The Fair-Parke Program for the Estimation and Analysis of Nonlinear Econometric Models

User's Guide

**Programmed by Ray C. Fair
and William R. Parke**

**User's Guide written by
Ray C. Fair**

November 11, 2013

Alphabetic Listing of all Commands

Page	Command	Description
	`@`	A @ in column 1 indicates that the line is a comment.
53	`2SLS`	2SLS.
53	`2SLAD`	2SLAD.
79	`3SLS`	3SLS.
48	`ADDFACT`	Read in add factors for the stochastic equations.
111	`AUTOREG`	Estimate autoregressive equations.
19	`BEGINNOPRINT`	Begin no printing.
29	`CAPITAL`	Create a capital stock series.
36	`CHANGEVAR`	Change exogenous variable values.
102	`CHECK`	Debug the model.
20	`CLOSE`	Close a unit.
63	`COEF`	Set coefficient values.
25	`CREATE`	Create a variable (which may not be part of the model).
100	`CREATEU`	Create estimated residuals.
50	`CTC`	Specify a coefficient constraint.
88	`DEVMEAN`	Take deviations from means.
166	`DISTAP`	Bootstrap distribution for AP statistic.
165	`DISTCOEF`	Bootstrap distribution for coefficients.
166	`DISTMULT`	Bootstrap distribution for multipliers.
166	`DISTPRED`	Bootstrap distribution for predicted values.
19	`DO`	Begin `DO` loop.
111	`DRAW`	Draw a normal deviate.
120	`DVALUES`	Compute and analyze d values from stoch. sim. results.
42	`EQ`	Specify a stochastic equation.
19	`ENDDO`	End `DO` loop.
19	`ENDNOPRINT`	End no printing.
48	`ENDOGENOUS`	Undo the `EXOGENOUS` command for a variable.
69	`ESTS`	Estimate the covariance matrix of residuals.
17	`EXIT`	Exit the program (same as `QUIT`).
47	`EXOGENOUS`	Take an endogenous variable to be exogenous.

17	`QUIT`	Exit the program (same as `EXIT`).
65	`READCOEF`	Read the coefficients.
67	`READCOV`	Read the coefficients' covariance matrix.
31	`READDATA`	Fast way of reading data into the model.
62	`READFSR`	Read the list of first stage regressors.
51	`READCONSTR`	Read information on the coefficient constraints.
71	`READS`	Read the covariance matrix of residuals.
23	`READNAMES`	Read variable names.
16	`RETURN`	Change the command unit to the previous command unit.
115	`RMSE`	Calculate RMSEs.
116	`RMSEA`	Calculate RMSEs by period ahead.
27	`SEAS`	Seasonally adjust a series.
110	`SEED`	Set the seed for stochastic simulation.
20	`SETLA`	Set a specified element of LA.
70	`SETS`	Set a specified element of the covariance matrix S.
20	`SETSPA`	Set a specified element of SPA.
37	`SETUPCHANGEVAR`	Set up options for the `CHANGEVAR` command.
103	`SETUPCHECK`	Set up options for the `CHECK` command.
60	`SETUPEST`	Set up options for limited information estimation.
87	`SETUPFIMLCOV`	Set up options for FIML estimation.
35	`SETUPPLOTS`	Set width for plot output.
144	`SETUPFIMLRE`	Set up options for FIML estimation of RE models.
93	`SETUPDFP`	Set up options for the DFP algorithm.
137	`SETUPRE`	Set up options for a rational expectations (RE) model.
97	`SETUPSOLVE`	Set up options for solution.
77	`SETUPTEST`	Set up options for chi-square tests.
78	`SETUPTESTEND`	Set up options for the `TESTEND1` and `TESTEND2` commands.
38	`SETYTOYY`	Set `Y` to `YY`.
39	`SETYYTOY`	Set `YY` to `Y`.
18	`SMPL`	Set the sample period.
95	`SOLVE`	Deterministic simulations.
15	`SPACE`	Initial space allocations.

48	`STATUSEQ`	Gives status of equations as to whether they are dropped.
128	`STOMULT`	Compute multipliers (stochastic).
104	`STOSIM`	Stochastic simulation.
118	`TABLE8-1`	Compute Table 8-1 in Fair (1984).
119	`TABLE8-3`	Compute Table 8-3 in Fair (1984).
76	`TESTH`	chi-square tests for Hansen's estimator
75	`TEST1`	chi-square tests for OLS.
75	`TEST2`	chi-square tests for 2SLS.
77	`TESTBEG1`	Andrews beginning of sample stability test for OLS.
77	`TESTBEG2`	Andrews beginning of sample stability test for 2SLS.
76	`TESTEND1`	Andrews end of sample stability test for OLS.
76	`TESTEND2`	Andrews end of sample stability test for 2SLS.
76	`TESTSTAB1`	Andrews-Ploberger stability test for OLS.
76	`TESTSTAB2`	Andrews-Ploberger stability test for 2SLS.
20	`TITLE`	Create title of up to 72 characters.
64	`WRITECOEF`	Write the coefficients.
67	`WRITECOV`	Write the coefficients' covariance matrix.
30	`WRITEDATA`	Write variable values for future reading by `READDATA`.
71	`WRITES`	Write the covariance matrix of the residuals.
125	`XMULT`	Set up exogenous variable changes for `MULT` and `STOMULT`.
101	`ZEROU`	Set the residuals to zero.

Contents

PREFACE

References

Most of the discussion of the techniques in this guide is in *Macroeconometric Modeling* (MM), which is on the site *fairmodel.econ.yale.edu*. Click the second picture and then "Macroeconometric Modeling." I have gathered most of my research in macroeconometrics in this document. In some cases there is discussion in Fair (1984) that is not in MM, and I have indicated this when relevant. A few other articles are mentioned in this guide, but most of the article references are in MM and Fair (1984).

A Note on Reading This Guide

Many of the options in this program can be ignored for beginning users. The best way to learn the program is 1) read the "Key Things to Know" material in this Preface, 2) read Chapter 1, and 3) begin working through the example in Appendix A. As you go through this example, you can read about the commands as they are used. (Use the alphabetic listing of the commands at the beginning of this guide for quick reference purposes.) You need not go through all the example, just the commands that you will be using.

The word "print" in this Guide is used to refer to writing ASCII text to either 1) the screen, 2) a piped output file, 3) a printer, or 4) a disk file. The word "write" is used to refer to writing binary text to a disk file.

No one, not even advanced users, need read Appendix B. It contains programming notes, notes that are not relevant unless one is planning to modify the code.

The installation instructions are in Appendix C.

Editing, Piping, and Listing

If you are going to use the FP program in batch mode, which is the main way the program is used, you should first put all of the commands in an input file. For the first example in Appendix A the input file is called `IS.INP`. These types of files can be constructed and edited using a text editor. The first step is thus to create an input file using a text editor.

The second step is to run the job and save the output to a file. (Most FP jobs generate a considerable amount of output, and printing the output to a printer as it is being generated is not recommended.) The way to do this is to pipe the output to a file, say, `OUT`. To do this, type at the prompt (assuming the input file is called `IS.INP`):

`FP > OUT` (hit the enter key)
`INPUT FILE=IS.INP;` (hit the enter key—don't forget the semicolon)

This will execute all the commands in `IS.INP` until the `QUIT;` command is reached, and then it will stop and return to the prompt. WARNING: After you type `FP > OUT`, you will not be prompted to type `INPUT FILE=IS.INP;`. You should just begin typing this line as soon as what you are typing is being listed on the screen. (It takes a few seconds to load FP, and so you *cannot* begin typing `INPUT FILE=IS.INP;` immediately after typing `FP > OUT` and hitting the enter key.)

The third step is to examine the output, which is in `OUT` in the current example. This can be done using a text editor. Just edit `OUT` and use the search option to find that part of the output that you want to examine or save to a smaller file.

To summarize, the recommended way to use the FP program in batch mode is to 1) create an input file using a text editor, 2) run the job and pipe the output to `OUT`, and 3) edit `OUT`. (You can obviously use different names than `OUT`.) While this procedure may seem awkward at first, especially if you are used to the WINDOWS environment, it is very fast and efficient for jobs as big as most FP jobs. One advantage of it is that you can save output from different jobs in different files (`OUT1`, `OUT2`, etc.) for later reference. You will (we think) become a real fan of the procedure after a while.

If you want to work in interactive mode rather than batch mode, you should put the command `KEYBOARD;` at the point in the input file where you want to begin working interactively. (Again, this can be done using a text editor.) Then at the prompt, type just `FP` (no piping) and hit the enter key. You will then be prompted for the input file, and at this prompt type `INPUT FILE=IS.INP;` and hit the enter key (assuming the input file is called `IS.INP`). Output will then flash by on the screen until the command `KEYBOARD;` is reached, at which point you can begin entering commands at the keyboard and working with the program interactively. This is sometimes useful to do, but only if a small amount of output is to be generated per command. Otherwise, the output flashes by on the screen too fast to read.

Finally, another way of doing batch work is to create a file, say `FPRUN.BAT`, which consists of just one line: `FP > OUT < IN`. Then create the file `IN`, which also consists of just one line: `INPUT FILE=IS.INP;` (assuming the name of the input file is `IS.INP`). Then at the prompt type `FPRUN` and hit the enter key. You will then not have to type anything else. This job will execute the commands in `IS.INP` and pipe the output to `OUT`.

Key Things to Know About the Format of Commands

Most commands have options associated with them. There are defaults for almost all options built into the program, and options are not required if the defaults are to be used. The options follow the command name; they can be in any order. A command plus its options ends with a semicolon.

Some commands have subcommands, names, or values following them. *Subcommands*, like commands, can have options associated with them, and they also end with a semicolon. *Names* are entered one name per line, with no semicolon after a name. The entire listing of the names is ended with a semicolon on a separate line. *Values* are entered one value per line, with no semicolon after a value. When values are listed, there is usually a fixed number of values to be read by the program, and the listing of the values is ended when the fixed number has been read. Otherwise, the listing of values is ended with a 0 on a separate line (a blank line is equivalent to a line with a 0).

The commands and options can be in upper or lower case. Commands longer than one line can be continued on the next line or lines. The program simply keeps reading until it finds a semicolon. Also, two or more commands can be on the same line as long as they are not followed by subcommands, names, or values and as long as they are separated by semicolons.

WARNING: The following expressions will not work: `A+-B`, `A*-B`, `A**-B`, `A- -B`, `A/-B`. The expressions need to be:`A+(-B)`, `A*(-B)`, `A**(-B)`, `A-(-B)`,`A/(-B)`.

Key Things to Know About Debugging a Model

A common problem that users have is that they cannot get their models to solve and have no idea why. The program is fairly good at allowing one to pinpoint the problem. In addition to the `CHECK` command, there are a number of other procedures that can be followed. These are explained in Section 9.3, and you should read this section carefully if you are having solution problems.

Key Things to Know About Missing Observations and Gaps in the Sample Period

The program checks for missing observations before estimating an equation. If observations on at least one variable in the equation are missing, the program automatically skips these observations before estimation even if they are within the sample period specified by the `SMPL` command. A message is given stating the number of observations skipped. Also, the program checks for missing observations before solution. Solution will not be done over missing observations.

The program assigns a value of 999999. to missing observations. If you would like to use some other value to denote missing observations, you can change the value in the main program of the FORTRAN code before compiling. Just change the 999999. number in the main program to the number you want.

It is possible to estimate an equation where the user wants there to be gaps in the sample period even if none of the observations on the variables in the equation are missing. Say that the overall sample period is 1954.1–2013.3, but that you want to omit 1970.1–1972.4 and 1985.3–1986.2. You first create a variable that has missing values for the two sub-periods. This is done as follows:

```
SMPL 1954.1 1969.4;
CREATE MISSV1=0;
SMPL 1973.1 1985.2;
CREATE MISSV1=0;
SMPL 1986.3 2013.3;
CREATE MISSV1=0;
```

These three `CREATE`s set `MISSV1` to `0` for the sub-periods that are not to be skipped. The observations of `MISSV1` for the rest of the sample are equal to the missing value number because no values have been created or loaded in for these periods — in this case the periods 1970.1–1972.4 and 1985.3–1986.2.

Having created the `MISSV1` variable, the next step is to use the `MISSING` option for the estimation commands. If, for example, you state `MISSING=MISSV1` as the option for the `2SLS` command, the program will skip the observations that are missing for `MISSV1`. It will skip these observations even though none of the observations on the variables included in the equation may be missing. The commands that have the `MISSING` option are `OLS`, `2SLS`, `LAD`, and `2SLAD`. This option is not available for `3SLS` and `FIML`.

If you have no missing observations in your model and if you are never going to use the `MISSING` option, you can use the `SETUPEST` command and specify `NOMISS`. This will increase the estimation speed. You can also use the `SETUPSOLVE` command and specify `NOMISS`. The `NOMISS` options, however, only increase speed, and if computer time is no problem, there is no need to specify these options.

Key Things to Know About Models that are Nonlinear in Coefficients

Nonlinearity in variables is much more common in econometric models than is nonlinearity in coefficients and much easier to deal with. Although the program handles both kinds of nonlinearity, it has been written to make models that are only nonlinear in variables very easy to deal with. The cost of this is that some extra work on the part of the user is needed for models that are also nonlinear in coefficients.

Sections 4.7, 4.8, and 5.6 explain how to deal with nonlinearity in coefficients, and you should study these sections carefully if your model is nonlinear in coefficients. Also, you should study the material near the end of the example in Appendix A, which estimates and analyzes equations that are nonlinear in coefficients.

Key Things to Know About Models with Rational Expectations

Models with rational expectations are considerably harder to deal with than models without rational expectations. Chapter 13 explains the extra steps that are needed for the estimation and solution commands for models with rational expectations. You should work through this material carefully and study the rational expectations example (the second example) in Appendix A. Section 15.4 may also be helpful; it shows how Sargent's RE model is set up and analyzed. The program allows models with rational expectations to be estimated by Hansen's method, by the method of Hayashi and Sims, and by `FIML`. Stochastic simulation can also be used for these models.

Test Jobs

As noted above, a good way to learn the program is to work through examples. The first example in Appendix A sets up, estimates, and analyzes a version of the IS model, which is introducted in Chapter 1. Many of the commands in the program are used in this example, and if you work through it carefully, you will have a good understanding of the program. Go back and forth between the example and the discussion of the commands in the chapters. It is easy to find where the commands are discussed by using the alphabetic listing of the commands at the beginning of this guide.

The second example in Appendix A sets up, estimates, and analyzes a rational expectations version of the IS model. If you are going to be working with a model with rational expectations, you should study this example carefully. Models with rational expectations require more setting up than do other models.

Finally, you may want to study some of the examples in Chapter 15. These also help in understanding some aspects of the program.

1 CHAPTER 1: Introduction

1.1 Features of the Program

This program allows one to estimate and analyze dynamic, nonlinear, simultaneous equations models. The models can include:

1. nonlinearity in variables,
2. nonlinearity in coefficients,
3. autoregressive and moving average errors of any order,
4. rational expectations.

The estimation techniques include:

1. ordinary least squares (OLS), (White's correction for heteroskedasticity is an option for OLS),
2. two stage least squares (2SLS),
3. three stage least squares (3SLS),
4. full information maximum likelihood (FIML),
5. least absolute deviations (LAD),
6. two stage least absolute deviations (2SLAD),
7. some versions of Hansen's method of moments estimator,
8. the Hayashi-Sims estimator,
9. various stability tests.

The analytic options include:

1. many single equation tests,
2. running forecasts, within sample and outside sample,
3. calculating root mean squared errors and mean absolute errors,
4. calculating multipliers,
5. solving optimal control problems,
6. estimating standard errors of forecasts by means of stochastic simulation,

7. estimating standard errors of multipliers by means of stochastic simulation,
8. estimating the degree of misspecification of a model by means of successive reestimation and stochastic simulation.

Also, general nonlinear functions of coefficients can be maximized using the program, which means that maximum likelihood estimates of the coefficients of any model can be obtained after one has written out the likelihood function.

One of the main advantages of the program is that it allows one to move automatically from estimation to solution. This means that one can move from, say, 2SLS estimation to things like the calculation of root mean squared errors, the calculation of multipliers, the performing of stochastic simulation experiments, and the solution of optimal control problems.

Another advantage of the program is that it can be used to solve problems that have heretofore not been computationally feasible for large nonlinear models. These problems include 3SLS and FIML estimation, LAD and 2SLAD estimation, some stochastic simulation problems, some optimal control problems, and most problems for models with rational expectations.

Because the program has many advanced features, it can seem overwhelming at first glance. If, however, one is primarily interested in the basic tasks of model building—specification, single-equation estimation, and solution—many of the advanced options can be ignored. It is fairly easy to set up, estimate, and solve a model using the program, and the steps that are needed to do this are discussed in Section 1.2. It is likely for most users that more than three fourths of what one wants to do is covered in the steps in Section 1.2.

It will be useful to memorize the following notation:

U	Structural error terms
S	Covariance matrix of the structural error terms
COEF	Structural coefficients
COV	Covariance matrix of the coefficient estimates
FSR	First stage regressors
RE	Rational expectations
LHS	Left hand side
Y	Predicted variable values
YY	Actual variable values

1.2 Specification, Estimation, and Solution of a Model

The easiest way to introduce the program is to use a simple example and describe the steps that are needed to get the example set up and working in the program. The example used is a version of the IS model; it is not meant to be realistic:

$$\log C_t = c_{11} + c_{21}\log C_{t-1} + c_{31}\log Y_t + c_{41}R_t + v_{1t} \tag{1}$$

where

$$v_{1t} = \rho_{11}v_{1t-1} + u_{1t}$$

$$\log I_t = c_{12} + c_{22}\log I_{t-1} + c_{32}\log Y_t + c_{42}R_{t-1} + u_{2t} \tag{2}$$

$$Y_t = C_t + I_t + G_t \tag{I1}$$

where C is consumption, I is investment, Y is income, G is government spending plus net exports, and R is the interest rate. Equations (1) and (2) are stochastic, and equation (I1) is an identity. (As will be seen below, it is best to number identities differently from stochastic equations.) The error term v_{1t} is assumed to be first order serially correlated. ρ_{11}, the serial correlation coefficient, is treated as a structural parameter to be estimated.

There are two stochastic equations and one identity in this model The model is nonlinear in variables, but linear in coefficients aside from the serial correlation coefficient ρ_{11}. The endogenous variables are C, I, and Y, and the exogenous variables are G and R. The steps needed to estimate and solve this model will now be discussed.

Step 1: Load in the Data

There are five variables in the above model aside from the constant term: C, I, Y, G, and R. Say that data are available from 1952.1 (first quarter of 1952) through 2002:1 (first quarter of 2002). The data can be loaded in as follows. First, the user needs to have the data in a file, which could be:

```
SMPL 1952.1 2013.3;
LOAD C;
```
[data on `C` in free format]
```
'END'
LOAD I;
```
[data on `I` in free format]

```
'END'
LOAD Y;
[data on Y in free format]
'END'
LOAD G;
[data on G in free format]
'END'
LOAD R;
[data on R in free format]
'END'
END;
```

Another possibility for this file is:

```
SMPL 1952.1 2013.3;
LOAD C I Y G R ;
[data in free format, five values for 1952.1, then five values for 1952.2, and so
on]
'END'
END;
```

A command in the program—`LOADDATA FILE=fn;`—reads the file named fn. If the above file were named `IS.DAT`, the data could be read in by the command `LOADDATA FILE=IS.DAT;`.

Annual observations are denoted 1952, 1953, etc., and monthly observations are denoted 1952.01, 1952.02, etc. No values are assigned to observations outside of the range of the sample period in effect. The program keeps track of the missing observations and will not allow estimation over missing observations. The command `PRINTVAR STATS var;`, where `var` is the name of a variable, will give various statistics for the variable, including whether any observations are missing for it.

If a variable with a certain name is loaded in and if another variable with the same name is loaded in later, the second variable overwrites the first one for the sample period used for the second variable.

Once the data are loaded in, they can be written out in binary. The command is

```
WRITEDATA FILE=IS.BIN;
```

This command writes the observations on all the variables that have been loaded in or generated to file `IS.BIN` for the sample period currently in effect. The file created by this command has the information on the missing observations in it. The next time the job is run, the command

```
READDATA FILE=IS.BIN;
```

will read the data back in. This is a faster way of reading in the data.

Step 2: Possibly Create Other Variables

In many cases data are read in that need to be transformed before use in a model. If, for example, three categories of government spending were loaded in, say variables $G1$, $G2$, and $G3$, one may want to use the sum of the three categories as the government spending variable in the model. New variables can be created using the `CREATE` command. For the government spending example, one might have

```
CREATE G=G1+G2+G3;
```

One can also use the `CREATE` command to create a vector of 1's to be used for the constant term estimates:

```
CREATE CNST=1;
```

The arithmetic functions that are allowed in the `CREATE` command are: `+, -, *, /, **, LOG, EXP, and ABS.` Also, negative numbers in parentheses are lags, and positive numbers in parentheses are leads. Spacing does not matter for the `CREATE` command except there has to be at least one space between `CREATE` and the variable name.

The `CREATE` command only creates data for the sample period in effect. Observations that are not in the sample period are counted as "missing". If, say, the entire sample period is in effect, but some of the observations on the right hand side variables are missing, the corresponding observations for the left hand side variable are also counted as missing.

Step 3: Specify the Model

Most equations in econometric models are nonlinear in variables but linear in coefficients, and for sake of the present discussion the model will be assumed to be linear in coefficients. The program also handles the general nonlinear case, but the setup is somewhat more complicated.

If the model is nonlinear in variables, which the above model is, it is necessary to generate new variables that are the nonlinear transformations of the original variables. These transformations need only be done for the variables that appear in the stochastic equations. This is done by the use of the GENR command. For the above model the GENR commands are:

```
GENR LOGC=LOG(C);
GENR LOGI=LOG(I);
GENR LOGY=LOG(Y);
```

The name on the left hand side is the name the user is assigning to the variable. The same rules apply for the GENR command as apply for the CREATE command.

Given the above GENR commands, the specification of the equations is:

```
EQ 1 LOGC CNST LOGC(-1) LOGY R RHO=1;
LHS C=EXP(LOGC);
EQ 2 LOGI CNST LOGI(-1) LOGY R(-1);
LHS I=EXP(LOGI);
IDENT Y=C+I+G;
```

The EQ command specifies the stochastic equations. For the EQ command the first variable is the left hand side variable of the equation and the remaining variables are the right hand side variables. The IDENT command specifies the identities. For identities the equations are simply written out in standard notation. The same rules apply for the IDENT command as apply for the CREATE command.

The LHS command requires some explanation. The program assigns variables to stochastic equations and takes the assigned variables as endogenous. Unless told otherwise, the program takes the left hand side variable in an equation to be the variable assigned to that equation. If the left hand side variable is a transformation, such as LOGC above, the user must specify the untransformed variable that is assigned to the equation. This is done by the LHS command. The above LHS command for C says that C is equal to EXP of LOGC, where LOGC is determined by equation 1.

The EQ, LHS, and IDENT commands can be in any order, although it is usually a good idea in order to avoid confusion to have the LHS command follow immediately the relevant EQ command. Note that only stochastic equations are numbered by the program; identities need not be numbered.

The user does not need to specify which variables are endogenous for solution purposes. The GENR, EQ, LHS, and IDENT commands determine which variables are taken to be endogenous. Any variable not determined by these commands is taken to be exogenous. It is all right for variables to be loaded in or generated that are not part of the model. These variables are simply taken to be exogenous, and they are never used if they do not appear on the right hand side of any equations in the model.

Step 4: Estimate the Equations

Assume that the above model is to be estimated by ordinary least squares for the 1954.1–2013.3 sample period. The commands to do this are:

```
SMPL 1954.1 2013.3;
OLS;
EST 1-2;
END;
```

After the model has been estimated, the coefficient estimates can be saved in a file by the command:

```
WRITECOEF FILE=COEF1.BIN;
```

where COEF1.BIN is the name of the file to which the coefficients are to be written. In future jobs, the coefficients can be read in by the command:

```
READCOEF FILE=COEF1.BIN;
```

If one wants to set some coefficients to particular values (rather than estimating them), this is done by the COEF command. For example:

```
COEF;
1 1 4.3
```

```
2 1 .2
0 or blank line
```

sets values for the first and second coefficients in equation 1. (If the values for the other coefficients in an equation are not set, their values are unchanged from what is in memory at the time.)

Step 5: Solve the Model and Examine its Accuracy

Once the model is estimated (or coefficient values assigned using the `COEF` command), it can be solved. There are a number of solution commands. One command is `SOLVE`. Say that one wants to solve the model dynamically for the 2011.4–2013.3 period. The commands are

```
SMPL 2011.4 2013.3;
SOLVE DYNAMIC FILEVAR=IS.VAR;
```

This command prints out the actual and predicted values of the variables that are listed in file `IS.VAR`, one variable per line. If `FILEVAR` is not specified, all the variables in the model are printed, which can be a lot of printing if the model is large.

If one wants to solve the model and calculate root mean squared errors and mean absolute errors, the `SOLVE` command can be replaced by the `RMSE` command:

```
RMSE DYNAMIC FILEVAR=IS.VAR;
```

This command prints out the root mean squared errors and mean absolute errors for the variables that are listed in file `IS.VAR`.

Step 6: Calculate Multipliers

There are a number of ways to calculate multipliers in the program. The easiest way is to use the `SOLVE` command in combination with the `CREATEU` and `CHANGEVAR` commands. Assume that the `SMPL 2011.4 2013.3;` command is in effect. The command

```
CREATEU;
```

creates residuals for the `2011.4-2013.3` period. The residuals are added to the stochastic equations, which means that when the model is solved with no changes to the exogenous variables, a perfect tracking solution is obtained. (You should check this the first few times you do it to see that you have done everything correctly.)

Now consider changing an exogenous variable of interest. Say that the user wants to increase `G` from its historical value by 50 each quarter to see the effect on the other variables in the model. The commands to do this are

```
CHANGEVAR;
G  ADDSAMEABS
50.
;
```

`ADDSAMEABS` means that the number on the next line (`50` in this case) is to be added to each of the values of `G` for the 2000.4–2013.3 period. If the model is now solved (with `CREATEU` still in effect), the difference between the predicted value of an endogenous variable and its actual value is the estimated effect of the change in `G` on the variable. When the `RMSE` command is used:

```
RMSE DYNAMIC FILEVAR=IS.VAR;
```

it prints out, among other things, the errors, and in this case the errors are simply the estimated multiplier effects. One can thus calculate multipliers by using the `CREATEU, CHANGEVAR`, and `RMSE` commands in that order and examining the "errors" printed out by `SOLVE`.

The `CREATEU` command remains in effect until it is undone by the `ZEROU` command:

```
ZEROU;
```

The change in `G` remains in effect until it is undone by another use of `CHANGEVAR`:

```
CHANGEVAR;
G ADDSAMEABS
-50.
;
```

Step 7: Perform Stochastic Simulation

Stochastic simulation can be used to estimate the uncertainty of forecasts. Say that one in interested in the uncertainty from the error terms in the stochastic equations. One first needs an estimate of the covariance matrix of the error terms, denoted S in the program. This is estimated for the `1954.1-2013.3` sample period by:

```
SMPL 1954.1 2013.3;
ESTS;
```

Stochastic simulation can now be performed. Say that the period of interest is 1975.1–1978.4, that one is interested in one through four quarter ahead predictions, and that the number of trials is to be 100. The commands are:

```
SMPL 1975.1 1978.4;
STOSIM NTRIALS=100 LENGTH=4;
```

This command prints out, among other things, the estimated standard errors of the forecasts.

It is also possible to draw coefficients as well as error terms for the stochastic simulation. In addition, historical errors can be drawn rather than errors from a distribution. See the discussion of the `STOSIM` command in Chapter 9 for the various uses of the `STOSIM` command.

Experimenting with Alternative Specifications

As can be seen, one can get rather quickly to options like multiplier calculations and stochastic simulation. One other useful feature of the program should be noted at this point. Much of the work of building a model consists of estimating many versions of the stochastic equations to see what the data seem to support. (Some people call this data mining.) The program allows this to be done very easily. New variables can be generated using the `GENR` command. Say in equation (1) one wanted to replace R_t with $\log R_t$ (where the `GENR` command is: `GENR LOGR = LOG(R);`). Equation (1) can be modified using the `MODEQ` command:

```
MODEQ 1 LOGR - R;
```

This command adds `LOGR` to and subtracts `R` from equation (3). This new equation can then be estimated. (The `MODEQ` command also works within the `OLS` and `2SLS` commands for ease of experimentation.) The new equation is now the one in memory; it has replaced the old one.

To continue the example, if, say, one wanted to try $\log R_{t-1}$ instead of $\log R_t$, the `MODEQ` command is:

```
MODEQ 1 LOGR(-1) - LOGR;
```

For one more example, if one wanted to add R_{t-1} to equation (1), this is done by

```
MODEQ 1 R(-1);
```

In short, it is very easy to add and subtract explanatory variables using the `MODEQ` command. If new variables are needed (other than simply different lags), these can be created using the `GENR` command. Once a specification is decided upon and estimated, the new version of the model can be solved and analyzed. One does not have to leave the program to analyze different versions of a model.

Debugging the Model

Most errors can be caught using the `CHECK` command in the program. One check is to see if the identities that the user has specified are consistent with the data. This is done using

```
SMPL 2013.3 2013.3;
CHECK IDENT;
```

This check calculates the identities (for the sample period in effect) and prints out the actual and predicted values. If some identity is wrong, the actual and predicted values will differ by more than just rounding error. In most cases this check need only be done for one observation.

Another check is to see if the `LHS` statements are correct:

```
SMPL  2013.3 2013.3;
CHECK LHS;
```

This check calculates the `LHS` expressions and prints out the actual and predicated values. Again, if some `LHS` expression is wrong, the actual and predicted values will differ.

Assume that the model has been estimated for the 1954.1–2013.3 period. In the process of estimating the model the program prints out the sum of squared residuals for each equation. The following check can be used to see if the program is handling the equations correctly:

```
SMPL 1954.1 2013.3;
CHECK RESID;
```

This check calculates the residuals for the sample period in effect and prints out the sum of squared residuals for each equation. Each sum should match the sum printed out at the time of estimation.

Finally, the command `CREATEU` calculates and stores residuals. If a model is solved after `CREATEU` has been used (with no changes in the exogenous variables), then a perfect tracking solution should be obtained. This can be checked using:

```
SMPL 2013.3 2013.3;
CREATEU;
CHECK SOLVE;
```

This check solves the model and prints out the actual and predicted values. These values should be the same aside from rounding error.

Even though all the checks using `CHECK` have been passed, it may still be possible that one's model is not solving. Fortunately, there are a number of other procedures that can be followed to try to find out why the model will not solve. These procedures are discussed in Section 8.3.

Two Stage Least Squares

The two stage least squares command is `2SLS`. The list of first stage regressors can be loaded in in one of two ways. The first is using the `EQ` command. For example, the command

```
EQ 1 FSR v1 v2 v3 v4(-1) v4(-2);
```

specifies `v1, v2, v3, v4` lagged once, and `v4` lagged twice to be the first stage regressors for equation 1.

The first stage regressors for each equation can also be listed in a file, which is read in by command `READFSR FILE=fn;`, where `fn` is the name of the file. For example, the file

```
EQUATION 1
v1
v2
v3
v4 -1
v4 -2
END
EQUATION 2
v5
v6
v7 -1
v8 -1
v9
END
;
```

would specify first stage regressors for equations 1 and 2. Note that there are no parentheses around the lags in this file. The first line for each equation is an identifier. It identifies the equation number. The names of the first stage regressors then follow, one per line. The word `END` ends the reading for each equation. The overall list ends with a semicolon.

1.3 Printing Results

The default for all commands is to print results for all variables in the model. For large models this is generally too much printing, and each relevant command has an option to print only a subset of the variables. As noted above for commands `SOLVE`, `RMSE`, and `STOSIM`, the options take the form `FILEVAR=fn`, where `fn` is a file of variable names, one name per line with a semicolon on the last line to end the file. When this option is specified, only the results for the variables listed in the file are printed. If `fn` is specified to be `KEYBOARD` and the program is being used interactively, the user is prompted for the variable names at the keyboard, one name per prompt, with a semicolon used to end the prompting. If `fn` is specified to be `KEYBOARD` and the program is being used in batch mode (which it usually is), the variable names occur in the input file, one name per line, with a semicolon on a separate line used to end the variable list.

1.4 Errors and Error Messages

Most errors will not cause the program to terminate. If an error is made, the program will issue a message and continue processing or wait for the next command. The user can, however, specify a maximum number of errors, and if this number is reached the program will terminate. This is done using the `MAXERR=n` option of the `SETUPSOLVE` command. This option is not relevant if the user is working from the keyboard, but it may be useful if a large job is being run from an input file.

2 CHAPTER 2: Getting Started

This chapter explains what you need to know to "get into" the program. You begin execution by typing `FP` or `RUN FP`, depending on the system, but before doing this you need to create an input file of initial commands.

2.1 Create an Input file

The first task is to use a text editor to create a file of initial commands. The default name for this file is `JOB.INP`. The first line of the file is a title of up to 72 characters.

Line 1:

Title of up to 72 characters

The next line, which is the `SPACE` command, sets up the space allocation for the user's model. There are defaults for each allocation, but for most jobs at least a few defaults will probably need to be overridden. If no defaults are to be overridden, then just say `SPACE;`. The `SPACE` command must come right after the title line. The command is:

Line 2:

SPACE options ;

The options are:

`MAXVAR=n`	Maximum number of variables to be loaded in, created, and generated. `MAXVAR` can be greater than the number of variables ever used. Default = 100.
`MAXS=n`	Maximum number of stochastic equations. `MAXS` is also the dimension of the covariance matrix (S) of the error terms. Default = 25. WARNING: If the S matrix is to be used (for, say, full information estimation or stochastic simulation), then `MAXS` must be the exact number of stochastic equations in the model.

`MAXCOEF=n`	Maximum number of coefficients in any one equation, counting autoregressive coefficients. Default = 20.
`MAXFSR=n`	Maximum number of first stage regressors in any one equation. Default = 40.
`MAXCOV=n`	Maximum dimension of the covariance matrix (COV) of all the coefficient estimates in the model, counting autoregressive coefficients. `MAXCOV` can be greater than the number of coefficients in the model. If `n` is zero, no space is allocated for the covariance matrix. Default = 0. When the program is asked to write out COV, the number of elements written out is (`n`(`n`+1))/2. Some of these elements will be zero if the actual size of COV in the model is smaller than the maximum allowable size. When the program is asked to read COV, the number of elements read is equal to the number written out.
`NOSMATRIX`	No space for the covariance matrix of the error terms (the S matrix) is allocated. If the S matrix is not going to be used, using this option saves space.
`NOYY`	No space for the predicted values is allocated. This option can be used if none of the solution commands are going to be used. It saves space. [LA(14)=1]
`YEAR` or `MONTH`	Use `YEAR` if the observations are yearly; use `MONTH` if the observations are monthly. The default is quarterly.
`FIRSTPER=p`	The first period ever used. Defaults are 1952.1 for quarterly data, 1952 for yearly data, and 1952.01 for monthly data.
`LASTPER=p`	The last period ever used. Defaults are 2010.4 for quarterly data, 2010 for yearly data, and 2010.12 for monthly data.

You can add as many commands following the `SPACE` command as you want. The input file is ended in one of two ways. When the program encounters the command `KEYBOARD` or `RETURN`:

KEYBOARD; or RETURN;

---

it transfers control to the keyboard. The user can then begin using the program interactively. If the program instead encounters `QUIT` or `EXIT`:

QUIT; or EXIT;

---

the job is terminated. If you use `SETLA 313 1;` somewhere before `QUIT;` or `EXIT;`, the program goes to the next command and starts over as a new job.

