Bergholm, Fredrik

Working Paper

The MOSES Manual, Part 1, How to Run the MOSES Model

IUI Working Paper, No. 75

Provided in Cooperation with:
Research Institute of Industrial Economics (IFN), Stockholm

Suggested Citation: Bergholm, Fredrik (1982) : The MOSES Manual, Part 1, How to Run the MOSES Model, IUI Working Paper, No. 75, The Research Institute of Industrial Economics (IUI), Stockholm

This Version is available at:
https://hdl.handle.net/10419/94710

A list of Working Papers on the last pages

No. 75, 1982

THE MOSES MANUAL

Part 1.

HOW TO RUN THE MOSES-MODEL

by

Fredrik Bergholm

December, 1982

CONTENTS Page

Preface

MOSES is short for "Model for Simulating the Economy of Sweden". Different versions of the model have been used within the institute for about five years by now. A number of simulation experiments have been performed*. The whole model is written in the programming language APL. The present version of the model is installed on a computer in Bergen, Norway. This has come about through cooperation with IØI (Bergen), the industrial institute for economic research in our neighbouring country.

For some time there has been a demand for a full documentation of the current version of the model. "The MOSES Manual" fulfils one part of this request. Anyone interested in a large-scale simulation model of this kind needs to get acquainted with the techniques involved in starting up (initializing) and running the model. Experiments have shown that the initialization procedure, which constructs an initial state of the model economy, is crucial indeed, for the behaviour of the model. Part 2 of the MOSES Manual (not included in this

---

* An earlier version is described in full detail, in Eliasson (1978, abbreviated in (1981)). A new, updated presentation of the full model plus a complete bibliography will be presented in Eliasson (1983).

Documentation on the economic contents of MOSES is complete for an earlier version which is still quite accurate as far as the core micro-to-macro machinery is concerned. An important addition is the individual firm purchasing process, which is described in this manual, Part 2. The need for a full, updated documentation, should however be remedied in a forthcoming research report, Eliasson (1983).

paper) has been devoted to describing this initialization process*.

This part of the MOSES Manual describes how to actually run the model. Previously, only a few persons knew how to handle the machinery. From now on this should have been remedied. The intention behind part 1 is that anyone - following the instructions - should be able to start up and run the model.

Part 1 of the manual is mainly a cook-book. The user can run the model without knowing what it is all about. To be able to carry out meaningful analytical experiments, however, one needs a rather deep understanding of the model and the initialization itself, which requires a considerable and timeconsuming effort on the part of the user.

---

* A more detailed description of the micro (firm) database can be found in Albrecht-Lindberg (1982).

Section 1 Introduction

The model consists of two parts, the model-simulation itself and the initialization procedure.

Before one can start a simulation of the Swedish economy with the "micro-to-macro model" MOSES, one has to initialize a vast number of variables.

"Initialization" means (mainly) that three kinds of variables are given values.

(1) Variables for 1976 (the earliest year for which a complete micro-to-macro data base exists) needed to start up the model.
(2) Variables needed to determine the future of certain variables which get their values irrespective of what happens during the simulation.
(3) Certain constants.(Some of these are parameters affecting for example the behaviour of firms.)

Let us see how the initialization and simulation can be performed on the DEC20-computer in Norway. We assume that you are sitting in front of a terminal equipped with a key board with APL-symbols. To be able to run the model one needs some elementary knowledge of APL. On the following pages we will therefore try to give a mini-lesson in APL. For more detailed information, see the APL reference manual.

1.1 The concept **workspace** and system commands

An APL **workspace** consists of functions and variables. The user enters functions and variables in the workspace and can save them for future usage.

A system command in the APL language starts with a right-hand bracket. One needs system commands to handle APL workspaces. The most frequently used system commands are:

) **LOAD** workspace name
) **CLEAR**
) **SAVE**

The **)** **LOAD-command** loads a previously saved workspace into the computer memory. Thus all functions and variables are instantaneously transferred to computer memory. In APL there is in general no program reading variables from a data base into computer memory. Instead one makes a **)** **LOAD**-command.

Example:

) **LOAD** PROV

The workspace PROV is loaded into the computer memory. If you wish to look at a variable in the workspace, just write the name of the variable. If you want to add a variable just write for example:

a←10.

You have then added a new variable **a** which has the value 10.

If you want to save the extended workspace with new or changed variables (for example) simply write:

) SAVE

If you just want to erase computer memory and start all over again write:

) CLEAR

The **) CLEAR**-command erases the workspace which is in the memory at that particular moment. The workspace in the library is of course unchanged. Note that the **) SAVE**-command destroys the old version of the workspace in the library.

Example:

We want to look at the workspace containing the MOSES-model. This workspace lies in the library and is called MOSES.

Perform:

(1) **) LOAD** MOSES
(2) we look at some variables or functions
(3) **) CLEAR**

A function is a program in APL. How to write and read functions is not described here, see instead the APL reference manual. To run an old experiment you don't need to be able to write functions, but if you wish to make new experiments this is necessary!

The most common system commands can be found in the appendix to the next section. A full description can be found in an APL-manual.

Information about the "login procedure", e.g. how to get in touch with the computer in Norway, can be received from authorized persons at IUI.

Section 2 How to run the initialization (jan.1982) and how to start the model

2.1 Overview

We assume that you have logged in and that you have access to:
1) the APL-language
2) Workspaces containing the micro-to-macro model, which are:

MOSES (=the model itself)
,INIT (=the initialization procedure)
,MACRO (=macro data base, for 1976)
,SI76 (=micro data base, for 1976)
,VLISTS (=variable names, initialized variables)
,FUNCTI, ISTART,
MSTART (=miscellaneous workspaces in connection with initialization etc).

