.573214    15568             4.050261

TROLL Command:
```

```
&prtdset value pcfd, dset indata, range 1988a to 2004a, vari yp cp;
```

By using the `oprtdset` instead of `prtdset` (just add a leading "`o`") we write the results to a file called TROLL.PRT that will be created on the Working Directory. It should be noted that by repeating this command the results would be appended to the TROLL.PRT file. A function to print the growth rate of a timeseries has been created by us. This is called `ypct` and gives the growth rate from the previous year. It is used as

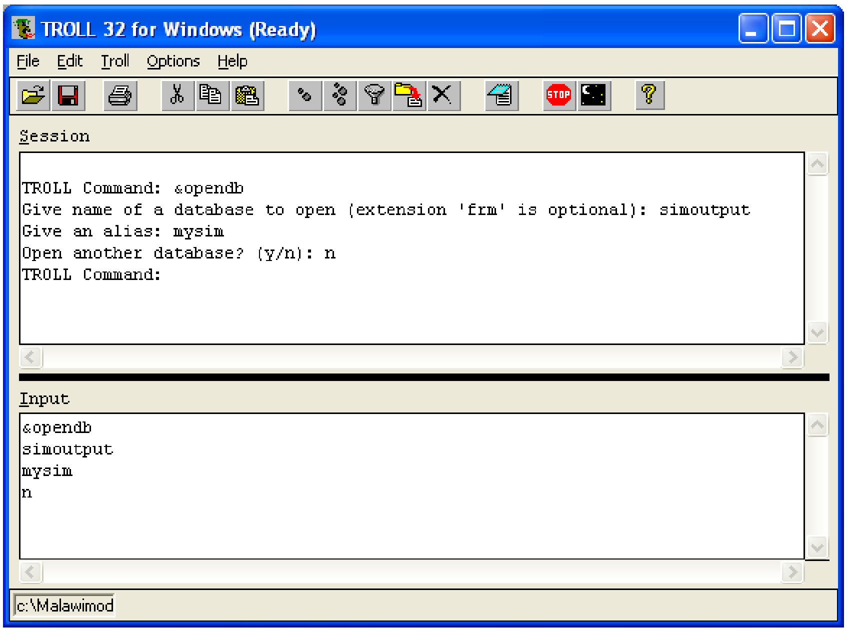
```
do prtime ( ypct'f( x ));
```

The suffix `'f` is needed to tell TROLL that `ypct` is a function and not a variable (with lag x).

3.2.4 Open other databases

In this chapter we have only considered the default database MAIN.FRM. Suppose we want to open another database, i.e. from a simulation we have carried out. Of course, we can always use the ordinary `access` and `search` commands within TROLL, but we have developed a small program to facilitate this task. The program is called OPENDB.PRG and its sole function is to open one or more databases (which must be located in the C:\MALAWIMOD\DATA directory). Figure 11 demonstrates the use of the program. It is invoked by the command *&opendb*, and it prompts for the name of the database to open. In this example the database we want to open is called SIMINPUT.FRM and as we can see the extension is optional. Next the program asks for the alias for the database. This is TROLL's internal name of the physical address, and could be whatever we like; here we have chosen to call it "`mysim`". Finally the program asks if we want to open another database, which we didn't, and typed `n` (for no).

Figure 11 Open a database



```
TROLL Command: &opendb
Give name of a database to open (extension 'frm' is optional): simoutput
Give an alias: mysim
Open another database? (y/n): n
TROLL Command:
```

```
&opendb
simoutput
mysim
n
```

Note that the `opendb` program adds entries to the access and search lists, and will not close already opened databases. The database opened is added to the bottom of the search list. This implies that, if the variables we want to access exist in a database that was already open (before we used `opendb)`, it will be read from that. We can always use the `alias` and an underscore to explicit tell TROLL which database to read from, i.e. the command

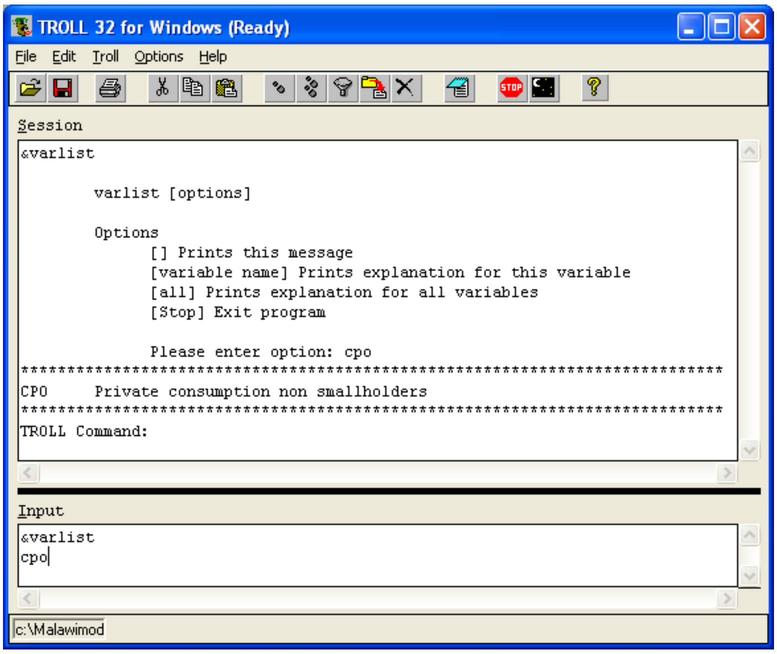
```
do prtime (mysim_CPO);
```

will read CPO from the database having the `alias "mysim"`.

3.2.5 Printing variable explanation to the screen

A model consists of many variables, and, particularly when new to the system, one doesn't know the meaning of every symbol. A program to print the symbols explanations to the screen has been developed. The program is demonstrated in Figure 12, and is executed by the command *&varlist*. It prints a list of options. We can type the variable, which explanation we want to print the to the screen, i.e. CPO as in this example, or we can type "`all`" to print the complete list.

Figure 12 Printing the symbol list



```
&varlist

        varlist [options]

        Options
              [] Prints this message
              [variable name] Prints explanation for this variable
              [all] Prints explanation for all variables
              [Stop] Exit program

              Please enter option: cpo
****************************************************************************
CPO     Private consumption non smallholders
****************************************************************************
TROLL Command:
```

```
&varlist
cpo
```

In this example we could just as well have typed

```
&varlist cpo
```

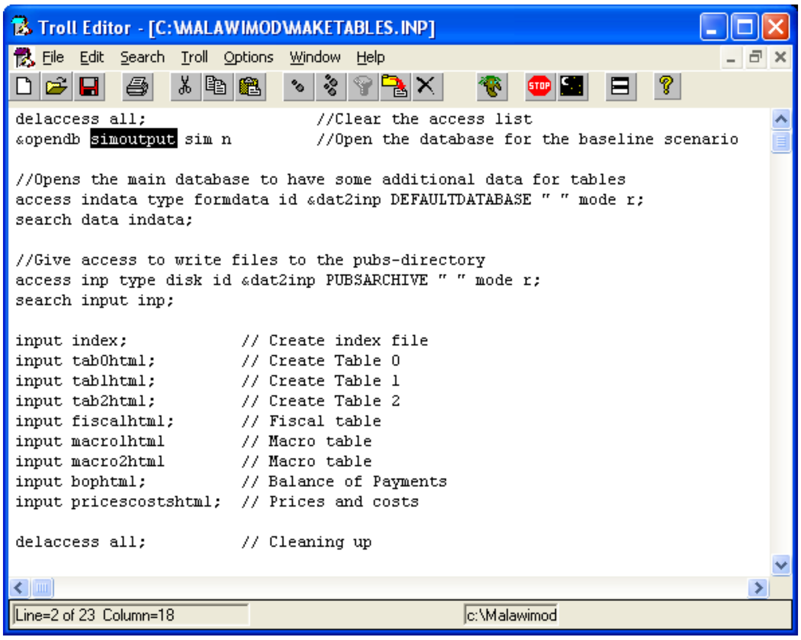
to produce the same result.

3.2.6 Creating tables in html format

Table programming is laborious work. The model system should be “finished”, or at least stable regarding changes and development. For this model system, that continuously undergoes changes, we have not put much emphasis into preparing routines for this important task. In any case, we have made a routine to write data from a simulation to tables.

The file MAKETABLES.INP must be used to create these tables. To use this we must specify which database to open for reading (see highlighted text in Figure 13). This file executes the files at the C:\MALAWIMOD\PUBS directory that will create the tables. An index file (INDEX.HTML) will be created at this directory, while the tables themselves will be stored at the C:\MALAWIMOD\PUBS\TABLES directory. By opening the INDEX.HTML in a browser we can click the links to the other tables to see them.

Figure 13 Creating tables


Troll Editor - [C:\MALAWIMOD\MAKETABLES.INP]
File Edit Search Troll Options Window Help
STOP
Line=2 of 23 Column=18
c:\Malawimod

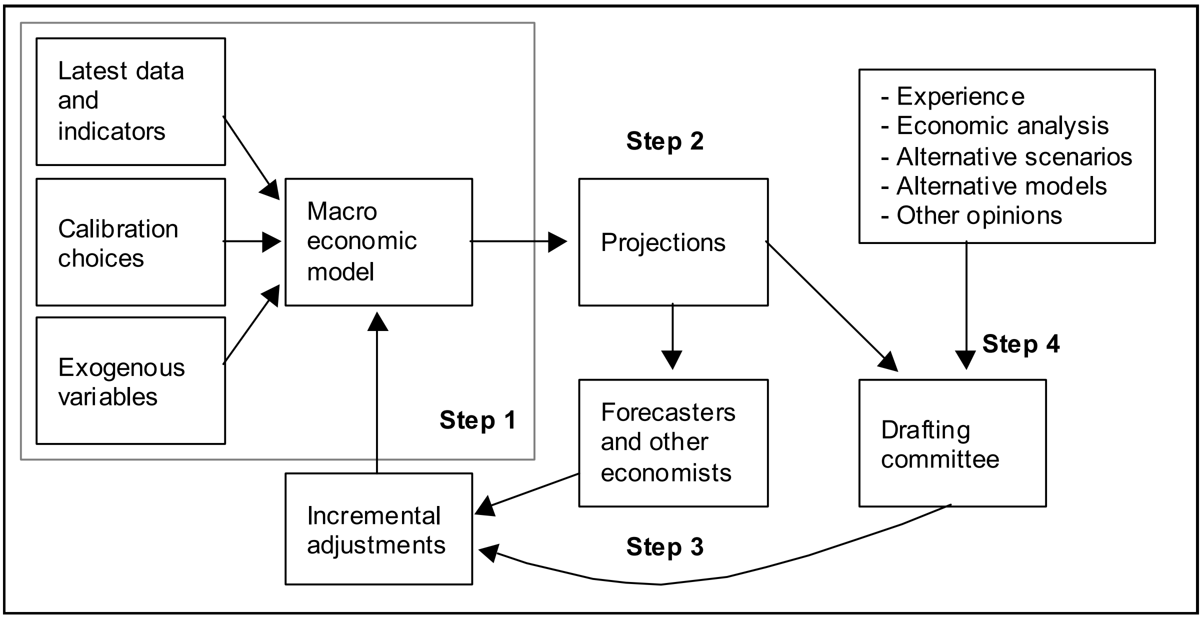
```
delaccess all;                  //Clear the access list
&opendb simoutput sim n         //Open the database for the baseline scenario

//Opens the main database to have some additional data for tables
access indata type formdata id &dat2inp DEFAULTDATABASE " " mode r;
search data indata;

//Give access to write files to the pubs-directory
access inp type disk id &dat2inp PUBSARCHIVE " " mode r;
search input inp;

input index;            // Create index file
input tab0html;         // Create Table 0
input tab1html;         // Create Table 1
input tab2html;         // Create Table 2
input fiscalhtml;       // Fiscal table
input macro1html        // Macro table
input macro2html        // Macro table
input bophtml;          // Balance of Payments
input pricescostshtml;  // Prices and costs

delaccess all;          // Cleaning up
```

4 The forecasting process

4.1 Introduction

The macro economic model system together with the staff of the organisations using it, are all parts of an effective system for creating forecasts and policy analysis. If the model system is used as a tool for expressing the institutions opinion on how the economy will evolve, the process can be described as in Figure 14 (Step 1). Together with the latest data available, we must make use of economic indicators, trends and tendencies to form our present view of the economy, and where it is likely to go. This will involve many data sources for input to the model. The model should also be calibrated, both against historical data, but also to reflect the understanding of the structure of the economy not captured in the model. To be able to make forecast we must make assumptions for the exogenous variables for the forecasting period. Exogenous variables in the model must be extrapolated before running the model. Projections for foreign variables, such as international prices and interest rates could be a consensus taken from other policy makers' forecasts. Key domestic fiscal assumptions can be taken from the Treasury's own estimates. In Step 2 the projections are evaluated. This preliminary examination by the forecasters always leads to adjustments to the exogenous variables, and are as a rule repeated "many times" before the projections are presented to a drafting committee (Step 3) for comments and suggestions. Step 4 is about employing alternative models or scenarios, either to overrule the results (i.e. due to known weaknesses of the model or uncertainty regarding important exogenous variables) or to present an alternative outcome for the economy that might be regarded just as probable.

Figure 14 The role of the forecasting system in projections



In this section we will explain the routines for extrapolating the models exogenous variables, and to implement our assumed values. In the model system this task is twofold; first we use an automatic and rough procedure, before we implement our detailed assumptions about future values for exogenous variables.

4.2 Extrapolating the exogenous variables

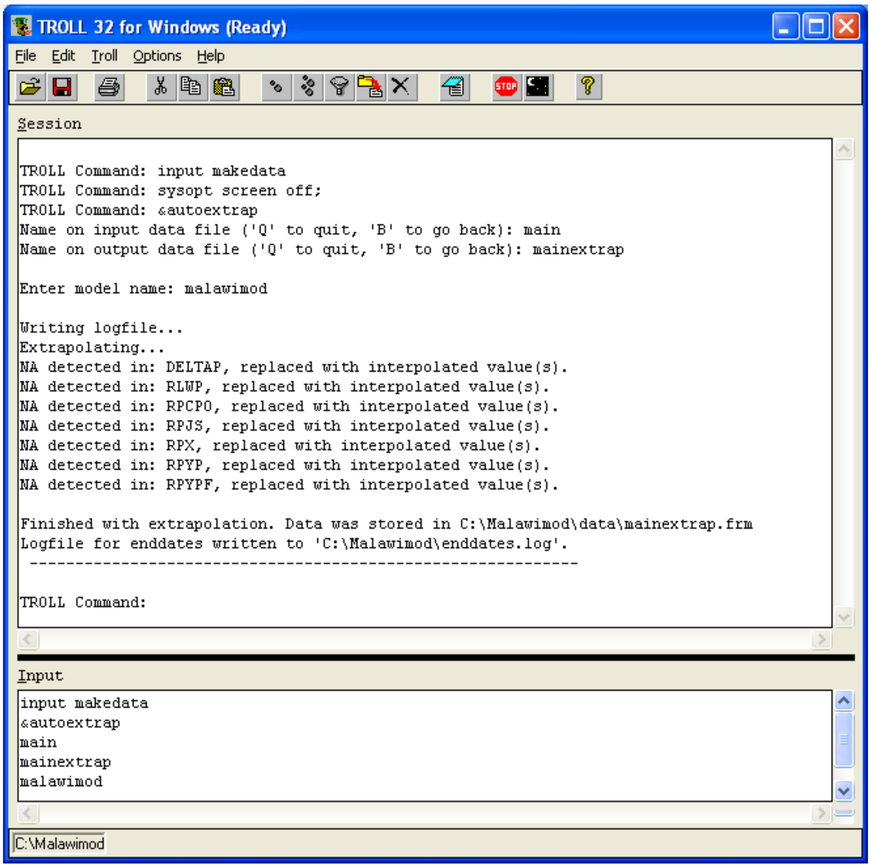
To simulate the model subsequent to the historical data, i.e. to create a baseline scenario, we must extrapolate the exogenous variables. If we disregard the lags in the variables, we can simulate the model as many years forward as we have extrapolated the exogenous variables.

4.2.1 Automatic extrapolation

To avoid making initial assumptions for all the exogenous variables (about 80) we have developed an automatic extrapolation routine. A demonstration of the program is given in Figure 15. After we have executed the batch file MAKEDATA.INP file, we execute the program with typing *&autoextrap*. It takes three parameters; an input data file, an output data file, and a model name. As we see from Figure 15 we want to perform an automatic extrapolation of the exogenous variables in the model `malawimod` from the database MAIN.FRM. We want to store the variables in the database MAINEXTRAP.FRM. All variables in MAIN.FRM will be copied to MAINEXTRAP.FRM before the extrapolation is done, leaving the input database MAIN.FRM unchanged. If we chose a database that already exists the program will print a warning and we will be given the options to overwrite or give a new name. It is not allowed to write the extrapolated variables back to the input database.

Note that the extrapolation program reports variables where there are NA's (Not-Available, i.e., data type is unspecified). If there are NA's in the input time series they will be replaced using growth rates to interpolate. The program writes a log file called ENDDATES.LOG to the Working Directory. This file shows the end dates of the variables (classified by exogenous and endogenous) before the extrapolation.

Figure 15 Automatic extrapolation of exogenous variables



```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: &autoextrap
Name on input data file ('Q' to quit, 'B' to go back): main
Name on output data file ('Q' to quit, 'B' to go back): mainextrap

Enter model name: malawimod

Writing logfile...
Extrapolating...
NA detected in: DELTAP, replaced with interpolated value(s).
NA detected in: RLWP, replaced with interpolated value(s).
NA detected in: RPCPO, replaced with interpolated value(s).
NA detected in: RPJS, replaced with interpolated value(s).
NA detected in: RPX, replaced with interpolated value(s).
NA detected in: RPYP, replaced with interpolated value(s).
NA detected in: RPYPF, replaced with interpolated value(s).

Finished with extrapolation. Data was stored in C:\Malawimod\data\mainextrap.frm
Logfile for enddates written to 'C:\Malawimod\enddates.log'.
 -------------------------------------------------------------