2.2 Begin Execution

As noted above, you begin the program by typing `FP` or `RUN FP`. The program then prompts you for the `INPUT` command:

INPUT option;

The option is:

`FILE=fn`	The name fn is the name of the input file discussed in the previous section. The default name is `JOB.INP`. Once the `INPUT` command is read, the program begins executing the commands in the input file.

The `INPUT` command can also be used once you are in the program. If, for example, you have created a file (using a text editor) of commands that you want to execute some time within the job, you can invoke these commands by entering `INPUT FILE=fn`, where fn is the name of the file that you created. All the commands in file `fn` will be invoked until `KEYBOARD`, `RETURN`, `QUIT`, or `EXIT` is reached.

The `INPUT` command can go two deep in the following sense. Say that you have created two files (using a text editor), `INPUT1.INP` and `INPUT2.INP`, where `INPUT1.DAT` is

```
command
⋮
INPUT FILE=INPUT2.INP;
```

```
command
⋮
RETURN;
```

and `INPUT2.INP` is

```
command
⋮
RETURN;
```

If in the main input file (default name `JOB.INP`), you have a line, `INPUT FILE=INPUT1.INP`, the program will execute the commands in `INPUT1.INP`, including the commands in `INPUT2.INP`, which `INPUT1.INP` calls, and then return to `JOB.INP`. There can be as many calls as desired to other input files in `INPUT1.INP`. Each will return to `INPUT1.INP`. There can be no calls to input files in `INPUT2.INP`. In other words, the limit is "two deep."

2.3 Sample Periods

The sample period is set using the `SMPL` command:

SMPL p1 p2;

For quarterly observations the `pi's` are numbers like `1952.1`, `1952.2`, etc. For monthly observations the `pi's` are numbers like `1952.01`, `1952.02`, etc. For yearly observations the `pi's` are numbers like `1952`, `1953`, etc. If the data are monthly you must use the `MONTH` option in the `SPACE` command. If the data are yearly, you must use the `YEAR` option in the `SPACE` command. If the data are not time series data, use the `YEAR` option and take `FIRSTPER=1` in the `SPACE` command. In this case the program will number the observations 1, 2, 3, etc.

2.4 The DO Command

If you want to execute the same set of commands many times, which is done, for example, when bootstrapping, this can be done using the `DO` command. The command is:

DO n ;

where `n` is the number of times to execute the commands. The `DO` loop is ended with the `ENDDO` command:

ENDDO ;

All commands between `DO n;` and `ENDDO;` are executed `n` times.

The `DO` command can generate a lot of printed output, and all printing can be stopped using the `BEGINNOPRINT` command:

BEGINNOPRINT ;

[LA(113)=1] No printing is done (i.e., nothing is printed to the piped output file) until the command `ENDNOPRINT` is reached:

ENDNOPRINT ;

When the `ENDNOPRINT` command is reached, printing resumes.

2.5 Miscellaneous Commands

The following are a few miscellaneous commands that are sometimes useful.

The `PVALUE` command:

PVALUE df chisq ;

returns the p-value for a chi-square value of `chisq` with degrees of freedom `df`.

A unit can be closed using the `CLOSE` command, although this command is rarely needed:

CLOSE option;

The option is:

`UNIT=n`	`n` is the unit number to close.

The current space allocation information is available using the `QUERYSPACE` command:

QUERYSPACE;

The title of the job can be changed using the title command:

TITLE ;
Title of up to 72 characters

In the old version of the program, some commands required setting values in the `LA` and `SPA` vectors. This is no longer true, but it is still possible to set values if desired. For the vector LA the command is:

SETLA i1 i2 ;

This command sets LA`(i1)=i2`. Both `i1` and `i2` are integers.

For the vector SPA the command is:

SETSPA i1 v1 ;

This command sets SPA `(i1)=v2`. `i1` is an integer, and `v2` is a real number.

A file can be printed to the piped output file (or to the screen if the program is being used interactively) using the `PRINTFILE` command:

PRINTFILE options ;

The options are:

`FILE=fn`	`fn` is the name of the file to print.
`BLOCKSIZE133`	The default blocksize is 80. If you use the `BLOCKSIZE133` option, the blocksize is taken to be 133.
`FIRSTLINE=n`	The default is to print the entire file. If you use the `FIRSTLINE=n` option, printing begins with line `n`.
`LASTLINE=n`	If you use this option, the last line of the file printed is `n`.

The command `HELP` provides a listing of all the commands in the program:

HELP;

---

3 CHAPTER 3: Reading, Creating, Writing, and Changing Data

3.1 Numbering the Variables

Unless told otherwise, the program numbers the variables in the order in which they are first introduced in the program. In many cases the user may want to number the variables differently. The user can control the numbering of the variables by using the `READNAMES` command:

READNAMES option;

The option is:

`FILE=fn`	`fn` is the name of a file containing variable names, one name per line, with a semicolon on the last line to end the list. If `fn` is specified to be `KEYBOARD`, the names are listed one name per line with a semicolon on a separate line to end. The default name for `fn` is `YNAME.DAT`.

When the `READNAMES` command is used before any variables have been introduced into the program, the program numbers the variables in the order that they appear in the file read by `READNAMES`, starting with `1`. In this way the user controls the ordering of the variables. If the `READNAMES` command is used after variables have been introduced, the variables in the file read by `READNAMES` are numbered in order following the last variable that was previously introduced.

WARNING: The total number of variables used in the program cannot exceed `MAXVARS` as specified in the `SPACE` command in Chapter 2.

3.2 Loading in Data

The program allows data to be read in from different files. A file is read using the `LOADDATA` command:

LOADDATA option;

The option is:

FILE=fn	The name fn is the name of the file to be read. If fn is specified to be KEYBOARD, the data are read in from the input file (or from the keyboard if the program is being used interactively).

If you use `SETLA 448 1;` before LOADDATA, the program does not close the file from which the data are read (file on unit 14). Then if after LOADDATA you use `SETLA 449 1;` and then use LOADDATA again, the same file continues to be read from the point of the last reading.

Three subcommands are allowed in file fn: LOAD, SMPL, and END. (You can recognize that you are within the LOADDATA command in interactive mode because a LOADDATA prompt appears.) Errors cause a return to the main program. If an error occurs, the data that have been read in prior to the error will remain in memory.

The SMPL subcommand is the same as the SMPL command in the main program, namely `SMPL p1 p2;` , where p1 and p2 are the first and last sample observations. If the SMPL subcommand is not used, the SMPL that was in effect in the main program at the time the LOADDATA command was called is used.

The END subcommand ends the LOADDATA command and returns control to the main program.

The LOAD subcommand has the syntax:

```
LOAD [FORMAT|NOFORMAT] v1 v2 …vk ;
```

`v1, v2 ,…, vk` are variable names. If a name is not already in the model, it is added to the list. No lag option is allowed for these variables.

The data are read by observation. The values for all variables listed after LOAD are read for the first observation first. Then the second observation is read for all variables, and so on. The SMPL in effect is used to determine which observations to read. Observations outside the SMPL range are not modified.

The standard input format is FORTRAN free format. Numbers are delimited by either blanks or commas. If nothing appears between two commas, the corresponding observation for that variable is left at its present value. This feature is useful in the event that one wishes to update certain observations and the use of the `SMPL` subcommand is not convenient. [In interactive mode, hitting the return or enter key has no effect (aside from going to a new line on the screen) until sufficient data have been typed.]

The option `FORMAT` for the `LOAD` subcommand is optional. If it is used, the next line must be a FORTRAN format statement, complete with parentheses (but not the word `FORMAT`). The statement `LOAD  FORMAT` can be used to read a format statement without reading any variables. The format statement remains in effect until a new format statement is read or the `NOFORMAT` option is used. Again, the statement `LOAD NOFORMAT` can be used to cancel the format without reading any variables. A format statement can be up to 80 characters long and must appear on a single line. No data can be on the same line with the format statement. The `FORMAT` option is useful primarily for special circumstances that preclude free form input. For example, the data may come without intervening blanks to delimit numbers. It might also be useful to skip some of the data by blocking it out on the format statement. Unless it is necessary to use a format statement, it is not particularly helpful to do so.

The data after each `LOAD` subcommand must end with `'END'` for either the free format or formatted alternatives. The apostrophes around `END` are required. For free format data, `'END'` can be on the last line of data or on its own line. If the `FORMAT` option is used, the `'END'` statement must appear on the line immediately following the data. The `'END'` statement is used to insure that the right number of values has been read in, which is the number of observations times the number of variables. Note that the file `fn` will end with both an `'END'` for the last set of data points and the subcommand `END;` (don't forget the semicolon) to terminate the `LOADDATA` command.

See the discussion under Step 1 in Chapter 1 for an example of the use of the `LOAD` subcommand.

3.3 Creating New Variables

New variables can be created using the `CREATE` command. The command is

---

CREATE variable = arithmetic expression ;

`variable` is the name of the new variable. The arithmetic functions that are allowed are $+$, $-$, $*$, $/$, $**$, `LOG`, `EXP`, and `ABS`. Negative numbers in parentheses are lags, and positive numbers in parentheses are leads. As many parentheses for the expression can be used as desired as long as left and right parentheses are paired.

The `CREATE` command only creates data for the sample period in effect. The remaining observations are counted as missing. Also, if an observation for at least one right hand side variable is missing within the sample period in effect, the corresponding observation for the left hand side variable is treated as missing. If, for example, the entire sample period is in effect and one of the right hand side variables is lagged once, the first observation for the left hand side variable is treated as missing (since the first observation for the right hand side variable lagged once does not exist). Also, if the entire sample period is in effect and one of the right hand side variables is led once, the last observation for the left hand side variable is treated as missing.

3.4 Extrapolating Variables

If data are loaded in through, say, period T and a forecast is to be made for periods $T + 1$ and beyond, it is sometimes useful to create initial values of the variables for periods $T + 1$ and beyond before solving the model for this period. This can be done using the `EXTRAPOLATE` command:

EXTRAPOLATE option;

The option is:

`VARIABLE=name`name is the `name` of the variable to extrapolate. If `name` is equal to `ALL`, then all variables are extrapolated. The default for `name` is `ALL`.

The `EXTRAPOLATE` command works as follows. Assume that the `SMPL` in effect begins in period $T+1$. The `EXTRAPOLATE` command creates observations on the specified variables for period $T + 1$ through the last period of the `SMPL`. The value of a variable for each of these periods is taken to be the value of the variable in period T. In other words, the variable is taken to be unchanged from its value in period T. These values remain in effect for the rest of the job unless they are changed by the user using the `CHANGEVAR` command, which is discussed in Section 3.6. The `EXTRAPOLATE` command is a useful way of forecasting many exogenous variables (such as dummy variables), where it is natural to assume that the variable will be unchanged in the future.

3.5 Seasonal Adjustment of Variables

The `SEAS` command allows one to seasonally adjust a variable. The command is:

SEAS options ;

The options are:

`I=name`	`name` is the name of the existing variable to be seasonally adjusted.
`K=name`	`name` is the name of the new seasonally adjusted variable.
`ADDITIVE`	If this option is used, the seasonal adjustment is additive; otherwise it is multiplicative. See below for a discussion of this.
`NOSUM`	Without this option, the program constrains for quarterly data the four seasonally adjusted quarterly variables to sum to the annual value and for monthly data the twelve seasonally adjusted monthly variables to sum to the annual value. If `NOSUM` is used, this constraint is not imposed.

The seasonal adjustment procedure is quite simple. The program first computes a centered moving average of the variable. It then computes the ratio of the variable to the moving average if `ADDITIVE` has not been used and the difference from the moving average if `ADDITIVE` has been used. It then averages the ratios or differences for each month or quarter over all the years in the sample. The averages are the seasonal factors. The seasonally adjusted variable is the original variable divided by the seasonal factors if `ADDITIVE` has not been used and subtracted from the seasonal factors if `ADDITIVE` has been used. When the seasonally adjusted variable is constrained to add to the yearly value, the difference between the unconstrained sum and the yearly value is allocated across the four quarterly values or the twelvemonthly values in proportion to this size of the unconstrained seasonally adjusted values.

This seasonal adjustment procedure is much less involved than, say, the X11 procedure that is widely used. This simplicity has its pluses and minuses. On the plus side, it is less likely to remove more than just the seasonal fluctuations than is X11. On the minus sign, it may not adequately remove the seasonal fluctuations. If you are worried about complicated seasonal fluctuations in your variables, then you should probably use something other than `SEAS`.

The seasonal adjustment is done for the sample period in effect. The `SEAS` command is not part of the model—it is like the `CREATE` command in this respect.

3.6 Peak to Peak Interpolation of Variables

The `INTER` command allows one to create a variable that is a peak to peak interpolation of another variable. The command is:

INTER options ;

The options are:

`I=name`	`name` is the name of the existing variable to be interpolated.
`K=name`	`name` is the name of the new variable of interpolated values.
`FILEPEAKS=fn`	`fn` is a file listing the observations to be used as peaks, one observation per line, with a blank line to end the file. For quarterly data the observations are values like 1952.4, 1956.2, etc., and similarly for yearly and monthly data. If `fn` is specified to be `KEYBOARD`, the observations are read in from the input file (or from the keyboard if the program is being used interactively), again one observation per line with a blank line to end.
`FLATBEG`	If this option is used, the values from the beginning of the sample period to the first peak are taken to be the value at the first peak. Otherwise, the first peak to peak line is extrapolated backwards to get the values.

`FLATEND`	If this option is used, the values from the last peak to the end of the sample period are are taken to be the value at the last peak. Otherwise, the last peak to peak line is extrapolated forwards to get the values.

The interpolation is done for the sample period in effect. You should print out the values of the original and interpolated series after using the `INTER` command to make sure the values are what you expect them to be. The `INTER` command is not part of the model—it is like the `CREATE` command in this respect.

3.7 Creating a Capital Stock Series

The `CAPITAL` command creates a capital stock series using the formula

$$K_t = I_t + (1 - \rho)K_{t-1}$$

where K_t is the capital stock, I_t is gross investment, and ρ is the depreciation rate. The command is

---

CAPITAL options ;

The options are:

`I=name`	`name` is the name of the (existing) investment series.
`K=name`	`name` is the name of the capital stock series (to be created).
`BENCHPER=p`	`p` is the benchmark period (a number like `1952.1` for quarterly data).
`BENCHVAL=v`	`v` is the benchmark value (the value of the capital stock in the benchmark period).
`DEPRATE=v`	`v` is the value of the depreciation rate.

The capital stock series is created for the sample period in effect. The above formula is used forwards and backwards if the benchmark period is not at the beginning of the sample period. You should print out the values of the investment and capital stock variables after using the `CAPITAL` command to make sure the values are what you expect them to be. The `CAPITAL` command is not part of the model—it is like the `CREATE` command in this respect.

3.8 Writing and Reading Data in Binary

At any point in the job you can write all the data on the variables to a binary file. The command is

---

WRITEDATA options ;

The options are:

`FILE=fn`	The name `fn` is the name of the file to be written to. The default name is `DATA.BIN`.
`MCMODEL`	Option for the MC model. Write in a different format. [LA(412)=1]

This command writes the observations on all the variables to file `fn` for the sample period in effect. The variables include the variables loaded in plus the variables created and generated. The data are written out in binary. The names of the variables are written out as well.

The (FORTRAN) format of the file that is created by `WRITEDATA` is

```
     WRITE( ) NY
     WRITE( ) (YNAME(I),I=1,NY)
     DO 100 J=NBEG,NEND
 100 WRITE( ) XID(J),(Y(J,I),I=1,NY)
```

where `YNAME` is `CHARACTER*8` and `XID` and `Y` are `REAL*8`. `NY` is the number of variables, `NBEG` is the first observation of the sample period, and `NEND` is the last observation of the sample period.

The data can be read back in using the `READDATA` command:

READDATA options ;

The options are:

`FILE=fn`	The name `fn` is the name of the file to be read. The default name is `DATA.BIN`.
`BYVARIABLE`	If this option is used, the data are read by variable. The format that is read is

```
     READ( ) NY
     DO 100 I=1,NY
     READ( ) YNAME(I)
 100 READ( ) (Y(J,I),J=NBEG,NEND)
```

The sample period that is in effect for `READDATA` must be equal to or contained within the sample period that was in effect for `WRITEDATA` when the file was created. The `READDATA` command is a fast way of reading in the data once the file has been created. The names of the variables are read in as well as the data.

It is best to use the `WRITEDATA` command once all the variables have been loaded in and created. Then the `READDATA` command can be used to read the data on all the variables in the model at the beginning of the job. It is possible, however, to have some variables be loaded in (using the `LOADDATA` command) or created (using the `CREATE` command) and then the `READDATA` command used. In this case the variables read in by the `READDATA` command are placed after the variables that are currently in the model.

WARNING. If you use `READNAMES` before you use `READDATA`, the variables read by `READDATA` will be put after the variables currently in the model even if the same variable names have been used before. Since this is not sensible, don't use `READDATA` after `READNAMES` if some of the variable names are the same. Also note that whenever the `WRITEDATA` command is used, all the variables in the model are written out, including variables that may not have been read by the `READDATA` command.

3.9 Printing and Plotting Variables

Variable values can be printed using the `PRINTVAR` command:

PRINTVAR options v1 v2 … vk ;

`v1, v2, ... , vk` are variable names for the printing. If no variable names are specified, then all the variables are printed unless the `FILEVAR=fn` option below has been used.

The options are:

`BOOTSTRAP`	This option prints (in ASCII) the variable values to `TEMP73.DAT`. Nothing else is printed. The only other option that is relevant if this option is in effect is `FILEVAR=fn`. `TEMP73.DAT` is read by the `DISTPRED` command. [LA(434)=1]
`FILEOUT=fn`	The default option is to have the printing directed to the piped output file (or to the screen if the program is being used interactively). If you use `FILEOUT=fn`, printing instead will be directed to file `fn`.
`FILEVAR=fn`	The default option is to print all the variables in the model (for the sample period in effect) unless variable names are included in the command. If you use `FILEVAR=fn`, only the variables listed in file `fn` plus any variables that may be included in the command are printed. The format of this file is one variable name per line, with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one name per line, with a semicolon to end. For example, to print `Y1, Y2,` and `Y3`, you would say:

```
Y1
Y2
Y3
;
```

STATS If this option is used, the variable values are not printed, and instead the mean, standard deviation, minimum value, maximum value, and number of non missing observations are printed for each specified variable. These statistics are computed for the sample period in effect.

MISS If this option is used, the program lists for each selected variable the observations for which there are missing values. WARNING: Do not use this option in combination with STATS or with any of the options that follow.

LOADFORMAT If this option is used, the data are written out in a format appropriate for the LOADDATA command. If it is used in conjunction with the FILEOUT=fn option, a data file fn can be created that is a subset of the complete data file and that can be read by LOADDATA. This is an easy way of creating a smaller model from a larger one.

FILEBIN=fn If this option is used, the data are written out in binary to file fn for the sample period in effect. The format for the writing is

```
    DO 100 I=1,NVAR
100 WRITE( ) (Y(J,I),J=NBEG,NEND)
```

where NVAR is the number of variables printed, NBEG is the first observation, and NEND is the last observation. This option and the FILEOUT=fn option cannot be used together.

FILEWRITE=fn

If this option is used, the data are written out in a (binary) format that READDATA can read. When this option is used, PRINTVAR is thus like WRITEDATA except that PRINTVAR allows for only selected variables to be written.

COMPARE If this option is used, the values in the Y matrix are compared to the values in the YY matrix. Normally the values in Y and YY are the same, but if the NORESET option has been used in an earlier SOLVE or RMSE command, the values in Y are the predicted values and the values in YY are the actual values. If the NORESET option has been used, then the COMPARE option here allows a comparison of the actual and predicted values.

LEVELS	If this option is used, the levels of the variables are printed in a particular format.
CHANGES	If this option is used, the changes in the variables are printed in a particular format.
PCHANGES	If this option is used, the percentage changes in the variables are printed in a particular format. The percentage changes are at annual rates.
MCOUT	Special for the MC model. Writes names and data in E format.
DIFONLY	This option is only relevant if the COMPARE option has been used. If both COMPARE and DIFONLY are used, only the differences between the actual and predicted values are printed. If COMPARE but not DIFONLY is used, the actual and predicted values are printed along with the differences.
FIXEDF	If this option is used, the format for the printing is fixed; otherwise it is floating. This option is only relevant if LEVELS, CHANGES, or PCHANGES has been used.
MCMODEL	Option for the MC model. Used with FILEOUT=fn. [Check LA(370),LA(371)]

The PRINTVAR command is the most useful command in the FP program for printing, and you should experiment with different options to see which you like best. You will see that the format of the printing when LEVELS, CHANGES, or PCHANGES is used differs from the format when none of these options is used, and you can choose which you like best.

Variable values can be plotted using the PLOTVAR command:

PLOTVAR options ;

The options are:

FILEOUT=fn	The default option is to have the plotting directed to the piped output file (or to the screen if the program is being used interactively). If you use FILEOUT=fn, plotting instead will be directed to file fn.

`FILEVAR=fn`	The variables to be plotted are listed in file `fn`. If `fn` is specified to be `KEYBOARD` or if the `FILEVAR=fn` option is not used, the variables are read in from the input file (or from the keyboard if the program is being used interactively). Up to 10 variables can be plotted per graph. The format of file `fn` or of the variables listed in the input file is one variable name per line, with `END` to end the group (of up to 10). A group may consist of just one variable. As many groups may be listed as desired, the groups separated by `END`. The overall `PLOTVAR` command is ended with a semicolon. For example, to plot `Y1` separately and then `Y3` and `Y4` together, the lines are:

```
Y1
END
Y3
Y4
END
 ;
```

If the first variable name is `ALL`, then all the variables in the model are plotted, one at a time. If `ALL` is used, there are no other lines (no `END` line and no `;` line).

The following command determines the width for the plots.

SETUPPLOTS options ;

The options are:

`WIDTH133`	The default option is to have the width for the plotting be 80 characters (to fit on the screen). If you specify `WIDTH133`, the width is 133 characters.
`-WIDTH133`	Undo `WIDTH133`.

3.10 Changing Variable Values

Variable values can be changed using the `CHANGEVAR` command:

CHANGEVAR ;
`variable option`
[values, one per line]
`variable option`
[values, one per line]
⋮
`;`

`variable` is the name of the variable to change. The options are listed below, where in the discussion it is assumed that the sample period in effect is period T through period $T + n - 1$. If only a semicolon is used for the option, then each variable value is read. In other words, if you use a semicolon for the option, the program assumes that you are going to enter each variable value, one value per line. Given the sample period in effect, the program knows how many values are to be read in, and it keeps reading until the required number are read.

The options are:

`;`	The n variable values are read.
`CHGSAMEABS`	The variable is assumed to change by the same absolute amount each period, beginning with period T. The one absolute amount is read.
`CHGSAMEPCT`	The variable is assumed to change by the same percentage amount each period, beginning with period T. The one percentage amount is read.
`CHGDIFABS`	The variable is assumed to change by a different absolute amount each period, beginning with period T. The n absolute amounts are read.
`CHGDIFPCT`	The variable is assumed to change by a different percentage amount each period, beginning with period T. The n percentage amounts are read.
`ADDSAMEABS`	Add the same absolute amount to each variable value. The one absolute amount is read.

`ADDSAMEPCT`	Add the same percentage amount to each variable value. The one percentage amount is read.
`ADDDIFABS`	Add a different absolute amount to each variable value. The n absolute amounts are read.
`ADDDIFPCT`	Add a different percentage amount to each variable value. The n percentage amounts are read.
`SAMEVALUE`	The variable is taken to be the same value each period. The one value is read.

You should print out the variable values (using `PRINTVAR`) after they have been changed to make sure that the changes are as desired. Note the difference between `CHGSAMEABS` and `ADDSAMEABS`. For the former the change is from period to period, whereas for the latter the change is from the variable value currently in memory. A similar difference holds for the other `CHG`... versus `ADD`...options.

There are two options for `CHANGEVAR` that are set ahead of time. This is done by use of the `SETUPCHANGEVAR` command. Once an option is set using the `SETUPCHANGEVAR` command, it remains in effect until undone by the same command. The command is:

SETUPCHANGEVAR options ;

The options are:

`ANNUALRATE`	The default option for the `CHANGEVAR` command when using the `CHGSAMEPCT` or `CHGDIFPCT` option is not to convert the entered values to annual rates. If, for example, the value .01 is entered, the variable is changed by 1 percent each period. If the data are quarterly, this means that the percentage change at an annual rate is a little over 4 percent—$100 \cdot ((1.01)^4 - 1)$ to be exact. If the `ANNUALRATE` option is specified, then the number entered (e.g., .01) is taken to be the desired annual rate of change. If the data are quarterly and .01 is entered, the quarterly change is taken to be $100 \cdot ((1.01)^{.25} - 1)$ percent rather than 1 percent. If the data are monthly and .01 is entered, the monthly change is taken to be $100 \cdot ((1.01)^{(1/12)} - 1)$ percent rather than 1 percent. (For annual data there is, of course, no difference.) [LA(367)=1]
`-ANNUALRATE`	Undo `ANNUALRATE`.
`MCMODEL`	Option for the MC model. [LA(366)=1] This needs to be used when changing variables for the annual countries. In this case the program changes only every fourth observation.
`-MCMODEL`	Undo `MCMODEL`.

3.11 Setting Actual and Predicted Values

The program stores the actual variable values in a matrix called YY. The predicted values are stored in a matrix Y. Unless specified otherwise, a command that has created predicted values sets Y back to YY upon completion of the command. Commands do, however, have the option of not doing this. If later on the user wants to set Y to YY, this can be done using the `SETYTOYY` command:

SETYTOYY ;

This command sets Y to YY (predicted to actual) for the sample period in effect.

If one wants to set YY to Y, this can be done using the `SETYYTOY` command:

SETYYTOY ;

This command sets YY to Y (actual to predicted) for the sample period in effect.

4 CHAPTER 4: Specifying, Printing, and Modifying a Model

Chapter 3 discussed how variables are loaded in and/or created. Once you have loaded in or created all the variables that you are going to need for a model, you are ready to specify the equations of the model. This chapter discusses how you specify a model. Only four commands are needed to do this: `GENR`, `IDENT`, `EQ`, and `LHS`.

4.1 The GENR Command

The `GENR` command is of the same format as the `CREATE` command in the last chapter:

GENR variable = arithmetic expression ;

`variable` is the name of the variable generated. The arithmetic functions that are allowed are $+$, $-$, $*$, $/$, $**$, `LOG`, `EXP`, and `ABS`. Negative numbers in parentheses are lags, and positive number in parentheses are leads. As many parentheses for the expression can be used as desired as long as left and right parentheses are paired.

If, say, the log of variable `Y1` is in the model, then it must be specified using the `GENR` command: `GENR LOGY1= LOG(Y1)`. The `GENR` command is primarily used to specify all the transformations of the variables that are used in the model.

When a `GENR` command is specified, the program immediately solves the expression for the sample period in effect. The user can then use the `PRINTVAR` command to see if the values of the left hand side variable are what the user expects.

WARNING: At least one variable name must be memory before the `GENR` command can work. Variable names can be read in using the `READNAMES` command or the `LOADDATA` command.

4.2 The IDENT Command

The `IDENT` command specifies the identities:

IDENT variable = identity expression ;

`variable` is the name of the variable on the left hand side of the identity. The identity expression can use the same arithmetic functions that are allowed for the `GENR` command, namely $+$, $-$, $*$, $/$, $**$, `LOG`, `EXP`, and `ABS`. Negative numbers in parentheses are lags, and positive number in parentheses are leads. As many parentheses for the expression can be used as desired as long as left and right parentheses are paired. If, for example, there is an identity $Y = C + I + G$, the `IDENT` command is `IDENT Y=C+I+G`.

The `IDENT` command should not be used to create new variables. For example, the command `IDENT Y= C+I+G` should not be used to create variable `Y`. If `Y` has not been loaded in, use the `CREATE` command `CREATE Y = C + I + G` to create it first. Contrary to the case for the `GENR` command, the program does not solve the identity at the time it is specified. The identity is used when the overall model is solved. The `IDENT` commands can be checked, however, using the `TEST` command, which is discussed below.

4.3 The EQ and LHS Commands

If a stochastic equation is linear in coefficients, it is specified using the `EQ` command. (See Sections 4.7 and 4.8 for the nonlinear case.) This is the only command of the four in which equation numbers are needed. If full information estimation or stochastic simulation is to be done, the equations should be numbered so that no equation numbers are missing. The total number of equations should be `MAXS`, which is specified on the `SPACE` command. If full information estimation and stochastic simulation are not to be done, then there is no restriction on the numbering of the equations except that the largest equation number cannot exceed `MAXS`. The `EQ` command can also be used to specify the first stage regressors for use by the `2SLS` and `2SLAD` commands.

The command is:

EQ number y1 v1 … vk FSR w1 …wh RHO=n ;

`number` is the equation number. `y1` is the name of the left hand side variable. `v1` through `vk` are the names of the right hand side variables. `w1` through `wh` are the names of the first stage regressors. The `FSR` option need not be used. If it is not used, the first stage regressors for the equation are not changed (if any have been specified). If it is used, the old first stage regressors (if there are any) are deleted before reading the new ones. It is possible to use the `EQ` command to read only first stage regressors. This is done by `EQ number FSR w1...wh`. Note in this case that the name of the left hand side variable is not given. `RHO=n` can go anywhere in the command. `n` is the order of the autoregressive process of the error term.

The variable names `v1 ...vk` and `w1...wh` can have lags or leads following them in parentheses, such as `v1(-1), v2(2), w1(-2)`, etc. Lags are negative numbers and leads are positive numbers. To specify for a model with rational expectations that a contemporaneous right hand side variable is an expectations variable, use 99 in parentheses. For example, `v1(99)` specifies that `v1` enters the equation with no lag or lead and is an expectations variable.

Unless specified otherwise, the equation number indicated on the `EQ` command is used to solve for the left hand side variable. If, for example, the specification is `EQ 1 C CNST Y C(-1) R;`, equation 1 would be used to solve for `C`. If, on the other hand, the left hand side variable is a transformation of another variable, say the log of `C`, and one wants to use the equation to solve for the other variable, another command, the `LHS` command is needed. This command is of the same format as the `GENR` command:

LHS variable = arithmetic expression ;

The `LHS` command "undoes" the transformation of the left hand side variable in the `EQ` command. If, for example, the specification is `EQ 1 LOGC C LOGY LOGC(-1) LOGR;`, where `LOGC` is the log of `C`, and one wanted to use equation 1 to solve for `C`, the `LHS` command would be `LHS C=EXP(LOGC);`. In order to avoid confusion, it is usually a good idea to have the `LHS` command follow immediately the `EQ` command, although the program does not require this.

The `LHS` commands are not executed until the model is solved. They can, however, like the `IDENT` commands, be checked using the `TEST` command described below.

4.4 Solving a Model

The above four commands are sufficient to allow a model to be solved (once it has been estimated) if all the equations are linear in coefficients. Although it is not necessary to know this, it may help to understand how the program solves a model. The solution technique is the Gauss-Seidel method. For each iteration (pass through the model), the program first solves the stochastic equations for the left hand side variables of the equations. It then solves for the variables determined by the `LHS` commands. The variables determined by the `IDENT` commands are solved next. Finally, the variables determined by the `GENR` commands are solved for. This completes one pass through the model. Therefore, one pass is the following:

1.	`EQs`	solved
2.	`LHSs`	solved
3.	`IDENTs`	solved
4.	`GENRs`	solved

One point about the solution should be noted. Say that there is a `GENR` command `GENR LOGC=LOG(C)`, an `EQ` command with `LOGC` as the left hand side variable, and a `LHS` command `LHS C=EXP(LOGC)`:

```
EQ 1 LOGC ...  ;
LHS C=EXP(LOGC) ;
GENR LOGC=LOG(C) ;
```

In this case `EQ` will solve for `LOGC`, and `LHS` will solve for `C`, given `LOGC` from the `EQ` solution. There is then nothing left for the `GENR` command to solve. In the coding of the program, the `GENR` command is actually used to solve for `LOGC`, given `C` from the `LHS` solution. This solution simply gives back the value of `LOGC` computed by the `EQ` command. This coding is thus inefficient, but harmless.

Note also the difference between the `CREATE` and `GENR` commands. The `CREATE` commands are never used in the solution of the model. They are simply used to create new variables and are not part of the model.

4.5 Printing or Listing a Model

Once the model has been specified, it is a good idea to print information about the model to make sure that the program thinks the model is the same as you think it is. There are three relevant commands.

The names of all the variables in the model can be printed using the `PRINTNAMES` command:

PRINTNAMES ;

This command simply prints all the variable names that are currently in the program, including variables that have been loaded in or created that may not be part of the model. The names are also numbered, which is the internal numbering of the variables in the program.

The specification of the stochastic equations can be printed using the `PRINTMODEL` command:

PRINTMODEL option ;

The option is:

`FILEOUT=fn`	The default is to print to the piped output file (or to the screen if the program is being used interactively). If you use the `FILEOUT=fn` option, the printing is done to file `fn`.

This command prints the stochastic equations and their coefficient estimates.

Much of the information about a model can be obtained using the `LIST` command. This command is

---

LIST options ;

The options are:

`VAR`	This option lists all the variables in the model.
`CREATE`	This option lists all the `CREATE` commands.
`GENR`	This option lists all the `GENR` commands.
`NLEQ`	This option lists all the `NLEQ` commands.
`LHS`	This option lists all the `LHS` commands.
`IDENT`	This option lists all the `IDENT` commands.
`CTC`	This option lists all the `CTC` commands.
`JACOB`	This option lists all the `JACOB` commands.
`XMULT`	This option lists all the `XMULT` commands.

Only one option is allowed per use of the `LIST` command. One subtle point about the `VAR` option needs to be made. If you first create a variable, say `LOGC`, using a `GENR` command and then specify an equation with this variable on the left hand side, the `LIST VAR` command will show only the equation information. You can see the original definition of the variable using `LIST GENR`.

4.6 Modifying a Model

Once a model has been specified, there are a variety of ways in which it can be modified aside from starting over with a new set of `GENR`, `IDENT`, `EQ`, and `LHS` commands.

The stochastic equations can be modified using the `MODEQ` command:

MODEQ num v1 ... vj - w1 ... wk FSR x1 ... xm - y1 ... yn RHO=n ;

`num` is the number of the equation to modify. `v1...vj` are names of variables to add as explanatory variables to the equation. `w1...wk` are names of explanatory variables to drop from the equation. `x1...xm` are names of first stage regressors to add to the equation. `y1...yn` are names of first stage regressors to drop from the equation. The `FSR` option does not have to be used. The minus sign separates the variables to be added from those to be subtracted. *A space must appear after the minus sign.* The `MODEQ` command cannot be used to change the left hand side variable. If this is needed, you must use the `EQ` command. The order of the autoregressive process of the error term can be modified using the `RHO=n` option, where `n` is the order of the process. If `n` is 0, then no autoregressive process is used.

Variables that are determined by stochastic equations can be taken to be exogenous using the `EXOGENOUS` command. The command is:

EXOGENOUS options ;

The options are:

`VARIABLE=name`	`name` is the name of the variable to take to be exogenous. The variable must be on the left hand side of a stochastic equation in order to use the `EXOGENOUS` command for it.
`DROP`	Normally, the variable is taken to be exogenous only for the sample period in effect. If `DROP` is used, the variable is taken to be exogenous for all observations. When `DROP` is used, the stochastic equation that is matched to the variable is completely dropped from the model.

Once a variable has been taken to be exogenous, values for it can be entered using the `CHANGEVAR` command.

If you have declared a variable to be exogenous and want to make it endogenous again, this can be done using the `ENDOGENOUS` command. The command is:

ENDOGENOUS option ;

The option is:

`VARIABLE=namename` is the name of the variable to take to be endogenous. It is matched to the same stochastic equation as it was matched to before it was taken to be exogenous.

If you want a status report on which equations are taken to be exogenous for which periods, this can be obtained using the `STATUSEQ` command:

STATUSEQ ;

Equations can be add factored using the `ADDFACT` command:

ADDFACT option ;
```
value for the first observation
⋮
value for the last observation
```

The option is:

`VARIABLE=namename` is the name of the variable to add factor. The stochastic equation that is add factored is the equation that is matched to the variable. The values for the add factors pertain to the sample period in effect.