One can check that all workspaces needed are in the (IUI-) library by doing the system command:

) LIB

Then all saved workspaces are listed.

It might also be a good idea to perform the **) MAXCORE**-command in the beginning. This command is described in an appendix to this section, and extends maximum available space in the computer memory.

To start the model do:

```
) LOAD INIT                (step 1)
  START NR1                (step 2)
) LOAD MOSES               (step 3)
  UPDATEMOSES NR2          (step 4)
  RUNEXP N                 (step 5)
) SAVE                     (step 6)
```

Step 1+2 = initialization of the model, which means that start-up variables for 1976 are given values etc.

Step 3+4 = update the MOSES model-code to it's latest version

Step 5 = run the model.

Step 6 = save the result.

NR1 NR2 and N are chosen numbers.

N = number of years to simulate (run the model)

NR1 = number of **initialization version**

NR2 = number of **experiment version**

For a definition of "**initialization version**" and "**experiment version**", see the following pages.
This 6-step procedure to run the model will be explained in more detail below.

In what follows, we will write system commands and function calls (for example the function **START**) in boldface letters, whereas parameters to the functions (for example NR1, NR2, N above) or workspace-names (for example INIT) will be written in ordinary letters. Note that there must be at least one blank between a function call and a parameter!

2.1.1 "Initialization version"

Definition:

Instead of changing the original initialization version (jan 1982), one puts all changes in a separate function. A new initialization version is made by using the original initialization program (workspace INIT) and by making a function **ISTARTXX** where all changes are defined. XX is the number of the initialization version. **ISTARTXX** should be put into workspace ISTART before doing anything else.

Example:

Make a function **ISTART4** where all changes are defined. Put it into workspace ISTART by using the system commands Load and Save. Then perform:

```
) LOAD INIT          (step1)
START    4           (step2)
```

Thus, initialization version 4 has been run, and the result is saved (automatically) in workspace R4.

How to make new initialization versions is described in section 3. If you just wish to repeat an old experiment just check that the proper **ISTARTXX**-function is stored in the ISTART-workspace.

The reason for this somewhat cumbersome way of labelling the experiments is that it is extremely difficult to keep track of changes in a program (in this case the initialization program **START**), if one were allowed to make small modifications all the time. If one works in the fashion outlined above, the original initialization code is unchanged, and all modifications thereafter are defined by small modules called ISTARTXX. To be more spe-

cific, the original initialization program is unchanged before a new initialization. During the initialization, the initialization program is updated* with the changes defined in an ISTARTXX-module. For more details see section 3.

* A program can update another program during execution in the APL-language. This is a somewhat unusual feature for a programming language.

2.1.2 "Experiment version"

Definition:
Instead of changing the original model (version 1978), one puts changes of the model, for experiment-purposes, in a separate function. A new experiment version is made by using the original model program (workspace MOSES) and by making a function **MSTARTXX**, where all changes connected with the specific experiment are defined. **XX** is the number of the experiment version. **MSTARTXX** should be put into workspace MSTART before doing anything else.

Example: (We extend the example given above on the previous page)

Make a function **MSTART8** where experiment-specific changes are defined. Put it into workspace MSTART.

Then perform:

```
) LOAD INIT                (step 1)
  START      4             (step 2)
) LOAD MOSES               (step 3)
  UPDATEMOSES      8       (step 4)
```

How to make new experiment versions is described in section 3. If you just wish to repeat an old experiment, simply check that the proper **MSTARTXX**-function is stored in the MSTART-workspace.

Note that each experiment is uniquely determined by the lines in a **MSTARTXX**-function, provided that the same indata is used. Note also that the above procedure means that one can use a certain set of indata and make a large number of different experiments with these. Step 2 yields output from initialization (= input to the model). By varying the

parameter to **UPDATEMOSES** (in this example 8) one can achieve different experiments with the same indata to the model.

One point ought to be clarified in this context. The word "experiment" in our concept "experiment version" means that we are experimenting with the model itself, for example introducing a certain micro or macro behaviour. One can also make experiments by varying input to the model by using different "initialization versions". Some parameters produced by the initialization procedure affect the micro behaviour of firms. So it is equally natural to keep the **MSTARTXX**-function constant and vary the **ISTARTXX**-functions. The difference between an **ISTARTXX**-function and a **MSTARTXX**-function is that the former can't change lines in the model.

2.2 Comments on the 6-step procedure to start and run the model

Step 1 The initialization program is loaded into computer memory

Step 2 "**START** XX" starts the initialization. **ISTARTXX** should be stored beforehand in workspace ISTART. The result from the initialization is stored in workspace RXX. XX=number of initialization version. Step 2 takes about 5 minutes. The monitor prompter resumes the original position when this step is ready.

Step 3 The model itself is loaded.

Step 4 "**UPDATEMOSES** XX" updates the model from the 1978-version to the 1982-version and makes experiment-specific changes in the model code. **MSTARTXX** should have been stored beforehand in workspace MSTART. **XX**=number of experiment version. The monitor prompter resumes the original position when this step is ready.

Step 5 The model is run for N years. This takes about 10 minutes per simulated year. The monitor prompter resumes the original position when the run is over.

Step 6 The simulation-result is saved.

During step 2 the program asks whether one wants to see the input-output matrix or not (described in part 2 of the manual). Answer, yes or no.

During step 4 a question comes up, where the user should give the name of the workspace from which the result of the initialization should be fetched. The name is R1 if you used initialization version 1, R2 if you used initialization version 2 etc.

IMPORTANT NOTE: If you already, (for example, some days before) have made the initialization (steps 1+2) you can start with step 3 at once. This might be a convenient way to work.