TROLL Command:
```

```
input makedata
&autoextrap
main
mainextrap
malawimod
```

The automatic extrapolation routine uses Holt's method, or double exponential smoothing. This is useful in short term forecasting for data with constant or no-constant trend and without a seasonal pattern. What the program does, in brief, is first to read the historical values for all exogenous variables in a model, then calculates a level and a trend component at each period based on historic values, and uses this pattern to extrapolate the timeseries. It uses two weights, or smoothing parameters, to update the components at each period. The double exponential smoothing equations are:

$$L_t = \alpha Y_t + (1-\alpha)\left(L_{t-1} + T_{t-1}\right)$$

$$T_t = \beta\left(L_t - L_{t-1}\right) + (1-\beta)T_{t-1}$$

$$Y_{t\ \text{(smoothed)}} = L_{t-1} + T_{t-1}$$

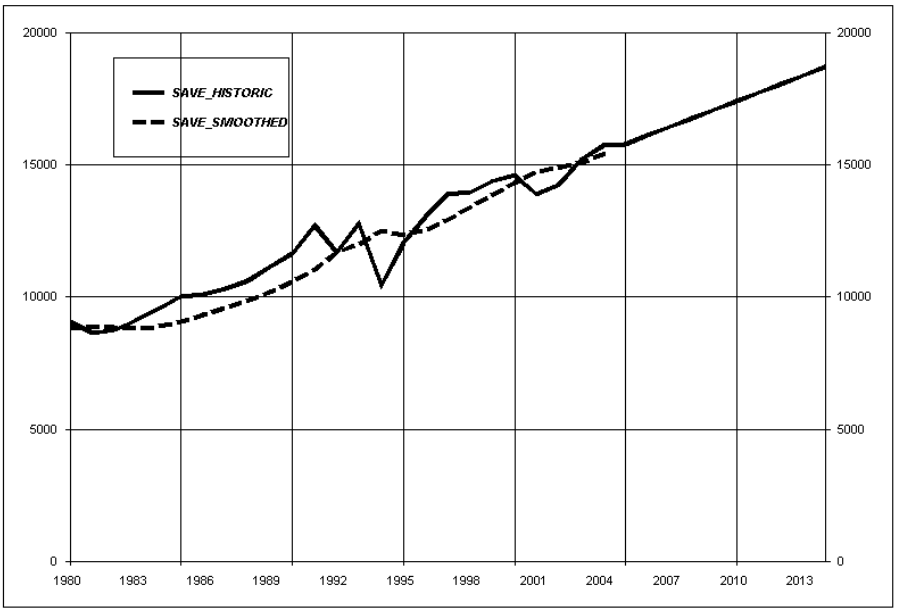
$$Y_{t+n\ \text{(forecast)}} = L_t + nT_t$$

L_t	level at time t, α is the weight for the level
T_t	trend at time t, β is the weight for the trend
Y_t	data value at time t
$Y_{t\ \text{(smoothed)}}$	fitted value, or one-step-ahead forecast, at time t
$Y_{t+n\ \text{(forecast)}}$	n-step-ahead forecast, at time t
n	number of periods to forecast

If the first observation is numbered one, then level and trend estimates at time zero must be initialised in order to proceed. The initialisation method used to determine how the smoothed values are obtained in done in a simple manner, we simply say that $L_1 = Y_1$ and that $T_1 = 0$, i.e. that the initial level is equal to the actual value, and that there is no trend initially[2]. The smoothing parameters α and β are both fixed at 0.2. To alter their values we must change in the PROFILE.SRC file.

In Figure 16 we have graphed the outcome of the smoothing method when applied on real GDP. The solid line shows the historic values from 1980 to 2004 and the extrapolated values appended until 2014. The dashed line shows the result from smoothing the history as well.

Figure 16 Historic, smoothed and extrapolated values

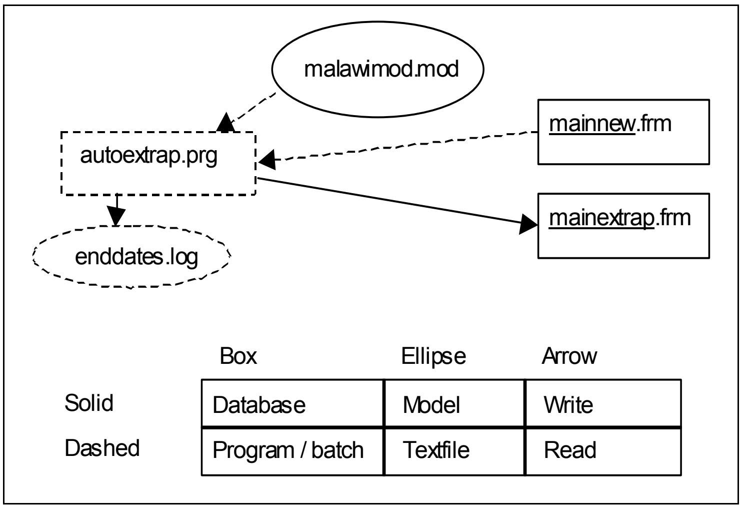


It is important to be aware of that for some exogenous variables we do not want this extrapolation method. For dummy variables (used in some econometric equations) we want a fixed value ahead in time, usually zero. The same might be the case for residuals. Residuals in econometric equations are assumed to have an expected value of zero if they are additive and one if they are multiplicative. Also for other residuals we might want to extrapolate them with a fixed, controllable value. This must be taken care of in the process of implementing our exogenous assumptions.

In Figure 17 we show a flowchart of this routine. As we can see, the program AUTOEXTRAP.PRG, which perform the automatic extrapolation for all exogenous variables in a model, reads the list of exogenous variables from the model. It then reads the timeseries for all these variables, calculates a level and a trend, which is used in the extrapolation, and then write the results to a new database. Note that the new database MAINEXTRAP.FRM is copied from input database MAINNEW.FRM before the exogenous variables are extrapolated. In this way we also bring with us the endogenous variables (which is needed as starting values for the simulation).

[2] An improvement of the initialisation procedure would be to fit a linear regression model to the input time series variable (Y) versus time, and let the constant term from this regression be used as the initial estimate of the level component, and the slope coefficient as initial estimate of the trend component.

Figure 17 Flowchart for automatic extrapolation



	Box	Ellipse	Arrow
Solid	Database	Model	Write
Dashed	Program / batch	Textfile	Read

Summary

• input makedata	// Give search and access to the program directory
• &autoextrap	// Execute program to extrapolate data
• mainnew	// Input database (optional name)
• mainextrap	// Output database (optional name)
• malawimod	// Name of model to extrapolate variables from

4.2.2 Implementing assumptions about the future

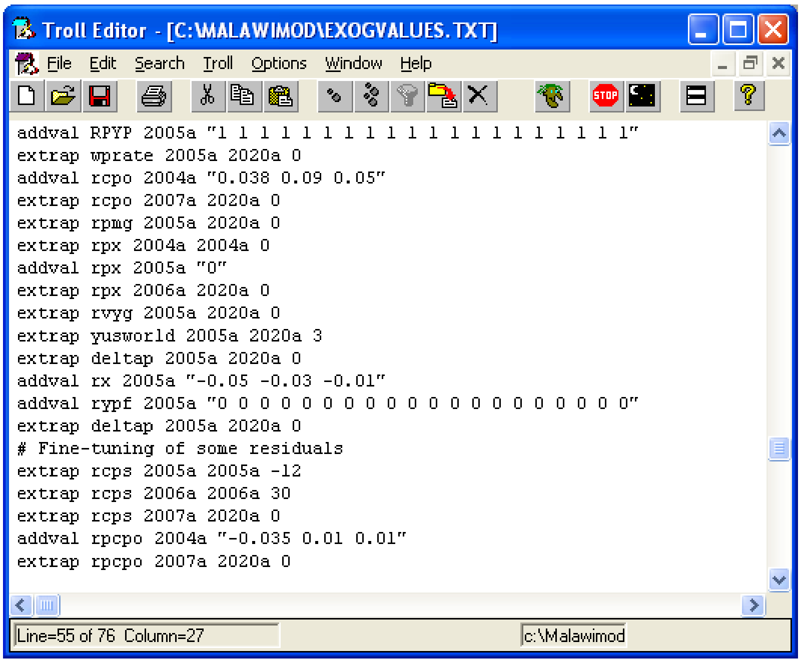
Having extrapolated all exogenous variables in the model in a mechanical way, we can now implement their details. There are a lot of ways we can express our detailed assumptions for variables forward in time. We might want, for instance, to set a variable at a fixed level, or to give it a specified growth rate for the future. Or, maybe we want to change the level by some percentage, or to change the growth rate. To facilitate suchlike implementations of our assumptions, some methods have been developed. It is important to realize exactly how these methods are used and what they do. In Table 9 we give a brief overview of the methods, leaving a more thoroughly examination for Appendix 1.

Table 9 Methods for adjusting timeseries

Function	Comment
ADDVAL	Add values to time series
ADJUST	Adjust time series by per cent and a constant added
EXTRAP	Adjust time series to a given growth rate
EXTRAPA	Adjust time series to a given growth rate and a constant added
PCFDADJ	Adjust time series by changing the growth rate by a constant
#	Symbol for comments

The way we put our detailed assumptions into operation is through an ordinary text file, which, in turn will be read by a TROLL program. In Figure 18 we show an example of such a text file called EXOGVALUES.TXT. The file must be in a particular format. The first "word" is a keyword

Figure 18 Text file with assumptions for exogenous variables


Troll Editor - [C:\MALAWIMOD\EXOGVALUES.TXT]
File Edit Search Troll Options Window Help
STOP
Line=55 of 76 Column=27
c:\Malawimod

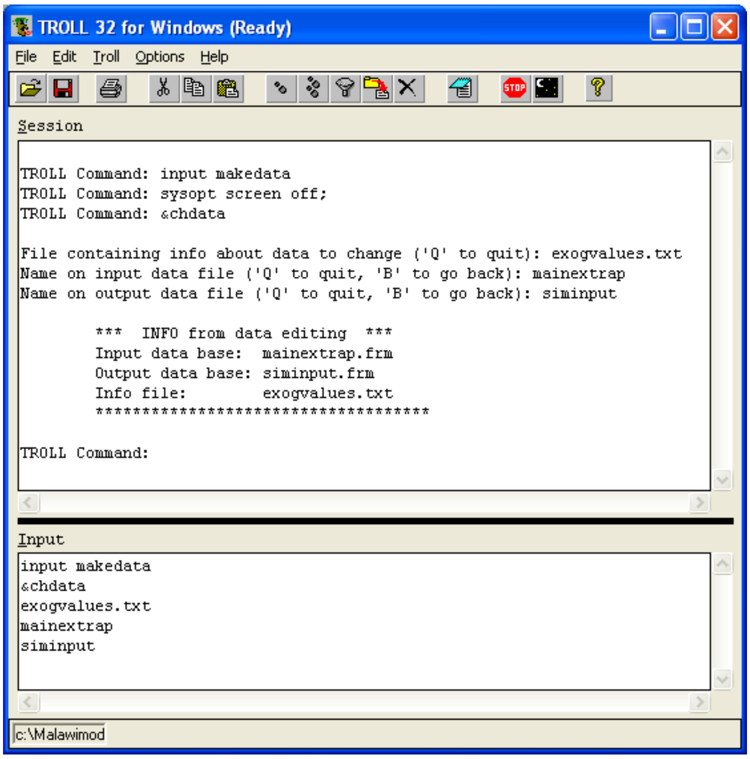
```
addval RPYP 2005a "1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1"
extrap wprate 2005a 2020a 0
addval rcpo 2004a "0.038 0.09 0.05"
extrap rcpo 2007a 2020a 0
extrap rpmg 2005a 2020a 0
extrap rpx 2004a 2004a 0
addval rpx 2005a "0"
extrap rpx 2006a 2020a 0
extrap rvyg 2005a 2020a 0
extrap yusworld 2005a 2020a 3
extrap deltap 2005a 2020a 0
addval rx 2005a "-0.05 -0.03 -0.01"
addval rypf 2005a "0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0"
extrap deltap 2005a 2020a 0
# Fine-tuning of some residuals
extrap rcps 2005a 2005a -12
extrap rcps 2006a 2006a 30
extrap rcps 2007a 2020a 0
addval rpcpo 2004a "-0.035 0.01 0.01"
extrap rpcpo 2007a 2020a 0
```

determining what function will be used to adjust the data. The next statement is the variable name whose value we want to change. Subsequently follows the date, indicating the first observation in the time series to change. Then, depending on what function is used follows either a sequence with numbers to put into the time series, implicit determining the end date of data manipulation, or a specific end date and information about how to change the data. We will come back to these details in Appendix 1. Note that it is possible to insert a comment in the text file. A comment should begin with a number (hash) symbol (#).

When we have given all the information about how to manipulate the time series we are ready to add them into the database. The program we have made to perform this is called CHDATA.PRG. In Figure 19 we show how this is done. After we have given access and search to the directory containing the programs by running the input file MAKEDATA.INP, we execute the program with the statement *&chdata.* First we are prompted for the text file containing our assumptions. In this example this is EXOGVALUES.TXT. This filename (including the extension) is optional and that is why we must type the extension as well[3]. After that the program asks for the name of the database to read data from, which should be a database where the variables have already been extrapolated. In Figure 19 the database MAINEXTRAP.FRM is the database we created with the extrapolation procedure (c.f. Figure 15). Next we must give the name of the database to write to. This database will be created. In the example in Figure 19 we chose to create and store our data in SIMINPUT.FRM. If we chose a database that already exists the program will print a warning and we will be given the options to overwrite or give a new name. In the present version of CHDATA.PRG it is not possible to write back into the database we are reading from. The program finally prints the name of the input and output databases, and the text file used for adjusting the data.

[3] For databases we only use the type "`Formdata`" (extension "`.` frm") which is hard coded in the system and consequently we don't need to provide it. The same applies for the model (".mod") and spreadsheets ("wks").

Figure 19 Implementing details about exogenous variables

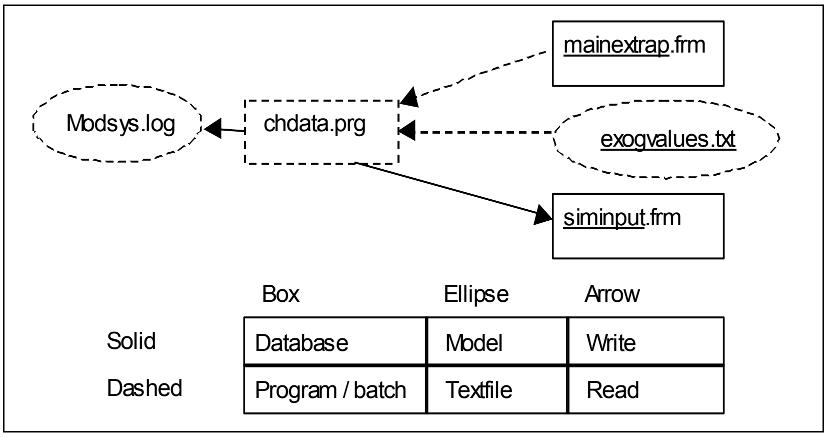


```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: &chdata

File containing info about data to change ('Q' to quit): exogvalues.txt
Name on input data file ('Q' to quit, 'B' to go back): mainextrap
Name on output data file ('Q' to quit, 'B' to go back): siminput

        ***  INFO from data editing  ***
        Input data base:  mainextrap.frm
        Output data base:  siminput.frm
        Info file:        exogvalues.txt
        ************************************

TROLL Command:
```

```
input makedata
&chdata
exogvalues.txt
mainextrap
siminput
```

In Figure 20 we take a look at the dataflow for this process. We see that the program used for this purpose (CHDATA.PRG) reads information about which variables, for what period and in what way, to change them, from the text file EXOGVALUES.TXT. Next it reads the data for variables to be changed, change them accordingly, and write the results to a new database (here called SIMINPUT.FRM). Note that all data in the input database (MAINEXTRAP.FRM) are transferred to the output database (SIMINPUT.FRM), not only those we are adjusting. The program also writes a log file from the extrapolation, which merely reports if each method (each line in the EXOGVALUES.TXT file) turned out OK or not. This log file is called MODSYS.LOG and is written to the Working Directory. Note that every time we use the programs `chdata` or `shift` to adjust data a line is appended to this log file.

Figure 20 Dataflow for implementing exogenous assumptions



	Box	Ellipse	Arrow
Solid	Database	Model	Write
Dashed	Program / batch	Textfile	Read

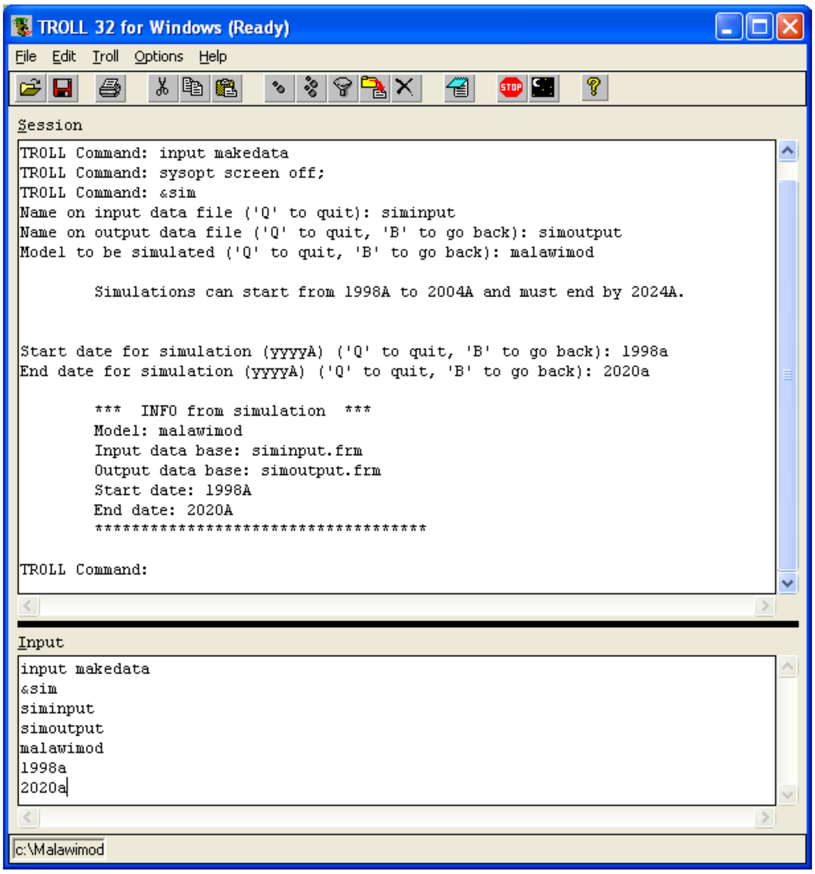
Summary

- input makedata // Search and access to programs
- &chdata // Program to manipulate data
- exogvalues.txt // Text file containing exogenous assumptions (optional name)
- mainextrap // Database to read from (optional name)
- siminput // Database to write to (optional name)

4.3 Simulation

When we have installed our assumptions for the exogenous variables we can simulate the model to solve the endogenous variables. The program we have written to perform this task is called SIM.PRG. In Figure 21 we show how the program is used to simulate the `malawimod` model using the database with our assumptions defined in the previous section. First we give access and search to the directory containing the programs

Figure 21 Using the sim.prg program to simulate a model



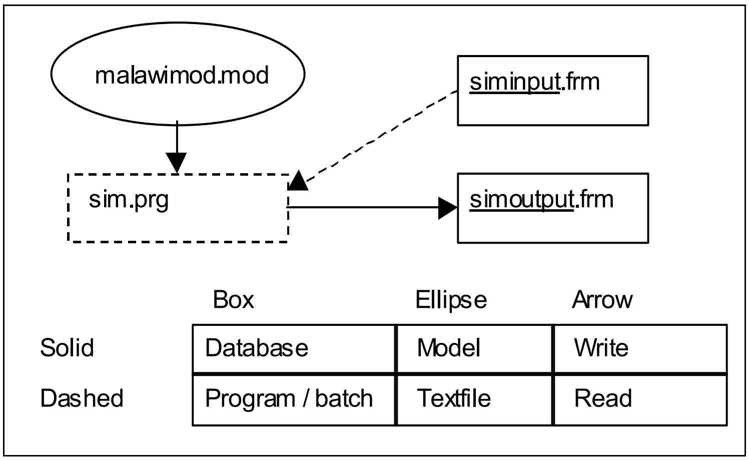
```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: &sim
Name on input data file ('Q' to quit): siminput
Name on output data file ('Q' to quit, 'B' to go back): simoutput
Model to be simulated ('Q' to quit, 'B' to go back): malawimod

        Simulations can start from 1998A to 2004A and must end by 2024A.


Start date for simulation (yyyyA) ('Q' to quit, 'B' to go back): 1998a
End date for simulation (yyyyA) ('Q' to quit, 'B' to go back): 2020a

        ***  INFO from simulation  ***
        Model: malawimod
        Input data base: siminput.frm
        Output data base: simoutput.frm
        Start date: 1998A
        End date: 2020A
        ************************************

TROLL Command:
```

```
input makedata
&sim
siminput
simoutput
malawimod
1998a
2020a
```

by running the input file MAKEDATA.INP. Then we execute the simulation program with the instruction *&sim* and are prompted for a database to read data from (input data base). This is the database where we have stored our assumptions for exogenous variables (`siminput`). Next we are prompted for a database to store our simulation results, this is an optional name and here we chose to call it *simoutput*. Then we must specify which model we want to simulate. With this information TROLL checks the model for possible simulation dates. In the example in Figure 21 the simulation start date can be between 1998 and 2004, and the end date for simulation must be no later than 2024. We then type in start and end dates (a common mistake that will result in an error is to forget the ‘a’ after the year when providing the dates), and TROLL solves the models endogenous variables. Both endogenous and exogenous variables are saved in the SIMOUTPUT.FRM database. Finally the simulation program prints information about the model, databases and dates used for this simulation.

The simulation program (SIM.PRG) creates an input file after each simulation called LASTSIM.INP. This file gives readable access to the model and the output database from the latest simulation. By running this file, typing *input lastsim;* we have easy access to the simulation output database and the model used for the latest simulation.

Figure 22 Dataflow of the simulation process



Summary

• input makedata	// Search and access to programs
• &sim	// Program for simulation
• siminput	// Simulation input database (optional name)
• simoutput	// Simulation output database (optional name)
• malawimod	// Name of model
• 1998a	// Simulation start date
• 2020a	// Simulation end date

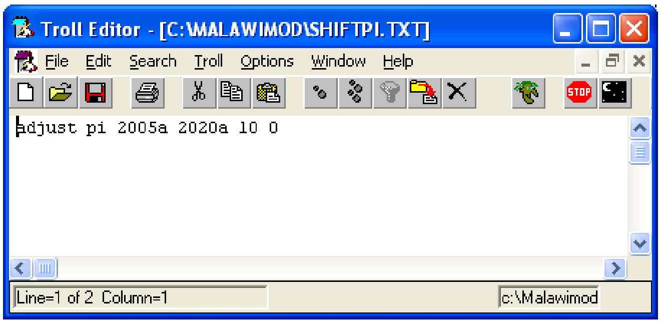
5 Policy analysis

When we have made a simulation, which may be our baseline forecast for the Malawian economy, we might be interested in studying the effects of various policy changes. In practice this means that we change some of our assumptions and simulate the model for this new set of exogenous input variables. These two scenarios are often referred to as the baseline (or reference) scenario and the alternative scenario respectively.

5.1 Implementing a shift

First we will go through the process of preparing the shift. After we have decided on what shift we would like to simulate, we must implement it as new assumptions for one or more variables. We start by creating a new text file where we specify the details about which variables to change, and in what way to change them. An example on such a text file is showed in Figure 23. Here the file is called SHIFTPI.TXT and consists of just one line. The information in this line tells that we

Figure 23 Information to alternative scenario


Troll Editor - [C:\MALAWIMOD\SHIFTPI.TXT]
File Edit Search Troll Options Window Help
Line=1 of 2 Column=1
c:\Malawimod