4.7 Models that are Nonlinear in Coefficients

NLEQ expression ;

If an equation is nonlinear in coefficients, it must be specified using the NLEQ command. Say that equation (1) is:

$$\log C_t = c_{11} + c_{21}\log C_{t-1} + (1/c_{21})\log Y_t + c_{31}R_t + u_{1t} \qquad (1)$$

The first thing to be done is to specify, using the EQ command, that the left hand side variable in equation (1) is $\log C_t$:

```
EQ 1 LOGC;
```

The NLEQ command is then used to specify the equation in the following manner:

```
NLEQ LOGC=COEF(1,1)+COEF(2,1)*LOGC(-1)
+(1/COEF(2,1))*LOGY +COEF(3,1)*R;
```

Note that the NLEQ command does not list the equation number. LOGC is matched to equation (1) using the EQ command. The first argument of COEF is the coefficient number in the equation, and the second argument is the equation number.

The NLEQ command, like the EQ command, needs a LHS command associated with it if the left hand side variable is a nonlinear transformation of an original variable. In the above case the LHS command would be:

```
LHS C=EXP(LOGC);
```

Coefficients from other equations can be used in the NLEQ command, as in for example:

```
NLEQ LOGC= COEF(1,1)+COEF(2,1)*LOGC(-1)+COEF(2,3)*LOGY;
```

In this case, COEF(2,3) is the second coefficient in equation 3. When equation 1 is estimated, COEF(2,3) is not estimated; it is taken as fixed for purposes of estimating equation 1. (It is whatever is in memory at the time.) Only COEF(1,1) and COEF(2,1) are estimated.

Another example of the use of NLEQ is:

```
NLEQ LOGC=COEF(1,1)
   +(LOGC(-1)**COEF(2,1))*(LOGY**COEF(3,1));
```

Equations specified by the NLEQ command can be estimated by ordinary least squares, two stage least squares, three stage least squares, and full information maximum likelihood. The ordinary least squares command is NLOLS, and the two stage least squares command is NL2SLS. These two commands are discussed in Section 5.6. The three stage least squares command and the full information maximum likelihood command are the same as for the EQ command, namely 3SLS and FIML. These two commands are discussed in Chapter 7.

The NL2SLS command needs a list of first stage regressors. The first stage regressors can be read in one of two ways. One is to use the EQ command discussed in this chapter, and the other is to use the READFSR command discussed in the next chapter. Unlike the EQ command, the NLEQ command does not have a FSR option. It doesn't need one because the EQ command can be used to read the first stage regressors. For equation 1 the command is simply `EQ 1 FSR` ....

WARNING: Do not specify an equation using the EQ command that has the same left hand side variable as a NLEQ command. The NLEQ specification does not override the EQ specification. When NLEQ is used, the EQ command should only be used to specify the equation number and the left hand side variable (and possibly the first stage regressors). In the above example it would be a mistake to have the EQ command be `EQ 1 LOGC CNST LOGC(-1) LOGY R`. If this were done, the program would use this specification of equation 1 rather than the NLEQ specification.

4.8 Coefficient Constraints

Many kinds of coefficient nonlinearity can be written as coefficient constraints. For example, in the first NLEQ command in Section 4.7, the coefficient of LOGY is `1/COEF(2,1)`, which is a coefficient constraint. In the second NLEQ command, the coefficient of LOGY is `COEF(2,3)`, which is also a coefficient constraint. On the other hand, the nonlinearity in the third NLEQ command is not of this type.

When the nonlinearity is a coefficient constraint, it can be specified in the program using the EQ, CTC, and READCONSTR commands rather than the NLEQ command. The CTC command is

CTC coefficient = arithmetic expression ;

If, for example, the fourth coefficient in equation 1 was equal to the sum of the second and third coefficients, the `CTC` command would be

```
CTC COEF(4,1)=COEF(2,1)+COEF(3,1) ;
```

WARNING: No spaces are allowed inside the parentheses of `COEF(i,j)`.

Constrained coefficients must come last in the equation. As many coefficients can be constrained as desired, as long as no constrained coefficient comes before an unconstrained one.

If you use the `CTC` command, you must tell the program the number of constraints in each equation. This is done using the `READCONSTR` command:

READCONSTR option ;

The option is:

`FILE=fn`	`fn` is the name of the file containing the information on the constraints. The default name for `fn` is `IZC.DAT`. If `fn` is specified to be `KEYBOARD`, then the information is read in from the input file (or from the keyboard if the program is being used interactively). The format of the file is:

```
equation   number of constraints
⋮
0 or blank line
```

If an equation has no constraints, it need not be listed.

The information on the constraints can be printed using the `PRINTCONSTR` command:

PRINTCONSTR ;

Using `EQ`, `CTC`, and `READCONSTR`, the specification of the first example in Section 4.7 (equation (1)) would be:

```
EQ 1 LOGC CNST LOGC(-1) R LOGY ;
CTC COEF(4,1)=1/COEF(2,1);
READCONSTR FILE=KEYBOARD;
1 1
0 or blank line
```

The specification of the second example, where `COEF(2,3)` appears, would be:

```
EQ 1 LOGC CNST LOGC(-1) LOGY;
CTC COEF(3,1)=COEF(2,3);
READCONSTR FILE=KEYBOARD;
1 1
0 or blank line
```

The `EQ, CTC, READCONSTR` specification is much better than the `NLEQ` specification for `3SLS` and `FIML`. Although `NLEQ` will work with `3SLS` and `FIML`, the estimation time is much slower than when the `EQ, CTC, READCONSTR` specification is used. For single equation estimation, on the other hand, the `EQ, CTC, READCONSTR` specification does not work for the `OLS` and `2SLS` commands. The `OLS` and `2SLS` commands do not handle coefficient restrictions (unless the equation is rearranged by the user to make it linear in the unconstrained coefficients—see pages 214–215 in Fair (1984)), and so for single equation estimation the `NLOLS` and `NL2SLS` commands must be used in combination with the `NLEQ` specification.

5 CHAPTER 5: Limited Information Estimation

This chapter presents the single equation estimation techniques that are available in the program. The techniques are discussed in Section 2.3 in MM, which includes a link to part of Chapter 6 in Fair (1984).

5.1 OLS, 2SLS, LAD, 2SLAD

The `OLS, 2SLS, LAD,` and `2SLAD` commands have very similar structures, and they will be discussed together. The commands are for equations that are linear in coefficients except for the presence of autoregressive coefficients of the error term. Equations that are nonlinear in coefficients are discussed in Section 5.6 below.

The `OLS, 2SLS, LAD`, and `2SLAD` commands have many options, which makes the commands look complicated. In fact, for most purposes the commands are quite easy to use. If, for example, you want to estimate equations 1 through 30 in a model by `2SLS`, all you need to say is:

```
2SLS ;
EST 1-30 ;
END;
```

The estimation commands are:

OLS or 2SLS or LAD or 2SLAD options ;
`subcommand options ;`
⋮
subcommand options ;
END;

There are four subcommands `EST, EQ, MODEQ`, and `SMPL`. `EQ` and `MODEQ` are exactly as in Chapter 4. They can be used to specify or modify an equation within the estimation commands. This is useful when one is experimenting with alternative specifications, since one does not have to exit the estimation command to change an equation. The `SMPL` subcommand is the same as the `SMPL` command in the program. It allows the user to change the sample period within the estimation command. Once a sample period has been specified, it remains in effect until changed, even after exiting from the estimation command.

The `EST` subcommand tells the program which equations to estimate. The format for the `EST` subcommand is:

```
EST equations options ;
```

`equations` is a list of one or more equations. Equation numbers can be delimited by spaces or by commas. A range of equations can be specified as, for example, 4–10 if you want to estimate equations 4 through 10. It is important that the hyphen follow immediately after the beginning equation number because a space would be interpreted as a delimiter. Also, a comma must follow immediately after a number if a comma is used. Otherwise, spaces and carriage returns can be added anywhere. To estimate equations 1, 3, 4, 5, 6, and 9, the subcommand could be `EST 1,3-6,9`.

The options for the `EST` subcommand are discussed below. They follow the same syntax as the options for commands. The equation numbers must come first in the `EST` subcommand, followed by any options. The `EST` options pertain only to the equations listed on the particular `EST` subcommand. They are not "remembered" for the next `EST` subcommand (if there is one).

Note that there are two types of options. One type pertains to the entire use of the estimation command (`OLS`, etc. options), and one type pertains only to the equations being estimated (`EST` options).

The overall options are:

`BIASCORRECTION`	If this option is in effect, the coefficients are adjusted by the bias correction vector after estimation. This option is sometimes used when bootstrapping is being done. [LA(243)=1]
`FILEOUT=fn`	The default is to print to the piped output file (or to the screen if the program is being used interactively). If you use `FILEOUT=fn`, the printing is done to file `fn`, and no printing is done to the piped output file (or to the screen).
`PRINTLESS`	If you use this option, there is less printing (to either the piped outupt file (or to the screen) or to file `fn`). Only a brief summary of each regression is printed.

`NOPRINT`	If you use `NOPRINT`, no printing of the regression results is done anywhere.
`NSETS=n`	This option allows more than one set of coefficients to be estimated with one use of the command. A "set" consists of all the equations listed after the command. Each set corresponds to a different set of sample periods. The first sample period for each equation is the sample period currently in effect for the equation. (Remember that sample periods can differ across equations; they are specified by the SMPL subcommand.) For the second period for each equation the last observation is increased by one; for the third sample period the last observation is increased by one more; and so on. If option `MOVEBEGOBS` is specified, the first observation is also increased by one for each successive period. See Chapter 14 for an example of the use of this option.
`MOVEBEGOBS`	See `NSETS=n` above.
`COV` or `COVBLKDIAG`	The covariance matrix of all the coefficients in the model is denoted `COV`. The default is not to estimate this matrix. If you use the `COV` option, the covariance matrix is estimated using formula (6.20) in Fair (1984). The covariance matrix is estimated after the last equation has been estimated. If you use the `COV` option, you should use the same sample period for all the equations; otherwise the COV matrix will be estimated over a different period than some of the equations are. Also, you cannot use the `MISSING` option below if you are going to use the `COV` option. `COV` does not handle missing observations. If you use the `COVBLKDIAG` option, the covariance matrix is estimated taking the off diagonal blocks to be zero. In this case you can use different sample periods for different equations. Each diagonal block for `COV` is simply the estimated covariance matrix for the particular equation. For `2SLS`, the `COVBLKDIAG` option can be used, but this is not recommended. `COVBLKDIAG` is a much faster way of computing the covariance matrix, but it is not correct (see Fair (1984), Section 6.3.2). At best it provides only a good approximation. NOTE: When `NSETS` is greater than 1 and `COV` or `COVBLKDIAG` is in effect, the covariance matrix is also estimated for each set.

NOTE: If you have already estimated the coefficients and only want to estimate COV, the commands are simply:

```
2SLS COV;
END;
```

For this to work, the coefficient estimates must be in memory. This will work even if some or all of the equations have been specified using the `NLEQ` command and estimated using the `NL2SLS` command. In other words, the equations need not be linear in coefficients in order for the `COV` option to work

NOTE: For the `COVBLKDIAG` option to work, all the equations in the model must be estimated in one use of `OLS` or `2SLS`. You cannot exit `OLS` or `2SLS` and then come back in to finish the estimation of the equations if `COVBLKDIAG` is going to work.

WARNING: To be on the safe side when using the `COV` or `COVBLKDIAG` option, you should set `MAXCOV` in the `SPACE` command to the *exact* number of rows (and columns) in the covariance matrix. In some cases you can get away with having `MAXCOV` be larger than the exact number, but why take a chance?

`S`
or `SDIAG`

The covariance matrix of the error terms in the model is denoted S. The default is not to estimate this matrix. If you use the `S` option, the matrix is estimated. The matrix is estimated after the last equation has been estimated. As with the `COV` option, you should use the same sample period to estimate all the equations when using the `S` option. If, on the other hand, you use the `SDIAG` option, different sample periods for different equations can be used. When the `SDIAG` option is used, the S matrix is taken to be diagonal. Each diagonal element is merely the estimated variance of the particular equation.

NOTE: For the `SDIAG` option to work, all the equations in the model must be estimated in one use of `OLS` or `2SLS`. You cannot exit `OLS` or `2SLS` and then come back in to finish the estimation of the equations if `SDIAG` is going to work.

NOTE: When `NSETS` is greater than 1 and `S` or `SDIAG` is in effect, the covariance matrix is also estimated for each set.

`FILECOEF=fn`	Upon completion of each set of estimates, write the coefficient estimates to file `fn`.
`FILECOV=fn`	Upon completion of each set of estimates, write COV to file `fn`. This option can only be used if the `COV` option has been used.
`FILECOVBLKDIAG=fn`	Upon completion of each set of estimates, write COV to to file `fn` in block diagonal form. This option can only be used if the `COVBLKDIAG` option has been used.
`FILES=fn`	Upon completion of each set of estimates, write S to file `fn`. This option can only be used if the `S` or `SDIAG` option `S` has been used.
`FILEMODEL=fn`	Upon completion of each set of estimates, write the information on the stochastic equations to file `fn`.
`FILEWRITE=fn`	This option is specific to the MC model. It writes information to unit 36 under file name `fn`.

The `EST` options are:

`MISSING=name`	When this option is used, observations are skipped if they are missing for variable name. This option is the way of allowing gaps in the sample period. This option does not work for the following options below: `WALD, WALD1, WALD2, MA=n, NEWEYWEST, SAVE3SLS, HANSEN, HAYSIMS`. Also, this option does not work under the `COV` and `S` options above, although it does work for the `COVBLKDIAG` and `SDIAG` options. Summary statistics like the DW statistic are adjusted for the missing observations.
`PLOT=n`	Plot the last `n` residuals of the sample period. If `n` is 0, all the residuals are plotted. The default is no plots.
`WHITE`	Use the White (1980) correction for heteroskedasticity for estimation of the covariance matrix of the coefficient estimates. This option is only relevant for `OLS` with no serial correlation of the error terms.

`WALD`	Compute the Wald statistic for the test of the hypothesis that all the coefficients in the equation except the first one are zero. This option only works if the `WHITE` option is used. It is only relevant for `OLS`.
`WALD1`	If you want to use the `WALD` option to test the hypothesis that the coefficients in one period are the same as in another period, the options `WALD1` and `WALD2` can be used. This test statistic is presented in equation (3.6) in Andrews and Fair (1988). To perform the test you do the following. First, pick the first sample period, declare `WALD1` as an equation option, and estimate the equation. Second, pick the second sample period, declare `WALD2` as an equation option, and estimate the equation. The summary statistics for this second estimation will include the `WALD` statistic. Once `WALD1` has been used, `WALD2` must be used (with the same equation specification except for the different sample period) before `WALD1` can be used again. This `WALD` test has the advantage that it works under very general assumptions about the properties of the error terms.
`WALD2`	See the discussion of `WALD1`.
`MA=n`	This option has two uses. If either the `HANSEN` or `HAYSIMS` option is used (see below), then `n` is the specified order of the moving average process of the error term. `n` should be one less than the maximum lead in the equation (see Section 2.12.1 in MM for an exposition of this). If `HANSEN` and `HAYSIMS` are not used, then the `MA=n` option means to correct for a `n`th order moving average process of the error term in the estimation of the covariance matrix of the coefficient estimates. It also means to correct for heteroskedasticity unless the `NOHETERO` option is used. If the `NOHETERO` option is used, heteroskedasticity is not corrected for. If the `NEWEYWEST` option is used, the Newey and West (1987) weights are used for the MA correction. This use of the `MA` option is not relevant for `2SLS`.
`NOHETERO`	See `MA=n` above.
`NEWEYWEST`	See `MA=n` above.

`SETLHSPRED`	Set the value of the left hand side variable to the predicted value for all sample period observations.
`SETLHSRESID`	Set the value of the left hand side variable to the residual value for all sample period observations.
`SAVE3SLS`	Save information relevant for `3SLS` estimation. The information is saved in files `TS1DAT, TS2DAT,` and `TS3DAT`. See the discussion of the `3SLS` command in Chapter 7.
`HANSEN`	Estimate the equations using Hansen's method of moments estimator (Hansen (1982)). See the `MA=n` option above.
`FILESAVEM=fn`	This option in conjunction with the `HANSEN` option writes the M matrix to file `fn`.
`FILEUSEM=fn`	When this option is used in conjunction with the `HANSEN` option, the M matrix is read in from file `fn` rather than computed. The `FILESAVEM=fn` and `FILEUSEM=fn` options allow two equations to be estimated using the same M matrix, which is needed for some hypothesis testing. See again Section 2.12.1 in MM.
`HAYSIMS`	Estimate the equations using the Hayashi-Sims estimator (Hayashi and Sims (1983)). For this option the equations must not contain autoregressive structural error terms. See the `MA=n` option above.
`LA(68)`	If LA(68) is set to 1, the starting values for the autoregressive coefficents are taken to be the values in SPA(81)...SPA(90). Otherwise, the starting values are taken to be zero.

It should be noted that the summary statistics presented for the `LAD` and `2SLAD` estimates are not right. In general the asymptotic distributions of the `LAD` and `2SLAD` estimates are not known (see Fair (1984), Sections 6.3.5 and 6.3.6). The summary statistics that are presented are those that would pertain if the estimates were `OLS` in the case of `LAD` and `2SLS` in the case of `2SLAD`.

There are some estimation options that are set ahead of time. This is done by use of the `SETUPEST` command. Once an option is set using the `SETUPEST` command, it remains in effect until undone by the `SETUPEST` command. The command is:

SETUPEST options ;

The options are:

`DIVIDET`	The default option is to adjust for degrees of freedom in computing summary statistics by dividing by $T - k$ rather than T, where T is the number of observations and k is the number of coefficients estimated. If instead you want to divide only by T, use the `DIVIDET` option. With this option in effect, there will be no adjustments for degrees of freedom. [LA(300)=1]
`-DIVIDET`	Undo `DIVIDET`.
`NOMISS`	Use this option if there are no missing observations in the model and the `MISSING=name` option for the `OLS, 2SLS, LAD,` and `2SLAD` commands is never going to be used. The `NOMISS` option decreases the computer time needed for estimation. [LA(24)=1]
`-NOMISS`	Undo `NOMISS`. [LA(24)=0]
`MAXRHO=n`	`n` is the limit of the number of iterations for estimating the serial correlation coefficients. The default is 200. [LA(104)]
`TOLRHO=v`	`v` is the value of the tolerance criterion for the iterations involved in estimating the serial correlation coefficients. The default is .00001. (Use .001 for `2SLAD`.) [SPA(33)]
`MAXLAD=n`	`n` is the limit for the number of iterations for LAD and 2SLAD. The default is 20. [LA(107)]
`TOLLAD=v`	`v` is the value of the tolerance criterion for the LAD and 2SLAD iterations. The default is .002. [SPA(35)]
`PRINTONLY`	Do not estimate the coefficients. Print summary statistics only. See the discussion of the `3SLS` and `FIML` commands in Chapter 7 for the use of this option. [LA(109)]
`-PRINTONLY`	Undo `PRINTONLY`.

`LHSLOG`	The left hand side variable is taken to be the log of the original variable for all equations estimated. [LA(100)]
`-LHSLOG`	Undo `LHSLOG`.
`ALT2SLS`	Use (6.13) and (6.14), p.212, in Fair (1984) to compute the 2SLS estimates. If the `ALT2SLS` option is not used, the following formulas are used in place of (6.13) and (6.14): $\hat{\alpha}_i = [(\hat{X}_i - X_{i-1}\hat{\rho}_i)'(X_i - X_{i-1}\hat{\rho}_i)]^{-1}(\hat{X}_i - X_{i-1}\hat{\rho})'(y_i - y_{i-1}\hat{\rho}_i)$ $\hat{\rho}_i = (\hat{u}'_{i-1}\hat{u}_i)/(\hat{u}'_{i-1}\hat{u}_{i-1})$ $\hat{X}_i = D_i X_i$ The equation for $\hat{\rho}_i$ is the same as (6.14)$'$, p.212, in Fair (1984). It should be noted that (6.15) in Fair (1984), not (6.15)$'$, is always used to compute $\hat{V}_{2ii}$. If X_{i-1} and y_{i-1} are in Z_i, then `ALT2SLS` and `-ALT2SLS` are the same aside from rounding error. `ALT2SLS` is slower, and so this option should not be used unless one has to. If `ALT2SLS` is not used and y_{i-1} and/or one or more variables in X_{i-1} are not in Z_i, the resulting estimates will not be consistent. WARNING: Do not use `ALT2SLS` with `2SLAD`.
`-ALT2SLS`	Undo `ALT2SLS`.
`HANSENM`	When using Hansen's method of moments estimator, estimate the M matrix the general way. The default is to estimate the matrix using equation (A2) in Hayashi and Sims (1983), p. 796.
`-HANSENM`	Undo `HANSENM`.
`Q=v`	`v` is the value of `q` for 2SLAD. See the discussion in Fair (1984), Section 6.3.6, regarding the use of `q`. The default value is .5.

If you are using the `2SLS` or the `2SLAD` command, the program needs to know the first stage regressors for each equation. There are two ways of providing the program with this information. One way is to use the `EQ` command discussed in

Chapter 4 (or the `EQ` subcommand for `2SLS` and `2SLAD`).

The second way to provide the program with the first stage regressor information is to create a file of the variable names and lags using a text editor and then read the file using the `READFSR` command:

READFSR option ;

The option is:

`FILE=fn`	`fn` is the name of the file that contains the `FSR` information. The default name for `fn` is `FSR.DAT`.

The format of the file is as follows:

```
EQUATION 1
VAR1
VAR2
⋮
END
EQUATION 2
VAR1
VAR2
⋮
END
⋮
EQUATION n
VAR1
VAR2
⋮
END
;
```

The first line for each equation gives the equation number. The names of the first stage regressors follow the first line for each equation, one name per line. The list of first stage regressors for each equation ends with the statement `END`. The overall file ends with a semicolon on a separate line. The variable names can be any names used in the job. Lags are denoted with minus signs, but are not put in parentheses. For example, `Y1` lagged twice would be written

```
Y1 -2
```

The first stage regressors can be printed using the `PRINTFSR` command:

PRINTFSR ;

This command is useful to make sure that the program has the first stage regressors right.

5.2 Setting Coefficient Values

The coefficients in the model are stored in a matrix of dimension `MAXS` by `MAXCOEF`. The `COEF` command allows the user to set coefficients to particular values. The command is

---

COEF ;

```
position equation value
position equation value
⋮
0 or blank line
```

`position` is the position of the coefficient within the equation. `equation` is the number of the equation. `value` is the value of the coefficient. The last line of the command is a 0 or a blank line.

For a given equation, the autoregressive coefficients for the error term come last in the COEF matrix. Coefficients that are constrained using the `CTC` command come between the unrestricted structural coefficients and the autoregressive error term coefficients. For equations specified using the `NLEQ` command, the order of the coefficients is as specified in the `NLEQ` command. You can use the `PRINTCOEF` command discussed below to see how the coefficients are ordered in the COEF matrix.

5.3 Writing and Reading Coefficients

The coefficient matrix can be written to a file using the `WRITECOEF` command. It may be that the user wants to write more than one set of coefficient estimates to the same file, and, as discussed below, the `WRITECOEF` command has this option. The command is

---

WRITECOEF option ;

The option is:

`FILE=fn`	`fn` is the name of the file to which the coefficients are to be written. The default name is `COEF.DAT`. If after writing one set of coefficients to a file, you want to write another set after it, then `fn` should be specified to be `SAME`. As many copies can be written in this way as desired—just always take `fn` to be `SAME` after the first time. If after writing to a file one or more times, you want to then read it, you must first "close" the data set. This is done by taking `fn` to be `CLOSE`. If `fn` is taken to be `CLOSE`, the file currently being written to by the `WRITECOEF` command is closed—no further writing is done.

An example may help to clarify the use of `FILE=SAME`. Say that you want to write a set of coefficients to file `COEF1.DAT`. The command is `WRITECOEF FILE=COEF1.DAT`. If you then reestimate the model and want to write the new set of coefficients to file `COEF1.DAT` and have this set come after the first set, the command is `WRITECOEF FILE=SAME`. If you instead had said `WRITECOEF FILE=COEF1.DAT`, the program would write over the first set of coefficients. If you want to read the first set of coefficients back in, you must first close the file. The command is `WRITECOEF FILE=CLOSE`. The file is also closed if you use the `WRITECOEF` command and write to a different file. If, for example, you use `WRITECOEF FILE=COEF2.DAT` after using `WRITECOEF FILE=COEF1.DAT` (and possibly some `WRITECOEF FILE=SAME` commands), file `COEF1.DAT` will be closed.

The (FORTRAN) format of the file that is created by `WRITECOEF` is

```
    DO 100 K=1,MAXS
```

```
 100 WRITE( ) (C(I,K),I=1,MAXCOEF)
```

where `C` is `REAL*8` and `MAXS` and `MAXCOEF` are read in from the `SPACE` command.

The coefficients can be read back in using the `READCOEF` command:

READCOEF option ;

The option is:

`FILE=fn`	`fn` is the name of the file from which to read. The default name is `COEF.DAT`. If after reading one set of coefficients from a file, you want to read another from the same file, then `fn` should be specified to be `SAME`. As many copies can be read in this way as desired—just always take `fn` to be `SAME` after the first time. If after reading from a file one or more times, you want to then write to it, you must first "close" the data set. This is done by taking `fn` to be `CLOSE`. If `fn` is taken to be `CLOSE`, the file currently being read by the `READCOEF` command is closed—no further reading is done. (Remember that if the file is written to, anything currently in the file will be written over.)

The coefficients can be printed using the `PRINTCOEF` command:

PRINTCOEF options ;

The options are:

`FILEOUT=fn`	The default is to print to the piped output file (or to the screen if the program is being used interactively). If `FILEOUT=fn` is used, the coefficients are written to file `fn`. NOTE: The `WRITECOEF` command writes the coefficients to a binary file, whereas the `PRINTCOEF` command writes them to an ASCII file. From the ASCII file it is easy for the user to read the coefficients into another program.

BOOTSTRAP This option prints (in binary) the coefficients and their t-statistics to file TEMP79.BIN. No other printing is done. Do not use this option with any other option. TEMP79.BIN is read by the DISTCOEF command.

BOOTSTRAPOUTER
This option prints (in binary) the coefficients and their t-statistics to file TEMP79.BIN. No other printing is done. Do not use this option with any other option. TEMP79.BIN is read by the DISTCOEF command.

BOOTSTRAPINNER
This option prints (in binary) the coefficients and their t-statistics to file TEMP75.BIN. No other printing is done. Do not use this option with any other option. TEMP75.BIN is read by the DISTCOEF command.

The (FORTRAN) format of the file that is created by PRINTCOEF (when the bootstrap options are not in effect) is:

```
     DO 100 K=1,MAXS
     WRITE( ,401) K
 100 WRITE( ,402) (C(I,K),I=1,MAXCOEF)
 401 FORMAT(1X,'EQUATION',I5)
 402 FORMAT(1X,4G19.10)
```

where C is REAL*8 and MAXS and MAXCOEF are read in from the SPACE command.

See the discussion of the COEF command for the order of the coefficients in the COEF matrix.

5.4 Writing and Reading COV

COV is the covariance matrix of all the coefficient estimates in the model. It can be written to a file using the WRITECOV command. Like the WRITECOEF command, the WRITECOV command allows more than one matrix to be written to a file. As noted in the discussion of the `MAXCOV=n` option of the `SPACE` command, the number of elements written out is (`n`(`n`+1))/2 even if the actual size of the covariance matrix is smaller. The command is:

WRITECOV options ;

The options are:

`FILE=fn`	`fn` is the name of the file to which COV is written. The default name is `COV.DAT`. If after writing one copy of COV to a file, you want to write another copy after it, then `fn` should be specified to be `SAME`. As many copies can be written in this way as desired—just always take `fn` to be `SAME` after the first time. If after writing to a file one or more times, you want to then read it, you must first “close” the data set. This is done by taking `fn` to be `CLOSE`. If `fn` is taken to be `CLOSE`, the file currently being written by the `WRITECOV` command is closed—no further writing is done. The number of elements written out is (`n`(`n`+1))/2, where `n` is declared in the `MAXCOV=n` option of the `SPACE` command. If the actual covariance matrix is smaller than `n`×`n`, the `WRITECOV` and `READCOV` commands will still work. The only cost is that extra disk space will be used. (The “extra” elements written out are zeros.)
`BLOCKDIAG`	If this option is used, the COV matrix is written out in block diagonal form. WARNING: This option will allow COV to be written out in block diagonal form even if the matrix is not block diagonal. The off diagonal blocks will simply be ignored.
`DIAG`	If this option is used, only the diagonal elements of COV are written to the file.

The COV matrix can be read back in using the `READCOV` command:

READCOV options ;

The options are:

`FILE=fn`	`fn` is the name of the file from which COV is read. The default name is `COV.DAT`. If after reading one copy of COV from a file, you want to read another from the same file, then `fn` should be specified to be `SAME`. As many copies can be read in this way as desired—just always take `fn` to be `SAME` after the first time. If after reading from a file one or more times, you want to then write to it, you must first "close" the data set. This is done by taking `fn` to be `CLOSE`. If `fn` is taken to be `CLOSE`, the file currently being read by the `READCOV` command is closed—no further reading is done. (Remember that if the file is written to, anything currently in the file will be written over.
`BLOCKDIAG`	If this option is used, the COV matrix is read in block diagonal form. This option can only be used if the COV matrix has originally been written out in block diagonal form.

The COV matrix can be printed using the `PRINTCOV` command:

PRINTCOV options ;

The options are:

`DIAG`	Print only the diagonal elements of the covariance matrix.
`HALF`	Print only the lower triangular portion of the matrix.
`CORR`	Print the matrix as a correlation matrix instead of as a covariance matrix.
`FILEOUT=fn`	The default is to print to the piped output file (or to the screen if the program is being used interactively). If `FILEOUT=fn` is used, COV is written to file `fn`. NOTE: The `WRITECOV` command writes COV to a binary file, whereas the `PRINTCOV` command writes it to an ASCII file.

5.5 Estimating the S Matrix

S is the covariance matrix of the error terms. For models other than models with rational expectations, S can be estimated using the `ESTS` command. These include models in which some or all of the equations are specified using the `NLEQ` command and models for which the `CTC` command is used. For models with rational expectations, S can be estimated using the command `FIML MAXITERS=0;` (see the discussion in Chapter 13). The `ESTS` command requires that the coefficient estimates be in memory. The command is:

ESTS options ;

The options are:

`SDIAG`	S is taken to be a diagonal matrix—the off diagonal elements are set to zero.
`NSETS=n`	This option allows more than one S matrix to be estimated with one use of the command. Each estimate corresponds to a different sample period. The first sample period is the current one in effect. For the second period the last observation is increased by one; for the third sample period the last observation is increased by one more, and so on. If option `MOVEBEGOBS` is specified, the first observation is also increased by one for each successive period.
`MOVEBEGOBS`	See `NSETS=n` above.
`FILECOEF=fn`	Before each estimate of S, read the coefficients from data set `fn`.
`FILEMODEL=fn`	Before each estimate of S, read the specification of the model from data set `fn`.
`FILES=fn`	Write each estimate of S to file `fn`.
`BLKDIAG=n`	If this option is used, the S matrix is estimated in block diagonal form, with the first block being `n×n` and the second block being the the rest. The off block elements are set to zero.

KEEP If some stochastic equations are excluded from the model, the default option is to set the rows and columns of S corresponding to these equations to zero. This means that S will be singular. This is not done if the KEEP option is used. If the KEEP option is used, the rows and columns are not changed from whatever is currently in them.

FILEZEROS=fn

If this option is used, some elements of S are set equal to zero. The information on what to set to zero is contained in file fn. If fn is specified to be KEYBOARD, the information is typed in from the keyboard. The format of the file (or of the information typed in from the keyboard) is:

```
i1 j1 j2 ...  j10
⋮
0 or blank line
```

i1 is the base equation, and the j's are other equations. The program sets S(i1,j1), ..., S(i1,j10) equal to zero (also S(j1,i1), ..., S(j10,i1)). As many lines can be read as desired. There need not be 10 numbers following i1 on the line. If there are less than 10, the program simply deals with the numbers that are on the line. The file ends with a 0 or a blank line.

WARNING: On return from ESTS the residuals are set to zero for the observations used for the estimation. This nullifies the CREATEU command for similar sample periods.

Individual elements of S can be set using the SETS command. The command is:

SETS n1 n2 v ;

This command sets S(n1,n2)=v and S(n2,n1)=v.

Once S is estimated, it can written to a file using the WRITES command. Like the `WRITECOEF` and `WRITECOV` commands, the `WRITES` command allows more than one copy to be written to a file. The command is:

WRITES options ;

The options are:

`FILE=fn`	`fn` is the name of the file to which S is written. The default name is `S.DAT`. If after writing one copy of S to a file, you want to write another copy after it, then `fn` should be specified to be `SAME`. As many copies can be written in this way as desired—just always take `fn` to be `SAME` after the first time. If after writing to a file one or more times, you want to then read it, you must first "close" the data set. This is done by taking `fn` to be `CLOSE`. If `fn` is taken to be `CLOSE`, the file currently being written to by the `WRITES` command is closed—no further writing is done.

The S matrix can be read back in using the `READS` command:

READS option ;

The option is:

`FILE=fn`	`fn` is the name of the file from which S is read. The default name is `S.DAT`. If after reading one copy of S from a file, you want to read another from the same file, then `fn` should be specified to be `SAME`. As many copies can be read in this way as desired—just always take `fn` to be `SAME` after the first time. If after reading from a file one or more times, you want to then write to it, you must first "close" the data set. This is done by taking `fn` to be `CLOSE`. If `fn` is taken to be `CLOSE`, the file currently being read by the `READS` command is closed—no further reading is done. (Remember that if the file is written to, anything currently in the file will be written over.)

The S matrix can be printed using the `PRINTS` command:

PRINTS options ;

The options are:

`FILEOUT=fn`	The default is to print to the piped output file (or to the screen if the program is being used interactively). If `FILEOUT=fn` is used, S is written to file `fn`. NOTE: The `WRITES` command writes S to a binary file, whereas the `PRINTS` command writes it to an ASCII file.
`CORR`	S is printed as a correlation matrix instead of a covariance matrix.

5.6 Nonlinearity in Coefficients

Equations that are nonlinear in coefficients are specified using the `NLEQ` command, which was discussed in Section 4.7. Equations specified by the `NLEQ` command cannot be estimated using the `OLS`, `2SLS`, `LAD`, and `2SLAD` commands. Single equation estimation is done using the `NLOLS` and `NL2SLS` commands (there are no `NLLAD` and `NL2SLAD` commands in the program). The commands are:

NLOLS or NL2SLS options ;

The options are:

`EQUATION=n`	`n` is the equation number to estimate. The default is 1.
`NCOEF=n`	`n` is the number of unrestricted coefficients to estimate in the equation.
`PLOT=n`	Plot the last n residuals of the sample period. If n is 0, all the residuals are plotted. The default is no plots.
`MAXITERS=n`	`n` is the maximum number of iterations for the DFP algorithm. The default is 20.

`FILESTART=fn`
: file `fn` contains the starting values, one value per line. The program reads the number of values as specified by the `NCOEF` option. If `fn` is `KEYBOARD`, the reading of the values is done from the input file (or from the keyboard if the program is being used interactively), again one value per line. If the `FILESTART=fn` option is not specified, the starting values are taken to be zero.

`FILESAVE=fn` If this option is used, the final values are written to file `fn`. This file can be read using the `FILESTART=fn` option in future uses of the `NLOLS` or `NL2SLS` command if desired.

`FILEHSAVE=fn`
: If this option is used, the final value of the H matrix used by the DFP algorithm is written to file `fn`.