The simulation result is not stored automatically. The user has to make the **) SAVE**-command himself. (step 6). The name of the simulation result is SXXVYY, where XX=number of experiment version, and YY=number of initialization version. S stands for simulation result and V for version. We conclude by giving two examples of a MOSES-run.

Example 1:

```
) LOAD INIT
START 1
) LOAD MOSES
UPDATEMOSES      7
RUNEXP      10
) SAVE
```

Initialization version 1 and experiment version 7 are made. The model is run for 10 years. The result from the initialization is stored in workspace R1. The result from the model-simulation is stored in workspace S7V1.

Example 2:

```
) LOAD MOSES
UPDATEMOSES      7
RUNEXP      10
) SAVE
```

The initialization is already made. The same things as in example 1 are being made. The user is asked for the name of the workspace where the result from the initialization is stored (for example R1 if initialization version 1 is used). The result from the simulation is stored in an workspace called S7V1.

IMPORTANT:
If one wants to look at the contents of a workspace, one makes the **) LOAD**-command. There is one exception, though.

To be able to look at the result of the initialization, RXX, after having performed steps 1 and 2 in the 6-step procedure, one has to do the **) COPY**-command.

Thus: **) COPY** R1 must be done to look at workspace R1, instead of **) LOAD** R1. (If you try to do the Load-command instead, a lot of nonsense will be printed out, for technical reasons.)

A complete list of variables coming out from initialization can be found in part 2 of the MOSES manual.

2.3 Batch-job

So far we have assumed that the user of the model makes the commands (the 6-step procedure to start and run the model) actually sitting in front of the computer terminal, doing so called time-sharing. This is by no means necessary.

The same commands can be written in a so called "command-file" beforehand. To perform a run on the computer in this way is called a batch-job. It is very convenient because the program can be run during low cost time (for example) during the night, without anybody at the computer terminal. The terminal need not even be switched on.

On the DEC20 computer (in Bergen) a batch-job is done thus:

Make a "command-file" (using the editor on the DEC 20). In the command-file prov.CTL we have:

```
monitor-prompter APLSF
* TTY
* )MAXCORE        352
* )LOAD       INIT
* START      4
* yes
* )LOAD        MOSES
* UPDATEMOSES          8
* R4
* RUNEXP        10
* )SAVE
* )MONITOR
```

The file could have any name, but the extension must always be CTL. In this example we called the file "prov.CTL". A DEC20-monitor command (the command **APLSF**)* should start with a monitorprompter (a helix-shaped symbol) in a commandfile. An APL-system command should start with an asterisk. More information about batch-jobs can be found in "DEC-SYSTEM Batch Reference Manual".

The batch job is ordered by doing the DEC20-command:

submit prov.CTL

One could add extra instructions to the operator, by writing things after "slashes", according to a syntax (see the Batch Ref. Manual).

We found the following extra instructions useful in practical work:

submit prov.CTL/after:24:00:00/TIME:03:00:00/
restart: YES/PAGE:300

The result from a batch-run is stored on a file called "prov.log". This file contains the output which would have been typed on the screen if the job had been done as time-sharing.

A disadvantage with batch-jobs is that if anything (no matter how trifle) goes wrong during the model-run, nothing can be done about it. The program is simply interrupted. The batch-job has to be connected and repeated.

---

* There are different versions of APL, one of which is APLSF, working on the DEC 20-system.

APPENDIX

Section 2

"System commands"

) LOAD workspace-name
Loads the workspace into computer memory.

) CLEAR
Clears the computer-memory.

) MAXCORE 352
If one has too little space in the computer-memory, one has to ask for more space. The command above gives the user maximum available space. One can choose any number between 6 and 352 (pages). 1 page ≈ 1/2 Kwords.

) WSID
This is a question. The computer answers by telling the user the name of the workspace which is in the computer-memory right now.

) WS name
Changes the name of workspace to the name written after WS.

) SAVE
Saves the workspace under current workspace-name.

) FNS
Lists all functions in the current workspace.

) VARS
Lists all variables in the current workspace.

) COPY workspace-name
A copy of the workspace is added to the workspace one is working with.

The system commands might differ somewhat on different computers. The commands mentioned above are used on DEC20 in Bergen, Norway.

Section 3 How to make new initialization versions and new experiment versions (simulation-variants)

3.1 New initialization versions

Do:

) LOAD ISTART (step 1)

This loads the workspace ISTART, where all previous initialization versions are stored (**ISTART1**, **ISTART2**...).

Make a function

ISTARTXX (step 2)

XX=number of initialization version.

How to make a function is desribed in the APL reference manual.

Save the extended work-space ISTART.

) SAVE (step 3)

ISTART1 and **ISTART2** on the next page show the pattern to be followed when making an **ISTARTXX**-function.

Comment lines in the APL-language start with a very particular symbol, the so called cap null-symbol. It appears frequently in the initialization-code (see for example section 5) and looks like an A which is smaller and more smooth than an ordinary A. For typographical reasons we write this symbol as a boldface **A** in the examples below.

Example 1:

```
    ∇   ISTART1
[1]   A TEST1
[2]   A FREDRIK B
[3]     SYNTH∆FIRMS← 8 16 18 8
[4]   A
    ∇
```

Example 2:

```
    ∇   ISTART2
[1]   A TEST
[2]   A JANUARI 1982
[3]   A FREDRIK B
[4]   A
[5]     SYNTH∆FIRMS← 8 16 18 8
[6]    'MARKETS∆DATA'  MODSUBST  'GAMMA←ωGAMMA←0.5'
[7]    'MARKETS∆DATA'  MODDEL    'KSI←'
[8]    'MARKETS∆DATA'  MODADD    'NITER←ωKSI←0.3'
    ∇
```

Note: If you by mistake put in an extra blank somewhere between the '-signs, you could fail to make the proper changes.

