INTRODUCTION TO TSP 4.5

Ingmar R. Prucha
Department of Economics
University of Maryland
College Park, MD 20742

January 2003

The purpose of this manual is to provide, in compressed form, a basic working knowledge of TSP and to explain how to execute TSP within the Economics Department's Microsoft Windows computing environment. TSP is one of the most widely used econometrics packages. It has its roots in economics, is typically quite up to date on modern econometric techniques, and is also very user friendly. TSP has many more features than presented here. Please consult the TSP User's Guide and the TSP Reference Manual (Version 4.5) if you are looking for additional features and/or for a more detailed description of the features presented here. The TSP User's Guide and TSP Reference Manual are also available online.

There are many other econometric and statistics software packages on the market. While TSP is a premier econometrics package with a wide array of techniques, no single package dominates uniformly all other packages in terms of features and user friendliness. For certain applications other packages may be more appropriate. For a review of several widely used econometrics and statistics software packages see, e.g., MacKie-Mason, Econometric Software: A User's View, *Journal of Economic Perspectives*, Fall 1992.

This manual has been written with considerable care for accuracy, but complete accuracy cannot be guaranteed. It is based on an earlier manual written jointly with Sylaja Srinivasan. As indicated above, the intention of writing this manual was not to replace the TSP User's Guide and the TSP Reference Manual. Rather, the intention was to provide you with a tool that should make it easier for you to get started using TSP within the Economics Department's computing environment. Any comments and suggestions you may have would be appreciated.

Table of Content

1. Running TSP on the Economics Departments Windows Computers
   - 1.1 Running TSP Interactively
   - 1.2 Running TSP in Batch Mode
   - 1.3 Obtaining a Hardcopy of the Output
   - 1.4 Customizing the TSP Output
   - 1.5 Executing Sets of TSP Commands
   - 1.6 Online Help Facility
   - 1.7 TSP Sample Program
2. Basic Conventions for TSP Statements and Names
3. Selection of the Observation Set
4. Time Series: Basic Input-Output Operations and Display
   - 4.1 Load or Read Command
   - 4.2 Print Command
   - 4.3 Plot Command
   - 4.4 Write Command
   - 4.5 Basic Descriptive Statistics
5. Parameters, Constants and Matrices: Basic Definitions
6. Basic Operations with Time Series, Parameters and Constants
   - 6.1 Genr Command
   - 6.2 Repl Command
   - 6.3 Set Command
   - 6.4 Trend Command
7. Basic Operations with Matrices
8. Equations
9. Estimation Techniques
   - 9.1 Ordinary Least Squares
   - 9.2 Nonlinear Least Squares
   - 9.3 Seemingly Unrelated Regression
   - 9.4 Two Stage Least Squares
   - 9.5 Three Stage Least Squares
   - 9.6 Full Information Maximum Likelihood
   - 9.7 Generalized Method of Moments Estimation
   - 9.8 Tobit Estimation
   - 9.9 Binary Probit and Logit Estimation
   - 9.10 Multinomial and Conditional Logit Estimation
   - 9.11 General M(aximum)-Estimation
   - 9.12 Panel Data Estimation
   - 9.13 Retrieval of Internal Estimation Results
10. Model Simulation
11. Random Number Generators
12. Do Loops, If and Goto Statements

Appendix: Background Theory on Generalized Method of Moments Estimation

1. RUNNING TSP ON THE ECONOMICS DEPARTMENT'S WINDOWS COMPUTERS

TSP can either be run interactively or in batch mode. The interactive mode is good for trying things out, and for quick one-time calculations. If you find yourself repeatedly typing the same commands, you can use a separate text editor such as TextPad or Notepad to create an ASCII file containing those TSP commands. Such a file is called a TSP program file. You can then execute those commands in batch mode. Rather than to use your text editor you can also use TSP through the Looking Glass (TLG) to edit and run your TSP program file in batch mode and examine the results. It seems good practice to use the extension .TSP for TSP program files. In the reminder of this section we outline how you can run TSP interactively and in batch mode. Knowledge of Windows is assumed. We note that the TSP program is also shipped with GiveWin. This additional interface will not be discussed in the following.