`FILEHSTART=fn`
: If this option is used, the H matrix written to file `fn` in a previous use of the `NLOLS` or `NL2SLS` command is read and used as the starting value of H. Using this option allows the DFP algorithm to begin where it left off. Otherwise, the initial value of the H matrix is the identity matrix.

The DFP algorithm is used to estimate the equations, which is the same algorithm used for the `MAX` and `OPTC` commands. The options set by the command `SETUPDFP`, which is explained in Chapter 8, are also relevant for the `NLOLS` and `NL2SLS` commands.

Remember that the `NLEQ` command requires that the `EQ` command be used first to number the equation. The number on the `EQ` command is the number for the `EQUATION=n` option.

It is important to be clear on how to use the `NCOEF=n` option. Say that equation 1 is to be estimated. If coefficients from other equations appear in the `NLEQ` specification for equation 1, as is the case in the second `NLEQ` example in Section 4.7, these coefficients are not estimated. `NCOEF=n` should *not* include these coefficients. Also, if there are, say, 5 coefficients for equation 1 used in the `NLEQ` command and `NCOEF` is set to 4 in the `NL2SLS` command, the program will not estimate the fifth coefficient. The value for the fifth coefficient will remain whatever it is in memory at the time.

6 CHAPTER 6: Testing Single Equations

This chapter presents methods for testing single equations. These methods are discussed in Section 2.8 in MM, and this section should be read before reading this chapter.

6.1 Chi-Square Tests

Many single equation tests are simply of the form of adding a variable or set of variables to an equation and testing whether the addition is statistically significant. The commands `TEST1`, `TEST2`, and `TESTH` allow these tests to be easily performed. The tests are chi-square tests. `TEST1` is for the OLS estimator, `TEST2` is for the 2SLS estimator, and `TESTH` is for Hansen's estimator.

The `TEST1` and `TEST2` commands are:

TEST1 or TEST2 num v1 ... vj RHO=n ;

`num` is the number of the equation to test. `v1...vj` are names of variables to add as explanatory variables to the equation. The order of the autoregressive process of the error term can be modified using the `RHO=n` option, where `n` is the order of the process.

If, for example, the command is `TEST2 4 v1 v2;`, the program will add `v1` and `v2` to equation 4 and test whether they are jointly significant. The chi-square value will be printed out along with its p value. Upon completion of the command, the equation in memory is as it was before (i.e., it does *not* include `v1` and `v2`). If the command is `TEST2 4 RHO=4`, the program tests if this addition is significant. (If the equation is, say, already estimated under the assumption of `RHO=1`, then the test is whether the addition of a second through fourth order autoregressive process of the error term is significant.)

For the `TEST2` command, the first stage regressors that are used are the ones currently in memory. `TEST2` does not allow modifications of the first stage regressors. (First stage regressors can be changed using the `MODEQ` command.)

The `TESTH` command is used when *leads* longer than one period are added to the equation. (For leads of one period, the `TEST2` command can be used.) The basic estimation technique is Hansen's (1982) estimator. This estimator requires that the moving average process of the error term be specified, which should be one less than the longest lead. The command is:

TESTH num ma v1 ... vj RHO=n ;

`num` is the number of the equation to test. `ma` is the order of the moving average process of the error term. `v1...vj` are names of variables to add as explanatory variables to the equation. The order of the autoregressive process of the error term can be modified using the `RHO=n` option, where `n` is the order of the process. As with the `TEST2` command, the first stage regressors that are used are the ones currently in memory.

6.2 Stability Tests

Andrews and Ploberger (1994) have proposed a class of stability tests that does not require that the date of the structural change be chosen *a priori*. One of these tests can be performed using the `TESTSTAB1` and `TESTSTAB2` commands, the first for the OLS estimator and the second for the 2SLS estimator. The commands are:

TESTSTAB1 or TESTSTAB2 num p1 p2 ;

`num` is the number of the equation to test. `p1` is the first observation for which the user specifies there might be a structural break, and `p2` is the last such observation. (The overall sample period is the one currently in memory at the time the command is used.) The program prints out the Andrews-Ploberger (AP) test value along with the individual chi-square values that are used in computing AP.

Andrews (2003) has proposed an end of sample stability test, where there may be fewer observations at the end than coefficients. This test can be performed using the `TESTEND1` and `TESTEND2` commands, the first for the OLS estimator and the second for the 2SLS estimator. The commands are:

TESTEND1 or TESTEND2 num p NAMEMISS=v

`num` is the number of the equation to test. `p` is the first observation of the period at the end of the sample to be tested. (The overall sample period is the one currently in memory at the time the command is used.) `v` is the name of a variable that can be used for temporary storage. It can be any variable not in the equation being tested. It is not permanently changed. The program prints out the p-value of the test.

Andrews test can also be used to test for stability at the beginning of the sample period. The commands in this case are `TESTBEG1` and `TESTBEG2` commands, the first for the OLS estimator and the second for the 2SLS estimator. The commands are:

TESTBEG1 or TESTBEG2 num p NAMEMISS=v

`num` is the number of the equation to test. `p` is the first observation following the period at the beginning of the sample to be tested. (The overall sample period is the one currently in memory at the time the command is used.) `v` is the name of a variable that can be used for temporary storage. It can be any variable not in the equation being tested. It is not permanently changed. The program prints out the p-value of the test.

6.3 Setup Options

The `TEST1`, `TEST2`, and `TESTH` commands run two regressions per call (one with and one without the additions), and the `TESTSTAB1` and `TESTSTAB2` commands run one regression per each possible break in the sample period. The default is *not* to print out the results from these regressions (only the final test values). If you want all of the results printed, you can specify this using the `SETUPTEST` command:

SETUPTEST options ;

The options are:

`PRINTTEST`	If this option is in effect, all the regression results from the `TEST1`, `TEST2`, and `TESTH` commands are printed. The default is not to print the regression results. [LA(393)=1]
`-PRINTTEST`	Undo `PRINTTEST`.

`PRINTSTAB`	If this option is in effect, all the regression results from the `TESTSTAB1` and `TESTSTAB2` commands are printed. WARNING: This can be a lot of printing. [LA(394)=1]
`-PRINTSTAB`	Undo `PRINTSTAB`.

`BOOTSTRAPSTAB`
Do not print the chi-square values. [LA(442)=1]

`-BOOTSTRAPSTAB`
Undo `BOOTSTRAPSTAB`.

SETUPTESTEND options ;

The options are:

`PRINTEND`	If this option is in effect, all the regression results from the `TESTEND1`, `TESTEND2`, `TESTBEG1`, and `TESTBEG2` commands are printed. WARNING: This can be a lot of printing. [LA(438)=1]
`-PRINTEND`	Undo `PRINTEND`.

7 CHAPTER 7: Full Information Estimation

7.1 Three Stage Least Squares (3SLS)

Three stage least squares estimates are obtained using the `3SLS` command. The command requires a prior estimate of S, which the program takes to be whatever is in memory at the time the `3SLS` command is used. The command also requires a list of first stage regressors in the file named in the `FILEFSR` option below. The 3SLS estimator is discussed in Section 2.4 in MM. The estimates are obtained by minimizing (6.24) in Section 6.3.3 in Fair (1984), which is linked to in Section 2.4 in MM.

The `3SLS` command is:

3SLS options ;

The options are:

`MAXFSR=n`	`n` is the maximum number of first stage regressors. The default value is 40.
`FILEFSR=fn`	`fn` is the name of the file containing the list of first stage regressors. The default name is `FSR3.DAT`. The format of this file is one variable name per line, with a semicolon on a separate line to end the list. Lags are denoted `-1`, `-2`, etc. after the variable name with *no* parentheses. `fn` cannot be `KEYBOARD` for this option.
`NAMECNST=name`	`name` is the name of the constant term in the model. The default name is `CNST`. The Parke algorithm needs to know name of the constant term.
`FILESTART=fn`	file `fn` contains the coefficient matrix to use as the starting matrix. If this option is not used, the coefficient matrix currently in memory is used as the the starting matrix. If the option is used, the file must have been created earlier by the program (such as using the `WRITECOEF` command or using the `FILECOEF=fn` option of the `2SLS` command).

`FILECOEF=fn` `fn` is the name of the file to write the final set of coefficients to. If this option is not used, the coefficients are not written to a file. Upon completion of the `3SLS` command, the coefficient estimates are in memory, and if desired, they can be written to a file using the `WRITECOEF` command. The coefficients that are written out either using the `FILECOEF=fn` option or the `WRITECOEF` command can be used as starting values for further 3SLS iterations using the `FILESTART=fn` option above.

`COV` Estimate the 3SLS covariance matrix along with the coefficients. The default is to estimate only the coefficients.

WARNING: To be on the safe side when using the `COV` option, you should set `MAXCOV` in the `SPACE` command to the *exact* number of rows (and columns) in the covariance matrix.

`FILECOV=fn` `fn` is the name of the file to write COV to. If this option is not used, COV is not written to a data set. COV can also be written to a file using the `WRITECOV` command after the `3SLS` command is completed. Upon completion of the `3SLS` command, COV is in memory if the `COV` option has been used.

`FILEFIXED=fn` This option allows some of the coefficients not to be estimated by `3SLS`. Coefficients that are "fixed" are not estimated. They are taken to be whatever is in memory at the time the `3SLS` command is used. file `fn` contains the information on the coefficients to fix. The format of this file is:

```
k i
⋮
0 or blank line
```

`k` is the equation number and `i` is the number of the coefficient in the equation that is not to be estimated. If all of the coefficients in a equation are to be fixed, then `i` should be 0. In this case, only one line is needed per equation. If `fn` is specified to be `KEYBOARD`, the information is read in from the input file (or from the keyboard if the program is being used interactively), with a 0 or blank line to end the reading.

`LINEAR`	This option can be used if the model is linear and contains no autoregressive error terms. It requires that the model first be estimated by `2SLS` using the `SAVE3SLS` option for all the estimated equations in the model. The entire model must have been estimated by a single use of the `2SLS` command. Also, the same set of first stage regressors should be used for all the estimated equations using `2SLS` for this option to be valid.
`SDIAG`	Take S to be a diagonal matrix for the estimation.
`MAXITERS=n`	`n` is the maximum number of iterations for the Parke algorithm. The default is 10.
`TOLCOEF=v`	`v` is the tolerance criterion in terms of the coefficients. The default is .001.
`TOLOBJ=v`	`v` is the tolerance criterion in terms of the objective function. The default is .0001.
`NOMEANS`	This option turns off the part of the Parke algorithm that uses the information on the means of the variables. This option should be used if the `DEVMEAN` command, which is discussed below, has been used.

WARNING: Unless changed by the `FILEFIXED=fn` option, the `3SLS` command assumes that all specified equations are in the model. Upon exit from the `3SLS` command, all specified equations are assumed to be in the model regardless of what was specified by the `FILEFIXED=fn` option.

7.2 Full Information Maximum Likelihood (FIML)

The FIML estimator is discussed in Section 2.4 in MM. See in particular Sections 6.3.4 and 6.5.2 in Fair (1984), which are linked to in Section 2.4 in MM. The estimation of a subset of the coefficients by `FIML` (the `FILEFIXED=fn` option below) is discussed in Section 6.4.2 in Fair (1984), which is also linked to in MM.

In order to obtain FIML estimates, the program needs to know the Jacobian. The nonzero derivatives are specified one per line by the `JACOB` command:

JACOB DERIV(i,j) = derivative ;

`derivative` is the derivative of equation j with respect to endogenous variable i. Equation j may be either a stochastic equation or an identity.

The main source of confusion about the `JACOB` command is how the equations and variables are to be ordered. In fact, the ordering can be anything that the user wants. For example, the ordering of the stochastic equations need not match the ordering of the stochastic equations specified by the `EQ` command. Likewise, the ordering of the identities need not match the order of the `IDENT` commands in the program. The first step is to write the equations in the desired order with 0 on the left hand side. For the model in Chapter 1, this could be:

$$\begin{aligned} 0 &= \log C_t - c_{11} - c_{21}\log C_{t-1} - c_{31}\log Y_t - c_{41}R_t - v_{1t} \quad (1)\\ 0 &= \log I_t - c_{12} - c_{22}\log I_{t-1} - c_{32}\log Y_t - c_{42}R_{t-1} - u_{1t} \quad (2)\\ 0 &= Y_t - C_t - I_t - G_t \quad (3)\end{aligned}$$

In this setup the stochastic equations are ordered as they are numbered by the `EQ` commands and the identity is put as the third equation. The order chosen for the variables is:

```
1.  C
2.  I
3.  Y
4.  R
5.  G
6.  CNST
7.  LOGC
8.  LOGI
9.  LOGY
```

The derivatives using this ordering are then:

```
JACOB DERIV(1,1)= 1/C;
JACOB DERIV(3,1)= -COEF(3,1)/Y;
JACOB DERIV(2,2)= 1/I;
JACOB DERIV(3,2)= -COEF(3,2)/Y;
JACOB DERIV(3,3)= 1;
JACOB DERIV(1,3)= -1;
JACOB DERIV(2,3)= -1;
```

To repeat, then, to specify the Jacobian, order the equations in a convenient way, order the variables, and take the derivatives. A good check on whether you have taken the derivatives right is to have a friend independently take the derivatives and then compare answers. It is easy to make small mistakes taking derivatives.

Note in the above example that the largest element used in the `DERIV` matrix is 3, even though there are 9 variables in the model. The actual dimension of the Jacobian matrix in the model is 9 x 9. In this example, the program would automatically set the fourth through ninth diagonal elements equal to one, with zeros for the other elements in the fourth through ninth rows and columns. This would be appropriate in this case because variables 4 through 9 are exogenous and/or are not used on the right hand side of the `JACOB` statements. The program looks for the largest element in the `DERIV` matrix specified by the user and sets all diagonal elements beyond this element equal to one. Therefore, you can save on keypunching by ordering your variables in such a way that most or all of the exogenous variables come last. Be warned, however, that the program does no automatic setting of diagonal elements smaller than the largest element in the `DERIV` matrix. If, for example, the statement `JACOB DERIV(5,5)=1;` were added, the program would not automatically set `DERIV(4,4)` to one. The user would have to do this.

Note finally that in many cases there is more than one way to take the derivatives. Consider, for example, `LOGC`, which is variable number 7 above. The equation for `LOGC`, written with 0 on the left hand side, is

$$0 = \mathtt{LOGC} - log(\mathtt{C}) \tag{7}$$

Now, equation (1) above can be considered to have either log`(C)` or `LOGC` in it. If the former, then the derivative of equation (1) with respect to `C` is `1/C`, as used in the example above. If the latter, then the derivative of equation (1) with respect to `LOGC` is 1, and the derivative of equation (7) with respect to `C` is `-1/C`. The new `JACOB` statements would be:

```
JACOB DERIV(7,1)= 1;
JACOB DERIV(1,7)=-1/C;
JACOB DERIV(7,7)=1;
JACOB DERIV(4,4)=1;
JACOB DERIV(5,5)=1;
JACOB DERIV(6,6)=1;
```

Also, `JACOB DERIV(1,1)` would not be set to `1/C`. (In other words, it would not appear as a line.) Note that diagonal elements 4 through 6 had to be set to 1 since the largest element used in the `DERIV` matrix is 7.

The `JACOB` commands need only be specified once if the model is not changed, while the `FIML` command below can be used many times (for different sample periods, for example). You cannot subtract or change derivatives once they have been specified. If the Jacobian is to be changed, you should exit the program and start over.

Once the Jacobian has been specified, the model can be estimated by FIML using the `FIML` command:

FIML options ;

The options are:

`NAMECNST=name`	`name` is the name of the constant term in the model. The default name is `CNST`. The Parke algorithm needs to know the name of the constant term.
`NJACOB=n`	`n` is the number of Jacobian determinants to compute in calculating the value of the likelihood function. The default is 2. See the discussion in Section 6.5.2 in Fair (1984) for the use of this option. For a linear model `n` should be 1. For `n` equal to 2, the program computes the determinant of the Jacobian for the first and last observations and uses this information to approximate the sum over all the observations. For `n` greater than 2, the observations for which determinants are computed are as evenly spaced as possible, with the first and last observations always being used. Exact results can be obtained by setting `n` equal to the number of observations, although this is likely to be expensive.

`FILESTART=fn`	file `fn` contains the coefficient matrix to use as the starting matrix. If this option is not used, the coefficient matrix currently in memory is used as the the starting matrix. If the option is used, the file must have been created earlier by the program (such as using the `WRITECOEF` command or using the `FILECOEF=fn` option of the `2SLS` command).
`FILECOEF=fn`	`fn` is the name of the file to write the final set of coefficients to. The coefficients are written to data set `fn` after each iteration and at the end. file `fn` is rewound before each writing. If this option is not used, the coefficients are not written to a file. Upon completion of the `FIML` command, the coefficient estimates are in memory, and if desired, they can be written to a file using the `WRITECOEF` command. The coefficients that are written out either using the `FILECOEF=fn` option or the `WRITECOEF` command can be used as starting values for further `FIML` iterations using the `FILESTART=fn` option above.
`COV`	Estimate the `FIML` covariance matrix along with the coefficients. The default is to estimate only the coefficients. WARNING: To be on the safe side when using the `COV` option, you should set `MAXCOV` in the `SPACE` command to the *exact* number of rows (and columns) in the covariance matrix.
`FILECOV=fn`	`fn` is the name of the file to write COV to. If this option is not used, COV is not written to a data set. COV can also be written to a file using the `WRITECOV` command after the `FIML` command is completed. Upon completion of the `FIML` command, COV is in memory if the `COV` option has been used.
`FILEFIXED=fn`	This option allows some of the coefficients not to be estimated by `FIML`. Coefficients that are "fixed" are not estimated. They are taken to be whatever is in memory at the time the `FIML` command is used. file `fn` contains the information on the coefficients to fix. The format of this file is:

```
k i
⋮
0 or blank line.
```

	`k` is the equation number and `i` is the number of the coefficient in the equation that is not to be estimated. If all of the coefficients in a equation are to be fixed, then `i` should be 0. In this case, only one line is needed per equation. WARNING: `i` cannot be set to 0 for models with rational expectations. For RE models you must list each coefficient number. If `fn` is specified to be `KEYBOARD`, the information is read in from the input file (or from the keyboard if the program is being used interactively), with a 0 or blank line to end the reading. Do not use this option to fix coefficients that are constrained using the `CTC` command.
`SDIAG`	Take S to be a diagonal matrix for the estimation. [LA(174)=1]
`MAXITERS=n`	`n` is the maximum number of iterations for the Parke algorithm or the DFP algorithm. The default is 10.
`TOLCOEF=v`	`v` is the tolerance criterion in terms of the coefficients. The default is .001. (.0001 best for P,W work.) [SPA(18)]
`TOLOBJ=v`	`v` is the tolerance criterion in terms of the objective function. The default is .0001. (.000001 best for P,W work.) [SPA(19)]
`TRIALSTEP=v`	`v` is the trial stepsize for the coefficient searches for the Parke algorithm. The default is .01. (.001 best for P,W work.) [SPA(50)]
`NOMEANS`	This option turns off the Parke algorithm that uses the information on the means of the variables. [LA(155)=1]
`DFP`	Use the DFP algorithm instead of the Parke algorithm to maximize the likelihood function.
`FILEHSAVE=fn`	If this option is used, the final value of the H matrix used by the DFP algorithm is written to file `fn`.
`FILEHSTART=fn`	If this option is used, the H matrix written to file `fn` in a previous use of the `FIML` command is read and used as the starting value of H. Using this option allows the DFP algorithm to begin where it left off. Otherwise, the initial value of the H matrix is the identity matrix.
`TEST=n`	The larger is `n` (up to a maximum of 3), the more printing is done during each iteration. The default is `n=0`.

WARNING: Unless changed by the `FILEFIXED=fn` option, the `FIML` command, like the `3SLS` command, assumes that all specified equations are in the model. Upon exit from the `FIML` command, all specified equations are assumed to be in the model regardless of what was specified by the `FILEFIXED=fn` option.

Additional options are available if one is going to estimate the COV matrix using `FIML` (the `COV` option of the `FIML` command). These are set by use of the `SETUPFIMLCOV command`. Once an option is set using the `SETUPFIMLCOV` command, it remains in effect until undone by the `SETUPFIMLCOV` command. The command is:

SETUPFIMLCOV options ;

The options are:

`MINCHANGE=v`	`v` is the smallest value of the change in the log of the likelihood function allowed in computing numerical derivatives in the calculation of the covariance matrix. The default is .00001. [SPA(40)]
`MAXCHANGE=v`	`v` is the largest value of the change in the log of the likelihood function allowed in computing numerical derivatives in the calculation of the covariance matrix. The default is .0001. [SPA(41)]
`STEPSIZE=v`	value of the initial step size for the derivatives for the calculation of the covariance matrix. The default is .01. [SPA(42)]
`SKIPSEARCH`	Skip the perturbation search for the derivative step sizes in the calculation of the covariance matrix. [LA(124)]
`-SKIPSEARCH`	Undo `SKIPSEARCH`.
`BRUTEFORCE`	Calculate the covariance matrix without using Parke's procedure. (See Section 6.5.2 in Fair (1984) for a discussion of Parke's procedure.) [LA(126)]
`-BRUTEFORCE`	Undo `BRUTEFORCE`.
`PRINTSTEPS`	Print the steps involved in arriving at the final perturbation size for each coefficient for the calculation of the covariance matrix. [LA(175)]

7.3 Coefficient Nonlinearity

Coefficient nonlinearity is automatically handled by the `3SLS` and `FIML` commands, and so no extra work is needed. In other words, `3SLS` and `FIML` will handle the `EQ` as well as the `NLEQ` specifications. Also, the program automatically accounts for the coefficient restrictions specified in the `CTC` commands. There is, however, one point to keep in mind when using the `3SLS` or `FIML` command with the `NLEQ` specification. Say that equation 1 is specified using the `NLEQ` command and that it has five unrestricted coefficients to estimate. If the `NLOLS` or `NL2SLS` command is used to estimate the equation (with `NCOEF=5`) before the `3SLS` or `FIML` command is used, no extra work is needed. If, however, the `NLOLS` or `NL2SLS` command is not used and the starting coefficients are put in in some other way, then a "dummy" call to `NLOLS` or `NL2SLS` must be used before `3SLS` or `FIML` is used. In this example, the call would be:

```
NLOLS EQUATION=1 NCOEF=5 MAXITERS=0;
```

This tells the program that there are 5 coefficients to estimate in equation 1. The equation is not estimated by this command because `MAXITERS` is set to 0. Remember that in the `NCOEF=n` option for `NLOLS` and `NL2SLS`, `n` is the number of unrestricted coefficients in the equation, not necessarily the total number.

Remember also that the `3SLS` and `FIML` commands are must faster when the `EQ, CTC, READCONSTR` specification is used (whenever possible) than when the `NLEQ` specification is used.

7.4 Taking Deviations from Means

The `DEVMEAN` command redefines the variables to be deviations from their means. This may be useful for some purposes, but it is not of general interest. It is possible once the `DEVMEAN` command has been used to estimate the model using `3SLS`. This is not possible, however, for the `FIML` command.

DEVMEAN options ;

The options are:

`NAMECNST=name`	`name` is the name of the constant term. The default name is `CNST`.
`FILESKIP=fn`	file `fn` contains the names of the variables to skip, i.e., not to take deviations from the mean. If this option is not used, only the constant term is skipped. The format of this file is one variable name per line, with a semicolon on a separate line to end. If `fn` is specified to be `KEYBOARD`, then the reading is done from the input file (or from the keyboard if the program is being used interactively), again one name per line with a semicolon to end.
`UNDO`	This option sets the variable values back to their original values. When this option is used, the `FILESKIP=fn` option should not be used even though it may have been used for the original use of `DEVMEAN`. The `UNDO` option is not relevant if there has not been a previous use of `DEVMEAN`.

The variables that are skipped are not changed. Otherwise, each variable has its mean over the sample period subtracted from it. The constant terms in the stochastic equations are changed to keep the means of the residuals unchanged. The `UNDO` option reverses all of this.

8 CHAPTER 8: Maximizing a General Nonlinear Function

The MAX command below requires that the user specify a FORTRAN subroutine that computes the value of the objective function for a given vector of parameter values. This subroutine must be compiled and linked to the main program. This command is meant for advanced users who are familiar with FORTRAN. The SETUPDFP command below, however, is for the general user. This command is used by the NLOLS, NL2SLS, and OPTC commands and by the DFP option of the FIML command.

The subroutine that is needed for the MAX command is:

```
      SUBROUTINE FUNCM(NP,B,F,K,NOBA,LA,SPA,NBEG,NEND,
     XY,YY)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION B(NP),LA(500),SPA(500),Y(NOBA,1),
     XYY(NOBA,1)
C
C     THE USER DETERMINES F HERE. THE FOLLOWING IS
C     AN EXAMPLE.
C
C     SUM=0.
C     DO 100 J=NBEG,NEND
C     ERR=Y(J,12)-B(1)-B(2)*Y(J-1,12)-B(3)*Y(J,13)-
C    XB(2)*B(3)*Y(J,5)
C 100 SUM=SUM+ERR*ERR
C     F=-SUM
C
      RETURN
      END
```

B is the vector of parameters to determine. NBEG is the first observation of the sample period, and NEND is the last observation. The variable values are in the Y matrix. The Y and YY matrices are the same unless in a previous solution command the user has not reset the predicted values to the actual values. In this case the Y matrix contains the predicted values and the YY matrix contains the actual values. The order of the variables in the Y and YY matrices is same as the order in the program. To see the order, issue the `PRINTNAMES` command when you are in the program. `F` is the value of the objective function that must be passed back to the program. The objective function can be made as complicated as possible, subject to the restriction that only data in the Y and YY matrices are available for use.

The `MAX` command allows the function that is specified in the `FUNCM` subroutine to be maximized. The maximization algorithm is DFP. The `DFP` command is discussed in Section 2.2. in MM, which links to Section 2.5 in Fair (1984). The command is

---

MAX options ;

The options are:

`NVALUES=n`	`n` is the number of values (parameters) to determine. The default is 1. This is the `NP` value in `FUNCM`.
`MAXITERS=n`	`n` is the maximum number of iterations for the DFP algorithm. The default is 20.
`FILESTART=fn`	file `fn` contains the starting values, one value per line. The program reads the number of values as specified by the `NVALUES` option. If `fn` is `KEYBOARD`, the reading of the values is done from the input file (or from the keyboard if the program is being used interactively), again one value per line. If the `FILESTART=fn` option is not specified, the starting values are taken to be zero.
`FILESAVE=fn`	If this option is used, the final values are written to file `fn`. This file can be read using the `FILESTART=fn` option in future uses of the `MAX` command if desired.
`FILEHSAVE=fn`	If this option is used, the final value of the H matrix used by the DFP algorithm is written to file `fn`.
`FILEHSTART=fn`	If this option is used, the H matrix written to file `fn` in a previous use of the `MAX` command is read and used as the starting value of H. Using this option allows the DFP algorithm to begin where it left off. Otherwise, the initial value of the H matrix is the identity matrix.

Various options are available when the DFP algorithm is used. These are set by the `SETUPDFP` command. Once an option is set using the `SETUPDFP` command, it remains in effect until undone by the `SETUPDFP` command. As noted above, the commands that use the DFP algorithm are `NLOLS`, `NL2SLS`, `FIML` (as an option), `OPTC`, and `MAX`. The `SETUPDFP` command is:

SETUPDFP options ;

The options are:

`PRINTOBJ`	Print the values of the objective function and lambda after each iteration. [LA(166)]
`-PRINTOBJ`	Undo `PRINTOBJ`.
`PRINTVALUES`	Print the values of the parameters after each iteration. [LA(173)]
`-PRINTVALUES`	Undo `PRINTVALUES`.
`PRINTSEARCH`	Print information about the search for the best value of lambda after each iteration. [LA(168)]
`-PRINTSEARCH`	Undo `PRINTSEARCH`.
`ONESIDED`	Use one sided numeric derivatives. The default is to use two sided numeric derivatives. [LA(169)]
`-ONESIDED`	Undo `ONESIDED`.
`STEPSIZE=v`	`v` is the value of the numeric derivative stepsize. The default value is .0001. (For RS control variable, .01 is good, not .001. FIML best is .001.) [SPA(31)]
`TOLCOEF=v`	`v` is the stopping criterion in terms of the percentage change in each parameter from one iteration to the next. The default value is .0001. (FIML best is .0001.) [SPA(37)]
`TOLGRADE=v`	`v` is the stopping criterion in terms of each gradient value as a percent of the objective function. The default value is .0000001. [SPA(48)]

`BFGS`	Use the BFGS algorithm. The default is to use the DFP algorithm. [LA(167)=1]
`STEEP`	Use the steepest descent algorithm. The default is to use the DFP algorithm. [LA(167)=2]
`DFP`	Use the DFP algorithm. This option is needed if `BFGS` or `STEEP` has been used as an option previously. [LA(167)=0]

9 CHAPTER 9: Solution

The solution of models, both deterministic simulation and stochastic simulation, is discussed in Section 2.5 in MM.

9.1 Solving a Model Using Deterministic Simulation

A model is solved by deterministic simulation using the `SOLVE` command:

SOLVE options ;

The options are:

`DYNAMIC`	Do a dynamic simulation. The default is to do a static simulation.
`OUTSIDE`	Use this option when the forecast is beyond the period for which there are data on the endogenous variables. If this option is used, the initial values for the current period for the solution procedure are set to the previous period's values if there are no current-period values. If there are current-period values, these values are used.
`FILEVAR=fn`	The default is to print the actual and predicted values of all the variables in the model. If you use the `FILEVAR=fn` option, only the variables listed in data set `fn` are printed. The format of this file is one variable name per line, with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.

`FILEVARPLOT=fn`

The default is to do no plotting. If you use the `FILEVARPLOT=fn` option, the variables listed in file `fn` are plotted. The format for this file is the same as for the file discussed above for the `FILEVAR=fn` option. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively). If the first line is `ALL`, then all the variables in the model are plotted. In this case there is just one line (no semicolon to end the list).

`NORESET` The default upon exit from `SOLVE` is to set the predicted values back to the actual values (unless `OUTSIDE` is used). If you use the `NORESET` option, the predicted values are left unchanged. The variable values in memory are then the predicted values rather than the actual values.

WARNING: If you use the `NORESET` option, the actual values are still in the `YY` matrix. (The predicted values are in the `Y` matrix.) If you want `YY` to contain the predicted values as well, use the `SETYYTOY` command.

`NORESETA` This option is the same as `NORESET` except that the variables specified in `FILEVAR=fn` are *not* reset. All the other variables are reset.

`NORESETB` This option pertains to models with rational expectations. If this option is used, the observations for the current period are *not* reset, but the observations for all variables and all future periods are reset.

`FILEWRITE=fn`

For the MC model. Write predicted values to file `fn`, where 16 past quarters are written first. [LA(360)=1]

`FILEWRITE2=fn`

For the MC model. Write actual values to file `fn`, where 16 past quarters are written first. [LA(364)=1]

If both `FILEVAR` and `FILEVARPLOT` are set equal to `KEYBOARD`, the variables for `FILEVAR` should be listed first (with a semicolon to end the `FILEVAR` reading), followed by the variables for `FILEVARPLOT`.

There are some solution options that are set ahead of time. This is done by the use of the `SETUPSOLVE` command. Once an option is set using the `SETUPSOLVE` command, it remains in effect until undone by the `SETUPSOLVE` command. The command is:

SETUPSOLVE options ;

The options are:

`NOMISS`	Use this option if there are no missing observations within any solution period. It cuts computer time. [LA(25)=1]
`-NOMISS`	Undo `NOMISS`. [LA(25)=0]
`MAXITERS=n`	Maximum number of iterations per period for the Gauss-Seidel algorithm. The default is 100. [LA(74)]
`MINITERS=n`	Minimum number of iterations per period for the Gauss-Seidel algorithm. The default is 1. [LA(127)]
`MAXCHECK=n`	When this option is used, only the first `n` variables are checked for convergence for the Gauss-Seidel algorithm. The default is to check all variables. [LA(121)]
`TOLALL=v`	`v` is the tolerance criterion for the Gauss-Seidel algorithm for all the variables in the model that are not listed in the `FILETOL=fn` option below. The default is .001. [SPA(6)]
`FILETOL=fn`	file `fn` contains the list of variables for which a different tolerance criterion is to be used than the overall criterion given in the `TOLALL=v` option. The tolerance criterion for each variable is also listed. The format of the file is:

```
name  value
  ⋮
;
```

`name` is the variable name, and `value` is the value of the tolerance criterion for that variable. If `fn` is specified to be `KEYBOARD`, the information is read in from the input file (or from the keyboard if the program is being used interactively), with the same format.

`TOLALLABS`	When this option is used, the tolerance criterion is in absolute terms rather than percentage terms for all the variables in the model. (The default is percentage terms.) [LA(71)]
`-TOLALLABS`	Undo `TOLALLABS`.

`FILETOLABS=fn`

file `fn` contains the list of variables for which the tolerance criterion is to be in absolute terms. The format of the file is one variable name per line, with a semicolon on a separate line to end. If `fn` is specified to be `KEYBOARD`, the information is read in from the input file (or from the keyboard if the program is being used interactively), with the same format. If the `FILETOLABS=fn` option is to be used, the `TOLALLABS` option cannot be used.

`DAMPALL=v`	`v` is the damping factor for all the variables in the model that are not listed in the `FILEDAMP=fn` option below. The default value is no damping, which is v equal to 1.0.
`FILEDAMP=fn`	file `fn` contains the list of variables for which a different damping factor is to be used than the overall damping factor given in the `DAMPALL=v` option. The damping factor for each variable is also listed. The format of the file is:

```
name  value
 ⋮
 ;
```

`name` is the variable name, and `value` is the value of the damping factor for that variable. If `fn` is specified to be `KEYBOARD`, the information is read in from the input file (or from the keyboard if the program is being used interactively), with the same format.

`NOMSG`	When this option is used, no message is printed as to the number of iterations it took for convergence for each period. [LA(76)=1] If you include the command `SETLA 120 1;` before the `SOLVE` command, the message "Error in RESIDT" is not printed when there is such an error.
`-NOMSG`	Undo `NOMSG`.
`DEBUGPRINT`	Print the variable values after each pass through the model. This should only be done for debugging purposes because it results in a lot of printing. [LA(77)=1]
`-DEBUGPRINT`	Undo `DEBUGPRINT`.
`MAXERR=n`	`n` is the maximum number of errors that the program allows before stopping if the program is being run in batch mode using an input file rather than from the keyboard. If `n` is 0 (the default value), the program does not stop until `EXIT` or `QUIT` is reached.
`TESTEQS`	This option can be used for helping to debug a model. When it is in effect, the model is solved in the following way. First, the stochastic equations are solved using the actual values of all the right hand side variables. Second, the `LHS` statements are solved using the predicted values from the stochastic equations. Third, the `IDENT` and `GENR` statements are *not* solved. This solution option allows one to examine how the stochastic equations and their associated `LHS` variables fit when they are solved individually. Any of the solution commands can be used when the `TESTEQS` option is in effect, although the most natural command to use in this case is the `RMSE` command. [LA(403]
`-TESTEQS`	Undo `TESTEQS`.

If two or more of `FILETOL`, `FILETOLABS`, and `FILEDAMP` are set equal to `KEYBOARD`, the order of the reading is 1) `FILEDAMP`, 2) `FILETOL`, and 3) `FILETOLABS`.

9.2 Creating and Printing Residuals

The default for the program is to set the residuals equal to zero when solving a model. For some purposes it is useful to have the residuals be equal to the estimated residuals. This is done using the `CREATEU` command:

CREATEU options;

The options are:

`ZEROMEAN`	If this option is in effect the residuals are adjusted so that their mean is zero over the sample period in effect.
`LIMIT=n`	The default is to create residuals for all `MAXS` equations, where `MAXS` is set in the `SPACE` command. If you use LIMIT=n, only the residuals for the first `n` equations are created. The residuals for any equations beyond `n` are left alone (they remain at whatever values are in memory).
`BIASCORRECTION`	If this option is in effect, the constant term in each equation is adjusted so that the residuals have mean zero over the sample period in effect. For this option to work the constant term must be the first coefficient in the equation. This option is sometimes used when bootstrapping. Don't use this option with the `ZEROMEAN` option.