```
adjust pi 2005a 2020a 10 0
```

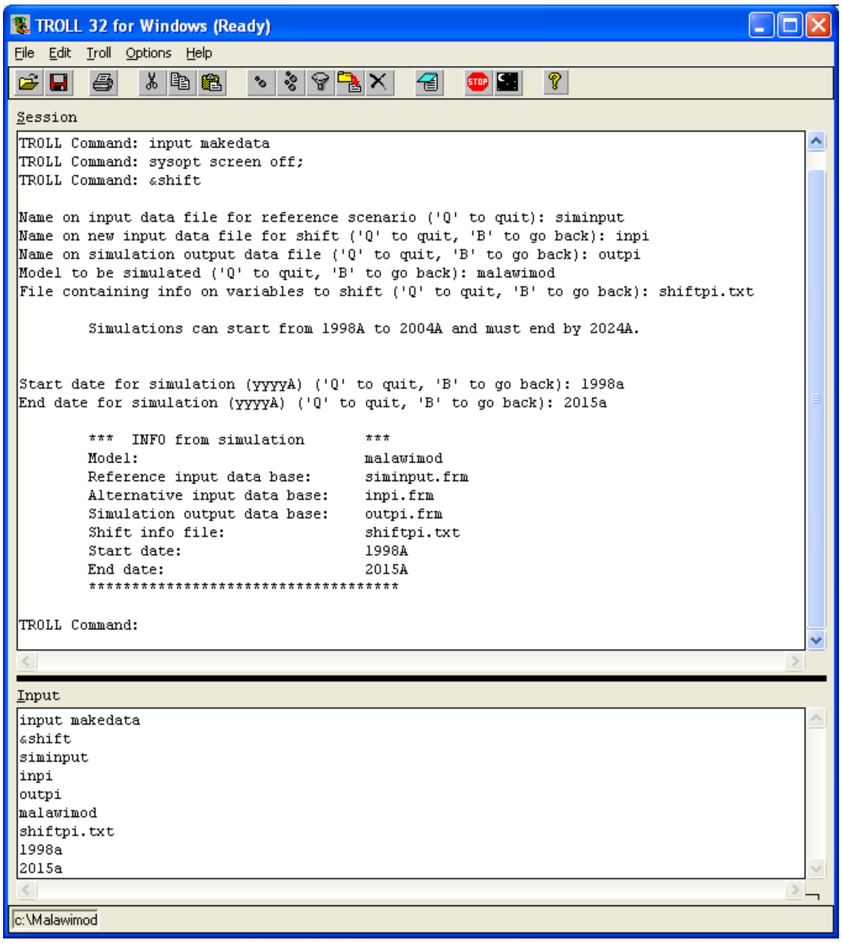
want to adjust the import price (PI) by 10 per cent each year for the period 2005 to 2020. Note the final 0 in this line. This is because is possible to add a level for each period as well. The instructions in this line will increase the level of PI by 10 percent relative to the existing values. If the shift we want to do is relative to our baseline scenario, and that PI has the value 10 in the baseline, then the input of PI to the policy shift will be 11.

5.2 Simulation of the policy shift

The program we have written for carrying out a policy shift is called SHIFT.PRG. A shift is always relative to another exogenous input dataset to the model. The SHIFT.PRG program, as seen in Figure 24, therefore first prompt for the name on the input data file for the database we want to compare against, for instance our baseline scenario. This means that the program reads the input database that was used for the simulation of the baseline scenario, here called SIMINPUT.FRM. This is the input dataset we will shift (change). Next the program prompts for a name to give this new input dataset (here INPI.FRM), which will be the same as the input for SIMINPUT.FRM except for the variable PI. Then we are asked to give a name for the database to store the simulation results in. We call this database OUTPI.FRM. Next we must provide the name of the model to simulate and the name of the text file (SHIFTPI.TXT) where we have specified which variable we want to change. Given this information, the possible simulation dates are evaluated and printed to the screen, and we must specify the start date and the en date for the simulation. The program now performs its tasks. First it copies all the variables in the SIMINPUT.FRM database to INPI.FRM, then it changes the variable specified according to the information in the text file SHIFTPI.TXT (here PI). Then it uses the INPI.FRM database as input to the model and carries out the simulation. It stores the simulation results (and the exogenous variables) in the OUTPI.FRM database. Now, all differences between the output from this shift (OUTPI.FRM) and the baseline (SIMOUTPUT.FRM) could be attributed to adjusting the exogenous variable PI, and consequently is the effect of a shift in import price according to our model.

It should be stressed that the only difference in exogenous assumptions in the input datasets SIMINPUT.FRM (for creating the baseline scenario) and INPI.FRM (for creating the alternative scenario) is PI. The differences in the simulation results (SIMOUTPUT.FRM and OUTPI.FRM) are many, but are all due to adjusting our assumptions for PI.

Figure 24 Using the shift.prg program to perform an alternative simulation

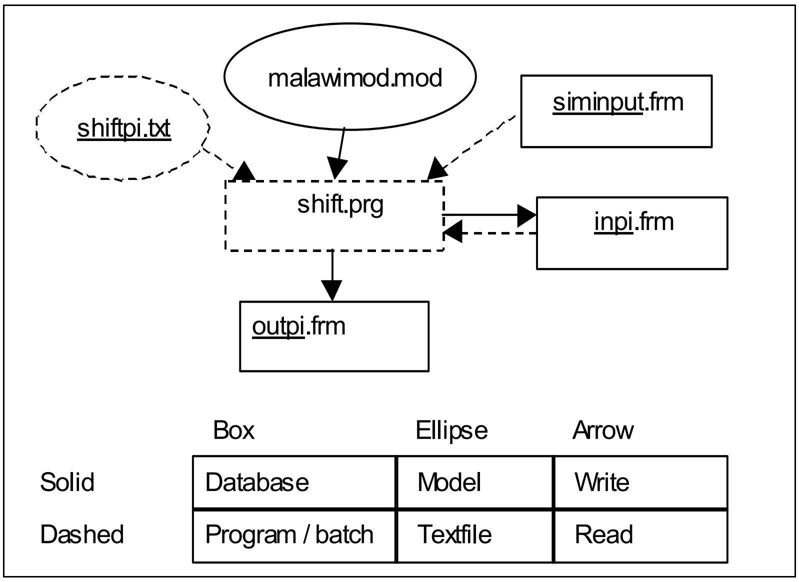


```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: &shift

Name on input data file for reference scenario ('Q' to quit): siminput
Name on new input data file for shift ('Q' to quit, 'B' to go back): inpi
Name on simulation output data file ('Q' to quit, 'B' to go back): outpi
Model to be simulated ('Q' to quit, 'B' to go back): malawimod
File containing info on variables to shift ('Q' to quit, 'B' to go back): shiftpi.txt

        Simulations can start from 1998A to 2004A and must end by 2024A.


Start date for simulation (yyyyA) ('Q' to quit, 'B' to go back): 1998a
End date for simulation (yyyyA) ('Q' to quit, 'B' to go back): 2015a

        ***  INFO from simulation       ***
        Model:                          malawimod
        Reference input data base:      siminput.frm
        Alternative input data base:    inpi.frm
        Simulation output data base:    outpi.frm
        Shift info file:                shiftpi.txt
        Start date:                     1998A
        End date:                       2015A
        ************************************

TROLL Command:
```

```
input makedata
&shift
siminput
inpi
outpi
malawimod
shiftpi.txt
1998a
2015a
```

The dataflow from the process of doing a policy shift is showed in Figure 25. We see that the shift program reads information from the text file SHIFTPI.TXT, reads data from SIMINPUT.FRM, and writes this information to INPI.FRM. Then it closes the SIMINPUT.FRM database and changes the variable PI in INPI.FRM. Next it uses this database to simulate the model, and writes the results to the OUTPI.FRM database.

Figure 25 Dataflow of a policy shift



	Box	Ellipse	Arrow
Solid	Database	Model	Write
Dashed	Program / batch	Textfile	Read

Summary

input makedata	// Search and access to programs
&shift	// Program for shift
siminput	// Input data file for reference scenario
inpi	// New input data file for shift (optional name)
outpi	// Simulation output data file (optional name)
malawimod	// Model to be simulated
shiftpi.txt	// File containing info on variables to shift (optional name)
1998a	// Start date for simulation
2015a	// End date for simulation

6 Installing and editing the model

6.1 Installing the model

In this section we will describe how to install and update the model. In TROLL there are many ways to create and manipulate a model. It can be done interactively through TROLL's input-window or through files that can be stored and edited. We have chosen to do this by editing an input file. This file is called MAKEMODEL.INP, and contains the entire model in text format. By running this file the model is installed in TROLL's own format for models.

To install the model (if the MAKEMODEL.INP file is open in the editor) we can just click the icon showing the two footprints, meaning execute the entire file. It is also possible to achieve the same by typing *input makemodel;* in TROLL's input window to execute the file. It is important to note that by executing this file, the previous version of the model is overwritten if it exists. We should consider creating a backup of the old model using a file manager before installing a new one.

The MAKEMODEL.INP has comments describing what is going on. In Table 10 is a brief explanation of the various commands in the file.

Table 10 Commands in the MAKEMODEL.INP file

Command	Explanation
sysopt screen off;	Turn off writing to screen
do hfdelete(modelarchive||model||".mod");	Delete existing model if any
access mod type disk id &dat2inp MODELARCHIVE " " mode w;	Opens databases and adds them to the list of accessed databases. In our case it opens the C:\MALAWIMOD\MODEL directory for writing
search first model mod w;	Adds the model directory to the SEARCH list.
usemod &dat2inp MODEL " " ;	Retrieves the specified MODEL file and establishes it as the current working model. In this case this command is equal to *usemod malawimod*;
addsym endogenous	Declares new symbols by symbol type. At the time of writing it is only ENDOGENOUS symbols that have to be declared. The EXOGENOUS are default and does not have to be specified. The COEFFICIENT is omitted because they are hard coded into the equations.
addeq bottom	Enters one or more equations written in the modelling language into the equation list of the model. The equation is put after the position specified. In this case "bottom" means that the new equations always are added at the bottom.
delsym nowarn all;	Deletes the specified symbols from the symbol table of the model, provided they do not appear in any equations. The option "nowarn" means we suppress any warnings.
lkord;	Prints a summary giving the number of simultaneous blocks and the number of equations in the largest blocks.
filemod &dat2inp MODEL " " ;	Stores the model we are editing as a permanent disk file and terminates the MODEDIT task. In this case this command is equal to *filemod malawimod*;
sysopt screen on;	Enables writing to the screen.

6.2 Editing the model

In general, to change the model we open the MAKEMODEL.INP file in the TROLL editor and do whatever changes we like, and save the file. By running the file (*input makemodel;*) we implement the changes to the model.

6.2.1 Changing an equation

Suppose we want to change an equation in the model. For example if we have re-estimated the consumption function and wants to replace the old equation with the new one. To accomplish this we localise the equation we want to replace in the MAKEMODEL.INP file, delete it and type in the new equation. Next we save and run the file by typing *input makemodel;* in the input-window as described above.

6.2.2 Adding an equation

If we want to add a new equation we type it into the MAKEMODEL.INP text file. The file aims at grouping equations belonging to the same model block, so we should try to put it where it naturally belongs. When we add a new equation we must remember to declare the endogenous variable in the list following the *addsym endogenous* statement. Next we save and run the file.

6.2.3 Deleting an equation

Deleting an equation is the opposite of adding one. We delete the equation from the MAKEMODEL.INP file and remove the symbol for the endogenous variable solved in that equation from the endogenous list (following the *addsym endogenous* statement). Next we save and run the file.

6.2.4 Miscellaneous

The same procedures as described for changing, adding and deleting equations applies to changing more than one equation.

It is important to be aware of that some of the equations are also used in generating data. For instance the timeseries for total government revenue (GRTOT) is installed into the database by stating that this is equal to the sum of government revenue (GREV) and transfers to the government from abroad (TRFG) as `grtot = grev + trfg` in the ADDDATA.INP file. If we want to e.g. disaggregate this expression in the model by replacing the equation with another, we must remember to change accordingly in the ADDDATA.INP file as well.

If we change an econometric equation we must recalculate the residual. Suppose we wanted to change the econometric equation for private consumption in MAKEMODEL.INP from

```
LOG(CPO/CPO(-1)) = 0.0536+0.9848*LOG(YHDR/YHDR(-1))-0.2869*LOG(CPO(-
1)/YHDR(-1))-0.6002*DUM94+RCPO
```

to for instance

```
LOG(CPO/CPO(-1)) = 0.08+0.96*LOG(YHDR/YHDR(-1))-0.25*LOG(CPO(-1)/YHDR(-
1))+RCPO
```

where we removed the dummy-variable and altered the coefficient somewhat. Then we must remember to change accordingly from

```
RCPO = LOG(CPO/CPO(-1)) - (0.0536+0.9848*LOG(YHDR/YHDR(-1))-
0.2869*LOG(CPO(-1)/YHDR(-1))-0.6002*DUM94)
```

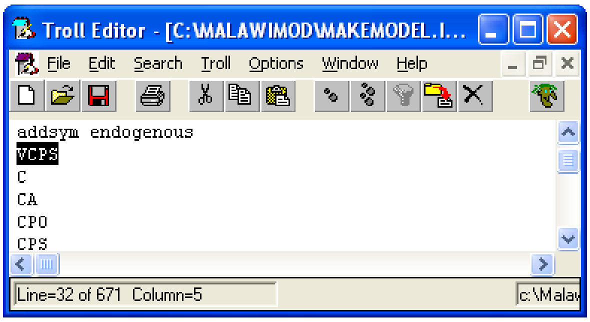
to

```
RCPO = LOG(CPO/CPO(-1)) - (0.08+0.96*LOG(YHDR/YHDR(-1))-0.25*LOG(CPO(-
1)/YHDR(-1)))
```

in ADDDATA.INP when calculating the values for the residual.

Generally, when adding new variables to the model we must also add their data. For new exogenous variables we must add their historic data in the ADDDATA.INP so that they will be stored in the MAIN.FRM database. But also endogenous variables, even though the model calculates them, need historic values in the database. This is because TROLL uses this value as a guess when simulating their value for the first year (thereafter it uses the previous years simulated value).

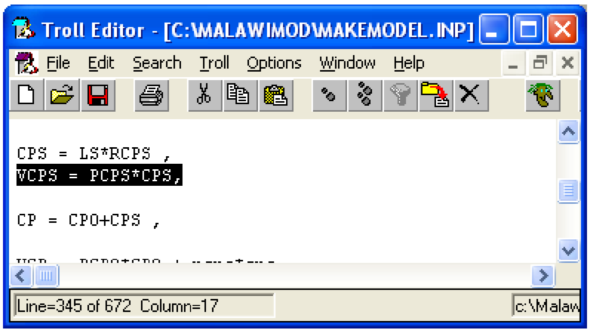
Figure 26 Declaring an endogenous variable



```
addsym endogenous
VCPS
C
CA
CPO
CPS
```

In Figure 26 and Figure 27 we gives a simple example on how to implement a new equation by using the input file MAKEMODEL.INP. We want to have an equation for the value of smallholder's consumption. First (c.f. Figure 26) we declare the new endogenous variable VCPS. Next, in Figure 27, we type in the new equation.

Figure 27 Installing a new equation


Troll Editor - [C:\MALAWIMOD\MAKEMODEL.INP]
File Edit Search Troll Options Window Help
Line=345 of 672 Column=17
c:\Malaw

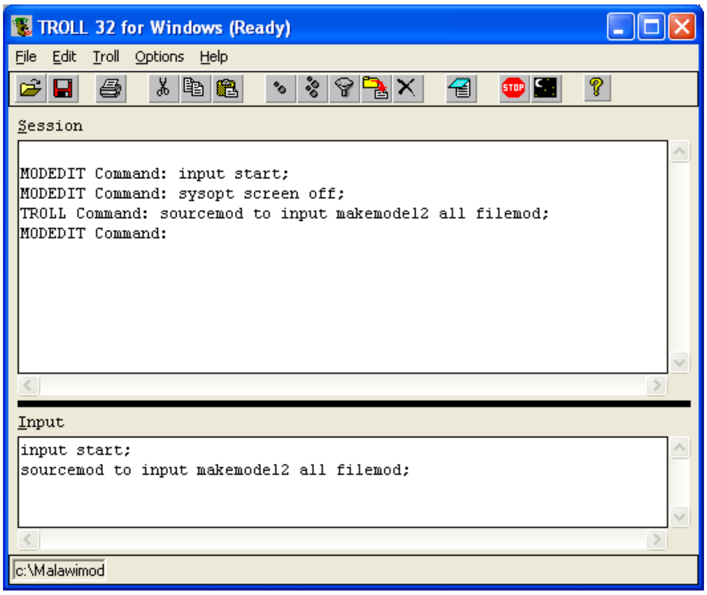
```
CPS = LS*RCPS ,
VCPS = PCPS*CPS,

CP = CPO+CPS ,
```

6.3 Creating a new input file from the model

Even though we go in for changing the model using the MAKEMODEL.INP input file we are not limited to this file. If we want so we can use the model in TROLL's internal model format (".mod") to create a new input file.

The TROLL command `sourcemod` creates an input file or string array containing the `modedit` commands needed to recreate all or part of a model. In Figure 28 we show one example on how to use this command.

Figure 28 Creating an input file from a model

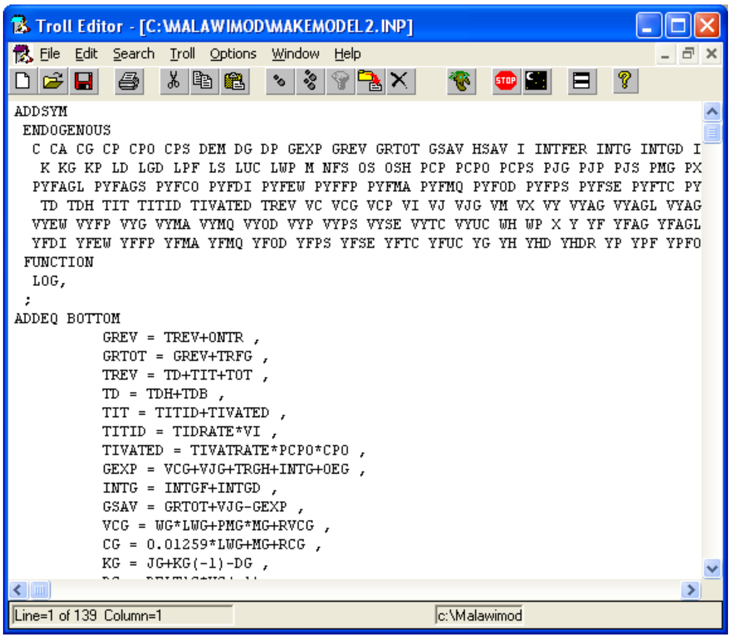


```
MODEDIT Command: input start;
MODEDIT Command: sysopt screen off;
TROLL Command: sourcemod to input makemodel2 all filemod;
MODEDIT Command:
```

```
input start;
sourcemod to input makemodel2 all filemod;
```

By using this example we will create a new input file called MAKEMODEL2.INP on the Working Directory, which will create a new model file when executed. By first using the *input start;* command we will have access to the default model `malawimod`. This model will be basis for the input to the `sourcemod` command, and, when adding the option `filemod` as in this example the input file created will terminate with the command *filemod malawimod;* Part of the input file is presented in Figure 29. Note that to run this input file to create a model we must have writeable access and search rules to the model directory. To obtain this just type *input makeeqn;* before running the MAKEMODEL2.INP input file. The `sourcemod` command has many more options and is worthwhile a closer study.

Figure 29 Input file created from a model



Troll Editor - [C:\MALAWIMOD\MAKEMODEL2.INP]
File Edit Search Troll Options Window Help
STOP

```
ADDSYM
 ENDOGENOUS
  C CA CG CP CPO CPS DEM DG DP GEXP GREV GRTOT GSAV HSAV I INTFER INTG INTGD I
   K KG KP LD LGD LPF LS LUC LWP M NFS OS OSH PCP PCPO PCPS PJG PJP PJS PMG PX
  PYFAGL PYFAGS PYFCO PYFDI PYFEW PYFFP PYFMA PYFMQ PYFOD PYFPS PYFSE PYFTC PY
   TD TDH TIT TITID TIVATED TREV VC VCG VCP VI VJ VJG VM VX VY VYAG VYAGL VYAG
  VYEW VYFP VYG VYMA VYMQ VYOD VYP VYPS VYSE VYTC VYUC WH WP X Y YF YFAG YFAGL
  YFDI YFEW YFFP YFMA YFMQ YFOD YFPS YFSE YFTC YFUC YG YH YHD YHDR YP YPF YPFO
 FUNCTION
  LOG,
 ;
ADDEQ BOTTOM
          GREV = TREV+ONTR ,
          GRTOT = GREV+TRFG ,
          TREV = TD+TIT+TOT ,
          TD = TDH+TDB ,
          TIT = TITID+TIVATED ,
          TITID = TIDRATE*VI ,
          TIVATED = TIVATRATE*PCPO*CPO ,
          GEXP = VCG+VJG+TRGH+INTG+OEG ,
          INTG = INTGF+INTGD ,
          GSAV = GRTOT+VJG-GEXP ,
          VCG = WG*LWG+PMG*MG+RVCG ,
          CG = 0.01259*LWG+MG+RCG ,
          KG = JG+KG(-1)-DG ,
```

Line=1 of 139 Column=1

c:\Malawimod

Summary

- Start TROLL
- Open the MAKEMODEL.INP file in the TROLL editor
- Edit the model
- Save the MAKEMODEL.INP file
- Click "Execute all" (or type *input makemodel*; in the input window)
- Check the session window for warnings and error messages

7 Database

7.1 Introduction

In this section we will describe how to install and update the database. The database is installed in several steps. TROLL is able to handle different database types, but in this model system we only use the type `formdata` that is a text format database. All database files of this type have the extension `frm`. In this version of the model we only use databases for time series data, and they are all located in the C:\MALAWIMOD\DATA\ – folder.

7.2 The basic database

7.2.1 Installation procedure

The basic database, or MAIN.FRM, as we call it, contains all the timeseries that we might be interested in. Some of them are used directly in the model, while others are used for creating model specific timeseries.

For some variables, the actual figures have been typed into this text file, and are installed into the database by using TROLL- functions. Other variables are installed by using already existing data. This implies that the sequence in which we write the timeseries to the database is of importance. For instance, for monetary consumption in constant prices and in current prices we have typed in actual figures, but the price deflator is calculated implicit based on the first two. This means that when calculating the price deflator, data for the constant price and the current price series must already exist in the database, i.e. have been installed previous in this file.

Box 2 Installing time series into the database

```
//Monetary consumption, constant 1994 prices
cpo = reshape( combine( 2027, 2824, 2827, 2591, 2548, 2625, 2746, 2690, 2301,
2662, 3875, 3890, 4669, 4765, 4741, 4409, 6617, 8707, 10388, 8139, 9372, 8829,
7914, 10416, 10713, 11151), 1979a ),

//Monetary consumption, current prices
vcpo = reshape( combine( 355, 329, 387, 405, 463, 546, 651, 724, 696, 1167,
1788, 1997, 2759, 3339, 4071, 4409, 12129, 21964, 28602, 29086, 48521, 59117,
67551, 102073, 115058, 137732), 1979a ),

//Implicit price deflator
//Monetary consumption
pcpo = vcpo/cpo,
```

To install the basic database (MAIN.FRM) we must type the following two commands:

```
input makedata;
input adddata;
```

The first command executes commands in the file MAKEDATA.INP, which closes any databases that might be open to ensure we are not writing into them. This file also includes some other commands to open other archives, but is really not needed for this model version, although part of the future model system.

The second command executes the commands in ADDDATA.INP, which first gives a writeable access to the database C:\MALAWIMOD\DATA\MAIN.FRM with the commands

```
access indata type formdata id &dat2inp DEFAULTDATABASE " " mode w;
search data indata w;
```

where the term `&dat2inp DEFAULTDATABASE " "` gets the physical address from TROLL's internal memory database. The value for DEFAULTDATABASE is defined in the PROFILE.PRG program that is executed automatically during start-up (se Appendix 2 for more information on the initial settings). Next are the commands to install the timeseries into the database. When all the timeseries has been installed the program closes the database.

7.2.2 Residuals

Many equations in the model contain residuals. They need a residual term to make the left hand side equal to the right hand side. There are several reasons they differ, i.e. the equation is not balanced. Some equations are econometric where the coefficients in the equation are estimated to explain the endogenous variable. For other equations we have even simpler relations between the variables, for instance two variables are related through a ratio. Another reason equations are not perfectly balanced is that some variables in the equation are measured during a calendar year while other variables in the equation are measured over a fiscal year.

For instance the equation for smallholders consumption is implemented as

```
LOG(CPO/CPO(-1)) = 0.0536+0.9848*LOG(YHDR/YHDR(-1))-0.2869*LOG(CPO(-
1)/YHDR(-1))-0.6002*DUM94+RCPO
```

and contains the residual RCPO. To calculate the values for this residual we simply turn about the equation and the expression in ADDDATA.INP is

```
rcpo = LOG(CPO/CPO(-1)) - ( 0.0536
+0.9848*log(YHDR/YHDR(-1))
-0.2869*log(cpo(-1)/YHDR(-1))
-0.6002*dum94),
```

Obviously, if we, for some reasons, change the equation for CPO in the model, we must change accordingly for RCPO in the file ADDDATA.INP, and reinstall the database.

7.2.3 Adding new data

New timeseries for variables should be added in the file ADDDATA.INP. We use several methods to install the timeseries. Suppose we want to install a new variable in the model called XX, and that we have timeseries observation for the years 2000 to 2005.

We might use the TROLL functions `reshape` and `combine` to create the timeseries as:

```
XX = reshape(combine( 972, 1006, 993, 1051, 1176, 1074), 2000a ),
```

After this entry in the ADDDATA.INP file we can use this data field to create other data fields. For instance, if XX is a variable measured in constant prices, and we have already installed the current price value for the same variable as VXX, then we can install the implicit price index PXX as

```
PXX = VXX/XX,
```

Note that in ADDDATA.INP we apply one *dofile* command to install many variables. To do this we have a `dofile` command at the top of the file, we then separate each data entry with a comma, and add a final semicolon at the end of the file (to terminate the initial `dofile` command).

7.3 Collecting model data

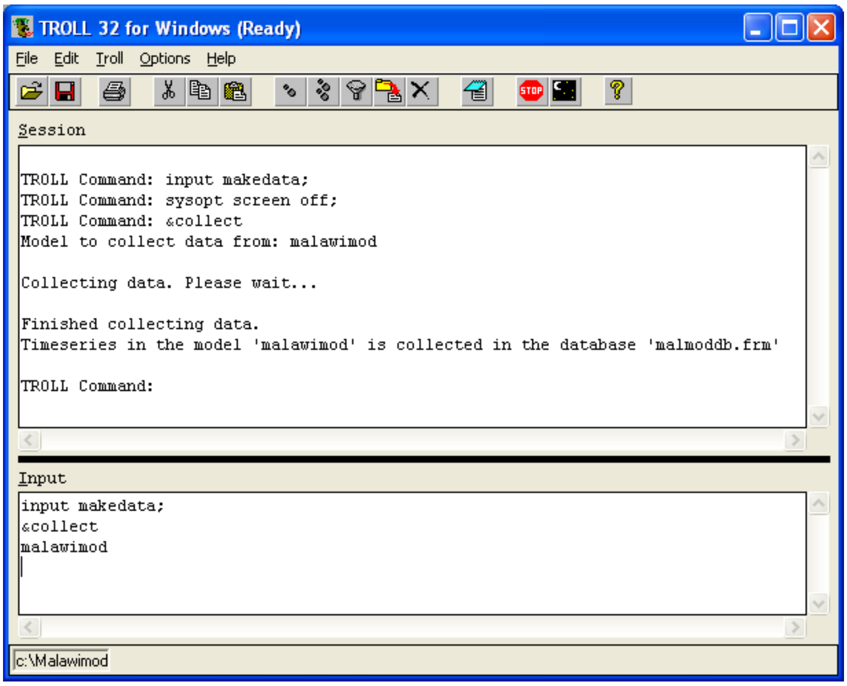
The main database contains many timeseries not needed to simulate the model. In this section we describe how to extract only the data needed from the MAIN.FRM database, to have a “pure” model database. The process is showed in Figure 30. We run the MAKEDATA.INP file, and then call the program COLLECT.PRG. The program asks for a model to collect data from. Then it loops through all variables defined in the various symbols list of the model and copies data from the MAIN.FRM timeseries database to a database called MALMODDB.FRM.

An important assumption is that the main timeseries database already exists. Here this is the MAIN.FRM database, but this name is not hard-coded into the COLLECT.PRG program. The link between the `collect` program and the MAIN.FRM is in the initial settings, and is in the variable `defaults` (short for default timeseries database). We can see this by printing the value of this variable:

```
TROLL Command: do prt.( defaultts );

DEFAULTTS:
   String scalar:  "main.frm"

TROLL Command:
```

The name of the model database MALMODDB.FRM, however, is hard-coded inside the `collect` program and cannot be altered in the initial settings.

Figure 30 Collection data for the model



```
TROLL Command: input makedata;
TROLL Command: sysopt screen off;
TROLL Command: &collect
Model to collect data from: malawimod

Collecting data. Please wait...

Finished collecting data.
Timeseries in the model 'malawimod' is collected in the database 'malmoddb.frm'

TROLL Command:
```

```
input makedata;
&collect
malawimod
```

7.4 Controlling the outcome of the model

The process of controlling the outcome of the model can be useful in many ways. The method can be used to achieve a desired value for one or more variables, or to consistently calculate missing values in the database. The procedure is also suitable in calibrating the dataset to the model. This step in creating a database for simulation is optional.

7.4.1 Override the outcome of a simulation

We will here demonstrate how to override the outcome of a model simulation to get a wanted result for a variable in the model. This can be due to a known weakness in the model, or just because we want a specific outcome for one or more variables. Suppose we, for some reason, want to achieve a given result for smallholder's consumption. The equation for smallholder's consumption is econometric and specified as

```
LOG(CPO/CPO(-1))=0.0536+0.9848*LOG(YHDR/YHDR(-1))-0.2869*LOG(CPO(-
1)/YHDR(-1))-0.6002*DUM94+RCPO
```

where RCPO is a residual term. Assume we want the result for smallholder's consumption (CPO) to be MKW 12000 millions for the period 2005-2009 when simulating the model. The "problem" here is of course that CPO is endogenous and is determined by (the simultaneous block of) the model. One approach to achieve this would be to use the residual term RCPO. We could, by trial and error, alter the residual and simulate the model until we got the desired results. A much more elegant method is to use the model to simulate the residual conditional on the desired outcome on the endogenous variable CPO. The procedure for achieving this includes the following:

1. Extrapolate smallholders consumption CPO with the desired value (12000)
2. Change the model so that CPO becomes exogenous and RCPO becomes endogenous
3. Simulate the model
4. Save the simulated value for RCPO
5. Change the model back so that CPO becomes endogenous and RCPO becomes exogenous
6. Simulate the model with the simulated values for RCPO as input

The outcome for CPO from the model simulation will now be 12000. We will now demonstrate this by utilising the program made for this.

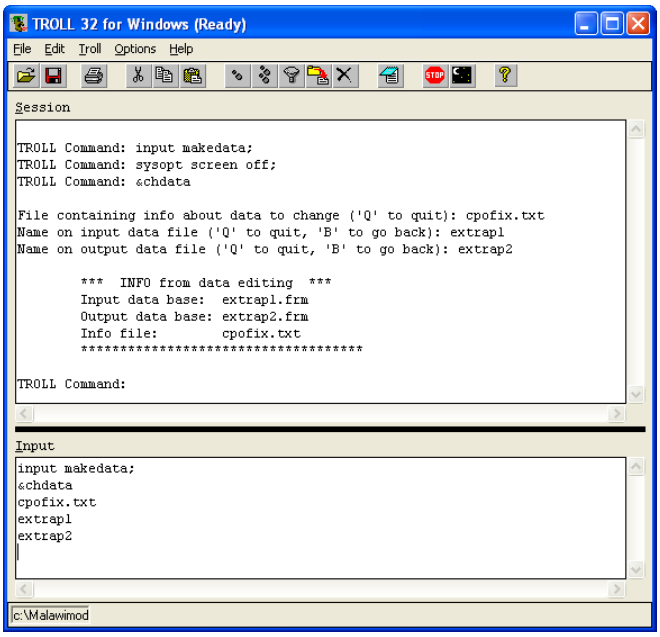
We assume that we already have a database where all variables have been extrapolated, and that this database is called EXTRAP1.FRM. First we must prepare for extrapolating the CPO variable at the value of 12000. We create a text file and call it CPOFIX.TXT. This file consist of one line:

```
addval cpo 2005a "12000 12000 12000 12000 12000"
```

This line, when read by the program, tells that we want to add values to CPO from 2005 until 2009. The end date 2009 is implicit defined by the number of values we want to add. The first value will be added in 2005, the second value in 2006, and so on. Note that CPO is endogenous in our "normal" model and consequently has not been automatically extrapolated.

Now we use the program CHDATA.PRG to implement the "target value" for CPO into a database. This is showed in Figure 31. We start by executing the MAKEDATA.INP file to access the needed directories. We further execute the program CHDATA.PRG that asks for the name of the input database, i.e. the database EXTRAP1.FRM. It next asks for the name of an output database to create, which we here

Figure 31 Implementing assumption about a variable



```
TROLL Command: input makedata;
TROLL Command: sysopt screen off;
TROLL Command: &chdata

File containing info about data to change ('Q' to quit): cpofix.txt
Name on input data file ('Q' to quit, 'B' to go back): extrapl
Name on output data file ('Q' to quit, 'B' to go back): extrap2

        ***  INFO from data editing  ***
        Input data base:  extrapl.frm
        Output data base: extrap2.frm
        Info file:        cpofix.txt
        ************************************

TROLL Command:
```

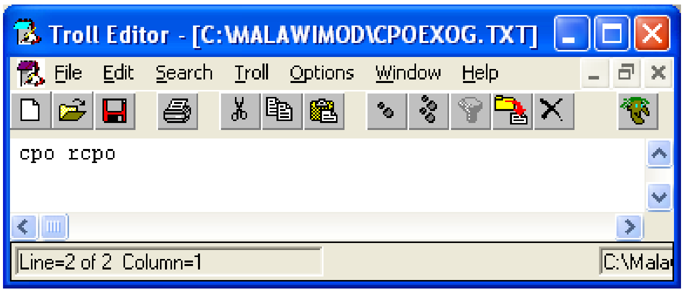
```
input makedata;
&chdata
cpofix.txt
extrapl
extrap2
```

call EXTRAP2.FRM. Finally the program asks for the text file (CPOFIX.TXT) where we have specified the values for CPO. The values for CPO is now set to 12000 for the period 2005 to 2009:

```
TROLL Command: do prt.( cpo );

CPO:
   Numeric scalar timeseries --
   Time dimension:  Annual, 1979A to 2009A (31 observations)

       Time dimension -->
1979A:     2027            2824            2827            2591
1983A:     2548            2625            2746            2690
1987A:     2301            2662            3875            3890
1991A:     4669            4765            4741            4409
1995A:     6617            8707           10388            8139
1999A:     9372            8829            7914           10416
2003A:    10713           11151           12000           12000
2007A:    12000           12000           12000
```

The next step is to change the model so that CPO becomes exogenous and RCPO becomes endogenous, and to simulate the value for RCPO using this model. We can do this by using the program ENDO2EXOG.PRG. The program takes information on which variables to endogenise/exogenise from a text file so we must first prepare this file. We create a new text file and name it CPOEXOG.TXT. The file consists solely of the name of the variable to become exogenous and of that to become endogenous as in Figure 32.

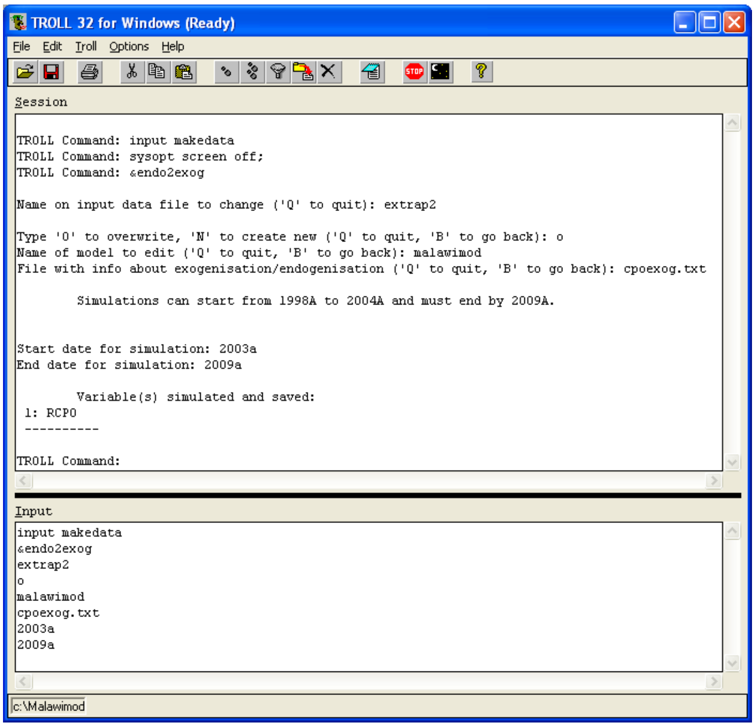
Figure 32 Changing symboltype from exogenous to endogenous


Troll Editor - [C:\MALAWIMOD\CPOEXOG.TXT]
File Edit Search Troll Options Window Help
Line=2 of 2 Column=1
C:\Mala

```
cpo rcpo
```

Now we are ready to simulate the values for the residual RCPO that will be consistent with the specified values for CPO. This is showed in Figure 33. The program ENDO2EXOG.PRG first asks for the name of the input database. This is the database EXTRAP2.FRM where we have extrapolated CPO with value 12000 for the period 2005 to 2009. Next the program asks if we want to overwrite (the variable in) the database or create a new. In this example we choose to overwrite, i.e. write back to the same database. Next we must specify which model to use, and the name of the text file CPOEXOG.TXT specifying for which variables to change symbol types between endogenous and exogenous (c.f. Figure 32). Provided with this information the program now analyses the model for possible simulation dates, prints this to screen and asks for the start date and end date for the simulation. We simulate the model for the period 2003 to 2009. Although the whole model is simulated, the only variable saved is that for RCPO as we can see in the message from the program. This value for RCPO, when used as exogenous input to the (normal) model, is the values that ensure the specified outcome for CPO. The program ENDO2EXOG.PRG creates a temporary model called TMPMOD.MOD in the model folder where RCPO is endogenous.

Figure 33 Enforce an outcome of the model

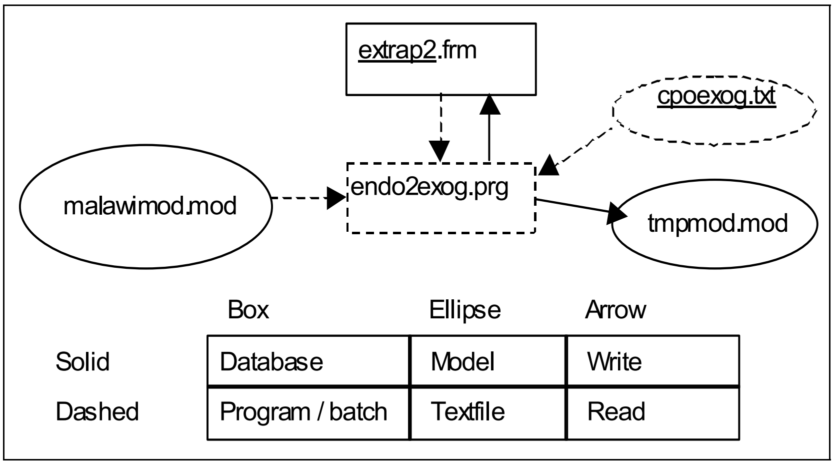


```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: &endo2exog

Name on input data file to change ('Q' to quit): extrap2

Type 'O' to overwrite, 'N' to create new ('Q' to quit, 'B' to go back): o
Name of model to edit ('Q' to quit, 'B' to go back): malawimod
File with info about exogenisation/endogenisation ('Q' to quit, 'B' to go back): cpoexog.txt

        Simulations can start from 1998A to 2004A and must end by 2009A.


Start date for simulation: 2003a
End date for simulation: 2009a

        Variable(s) simulated and saved:
 1: RCPO
 ----------

TROLL Command:
```

```
input makedata
&endo2exog
extrap2
o
malawimod
cpoexog.txt
2003a
2009a
```

The procedure described here is outlined in Figure 34. We see that the `endo2exog` program reads data from the database EXTRAP2.FRM, reads information on which variables to endogenise and exogenise from the text file CPOEXOG.TXT and then simulates the model. The results for the new endogenous variable RCPO is written back to EXTRAP2.FRM.

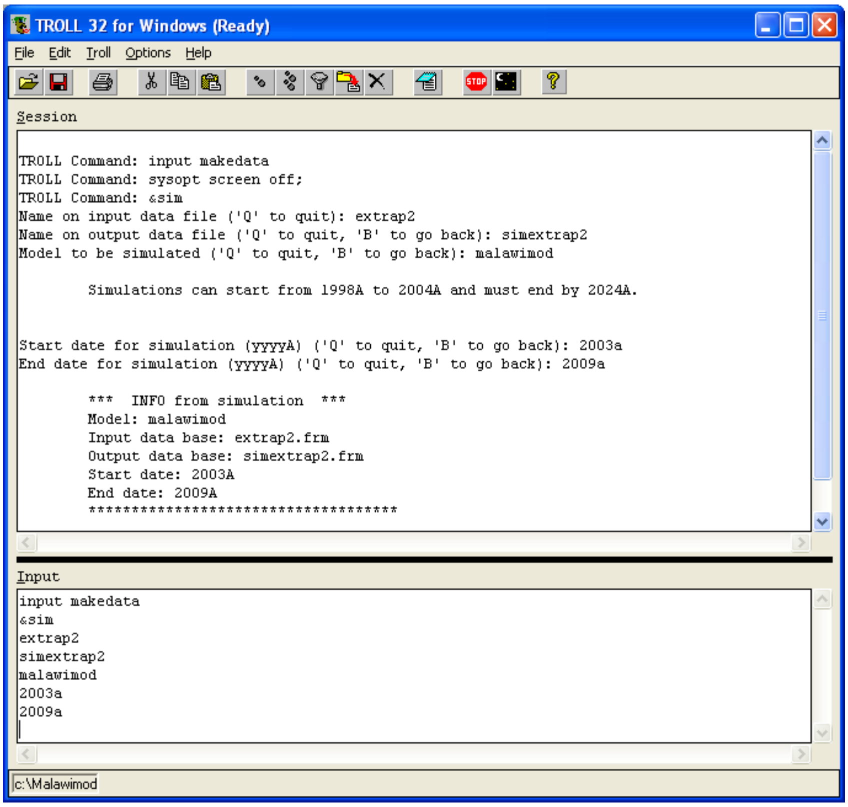
Figure 34 Dataflow for endo2exog (example)



	Box	Ellipse	Arrow
Solid	Database	Model	Write
Dashed	Program / batch	Textfile	Read

Now we can simulate the model and we will see that the endogenous CPO will have the value of 12000 for the years 2005 to 2009. In Figure 35 we show how this is done. The input database EXTRAP2.FRM now contains, amongst all other variables, the simulated values for RCPO. To put it simple; we have simulated the model to find the exogenous input values that will result in a desired outcome.

Figure 35 Simulating a designed outcome



```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: &sim
Name on input data file ('Q' to quit): extrap2
Name on output data file ('Q' to quit, 'B' to go back): simextrap2
Model to be simulated ('Q' to quit, 'B' to go back): malawimod

        Simulations can start from 1998A to 2004A and must end by 2024A.