GAMMA, KSI och NITER are three parameters in the MOSES-model. **MARKETS∆DATA** is the name of a sub-function in the initialization procedure, (a sub-function to the function **START**). To be able to make changes in the initialization one has to get well acquainted with the initialization program, which is described in part 2. Many parameters which guide firm-behaviour are given their values in the sub-function **MARKETS∆DATA** which is easy to read. One can without difficulty change such parameters according to the pattern above. The examples above are explained in detail on the following pages.

ISTART1:

No changes in the initialization-program are made. The line "synth△firms ← 8 16 18 8" is compulsory in any **ISTARTXX**-function. This line means that there will be 8 synthetic* firms in sector 1, 16 synthetic firms in sector 2, 18 in sector 3 and 8 in sector 4. Thus, this line tells how many synthetic firms will be created during the initialization procedure.

ISTART2:

The APL-functions **MODDEL**, **MODADD** and **MODSUBST** are used to change lines in the initialization-program. The changes will take place when the ISTARTXX-function is called, and this happens in the very beginning of the initialization.

Thus: Line 6, example 2, means that we will change a line in the sub-fuction **MARKETS△DATA** (see appendix C, part 2). The text before **MODSUBST** (All textstrings should stand between '-symbols) tells the name of the function where the changes are to be made. The text after **MODSUBST** tells what line to be changed and defines the new line. The beginning of the old line stands before the "omega-symbol" (ω) and the new line after this symbol. **MODSUBST** deletes the old line beginning with "GAMMA" and puts in the new line: GAMMA ← 0.5. **MODADD** works like **MODSUBST**, with one exception. **MODADD** does not delete the old line. The new line is put immediately after the old line. Line 7

---

* artificial firms,, which define the difference between macro data (national accounting) and micro data (real firms). See Albrecht-Lindberg (1982).

means that we will delete any line in the sub-function **MARKETSΔDATA** beginning with KSI ←.

3.1.1 Summing up

Syntax:
'function-name' **MODSUBST** 'old line ω new line'
'function-name' **MODADD** 'old line ω new line'
'function-name' **MODDEL** 'line'

Don't take too many letters! This may cause overflow during execution. Error: Workspace full. Don't take too few letters! It might be ambiguous what line you're referring to. Any line (in the function you're looking at) with the same string of letters (in the **beginning** or in the **middle** of the line) might be affected by the **MODSUBST**-, **MODDEL**- or **MODADD**-call.*

3.2 New experiment versions

Do:
) LOAD MSTART (step 1)
This loads the workspace MSTART, where all previous experiment versions are stored (**MSTART1**, **MSTART2**,...). Make a function
MSTARTXX (step 2)
) SAVE (step 3)

On the next page you can see an example of a **MSTART**-function. **MODSUBST**, **MODDEL** and **MODADD** are used in the same way as was done above, in connection with new initialization versions.

---

* Note: It is not allowed to change the function where you are at the moment. Thus **MODSUBST**, **MODDEL**, **MODADD** cannot make changes in the function **START**.

An example of a MSTART-function:

```
     V MSTART1
[13]   A
[14]   AEXPERIMENT
[15]   A*********************
[16]     'INVFIN'  MODSUBST  'QDIV←ωQDIV←0.6×QTAX'
[17]   A
       V
```

Note: **A** is a symbol starting comment-lines, for typographical reasons written as a boldface A.

In a sub-function, called **INVFIN**, in the MOSES-workspace one line is altered.

The new line

QDIV ← 0.6 • QTAX

means that each firm will, in this particular experiment, pay dividends to the household sector amounting to 60 percent of the corporate tax.

The changes in the MOSES-program take place when the function **UPDATEMOSES** is called. The MSTART-function is, namely, called on a line in the function **UPDATEMOSES**.

If you wish to check that the changes in the MOSES-program have been performed correctly, list the functions you are interested in (in the example above the function **INVFIN**) after the call of the function **UPDATEMOSES** (step 4 in the 6-step procedure presented in section 2). How to read (list) functions is described in the APL reference manual.

Section 4 How to add new model-variables

Say that you have a new variable I which you wish to give as input to the model. The variable I is to be given a value in the initialization procedure.

It is not enough just to add this variable to the initialization-program in a **ISTARTXX**-function, for example with a "MODADD-line".

You must also add the name of the variable to a variable-list in workspace VLISTS before making the initialization. All output-variables from initialization should be listed in this workspace. If this is not done, the variable I will be deleted during the initialization procedure, since it is not mentioned among the output-variables in a variable-list in workspace VLISTS. Thus, this system forces the user to mention all new output-variables from initialization.

There are 6 variable-lists in workspace VLISTS, and the user can extend 5 of them, namely:

Variabelgrupp 1	(=exogenous variables)
Variabelgrupp 2	(=endogenous variables)
Variabelgrupp 3	(=constants)
Variabelgrupp 4	(=indices, parameters of a technical nature)
Variabelgrupp 5	(=miscellaneous)

To add the variable I to a variable-list in workspace VLISTS do:

```
) LOAD VLISTS                                    (step 1)
Variabelgrupp 1 ← variabelgrupp 1, ' I '         (step 2)
) SAVE                                           (step 3)
```

Note: We assumed above that I was an exogenous variable, and that was why variabelgrupp 1 was used. The division of variables into 6 groups is purely for book-keeping purposes. If you put a new variable in the wrong group, nothing will happen, but the book-keeping will be messed up.

Note: The text string between the '-signs in step (2) must begin with a blank!

Section 5 Description of the function UPDATEMOSES

UPDATEMOSES is the function doing changes in the model (workspace MOSES) itself, and is step 4 in the 6-step procedure presented in section 2.