This command sets the residuals equal to the estimated residuals for the sample period in effect. This means that if you solve the model with the exogenous variables equal to their historical values, a perfect tracking solution is obtained. See Step 5 in Section 1.3 for a use of this option.

The default uses Y for the actual values. If you set `SETLA 407 1;`, YY is used instead.

If you include the command `SETLA 118 1;` before the `CREATEU` command, it sets the residuals for the particular equation being estimated over the estimation `SMPL` for the equation.

WARNING: The `CREATEU` command does not work for rational expectations models. See the discussion at the end of Chapter 13.

The residuals can be set back to zero by use of the `ZEROU` command:

ZEROU option;

The option is:

`LIMIT=n`	The default is to set the esiduals to zero for all `MAXS` equations, where `MAXS` is set in the `SPACE` command. If you use LIMIT=n, only the residuals for the first `n` equations are set to zero. The residuals for any equations beyond `n` are left alone (they remain at whatever values are in memory). [LA(396]

The residuals can be printed using the `PRINTU` command:

PRINTU option ;

The option is:

`FILEEQ=fn`	The default is to print the residuals for all the equations (for the sample period in effect). If you use the `FILEEQ=fn` option, only the residuals are printed for the equations listed in file `fn`. The format of the file is one equation number per line, with a 0 or a blank line on the last line to end. If `fn` is specified to be `KEYBOARD`, the equation numbers are read in from the input file (or from the keyboard if the program is being used interactively), again one number per line, with a 0 or a blank line to end.

9.3 Debugging a Model

A common problem that model builders have is that after a model is specified and estimated, it will not solve. There are a number of features in the program that allow one to determine why a model is not solving. After a model has been estimated, the first thing to try is the `CHECK` command:

CHECK options ;

The options are:

`IDENT`	Check the `IDENT` commands. You should check for one observation only to save on printing.
`LHS`	Check the `LHS` commands. You should check for one observation only to save on printing.
`RESID`	Compute sum of squared residuals for each equation. You should take the sample period to be the estimation period for this option. WARNING: If this option is used, you cannot use the `NOSMATRIX` option of the `SPACE` command.
`SOLVE`	Check the overall solution of the model. You should check for one observation only to save on printing. Use the `CREATEU` command before using this option.

For the `IDENT`, `LHS`, and `SOLVE` options, the program prints the actual and predicted values for all the variables in the model for the sample period in effect. For these options it is best to have the sample period just be one month, quarter, or year. If any of the `IDENT` or `LHS` commands have been incorrectly specified for a variable, the actual and predicted values of that variable will differ (aside from rounding error), and so these errors are easy to spot.

The `RESID` option computes the sum of squared residuals for each stochastic equation for the sample period in effect. For this option the sample period should be the estimation period. When this is true, the sum of squared residuals for an equation should be the same as the sum of squared residuals that was printed out at the time of estimation. The `RESID` option thus allows the estimation of the equations to be checked.

The `SOLVE` option should be used after the `CREATEU` command has been used. If this is done, the `SOLVE` option should result in the actual and predicted values of all variable being equal (aside from rounding error). In other words, a perfect tracking solution should be generated. Remember that this only needs to be done for one observation.

If all the `CHECK` options have been successfully passed and yet the model will not solve unless `CREATEU` is in effect, then more work needs to be done. It may be that the Gauss-Seidel algorithm that is used to solve the model needs to be damped, and so the next step is to try the options in `SETUPSOLVE`. The `DAMPALL=v` option in `SETUPSOLVE` damps everything, which may be too much damping. You may want to use the `FILEDAMP=fn` option for specific damping of variables. For this option the damping factor can differ across variables. You may also want to change the tolerance criterion. The `FILETOL=fn` option allows you to specify a separate tolerance criterion for each equation. See also the `FILETOLABS=fn` option.

If damping and different tolerance criteria do not work, try the `DEBUGPRINT` option in the `SETUPSOLVE` command. This option results in printing after each pass through the model (a lot of printing), which may help you see what is happening.

Finally, if none of the above works, it may be that the model simply does not have a solution. The final step is to start dropping stochastic equations from the model using the `EXOGENOUS` command. You can keep dropping equations until the model solves. Once it has solved, you can try adding some of the dropped equations back in (using the `ENDOGENOUS` command) to try to narrow the list of bad equations. Once you have found the equation or equations that prevent the model from being solved, you will need to change the equation or equations. If you have passed the `CHECKT SOLVE` test above, you know that if you continue to drop equations you will eventually get a solution, since dropping all stochastic equations is equivalent to using the `CREATEU` command.

There is a special use of the `CHECK LHS;` command for the MC model. This requires the following setup command:

SETUPCHECK options ;

The options are:

`NOPRINT`	For `CHECK LHS` do not print any output. This option is used for the MC model. [LA(422)=1]
`-NOPRINT`	Undo `NOPRINT`.

`NORESET`	For `CHECK LHS` do not reset `Y` to `YY`. This option is used for the MC model. [LA(423)=1]
`-NORESET`	Undo `NORESET`.

9.4 Solving a Model Using Stochastic Simulation

A model is solved by stochastic simulation using the `STOSIM` command. In order to use this command the covariance matrix S of the residuals must be in memory unless historical errors are drawn. In addition, if coefficient draws are to be made, the covariance matrix COV of the coefficient estimates must be in memory. If draws of the exogenous variables are to be made, the `FILESE=fn` option of the `AUTOREG` command must have been used earlier. The simulations for the `STOSIM` command are always dynamic simulations. The command also works for models with rational expectations, which is explained in Chapter 12.

The `STOSIM` command is:

STOSIM options ;

The options are:

`SETU`	Use this option (with no other option) to set the residuals in `STOSIM`. These are the residuals used for the error draws for subsequent uses of the `STOSIM` command when the `DRAWHIST` option below is in effect. The residuals are set to the residuals currently in memory. These residuals must be created ahead of time using the `CREATEU` command. Residuals that have not been created are set to zero. [LA(244)=1]
`RETURNY`	Draw one set of errors for the sample period in effect, solve the model, and return. Upon return do *not* reset `Y` to `YY`. This option only draws errors (not coefficients and exogenous variable values). The other possible options when this option is in effect are `DRAWUHIST`, `FIRSTUHIST=p`, `LASTUHIST=p`, `MCMODEL`, `LIMITEQS=n`, `SEED=v`, `NODRAWU`, `SDIAG`, and `FILEFIXED=fn`. The errors are set to zero on return. [LA(249)=1]

`RETURNU`	Draw one set of errors for the sample period in effect and return. The other possible options when this option is in effect are `DRAWUHIST`, `FIRSTUHIST=p`, `LASTUHIST=p`, `MCMODEL`, `LIMITEQS=n`, `SEED=v`, `NODRAWU`, `SDIAG`, and `FILEFIXED=fn`. [LA(249)=2 or LA(250)=1??]
`NSETS=n`	`n` is the number of stochastic simulations to perform. The default is 1. The length of each simulation period is set by the `LENGTH` option below. The sample period for the first simulation begins with the first period specified by the `SMPL` command currently in effect. For the second simulation (if `n` is greater than 1), the first period is increased by 1; for the third simulation (if n is greater than 2), the first period is increase by 1 more; and so on. The last period that is used for any simulation is the last period specified by the `SMPL` command currently in effect. The length of a simulation may thus be less than the length specified by the `LENGTH` option if the simulation would otherwise run beyond the last period specified. If you have estimated a model a number of times and want to do successive stochastic simulations, this is done using the `NSETS=n` option. The `FILECOEF` option below allows different sets of the coefficients to be used for the simulations. Similarly, the `FILES` option below allows different S matrices; the `FILECOV` or `FILECOVBLKDIAG` option below allows different COV matrices; and the `FILEMODEL` option below allows different specifications of the model. These options allow one first to do successive estimation and then to do successive stochastic simulations.
`LIMITEQS=n`	The default is to draw errors for all the equations in the model. If you use LIMITEQS=n, errors are drawn only for the first `n` equations. The residuals for any equations beyond `n` are left alone (they remain at whatever values are in memory).
`LENGTH=n`	`n` is the length of each simulation period. See the discussion of the `NSETS=n` option. This option is only relevant if the `NSETS=n` option is used. Otherwise, the length is determined by the `SMPL` in effect.
`NTRIALS=n`	`n` is the number of trials per stochastic simulation. The default is 1.

`OUTSIDE`	Use this option when the forecast is beyond the period for which there are data on the endogenous variables. If this option is used, the initial values for the current period for the solution procedure are set to the previous period's values if there are no current-period values. If there are current-period values, these values are used.
`SEED=v`	If you want to set the seed, this can be done using the `SEED=v` option. `v` is the new value of the seed. If this option is not used, the seed is whatever is in memory.
`NODRAWU`	The default is to draw structural error terms for the stochastic simulation. If you use the `NODRAWU` option, the error terms are not drawn—they are taken to be fixed at whatever values are currently in memory (zero unless the `CREATEU` command has been used). For RE models the residuals in memory must be zero.
`SDIAG`	S is taken to be diagonal for purposes of the draws.
`DRAWUHIST`	Draw historical errors, not errors from the distribution with covariance matrix S. This option must be used for the MC model. [LA(416)=1]
`FIRSTUHIST=p`	`p` is the first possible period to use for drawing the historical errors. Remember that the historical errors must be set ahead of time in a separate use of the `STOSIM` command using the `SETU` option. The default for `p` is the first period listed in the `SPACE` command. This option is not relevant unless the `DRAWUHIST` option is used. [LA(417)=p]
`LASTUHIST=p`	`p` is the last possible period to use for drawing the historical errors. Remember that the historical errors must be set ahead of time in a separate use of the `STOSIM` command using the `SETU` option. The default for `p` is the last period listed in the `SPACE` command. This option is not relevant unless the `DRAWUHIST` option is used. [LA(418)=p]
`MCMODEL`	This option pertains to the MC model. The error vectors are drawn for the four quarters of the year. The `DRAWUHIST` option must be in effect for the MC model. LA(419)=1

`DRAWCOEF`	The default is not to draw coefficients. If you use the `DRAWCOEF` option, the coefficients are drawn.
`DRAWEXOG`	The default is not to draw exogenous variable values. If you use the `DRAWEXOG` option, exogenous variable values are also drawn. The `FILEEXOG=fn` option must also be used with this option.
`FILEEXOG=fn`	If the `DRAWEXOG` option is used, the `FILEEXOG=fn` option must be used. file `fn` contains the estimated standard errors for the exogenous variable draws. This file must be created ahead of time using the `AUTOREG` command. See the discussion of the `AUTOREG` command.
`EXOGCHANGES`	The draws of the exogenous variable errors pertain to the changes in the exogenous variables rather than to the levels of the variables. The default is levels. See Fair (1984), Section 8.4.2, especially equation (8.4) on page 268, for a discussion of the change option.
`FILEVAR=fn`	The default is to print the actual and predicted values of all the variables in the model. If you use the `FILEVAR=fn` option, only the variables listed in data set `fn` are printed. The format of this file is one variable name per line, with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.

FILETRIALS=fn

This option prints all the trial values to data set fn for each variable listed in the FILEVAR=fn option. The format is ASCII, so you can read this file into other programs to compute various statistics and event probabilities. The format of the file is:

```
TRIAL 1
Variable name
n values, where n is the length of the
 simulation period, FORMAT(1X,4E19.11)
Variable name
n values
   .
   .
TRIAL 2
   .
   .
```

If you list this file after you have created it, the format should be clear. Remember that there are n values per variable per trial, where n is the length of the simulation period. WARNING: For many trials and variables, this option leads to a lot of output.

FILEMEDIAN=fn

The default is not to compute medians and ranges. If you use the FILEMEDIAN=fn option, medians and ranges are computed for the variables listed in file fn. The format of this file is one name per line, with a semicolon on the last line to end the file. If fn is specified to be KEYBOARD, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.

`FILEFIXED=fn`

The default is to draw error terms for all of the equations. If you want some of the error terms not to be drawn (to be fixed), this can be done using the `FILEFIXED=fn` option. The file `fn` should contain the equation numbers for the equations whose error terms are to be fixed. The format of this file is one number per line, with a 0 or a blank line to end. If `fn` is specified to be `KEYBOARD`, the numbers are read in from the input file (or from the keyboard if the program is being used interactively), again one number per line, with a 0 or a blank line to end. The error terms that are fixed are fixed at whatever values currently exist. They are not touched by the `STOSIM` command. This option does not work for RE models.

`FILECOEF=fn` If this option is used, a new set of coefficient estimates is read for each simulation, beginning with the first simulation. The coefficient estimates are read from data set `fn`.

`FILES=fn` If this option is used, a new S matrix is read for each simulation, beginning with the first simulation. The S matrices are read from file fn.

`FILECOV=fn` If this option is used, a new COV matrix is read for each simulation, beginning with the first simulation. The COV matrices are read from file fn. COV is assumed not to written out in block diagonal form.

`FILECOV1=fn` If this option is used, only one COV matrix is read (I think).

`FILECOVBLKDIAG=fn`

This option should be used in place of `FILECOV=fn` if the COV matrices have been written out in block diagonal form.

`FILEMODEL=fn`

If this option is used, the specification of the model is read for each simulation, beginning with the first simulation. The specifications are read from data set `fn`.

`NPLACE=n`	The default is to begin reading the files listed in the `FILECOEF=fn`, `FILES=fn`, `FILECOV=fn`, `FILECOVBLKDIAG=fn`, and `FILEMODEL=fn` options from the beginning of the files. If the `NPLACE=n` option is used, the reading begins with the nth copy. The first `n-1` copies are read and ignored.
`FILESTOSIM=fn`	Save the results in file `fn` for possible future use. This option has two uses. If only one stochastic simulation is performed (`NSETS=1` above), then the simulation can begin where it left off by using the `FILESTART=fn` option below in a future use of the `STOSIM` command. If you do continue in this way, be sure that the seed is different when you continue from the one that existed at the start. Otherwise you are just duplicating what you did before. The other use is to do successive stochastic simulations (`NSETS` greater than 1) and save the results for use by commands `TABLE8-1`, `TABLE8-2`, and `DVALUES`. See the discussion of these commands in Section 10.2.
`FILESTART=fn`	Read the results in file `fn` before starting. This allows you to pick up where you left off on a previous job. See the discussion of the `FILESTOSIM=fn` option above. The `FILESTART=fn` option does not work for medians and ranges.

If two or more of `FILEVAR`, `FILEMEDIAN`, and `FILEFIXED` are set equal to `KEYBOARD`, the order of the reading is 1) `FILEMEDIAN`, 2) `FILEVAR`, and 3) `FILEFIXED`.

The seed can also be set using the `SEED` command:

SEED option ;

The option is:

`VALUE=v`	`v` is the new value of the seed. `v` must be positive.

If you want to draw a normal random variable with mean 0 and variance 1, this can be done using the `DRAW` command:

DRAW ;

If the `DRAW` command is used, this changes the seed value in memory.

9.5 Estimating Autoregressive Equations

The `AUTOREG` command allows an autoregressive equation to be estimated for any variable in the model. If you are going to draw values of the exogenous variables in the `STOSIM` command, autoregressive equations for the relevant exogenous variables must first be estimated. The command is

---

AUTOREG options ;

```
name of constant term (unless NOCONSTANT is specified)
name of time trend (if TIME is specified)
variable name for regression
⋮
;
```

The options are:

`ORDER=n`	`n` is the order of the autoregressive equation. The default is 1. `n` can be 0.
`NOCONSTANT`	The default is to include a constant term in the regression. If you use the `NOCONSTANT` option, the constant term is not included.
`TIME`	The default is not to include a linear time trend in the regression. If you use the `TIME` option, the time trend is included.
`PLOT`	Plot the residuals for each regression. The default is no plotting.
`FILESE=fn`	Write the estimated standard error for each equation to file `fn`. One standard error is written per line. This file is read by the `STOSIM` command if exogenous variable values are to be drawn.

`LOGLHS`	Take the left hand side variable to be the log of the basic variable.

This command estimates `n`th order autorgressive equations for each of the variables listed. It is important to remember that the equations estimated by the `AUTOREG` command are not part of the basic model. If the user wants her or his model to include autoregressive equations, these must be specified using the `EQ` command.

9.6 Computing Confidence Intervals

In many cases one may want to use the `STOSIM` command to compute confidence intervals of forecasts. The program prints out the mean value and standard error of the forecast of each variable for each period (rows 3 and 4 of the output). If you are willing to assume that the forecast is normally distributed, the mean and standard error can be used to compute confidence intervals in the usual way. If, for example, the mean is 100 and the standard error is 10, the 95 percent confidence interval is about 80.4 to 119.6.

The normality assumption is at best only a good approximation because the forecast is not in general normally distributed even if the error terms are normally distributed. If you are uncomfortable with the normality assumption for the forecast values, you can compute confidence intervals using the `FILEMEDIAN=fn` option of the `STOSIM` command. When this option is used, the program prints out additional results. Row 11, for example, which is labelled, LHS .475, gives for each variable and period the value below which 2.5 percent of the forecast values lie (or the value above which 47.5 percent of the values below the median lie). Row 12, which is labelled, RHS .475, gives the value above which 2.5 percent of the forecast values lie (or the value below which 47.5 percent of the values above the median lie). A 95 percent confidence interval is thus LHS .475 to RHS .475. Similarly, row 13 gives LHS .45 and row 14 gives RHS .45, and so a 90 percent confidence interval is LHS .45 to RHS .45.

The program prints out a number of other results when the `FILEMEDIAN=fn` option is used, most of which are not of general interest. The largest and smallest values are presented (in rows 15 and 16) along with the range (in row 17). The value printed in row 9 is δ_{itk} in equation (7.9), page 254, in Fair (1984), which is linked to in Section 2.6 in MM. If the forecast values were normally distributed and if there were no stochastic simulation error, the value in row 9 would be the same as the standard error in row 4. The differences between row 4 and row 9 can thus be used to judge the effects of deviations of the forecast values from normality.

It should be noted that the normal distribution is always used for the draws of the error terms, coefficients, and exogenous variable errors. The only difference between the median and non median option is that more statistics are computed for the median option.

Finally, it should be noted that results pertaining to the size of the stochastic simulation errors are presented. The square of the standard error in row 4 is the variance, which is presented in row 44. The standard error of this variance is then presented in row 66. The more trials one takes, the smaller will be the standard error in row 66. The error in row 66 is the stochastic simulation error of the estimated variance. If this error is large relative to the size of the estimated variance, then not enough trials have been take to allow much confidence to be placed on the results.

10 CHAPTER 10: Evaluating Predictive Accuracy

The main reference for this chapter is Chapter 8 in Fair (1984). Part of this chapter is linked to in Section 2.9.1 in MM, but not all of the relevant sections. Chapter 8 in Fair (1984) should thus be used as a reference for the commands in this chapter.

Predictive accuracy can be analyzed using either deterministic or stochastic simulation. The deterministic simulation commands in the program are `RMSE` and `RMSEA`. The stochastic simulation commands are `TABLE8-1, TABLE8-3`, and `DVALUES`. In order to use these last three commands, the `STOSIM` command, which is discussed in Chapter 8, must have been used first.

10.1 Deterministic Simulation

There are two commands available to evaluate predictive accuracy using deterministic simulation. The first is the `RMSE` command:

RMSE options ;

The options are:

`DYNAMIC`	Do a dynamic simulation. The default is to do a static simulation.
`FILEVAR=fn`	The default is to print results for all the variables in the model. If you use the `FILEVAR=fn` option, only the results for the variables listed in file `fn` are printed. The format of this file is one variable name per line with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.

`FILEVARPLOT=fn`	The default is to do no plotting. If you use the `FILEVARPLOT=fn` command, the actual and predicted values of the variables listed in file `fn` are plotted. The format of this file is the same as for the file discussed above for the `FILEVAR=fn` option. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively). If the first line is `ALL`, then all the variables in the model are plotted. In this case there is just one line.
`RMSEONLY`	The default is to print the actual and predicted values of the variables as well as the root mean squared errors. If you use the `RMSEONLY` option, only the root mean squared errors are printed.
`NORESET`	The default upon exit from `RMSE` is to set the predicted values back to the actual values. If you use the `NORESET` option, the predicted values are left unchanged. The variable values in memory are then the predicted values rather than the actual values. WARNING: If you use the `NORESET` option, the actual values are still in the `YY` matrix. (The predicted values are in the `Y` matrix.) If you want `YY` to contain the predicted values as well, use the `SETYYTOY` command.
`FILEWRITE=fn`	Write errors and percentage errors to file `fn` for the MC model.

The `RMSE` command is limited in the sense that it can compute RMSEs for one period ahead predictions (when the simulation is static) and for dynamic predictions over the entire sample period. It cannot compute `RMSEs` for n period ahead predictions for n greater than one. This is done by the `RMSEA` command:

RMSEA options ;

The options are:

`FILEVAR=fn`	The default is to print results for all the variables in the model. If you use the `FILEVAR=fn` option, only the results for the variables listed in file `fn` are printed. The format of this file is one variable name per line with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.
`LENGTH=n`	`RMSEA` will compute root mean squared errors for predictions of length 1 through `n`. The default for `n` is 1.
`RMSEONLY`	The default is to print the actual and predicted values of the variables as well as the root mean squared errors. If you use the `RMSEONLY` option, only the root mean squared errors are printed.
`FILECOEF=fn`	If you use this option, a new set of coefficients is read from file `fn` for each new simulation that is run. This can be used to have all the root means squared errors be outside sample. See the discussion below.
`FILEMODEL=fn`	If you use this option, a new specification of the model is read from file `fn` for each new simulation that is run.
`FILEOUT=fn`	If you use this option, the program writes the predicted values to file fn. [LA(67)=1]
`FILEWRITE=fn`	Write errors and percentage errors to file `fn` for the MC model.

To see what `RMSEA` does, assume that the sample period in effect is 1970.1–1979.4 and that `LENGTH=8`. There are 40 quarters between 1970.1 and 1979.4, and `RMSEA` will run 40 simulations. The first simulation starts in 1970.1, the second starts in 1970.2, and so on. The length of the first 33 simulations is 8 quarters. The length of the 34th simulation, which begins in 1978.2, is 7 quarters because the command does not use date beyond 1979.4. Similarly, the length of the 35th simulation is 6 quarters, and so on. There are thus 40 observations for the one quarter ahead `RMSE`s, 39 observations for the two quarter ahead RMSEs, and so on through 33 observations for the eight quarter ahead RMSEs.

If you use the `FILECOEF=fn` option, a new set of coefficients is read for each simulation. In the above example, 40 sets of coefficients would be read. If, say, the first set was based on an estimation period that ended in 1969.4, the second set on one that ended in 1970.1, and so on, then each of the 40 simulations would be outside sample.

10.2 Stochastic Simulation

The three commands in this section pertain to Chapter 8 in Fair (1984). In order to understand these commands, you will have to have read this chapter. The discussion in this chapter is not repeated here.

To use the three commands in this section—`TABLE8-1, TABLE8-3, DVALUES`—you first have to do successive stochastic simulations using the `STOSIM` command described in the last chapter. The `STOSIM` command must be used to create the file specified in the `FILESTOSIM=fn` option. For sake of an example, assume that this data set is called `SIM.DAT`. Then for each of the three commands below, the option `FILESTOSIM=SIM.DAT` must be used. In addition, the sample period used for the three commands must be the same as the sample period that was used for the `STOSIM` command. Also, the options `NSETS=n` and `LENGTH=n` must be the same as were used for the `STOSIM` command.

If after performing successive stochastic simulations you want to compute the equivalent of Table 8-1 in Fair (1984), this is done using the `TABLE8-1` command:

TABLE8-1 options ;

The options are:

`FILEVAR=fn`	The default is to print results for all the variables in the model. If you use the `FILEVAR=fn` option, only the results for the variables listed in file `fn` are printed. The format of this file is one variable name per line with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.

`FILESTOSIM=fn`	`fn` is the name of the file created by the `STOSIM` command using the `FILESTOSIM=fn` option.
`NSETS=n`	`n` must be the same value as used for the `STOSIM` command.
`LENGTH=n`	`n` must be the same value as used for the `STOSIM` command.

If you want to compute the equivalent of Table 8-3 in Fair (1984), this is done using the `TABLE8-3` command:

TABLE8-3 options ;

The options are:

`FILEVAR=fn`	The default is to print results for all the variables in the model. If you use the `FILEVAR=fn` option, only the results for the variables listed in file `fn` are printed. The format of this file is one variable name per line with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.
`FILESTOSIM=fn`	`fn` is the name of the file created by the `STOSIM` command using the `FILESTOSIM=fn` option.
`NSETS=n`	`n` must be the same value as used for the `STOSIM` command.
`LENGTH=n`	`n` must be the same value as used for the `STOSIM` command.

If you want to compute the d values as discussed in Chapter 8 in Fair (1984), this is done using the `DVALUES` command. The command is:

DVALUES options ;

The options are:

`FILEVAR=fn`	The default is to print results for all the variables in the model. If you use the `FILEVAR=fn` option, only the results for the variables listed in file `fn` are printed. The format of this file is one variable name per line with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.
`FILESTOSIM=fn`	`fn` is the name of the file created by the `STOSIM` command using the `FILESTOSIM=fn` option.
`NSETS=n`	`n` must be the same value as used for the `STOSIM` command.
`LENGTH=n`	`n` must be the same value as used for the `STOSIM` command.
`PLOTD=n`	This option results in the d values being plotted over time. One through `n` period ahead d values are plotted.
`FILEOLS=fn`	This option and the next three are used if you want to run regressions with the d values on the left hand side. Data set `fn` contains the names of the right hand side variables.

The format of the file is:

```
variable lag
variable lag
⋮
;
variable lag
variable lag
⋮
;



  ⋮
  variable lag
  variable lag
  ⋮
  ;
  ;
```

`variable` is the name of the explanatory variable, and `lag` is its lag length. (`lag` should be negative. For a lag length of 1, for example, `lag` should be -1.) As many sets of explanatory variables can be read as desired. An extra semicolon ends the entire reading. It is possible to use the d variable lagged once as an explanatory variable even though it is not a variable in the model. This is done by using the special name `DLAG1`. The lag number for `DLAG1` is 0. If `fn` is specified to be `KEYBOARD`, the information is read in from the input file (or from the keyboard if the program is being used interactively).

`OLSLENGTH=n` The d regressions are run for the one through `n` period ahead values of d. The default for `n` is 1.

`PLOTOLS`	If this option is used, the residuals from the d regressions are plotted.
`TEMPVAR=name`	When the d regressions are run, the program needs the name of a variable in the model that will not be used as an explanatory variable in the d regressions. The `TEMPVAR=name` option is used for this. The variable specified by this option is not affected in any way; the program just needs some temporary storage space. The variable is returned safe and sound upon exit from the command.

The advantage of the `DVALUES` command is that it can be used over and over without having to perform any stochastic simulations. The results in the file specified by the `FILESTOSIM` option is merely read over and over.

It will be useful to discuss briefly what the various values that are printed out are. In the printout, "`d`" refers to the d values in absolute units and "`dpc`" refers to the d values in percentage terms (percent of the forecast means). The `d` values are in units of the variables squared, and the `dpct` values are in units of percent squared (1.0 percent squared is .0001).

One should be aware that the d row values in Table 8-2 in Fair (1984) are not the d or `dpct` values printed by the `DVALUES` command. To compute the d row values, the following steps must be followed:

1. Use command `STOSIM` to get the c row values for a particular period. This is one stochastic simulation, with draws of the error terms, coefficients, and exogenous variable errors. Record these values in a table. (The a row and b row values can also be computed by two more uses of `STOSIM`, but these values are not necessary for the computation of the d row values.)

2. Use command `2SLS` to perform successive reestimation of the model, and use command `STOSIM` to perform successive stochastic simulation. When using the `STOSIM` command, use the `FILESTOSIM=fn` option to write the results to a file. If the model has been estimated, say, 50 times, then 50 stochastic simulations will need to be performed.

3. Use the `DVALUES` command to compute the mean of `d` and the mean of `dpct` for each variable and length ahead of the forecast. (The means are automatically printed out by the `DVALUES` command.)

4. For each variable and length ahead, add the mean value of `d` or `dpct` to the square of the c row value and then take the square root of this sum. This value is the final d row value as in Table 8-2.

11 CHAPTER 11: Evaluating Static and Dynamic Properties

The evaluation of static and dynamic properties of models is discussed in Chapter 9 in Fair (1984), and this is the best reference for the commands in this chapter. Both deterministic and stochastic simulation can be used for this purpose.

11.1 Deterministic Simulation

One way to calculate multipliers using deterministic simulation is to use the `CREATEU` and `SOLVE` commands. This is discussed in Step 6 in Section 1.3. Another way is to use the `MULT` command, which is discussed in this section.

The `MULT` command first requires that the `XMULT` command be used. Say that you want to calculate multipliers for an increase in variable `G` of 10. The `XMULT` command in this case is:

```
XMULT G = G + 10;
```

The general form of the `XMULT` command is:

XMULT variable = arithmetic expression ;

The format for `XMULT` is the same as the format for `CREATE` and `GENR`. In the above example `G` on the right hand side is the old value of `G` and `G` on the left hand side is the new value of `G`. The command `XMULT` pertains only to the use of `MULT;` it has no affect on anything else in the model, including in the above example `G`.

You can have as many `XMULT` commands as you like before using `MULT`. All `XMULT` commands are executed when `MULT` is called. The `MULT` command is:

MULT options ;

The options are:

`BOOTSTRAP`	This option prints (in ASCII) the output to `TEMP74.DAT`. Nothing else is printed. `TEMP74.DAT` is read by the `DISTMULT` command. [LA(435)=1]
`MAXXMULT=n`	`n` is the maximum number of `XMULT` commands allowed. The default is 10.
`STATIC`	The simulation is static. The default is dynamic.
`OUTSIDE`	Use this option when the experiments are for a period that is beyond the period for which there are data on the endogenous variables. If this option is used, the initial values for the current period for the solution procedure are set to the previous period's values if there are no current-period values. If there are current-period values, these values are used.
`NOBASERUN`	If this option is used, a base simulation is not run. The predicted values after the exogenous variable changes are simply compared to the actual values. This option can be used if the `CREATEU` command has been used so that the base simulation would result in a perfect tracking solution.
`FILEVAR=fn`	The default is to print results for all the variables in the model. If you use the `FILEVAR=fn` option, only results for the variables listed in file `fn` are printed. The format of this file is one variable name per line, with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.
`FILEVARPLOT=fn`	The default is to do no plotting. If you use the `FILEVARPLOT=fn` option, the variables listed in file `fn` are plotted. The new predicted values are plotted against the base values. The format of the file is one variable name per line, with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end. If the first name is `ALL`, then all the variables are plotted. In this case there is just one line.

`NORESET`	The default upon exit from `MULT` is to set the predicted values back to the actual values (unless `OUTSIDE` is used). If you use the `NORESET` option, the predicted values are left unchanged. The variable values in memory are then the predicted values from the experiment. WARNING: If you use the `NORESET` option, the actual values are still in the `YY` matrix. (The predicted values are in the `Y` matrix.) If you want `YY` to contain the predicted values as well, use the `SETYYTOY` command.

If both `FILEVAR` and `FILEVARPLOT` are set equal to `KEYBOARD`, the variables for `FILEVAR` are read first.

If after doing one `MULT` you want to do another `MULT` with different `XMULT` commands, this can be done, but the old `XMULT` commands must be cancelled. In the above example you would need the command `XMULT G = G`, which simply tells the program that in any future use of `MULT`, `G` is equal to `G`. (The value of `G` in memory is the value of `G` before the `MULT` command was used, not this value plus 10.) The old commands need to be cancelled because the program executes all the `XMULT` commands when `MULT` is called, not just the last one entered.

If you want, say, `G` changed by 10 only in the first period, the easiest way to do this is to create a variable that is 10 in the relevant period and 0 otherwise. If, say, you want variable Z1 to be 10 in 1985.1 and 0 otherwise for the sample period 1952.1–1988.2, the commands are:

```
SMPL 1952.1 1984.4;
CREATE Z1=0;
SMPL 1985.2 1988.2;
CREATE Z1=0;
SMPL 1985.1 1985.1;
CREATE Z1=10 ;
```

The `XMULT` command would then be `XMULT G = G + Z1;` .

11.2 Stochastic Simulation

Computing multipliers by means of stochastic simulation allows standard errors of the multipliers to be estimated. The command to do this is `STOMULT`. Like

MULT, it requires XMULT to be specified first. This command does not work for models with rational expectations. Also, this command, unlike the STOSIM command, does not allow historical errors to be drawn (only errors from estimated distributions). The command is:

STOMULT options ;

The options are:

MAXXMULT=n	n is the maximum number of XMULT commands allowed. The default is 10.
NTRIALS=n	n is the number of trials. The default is 25. If n is 0, then the simulations are deterministic. The results are the same as would be computed using the MULT command.
OUTSIDE	Use this option when the experiments are for a period that is beyond the period for which there are data on the endogenous variables. If this option is used, the initial values for the current period for the solution procedure are set to the previous period's values if there are no current-period values. If there are current-period values, these values are used.
STATIC	Perform static simulations. The default is dynamic simulations.
SEED=v	If you want to reset the seed, this can be done using the SEED=v option. v is the new value of the seed. (Maybe v must be a negative number, but then again maybe not.)
NODRAWU	The default is to draw structural error terms for the stochastic simulation. If you use the NODRAWU option, the error terms are not drawn—they are taken to be fixed at whatever values are currently in memory (zero unless the CREATEU command has been used).
SDIAG	S is taken to be diagonal for purposes of the draws.
FILESAVE=fn	Save the results in file fn for possible future use.

`FILESTART=fn`	Read the results in file `fn` before starting. This allows you to pick up where you left off on a previous job. This option does not work for medians and ranges (see below).
`FILEVAR=fn`	The default is to print the actual and predicted values of all the variables in the model. If you use the `FILEVAR=fn` option, only the variables listed in data set `fn` are printed. The format of this file is one variable name per line, with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end.
`FILEMEDIAN=fn`	The default is not to compute medians and ranges. If you use the `FILEMEDIAN=fn` option, medians and ranges are computed for the variables listed in file `fn`. The format of this file is one name per line, with a semicolon on the last line to end the file. If `fn` is specified to be `KEYBOARD`, the variables names are read in from the input file (or from the keyboard if the program is being used interactively), again one variable name per line, with a semicolon to end. There is a limit of 5 variables for this option.
`FILEFIXED=fn`	The default is to draw error terms for all of the equations. If you want some of the error terms not to be drawn (to be fixed), this can be done using the `FILEFIXED=fn` option. The file `fn` should contain the equation numbers for the equations whose error terms are to be fixed. The format of this file is one number per line, with a 0 or a blank line to end. If `fn` is specified to be `KEYBOARD`, the numbers are read in from the input file (or from the keyboard if the program is being used interactively), again one number per line, with a 0 or a blank line to end. The error terms that are fixed are fixed at whatever values currently exist. They are not touched by the `STOSIM` command.

If two or more of `FILEVAR`, `FILEMEDIAN`, and `FILEFIXED` are set to `KEYBOARD`, the order of the reading is 1) `FILEVAR`, 2) `FILEMEDIAN`, 3) `FILEFIXED`.

12 CHAPTER 12: Optimal Control Analysis

The solution of optimal control problems for macroeconometric models is discussed in Section 10.2 in Fair (1984), which is linked to in Section 2.10 in MM. The command to solve optimal control problems is `OPTC`. Command `OPTC` requires that a target be specified first using the `GENR` command. If, say, variable `YS` is the target for variable `Y` and variable `PS` is the target for variable `P` and if the loss function is quadratic, where `YS` and `PS` are exogenous, the relevant `GENR` command is:

```
GENR TAR=(Y-YS)*(Y-YS)+(P-PS)*(P-PS) ;
```

The loss function to be minimized is assumed to be the sum of the target variable (`TAR` in this example) over the sample period in effect. The target can be any valid arithmetic expression.