Start date for simulation (yyyyA) ('Q' to quit, 'B' to go back): 2003a
End date for simulation (yyyyA) ('Q' to quit, 'B' to go back): 2009a

        ***  INFO from simulation  ***
        Model: malawimod
        Input data base: extrap2.frm
        Output data base: simextrap2.frm
        Start date: 2003A
        End date: 2009A
        ************************************
```

```
input makedata
&sim
extrap2
simextrap2
malawimod
2003a
2009a
```

To check the result for CPO we print it:

```
TROLL Command: do prtime ( cpo );

             CPO

2003A    10713
2004A    11151.00001
2005A    11999.99912
2006A    11999.988211
2007A    11999.993245
2008A    11999.995377
2009A    11999.996093

TROLL Command:
```

Which is (apart from the exactness) what we wanted to achieve.

7.4.2 Calculating missing values

We will now demonstrate another way of using the technique involving interchanging exogenous and endogenous variables. Even though this is a model based on annual data, it might be that some values are not available when we update the model with the latest years data. We will here explain briefly a situation where we can use the above-mentioned technique to calculate (model) consistent values for missing values.

Consider the following situation. All the variables in the historical database have values for 2004, except the variable for government investment in constant prices (JG), which ends in 2003. The situation is illustrated in Table 11. This variable is exogenous in the model and the simulation is

Table 11 Missing a historical value

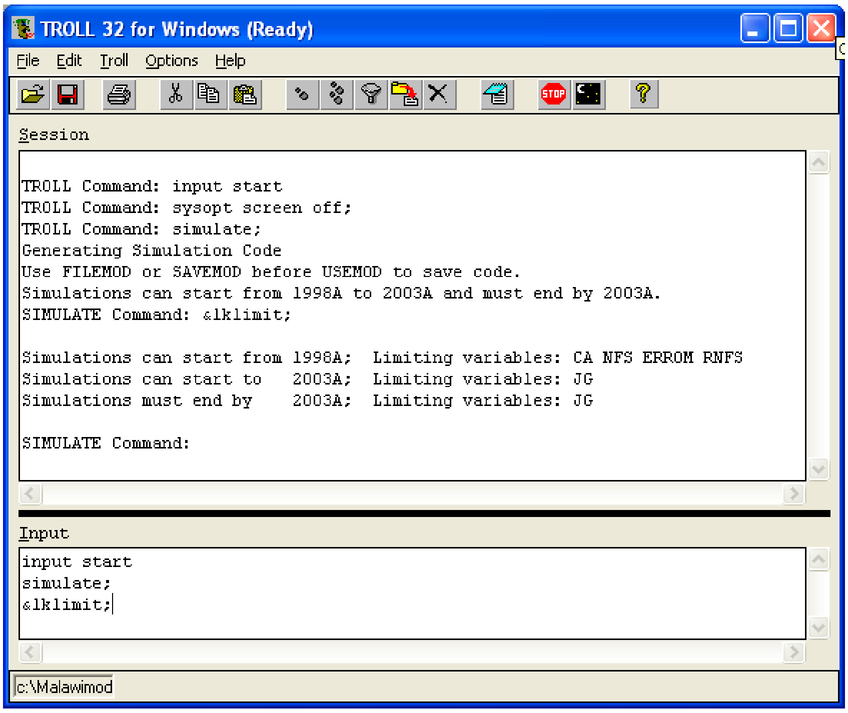
	VJG	PJG	JG
2004	15436	11.7	**NA**

limited because of missing data. One solution then is to use a definition equation where JG appears to calculate it. One such equation is

```
VJG = PJG*JG
```

In this equation government investment in current value is determined as the price index multiplied by the government consumption in constant prices. It is obvious that when JG ends in 2003 it will limit the calculation of VJG to the same end date. This can also be seen from Figure 36 where we have used the TROLL-program `lklimit` to check for limiting variables. The program tells us that we can start a simulation from 1998 (limited by CA, NFS, ERROM and RNFS) and that it can start to 2003 (limited by JG). This somewhat cryptically message means in practice that we can start the simulation between 1998 and 2003, and that it must end by 2003 (limited only by JG).

Figure 36 Check of limiting variables in a database

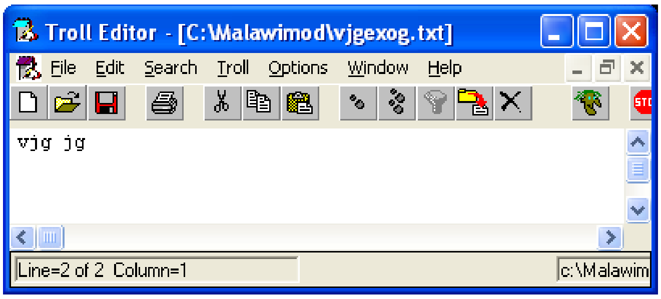


```
TROLL Command: input start
TROLL Command: sysopt screen off;
TROLL Command: simulate;
Generating Simulation Code
Use FILEMOD or SAVEMOD before USEMOD to save code.
Simulations can start from 1998A to 2003A and must end by 2003A.
SIMULATE Command: &lklimit;

Simulations can start from 1998A;  Limiting variables: CA NFS ERROM RNFS
Simulations can start to   2003A;  Limiting variables: JG
Simulations must end by    2003A;  Limiting variables: JG

SIMULATE Command:
```

```
input start
simulate;
&lklimit;
```

By using the same method as in the previous section we will now simulate the missing value for JG in 2004 that is consistent with the models equations. First we must prepare the text file that tells which variables we want to make endogenous and exogenous. We create a text file called VJGEXOG.TXT (c.f. Figure 37) that specifies that we want to make VJG exogenous and JG

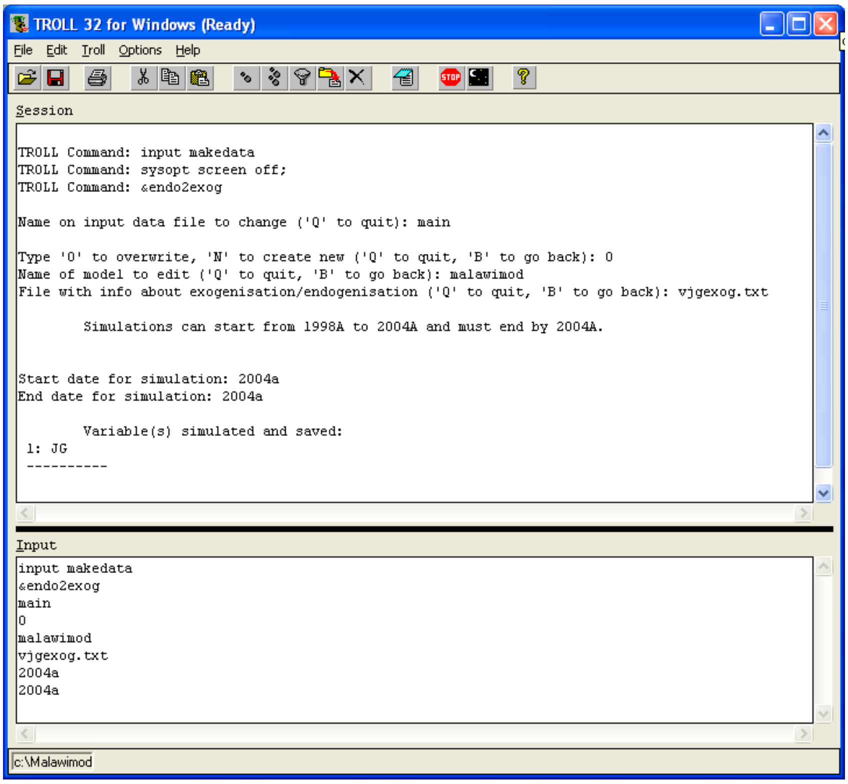
Figure 37 Information on changing symbol types in a model


Troll Editor - [C:\Malawimod\vjgexog.txt]
File Edit Search Troll Options Window Help
Line=2 of 2 Column=1
c:\Malawim

```
vjg jg
```

endogenous. In Figure 38 we show the use of the program ENDO2EXOG.PRG to simulate the value for JG in 2004. For the purpose of demonstration we have used the MAIN.FRM database and chosen to write the simulated value for JG back to the same database. This is the only value written back to the database, all others remaining untouched.

Figure 38 Simulating a missing value



```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: &endo2exog

Name on input data file to change ('Q' to quit): main

Type 'O' to overwrite, 'N' to create new ('Q' to quit, 'B' to go back): 0
Name of model to edit ('Q' to quit, 'B' to go back): malawimod
File with info about exogenisation/endogenisation ('Q' to quit, 'B' to go back): vjgexog.txt

        Simulations can start from 1998A to 2004A and must end by 2004A.


Start date for simulation: 2004a
End date for simulation: 2004a

        Variable(s) simulated and saved:
 1: JG
 ----------
```

```
input makedata
&endo2exog
main
0
malawimod
vjgexog.txt
2004a
2004a
```

In Table 12 we see that we now have a value for JG that is consistent.

Table 12 Simulated JG

	VJG	PJG	JG
2004A	15436	11.728	1316.166

The model can now be simulated until 2004 using this MAIN.FRM database.

It should be noted that the text file with information on which variables we want to change to endogenous and exogenous could have multiple pairs of variables. One should consider carefully however, which couples of variables to change. The pair of variables does not have to be in the same equation, but they must be related, and we must ensure there are assignable variables in all equations. For instance, if we want to change the variable WP to be exogenous and JG to be endogenous we will have the following message from TROLL:

```
WARNING 3211
There are no assignable variables in some equations:
  70
Problem was detected in macro file 'ENDO2EXOG' near line #178.

ERROR 3212
The model cannot be block-ordered due to problems with the incidence
matrix.
Problem was detected in macro file 'ENDO2EXOG' near line #178.
```

According to this TROLL - warning the error occurs because equation 70, which is

```
WP = WPRATE*WG
```

now only contains exogenous variables (and consequently no variable to simulate). The TROLL – error occurs because there are now only 115 equations to solve 116 variables.

By printing the equations containing WP and JG respectively we see that there are no relation between them:

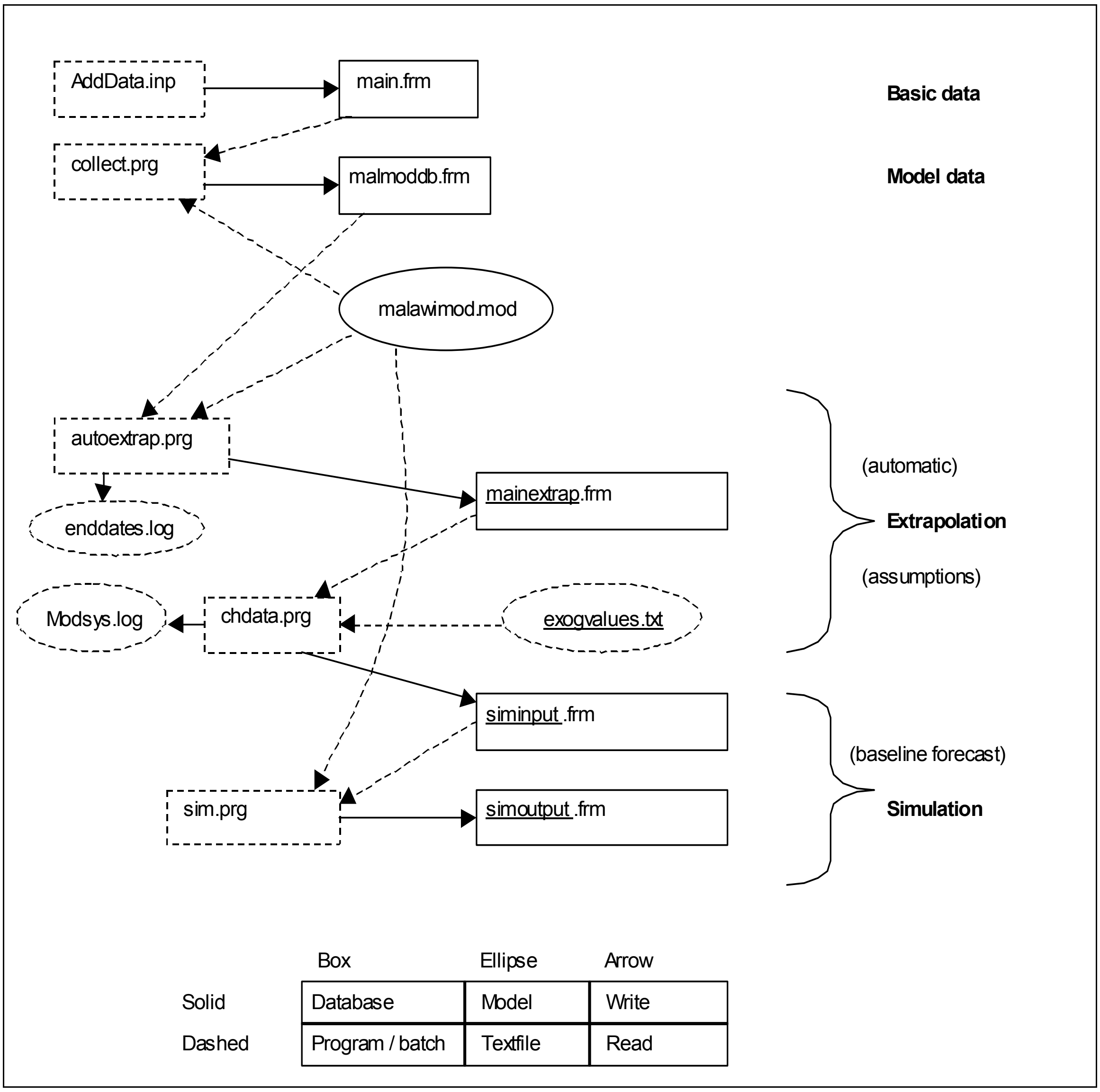
```
TROLL Command: lksym wp jg;

Variable     Used in Equations

JG            13  17  42  61  67
WP            26  48  50  70

TROLL Command:
```

The text file containing the paired endogenous and exogenous variables must be stored at the Working Directory.

7.4.3 Data structure of the model

In Figure 39 we show a flowchart of the data structure for the model system. To the right is an explanation of what the task to the left does. Note that neither the calibration nor the shift processes have been included here. Attention should be paid to the explanatory boxes at the bottom, as they are important for understanding the flowchart. All or parts of file names that are underlined are optional. As an example, we see that the dashed box named ADDDATA.INP (on the top) symbolises a batch (input) file and will write (solid arrow) to the database (solid box) called MAIN.FRM (not underlined so it's not an optional name).

Figure 39 Data structure in the model



	Box	Ellipse	Arrow
Solid	Database	Model	Write
Dashed	Program / batch	Textfile	Read

8 Interaction with spreadsheets

8.1 Introduction

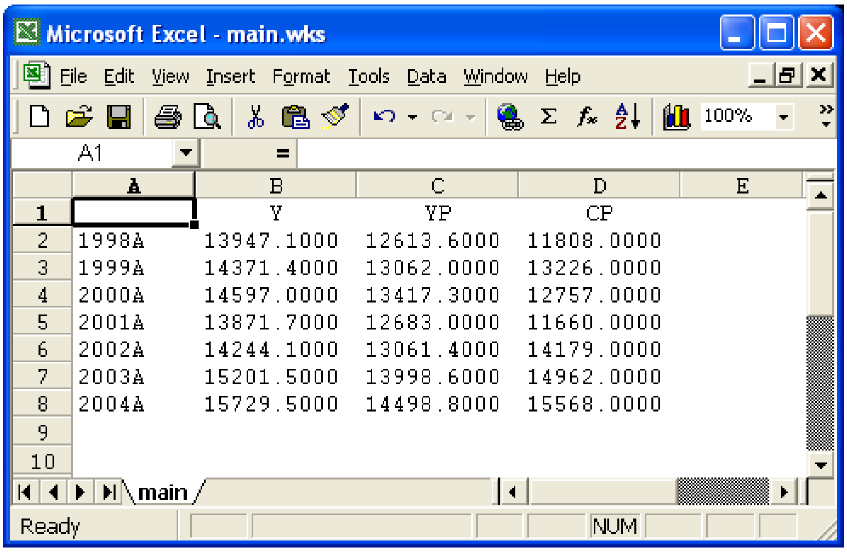
In the first place the data for the model was provided in spreadsheets. Therefore we developed programs to interact with spreadsheets from TROLL, i.e. install data from a spreadsheet into a TROLL database and exporting data from a TROLL database to spreadsheets. As the basis for our model database we don't use the spreadsheet database anymore, however, the use of spreadsheets are increasingly popular, so we will go through how we can read from and write to spreadsheets.

TROLL can read various types of spreadsheets but not Microsoft Excel format. The programs we have developed will read from and write to WKS format. Microsoft Excel though, handles WKS format, and can be used both to create spreadsheets that we want to install into a TROLL database, or to open spreadsheets in WKS format created from our TROLL databases. In this chapter we will first describe the format of the spreadsheets to be used with TROLL. Then we go through how to convert between Excel and WKS formats, before we demonstrate how to import and export data between TROLL's databases and spreadsheets.

8.2 The format of the spreadsheet

To use spreadsheets in TROLL they require a certain format. In Figure 40 we show a spreadsheet that was created from the model system, and this format is the one that must be used. Column A must consist of the periodicity, i.e. the dates for the observation. They must be in TROLL's format, i.e. include an "A" after the number of the year. This should start from cell A2 (leaving cell A1 empty) and continue until the last observation. Note that the timeseries does not have to be of equal length, and Column A should reflect the span of the timeseries with the earliest and the latest dates included. If the timeseries with the earliest observation starts in 1970, and the timeseries with the latest observation ends in 2005, then cell A2 should be 1970A and A37 should be 2005A. The variable names should be on the first line (row 1), starting from cell B1, and must

Figure 40 Format of spreadsheet


Microsoft Excel - main.wks
File Edit View Insert Format Tools Data Window Help
100%
A1
=
main
Ready
NUM

	A	B	C	D	E
1		Y	YP	CP	
2	1998A	13947.1000	12613.6000	11808.0000	
3	1999A	14371.4000	13062.0000	13226.0000	
4	2000A	14597.0000	13417.3000	12757.0000	
5	2001A	13871.7000	12683.0000	11660.0000	
6	2002A	14244.1000	13061.4000	14179.0000	
7	2003A	15201.5000	13998.6000	14962.0000	
8	2004A	15729.5000	14498.8000	15568.0000	
9					
10					

be in uppercase. In Figure 40 the spreadsheet file MAIN.WKS contains 3 timeseries of equal length. The spreadsheet program we use restricts total numbers of timeseries in a single spreadsheet. In Microsoft Excel's case it is limited to 256 timeseries. The length of the timeseries is virtually not a problem.

8.3 Microsoft Excel and WKS format

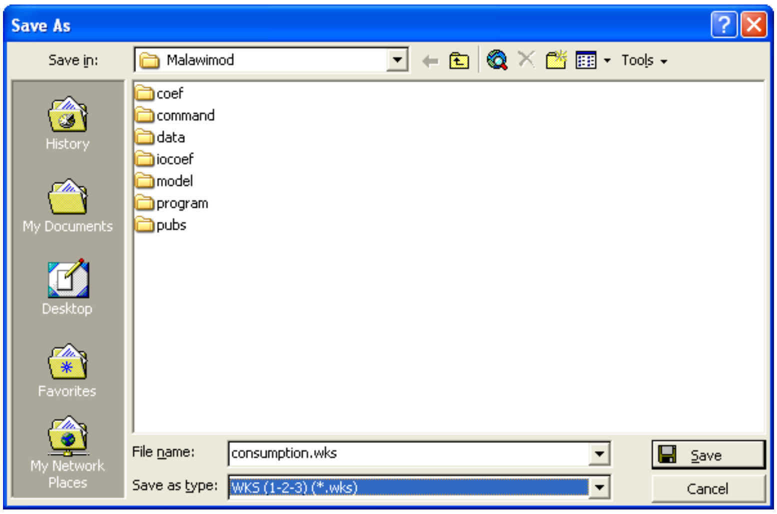
8.3.1 From Microsoft Excel to WKS format

To be able to read data from a Microsoft Excel spreadsheet into a TROLL database we must first convert it to WKS format. To accomplish this we should follow this procedure:

- Open the Excel spreadsheet to convert
- Go to File -> Save as
- Click the dropdown list "Save as type:" (see Figure 41)
- Choose WKS (1-2-3)
- Click Save (Note: Depending on your set-up and version of Excel/Windows there may be some warnings. WKS do not support multiple sheets so we should agree to save only the active sheet. Further, WKS do not support all the features in Excel so we will be given some information regarding this.)
- Close the spreadsheet

In Figure 41 we are saving an Excel spreadsheet file called CONSUMPTION.XLS as CONSUMPTION.WKS.

Figure 41 Converting an Excel file to WKS format



Save in: Malawimod

coef
command
data
iocoef
model
program
pubs

File name: consumption.wks

Save as type: WKS (1-2-3) (*.wks)

8.3.2 From WKS format to Microsoft Excel

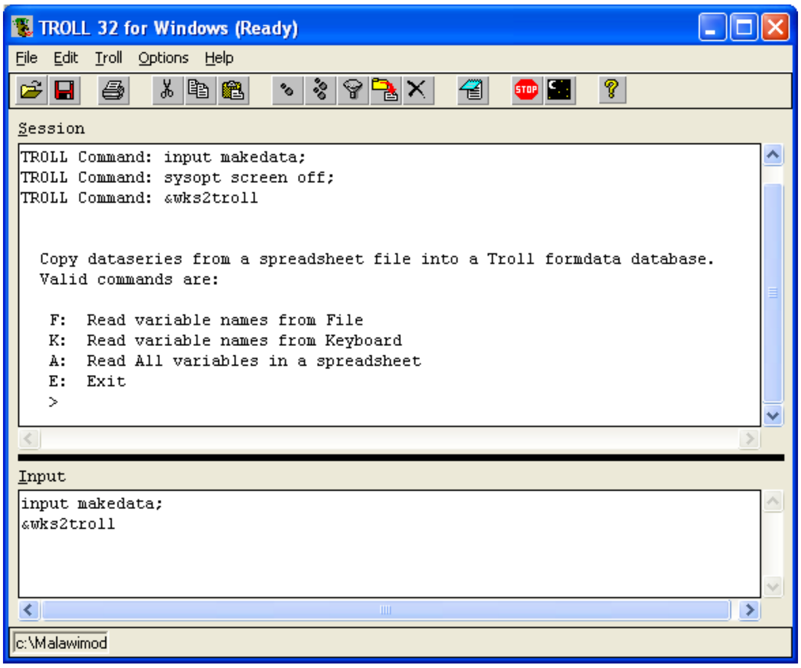
We also want to go the other way, i.e. open a WKS file that we have created from TROLL in Excel. Depending on the computer settings, we might have to specify which program to use to open the WKS file. If the computer doesn't recognise the WKS file type, we try to open it and then (when requested) specify that we want to use Microsoft Excel (or another spreadsheet program if we prefer). Note that when we open the spreadsheet we have created from TROLL, the cells might consist of number (hash) signs (####). This is either because of the potentially large numbers of decimals used when exporting from TROLL, or because the numbers themselves are large. To view the proper content of the cells we must adjust the column width and/or reduce the number of decimals. To save this file as an Excel file we must use the option

"Save as" from the file menu in Excel, and then chose "Microsoft Excel Workbook (*.xls)" from the "Save as type:" field.

8.4 Copy data from a spreadsheet to a TROLL database

The program for copying timeseries from a spreadsheet to a TROLL database is called WKS2TROLL.PRG. Let us start with executing the program to see its options. Figure 42 show that we have three options (in addition to exit) when it comes to specify the variables we want to copy from a spreadsheet. By typing F we can support the variables names from a file, we can chose K to give the variable names from the keyboard, or we can chose A to read all variables in a spreadsheet. By choosing E the program terminates.

Figure 42 The wks2troll program



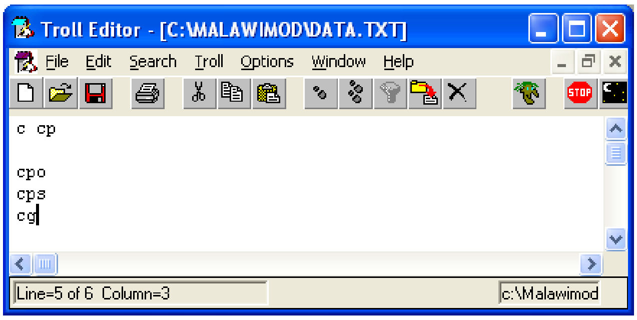
```
TROLL Command: input makedata;
TROLL Command: sysopt screen off;
TROLL Command: &wks2troll