The function is documented below, with the APL-code itself. It should be noted that this documents all permanent changes in the model program since 1978.*

In **UPDATEMOSES** four functions are called:

1) **PREPARE∆RUN**	(Fixes headlines etc for printing out simulation-results)
2) **PERMANENT∆CHANGES****	(Permanent changes in the model-program since 1978)
3) **MOSES∆VARIANTS****	(Larger permanent changes in the model-program since 1978)
4) **MSTARTXX**	(The experiment version function)

Note: The MSTART-function should never be used for permanent changes of the model.

The model-program is updated in this fashion, so as to make it possible to repeat old experiments from 1978 and onwards. Another reason for this updating procedure is that the changes are clearly

---

* For a complete understanding of these changes one needs, however, the model program itself, the so called MOSES code, which will be presented in another part of the MOSES manual.

** The changes in **PERMANENT∆CHANGES** are maybe more permanent then **MOSES∆VARIANTS**, that is why they are separate.

defined, and could be checked up by anyone wishing to convince himself that the changes have been properly done.

Section 5 Description of update MOSES

```
    ∇  UPDATE∆MOSES NUM
[1]    ⍝THIS FUNCTION DOES:
[2]    ⍝(1): PREPARES HEADLINES ETC. FOR PRINT-OUT FROM MOSES-RUN
[3]    ⍝(2): MAKES CHANGES IN THE MOSES PROGRAM FROM 1978
[4]    ⍝
[5]     PREPARE∆RUN
[6]     PERMANENT∆CHANGES
[7]     MOSES∆VARIANTS
[8]    ⍝EXPERIMENT-MODULES IN WORKSPACE MSTART.
[9]    ⍝THEY ARE CALLED MSTARTXX WHERE XX IS THE NUMBER IN
[10]   ⍝THE CALL 'UPDATEMOSES XX '    .
[11]   ⍝
[12]    'GIVE THE NAME OF THE WORKSPACE WITH START-'
[13]    'VALUES FROM INITIALIZATION (FOR EXAMPLE R1 ETC.) :'
[14]    INITWORKSPACE←⍞
[15]    ⍎')COPY ',INITWORKSPACE
[16]   ⍝THIS LINE FETCHES INDATA FROM INITIALIZATION...
[17]    PREPARE2
[18]    ⍎')COPY MSTART  MSTART',⍕NUM
[19]    ⍎'MSTART',⍕NUM
[20]   ⍝LINE ABOVE MEANS THAT MSTARTXX IS EXECUTED.
[21]   ⍝
[22]   ⍝ IF YOU WANT MARKET TIME-SERIES RESULTS TO BE PRINTED OUT
[23]   ⍝ DURING SIMULATION REMOVE COMMENT ON NEXT LINE...
[24]   ⍝ TRACE1
[25]   ⍝
[26]   ⍝TRACE1 PRINTS OUT TIME-SERIES RESULTS...
[27]   ⍝
    ∇
```

Section 5 Description of update MOSES

```
    ∇  PERMANENTΔCHANGES
[1]    ⍝ 'PERMANENT' CHANGES IN MOSES:
[2]    ⍝
[3]     'MARKETΔCONFRONT' MODADD 'PTωQPURCHG←QPURCHG×PT[⍳10]×(1÷(QWG÷WGΔR
       EF))÷100'
[4]    ⍝CORRECTION OF DEFLATOR FOR GOVERNMENT'S PURCHASES
[5]    ⍝
[6]     'INVFIN' MODSUBST 'K1BOOK←K1BOOKωK1BOOK←K1BOOK-QDEPRBOOK←0⌈QREVLR
       HOBOOK×K1BOOK←QINV+K1BOOK'
[7]    ⍝ BOOK-VALUE SHOULD NOT BE UPDATED WITH INFLATION
[8]    ⍝
[9]    ⍝ CHANGES TO VECTOR-FORMAT ON MAX/MIN/OPT-STO MADE PERM.
[10]   ⍝ CORRESPONDING FOR MAX/MIN/OPT-IMSTO IN FN.'INDΔPURCHΔSHARES'
[11]   ⍝
[12]   ⍝
[13]    'PLANQREVISE' MODSUBST 'QIMQ←0⌈ω⍝ QIMQ-LINE MOVED ABOVE IMSTO'
[14]    'PLANQREVISE' MODADD 'QQ←QQωQIMQ←0⌈((⍉IO)[MARKET;]MULT7 QPLANQSAV
       E)+(OPTIMSTO-IMSTO)÷4×TMIMSTO'
[15]   ⍝
[16]    EXOΔQDWG←''
[17]   ⍝THE Z-SECTOR IS OBSOLETE AND ISN'T ANY LONGER USED.
[18]   ⍝THE FOLLOWING LINES DELETE OR CHANGE LINES SO THAT
[19]   ⍝THIS SECTOR IS IGNORED.
[20]    'LABOURΔMARKET' MODDEL 'ZLABOUR'
[21]    'MARKETΔCONFRONT' MODSUBST 'PT←QωPT←QPRELPDOM,(QPDOM[IN]×1+QDPIN)
       ,1'
[22]    'HOUSEHOLDΔINIT' MODSUBST 'INMONEYωINMONEYHH←QTRANS+QINPAY+QTDIV+
       (SUM2 L×QW÷4)+(LG×QWG÷4)+(QINTH←NH+.×RIH×WH÷4)'
[23]    'HOUSEHOLDΔINIT' MODSUBST 'QTWS←(LGωQTWS←(LG×QWG÷4),SUM2 L×QW÷4'
[24]    'HOUSEHOLDΔINIT' MODSUBST 'QTI←QTDIωQTI←QTDIV+QTRANS+((+/QTWS)-QW
       TAX)+QINTH'
[25]    'LUUPDATE' MODSUBST 'LF←ωLF←LU+LG+SUM2 L'
[26]    'QΔEXO' MODDEL 'TXVAZ←'
[27]    'LABOURΔUPDATE' MODSUBST 'RU←RU+QωRU←RU+QCHRU←(LU÷LU+LG+SUM2 L)-R
       U'
[28]    'DOMESTICΔRESULT' MODDEL 'QPZ←'
[29]    'FINALQPQSQM' MODDEL 'QMZ←'
[30]    'FINALQPQSQM' MODDEL 'QCHKZ←'
```