OPTC options ;

The options are:

`TARGET=name`	`name` is the name of the target (`TAR` in the above example). Remember that the loss function is the sum of the target over the sample period.
`OUTSIDE`	Use this option when the control problem is for a period that is beyond the period for which there are data on the endogenous variables. If this option is used, the initial values for the current period for the solution procedure are set to the previous period's values if there are no current-period values. If there are current-period values, these values are used.

`FILECON=fn`	file `fn` contains the names of the control variables, one name per line, with a semicolon to end the file. The default for `fn` is `KEYBOARD`. If `fn` is `KEYBOARD`, the reading of the values is done from the input file (or from the keyboard if the program is being used interactively), again one value per line. The total number of control values to compute is equal to the number of control variables times the number of observations. The number of observations is determined by the `SMPL` command.
`MAXITERS=n`	`n` is the maximum number of iterations for the DFP algorithm. The default is 20.
`FILESTART=fn`	file `fn` contains the starting values, one value per line. The program reads the number of values as specified by the `FILECON` option. If `fn` is `KEYBOARD`, the reading of the values is done from the input file (or from the keyboard if the program is being used interactively), again one value per line. If the `FILESTART=fn` option is not specified, the starting values are taken to be the current values in memory. If there is more than one control variable, all the values for the first variable are read first, then all the values for the second variable, and so on. The number of values read for each control variable is the number of observations as specified by the `SMPL` command.
`FILESAVE=fn`	If this option is used, the final values are written to file `fn`. This file can be read using the `FILESTART=fn` option in future uses of the `MAX` command if desired.
`FILEHSAVE=fn`	If this option is used, the final value of the H matrix used by the DFP algorithm is written to file `fn`.
`FILEHSTART=fn`	If this option is used, the H matrix written to file `fn` in a previous use of the `MAX` command is read and used as the starting value of H. Using this option allows the DFP algorithm to begin where it left off. Otherwise, the initial value of the H matrix is the identity matrix.
`NORESET`	The default upon exit for `OPTC` is to set the predicted values back to the actual values (unless `OUTSIDE` is used). If you use the `NORESET` option, the predicted values are left unchanged. The variable values in memory are then the predicted values rather than the actual values.

`NORESETA`	Upon exit do not reset the first period observation (i.e., the first period of the `SMPL` in effect) of the first control variable. Reset everything else. [LA(413)=1]
`NORESETB`	Upon exit do not reset any of the observations of the first control variable, including any future values for RE models. Reset everything else. [LA(414)=1] WARNING: If you use the `NORESET`, `NORESETA`, or `NORESETB` option, the actual values are still in the `YY` matrix. (The predicted values are in the `Y` matrix.) If you want `YY` to contain the predicted values as well, use the `SETYYTOY` command.
`STOCHASTIC`	If you use this option the value of the objective function is computed using stochastic simulation, which means that the certainty equivalence assumption is not used. If you use this option, the following other options are relevant.
`LIMITEQS=n`	The default is to draw errors for all the equations in the model. If you use LIMITEQS=n, errors are drawn only for the first `n` equations. The residuals for any equations beyond `n` are left alone (they remain at whatever values are in memory).
`NTRIALS=n`	`n` is the number of trials per stochastic simulation. The default is 1.
`SDIAG`	S is taken to be diagonal for purposes of the draws.
`DRAWUHIST`	Draw historical errors, not errors from the distribution with covariance matrix S. This option must be used for the MC model. [LA(416)=1]
`FIRSTUHIST=p`	`p` is the first possible period to use for drawing the historical errors. Remember that the historical errors must be set ahead of time in a separate use of the `STOSIM` command using the `SETU` option. The default for `p` is the first period listed in the `SPACE` command. This option is not relevant unless the `DRAWUHIST` option is used. [LA(417)=p]

`LASTUHIST=p`	`p` is the last possible period to use for drawing the historical errors. Remember that the historical errors must be set ahead of time in a separate use of the `STOSIM` command using the `SETU` option. The default for `p` is the last period listed in the `SPACE` command. This option is not relevant unless the `DRAWUHIST` option is used. [LA(418)=p]
`MCMODEL`	This option pertains to the MC model. The error vectors are drawn for the four quarters of the year. The `DRAWUHIST` option must be in effect for the MC model. LA(419)=1
`FILEFIXED=fn`	The default is to draw error terms for all of the equations. If you want some of the error terms not to be drawn (to be fixed), this can be done using the `FILEFIXED=fn` option. The file `fn` should contain the equation numbers for the equations whose error terms are to be fixed. The format of this file is one number per line, with a 0 or a blank line to end. If `fn` is specified to be `KEYBOARD`, the numbers are read in from the input file (or from the keyboard if the program is being used interactively), again one number per line, with a 0 or a blank line to end. The error terms that are fixed are fixed at whatever values currently exist. They are not touched by the `STOSIM` command. This option does not work for RE models.
`SEED=v`	If you want to set the seed, this can be done using the `SEED=v` option. `v` is the new value of the seed. If this option is not used, the seed is whatever is in memory.

The order of the `KEYBOARD` reading is `FILEFIXED` first, then `FILECON`, and then `FILESTART`.

The DFP algorithm is used to minimize the loss function, which is the same algorithm used for the `MAX`, `NLOLS`, and `NL2SLS` commands. The options set by the command `SETUPDFP` are also relevant for the `OPTC` command. See Chapter 8 for the `SETUPDFP` command.

When the `ONESIDED` option is used in the `SETUPDFP` command and there is only one control variable specified in `FILECON=fn`, the program does the trick of taking numerical derivatives by starting from the last period first. For example, in computing the derivative of the objective function with respect to the last control value, one only needs to solve the model for the last period. [LA(405) and LA(406) are used internally for this work.]

13 CHAPTER 13: Models with Rational Expectations

13.1 Introduction

The program allows one to estimate and solve models in which expectations are assumed to be rational. (Models of this type will be referred to as "RE models.") The limited information estimators are Hansen's (1982) method of moments estimator and the Hayashi-Sims (1983) estimator. These estimators are options of the `2SLS` command in Chapter 5. The full information estimator is FIML. The solution method and FIML estimation method used in the program for RE models are discussed Section 2.12 in MM, which is based on Fair and Taylor (1983). The stochastic simulation method used in the program for RE models is also discussed in Section 2.12 in MM, which is based on Fair and Taylor (1990). The user should be familiar with Section 2.12 in MM before reading this chapter.

Leads are specified in the program as positive numbers. For example, `Y(2)` is variable `Y` lead two periods. In addition, if a contemporaneous endogenous variable is an expectations variable in an equation, this is indicated by putting 99 in parentheses. For example, `Y(99)` is the current value of variable `Y` entered as an expectations variable.

13.2 An Example

It will be convenient to begin with an example. The example is a modification of the model in Chapter 1 to make it a RE model. The revised model is:

$$\begin{aligned} \log C_t &= c_{11} + c_{21}\log C_{t-1} + c_{31\ t-1}E(\log Y_{t+2}) \qquad (4)\\ &\quad + c_{41}R_t + v_{1t} \\ v_{1t} &= \rho_{11}v_{1t-1} + u_{1t} \\ \log I_t &= c_{12} + c_{22}\log I_{t-1} + c_{32\ t-1}E(\log Y_{t+1}) \qquad (5)\\ &\quad + c_{42}R_{t-1} + u_{2t} \end{aligned}$$

$$Y_t = C_t + I_t + G_t \qquad \text{(I1)}$$

${}_{t-1}E$ is the expectations operator conditional on information through period $t-1$. Expectations appear in equations (1) and (2). The longest lead length is 2. Again, this model is not meant to be realistic; it is simply for illustration.

The specification of this model using the EQ, LHS, and IDENT commands is straightforward. The GENR commands in Step 3 in Chapter 1 remain the same. The remaining commands are:

```
EQ 1 LOGC CNST LOGC(-1) LOGY(2) R RHO=1;
LHS C=EXP(LOGC);
EQ 2 LOGI CNST LOGI(-1) LOGY(1) R;
LHS I=EXP(LOGI);
IDENT Y=C+I+G;
```

Note that there are positive values in parentheses in equations 1 and 2.

If, say, the expectations variable in equation 1 were ${}_{t-1}E(\log Y_t)$ instead of ${}_{t-1}E(\log Y_{t+2})$, then LOGY(2) would be replaced by LOGY(99). LOGY(99) differs from LOGY in that the former is an expectation based on information through period $t-1$, whereas the latter is the actual value.

13.3 Single Equation Estimation

As noted above, single equation estimation can be done using the HANSEN or HAYSIMS options of the 2SLS command. The commands to estimate the above model by Hansen's method are:

```
2SLS;
EST 1 HANSEN MA=1;
EST 2 HANSEN;
END;
```

The moving average order is one less than the lead length, and so the order is 1 for equation 1 and 0 for equation 2.

Remember when estimating RE models to allow "extra" observations at the end of the estimation period to handle the leads. The program does not automatically adjust for this. For example, if the data end in 2013.3 and the longest lead length is 4, the estimation period must end no later than 2001.1.

13.4 Solution

The FIML estimation command requires that the model be solved, and so solution must be discussed before FIML estimation. Assume that the model has been estimated by 2SLS using the HANSEN or HAYSIMS option. Having done this, all the solution commands in the program (like SOLVE and RMSE) will work for RE models. The only extra work needed for a RE model is to use the SETUPRE command before any solution is done. The command is:

SETUPRE options ;

The options are:

FILEENDO=fn	file fn contains the names of the expectational variables, one name per line, with a semicolon to end the list of names. If fn is specified to be KEYBOARD, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one name per line, with a semicolon to end. [LA(256)], NYE
H=n	n is the longest lead of any variable in the model. If the longest lead is, say, $t+2$, then n is 2. H is the same as h in Section 2.12 in MM. [LA(261)]
KFIRST=n	n is the shortest extra lead that is tried for the solution method. In terms of the notation in Section 2.12 in MM. KFIRST is the first value of k tried. [LA(259)]
KLAST=n	n is the longest extra lead that is tried for the solution method. In terms of the notation in Section 2.12 in MM. KLAST is the last value of k tried. Note: the limit of the number of Type III iterations tried is KLAST-KFIRST+1. If one wants only one Type III iteration, set KLAST=KFIRST. [LA(260)]
JFIRST=n	n is the shortest extra lag that is tried for the solution method. In terms of the notation in Section 2.12 in MM. JFIRST is the first value of j tried. If none of the equations of the model in which an expectations variable appears are assumed to have serially correlated errors, JFIRST should be zero. The default for JFIRST is zero. [LA(257)]

`JLAST=n`	`n` is the longest extra lag that is tried for the solution method. In terms of the notation in Section 2.12 in MM. `JFIRST` is the last value of j tried. If none of the equations of the model in which an expectations variable appears are assumed to have serially correlated errors, `JLAST` should be zero. The default for `JFIRST` is zero. Note: the limit of the number of Type IV iterations tried is `JLAST-JFIRST+1`. If one wants only one Type IV iteration, set `JLAST=JFIRST.` [LA(258)]
`FILEEXOG=fn`	file `fn` contains the names of the exogenous variables for which some assumption is made about expectations of future values other than that of perfect foresight. If the `FILEEXOG` option is not used, all future expectations of the exogenous variables that are needed for solution purposes are taken to be the actual values. The format of the file is one name per line, with a semicolon to end the list of names. If `fn` is specified to be `KEYBOARD`, the variable names are read in from the input file (or from the keyboard if the program is being used interactively), again one name per line, with a semicolon to end. See the discussion in Section 13.5 for more about this option.
`ALTRHO=n`	This option is only relevant when at least one of the equations in which there is an expectations variable has a serially correlated error. If this is true, then `ALTRHO=n` means to solve the model without going back more than one period. This option automatically sets `JFIRST=JLAST=1`. `n` is the maximum number of iterations to perform. [LA(198)]
`DAMP=v`	Damping factor for `ALTRHO=n option`. Default is no damping.
`FIRSTONLY`	This option can be used if the expectations of the exogenous variables are taken to be equal to the actual values. In this case, one can save computer time by only solving the model for the first period of interest. When `FIRSTONLY` is used, the model is solved only for the period of the `SMPL` command even though a longer period may be specified. Printing, however, is done for the longer period. [LA(164)]
`-FIRSTONLY`	Undo `FIRSTONLY.`

`MAXITERS2=n`	`n` is the maximum number of Type II iterations. The default is 100.
`TOL2=v`	`v` is the value of the tolerance criterion for the Type II iterations. The default is .002.
`TOL3=v`	`v` is the value of the tolerance criterion for the Type III iterations. The default is .003.
`TOL4=v`	`v` is the value of the tolerance criterion for the Type IV iterations. The default is .004.
`ABS2`	If this option is used, the tolerance criterion for the Type II iterations is in terms of absolute rather than percentage differences.
`-ABS2`	Undo `ABS2.`
`ABS3`	If this option is used, the tolerance criterion for the Type III iterations is in terms of absolute rather than percentage differences.
`-ABS3`	undo `ABS3`.
`ABS4`	If this option is used, the tolerance criterion for the Type IV iterations is in terms of absolute rather than percentage differences.
`-ABS4`	Undo `ABS4.`
`NOMSG2`	If this option is used, no message is printed after achieving Type II convergence.
`-NOMSG2`	Undo `NOMSG2.`
`NOMSG3`	If this option is used, no message is printed after achieving Type III convergence.
`-NOMSG3`	Undo `NOMSG3.`
`NOMSG4`	If this option is used, no message is printed after achieving Type IV convergence.
`-NOMSG4`	Undo `NOMSG4.`
`NOMSGLIMIT2`	If this option is used, no message is printed if Type II convergence is not achieved.
`-NOMSGLIMIT2`	Undo `NOMSGLIMIT2.`

`NOMSGLIMIT3`	If this option is used, no message is printed if Type III convergence is not achieved.
`-NOMSGLIMIT3`	Undo `NOMSGLIMIT3`.
`NOMSGLIMIT4`	If this option is used, no message is printed if Type IV convergence is not achieved.
`-NOMSGLIMIT4`	Undo `NOMSGLIMIT4`.

If both `FILEENDO` and `FILEEXOG` are set equal to `KEYBOARD`, reading for `FILEENDO` is done first.

The `SETUPRE` command that might be used for the above model is:

```
SETUPRE FILEENDO=KEYBOARD H=2 KFIRST=10 KLAST=10
ALTRHO=4 NOMSG2 NOMSG3 NOMSGLIMIT3 MAXITERS2=25
TOL2=.00002 TOL3=.00003 TOL4=.00004;
LOGY
;
```

The `SETUPSOLVE` command that might be used is:

```
SETUPSOLVE NOMSG NOMISS;
```

Some of the options for these commands can be chosen after some experimentation to see what is needed for convergence. If, for example, `KFIRST` is set equal to `KLAST`, which it is in the above `SETUPRE` command, this saves considerable computer time. A value of 10 is used here, although in practice one would have to experiment to see how large a value is needed. Note in this example that there is only one expectations variable, `LOGY`.

Once the `SETUPRE` command has been issued, the model can be solved over whatever sample period is chosen. Remember that extra observations are needed at the end of the solution period for the future predicted values. Upon exit from the `SETUPRE` command, the program indicates the last possible period that can be solved. This period depends in part on the choice for `KLAST`.

The `SETUPRE` command can be issued more than once in the program. With two exceptions, the options set by a previous use of the command remain in effect unless changed. The exceptions are the `FILEENDO=fn` and `FILEEXOG=fn` options. These have to be repeated each time the `SETUPRE` command is issued. If they are not used, the program will assume that there are no expectations variables and thus that the model is not a RE model.

13.5 Exogenous Variable Expectations

It is often assumed in models with rational expectations that agent's expectations of the current and future values of the exogenous variables are equal to the actual values. This is obviously not a realistic assumption in many cases, and the program allows it to be relaxed. The program allows the user to specify and estimate equations that agents are assumed to use to form their exogenous variable expectations. Consider, for example, exogenous variables `R` and `G` in the above model. Instead of assuming that agents know the current and future values of `R` and `G`, it could be assumed that agents forecast these values from autoregressive equations. The equations could be:

```
EQ 3 R CNST T R(-1) R(-2);
EQ 4 G CNST T G(-1) G(-2);
```

These equations could be estimated using:

```
OLS;
EST 3-4;
END;
```

The first step is thus to specify and estimate equations for the exogenous variables. These are the equations that agents are assumed to use to forecast the exogenous variable values. The second step is to use the `FILEEXOG=fn` option of the `FIML` command. The exogenous variables that are listed in file `fn` (or in the input file) must match the variables used on the left hand side of the exogenous variable equations. In the above example, the variables would be `R` and `G`. In addition, any nonlinear transformations of the listed variables that are in the model, such as log`G`, must be done using `GENR` commands and not `CREATE` commands. These nonlinear transformations must be part of the model, which the `GENR` commands are. For example, for log`G`, one would need the command `GENR LOGG=LOG(G)`. Finally, a `LHS` command may be needed for an exogenous variable equation. Say that equation 4 above was changed to be:

```
EQ 4 LOGG CNST T LOGG(-1) LOGG(-2) ;
```

where `LOGG` is the log of `G`. If `G` is used in the model, which it is in the above model, the command

```
LHS G=EXP(LOGG) ;
```

is needed in addition to equation 4. Note in this case that the relevant variable for the `FILEEXOG=fn` option is `LOGG`, not `G`, since `LOGG` is on the left hand side of equation 4.

The third and last step is to change the `MAXS=n` option of the `SPACE` command to have `n` be the total number of equations in the model, including the exogenous variable equations. All the exogenous variable equations must come after the regular stochastic equations, and there can be *no* gaps in the numbering between the last regular stochastic equation and the first exogenous variable equation.

The way the program works is that it uses the exogenous variable equations in solving for the endogenous variable expectations. Given the computed endogenous variable expectations, it then solves the model for the current period using the actual values of the exogenous variables (rather than the predicted values from the exogenous variable equations). In this case there is a difference between the expectation of an endogenous variable and its final predicted value.

13.6 FIML Estimation

Once the `SETUPRE` command has been used, the `FIML` command in Chapter 7 can be used with no changes. FIML estimation of RE models is, however, very expensive (in computer time) because the model has to be solved for each observation in the sample period for every evaluation of the likelihood function. Fortunately, there is a less expensive method that seems to work fairly well for most problems (see Section 2.12 in MM and Fair and Taylor (1990)). The program allows this less expensive method to be used. If it is used, additional options must be specified for the `FIML` command. These additional options are:

`RECHEAP`	If this option is specified, the less expensive method is used.
`REMAXITERS=n`	`n` is the maximum number of iterations for the less expensive method. The default is 1.

`REDER=n`	If this option is used, new derivatives are computed after every `n` iterations. The default is to compute new derivatives after every iteration. If you never want the derivatives computed (except at the beginning when the `FILEDERREAD` option below is not specified), set `n` equal to `REMAXITER+1` or to `0`.
`FILEDERWRITE=fn`	If this option is used, each time the derivatives are computed, they are written to file `fn`. Upon exit from the command, file `fn` will contain the latest set of derivatives.
`FILEDERREAD=fn`	If this option is used, the initial set of derivatives is read from file `fn` and new derivatives are not computed until after `REDER` iterations. file `fn` must have been created from a previous use of the `FILEDERWRITE` option.
`FILEYEWRITE=fn`	If this option is used, the solution values for the expectational variables are written to file `fn` after each iteration. Upon exit from the command, data set `fn` will contain the latest set of solution values.
`FILEYEREAD=fn`	If this option is used, the initial set of solution values is read from file `fn` and an initial solution to get the values is not performed. file `fn` must have been created from a pervious use of the `FILEYEWRITE` option.

No additional problems arise in the specification of the Jacobian for RE models. Since the expectations are assumed to be conditional on information through period $t-1$, the expectations are predetermined from the point of view of taking the derivatives. In other words, the expectations can be treated like lagged dependent variables and exogenous variables when the Jacobian is specified.

If the procedure discussed in Section 13.5 above is used for the exogenous variable expectations, namely to estimate equations for the exogenous variables, the coefficients of these equations should not be estimated by FIML. You should thus use the `FILEFIXED=fn` option of the `FIML` command to tell the program that you do not want to estimate any of these coefficients. For the above example, the `FILEFIXED` file would be:

```
3 1
3 2
3 3
3 4
4 1
4 2
4 3
4 4
0
```

Note that for non RE models, one could specify the above information as:

```
3 0
4 0
0
```

For RE models, however, this feature is not available. See the discussion of the `FILEFIXED` option of the `FIML` command in Chapter 7.

If the less expensive method is used for the `FIML` command, there are three setup options that may be useful. These options are set using the `SETUPFIMLRE` command:

SETUPFIMLRE options ;

The options are:

`DAMPCOEF=v`	`v` is the value of the damping factor for the successive coefficient estimates when the less expensive method is used. The default is 1.0, which is no damping. [SPA(17)]
`TOLCOEF=v`	`v` is the value of the tolerance criterion for the successive coefficient estimates when the less expensive method is used. The default is .002. [SPA(46)]
`STEPSIZEDER=v`	`v` is the perturbation size for the calculation of the numerical derivatives for the less expensive method. The default is .05. [SPA(47)]

If you are happy with the default values, the `SETUPFIMLRE` command does not have to be used.

13.7 Stochastic Simulation

The `STOSIM` command works for RE models with no modifications. The `SETUPRE` command must be used first to define the model, and the relevant covariance matrices must be in memory. The covariance matrix S can be estimated by the command `FIML MAXITERS=0;`. Upon return from the `FIML` command, the S matrix is in memory. The covariance matrix COV can also be estimated using the `FIML` command, where in this case the `RECHEAP` option should probably be used to save computer time (assuming it leads to a positive definite covariance matrix).

When doing stochastic simulation of RE models, one must be very clear on exactly what is being done. The program performs stochastic simulation in the manner discussed in Section 2.12.4 in MM and in Fair and Taylor (1990).

Finally, it is important to note that the `STOSIM` option of drawing from estimated residuals (instead of from estimated distributions) is *not* available for RE models. The reason is that the `CREATEU` command does not create the correct residuals for RE models. If you use the `CREATEU` command for a RE model, the values used for the expectations variables are the actual future values of the variables rather than computed expectations values, which is not right. (The `CREATEU` command uses the data in the `YY` matrix, which are the actual data.) In general, any command or procedure in the FP program that requires the `CREATEU` command is not available.

14 CHAPTER 14: Equations with ARMA Errors

The material in this chapter is not discussed in MM, and so the following discussion of the econometrics is somewhat more detailed than it would otherwise be. This chapter discusses the use of the FP commands to handle the case in which the error term in an equation follows an autoregressive moving average (ARMA) process. The focus in this guide up to this point has been on an autoregressive process only.

14.1 Estimation

In the example in Chapter 1 the error term v_{1t} in equation (1) was assumed to be:

$$v_{1t} = \rho_{11} v_{1t-1} + u_{1t}$$

where u_{1t} is *iid*. This is a first order autoregressive process. For sake of an example, assume instead that the error term is ARMA(2,2):

$$v_{1t} = \rho_{11} v_{1t-1} + \rho_{21} v_{1t-2} + u_{1t} + \gamma_{11} u_{1t-1} + \gamma_{21} u_{1t-2} \tag{1.1}$$

The following procedure can be used to obtain estimates of ρ_{11}, ρ_{21}, γ_{11}, γ_{21}, and the coefficients in the structural equation itself—equation (1) in Chapter 1. This procedure is due to Hannan and Rissanen (1982).[1]

The first step is to approximate (1.1) by an autoregressive process of order n:

$$v_{1t} = \rho_{11} v_{1t-1} + \ldots + \rho_{n1} v_{1t-n} + u_{1t} \tag{1.2}$$

where n is taken to be large enough to make it likely that the approximation is a good one. Equation (1) under assumption (1.2) can then be directly estimated in FP using the OLS or 2SLS command (under the `RHO=n` option). This completes the first step.

Assume that the equation has been estimated (for `RHO=n`) for the period $t = 1, \ldots, T$, and let $\hat{u}_{1t}$ denote the predicted value of u_{1t} from this estimation. Now substitute these predicted values for u_{1t-1} and u_{1t-2} into (1.1):

$$v_{1t} = \rho_{11} v_{1t-1} + \rho_{21} v_{1t-2} + u_{1t} + \gamma_{11} \hat{u}_{1t-1} + \gamma_{21} \hat{u}_{1t-2} \tag{1.3}$$

Given $\hat{u}_{1t-1}$ and $\hat{u}_{1t-2}$, the second step is to estimate equation (1) under the assumption (1.3), where the sample period now must begin with $t = 3$ since observation

[1] We are indebted to Clive Granger for pointing this procedure out to us. It is discussed in Granger and Newbold (1986), pp. 81-83.

1 is needed for $\hat{u}_{1t-1}$ and observation 2 is needed for $\hat{u}_{1t-2}$. This estimation yields estimates of ρ_{11}, ρ_{21}, γ_{11}, γ_{21}, and the coefficients in equation (1).

The estimation in Steps 1 and 2 can be either OLS or 2SLS. The structural equation being estimated—equation (1) in this example—can also be nonlinear in the structural coefficients. The generalization to ARMA processes other than 2,2 is also straightforward. The estimation in the second step above is nonlinear in coefficients even if the structural equation is not, but this can be easily handled by the FP program.

The following FP commands correspond to the above example using the OLS estimator and using n=8 (the lines with @ in front of them are comments):

```
SMPL 1952.1 2013.3;
@ LOGCR will be used to store the residuals.
CREATE LOGCR=LOGC;
@ The following equation is estimated in Step 1.
@ Note that equation (1) in Chapter 1 had RHO=1
@  rather than RHO=8.
EQ 1 LOGCR CNST LOGC(-1) LOGY R RHO=8;
@ The estimation period must begin in 1954.2 rather
@  than 1954.1 because RHO=8 and the use of LOGC(-1)
@  require 9 initial observations.
SMPL 1954.2 2013.3;
@ The SETLHSRESID option stores the residuals in the left
@  hand side variable (LOGCR) upon return from estimation.
@  These are the u residuals in (1.2) above
@  (not the v residuals).
OLS;
EST 1 SETLHSRESID;
END;
@ Now do Step 2.  Here we need the NLEQ and NLOLS commands.
EQ 1 LOGC;
NLEQ LOGC=COEF(1,1)+COEF(2,1)*LOGC(-1)+COEF(3,1)*LOGY
   +COEF(4,1)*R
 -COEF(5,1)*(COEF(1,1)+COEF(2,1)*LOGC(-2)+COEF(3,1)*LOGY(-1)
   +COEF(4,1)*R(-1))
 -COEF(6,1)*(COEF(1,1)+COEF(2,1)*LOGC(-3)+COEF(3,1)*LOGY(-2);
   +COEF(4,1)*R(-2))
 +COEF(5,1)*LOGC(-1)+COEF(6,1)*LOGC(-2)
 +COEF(7,1)*LOGCR(-1)+COEF(8,1)*LOGCR(-2);
@ The estimation period must begin in 1954.4 because of
@  the use of LOGCR(-1) and LOGCR(-2).
SMPL 1954.4 2013.3;
SETUPDFP TOLCOEF=.000001 STEPSIZE=.00001;
NLOLS EQUATION=1 NCOEF=8 MAXITERS=100 FILESTART=KEYBOARD;
0.
.97
.03
0.
0.
0.
0.
0.
@ Step 2 is now done.
```

The above comments explain some of the commands. You may need to review a few of the commands, especially the nonlinear commands, to be completely clear as to what is going on. One advantage of this procedure is that one gets t-statistics on all eight coefficients estimated by the `NLOLS` command, which includes the two autoregressive coefficients and the two moving average coefficients.

If the 2SLS estimator is to be used, then `2SLS` replaces `OLS` above and `NL2SLS` replaces `NLOLS`. Remember when using 2SLS that extra first stage regressors are likely to be needed in Step 1. For example, after the `EQ 1` command above one might add (assuming these variables are not part of the original `EQ 1 FSR` command):

```
MODEQ 1 FSR LOGC(-3) LOGC(-4) LOGC(-5) LOGC(-6) LOGC(-7)
 LOGC(-8) LOGC(-9)
 LOGY(-2) LOGY(-3) LOGY(-4) LOGY(-5) LOGY(-6) LOGY(-7)
 LOGY(-8) LOGY(-9)
 R(-2) R(-3) R(-4) R(-5) R(-6) R(-7) R(-8)
 R(-9);
```

After the `NLEQ` command, one should add:

```
MODEQ 1 FSR LOGCR(-1) LOGCR(-2);
```

because `LOGCR(-1)` and `LOGCR(-2)` are predetermined variables in the equation being estimated.

14.2 Solution

Once the equations with ARMA error terms and any other equations are estimated, solution is fairly straightforward. Consider, for example, the solution of the model in Chapter 1 in which equation (1) is estimated as above—under the ARMA(2,2) assumption. In this case the equation determining `LOGC` is the `NLEQ` command above. This command replaces the `EQ 1 LOGC CNST LOGC(-1) LOGY R RHO=1;` command listed in Step 3 in Chapter 1. (`C` is still determined by the command `C=EXP(LOGC);` as listed in Step 3 in Chapter 3.)

The only extra work that one needs to do for solution purposes when there are ARMA error terms is to make sure that the appropriate residuals are in memory. In the current example these residuals are $\hat{u}_{1t-1}$ and $\hat{u}_{1t-2}$, which are stored in `LOGCR`. After Step 2 estimation, the residuals must be generated and stored in `LOGCR`. (The residuals stored in `LOGCR` after Step 1 estimation are not the appropriate residuals because they are based on the approximating equation only.) The FP commands to generate the residuals are:

```
@ Use zero for the initial values of the residuals.
@  Remember that these are the u residuals, not
@  the v residuals.
SMPL 1954.2 1954.3;
CREATE LOGCR=0;
@ Given the initial values, generate the rest of the values.
SMPL 1954.4 2013.3;
CREATE LOGCR=LOGC-(COEF(1,1)+COEF(2,1)*LOGC(-1)
  +COEF(3,1)*LOGY+COEF(4,1)*R
 -COEF(5,1)*(COEF(1,1)+COEF(2,1)*LOGC(-2)+COEF(3,1)*LOGY(-1)
  +COEF(4,1)*R(-1))
 -COEF(6,1)*(COEF(1,1)+COEF(2,1)*LOGC(-3)+COEF(3,1)*LOGY(-2)
  +COEF(4,1)*R(-2))
 +COEF(5,1)*LOGC(-1)+COEF(6,1)*LOGC(-2)
 +COEF(7,1)*LOGCR(-1)+COEF(8,1)*LOGCR(-2));
@ Store the residuals in LOGCRS for possible
@  resetting purposes.
SMPL 1954.2 2013.3;
CREATE LOGCRS=LOGCR;
```

Assume that the model is to be solved for the 1960.1–1962.4 period and assume that the `RMSE` command is to be used for the solution command. If the solution is to be a static one, then the commands are simply:

```
SMPL 1960.1 1962.4;
RMSE;
```

A static simulation is based on the use of the actual values of `LOGCR(-1)` and `LOGCR(-2)` (i.e., the stored values) for each quarter forecasted; these values are taken as predetermined.

If the solution is dynamic, then the stored values of `LOGCR` should be used for 1959.3 and 1959.4, but zero values should be used for 1960.1 on. In other words, only the values before the first quarter of the prediction period should be taken as predetermined. The commands to run a dynamic simulation are:

```
SMPL 1960.1 1962.4;
@ Set the residuals to zero for the prediction period (but
@  not earlier).
CREATE LOGCR=0;
RMSE DYNAMIC;
@ Reset the residuals for possible use later.  (This command
@  is not needed if all the future solutions are going to be
@  dynamic and begin in 1960.1 or earlier.)
CREATE LOGCR=LOGCRS;
```

To summarize, the solution process will always use the `NLEQ` command to calculate `LOGC` and will use for `LOGCR(-1)` and `LOGCR(-2)` whatever values are in memory. For a static solution the values in memory are the residuals derived from the estimated equation in Step 2. For a dynamic solution, the values are zero except for the values before the beginning of the prediction period.

15 CHAPTER 15: Examples

A useful way of learning how to use the more advanced options of the program is to study examples of actual jobs. A number of examples are presented in this chapter. You should also go through the examples in Appendix A carefully.

15.1 Successive Reestimation and Stochastic Simulation

Successive Reestimation

Assume first that you want to estimate your model by 2SLS 123 times, where the first sample period is 1954.1–1982.4, the second is 1954.1–1983.1, and so on through 1954.1–2013.3. Assume that the model has 12 equations, numbered 1 through 12. Assume that you want to estimate the S and COV matrices 123 times as well. The commands to do this are:

```
SMPL 1954.1 1982.4;
2SLS NSETS=123 COV S FILECOEF=COEF.BIN FILES=S.BIN
   FILECOV=COV.BIN ;
EST 1-12;
END;
```

The program will write the coefficients to file `COEF.BIN`, the S matrices to file `S.BIN`, and the COV matrices to file `COV.BIN`. The files will contain 123 sets of estimates.

If you wanted the COV matrix to be estimated and written out in block diagonal form, then `COV` would be replaced by `COVBLKDIAG` and `FILECOV=COV.BIN` would be replaced by `FILECOVBLKDIAG=COV.BIN`. If you wanted S to be diagonal, `S` would be replaced by `SDIAG`. If you wanted the first observation to be increased by one for each new sample period, you would add `MOVEBEGOBS` as one of the options following 2SLS.

15.1.1 Reading Different Estimates

If you want to read, say, the fifth set of estimates of the coefficients, of S, and of COV, you would use the following commands:

```
READCOEF FILE=COEF.BIN;
READCOEF FILE=SAME;
READCOEF FILE=SAME;
READCOEF FILE=SAME;
READCOEF FILE=SAME;

READS FILE=S.BIN;
READS FILE=SAME;
READS FILE=SAME;
READS FILE=SAME;
READS FILE=SAME;

READCOV FILE=COV.BIN;
READCOV FILE=SAME;
READCOV FILE=SAME;
READCOV FILE=SAME;
READCOV FILE=SAME;
```

After these commands have been used, the fifth set of estimates is in memory.

15.1.2 Computing Outside Sample Root Mean Squared Errors

If after estimating the model 123 times, you want to compute, say, one through eight quarter ahead root mean squared errors for the 1983.1–2013.3 period, where all of the forecasts are outside sample (i.e., outside the estimation period) forecasts, this can be done using the `RMSEA` command:

```
SMPL 1983.1 2013.3;
RMSEA LENGTH=8 FILECOEF=COEF.BIN;
```

There will be 123 one quarter ahead forecasts for the root mean squared error calculations, 122 two quarter ahead forecasts, and so on. For each simulation the program will read a new set of coefficients from `COEF.BIN`.

15.1.3 Successive Stochastic Simulation

If you want to do successive eight quarter stochastic simulations to compute the "*d*" values discussed in Chapter 8 in Fair (1984), this can be done using the

`STOSIM` command. The command is:

```
SMPL 1983.1 2013.3;
STOSIM NTRIALS=100 LENGTH=8 DRAWCOEF
  FILESTOSIM=STOSIM.BIN NSETS=123 FILECOEF=COEF.BIN
  FILES=S.BIN FILECOV=COV.BIN ;
```

There will be 123 stochastic simulations of 100 trials each. All but the last seven will be eight quarters ahead. The seventh to last will be seven quarters ahead, the sixth to last will be six quarters ahead, and so on through the last one, which will be one quarter ahead. For each stochastic simulation the program will read a new set of coefficients from `COEF.BIN`, a new S matrix from `S.BIN`, and a new COV matrix from `COV.BIN`. All the results will be saved in file `STOSIM.BIN`. This file can then be read by the commands `DVALUES`, `TABLE8-1`, and `TABLE8-3`.