1.1 Running TSP Interactively

To run TSP Version 4.5 interactively on the Economics Department's Windows network click on 'Start/Applications Network Menu/Econometrics Programs/TSP/TSP Win32 4.5'. You will be asked:

Enter name of TSP program file or press ENTER:

If you simply press ENTER the TSP program will start an interactive session. You can now type in your TSP commands. To get help type HELP and press ENTER. To end your interactive TSP session type END, QUIT or STOP and press ENTER. You can also start an interactive session from the a Command Prompt window by typing G:\W2K\TSP45\TSPW.EXE and pressing ENTER.

1.2 Running TSP in Batch Mode

To run TSP in batch mode requires that you create an ASCII file containing the TSP command you would like to execute. As remarked above, such a file is called a TSP program file. The following supposes that you work with the TSP program file, say, I:\HOMEWORK\MYFILE.TSP and that you use TextPad as your ASCII editor.

(A) One way of editing and running your TSP program file in batch mode and examining the output can now be described as follows:

(i) Using Widows Explorer open the directory I:\HOMEWORK
(ii) Right click on your TSP Program file MYFILE.TSP, go to 'Send To' and select 'TextPad'
(iii) TextPad now opens the file MYFILE.TSP and you can start editing your TSP command in that file. At the end of you editing session save your file to disk by clicking on 'File', 'Save', etc.
(iv) To run your TSP program right click again on your TSP Program file

MYFILE.TSP, go to 'Send To' and select 'TSP Win32 4.5'. TSP opens a command prompt window. This window remains visible on the desk top until TSP ends the batch session. The output of the batch session is written to the file

I:\HOMEWORK\MYFILE.OUT

That is, TSP creates an output file in the same directory as where the TSP program is located.

v) To examine the output right click on the file MYFILE.OUT, go to 'Send To' and select 'TextPad'. This opens the output file in TextPad.

This cycle of editing and running your TSP program file and examining the TSP output file can be performed repeatedly as you modify and expand your TSP program.

(B) The front end program called TSP through the Looking Glass (TLG) provides another way of editing and running your TSP program file in batch mode and examining the output. TLG can be used as follows:

(i) Click on 'Start/Applications Network Menu/Econometrics Programs/TSP/TSP Looking Glass' to run the TLG program.

(ii) Click on 'File', 'Open', etc., to open the TSP program file, say,

I:\HOMEWORK\MYFILE.TSP

with the TLG editor. The program file appears in the left panel of TLG.

(iii) To run the TSP program click on the TSP icon on the TLG menu bar. The TSP output appears in the right panel of TLG.

This cycle of editing and running your TSP program file and examining the TSP output file can again be performed repeatedly as you modify and expand your TSP program. The TSP program and output files can be saved to disk by clicking on 'File', 'Save', etc.

(C) A third way of running TSP programs in batch mode is from a Command Prompt window. To run the TSP program I:\HOMEWORK\MYFILE.TSP and to write the output to I:\HOMEWORK\MYFILE.OUT open a Command Prompt window. At the command prompt first type

```
I:
```

and press ENTER and then type

```
CD:\HOMEWORK
```

and press ENTER. This makes the directory I:\HOMEWORK the active directory. Then type

```
G:\W2K\TSP45\TSPW.EXE MYFILE.TSP MYFILE.OUT
```

Obtaining a Hardcopy of the Output

As discussed, in batch mode TSP automatically saves the output of the TSP session to a file. By convention TSP automatically stores the output from, e.g., the file MYFILE.TSP in the file MYFILE.OUT. Often users also want the output of an interactive TSP session not only to appear on the screen but also to be stored in a file.

In interactive mode the TSP command that automatically creates and designates a file as the output file is

OUTPUT *filename*;

If the output is to be stored in a file in a directory other than the working directory, then the filename including the full path must be given in quotes. Quotes are not needed if the output is to be stored in a file in the working directory. If no extension is given, TSP automatically assumes that the extension of the file is OUT. For example, the command

OUTPUT 'A:\MYOUT'

designates the file A:\MYOUT.OUT as the output file. Once the OUTPUT command has been entered, all subsequent output is saved in the designated file until the command

TERMINAL

or another OUTPUT command is encountered. The TERMINAL command redirects the output to the screen.