Section 5 Description of update MOSES

```
[31]    'FINALQPQSQM' MODDEL 'QDIVZ←'
[32]    'INVFIN' MODSUBST 'QCTAX←(SU⍵QCTAX←(SUM2 QTAX)'
[33]    'YΔCOUNTRYΔTOTAL' MODDEL 'CTΔCHKZ←'
[34]    'DOMESTICΔRESULT' MODDEL 'QSZ←'
[35]    'INDIRECTΔTAXES' MODSUBST 'QVATAX←⍵QVATAX←(TXVA2×QPURCHG+QSPEMKT,
      IN;]+.×NH),0'
[36]    'YΔINDUSTRYΔTOTAL' MODSUBST 'LTOT←⍵LTOT←LG+SUM2 L'
[37]  ⍝
[38]    VARIANTS←VARIANTS,' PERMANENT CHANGES 1980-81 '
    ∇
```

Section 5 Description of update MOSES

```
    ∇  MOSES∆VARIANTS
[1]     NEG∆IMSTO
[2]     IND∆PURCH∆SHARES
[3]     POSITIVE∆NET∆WORTH
    ∇


    ∇  PREPARE∆RUN
[1]     LOEPNR←NUM
[2]     DATUM
[3]     DSCR←'**** EXPERIMENT ',⍕NUM
[4]     ⎕PW←120
[5]     ⍝')MAXCORE 352'
[6]    ⍝MAXIMUM CORE IN COMPUTER-MEMORY...
    ∇


    ∇  PREPARE2
[1]    ⍝ THIS LINE GIVES THE WORKSPACE A NEW NAME,
[2]    ⍝ FOR EXAMPLE  S3V7
[3]    ⍝
[4]     ⍝')WSID S',(⍕NUM),'V',1↓INITWORKSPACE
[5]     DSCR←DSCR,'  ,ISTART=  ',1↓INITWORKSPACE
[6]     DSCR←DSCR,'  *****'
    ∇


    ∇  DATUM;TS
[1]     TS←⎕TS
[2]     TIMESTAMP←(TWO TS[1]),'-',(TWO TS[2]),'-',(TWO TS[3]),' ',(TWO TS
      [4]),':',(TWO TS[5])
    ∇


    ∇  RESULT←TWO NUMBER
[1]    ⍝
[2]    ⍝ TO REPRESENT ANY NUMBER WITH TWO INTEGER PLACES.
[3]    ⍝ FRACTIONAL PART IS ROUNDED, BIGGER THAN 99 GETS TRUNCATED,
[4]    ⍝ AND SMALLER THAN 10 GETS LEADING ZEROES.
[5]    ⍝
[6]     ALWAYS '(NUMBER≥0)∧(0=ρρNUMBER)' _
[7]     RESULT←¯2↑'00',⍕⌊0.5+NUMBER
    ∇
```

Section 5 Description of update MOSES

```
    ∇  NEGΔIMSTO
[1]   ⍝ ALLOWS INPUT-GOODS INVENTORIES TO HAVE LEVELS BELOW ZERO
[2]    ENS 1=1↑⍴'PLANQREVISE' MODFNP 'SHORTAGE←0⌈⍵SHORTAGE←0×'
[3]    VARIANTS←VARIANTS,' NEG-IMSTO'
    ∇


    ∇  POSITIVEΔNETΔWORTH
[1]   ⍝ FROM 81-02-02 A COMBINATION WITH 'NULLIFYΔNEGΔNW'
[2]   ⍝
[3]   ⍝ TO MAKE SURE BORROWING DO NOT EXCEED ASSETS,
[4]   ⍝ AND TO , THOUGH IN QUITE A CRUDE WAY , ADJUST
[5]   ⍝ NEW-BORROWING TO THE DEBT/EQUITY-RATIO
[6]   ⍝ STEP 1: QΔBW ≤ REDCHBW(=.15)×BW
[7]   ⍝       2: QΔBW REDUCED IF 0.1<(NW/A)≤0.3
[8]   ⍝       3: QΔBW =0       IF 0.1≥(NW/A)
[9]   ⍝       4: FIRMS ARE NULLIFIED THE 6'TH QUARTER WITH   NW<0
[10]  ⍝
[11]   'INVFIN' MODADD 'QDESCHBW←(0⍵QDESCHBW[THO]←REDCHBW×BW[THO←(QDESCH
      BW>REDCHBW×BW)/⍳⍴BW]'
[12]   'INVFIN' MODADD 'QDESCHK2←(RW⍵QDESCHBW←(BWΔCHECK ((BW+QDESCHBW)÷(
      K1+(K2+QDESCHK2)+K3)))×QDESCHBW'
[13]  ⍝
[14]   'INVFINΔADJUSTMENTS' MODADD '⍝ NW IS⍵BAD←BAD+(NW<0)'
[15]   'INVFINΔADJUSTMENTS' MODADD 'BAD⍵REALLYΔBAD←BAD=6'
[16]   'INVFINΔADJUSTMENTS' MODADD 'REALLY⍵⍳(0<(+/REALLYΔBAD))/'' NULLIF
      YΔNW REALLYΔBAD '''
[17]  ⍝
[18]   'NULLIFY' MODADD 'SHRINK ''AMAN⍵SHRINK ''BAD'''
[19]   'NULLIFY' MODADD 'SHRINK ''QW''⍵SHRINK ''REALLYΔBAD'''
[20]  ⍝
[21]   VARIANTS←VARIANTS,'  POS-NET-WORTH-ELSE-NULLIFY '
[22]  ⍝
    ∇
```