15.1.4 Printing Tables 8-1 and 8-2 and Computing the d Values

```
SMPL 1983.1 2013.3;
TABLE8-1 FILESTOSIM=STOSIM.BIN NSETS=123 LENGTH=8;
TABLE8-2 FILESTOSIM=STOSOM.BIN NSETS=123 LENGTH=8;
DVALUES FILESTOSIM=STOSIM.BIN NSETS=123 LENGTH=8;
```

15.2 Saving and Using Different Versions of a Model

If, say, a model is unchanged through 1992.4 and then a change is made, the commands to save the different versions are:

```
SMPL 1954.1 1982.4;
2SLS NSETS=40 COV S FILECOEF=COEF1.BIN FILES=S1.BIN
  FILECOV=COV1.BIN FILEMODEL=MODEL1.BIN ;
EST 1-12;
END;
@
@ Then make the changes to the model.  Then do:
@
SMPL 1954.1 1992.4;
```

```
2SLS NSETS=83 COV S FILECOEF=COEF2.BIN FILES=S2.BIN
  FILECOV=COV2.BIN FILEMODEL=MODEL2.BIN ;
EST 1-12;
END;
```

Then concatenate to create COEF.BIN, S.BIN, COV.BIN, and MODEL.BIN. Then add FILEMODEL=MODEL.BIN to STOSIM options.

15.3 Specifying a Vector Autoregressive Model

It is quite easy to specify a VAR model in the program. Assume that the variables are Y1 through Y4, that the lag length is 4, and that the constant term is named CNST. The specification is:

```
CREATE CNST=1.;

EQ 1 Y1 CNST Y1(-1) Y1(-2) Y1(-3) Y1(-4) Y2(-1) Y2(-2)
             Y2(-3) Y2(-4) Y3(-1) Y3(-3) Y3(-3) Y3(-4)
             Y4(-1) Y4(-2) Y4(-3) Y4(-4);
EQ 2 Y2 CNST Y1(-1) Y1(-2) Y1(-3) Y1(-4) Y2(-1) Y2(-2)
             Y2(-3) Y2(-4) Y3(-1) Y3(-3) Y3(-3) Y3(-4)
             Y4(-1) Y4(-2) Y4(-3) Y4(-4);
EQ 3 Y3 CNST Y1(-1) Y1(-2) Y1(-3) Y1(-4) Y2(-1) Y2(-2)
             Y2(-3) Y2(-4) Y3(-1) Y3(-3) Y3(-3) Y3(-4)
             Y4(-1) Y4(-2) Y4(-3) Y4(-4);
EQ 4 Y4 CNST Y1(-1) Y1(-2) Y1(-3) Y1(-4) Y2(-1) Y2(-2)
             Y2(-3) Y2(-4) Y3(-1) Y3(-3) Y3(-3) Y3(-4)
             Y4(-1) Y4(-2) Y4(-3) Y4(-4);
```

This is all that has to be done other than loading in values for Y1, Y2, Y3, and Y4. There are no identities and no LHS commands. The following commands estimate the model for the sample period in effect:

```
OLS;
EST 1-4;
END;
```

After this, you are ready to solve the model.

15.4 Sargent's Rational Expectations Model

Sargent's (1976) classical macroeconomic model is useful for illustrating some of the features of the program. The model has rational expectations and has the characteristic that the exogenous variable expectations are not equal to the actual exogenous variable values. In addition to Sargent's paper, you should read Sections 5.4 and 11.8 in Fair (1984) to see how the model is specified. This discussion will not be repeated here.

The following commands, which are in file `SAR.INP`, set up and estimate the model. This example requires file `SAR.DAT` to run. Both `SAR.INP` and `SAR.DAT` are included with the Fair-Parke material. The file `SAR.OUT` is also included, which is the output from running the example.

This example does not duplicate the results in Table 11-6 in Fair (1984) because the sample period is different (slightly longer). In this example the coefficient estimate of the expectations variable in equation 1 is of the wrong sign (contrary to the case in Table 11-6). This example is not meant to be realistic, but simply to show how some of the features of the FP program work.

A key variable in Sargent's model is the difference between the actual and expected values of `P`. If `M` and `POP` were known by the agents, then the expected value of `P` and the model's prediction of P would always be the same. These two differ only because the agents' expectations of `M` and `POP` differ from the actual values. The specification of equations 6 and 7 is thus quite important in this model. This example uses second order autoregressive equations for `M` and `POP`, but these equations can be easily changed to other specifications.

There is one feature of Sargent's model that the program cannot handle. The model needs to assume that the error term in equation 5 is uncorrelated with the error terms in the other four structural equations. This is a restriction on S in the program, which `FIML` cannot handle. What `FIML` can do, however, is to take the coefficients of equation 5 as fixed when the other equations are estimated, which is what the `FILEFIXED` option in the example does. This is not quite the same as taking `S` to be restricted, because even though the coefficients in equation 5 are not estimated, the calculation of the likelihood function uses the full 5×5 S matrix. The fifth row and column of S are never changed, but they are used in the calculations. The other coefficients will in general be slightly affected by this, but these effects are likely to be quite small. The program does, however, handle the exogenous variable equations 6 and 7 correctly. The sixth and seventh rows and columns of S are constrained to be zero except for the diagonal elements, which are taken to be one.

SAR.INP:

```
SARGENT’S MODEL
SPACE MAXCOV=55 MAXS=7 FIRSTPER=1952.1 LASTPER=1989.2
 MAXCOEF=30 MAXFSR=60;
@
@ Load the data, which are in data set SAR.DAT.
@
LOADDATA FILE=SAR.DAT;
@
@ Set up the model, first without the price expectations variable
@ in equations 1 and 2.  Equations 6 and 7 are the equations for
@ the exogenous variables.  Agents are assumed to use these
@ equations to form their expectations of M and @ POP.  Note that
@ MAXS in the SPACE command is 7: there are 5 regular stochastic
@ equations in the model and 2 exogenous variable equations.
@
SMPL 1952.1 1989.2;
CREATE CNST=1;
CREATE N=NF-UN+POP;
CREATE MACT=M;
CREATE POPACT=POP;
CREATE MMP=M-P;
EQ 1 UN CNST T UN(-1) UN(-2) UN(-3) UN(-4);
EQ 2 NF CNST T UN NF(-1) NF(-2) NF(-3) NF(-4);
EQ 3 Y CNST T N N(-1) N(-2) N(-3) N(-4) RHO=2;
EQ 4 MMP CNST T R R(-1) R(-2) R(-3) R(-4) R(-5) R(-6) R(-7)
 Y Y(-1) Y(-2) Y(-3) Y(-4) Y(-5) Y(-6) Y(-7) RHO=2;
LHS P=M-MMP;
EQ 5 R CNST T R(-1) R(-2) R(-3) R(-4);
IDENT N=NF-UN+POP;
EQ 6 M CNST T M(-1) M(-2);
EQ 7 POP CNST T POP(-1) POP(-2);
EQ 1 FSR CNST T UN(-1) UN(-2) UN(-3) UN(-4) NF(-1) NF(-2) NF(-3)
 NF(-4) N(-1) N(-2) N(-3) N(-4) N(-5) N(-6) R R(-1) R(-2) R(-3)
 R(-4) R(-5) R(-6) R(-7) R(-8) R(-9) Y(-1) Y(-2) Y(-3) Y(-4) Y(-5)
 Y(-6) Y(-7) Y(-8) Y(-9) POP M MMP(-1) MMP(-2);
EQ 2 FSR CNST T UN(-1) UN(-2) UN(-3) UN(-4) NF(-1) NF(-2) NF(-3)
 NF(-4) N(-1) N(-2) N(-3) N(-4) N(-5) N(-6) R R(-1) R(-2) R(-3)
 R(-4) R(-5) R(-6) R(-7) R(-8) R(-9) Y(-1) Y(-2) Y(-3) Y(-4) Y(-5)
 Y(-6) Y(-7) Y(-8) Y(-9) POP M MMP(-1) MMP(-2);
EQ 3 FSR CNST T UN(-1) UN(-2) UN(-3) UN(-4) NF(-1) NF(-2) NF(-3)
```

```
 NF(-4) N(-1) N(-2) N(-3) N(-4) N(-5) N(-6) R R(-1) R(-2) R(-3)
 R(-4) R(-5) R(-6) R(-7) R(-8) R(-9) Y(-1) Y(-2) Y(-3) Y(-4) Y(-5)
 Y(-6) Y(-7) Y(-8) Y(-9) POP M MMP(-1) MMP(-2);
EQ 4 FSR CNST T UN(-1) UN(-2) UN(-3) UN(-4) NF(-1) NF(-2) NF(-3)
 NF(-4) N(-1) N(-2) N(-3) N(-4) N(-5) N(-6) R R(-1) R(-2) R(-3)
 R(-4) R(-5) R(-6) R(-7) R(-8) R(-9) Y(-1) Y(-2) Y(-3) Y(-4) Y(-5)
 Y(-6) Y(-7) Y(-8) Y(-9) POP M MMP(-1) MMP(-2);
EQ 5 FSR CNST T UN(-1) UN(-2) UN(-3) UN(-4) NF(-1) NF(-2) NF(-3)
 NF(-4) N(-1) N(-2) N(-3) N(-4) N(-5) N(-6) R R(-1) R(-2) R(-3)
 R(-4) R(-5) R(-6) R(-7) R(-8) R(-9) Y(-1) Y(-2) Y(-3) Y(-4) Y(-5)
 Y(-6) Y(-7) Y(-8) Y(-9) POP M MMP(-1) MMP(-2);
@
@ Estimate the model without the expectations variables.
@
SMPL 1954.3 1989.2;
2SLS; EST 1-7; END;
@
@ Add the expectations variables.  Note that 99 is used for the
@ lead, because the price expectations variable enters
@ contemporaneously.
@
EQ 1 UN CNST T UN(-1) UN(-2) UN(-3) UN(-4) P P(99);
EQ 2 NF CNST T UN NF(-1) NF(-2) NF(-3) NF(-4) P P(99);
@
@ Set initial values for the coefficients of P.
@
COEF;
7 1 -.02
8 2 .02
0
@
@ In each equation the coefficient of P equals minus the
@ coefficient of P(99), so impose this constraint.
@
READCONSTR FILE=KEYBOARD;
1 1
2 1
0
CTC COEF(8,1)=-COEF(7,1);
CTC COEF(9,2)=-COEF(8,2);
PRINTMODEL;
WRITECOEF FILE=SARCOEF2.BIN;
@
@ Set up the Jacobian.
@
```

```
JACOB DERIV(1,1)=1;
JACOB DERIV(4,1)=-COEF(7,1);
JACOB DERIV(2,2)=1;
JACOB DERIV(4,2)=-COEF(8,2);
JACOB DERIV(1,2)=-COEF(4,2);
JACOB DERIV(3,3)=1;
JACOB DERIV(6,3)=-COEF(3,3);
JACOB DERIV(4,4)=-1;
JACOB DERIV(3,4)=-COEF(11,4);
JACOB DERIV(5,5)=1;
JACOB DERIV(6,6)=1;
JACOB DERIV(2,6)=-1;
JACOB DERIV(1,6)=1;
@
@ Set up the model for solution purposes.  Note that the
@ FILEEXOG option is used for M and POP.
@
SETUPSOLVE NOMSG NOMISS MAXCHECK=4 TOLALL=.00001;
SETUPRE FILEENDO=KEYBOARD H=0 FILEEXOG=KEYBOARD NOMSG2
 TOL2=.00002;
P
;
M
POP
;
SETUPFIMLRE STEPSIZEDER=.001;
@
@ Estimate the model by FIML.  Note that the FILEFIXED option
@ is used to tell the program not to estimate equations 6 and 7.
@ In addition, the coefficients of equation 5 are not
@ estimated---see the discussion in Sections 5.4 in Fair (1984)
@ for why the equation is not estimated.  Some of the less
@ important coefficients in equations 1-4 are also not
@ estimated by FIML.  This is done to save on computer time.
@
FIML FILEFIXED=KEYBOARD MAXITERS=20 ;
1 5
1 6
2 5
2 6
2 7
3 4
3 5
3 6
3 7
```

```
3 9
4 4
4 5
4 6
4 7
4 8
4 9
4 10
4 12
4 13
4 14
4 15
4 16
4 17
4 18
5 1
5 2
5 3
5 4
5 5
5 6
6 1
6 2
6 3
6 4
7 1
7 2
7 3
7 4
0
@
@ Estimate the COV matrix.
@
FIML FILEFIXED=KEYBOARD MAXITERS=0 COV;
1 5
1 6
2 5
2 6
2 7
3 4
3 5
3 6
3 7
3 9
4 4
```

```
4 5
4 6
4 7
4 8
4 9
4 10
4 12
4 13
4 14
4 15
4 16
4 17
4 18
5 1
5 2
5 3
5 4
5 5
5 6
6 1
6 2
6 3
6 4
7 1
7 2
7 3
7 4
0
PRINTCOV DIAG;
@
@ Check out the CHEAP option
@
FIML FILEFIXED=KEYBOARD MAXITERS=20
 RECHEAP REMAXITERS=5 FILEDERWRITE=SARDER.BIN REDER=1 ;
1 5
1 6
2 5
2 6
2 7
3 4
3 5
3 6
3 7
3 9
4 4
```

```
4 5
4 6
4 7
4 8
4 9
4 10
4 12
4 13
4 14
4 15
4 16
4 17
4 18
5 1
5 2
5 3
5 4
5 5
5 6
6 1
6 2
6 3
6 4
7 1
7 2
7 3
7 4
0
@
@ Check the final value of the likelihood function obtained
@ using the less expensive method.
@
FIML MAXITERS=0 ;
@
@ Estimate the COV matrix using the CHEAP option: (this did
@  give the correct t-statistics, so in this case the
@  CHEAP option did not work)
@
FIML FILEFIXED=KEYBOARD MAXITERS=0 COV
 RECHEAP REMAXITERS=0 REDER=0 FILEDERREAD=SARDER.BIN ;
1 5
1 6
2 5
2 6
2 7
```

```
3 4
3 5
3 6
3 7
3 9
4 4
4 5
4 6
4 7
4 8
4 9
4 10
4 12
4 13
4 14
4 15
4 16
4 17
4 18
5 1
5 2
5 3
5 4
5 5
5 6
6 1
6 2
6 3
6 4
7 1
7 2
7 3
7 4
0
PRINTCOV DIAG;
@
@ Exit the job.
@
QUIT;
```

16 CHAPTER 16: Bootstrapping

Bootstrapping is discussed in Sections 2.7, 3.9, and 3.10 in MM. The FP program was used for the results in these sections. The commands that were used for this work have already been presented earlier in this guide except for four new ones in this chapter. Setting up the commands is tedious, and there are many possibilities. The four additional commands are discussed in this chapter, but examples of bootstrapping are not presented. If you are interested in bootstrapping, email *ray.fair@yale.edu* and I will send you the relevant set of commands for your problem. You will need to read Sections 2.7, 3.9, and 3.10 in MM first.

The four new commands are the following.

16.1 The DIST commands

DISTCOEF options ;

The options are:

`NTRIALS=n` `n` is the number of trials.

`BIASCORRECTION`

This option computes the bias-correction vector and bias corrects the coefficients. If you use the `PRINTMODEL` command both before and after the `DISTCOEF` command when the `BIASCORRECTION` option is used, you can see how the coefficients were changed.

`BOOTSTRAPINNER`

This option is used when computing coverage accuracy. In this case `NTRIALS` is the number of inner trials. See the above example.

`BOOTSTRAPOUTER`

This option is used when computing coverage accuracy. This option does not use the `NTRIALS` option. See the above example.

`COVERAGE`	This option is used when computing coverage accuracy. It requires the `NTRIALSOUTER` option. See the above example.
`NTRIALSOUTER=n`	`n` is the number of "outer" trials, i.e., the number of repetitions.

DISTAP options ;

The options are:

`NTRIALS=n`	`n` is the number of trials.
`NAP=n`	`n` is the number of equations tested.

DISTPRED options ;

The options are:

`NTRIALS=n`	`n` is the number of trials.
`NP=n`	`n` is the number of variables printed.
`NPER=n`	`n` is the length of the prediction period.

DISTMULT options ;

The options are:

`NTRIALS=n`	`n` is the number of trials.

`NP=n`	`n` is the number of variables printed.
`NPER=n`	`n` is the length of the prediction period.

A APPENDIX A: Examples for Testing

A.1 IS Model

This example uses many of the commands and options in the FP program. You should use it to help learn the program and to test that the program has been installed correctly on your computer. The model used is simply the IS model in Chapter 1, which consists of two stochastic equations and one identity. It is not meant to be realistic. To run this example on your computer you need the following files:

`IS.INP`	Contains the input commands.
`IS.DAT`	Contains the data on the variables.
`IS.VAR`	Contains the names of the variables for which output is to be printed.
`ISFSR3.INP`	Contains the names of the first stage regressors for 3SLS.

The main file to study is `IS.INP`, which is listed below and which has many comments in it. (A comment is preceeded by a @.) To run the example, you type `FP > OUT`, hit return, and then type `INPUT FILE=IS.INP;`. When the job has completed, the file `OUT` should be the same (aside from rounding error) as the file `IS.OUT`, which is included with this example.

IS.INP:

```
IS MODEL FOR FP PROGRAM  NOVEMBER 11, 2013
SPACE MAXVAR=200 MAXS=2 MAXCOEF=20  MAXFSR=30 FIRSTPER=1952.1
 LASTPER=2013.3 MAXCOV=9;
@
@ For solution, do at least 2 iterations:
@
SETUPSOLVE MINITERS=2 ;
@
@ No missing values in IS MODEL:
@
SETUPSOLVE NOMISS;
@
@ For estimation divide by T rather than T-K:
@
```

```
SETUPEST DIVIDET;
@
@ Use (6.13) and (6.14), p. 212, in Fair (1984) to compute the
@ 2SLS estimates:
@
SETUPEST ALT2SLS;
@
@ Set up some options for the DFP algorithm:
@
SETUPDFP PRINTOBJ PRINTVALUES;
@
@ Load the data:
@
SMPL 1952.1 2013.3;
LOADDATA FILE=IS.DAT;
@
@ Create the constant term:
@
CREATE CNST=1;
@
@ Generate the needed variables:
@
GENR LOGC=LOG(C);
GENR LOGI=LOG(I);
GENR LOGY=LOG(Y);
@
@ Create variables needed only for FSRs:
@
CREATE LOGG=LOG(G);
@
@ Create the time trend:
@
CAPITAL I=CNST K=T BENCHPER=1952.1 BENCHVAL=1. DEPRATE=0.;
@
@ Specify the equations and the LHS commands:
@
EQ 1 LOGC CNST LOGC(-1) LOGY R RHO=1;
LHS C=EXP(LOGC);
EQ 2 LOGI CNST LOGI(-1) LOGY R(-1);
LHS I=EXP(LOGI);
IDENT Y=C+I+G;
@
@ Specify the FSRs for each equation:
@
EQ 1 FSR CNST LOGC(-1) LOGC(-2) LOGY(-1) R(-1) LOGI(-1) LOGG;
EQ 2 FSR CNST LOGC(-1) LOGC(-2) LOGY(-1) R(-1) LOGI(-1) LOGG;
@
@ Check the IDENT and LHS commands:
@
SMPL 2013.3 2013.3;
CHECK IDENT;
CHECK LHS;
@
@ Estimate the coefficients, S, and COV and save in files:
@
SMPL 1954.1 2013.3;
2SLS S COV FILECOEF=ISCOEF2.BIN FILES=ISS2.BIN
 FILECOV=ISCOV2.BIN;
EST 1 2;
END;
@
@ Print the model for checking purposes (PRINTMODEL can also
@ be used):
@
PRINTCOV DIAG;
@
@ Perform an AP test of equationS 1 and 2:
@
SMPL 1954.1 2013.3;
TESTSTAB2 1  1970.1 1979.4;
TESTSTAB2 2  1970.1 1979.4;
PRINTMODEL;
@
@ Perform an end-of-sample stability test:
@
SMPL 1952.1 2013.3;
CREATE ZERO=0;
SMPL 1954.1 2013.3;
TESTEND2 1 1995.1 NAMEMISS=ZERO;
TESTEND2 2 1995.1 NAMEMISS=ZERO;
@
@ Test equations 1 and 2 by adding lagged values.  Add FSRs first
@  to account for extra lags:
@
SETUPTEST PRINTTEST;
SMPL 1954.1 2013.3;
MODEQ 1 FSR LOGC(-3) LOGY(-2) R(-2);
TEST2 1 LOGC(-2) LOGY(-1) R(-1);
MODEQ 2 FSR LOGI(-2) R(-2);
TEST2 2 LOGI(-2) LOGY(-1) R(-2);
```

```
MODEQ 1 FSR - LOGC(-3) LOGY(-2) R(-2);
MODEQ 2 FSR - LOGI(-2) R(-2);
PRINTMODEL;
@
@ Test for perfect tracking solution:
@
SMPL 2011.4 2013.3;
CREATEU;
RMSE DYNAMIC FILEVAR=IS.VAR;
ZEROU;
@
@ Solve the model and compute RMSEs for 2000.4-2013.3:
@
SMPL 2011.4 2013.3;
RMSE DYNAMIC FILEVAR=IS.VAR;
RMSEA LENGTH=4 FILEVAR=IS.VAR;
@
@ Do a multiplier experiment using CREATEU:
@
SMPL 2011.4 2013.3;
CREATEU;
CHANGEVAR;
G ADDSAMEABS
50.
;
RMSE DYNAMIC FILEVAR=IS.VAR;
ZEROU;
CHANGEVAR;
G ADDSAMEABS
-50.
;
@
@ Do a multiplier experiment using MULT:
@
SMPL 2011.4 2013.3;
CREATEU;
XMULT G=G+50;
MULT NOBASERUN FILEVAR=IS.VAR;
ZEROU;
@
@ Estimate the model using 2SLAD:
@
SETUPEST -ALT2SLS;
SMPL 1954.1 2013.3;
2SLS; EST 1 2 ; END;
```

```
2SLAD ;
EST 1 2;
END;
SETUPEST ALT2SLS;
@
@ Read 2SLS coefficient estimates back in:
@
READCOEF FILE=ISCOEF2.BIN;
@
@ 3SLS estimation:
@
SMPL 1954.1 2013.3;
PRINTS;
3SLS FILEFSR=ISFSR3.DAT FILECOEF=ISCOEF3.BIN COV MAXITERS=30;
PRINTS;
PRINTCOV DIAG;
@
@ Get summary statistics for 3SLS:
@
SETUPEST PRINTONLY;
2SLS;
EST 1 2;
END;
SETUPEST -PRINTONLY;
@
@ Do 3SLS with S diagonal.  Should get 2SLS estimates.
@
READCOEF FILE=ISCOEF2.BIN;
READS FILE=ISS2.BIN;
PRINTS;
3SLS FILEFSR=ISFSR3.DAT SDIAG COV MAXITERS=30;
PRINTS;
PRINTCOV DIAG;
READCOEF FILE=ISCOEF2.BIN;
READS FILE=ISS2.BIN;
@
@ FIML estimation:
@
SMPL 1954.1 2013.3;
JACOB DERIV(1,1)= 1/C;
JACOB DERIV(3,1)= -COEF(3,1)/Y;
JACOB DERIV(2,2)= 1/I;
JACOB DERIV(3,2)= -COEF(3,2)/Y;
JACOB DERIV(3,3)= 1;
JACOB DERIV(1,3)= -1;
JACOB DERIV(2,3)= -1;
FIML FILECOEF=ESCOEF4.BIN COV MAXITERS=30;
PRINTCOV DIAG;
@
@ Test DFP option for FIML:
@
READCOEF FILE=ISCOEF2.BIN;
FIML DFP MAXITERS=30;
READCOEF FILE=ISCOEF2.BIN;
READS FILE=ISS2.BIN;
READCOV FILE=ISCOV2.BIN;
@
@ The rest of this file consists of advanced commands that you may
@ or may not want to examine.  If you do, comment out QUIT; below.
@
@QUIT;
@
@ Create historical errors to draw; center residuals at zero:
@
SMPL 1954.1 2013.3;
CREATEU ZEROMEAN;
@
@ Set residuals in STOSIM:
@
STOSIM SETU;
ZEROU;
@
@ Set the seed:
@
SEED VALUE=56789;
@
@ Don't print iteration numbers when solving:
@
SETUPSOLVE NOMSG;
@
@ Begin OPTC STOCHASTIC work
@
SETUPDFP PRINTOBJ STEPSIZE=.001;
SMPL 1952.1 2013.3;
CREATE YACT=Y;
GENR TAR=((Y-YACT)/YACT)*((Y-YACT)/YACT);
SMPL 1994.1 1998.4;
CREATEU;
@
@ Do non stochastic first (should go nowhere):
```

```
@
OPTC TARGET=TAR FILECON=KEYBOARD MAXITERS=4;
G
;
@
@ Now do stochastic: (This is commented out for the
@ example because it is costly in computer time.)
@
@OPTC TARGET=TAR FILECON=KEYBOARD MAXITERS=4
@ STOCHASTIC NTRIALS=1000 DRAWUHIST FIRSTUHIST=1954.1
@ LASTUHIST=2013.3;
@G
@;
@
@ End OPTC STOCHASTIC work
@
@ Do stochastic simulation.
@
@ Draw error terms only; draw from estimated residuals.
@
SMPL 2011.4 2013.3;
STOSIM NTRIALS=1000 FILEVAR=IS.VAR FILEMEDIAN=IS.VAR
 DRAWUHIST FIRSTUHIST=1954.1 LASTUHIST=2013.3;
@
@ Draw error terms only; draw from estimated distribution.
@
SMPL 2011.4 2013.3;
STOSIM NTRIALS=1000 FILEVAR=IS.VAR FILEMEDIAN=IS.VAR;
@
@ Draw error terms and coefficients; draw from distributions.
@
STOSIM NTRIALS=1000 FILEVAR=IS.VAR FILEMEDIAN=IS.VAR DRAWCOEF;
@
@ Estimate COV again because it has been destroyed by STOSIM.
@
QUIT;
SMPL 1954.1 2013.3;
2SLS COV; END;
@
@ Prepare to draw exogenous variable values (for G).  First
@ estimate autoregressive equations for G and save results.
@
AUTOREG ORDER=4 TIME FILESE=ISSEEXOG.TMP;
C
T
```

```
G
;
@
@ Draw error terms, coefficients, and exogenous variables.
@
SMPL 2011.4 2013.3;
STOSIM NTRIALS=1000 FILEVAR=IS.VAR FILEMEDIAN=IS.VAR DRAWCOEF
 DRAWEXOG FILEEXOG=ISSEEXOG.TMP;
@
@ Now do successive reestimation and stochastic simulation.
@ Estimate model 123 times and save results for command STOSIM.
@
SMPL 1954.1 1982.4;
2SLS NSETS=123 COV S FILECOEF=ISCOEF.BIN FILECOV=ISCOV.BIN
 FILES=ISS.BIN PRINTLESS;
EST 1 2;
END;
@
@ First do outside sample RMSEs.
@
SMPL 1983.1 2013.3;
RMSEA LENGTH=8 FILECOEF=ISCOEF.BIN FILEVAR=IS.VAR;
@
@ Now do 123 stochastic simulations and save results for other
@ commands.
@
STOSIM NTRIALS=1000 LENGTH=8 DRAWCOEF FILESTOSIM=ISSTOSIM.BIN
 NSETS=123 FILECOEF=ISCOEF.BIN FILECOV=ISCOV.BIN FILES=ISS.BIN
 FILEVAR=IS.VAR;
@
@ Analyze the stochastic simulation results.  Remember that
@ 123 stochastic simulations were made (NSETS=123 above), and so
@ NSETS must equal 123 for the following commands.  Also, the
@ sample period must remain the same, LENGTH must be 8, and
@ FILESTOSIM must equal ISSTOSIM.BIN.
@
TABLE8-1 FILEVAR=IS.VAR LENGTH=8 NSETS=123 FILESTOSIM=ISSTOSIM.BIN;
TABLE8-3 FILEVAR=IS.VAR LENGTH=8 NSETS=123 FILESTOSIM=ISSTOSIM.BIN;
DVALUES FILEVAR=IS.VAR LENGTH=8 NSETS=123 FILESTOSIM=ISSTOSIM.BIN
 OLSLENGTH=2 TEMPVAR=ZERO FILEOLS=KEYBOARD;
C
T
;
;
@
```

```
@ Get back basic estimates:
@
SMPL 1954.1 2013.3;
2SLS COV S; EST 1 2; END;
@
@ Do MULT and STOMULT:
@
SMPL 2011.4 2013.3;
CREATEU;
MULT FILEVAR=IS.VAR;
STOMULT NTRIALS=1000 FILEVAR=IS.VAR;
@
@ Quit the job:
@
QUIT;
```

ISFSR3.DAT:

```
CNST
LOGC -1
LOGC -2
LOGY -1
R -1
LOGI -1
LOGG
;
```

IS.VAR:

```
Y
C
I
;
```

A.2 ISRE Model

The following example estimates and solves the model with rational expectations in Chapter 13. You should study this example carefully if you are going to be working with a model with rational expectations. To run this example you need the following files:

```
IS.DAT
ISRE.INP
IS.VAR
```

To run the example, you type `FP > OUT`, hit return, and then type `INPUT FILE=ISRE.INP;`. When the job has completed, the file `OUT` should be the same (aside from rounding error) as the file `ISRE.OUT`, which is included with this example.

ISRE.INP:

```
ISRE MODEL FOR FP PROGRAM  NOVEMBER 11, 2013
SPACE MAXVAR=200 MAXS=2 MAXCOEF=20  MAXFSR=30 FIRSTPER=1952.1
 LASTPER=2013.3 MAXCOV=9;
@
@ For solution, do at least 2 iterations:
@
SETUPSOLVE MINITERS=2 ;
@
@ No missing values in IS MODEL:
@
SETUPSOLVE NOMISS;
@
@ Do not print iteration numbers:
@
SETUPSOLVE NOMSG;
@
@ For estimation divide by T rather than T-K:
@
SETUPEST DIVIDET;
@
@ Use (6.13) and (6.14), p. 212, in Fair (1984) to compute the
@ 2SLS estimates:
@
SETUPEST ALT2SLS;
@
@ Set up some options for the DFP algorithm:
@
SETUPDFP PRINTOBJ PRINTVALUES;
```

```
@
@ Load the data:
@
SMPL 1952.1 2013.3;
LOADDATA FILE=IS.DAT;
@
@ Create the constant term:
@
CREATE CNST=1;
@
@ Generate the needed variables:
@
GENR LOGC=LOG(C);
GENR LOGI=LOG(I);
GENR LOGY=LOG(Y);
@
@ Create variables needed only for FSRs:
@
CREATE LOGG=LOG(G);
@
@ Specify the equations and the LHS commands:
@
EQ 1 LOGC CNST LOGC(-1) LOGY(2) R RHO=1;
LHS C=EXP(LOGC);
EQ 2 LOGI CNST LOGI(-1) LOGY(1) R(-1);
LHS I=EXP(LOGI);
IDENT Y=C+I+G;
@
@ Specify the FSRs for each equation:
@
EQ 1 FSR CNST LOGC(-1) LOGC(-2) LOGY(-1) R(-1) LOGI(-1) LOGG
;
@ LOGY LOGY(1) LOGY(2);
EQ 2 FSR CNST LOGC(-1) LOGC(-2) LOGY(-1) R(-1) LOGI(-1) LOGG
;
@ LOGY LOGY(1) LOGY(2);
@
@ Check the IDENT and LHS commands:
@
SMPL 2013.1 2013.1;
CHECK IDENT;
CHECK LHS;
@
@ Estimate the coefficients using Hansen's method and save them:
@
```

```
SMPL 1954.1 1998.4;
2SLS FILECOEF=ISRECOE2.BIN ;
EST 1 HANSEN MA=1;
EST 2 HANSEN;
END;
@
@ Print the model for checking purposes:
@
PRINTMODEL;
@
@ Set up the Jacobian for FIML:
@
JACOB DERIV(1,1)= 1/C;
JACOB DERIV(2,2)= 1/I;
JACOB DERIV(3,3)= 1;
JACOB DERIV(1,3)= -1;
JACOB DERIV(2,3)= -1;
@
@ Set up RE options.  Note that KFIRST=KLAST, which save time.
@
@ ALTRHO=4
SETUPRE FILEENDO=KEYBOARD H=2 KFIRST=10 KLAST=10
 NOMSG2 NOMSG3 NOMSGLIMIT3 MAXITERS2=25 TOL2=.00002 TOL3=.00003
 TOL4=.00004
 JFIRST=6 JLAST=6
;
LOGY
;
SETUPFIMLRE STEPSIZEDER=.001;
SETUPFIMLCOV MINCHANGE=.00000000001;
@
@ Estimate the model by FIML using the derivative (cheap) method:
@
FIML RECHEAP REMAXITERS=2 FILEDERWRITE=ISREDER.BIN
 FILEYEWRITE=ISREYE.BIN MAXITERS=30 REDER=0;
@
@ Estimate the COV matrix:
@
FIML RECHEAP FILEDERREAD=ISREDER.BIN MAXITERS=0 COV ;
@
@ Print COV, S, and the model:
@
PRINTCOV;
PRINTS;
PRINTCOV DIAG;
```

```
@
@ Estimate the model by FIML regular way:
@
FIML MAXITERS=30;
@
@ Estimate the COV matrix:
@
FIML MAXITERS=0 COV ;
@
@ Print COV, S, and the model:
@
PRINTCOV;
PRINTS;
PRINTCOV DIAG;
@
@ Solve the model:
@
SMPL 1998.1 1998.4;
RMSE DYNAMIC FILEVAR=IS.VAR;
@
@ Do stochastic simulation:
@
SMPL 1998.1 1998.4;
STOSIM NTRIALS=100 FILEVAR=IS.VAR FILEMEDIAN=KEYBOARD DRAWCOEF;
C
I
Y
;
STOSIM NTRIALS=100 FILEVAR=IS.VAR FILEMEDIAN=KEYBOARD ;
C
I
Y
;
@
@ Quit the job:
@
QUIT;
```

B APPENDIX B: Programming Notes

LA and SPA Values

All the data and information in the program are stored in three vectors: LA, SPA, and DPA. The LA elements are INTEGER*4; the SPA elements are REAL*8; and the DPA elements are CHARACTER*8. The first 500 elements of the LA vector and the first 100 elements of the SPA vector are reserved for programming. The following is a list of these elements. When a default value is given for a user supplied element, this is not always the value that the element actually equals. The value that the element actually equals is usually zero. For the cases in which a default value listed below is not zero but the value of the element is zero, the program knows to use the default value at the time of execution. The DPA vector contains only the names of the variables, and it is not of general interest. There is no `SETDPA` option. Items that have RAYFAIR in parentheses are not of general use.

LA VECTOR:

1. dimension of the LA vector in `MAIN`. This can be changed before `MAIN` is compiled.
2. number of elements currently used in the LA vector.
3. largest element number of the LA vector that can be user supplied (number is 500).
4. `NY`
5. counter for subroutine `RDS` (unit 15).
6. `NS`
7. 1 = space for the `S` matrix is not allocated.
8. `NZ1`
9. `NQ1`
10. `NC`
11. `ICR`

12. counter for subroutine `WRS` (unit 16).

13. counter for subroutine `RDCOV` (unit 17).

14. 1 = no space for `YY` is allocated.

15. dimension of the `DPA` vector in `MAIN`. This can be changed before `MAIN` is compiled. It should be roughly twice `NY`.

16. number of elements currently used in the `DPA` vector.

17. `MAXVAR` (`NYMAX` in `MAINA`)

18. `MAXCMD`

19. `MAX1`

20. `NYOLD` (for old way)

21. `NXOLD` (for old way)

22. counter for subroutine `WRCOV` (unit 18).