1.4. Customizing the TSP Output

The OPTIONS command is handy for customizing various TSP options. The most common use of this command is to customize the output file for viewing on an 80-column terminal or similar printer. This is achieved by issuing the command

OPTIONS CRT;

This must be the very first TSP command if it is to take effect for the listing of the program in the output file.

The NAME command can be used to supply a job or user name as well as a title, both of which will then be printed as a header on top of each page of the output. The title can be any string of up to 60 characters (except for the ; and $ character) enclosed in quotes. For example, the command

NAME KMARX 'FINAL RESULTS FOR DAS KAPITAL';

specifies KMARX as the job name and FINAL RESULTS FOR DAS KAPITAL as the title. The title can be changed during the job by the TITLE command. For example,

TITLE 'FINAL ESTIMATION RESULTS';

specifies FINAL ESTIMATION RESULTS as the new title. If OPTIONS CRT; is used no paging of TSP output is done and the headers will not be printed at the top of each page of the output.

The ? character marks a comment -- TSP will ignore everything which follows a ? until the end of the line. This is useful for reminding yourself what the program is doing at the top and at any critical sections.

1.5 Executing Sets of TSP Commands

To run TSP in batch mode the user would store the commands for the entire session in a program file and then execute the commands in that file as described in section 1.1.

> All files that contain TSP commands must be written in ASCII for TSP to be able to execute them. As ASCII editors you may e.g. use the ED editor or the ASCII portion of ChiWriter. Under OS/2 the recommendation is to use the Enhanced Editor.

During TSP sessions the user can execute sets of commands stored in a by issuing the command

INPUT *filename*;

If the file is in a directory other than the working directory, then the filename including the full path must be given in quotes. Quotes are not needed if the file is in the working directory. If no extension is given, TSP automatically assumes that the extension of the file is TSP. For example, the command

```
INPUT 'A:\MYPROG';
```

initiates the execution of the commands stored in the file A:\MYPROG.TSP

Online Help Facility

TSP has an online help facility To access this help facility all you have to do is issue the command:

```
HELP;
```

and then follow the self-explanatory instructions to obtain more specific help information on a group of commands or on a specific command.

1.7 TSP Sample Program

Here is a simple example of how TSP works. It shows how (i) data can be read in from a file, say I:\ILLUS.DAT; (ii) how the data can be transformed by taking logarithms; (iii) how data can be printed; and (iv) how the data can be used in an ordinary least squares regression. Comment lines start with question mark.

```
? Set the data frequency to annual.
FREQ A;
? Specify the range of data as 1961-1975.
SMPL 61 75;
? Read in data on GNP, consumption and investment (in free format)
READ(file='I:\ILLUS.DAT') GNP CONS INV ;
? Print the data.
PRINT GNP CONS INV;
? Compute the natural log of GNP.
GENR LNGNP = LOG(GNP);
? Specify the range of the data as 1962-1975 to accommodate lags.
SMPL 62 75;
? Regress in log form GNP and lagged GNP and a constant.
OLSQ LNGNP C LNGNP(-1);
? Stop and end the program.
STOP;
END;
```

2. BASIC CONVENTIONS FOR TSP STATEMENTS AND NAMES

Every TSP statement starts with a command name. Many statements have options specified in parentheses after the command name. Statements may contain algebraic formulae and/or lists of objects (e.g. time series) separated by commas or spaces. The end of a statement is marked by a semicolon (;). It is possible to write several (short) statements into one line.

In interactive mode a carriage return defines the end of a TSP statement - semicolons are not necessary. To enter statements in interactive mode that are longer that one line you can use backlash (\) at the end of a line to connect lines (to form one statement).

In TSP you refer to all objects by some alphanumeric name. This means all names can be composed of letters A-Z and the numbers 0-9. Furthermore you can use the symbols @, %, , _,# . (# and % are not allowed in matrix names and ; and $ are not allowed at all in any text string. Furthermore C is not allowed as a variable name as it is reserved for the name of the intercept in a regression.) All names must begin with a letter. Objects are e.g. time series, vectors, constants, parameters, equations, models, matrices.