Section 5 Description of update MOSES

```
    ∇  INDΔPURCHΔSHARES
[1]    ⍝ PURCHASING-SHARE INDIVIDUALIZED IN THE FOLLOWING WAY :
[2]    ⍝ I/O-MATRIX ENDOGENOUS IN VOLUME TERMS
[3]    ⍝ PURCHASING SHARE: (SUM I=1...10 IO[X I])÷(SUM I=1...13 IO[X I])
[4]    ⍝ PURCH.-SHARES ARE INDIVIDUAL FOR MKT X=1...4
[5]    ⍝ THE RELATION (IO[X I])÷(IO[X J]); I,J∈[1,10],I≠J; IS
[6]    ⍝ FIXED THOUGH.......
[7]    ⍝ NOTE: IF FN. *ADDFIRM* SHOULD BE USED, CHANGE LINES
[8]    ⍝ 'Q←' AND 'QQ←'
[9]    ⍝
[10]   ⍝ FROM FRED OCT-80
[11]   ⍝
[12]    'TARGΔSEARCH' MODSUBST 'QEXPPNET←⍵QEXPPNET←QEXPP-SHARE×(QEXPPIM+.
      ×IO)[MARKET]'
[13]    'MAXIMSTO' MODSUBST 'R←(⍴IO⍵R←((⍴IO)[MARKET;] MULT7 SHARE)MULT7 R
      EFQIMSTO × IMBIG'
[14]    'MINIMSTO' MODSUBST 'R←(⍴IO)⍵R←((⍴IO)[MARKET;] MULT7 SHARE)MULT7
      REFQIMSTO × IMSMALL'
[15]    'OPTIMSTO' MODSUBST 'R←(⍴IO)⍵R←((⍴IO)[MARKET;] MULT7 SHARE)MULT7
      REFQIMSTO × IMSMALL+IMBETA×IMBIG-IMSMALL'
[16]    'PLANQREVISE' MODSUBST 'MAT←(⍴IO)⍵MAT←(⍴IO)[MARKET;] MULT7 SHARE'
[17]    'PLANQREVISE' MODSUBST 'IMSTO←⍵IMSTO←MAXIMSTO⌊IMSTO+QIMQ-MAT MULT
      7 QQ'
[18]    'PLANQREVISE' MODSUBST 'QIMQ←0⌈((⍴IO⍵QIMQ←0⌈(MAT MULT7 QPLANQSAVE
      )+(OPTIMSTO-IMSTO)÷4×TMIMSTO'
[19]    'FINALQPQSQM' MODSUBST 'QVA←⍵QVA←QVA×1+QDVA←¯1+(QQ×QP-SHARE×((QPD
      OM×1-TXVA2)+.×IO)[MARKET])÷QVA'
[20]   ⍝ WE ALSO SHRINK VAR. *SHARE* IN NULLIFY
[21]   ⍝
[22]    VARIANTS←VARIANTS,'  INDIVIDUAL-PURCHASING-SHARES '
    ∇
```

Section 6 In case of trouble...

Here are some tricky situations in connection with the running of the model:

1) Error during simulation. Not the model itself.

Make the system command

) SI.

You can then see in what function the simulation was interrupted.

If you are interrupted in a "print-out"-function (not the model itself) you can usually continue the simulation by doing:*

→⍞LC+1

This means that you skip the line where you were interrupted. This can damage the printing out of results in some tables, but not the simulation itself.

2) Error during simulation. The model itself.

If you are interrupted in a function belonging to the model itself, this is usually due to one firm (out of 147) behaving in a perverse way (getting production volume less or equal to zero or something like that).

You must then either change some lines in the model or some lines in the initialization. Due to the character of the problem it may take anything

---

* In front of LC should be an APL-symbol which is an empty square. For typographical reasons this is written as ⍞.

from a couple of hours to weeks to correct this error, since one has to get well acquainted with the model-code itself to understand why things have gone wrong. (A quick solution may be to nullify (=delete) this firm in the beginning of the simulation, but this is not the best solution.)

3) Error during initialization

Usually due to some technicalities when using the functions MODADD, MODSUBST, etc... See section 3.

4) Editing an APL-function

While making changes in an APL-function one might sometimes wind up in a situation where it is impossible to get a new line-number, when pressing the RETURN-key. This comes about if one is writing an expression with ' in, and for some reason just has given one apostrophe. One must write both a left-hand apostrophe and a right-hand apostrophe to get a new line-number.

Section 7 Printing out the result

The result is, as mentioned previously, stored in a workspace called SXXVYY, where XX is the number of the experiment version, and YY is the number of the initialization version.

A function **PRINT** should be used to print out the result.

The result is a number of tables with different names.

To print out a table perform:

PRINT 'name of table'.

The names of the tables, available after a run, are shown if one performs the function call:

ALLREPORTS

Example:

We wish to print out the table called
YEARLY△INDUSTRY△TOTAL.
Perform:

PRINT 'YEARLY△INDUSTRY△TOTAL'

You then get yearly performance of some main economic indicators.

This result is printed out on the screen and on a printer connected in series with the terminal, after each simulated year. Deleted firms (bankruptcy) are also printed out during simulation.

LITERATURE

"The APL Reference Manual": For example:
APLSF programmer's reference manual, DEC system 10, DEC-10-LPLSA-A-D, 1976.

Batch Reference Manual, DEC system 20, DEC-20-OBRMA-A-D, 1978.