23. 1 = old way of doing `EST` subcommand

24. 1 = assume no missing data in `GMPRD`

25. 1 = assume no missing data for solution purposes

26. `NC2` [=`(NC*(NC+1))/2`]

27. `NZ2` [=`NZ1*2+3`]

28. `NQ2` [=`NQ1*2+2`]

29. counter for subroutine `RDC` (unit 19).

30. counter for subroutine `WRC` (unit 20).

31. 1 = if `WALD1` specified (unit 25 written to)
    2 = if `WALD2` specified (unit 25 read)

32. 1 = use `READJOB` rather than `INPUT` to begin.

33. first period (for example, 1952, 1952.1, or 1952.01).

34. 0 for quarterly, 1 for yearly, 2 for monthly.

35. noba [=nend-nbeg+1]

36. used by `FIML0` and `FUNC4`. [1 if nj = 0 or 1]

37. iunit (the current unit reading from)

38. ifair (RAYFAIR)

39. igov (RAYFAIR)

40. itype (RAYFAIR)

41. ifairf (RAYFAIR)

42. nfy (RAYFAIR)

43. ify (RAYFAIR)

44. kfy (RAYFAIR)

45. used for `FILEFIXED=fn` option in `FIML`.

46. equals LA(307) on entry (`MAXSTK`).

47. input unit if command `INPUT` has been called.

48. used by `FIML`

49. `NENDE`

50. used by subroutine `DRUN`

51. `NBEG`

52. `NEND`

53. used by subroutine `TSLS0`

54. `NENDQ`

55. variable number of the constant term.

56. current open slot is `ISTACK`

57. current open slot in `LSTACK`

58. dimension of Jacobian matrix

59. 1 = use specified damping factors for specified variables.

60. 1 = use specified tolerance criterion for specified variables.

61. 1 = use absolute criterion for specified variables.

62. idw for `FIML` and `RECHEAP`.

63. idr for `FIML` and `RECHEAP`.

64. iyw for `FIML` and `RECHEAP`.

65. iyr for `FIML` and `RECHEAP`.

66. 1 = plot residuals to use 133 character spaces rather than 80 for `OLS`, `2SLS`, `LAD`, `2SLAD`, and `PLOTV`.

67. write in `RMSEA`.

68. 1 = use starting values for `RHOs`, which are stored in `SPA(81),...,` `SPA(90).`

69. 1 = set `SPA(81),...,` to `RHOs` when ??.

70. used by command `DEVMEAN`.

71. 1 = tolerance criterion for the Gauss-Seidel technique is in terms of absolute rather than percentage differences.

72. 1 = tolerance criterion for the Type III iterations is in terms of absolute rather than percentage differences.

73. 1 = tolerance criterion for the Type IV iterations is in terms of absolute rather than percentage differences.

74. limit for the number of Gauss-Seidel iterations (default = 100).

75. limit for the number of Type II iterations (default = 100).

76. 1 = the number of Gauss-Seidel iterations is not printed after convergence has been achieved.

77. 1 = the predicted values of the endogenous variables are printed after each Gauss-Seidel iteration. This option results in considerable printing, and it should only be used for debugging purposes.
78. 1 = tolerance criterion for the Type II iterations is in terms of absolute rather than percentage differences.
79. used by subroutine `SOL` for RE calculations.
80. used by subroutine `SOL` for RE calculations.
81. limit for the number of Type IVb iterations (default = 20).
82. iiye for `FIML` and `RECHEAP`. (1 if compute expectations before the first iteration.)
83. used by command `FMLRE`.
84. counter for number of times subroutine `RESID` is called.
85. position of `CBASE` for `FIML` and `RECHEAP`.
86. used by `PRINTVAR`.
87. `ICHP` for `FIML` and `RECHEAP`.
88. `LMTRE` for `FIML` and `RECHEAP`.
89. `IDER` for `FIML` and `RECHEAP`. (Compute every `IDER` iterations.)
90. `IIDER` for `FIML` and `RECHEAP`. (1 if compute derivatives before the first iteration.)
91. 1 = skip the call to `CTZ` in `TSLS00`. (RAYFAIR)
92. open
93. number of `ZLOG` errors permitted when command `CRZ` is used (default = 100).
94. number of `ZLOG` errors permitted in `SOL1` before an exit (default = 0).
95. 1 = print `Y` and `YY` variables when a `ZLOG` error is obtained in `SOL1`.

96. number of lags to print for the forecast memos for FM (default = 12).

97. maximum number of errors before stopping. Not in effect if the program is being used interactively.

98. Used as a counter for LA(97).

99. used for error trapping.

100. 1 = the `LHS` variable is taken to be the log of the original variable for all equations estimated using the `OLS` or `2SLS` command. (See the discussion of the `EAUTO` command for the use of this option.)

101. used by `OPTC STOCHASTIC`. (1 if `STOCHASTIC`.)

102. used by `OPTC STOCHASTIC`. (space pointer for `UUUU`.)

103. used by `OPTC STOCHASTIC`. (`NTRIAL`.)

104. limit for the number of iterations in estimating the serial correlation coefficients (default = 200).

105. used by `OPTC STOCHASTIC`. (space pointer for *IFIX*.)

106. used by `OPTC STOCHASTIC`. (`NFIX`.)

107. limit for the number of iterations for `LAD` and `2SLAD` (default = 20).

108. 0 = default value (all residuals are plotted).
n =only the last n residuals of the sample period are plotted for the `OLS` and `2SLS` commands.

109. 1 = the `2SLS` command prints summary statistics using the coefficients currently in core (no estimation is done). (See the discussion of the `3SLS` and `FIML` commands for the use of this option.)

110. 1 = do not print forecast memos when running a forecast for FM. (RAYFAIR)

111. `VSTACK` location in SPA.

112. `JSTACK` location in LA.

113. see `BEGINNOPRINT`.

114. used by `OPTC STOCHASTIC`. (`ITRGT`.)

115. for term structure paper. (RAYFAIR)

116. for exchange rate paper. (RAYFAIR)

117. for tern structure paper, revision 4. (RAYFAIR)

118. 1 = set `RESID` using `CREATEU` for the equation being estimated for the estimation SMPL for the equation.

119. 1 = the number of the equation to use in `CREATEU`.

120. 1 = don't print error message in RESIDT.

121. the first LA(121) endogenous variables are tested for convergence for the Gauss-Seidel technique (default = `NY`).

122. used by command `STOABC`. (RAYFAIR)

123. open

124. 1 = skip the perturbation search for the derivative step sizes in the calculation of the `FIML` covariance matrix.

125. current iteration number in `SOL1`.

126. 1 = estimate the `FIML` covariance matrix without using Parke's procedure.

127. number of Gauss-Seidel iteration checks to skip (default = 0). When this is greater than 0, the Gauss-Seidel technique will not be stopped in less than LA(127) + 1 iterations. This option cannot be used if damping is desired.

128. open

129. used by commands `ADDFACT`, `ASSIGN`, `COEF`, `XDATA`, and `YEXOG`.

130. open

131. 1 = use $GNPR = GNPR^*$ as starting position for command `OPTC` for FM. (RAYFAIR)

132. 1 = `S` is taken to be diagonal for the draws of the errors terms for command `STOABC` even though it may not in fact be diagonal.

133. used by command `FIML`.

134. used by command `FIML`.

135. 0 = the draws of the exogenous variable errors pertain to the changes of the exogenous variables. (See the discussion in Section 10.9.3.) 0 is the default value.

     1 = the draws of the exogenous variable errors pertain to the levels of the exogenous variables.

136. open

137. open

138. open

139. used by command `FIML`.

140. 1 = (if relevant) write out `COV` matrix in block diagonal form in the `2SLS` command with only the blocks stored on the disk. (If relevant) read the `COV` matrix in block diagonal form in the `STOABC` command. Also used in `SOLK` for the MC model.

141. 1 = in command `CRS` do not zero out rows and columns of the S matrix for equations that are dropped from the model.

142. open

143. open

144. open

145. open

146. open

147. 1 = do not print a message after achieving Type II convergence.

148. 1 = do not print a message after achieving Type III convergence.

149. 1 = do not print a message after achieving Type IV convergence.

150. small (actual) value of `NCOEF` for `FIML` and `RECHEAP`.

151. large value of `NCOEF` for `FIML` and `RECHEAP`.

152. position of `IC1` for `FIML` and `RECHEAP`.

153. used by `WRITECOV` (`SUBROUTINE WRCOV`).

154. open

155. 1 = when the Parke algorithm is used for `FIML` estimation, take all the means of the `Z` variables (except the constant term) to be zero. In other words, turn off that part of the Parke algorithm that uses the `Z` means.

156. open

157. open

158. open

159. used by `OPTC`

160. open

161. open

162. unit to read the `H` matrix to restart. Only relevant for `IROUT = 0`. (`IROUT` is always 0.)

163. unit to write the `H` matrix at the stopping point for possible future use. Only relevant for `IROUT = 0`.

164. 1 = when solving a model, do the solution only for period `NBEGQ` even though the period `NBEGQ-NENDQ` has been specified. Do printing, however, for the entire `NBEGQ-NENDQ` period. This option is useful for solving rational expectations models under the assumption that the expected values of the exogenous variables are equal to the actual values. See the second to last paragraph in Section 11.2.1 in Fair (1984) for a discussion of this.

165. open

166. 1 = print `LAMBDA` and `F` values after each iteration. See LA(173) for even more printing.

167. 0 = DFP algorithm.
1 = BFGS algorithm. Only relevant for `IROUT =0`.
2 = Steepest descent algorithm. Only relevant for `IROUT =0`.

168. 1 = print details of the line search. Only relevant for `IROUT =0`.

169. 0 = default value (two sided derivatives used).
1 = use one sided derivatives. Only relevant for `IROUT =0`.

170. open

171. open

172. special for FIML and RE. S is x 0 0 0 0 x x x/0 x 0../ 0 0 x 0 ../0 0 0 x 0 ../0 0 0 0 x 0 ../x 0 0 0 0 x x x x/ x 0 0 0 0 x x x x.

173. 1 = print `X` values after each iteration. When LA(173)=1, the option for LA(166)=1 is also in effect.

174. 1 = take `S` to be diagonal for `FIML` estimation.

175. 1 = print the steps involved in arriving at the final perturbation size for each coefficient for the calculation of the `FIML` covariance matrix.

176. 1 = `NL2SLS` command adds to `COV` matrix.

177. number of unconstrained coefficients up to equation being estimated by `NL2SLS`. Only relevant if LA(176)=1.

178. open

179. 1 = `S` matrix is estimated as a diagonal matrix in command `2SLS`.

180. 1 = `NENDQ` desired for command `DRUN`.

181. used by command `DRUN`.

182. 1 = do not call `SOL` in `STOABC`. Just do the draws of the random variables. This option is not of general interest.

183. used by `FUNC4` to `RESID` when `DZEA` used.

184. used by `FUNC4` to `SOL1` and `RESID`.

185. used for experimenting with the Parke algorithm regarding the return of 0 when the model will not solve.

186. 1 = Hayashi-Sims method (HS).

187. order of the moving average of the error term for the HS, H1, H2, H1R, and H2R methods.

188. 1 = Hansen's method (H1 or H2).
See also `PDL` discussion.

189. 1 = Hayashi-Sims method under the assumption of an autoregressive process for the error term (HSRR).

190. 1 = M for Hansen's method (LA(188)=1 or LA(192)=1) estimated the general way. (H1 and H1R methods.)
0 = M for Hansen's method (LA(188)=1 or LA(192)=1) estimated using (A2), p. 796, in Hayashi and Sims (1983). (H2 and H2R methods.) (0 is the default value.)

191. 1 = use (6.13) and (6.14), p. 212, in Fair (1984) to compute the 2SLS estimates.

If LA(191)=0, the following formulas are used in place of (6.13) and (6.14):

$$\hat{\alpha}_i = [(\hat{X}_i - X_{i-1}\hat{\rho}_i)'(X_i - X_{i-1}\hat{\rho}_i)]^{-1}(\hat{X}_i - X_{i-1}\hat{\rho})'(y_i - y_{i-1}\hat{\rho}_i)$$
$$\hat{\rho}_i = (\hat{u}'_{i-1}\hat{u}_i)/(\hat{u}'_{i-1}\hat{u}_{i-1})$$

$$\hat{X}_i = D_i X_i$$

The equation for $\hat{\rho}_i$ is the same as (6.14)$'$, p.212, in Fair (1984). It should be noted that (6.15) in Fair (1984), not (6.15)$'$, is always used to compute $\hat{V}_{2ii}$. If X_{i-1} and y_{i-1} are in Z_i, then LA(191)=0 and LA(191)=1 are the same aside from rounding error. LA(191)=1 is slower, and so this option should not be used unless one has to. If LA(191)=0 and y_{i-1} and/or one or more variables in X_{i-1} are not in Z_i, the resulting estimates will not be consistent.

192. 1 = Hansen's method with an autoregressive structural error (H1R or H2R).

193. open

194. used by `HANSEN`

195. open

196. open

197. 1 = use `C( , )` for storage for t statistics for the MC model in subroutine `TSLS0`. Also maybe used by subroutine `TSLS00`.

198. used in `SOL4, SETUPRE. ALTRHO.`

199. open

200. open

201. open

202. open

203. open

204. open

205. open

206. open

207. open

208. open

209. open

210. open

211. used for White heteroskedasticity correction.

212. 1 = test hypothesis that first coefficient in the equation is zero when White correction is used (`WALD` test).

213. used by `EAUTO`, passed to `TSLS0`. (? if use LA(211) here) Also ? do not set `C(...` after estimation. For `TEST1, TEST2, TESTH, TESTSTAB`.

214. order of `MA` process for `OLS`.

215. open

216. 1 = use Newey-West correction.
or
0 = print “Solution error” in `SOL1`.
1 = do not print “Solution error” in `SOL1`.

217. open

218. open

219. open

220. open

221. open

222. open

223. open

224. open

225. open

226. open

227. open

228. open

229. open

230. open

231. open

232. open

233. open

234. open

235. open

236. open

237. open

238. open

239. open

240. (RAYFAIR)

241. (RAYFAIR)

242. see `STOSIM NORESETU`

243. see `2SLS or OLS BIASCORRECTION`

244. see `STOSIM SETU`

245. used by `STO0` and `STOABC`

246. open

247. number of variables to check in `RETURNY` for the MC model. Returns IBAD=1 if there is a big UR error.

248. pointer in `STOABC`. Equals 1 if LA(99)=1 and LA(249)=1 (`RETURNY`).

249. see `STOSIM RETURNY`.

250. see `STOSIM RETURNU`

251. open

252. (RAYFAIR)

253. (RAYFAIR)

254. used in `FMLRE2`.

255. 1 = `PDL` linear plus constraint. `COV` computed.
     2 = `PDL` linear plus constraint, 2 variables. Stoch. sim. does not work for this.
     See `PDL` discussion.

256. `NYE` for RE option. Also, beginning `NLEN` for `PDL`.
See `PDL` discussion.

257. `JFIRST` for RE option. Also, `NTRIAL` for `PDL`.
See `PDL` discussion.

258. `JLAST` for RE option. Also, 1 if LDV is the first variable in the equation for `PDL`. Used only for stoch. sim.
See `PDL` discussion.

259. `KFIRST` for RE option. Also, 1 for bootstrap for `PDL`. Keep LA(257) also. For both LA(256)=1 and LA(260)=1.
See `PDL` discussion.

260. `KLAST` for RE option. Also, 1 if `PDL` linear and no constraint (2 variables at the end of the equation). This has not been checked. Maybe does not work with `RHO`
See `PDL` discussion.

261. `NH` for RE option. Also, 1 for printing test if `PDL`. (? if true)
See `PDL` discussion.

262. `NXE` for RE option. Also, 1 means draw normal for `PDL`, then bootstrap. Need LA(259)=1 plus other stuff.
See `PDL` discussion.

263. Position of `IYE` for RE option.

264. `KCUR` for RE option.

265. `JJ` for RE option.

266. Position of `IXE` for RE option.

267. `ISMALL` for RE option.

268. SPA(2)+1 for RE option (maybe).

269. counter for `RETURNY` in `STO0`. Limit of 10 times to try. (Maybe now only equals 1 since returning more screws up draws.??)

270. Equals 1 if `J` is greater than `NBEG` in `FIML0`. Used in `SOL4` for RE option. Used in `FMLRE2`.

271. used in `FMLRE2`.

272. minimum number of Type II iterations (set only by `SETLA` command).

273. open

274. open

275. open

276. open

277. open

278. open

279. open

280. see `STOMULT NODRAWU`

281. open

282. Position of `YE0` for RE option.

283. open

284. open

285. 1 = `FILESAVEM` option.

286. 1 = `FILEUSEM` option.

287. open

288. open

289. open

290. `NBEG-1-JFIRST-(ITER4+1)` for RE option.

291. open

292. open

293. open

294. open

295. used in `FUNC4`, `DZEB`.

296. used in `FIML0`.

297. `NL2` so can pass `JPTR, JTEST` vectors.

298. number of Jacobian elements (automatic).

299. 0 = new way for Jacobian.
1 = old way for Jacobian (still works).

300. 0 = default value (divide by T-k).
1 = divide by T (as in the old version of the program).

301. used in `RESID, SOL, SOLA` and elsewhere. `NL8` in `MAINA`.

302. used in `RESID, SOL, SOLA` and elsewhere. `NL9` in `MAINA`.

303. used in `RESID, SOL, SOLA` and elsewhere. `NL10` in `MAINA`.

304. used in `RESID, SOL, SOLA` and elsewhere. `NL11` in `MAINA`.

305. used in `RESID, SOL, SOLA` and elsewhere.

306. `NL12` in `MAINA`.

307. `MAXSTK`.

308. `MAXLNK`.

309. `MAXLS`.

310. `NL14`.

311. `NL15`.

312. `NL16`.

313. 1 = go on in `MAIN` after `QUIT` command from previous job.

314. used by `CNTRL` and `MAINSW`.

315. open

316. open

317. open

318. open

319. open

320. (RAYFAIR)

321. (RAYFAIR)

322. (RAYFAIR)

323. (RAYFAIR)

324. open

325. open

326. open

327. open

328. open

329. open

330. open

331. open

332. maximum q. Default is 72. For `PDL`.
See `PDL` discussion.

333. 1 = do `COV` matrix not incorporating q constraint. For `PDL`.
See `PDL` discussion.

334. 1 = do two searches rather than three. For `PDL`.
See `PDL` discussion.

335. 1 = do Monte Carlo for unit root problem. Also want LA(258)=1.
See `PDL` discussion.

336. used by `PDL`.
See `PDL` discussion.

337. open

338. open

339. open

340. open

341. 1 = do every four quarters in `RMSEA` (to match MC). Used with LA(344)=1. (MCMODEL)

342. set internally. Used in `SOLA` and `SOLK` for the MC model. It is 1 for the first time in `SOLK`, 2 for the first time for the second or higher set of four quarters, and 0 otherwise (no reading). (MCMODEL)

343. 1 = create residuals for the trade share equations for the MC model. (MC-MODEL)

344. 1 = for the MC model. Information used by `RDC`.
(MCMODEL)

345. set internally in `SOLK` (for MC Model). It is 1 if files have already been opened. (MCMODEL)

346. set internally. It is used in `EQ` for the MC model. (MCMODEL)

347. set internally. Used in `RDC` for the MC model. (MCMODEL)

348. open

349. 1 = call `SOLK` but do nothing for the MC model. 2 = call `SOLK` but do nothing except for $GERSN$, $GEEN$, $GEPYN$, RSN, and $GDPDN$ for the MC model. [This needs to be updated.] (MCMODEL)

350. used in `SOL1` (continue the solution even if an error is found for a particular country in the MC model). (MCMODEL)

351. used by the `FILEWRITE` option of `RMSE`.

352. used by the `FILEWRITE` option of `OLS, 2SLS`.

353. used in `SOL1` for the MC model. (MCMODEL)

354. 1 = do printing in subroutine `SOLK` for the MC model. (MCMODEL)

355. 1 = print α's in `SOLK` for the MC model. (MCMODEL)

356. 1 = print the sum of the α's in `SOLK` for the MC model. (MCMODEL)

357. used in `SOL1` for the MC model (number of iters overall). Also, used in `STO0` with respect to `SEED`. (MCMODEL)

358. used in `SOLA` for the MC model (number of Type I iters). (MCMODEL)

359. used in `SOLA` for the MC model. (MCMODEL)

360. used by `GET0` for `FILEWRITE` for `SOLVE`. Also, used in `STOEXG` regarding FM. Also, used in `STOABC`, maybe tied to `STOEXG`. (Two uses.)

361. used by `STOABC`.

362. open

363. used by `STOABC` (`NSSS`). Also `PRNTCV`.

364. used by `GET0` for `FILEWRITE2` for `SOLVE`. Also, `PRNTF` (RAYFAIR). (Two uses.)

365. `PRNTF` (RAYFAIR).

366. see `SETUPCHANGEVAR`. (MCMODEL)

367. see `SETUPCHANGEVAR ANNUALRATE`. (MCMODEL)

368. used in `RMSE`, but not clear what. Also used in `SOL1`. (MCMODEL)

369. used in `SOLK` for the MC model. 1 if use autoregressive equations for the trade shares. (MCMODEL)

370. number where the annual variables begin for the MC model. (MCMODEL)

371. number where the annual variables end for the MC model. (MCMODEL)

372. 1 = take alphas to be unchanged when forecasting with the MC model. (MCMODEL)

373. 1 = do annual averages of quarterly data when using `PRINTVAR`.

374. open

375. 0 for `ITS` = 1.
1 for `ITS` = 0. (Trade shares exogenous) (MCMODEL)

376. open

377. 1 for `IBEG=1`. (MCMODEL)

378. 0 = use `PEX = DEL3*USPX` and not `PEX = PSI1*PX` in `SOLK`.
1 = use `PEX = PSI1*PX` and `DEL3 = PEX/USPX` in `SOLK`. (MCMODEL)

379. position of the lagged dependent variable in the equation (for `MUE`). Used with SPA(80).

380. used in `FP2G.FOR` I think for How Fast paper. Also, M̑UE in `FP2C` I think (number of repetitions for the median estimates). (RAYFAIR)

381. used in `FP2G.FOR` I think for How Fast paper. Also, `MUE` in `FP2C` I think (number of iterations for the median estimates). (RAYFAIR)

382. used in `FP2G.FOR` I think for How Fast paper. Also, `MUE` in `FP2C` I think (counter—set internally). (RAYFAIR)

383. `MUE` in `FP2C` I think (current number of iterations—set internally). (RAYFAIR)

384. `MUE` in `FP2C` I think (equals 1 after the first iteration—set internally). (RAYFAIR)

385. used in `FP2G.FOR` I think for How Fast paper. Also, `MUE` in `FP2C` I think (equals 1 if all done—set internally). (RAYFAIR)

386. used in `FP2G.FOR` I think for How Fast paper. Also, `MUE` in `FP2C` I think (1 = do median estimation). (RAYFAIR)

387. used in `FP2G.FOR` I think for How Fast paper. Also, `MUE` in `CRDATA` (1 = for testing median estimates—don't change data for stochastic simulations). (RAYFAIR)

388. `MUE` in `FP2C` I think (1 = do median systems estimation). (RAYFAIR)

389. `MUE` in `CRDATA` (number of largest lag for first stage regressors). (RAYFAIR)

390. used in `FP2G.FOR` I think for How Fast paper. Also, in `TSLS00` write `C` to unit 66 (for MUE—see LA(401)). (RAYFAIR)

391. used by `TESTSTAB1, TESTSTAB2`. Also, 1 = do RMSEs for α's in `SOLK` for the MC model. (part MCMODEL)

392. used by `TESTSTAB1, TESTSTAB2`. Also call `LINK2.BIN` in `SOLK` for the MC model. (part MCMODEL)

393. see `SETUPTEST PRINTTEST`

394. see `SETUPTEST PRINTSTAB`

395. used by `TEST1`, `TEST2`, and `TESTH` with respect to calls to `MODEQ`.

396. used for How Fast (RAYFAIR)

397. 1 = print to unit 6 even when `FILEWRITE=fn` in `RMSE` (i.e., when LA(351)=1).

398. used by `TESTSTAB1, TESTSTAB2` for `NQTR` call. (Old 382) Also for `TESTEND1, TESTEND2`.

399. 1 = don't print extra space for gaps in `PLOTU`. (Must use `SETLA` for this.)

400. used by `PDL`.

401. used in `FUNC2` I think for How Fast paper. (Old 380) Also used by `PDL`. Also used with LA(390) to write `C's` to unit 66 (file `TEMP1` in `TSLS00` for `MUE`. If LA(390)=1, then write lagged dependent variable coefficients in ASCII if LA(401)=1. If LA(401)=0, then write all coefficients in binary. Remember to check `FP0.FOR`, where `TEMP1` should be binary for LA(401)=0 and regular for LA(401)=1. (This is a mess.)

402. used by `PDL`

403. see `SETUPSOLVE TESTEQS`.

404. (RAYFAIR)

405. used by `OPTC`. 1 if one sided derivatives and one control variable.

406. used by `OPTC`. used when LA(405) = 1.

407. used by `CRU`. For `CREATEU` when LA(407)=1, use `YY` and not `Y` in call to `RESID`. Also used for RS for `OPTC`.

408. 1 = for special treatment for some US equations in `SUBROUTINE CRU` when `CREATEU BIASCORRECTION` is used. (RAYFAIR)

409. open

410. used by `PDL` in `PDL3`.

411. used by `PDL` in `PDL2`.

412. open

413. see `OPTC NORESETA`

414. see `OPTC NORESETB`

415. (RAYFAIR)

416. see `STOSIM DRAWUHIST`

417. see `STOSIM FIRSTUHIST=p`

418. see `STOSIM LASTUHIST=p`

419. see `STOSIM MCMODEL`

420. (RAYFAIR)

421. (RAYFAIR)

422. see `SETUPCHECK`

423. see `SETUPCHECK`

424. for RE and `OPTC`

425. 1 = turn off trick for one sided (see LA(405). (For RE and `OPTC`.)

426. open

427. open

428. 1 = test `MAA` in `FP1K.FOR`.
2 = test `MAB` in `FP1K.FOR`.
(MCMODEL: used for checking `MAA, MAB` in `RDMC`)

429. 1 = create `MAA.MCB` first time in `FP1K.FOR`.
2 = create `MAB.MCB` first time in `FP1K.FOR`.
(MCMODEL: used for `MAA.MCB, MAB.MCB` in `WRMC`.)

430. open

431. (RAYFAIR) MP effectiveness paper)

432. open

433. open

434. see `PRINTVAR`

435. see `MULT`

436. see `OPTC`

437. 1 = error in solution, do not print PRED and MULT on units 73 and 74 if error earlier with respect to LA(434) and LA(435).

438. see `SETUPTESTEND`.

439. used for baseball work. (RAYFAIR)

440. open

441. open

442. see `SETUPTEST`.

443. open

444. open

445. open

446. open

447. open

448. 1 = do not close unit 14 for `LOADDATA`.

449. 1 = do not open unit 14 for `LOADDATA`. It should already be opened using previous LA(448)=1.

450. 1 = do not close unit 14 for `LOADDATA`. ?? open

451. 1 = do not open unit 14 for `LOADDATA`. It should already be opened using previous LA(448)=1. ?? open

452. (RAYFAIR)

453. (RAYFAIR)

454. – 500. open

SPA VECTOR:

1. dimension of the SPA vector in `MAIN`. This can be changed before `MAIN` is compiled.
2. number of elements currently used in the SPA vector.
3. largest element number of the SPA vector that can be user supplied (number is 100).
4. standard error of the last equation estimated by `OLS` or `2SLS, SIG`.
5. sum of squared residuals of the last equation estimated by `OLS` or `2SLS, SUM2`.
6. value of the tolerance criterion for the Gauss-Seidel technique (default = .001).
7. value of the tolerance criterion for the Type III iterations (default = .003).
8. value of the tolerance criterion for the Type IV iterations (default = .004).
9. value of the damping factor for the Type IVb iterations (default = 1.0, which is no damping).
10. used by command `FMLRE` (maybe).
11. greater than 0 means seed for `STOABC`.
12. counter for RESID and RESIDF.
13. open
14. open
15. open
16. value of the tolerance criterion for the Type II iterations (default = .002).
17. value of the damping factor for the successive coefficient estimates when the less expensive method is used for the `FMLRE` command (default = 1.0, which is no damping).

18. value of the tolerance criterion for the change in the coefficient values for the Parke algorithm (default = .001).

19. value of the tolerance criterion for the change in the value of the log of the likelihood function for the Parke algorithm (default = .0001).

20. open

21. open

22. open

23. open

24. open

25. open

26. beginning of the sample period for a country for the MC model. The country is not used in the solution for a given period if the period is before SPA(26). This option cannot be used in the solution of the overall model (`NK1=0`).

27. beginning of the flexible exchange rate period for a country for the MC model. This option cannot be used in the solution of the overall model (`NK1=0`).

28. end of the sample period for a country for the MC model. The country is not used in the solution for a given period if the period is after SPA(28). This option cannot be used in solution of the overall model (`NK1=0`).

29. open

30. value of the tolerance criterion for the solution of the MC model (default = .002).

31. value of the numeric derivative stepsize for the DFP algorithm (default = .0005).

32. open

33. value of the tolerance criterion for the iterations involved in estimating the serial correlation coefficients (default = .00001).

34. used in `SETUPRE (DAMP)`

35. value of the tolerance criterion for the `LAD` and `2SLAD` iterations (default = .002).

36. value of `q` for `2SLAD` (default = .5).

37. value of the stopping criterion for the DFP algorithm in terms of percentage change in each coefficient from one iteration to the next (default = .0001 percent).

38. open

39. value of the damping factor for the Gauss-Seidel iterations (default = 1.0, which is no damping). See the discussion in Section 10.1.

40. value of the smallest change in the log of the likelihood function allowed in computing numerical derivatives in the calculation of the `FIML` covariance matrix (default = .00001).

41. value of the largest change in the log of the likelihood function allowed in computing numerical derivatives in the calculation of the `FIML` covariance matrix (default = .0001).

42. value of the initial step sizes for the derivatives for the calculation of the `FIML` covariance matrix (default = 1.001).

43. open

44. damping factor in `RESID` for `RS` for FM (default = .2).

45. tolerance criterion for the MC model for iterations between countries (maybe).

46. value of the tolerance criterion for the successive coefficient estimates when the less expensive method is used for the `FMLRE` command (default = .002).

47. perturbation size for the calculation of the numerical derivatives for the less expensive method for the `FMLRE` command (default = .01).

48. value of the stopping criterion for the DFP algorithm in terms of each gradient value as a percent of the objective function (default = .0000001).

49. value of `a` in expression `log(ZZ + a)` for FM.

50. trial stepsize for the coefficient searches for the Parke algorithm (default = .01).

51. used internally when LA(195)=1.

52. used internally when LA(195)=1.

53. used internally when LA(195)=1.

54. used internally when LA(195)=1.

55. R^2 from last estimated equation.

56. DW from last estimated equation.

57. coefficient of lagged dependent variable from last estimated equation (?).

58. weight on `RS` in `M1` equation for FM for optimal combination policy (`ITYPE = 1`).

59. weight on `RS` in `M1` equation for FM for optimal combination policy (`ITYPE = 0`).

60. missing data value.

61. $u'Du$ for `2SLS`.

62. number of coefficients estimated by `OLS` or `2SLS`, `TK`.

63. number of observations used by `OLS` or `2SLS`, `TNOBM`.

64. t-statistic (for writing to `TEMP`).

65. open

66. open

67. open

68. open

69. open

70. stepsize for `PDL`. Default is 1.0. Must be 1.0 when LA(260)=1.
See `PDL` discussion.

71. open
72. open
73. open
74. open
75. open
76. open
77. open
78. open
79. open
80. guess for the bias for `MUE`. Default is 0.
81. ... 90. Starting values for `RHOs`, 81=`RHO1`, 82=`RHO2`, etc. Also, 81 is the stopping tolerance for the iterations for `MUE`.

C APPENDIX C: Installation Instructions for the Fair-Parke Program

1. If you have downloaded `FP.ZIP`, unzip it, which creates `FP.EXE`. Then skip to step 3.

2. If you have downloaded `FPFOR.ZIP`, unzip it, which creates `FP.FOR`. Then compile and link edit `FP.FOR` and call the link edited file `FP.EXE`.

3. Unzip the file `FPEG.ZIP`, after downloading it. This will result in the following files:

   `IS.INP`, `IS.DAT`, `IS.VAR`, `ISFSR3.DAT`, `IS.OUT`, `ISRE.INP`, `ISRE.OUT`, `SAR.INP`, `SAR.DAT`, `SAR.OUT`.

4. At the DOS prompt type `FP > OUT` and hit the enter key. Then wait a couple of seconds and type `INPUT FILE=IS.INP;` and hit the enter key. This will run the first example in Appendix A and store the output in file `OUT`. When the job is done, compare `OUT` to `IS.OUT`. The output in these two files should be the same aside from rounding error. If this is true, you are all set.

5. If you are going to be dealing with models with rational expectations, you should also run `ISRE.INP`, which is the second example in Appendix A, and `SAR.INP`, which is the example in Section 15.4. The output from running these two examples is in `ISRE.OUT` and `SAR.OUT`, respectively.

Compiling Notes

You can change the amount of memory that the FP program takes by changing the dimensions of SPA and LA near the beginning of the FORTRAN code. If you change these dimensions, change also `NSPA` and `NLA` to match. The missing value number is in `SPA(60)`, which is set at 999999. You can change this before compilation if desired.

D References

References

[1] The main reference is *Macroeconometric Modeling* (MM), *fairmodel.econ.yale.edu*. Click the second picture and then "Macroeconometric Modeling."

[2] Andrews, Donald W.K., 2003, "End-of-Sample Instability Tests," *Econometrica*, 71, 1661–1694.

[3] Andrews, Donald W.K., and Ray C. Fair, 1988, "Structural Inference in Nonlinear Econometric Models," *Review of Economic Studies*, 55, 615-639.

[4] Andrews, Donald W.K., and Werner Ploberger, 1994, "Optimal Tests When a Nuisance Parameter is Present Only Under the Alternative," *Econometrica*, 63, 1383-1414,

[5] Fair, Ray C., 1984, *Specification, Estimation, and Analysis of Macroeconometric Models*. Cambridge, MA: Harvard University Press.

[6] Fair, Ray C., and John B. Taylor, 1983, "Solution and Maximum Likelihood Estimation of Dynamic Rational Expectations Models," *Econometrica*, 51, 1169-1185.

[7] Fair, Ray C., and John B. Taylor, 1990, "Full Information Estimation and Stochastic Simulation of Models with Rational Expectations," *Journal of Applied Econometrics*, 5, 381-392.

[8] Granger, C. W. J., and Paul Newbold, 1986, *Forecasting Economic Time Series*. Orlando, FL: Academic Press, Inc.

[9] Hannen, E. J., and J. Rissanen, 1982, "Recursive Estimation of Mixed Autoregressive-Moving Average Order," *Biometrika*, 69, 81-94. Correction, 1983, 70, 303.

[10] Hansen, Lars, 1982, "Large Sample Properties of Generalized Method of Moments Estimators," *Econometrica*, 50, 1029-1054.

[11] Hayashi, Fumio, and Christopher Sims, 1983, "Nearly Efficient Estimation of Time Series Models with Predetermined, but not Exogenous, Instruments," *Econometrica*, 51, 783-798.

[12] Newey, Whitney K., and Kenneth D. West, 1987, "A Simple Positive Semi-Definite Heteroskedasticity and Autocorrelation Consistent Covariance Matrix," *Econometrica*, 55, 703-708.

[13] Parke, William R., 1982, "An Algorithm for FIML and 3SLS Estimation of Large Nonlinear Models," *Econometrica*, 50, 81-96.

[14] Sargent, Thomas J., 1976, "A Classical Macroeconomic Model for the United States," *Journal of Political Economy*, 84, 207-237.

[15] White, Halbert, 1980, "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test of Heteroskedasticity," *Econometrica*, 48, 817-838.