  Copy dataseries from a spreadsheet file into a Troll formdata database.
  Valid commands are:

   F:  Read variable names from File
   K:  Read variable names from Keyboard
   A:  Read All variables in a spreadsheet
   E:  Exit
   >
```

```
input makedata;
&wks2troll
```

If reading data from a spreadsheet into a TROLL database were a regular task, especially if it involves many variables, it would be wise to store the variable names that we want to read in a text file so that it can be reused. In Figure 43 we shows how this file could look like. We can write many variables on the same line or one variable on each line. If we specify variables in the text file that doesn't exist in the spreadsheet, the program will write a warning to screen. The TROLL databases created will be written to the C:\MALAWIMOD\DATA folder.

Suppose we have a spreadsheet file CONSUMPTION.WKS that we want to transfer to a TROLL database, and that we want to specify which data from the spreadsheet we want to copy by using a file named DATA.TXT.

Figure 43 Specifying data to copy from spreadsheet from a text file


Troll Editor - [C:\MALAWIMOD\DATA.TXT]
File Edit Search Troll Options Window Help

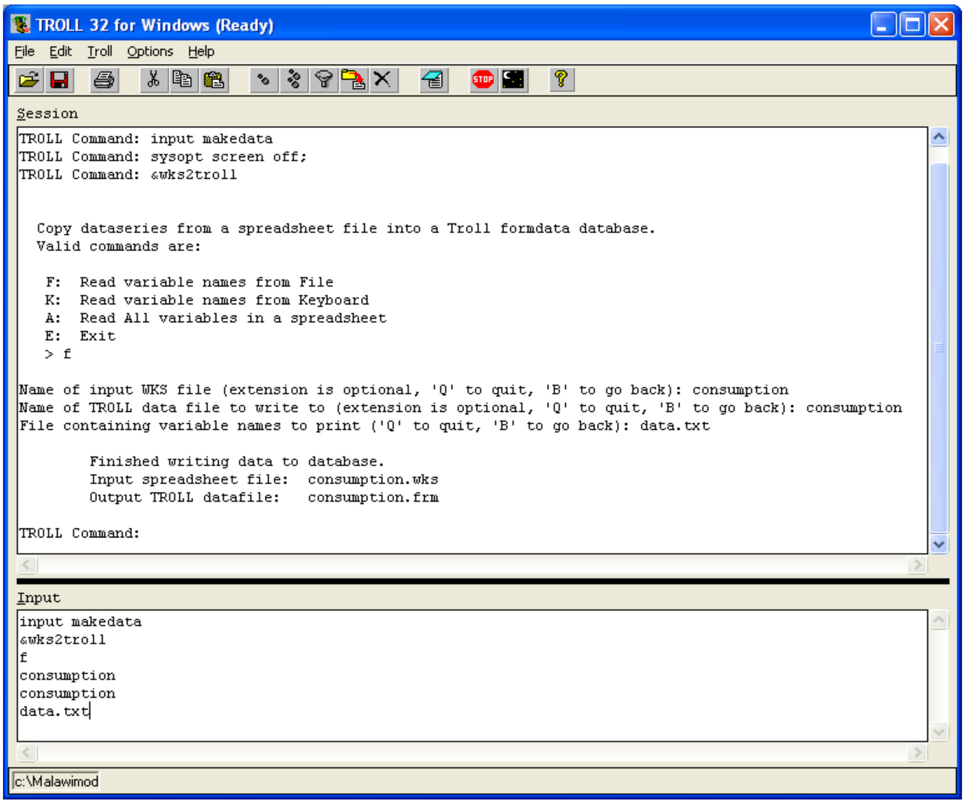
```
c cp

cpo
cps
cg
```

Line=5 of 6 Column=3

c:\Malawimod

Accordingly we have chosen "*F*" as option from the menu presented by the WKS2TROLL.PRG program (see Figure 44). Next the program asks for the name of the input WKS file, and we give the name of the spreadsheet we want to read from (*consumption*). Next we must specify the name

Figure 44 Copy data from spreadsheet



```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: &wks2troll


  Copy dataseries from a spreadsheet file into a Troll formdata database.
  Valid commands are:

   F:  Read variable names from File
   K:  Read variable names from Keyboard
   A:  Read All variables in a spreadsheet
   E:  Exit
   > f

Name of input WKS file (extension is optional, 'Q' to quit, 'B' to go back): consumption
Name of TROLL data file to write to (extension is optional, 'Q' to quit, 'B' to go back): consumption
File containing variable names to print ('Q' to quit, 'B' to go back): data.txt

        Finished writing data to database.
        Input spreadsheet file:  consumption.wks
        Output TROLL datafile:   consumption.frm

TROLL Command:
```

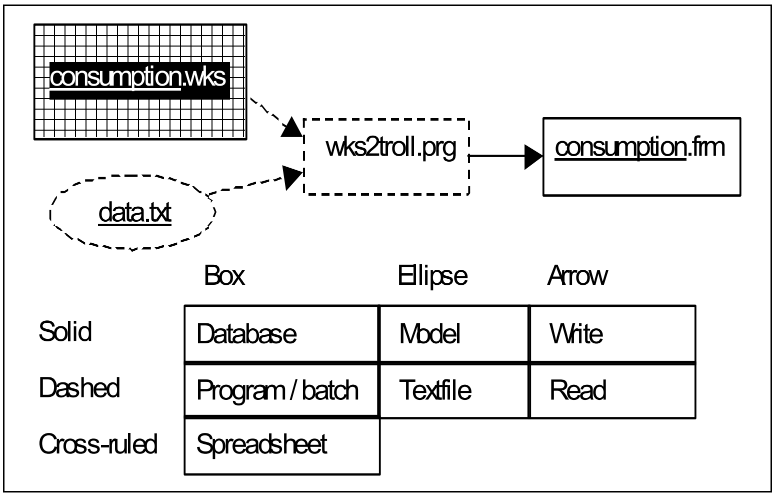
```
input makedata
&wks2troll
f
consumption
consumption
data.txt
```

c:\Malawimod

of the TROLL database we want to create, we call this *consumption* too (but it will be CONSUMPTION.FRM in the C:\MALAWIMOD\DATA folder). Finally the program asks for the name of the text file containing the variable names we want to copy. In this case it is DATA.TXT. The TROLL database CONSUMPTION.FRM, containing the variables specified in DATA.TXT, will now be created.

In Figure 45 we show the data structure for the use of the WKS2TROLL.PRG program as it has been used in this example.

Figure 45 Data structure for reading from spreadsheet (example)



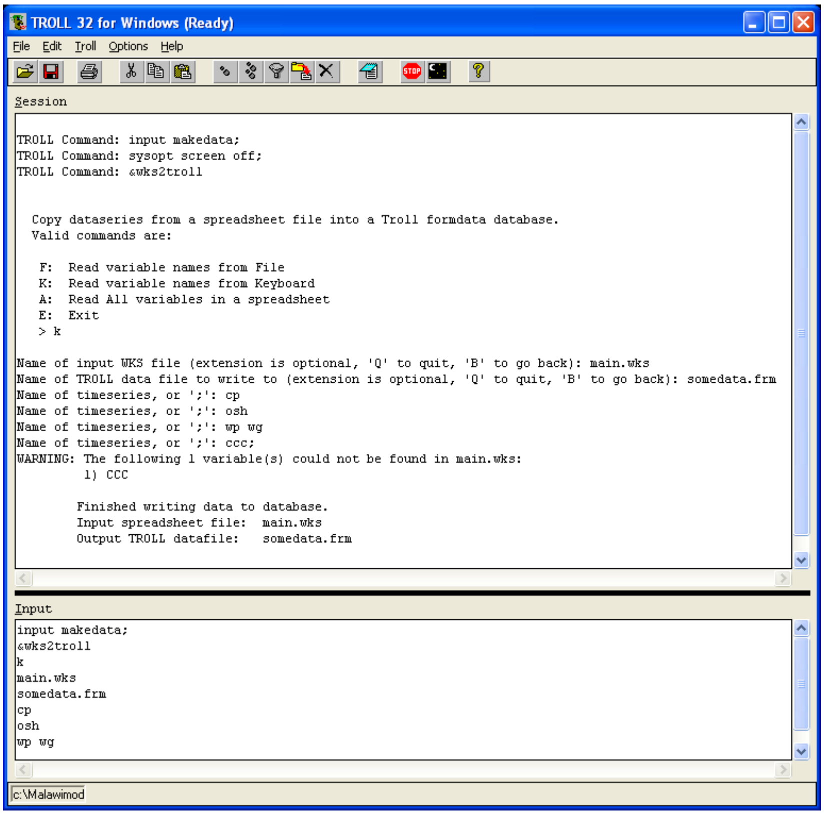
	Box	Ellipse	Arrow
Solid	Database	Model	Write
Dashed	Program / batch	Textfile	Read
Cross-ruled	Spreadsheet		

In Figure 46 we demonstrate how to specify which variables to read from the keyboard. Note that the variable "ccc" was not found in the spreadsheet MAIN.WKS and this caused a warning.

If the TROLL database we want to write to already exists (SOMEDATA.FRM in this example), we will be warned and given the following options:

- Overwrite file (data will be deleted before the new variables is written to this file)
- Rename (give a new name, leaving existing file untouched)
- Append (write the variables specified to the TROLL database, overwriting their existing values, leaving other variables unchanged)
- Exit

Figure 46 Copying from spreadsheet to TROLL database from keyboard



```
TROLL Command: input makedata;
TROLL Command: sysopt screen off;
TROLL Command: &wks2troll


  Copy dataseries from a spreadsheet file into a Troll formdata database.
  Valid commands are:

   F:  Read variable names from File
   K:  Read variable names from Keyboard
   A:  Read All variables in a spreadsheet
   E:  Exit
   > k

Name of input WKS file (extension is optional, 'Q' to quit, 'B' to go back): main.wks
Name of TROLL data file to write to (extension is optional, 'Q' to quit, 'B' to go back): somedata.frm
Name of timeseries, or ';': cp
Name of timeseries, or ';': osh
Name of timeseries, or ';': wp wg
Name of timeseries, or ';': ccc;
WARNING: The following 1 variable(s) could not be found in main.wks:
         1) CCC

        Finished writing data to database.
        Input spreadsheet file:  main.wks
        Output TROLL datafile:   somedata.frm
```

```
input makedata;
&wks2troll
k
main.wks
somedata.frm
cp
osh
wp wg
```

8.5 Copy data from a TROLL database to a spreadsheet

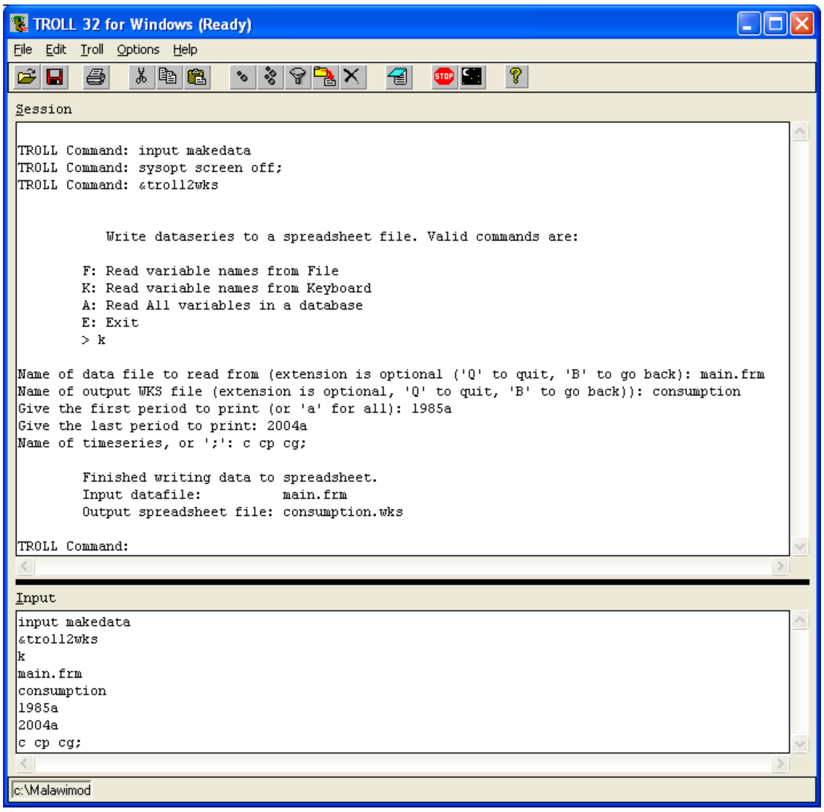
There can be many reasons for wanting to export data from a TROLL database to a spreadsheet. We may want to make some nice presentations from a simulation or policy analysis, or use the numbers for further calculations. Regardless of what reason, we want to show how to perform this by using a program called TROLL2WKS.PRG. In Figure 47 we demonstrates how to use the program. To access the program we must run the input file MAKEDATA.INP. When we execute TROLL2WKS.PRG we are given the following 3 options (in addition to exit):

- Read the variable names we want to copy to spreadsheet from a file
- Read the variable names we want to copy to spreadsheet from the keyboard
- Read all the variables in the database to a spreadsheet

In our example we have chosen option *K* to give the variables to export from the keyboard. We typed *main* as input database and *consumption* as output spreadsheet, which will result in reading from the MAIN.FRM TROLL database to the spreadsheet called CONSUMPTION.WKS. In this example we have specified a particular time period (1985 to 2004) for the timeseries we wish to export to the spreadsheet,

If a spreadsheet with the same name as we specify already exists it will be overwritten without any warnings given.

Figure 47 Copy data from TROLL to spreadsheet

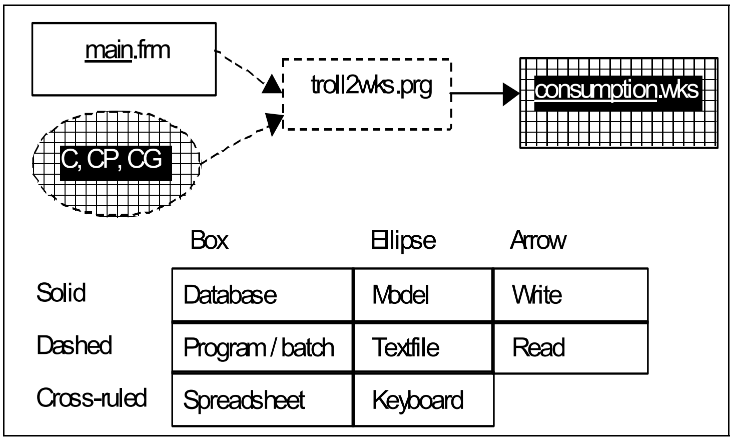


```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: &troll2wks


           Write dataseries to a spreadsheet file. Valid commands are:

        F: Read variable names from File
        K: Read variable names from Keyboard
        A: Read All variables in a database
        E: Exit
        > k

Name of data file to read from (extension is optional ('Q' to quit, 'B' to go back): main.frm
Name of output WKS file (extension is optional, 'Q' to quit, 'B' to go back)): consumption
Give the first period to print (or 'a' for all): 1985a
Give the last period to print: 2004a
Name of timeseries, or ';': c cp cg;

        Finished writing data to spreadsheet.
        Input datafile:         main.frm
        Output spreadsheet file: consumption.wks

TROLL Command:
```

```
input makedata
&troll2wks
k
main.frm
consumption
1985a
2004a
c cp cg;
```

In Figure 48we show the data structure for the use of the TROLL2WKS.PRG program as it has been used in this example. We have illustrated how to copy 3 consumption variables from the MAIN.FRM TROLL database into a spreadsheet called CONSUMPTION.WKS. The spreadsheet will be created at the Working Directory.

Figure 48 Data structure for copying to a spreadsheet (example)



	Box	Ellipse	Arrow
Solid	Database	Model	Write
Dashed	Program / batch	Textfile	Read
Cross-ruled	Spreadsheet	Keyboard	

Index

Appendix 1. Functions

Appendix 1.1. Introduction

To facilitate the task of implementing our assumptions for the exogenous variables some functions have been developed. We have already demonstrated these in chapter 4.2 “Extrapolating the exogenous variables” and in 5.1 “Implementing a shift”. There we used the programs CHDATA.PRG and SHIFT.PRG, which read the assumptions specified from a text file. What the programs actually do is to call some TROLL functions we have developed and pass over the information from this text file. These are actually stand-alone functions and might be used as such, as we will demonstrate here. The functions are located in the C:\TROLLRESOURCES\LIB directory. In Table 13 we list the functions used to manipulate variables and give a short explanation on their use. To be familiar with the functions and the way they operate we should try them out.

Table 13 Functions used for manipulating variables

Function	Comment
ADDVAL	Add values to time series
ADJUST	Adjust time series by per cent and a constant added
EXTRAP	Extrapolate time series with a given growth rate
EXTRAPA	Extrapolate time series with a given growth rate and a constant added
PCFDADJ	Adjust time series by changing the growth rate by a constant
#	Symbol for comments

Appendix 1.2. Functions for adjusting time series

All the functions take parameters as arguments. These parameters consist of the variable name of the time series to change and the first period or start date for changing. Then, depending on which function, they take other parameters such as end date, per cent growth rate, change in growth rate, a constant to add, a given value to insert, etc.

The functions `adjust` and `pcfdadj` adjust already existing data so their time series must have been extrapolated before we can manipulate on them, whereas the others will add the values specified if they does not exist.

When the functions are used as in CHDATA.PRG and SHIFT.PRG they have their parameter values from a text file (c.f. Figure 18 and in Figure 23). To use them as a stand-alone function we can follow the example in Figure 49. There we have written the access and search commands for opening the database where the exogenous variables have been automatically extrapolated (MAINEXTRAP.FRM), and for creating a database to store the adjusted time series (EXTRAPPED.FRM).

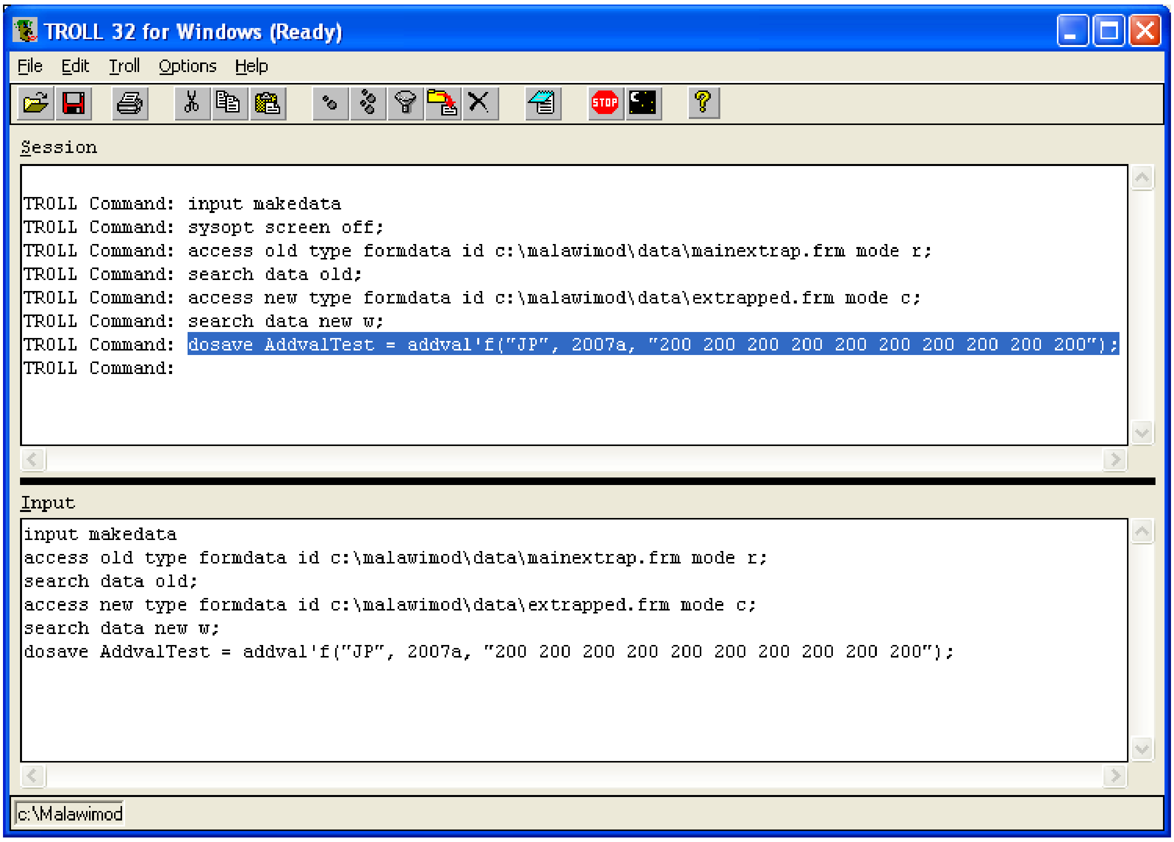
Definitionwise a function returns a value. These functions does not return the adjusted timeseries as one might expect, but a boolean scalar which has the value TRUE if the adjusted timeseries was stored successfully or FALSE if the timeseries cannot be stored according to the writeable search (writeable search has alias `new` in Figure 49). If we print the value saved by the `dosave` command in Figure 49 we get:

```
TROLL Command: do prt.(addvaltest);

ADDVALTEST:
   Boolean scalar:  TRUE

TROLL Command:
```

which indicates that the use of the function was successful.

The adjusted time series will be written to the database according to the search rules that applies. In the example in Figure 49 we created the database EXTRAPPED.FRM because these functions will result in an error if there is no writeable search rule (it will not work for TROLL's save database). Note that in Figure 49 we have used *mode c* to create the database if it does not exist (overwriting any existing ones with the same name).

Figure 49 Using addval as a stand-alone function



```
TROLL Command: input makedata
TROLL Command: sysopt screen off;
TROLL Command: access old type formdata id c:\malawimod\data\mainextrap.frm mode r;
TROLL Command: search data old;
TROLL Command: access new type formdata id c:\malawimod\data\extrapped.frm mode c;
TROLL Command: search data new w;
TROLL Command: dosave AddvalTest = addval'f("JP", 2007a, "200 200 200 200 200 200 200 200 200 200");
TROLL Command:
```

```
input makedata
access old type formdata id c:\malawimod\data\mainextrap.frm mode r;
search data old;
access new type formdata id c:\malawimod\data\extrapped.frm mode c;
search data new w;
dosave AddvalTest = addval'f("JP", 2007a, "200 200 200 200 200 200 200 200 200 200");
```

The function is called with the suffix `'F` to tell TROLL that this is a name of a function and not a variable. We will now turn to the details on these functions.

Appendix 1.3. The addval function

The purpose of the addval function is to insert values to a time series. Here we show how to use the function when we will read the information from a text file. Then the syntax is

```
addval MG 2005a "1000 1000 1000 900 900 900"
```

This line is read from the program (CHDATA.PRG or SHIFT.PRG) that first read the word `addval` that identifies which function to use. Then the other information is interpreted and passed on to the addval function as

`addval var start stop` (input from file)
`bTF = addval'f(var, start, stop);` (as stand-alone function)

where
bTF = return value, boolean scalar TRUE or FALSE
var = MG (variable to adjust)
start = 2005a (start date)
stop = "1000 1000 1000 900 900 900" (values to insert, implicit defines the end date)

Box 3 Effect of using the addval function

Here we see how the function `addval` change the values of a time series. For the variable MG the outcome of the automatic extrapolation is presented in the column named *OLD_MG* in the table. After executing the `addval`-function, reading the following line

```
addval MG 2005a "1000 1000 1000 900 900 900"
```

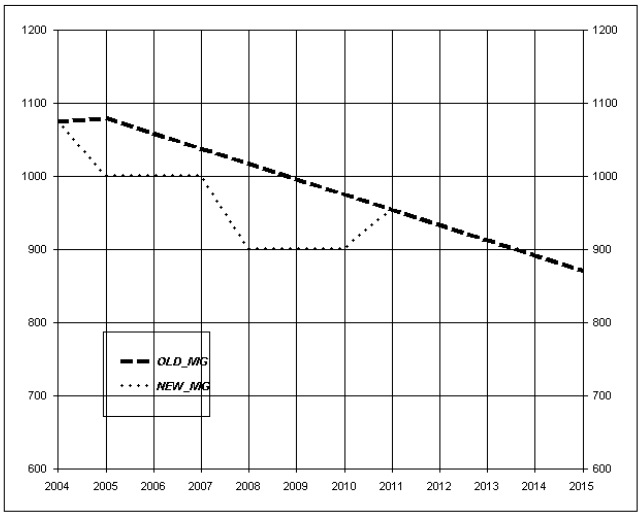
from a file, the result is presented in the column named *NEW_MG* in the table.

First we set the value to 1000 from 2005 to 2007 (including), and then we set the value to 900 for the period 2008 to 2010. Thereafter MG takes its old values.

	OLD_MG	NEW_MG
2004A	1074.6	1074.6
2005A	1079.6	1000
2006A	1058.7	1000
2007A	1037.9	1000
2008A	1017	900
2009A	996.1	900
2010A	975.3	900
2011A	954.4	954.4
2012A	933.6	933.6
2013A	912.7	912.7
2014A	891.9	891.9
2015A	871	871

If the length of the OLD_MG is shorter than what is implied by the start date (for adding values) and the number of values to add, the NEW_MG will be extended so that all values is added. The use of the addval function as in Box 3 is visualized in Figure 50.

Figure 50 The addval function



Appendix 1.4. The adjust function

The `adjust` function change the level of a time series by a percentage and then add a constant. For instance, we may want to increase MG by 2 per cent, and add 200 in the period 2006 to 2012. Obviously, to be able to do this, the values for MG must have been extrapolated before. Apart from the first and the last year we are altering, the growth rate is unchanged. Here we show how to use the function when we will read the information from a text file. Then the syntax is

```
adjust MG 2006a 2012a 3 200
```

This line is read from the program (CHDATA.PRG or SHIFT.PRG), which first read the word `adjust` that identifies which function to use. Then the other information is interpreted and passed on to the `adjust` function as

`adjust var start stop num1 num2` (input from file)
`bTF = adjust'f(var, start, stop, num1, num2);` (as stand-alone function)

where
bTF = return value, boolean scalar TRUE or FALSE
var = MG (variable to adjust)
start = 2006a (start date)
stop = 2012a (end date)
num1 = 3 (percentage to add to new series)
num2 = 200 (constant to add)

For instance, in Box 4, we can see the old value in 2006 is 1058.7. If we add 3 percent and 200 we have: 1058.7*1.03 + 200 = 1290.5. Similarly, in 2007 we get 1037.9*1.03+200 = 1269.

Box 4 Effect of using the adjust function

Here we see how the function `adjust` change the level of a time series by a specified percentage and then add a constant value. For the variable MG the outcome of the automatic extrapolation is presented in the column named *OLD_MG* in the table. After executing the `adjust`-function, reading the following line

```
adjust MG 2006a 2012a 3 200
```

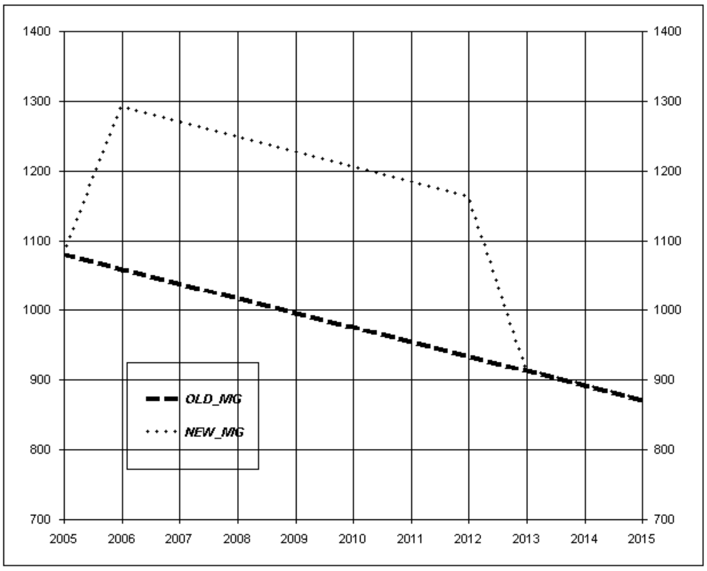
from a file, the result is presented in the column named *NEW_MG* in the table.

We add 3 per cent and then 200 to the series from 2006 to 2012 (including). Thereafter MG takes its old value.

	OLD_MG	NEW_MG
2004A	1074.6	1074.6
2005A	1079.6	1079.6
2006A	1058.7	1290.5
2007A	1037.9	1269
2008A	1017	1247.5
2009A	996.1	1226
2010A	975.3	1204.6
2011A	954.4	1183.1
2012A	933.6	1161.6
2013A	912.7	912.7
2014A	891.9	891.9
2015A	871	871

The use of the adjust function as in Box 4 is visualized in Figure 51.

Figure 51 The adjust function



Appendix 1.5. The extrap function

The purpose of the `extrap` function is to extrapolate a time series with a given growth rate. The values for the specified period will be created if they don't exist. Here we show how to use the function when we will read the information from a text file. In this example we have used the function twice. Then the syntax is

```
extrap mg 2005a 2010a 2
extrap mg 2011a 2020a 1
```

These lines are read from the program (CHDATA.PRG or SHIFT.PRG) that first read the word `extrap` that identifies which function to use. Then the other information is interpreted and passed on to the `extrap` function as

`extrap var start stop num1` (input from file)
`bTF = extrap'f(var, start, stop, num1);` (as stand-alone function)

where
bTF = return value, boolean scalar TRUE or FALSE
var = MG (variable to adjust)
start = 2005a (start date)
stop = 2010a (end date)
num1 = 2 (per cent growth rate to give the time series)

Box 5 Effect of using the extrap function two times

Here we see how the function `extrap` gives the growth rate of a time series. For the variable MG the outcome of the automatic extrapolation is presented in the column named *OLD_MG* in the table. After executing the `extrap` -function, reading the following two lines

```
extrap mg 2005a 2010a 2
extrap mg 2011a 2020a 1
```

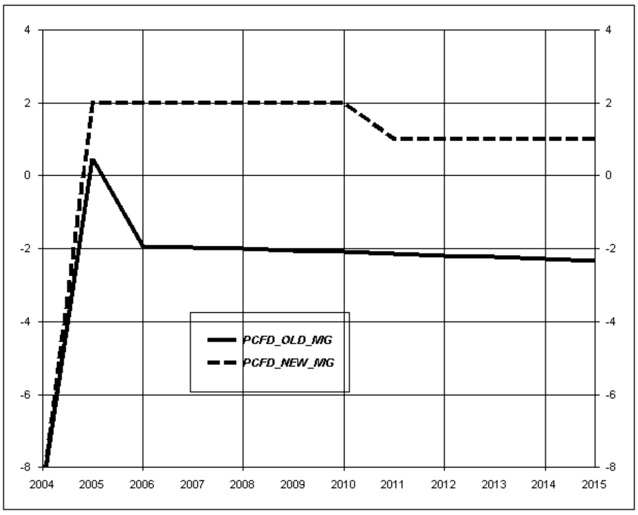
from a file, the result is presented in the column named *NEW_MG* in the table.

First we set the growth rate to 2 per cent from 2005 to 2010 (including), and then we set the growth rate to 1 percent from 2011 to 2020.

Year	OLD_MG	NEW_MG
2004	-8.6	-8.6
2005	0.5	2
2006	-1.9	2
2007	-2	2
2008	-2	2
2009	-2.1	2
2010	-2.1	2
2011	-2.1	1
2012	-2.2	1
2013	-2.2	1
2014	-2.3	1
2015	-2.3	1

The use of the `extrap` function as in Box 5 is visualized in Figure 52.

Figure 52 The extrap function (per cent growth from previous year)



Appendix 1.6. The extrapa function

The purpose of the `extrapa` function is to extrapolate a time series with a given growth rate and a constant added in every period. Here we show how to use the function when we will read the information from a text file. If the data for the period we want to adjust does not exist it will be created. Then the syntax is

```
extrapa MG 2005a 2012a 4 50
```

This line is read from the program (CHDATA.PRG or SHIFT.PRG) that first read the word `extrapa` identifying which function to use. Then the other information is interpreted and passed on to the addval function as

`extrapa var start stop num1 num2;` (input from file)
`bTF = extrapa'f(var, start, stop, num1, num2);` (as stand-alone function)

where
bTF = return value, boolean scalar TRUE or FALSE
var = MG (variable to adjust)
start = 2005a (start date)
stop = 2012a (end date)
num1 = 4 (per cent growth rate to give the time series)
num2 = 50 (constant to add to the time series in each period)

The EXTRAPA.PRG program first calculates the new value for the year 2005. This is done by increasing the value of MG in 2004 by 4 per cent then adds 50, i.e. the new 2005 value (c.f. Box 6) is calculated as 1074.6*1.04+50 = 1167.6. Next this value is increased by 4 per cent and then added 50 by so that we get 1167.6*1.04+50 = 1264.3.

Box 6 Effect of using the extrapa function

Here we see how the function `extrapa` change the time series. For the variable MG the outcome of the automatic extrapolation is presented in the column named *OLD_MG* in the table. After executing the `extrapa`-function, reading the following line

```
extrapa MG 2005a 2012a 4 50
```

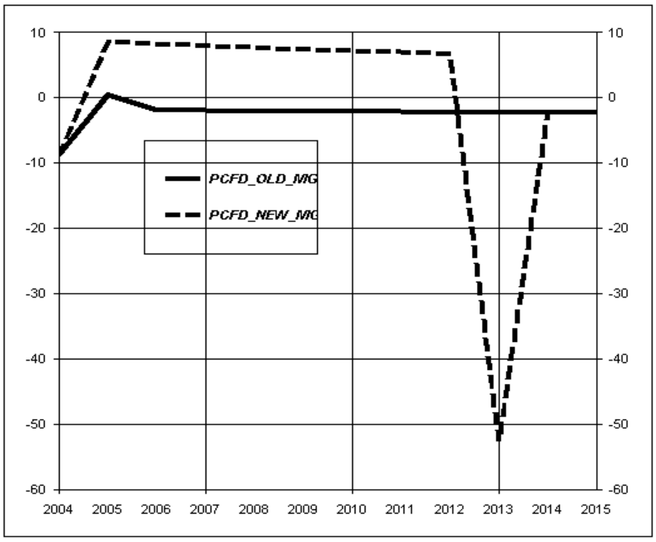
from a file, the result is presented in the column named *NEW_MG* in the table.

We set the growth rate to 4 per cent and add 50 from 2005 to 2012 (including). After 2012 MG takes its previous values if any.

	OLD_MG	NEW_MG
2004A	1074.6	1074.6
2005A	1079.6	1167.6
2006A	1058.7	1264.3
2007A	1037.9	1364.9
2008A	1017.0	1469.5
2009A	996.1	1578.2
2010A	975.3	1691.4
2011A	954.4	1809.0
2012A	933.6	1931.4
2013A	912.7	912.7
2014A	891.9	891.9
2015A	871	871

The use of the `extrapa` function as in Box 6 is visualized in Figure 53 by plotting the growth rates. Note that the effect of adding 50 millions Malawian Kwacha alone (disregarding the given growth rate of 4 per cent) increases MG by about 4.6 per cent in 2005, making the total increase for that year about 8.6 per cent. Note also the sharp drop in the growth rate following the last year of the period for which we edited the original time series. This is because the time series takes its previous values in 2013.

Figure 53 The extrapa function



If we set `num2 = 0` `extrapa` is identical to `extrap`, and if we set `num1 = 0` the result will be to add 50 to the previous years level.

Appendix 1.7. The pcfdadj function

The purpose of the `pcfdadj` function is to change the growth rate with a specified percentage point. For instance, as in this example, we may want to increase the growth rate of MG by 3.5 percentage points in the period 2006 to 2012. Apart from the first and the last year we are adjusting, the growth rate is unchanged, but the levels after the latest period we change will be different because we applies the old growth rates to a new level. Here we show how to use the function when we will read the information from a text file. To change a growth rate the data for the period we want to adjust must already exist. Then the syntax is

```
pcfdadj MG 2006a 2012a 3.5
```

This line is read from the program (CHDATA.PRG or SHIFT.PRG) that first read the word `pcfdadj` that identifies which function to use. Then the other information is interpreted and passed on to the `pcfdadj` function as

`pcfdadj var start stop num1` (input from file)
`bTF = pcfdadj'f(var, start, stop, num1);` (as stand-alone function)

where
bTF = return value, boolean scalar TRUE or FALSE
var = MG (variable to adjust)
start = 2006a (start date)
stop = 2012a (end date)
num1 = 3.5 (percentage points to add to new series)

Box 7 Effect of using the pcfdadj function

Here we see how the function `pcfdadj` change the growth rate of a time series by a specified percentage. For the variable MG the outcome of the automatic extrapolation is presented in the column named *OLD_MG* in the table (both for levels and growth rates). After executing the `pcfdadj`-function, reading the following line

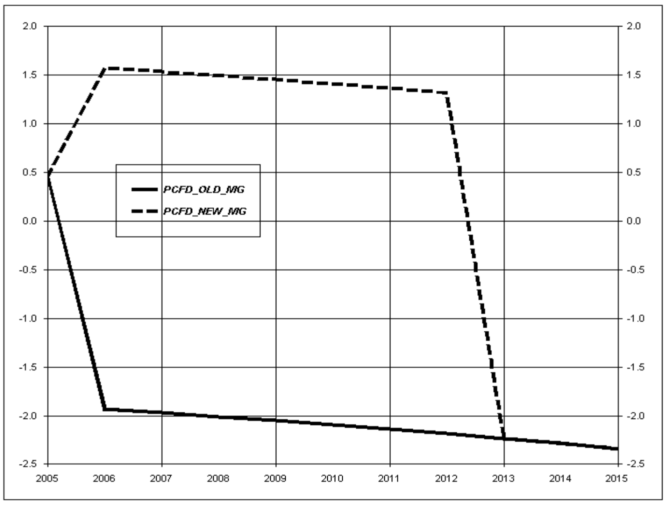
```
pcfdadj MG 2005a 2012a 3.5
```

from a file, the result is presented in the column named *NEW_MG* in the table. We see that the difference between the new and old growth rates are 3.5 percentage points. After 2012 MG takes its old growth rate, but they are applied to new levels.

	Level		Growth rate		Difference
	OLD_MG	NEW_MG	OLD_MG	NEW_MG	
2005A	1079.6	1079.6	0.5	0.5	0
2006A	1058.7	1096.5	-1.9	1.6	3.5
2007A	1037.9	1113.3	-2	1.5	3.5
2008A	1017	1129.9	-2	1.5	3.5
2009A	996.1	1146.2	-2.1	1.4	3.5
2010A	975.3	1162.4	-2.1	1.4	3.5
2011A	954.4	1178.2	-2.1	1.4	3.5
2012A	933.6	1193.7	-2.2	1.3	3.5
2013A	912.7	1167	-2.2	-2.2	4.33E-13
2014A	891.9	1140.4	-2.3	-2.3	-4.11E-13
2015A	871	1113.7	-2.3	-2.3	4.33E-13

The use of the `pcfdadj` function as in Box 7 is visualized in Figure 54. The growth rate of the new series is 3.5 percentage points above the old one.

Figure 54 The pcfdadj function

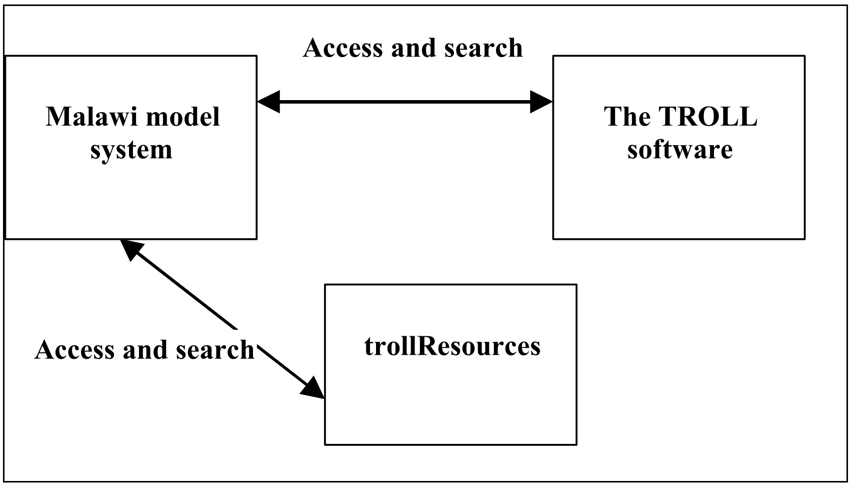


Appendix 2. Initialisation program

Appendix 2.1. Background

When we use TROLL we want to have access to information stored on the computer. For instance we want to read data from databases and write data to databases, and we want to use our own programs and macros to facilitate various tasks. As we have seen before (c.f. Figure 1) the Malawi model system and the TROLL software are not at the same physical location. In addition to this we also use some functions stored in a separate directory. This is illustrated in Figure 55. We have organised the system as to have a main model directory (Malawi model system called `Malawimod`) and a folder containing functions not specific for the Malawi model (C:\TROLLRESOURCES\LIB\). This means there has to be some kind of link between the model system (our files) and the TROLL software.

Figure 55 Components of the model system



Admission to the `Malawimod` directory and sub-directories from the TROLL software is organized through TROLL's access and search commands[4]. For example, every time we want to read from our main historical database we must issue the following statements:

```
access hist type formdata id C:\Malawimod\data\main.frm mode r;
search data hist;
```

Every time we want to use one of our own developed programs or macros, we must implement

```
access prog type disk id C:\Malawimod\program mode r;
search program prog;
```

The access and search commands are used in many files, also inside compiled programs. If we, for some reason, want to change the path components i.e. the disc drive or any of the folder names, it will involve quite a few files to change and programs to recompile. This is also a major

[4] Generally, information management in TROLL depends critically on two lists kept during a TROLL session: The ACCESS list and the SEARCH list. These lists determine the locations where TROLL will store and retrieve data, as well as where TROLL will store and retrieve programs, models, and any other information that it needs.

source of errors, i.e. if we forget to change all the names, or to compile the source code after changing. Of course there is a better way to handle this issue. The idea we have adopted is to store the names of the path components in only one place.

Appendix 2.2. The initial settings

We have chosen to store the path components in TROLL's memory. Every time we start TROLL and chose the Working Directory according to where our model is, these initial settings are read into memory. We will now elaborate on how this is done. When we start TROLL and chose our Working Directory (C:\MALAWIMOD) a file called PROFILE.INP is executed automatically. The automatic execution of PROFILE.INP is a functionality of TROLL to let the user initiate some settings. This can be used for many things. In our system its sole role is to execute a program called PROFILE.PRG that reads our path components into memory. We will not present the source code for this file, PROFILE.SRC, but extract some of its content to explain how it is constructed (to see its content open PROFILE.SRC at the Working Directory). In the following list we have taken out 6 lines from the file for this purpose (the line numbers are only for the sake of reference and is not part of the file).