Albrecht-Lindberg, The Micro Initialization of MOSES, IUI, 1982, Working Paper No. 72.

Eliasson, G, A Micro-to-Macro Model of the Swedish Economy, IUI Conference reports, Stockholm 1978.

Eliasson, G, Experiments with Fiscal Policy Parameters on a Micro-to-Macro Model of the Swedish Economy, Reprint from Robert H. Haveman and Kevin Hollenbeck, eds., Microeconomic Simulation Models for Public Policy Analysis, vol. 2, pp.49-95, 1980.

Eliasson, G, (forthcoming), The firm and financial markets in the Swedish Micro-to-Macro Model (MOSES), IUI, 1983.

WORKING PAPERS (Missing numbers indicate publication elsewhere)

1976

1. Corporate and Personal Taxation and the Growing Firm
by Ulf Jakobsson

7. A Micro Macro Interactive Simulation Model of the Swedish Economy.
Preliminary model specification
by Gunnar Eliasson in collaboration with Gösta Olavi

8. Estimation and Analysis with a WDI Production Function
by Göran Eriksson, Ulf Jakobsson and Leif Jansson

1977

11. A Comparative Study of Complete Systems of Demand Functions
by N Anders Klevmarken

12. The Linear Expenditure System and Demand for Housing under Rent Control
by Per Högberg and N Anders Klevmarken

14. Rates of Depreciation of Human Capital Due to Nonuse
by Siv Gustafsson

15. Pay Differentials between Government and Private Sector Employees in Sweden
by Siv Gustafsson

1979

20. A Putty-Clay Model of Demand Uncertainty and Investment
by James W. Albrecht and Albert G. Hart

1980

25. On Unexplained Price Differences
by Bo Axell

26. The West European Steel Industry - Structure and Competitiveness in Historical Perspective
by Bo Carlsson

27. Crises, Inflation and Relative Prices in Sweden 1913-1977
by Märtha Josefsson and Johan Örtengren

33. The Demand for Energy in Swedish Manufacturing
by Joyce M. Dargay

34. Imperfect Information Equilibrium, Existence, Configuration and Stability
by Bo Axell

1981

35. Value Added Tax: Experience in Sweden
by Göran Normann

36. Energi, stabilitet och tillväxt i svensk ekonomi (Energy, Stability and Growth in the Swedish Economy)
by Bengt-Christer Ysander

37. Picking Winners or Bailing out Losers? A study of the Swedish state holding company and its role in the new Swedish industrial policy
by Gunnar Eliasson and Bengt-Christer Ysander

38. Utility in Local Government Budgeting
by Bengt-Christer Ysander

40. Wage Earners Funds and Rational Expectations
by Bo Axell

41. A Vintage Model for the Swedish Iron and Steel Industry
by Leif Jansson

42. The Structure of the Isac Model
by Leif Jansson, Tomas Nordström and Bengt-Christer Ysander

43. An Econometric Model of Local Government and Budgeting
by Bengt-Christer Ysander

44. Local Authorities, Economic Stability and the Efficiency of Fiscal Policy
by Tomas Nordström and Bengt-Christer Ysander

45. Growth, Exit and Entry of Firms
by Göran Eriksson

47. Oil Prices and Economic Stability. The Macroeconomic Impact of Oil Price Shocks on the Swedish Economy
by Bengt-Christer Ysander

48. An Examination of the Impact of Changes in the Prices of Fuels and Primary Metals on Nordic Countries Using a World Econometric Model
by K. S. Sarma

50. Flexibility in Budget Policy. Changing Problems and Requirements of Public Budgeting
by A. Robinson and B.-C. Ysander

51. On Price Elasticities in Foreign Trade
by Eva Christina Horwitz

52. Swedish Export Performance 1963-1979. A Constant Market Shares Analysis
by Eva Christina Horwitz

53. Overshooting and Asymmetries in the Transmission of Foreign Price Shocks to the Swedish Economy
by Hans Genberg

54. Public Budgets in Sweden. A Brief Account of Budget Structure and Budgeting Procedure
by Bengt-Christer Ysander

55. Arbetsmarknad och strukturomvandling i de nordiska länderna
av Bertil Holmlund

56. Central Control of the Local Government Sector in Sweden
by Richard Murray

58. Industrial Subsidies in Sweden: Macro-economic Effects and an International Comparison
by Bo Carlsson

59. Longitudinal Lessons from the Panel Study of Income Dynamics
by Greg J. Duncan and James N. Morgan

1982

60. Stabilization and Growth Policy with Uncertain Oil Prices: Some Rules of Thumb
by Mark Sharefkin

61. Var står den nationalekonomiska centralteorin idag?
av Bo Axell

63. General Search Market Equilibrium
by James W. Albrecht and Bo Axell

64. The Structure and Working of the Isac Model
by Leif Jansson, Thomas Nordström and Bengt-Christer Ysander

65. Comparative Advantage and Development Policy Twenty Years Later
by Anne O. Krueger

67. Computable Multi-Country Models of Production and Trade
by James M. Henderson

68. Payroll Taxes and Wage Inflation: The Swedish Experiences
by Bertil Holmlund (Revised, September 1982).

69. Relative Competitiveness of Foreign Subsidiary Operations of a Multinational Company 1962-77
by Anders Grufman

70. Optimization under nonlinear constraints
by Leif Jansson and Erik Mellander

71. Technology, Pricing and Investment in Telecommunications
by Tomas Pousette

72. The Micro Initialization of MOSES
by James W Albrecht and Thomas Lindberg

73. Measuring the Duration of Unemployment: A Note
by Anders Björklund

74. On the Optimal Rate of Structural Adjustment
by Gunnar Eliasson

75. The MOSES Manual
by Fredrik Bergholm