```
1. _ds_             = hfdirsep(),
2. SystemDrive      = "C:",
3. ProjectDir       = "Malawimod",
4. DataDir          = "data",
5. archive          = SystemDrive||_ds_||ProjectDir||_ds_ ,
6. tsarchive        = SystemDrive||_ds_||ProjectDir||_ds_||DataDir||_ds_,
```

In line 1 we use TROLL to get information from the host computer. We use the function `hfdirsep` to get the character used by the host file system to separate directories. The return value from this function is stored in memory in the variable "`_ds_`" (ds for directory separator). The actual value returned on Windows computer is "`\\`". The reason for returning two backslashes is that a single backslash has a special meaning in TROLL[5]. In line 2 we set the drive, i.e. the letter defining the drive for the model system and the colon, i.e. "`C:`".

In line 3 and 4 we have hard coded the name of our Working Directory and the sub-folder for storing databases. This is the only place where these names are written. In fact, in this file is the only place any path component names are written. In line 3 we read the name of our main directory, or Working Directory, into TROLL's memory, so that the physical name "`Malawimod`" is stored in memory with the name "`ProjectDir`". In line 4 we show how we store the name of one of the sub-folders, the "data" folder as "`DataDir`". In the lines 5 and 6 we show how we use this information to assemble path-names. In line 5, the Working Directory is assembled and stored as "`archive`". The "`||`" is TROLL's syntax for concatenating strings. Lets study line 5 in detail:

We have the string "`SystemDrive||_ds_||ProjectDir||_ds_`"

Where

```
SystemDrive   = "C:" (could be any letter)
_ds_          = "\\"
ProjectDir    = "MalawiMod"
```

[5] In TROLL the backslash "\"is used as a continuation or escape character. If we print "c:\malawimod" TROLL will return "c:malawimod" without a backslash. Two backslashes, like in "c:\\malawimod" will be interpreted as "c:\malawimod". We can see more on this in Chapter 2.3.2 in the on-line manual.

When we put this together we get the string “`c:\\Malawimod\\`” which is interpreted as C:\MALAWIMOD\.

In line 6 we have added the “`DataDir`” variable as well. This means that the string

```
SystemDrive||_ds_||ProjectDir||_ds_||DataDir||_ds_
```

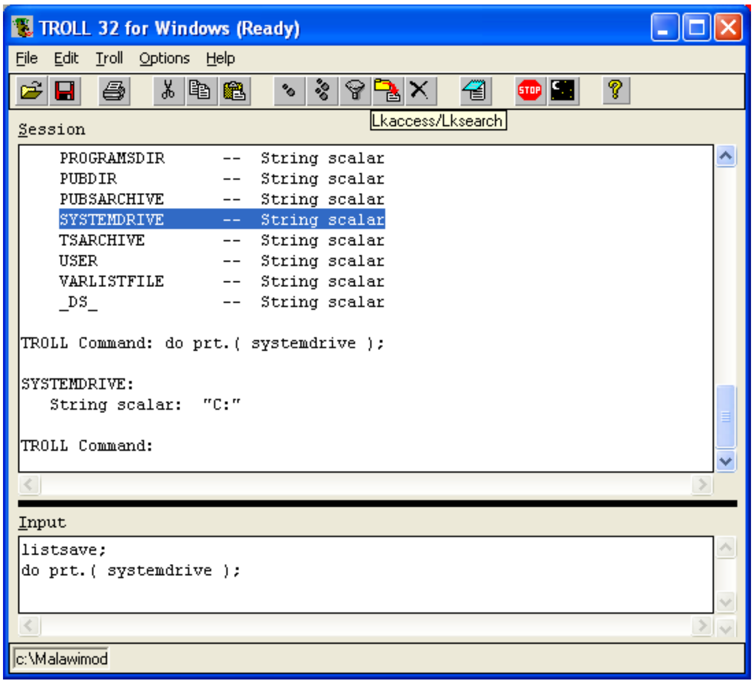
stored in memory as “`tsarchive`”, is assembled as the string "`C:\\Malawimod\\data\\`", and interpreted as C:\MALAWIMOD\DATA\.

Also some other parameters are stored in memory. The default database “MAIN.FRM” is stored as “`defaults`”, and the default model “`malawimod`” is stored as “`model`”. A list of these memory variables, together with their explanations, is presented in Table 14. To see these settings we can type the following commands in the Input window (after you have started TROLL):

listsave;
do prt.(systemdrive);

The first command is to print a list of what is in TROLL’s memory, or in the TROLL’s save data table. The command prints a list to the Session window as in **Feil! Ugyldig selvreferanse for bokmerke.**. If we want to see the

Figure 56 Initial setting for the Malawi model



```
    PROGRAMSDIR     --  String scalar
    PUBDIR          --  String scalar
    PUBSARCHIVE     --  String scalar
    SYSTEMDRIVE     --  String scalar
    TSARCHIVE       --  String scalar
    USER            --  String scalar
    VARLISTFILE     --  String scalar
    _DS_            --  String scalar

TROLL Command: do prt.( systemdrive );

SYSTEMDRIVE:
   String scalar:  "C:"

TROLL Command:
```

```
listsave;
do prt.( systemdrive );
```

value stored in these string scalars we can print their content by executing the second command. This means that there exists variable in TROLL’s memory called “`systemdrive`” which is equal to the string “`C:`”.

Now we must elaborate on how TROLL may retrieve these values from the memory database to actually use them. For example the default model is set to be “`malawimod`” and the default database is set be “MAIN.FRM”. When we run the command *input start;* it actually run commands to open the default database MAIN.FRM and to establish `malawimod` as the current working model (by the command *usemod malawimod;*). Let us look at the relevant lines from the input file START.INP.

```
1. access indata type formdata id &dat2inp DEFAULTDATABASE " " mode r;
2. usemod  &dat2inp MODEL " " ;
```

The line numbers are for references only and are not part of the file. In line 1 we access our default database as specified in the initial setting (PROFILE.SRC). In the access command what follows the "`id`" option should be the physical address for the database. If we print the content of the memory variable `defaultdatabase` we get

```
TROLL Command: do prt.( defaultdatabase );

DEFAULTDATABASE:
   String scalar:  "C:\\Malawimod\\data\\main.frm"

TROLL Command:
```

which is the complete path for our default database. TROLL uses the program `dat2inp` to expand the value of the string data object `defaultdatabase` into the access line syntax, so TROLL interpret this line as:

```
access indata type formdata id C:\Malawimod\data\main.frm mode r;
```

which opens MAIN.FRM for reading.

In line 2 we tell TROLL which model we want to use. The expression is interpreted as:

```
usemod malawimod;
```

If the database MAIN.FRM or the model `malawimod` is not present when executing the START.INP file, an error message will appear.

Table 14 gives the details on the variables stored in memory during a session using our model system.

Table 14 Session variables stored in memory

Variable name	Data type	Value	Comment
ALPHA_LEVEL	Numeric	0.2	Parameter for estimating level in automatic extrapolation
ARCHIVE	String	C:\\Malawimod\\	Working Directory
BETA_TREND	Numeric	0.2	Parameter for estimating trend in automatic extrapolation
COEFARCHIVE	String	C:\\Malawimod\\coef\\	Path for folder containing database with estimated regression coefficients
COEFDATABASE	String	C:\\Malawimod\\coef\\regrcoef.frm	Path for database with estimated regression coefficients
COEFDSET	String	regrcoef.frm	Name of database with estimated regression coefficients
COEFFICIENTDIR	String	coef	Name of folder containing database with estimated regression coefficients
COMMANDARCHIVE	String	C:\\Malawimod\\command\\	Path for folder containing input files
COMMANDSDIR	String	command	Name of folder containing input files

CONSTDATABASE	String	C:\\Malawimod\\coef\\coef.frm	Path for database with coefficients
CONSTDSET	String	coef.frm	Name of database with coefficients
DATADIR	String	data	Name of folder containing time series databases
DEFAULTDATABASE	String	C:\\Malawimod\\data\\main.frm	Path for database with historical timeseries
DEFAULTTS	String	main.frm	Name of database for historical timeseries
IOCOEFARCHIVE	String	C:\\Malawimod\\iocoef\\	Path for folder containing database with input-output coefficients
IOCOEFDATABASE	String	C:\\Malawimod\\iocoef\\io.frm	Path for database with input-output coefficients
IOCOEFDSET	String	io-1999.DAT	Name of database for input-output database
IOCOEFFICIENTDIR	String	iocoef	Name of folder containing input-output database
LIBRARYARCHIVE	String	C:\\trollResources\\lib\\	Path for external Troll programs
LIBRARYDIR	String	lib	Name of sub-folder for external Troll programs
LIBRARYMAINDIR	String	trollResources	Name of main-folder for external Troll programs
LOGFILE	String	C:\\Malawimod\\ Modsys.log	Path for log-file
PROJECTDIR	String	Malawimod	Name of Working Directory
MODEL	String	malawimod	Name of default model
MODELARCHIVE	String	C:\\Malawimod\\model\\	Path for folder containing models
MODELDIR	String	model	Name of model directory
PERIODS_TO_EXTRAP	Numeric	20	Number of periods to extrapolate exogenous variables (for automatic extrapolation)
PROGRAMARCHIVE	String	C:\\Malawimod\\program\\	Path for folder containing programs
PROGRAMSDIR	String	program	Name of folder containing programs
PUBDIR	String	pubs	Name of directory for tables
PUBSARCHIVE	String	C:\\Malawimod\\pubs\\	Path for folder containing tables
SYSTEMDRIVE	String	C:	Name of system drive
TSARCHIVE	String	C:\\Malawimod\\data\\	Path for folder containing time series databases
USER	String	NA	Username
VARLISTFILE	String	C:\\Malawimod\\varlist.txt	Path for variable list
_DS_	String	\\	Directory separator

Appendix 2.3. Changing the initial settings

In this chapter we will go through the process of changing the initial settings. There may be many reasons for doing so. We may want to specify another default model or database, or we might even want to alter the directory structure. Here we will illustrate changing the settings with an example. Suppose we would like to do some testing with our model, and want to leave our original model work unchanged. Suppose we make a copy of the whole model directory MALAWIMOD and call it MALAWIMODTEST. To be able to use the new model directory, we must make it our Working Directory, and change the settings accordingly. Here is a recipe on how to change the Working Directory from C:\MALAWIMOD to C:\MALAWIMODTEST:

1. Copy C:\MALAWIMOD to C:\MALAWIMODTEST (and all sub-folders using a file manager)
2. Start TROLL
3. Choose C:\MALAWIMODTEST\ as Working Directory
4. Open the file PROFILE.SRC in the TROLL-editor
5. Find the line `>> ProjectDir = "Malawimod";`
6. Change to `>> ProjectDir = "Malawimodtest";`
7. Click the “Compile with save” icon
8. Stop TROLL
9. Start TROLL
10. Choose C:\MALAWIMODTEST\ as Working Directory

Now we are ready to use our new “test directory”. To check that the change has been correctly installed type the following in TROLL:

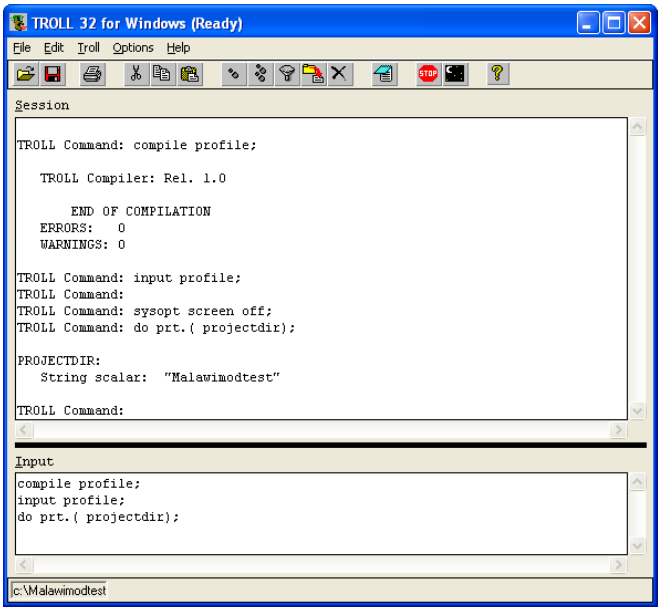
```
do prt.( ProjectDir );
```

The result should be:

```
PROJECTDIR:
   String scalar:   "Malawimodtest"

TROLL Command:
```

The reasons for stopping and starting TROLL is to automatically execute the files reading the new settings into TROLL’s memory. Alternatively this can be done manually. In Figure 57 we compile the PROFILE.SRC file manually, then run the input file PROFILE.INP to manually implement the changes.

Figure 57 Manually implementing new settings



```
TROLL Command: compile profile;

   TROLL Compiler: Rel. 1.0

       END OF COMPILATION
   ERRORS:    0
   WARNINGS:  0

TROLL Command: input profile;
TROLL Command:
TROLL Command: sysopt screen off;
TROLL Command: do prt.( projectdir);

PROJECTDIR:
   String scalar:  "Malawimodtest"

TROLL Command:
```

```
compile profile;
input profile;
do prt.( projectdir);
```

c:\Malawimodtest

If we want to change other settings the process is similar. If we, for instance, have created a new model that we want to use as a default model, we open the PROFILE.SRC file at the Working Directory, find the appropriate line where the default model is specified and change it. We must remember to recompile the source code.

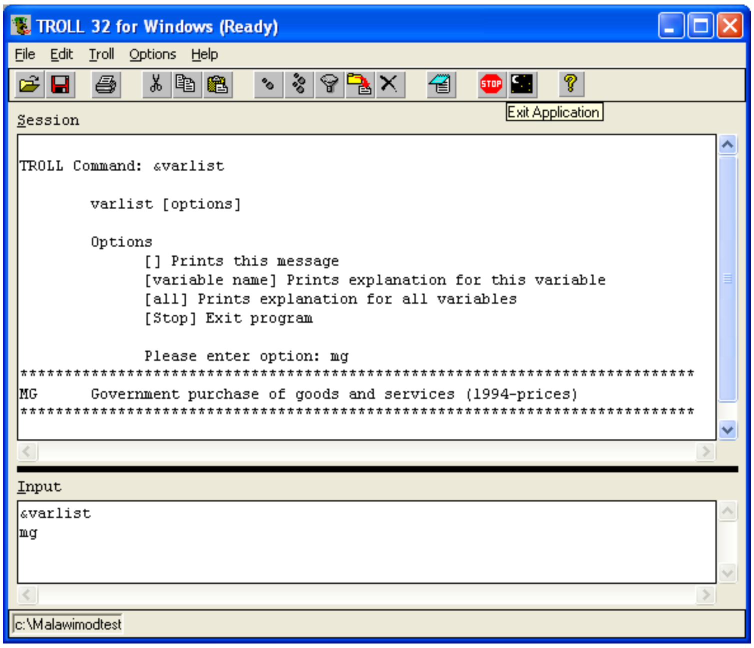
Appendix 3. The trollResources directory

When we use the model system we use many functions and procedures to facilitate the various tasks. Some of these are made especially for handling this model, and are stored at the "C:\MALAWIMOD\PROGRAM" directory. In addition we have developed some other functions that are not specific to this model, and consequently we have stored them elsewhere. This "library" is stored in a directory named C:\TROLLRESOURCES. It has got two sub-directories, called "DOC" and "LIB". The purpose of the "DOC" folder is to collect various TROLL related publications that might be useful. The "LIB" folder contains TROLL functions.

Appendix 4. Maintaining the variables list

We have created a simple program to print the explanation of the symbols in the model. The program is called VARLIST.PRG and is demonstrated in Figure 58. When we execute the program without any parameters, it gives a list of options. They are either to give the name of the variable whose explanation we want to display, or "all" to print the whole list. In this example (c.f. Figure 58) we have typed "*mg*" and the program prints the explanation for this variable.

Figure 58 Using the variables list



```
TROLL Command: &varlist

        varlist [options]

        Options
              [] Prints this message
              [variable name] Prints explanation for this variable
              [all] Prints explanation for all variables
              [Stop] Exit program

              Please enter option: mg
******************************************************************************
MG      Government purchase of goods and services (1994-prices)
******************************************************************************
```

```
&varlist
mg
```

The VARLIST.PRG program reads its information from a text file called VARLIST.TXT on the working Directory. The varlist.txt file has the following format:

C	Total consumption
CA	Current account
CG	Government consumption

The left column with the symbol and the right column with the explanation are separated by a tabulator character. When we introduce new variables in the model we must update this text file.

Recent publications in the series Documents

2004/9 L. Røgeberg, T. Skoglund and S. Todsen: Report on the Project Quality Adjusted Input Price Indicies for Collective Services in the Norwegian National Accounts. Report from a Project Co-financed by Eurostat.

2004/10 A-K. Mevik: Uncertainty in the Norwegian Business Tendency Survey.

2004/11 A.G. Hustoft, J. Linnerud and H.V. Sæbø: Quality and metadata in Statistics Norway.

2004/12 E. Engelien, R. Klæboe and Margrete Steinnes: Neighbourhood sonoscapes. Context sensitive noise impact mapping .

2004/13 P. V. Hansen: Regional electricity spot price responses in Norway.

2004/14 A.G. Hustoft and J. Linnerud: Development of a variables documentation system in Statistics Norway. International Statistical Conference "Investment in the future", Prague, Czech Republic, 6-7 September 2004.

2004/15 J.L. Hass: Compilation of data on expenditure in Environmental protection by businesses. Report to the European Commission DG for Environment.

2004/16 A. Raknerud, J. Rønningen og T. Skjerpen: Documentation of the capital database. A database with data for tagible fixed assets and economic data at the firm level.

2004/17 B.K. Wold D. Roll-Hansen A. Mathiassen and S. Opdahl: A Sustainable Household Survey Based Poverty Monitoring System. A Poverty Monitoring System Based upon Household Survey Estimation of Total Consumption. A Preliminary Paper Asking for Cooperation

2004/18 T. Karlsen, D. Quang Pham and T. Skjerpen: Seasonal adjustment and smoothing of manufacturing investments series from the quartely Norwegian national accounts

2005/1 V. Skirbekk: The Impact of a Lower School Leaving Age and a Later Retirement on the Financing of the Norwegian Public Pension System.

2005/2 H. Utne: The Population and Housing Censushandbook 2001.

2005/3 J. L.Hass and R. Straumann: Environmental Protection Expenditure: Methodological work for the Oil and Gas Extraction Industry. Report to Eurostat.

2005/4 L. Hobbelstad Simpson: National Accounts Supply and Use Tables (SUT) in Constant Prices SNA-NT "SUT/CONSTANT"

2005/5 L. Hobbelstad Simpson: National Accounts Supply and Use Tables (SUT) in Current Prices. SNA-NT "SUT/STARTER"

2005/6 S. Todsen: SNA-NT User's Guide for Supply and Use Tables in Current and Constant Prices.

2005/7 E. Ugreninov, T.M. Normann and A. Andersen: Intermediate Quality Report EU-SILC 2003 Statistics Norway.

2005/8 H.V. Sæbø: Metadata strategy in Statistics Norway. Eurostat Metadata Working Group Luxembourg, 6-7 June 2005.

2005/9 J.L. Hass, K.Ø. Sørensen , K. Erlandsen and T. Smith: Norwegian Economic and Environment Accounts (NOREEA). Project Report 2002.

2005/10 A. Benedictow and T. Harding: Modeling Norwegian balances of financial capital.

2005/11 A.L. Mathiassen,J.B Musoke, P.Opio and P. Schøning: Energy and Poverty A feasibility study on statistics on access and use of energy in Uganda.

2005/12 E. Vinju, R. Strauman, Ø. Skullerud, J. Hass and B. K Frøyen: Statistics on pre-treatment of waste. Pilot study - Norway 2004. Report to Eurostat

2005/13 H. Skullerud, Ø. Skullerud and S. Homstvedt: Pilot study: Treatment of Hazardous Waste. Final report to Eurostat.

2005/14 H. Skiri, B. Strand, M. Talka and H. Brunborg: Selected Documents on the modernisation of the Civil Registration System in Albania Vol. II.

2006/1 O. Andersen og M. Macura: Evaluation of theproject "Modernisation of the Civil Registration System in Albania"

2006/2 T. Åvistland: The problem with a risk premium in a non-stochastic CGE model.

2006/3 Å Cappelen, R. Choudhury and T. Harding: A small macroeconomic model for Malawi.

2006/4 J. Ramm og A. Sundvoll: Translating and Testing the European Health Status Module in Norway, 2005.

2006/5 A.G. Hustoft og J. Linnerud: Statistical Metadata in Statistics Norway. 10s.

2006/6 H.V. Sæbø: Systematic Quality Work in Official Statistics - Theory and Practice

2006/7 H. Skullerud: Methane emissions from Norwegian landfills Revised calculations for waste landfilled 1945-2004. 15s.