As an example, you can refer to a time series containing data on the US gross domestic product as GDP or US_GDP.

You can refer to leads and lags of a time series through subscripted arguments. For example GDP(-2) refers GDP lagged two periods and GDP(1) refers to GDP lead one period.

3. SELECTION OF THE OBSERVATION SET

The range of observations over which subsequent TSP commands are performed is defined by the SMPL command. The SMPL command remains active until it is overwritten by another SMPL command (or SELECT or SMPLIF command described below). The structure of the SMPL command is:

```
SMPL begin-1,end-1 begin-2,end-2
```

where 'begin-i' and 'end-i' refer to the i-th beginning and ending period. For example,

```
SMPL 1950,1960 1971,1983;
```

defines as the range of observations the periods 1950 to 1960 and 1971 to 1983.

TSP allows the definition of annual, quarterly, monthly, and undated data types. The periodicity of time series data is defined by the FREQ command. The structure of the FREQ command is:

```
FREQ per;
```

where 'per' stands for the periodicity: A annual, Q quarterly, M...monthly, N...undated.

As an illustration of the SMPL and FREQ command consider the following examples:

Command	Periodicity	Time period
FREQ A; SMPL 29,88;	Annual:	1929 to 1988
FREQ Q; SMPL 40:1,70:2;	Quarterly:	1st quarter of 1940 to 2nd quarter of 1970
FREQ M; SMPL 81:3,83:11;	Monthly:	3rd month of 1981 to 11th month of 1983
FREQ N; SMPL 33,673;	Undated:	33 to 673

It is sometimes convenient to select the observation set according to some selection rule based on the values of some series. This can be done with the SELECT command. For example, the commands

```
SMPL 1,50;
SELECT Y>0 & Y<=10;
```

select as the active observation set all observations between 1 and 50 for which the variable Y is greater than zero and less than or equal to ten. The SMPLIF command is similar to the SELECT command. The difference between the SELECT and SMPLIF command is that successive SMPLIF commands nest (define

smaller and smaller subsets of observations) while successive SELECT statements do not. (Both the SELECT and SMPLIF command allow for more complex arguments than given in the example.)

4. TIME SERIES: BASIC INPUT-OUTPUT OPERATIONS AND DISPLAY

4.1 LOAD or READ Command

The LOAD and READ command are synonymous (and so the user can use either LOAD or READ). The LOAD command is used to create one or more new time series and to store data in them. If you have only a small amount of data it is often convenient to load those data directly into your program with the LOAD command in free format. The LOAD command for one series in free format has the following structure:

```
LOAD var;
value-1 value-2
```

where 'var' stands for the variable name and 'value-i' stands for the i-th data value to be stored in the time series. For example,

```
SMPL 11 15;
LOAD IMP;
5 4.0 3 2 1.0;
```

creates a new series called IMP with values 5, 4, 3, 2, 1 in periods 11, 12, 13, 14, 15. In general the LOAD command requires that the user supplies as many data values as indicated by the SMPL command.

Suppose you are in interactive mode and you have more data than fit on one line. In this case case you can, as discussed above, use the symbol \ at the end of one line to connect two (or more) lines. For example, the above LOAD command can be written equivalently as:

```
LOAD IMP; 5 4.0 \
3 2 1.0;
```

The following generalization of the LOAD command is convenient to read in jointly the data for several time series in free format:

```
LOAD var-1,var-2, .   .,var-k ;
x11 x12 ..... x1k
x21 x22 ..... x2k


xn1 xn2       xnk ;
```

where 'var-i' stands for the i-th variable name and 'xti' for the t-th value of the i-th variable. If the volume of data entered in this way is large, the command and data can be moved to the bottom of the program with an END; statement added before and after. A simple LOAD; command without arguments in the main section of the program then causes the loading of the data from the bottom of the program. Printing of the data can be suppressed with a NOPRINT; command after the first END; statement.

For large data sets it is often best to read data from a separate file. Suppose the user has stored the data in a file, say, MYDATA.DAT in the form:

```
x11 x12       x1k
x21 x22       x2k


xn1 xn2       xnk
```

In this case the data can be read with the following LOAD command:

```
LOAD (FILE='MYDATA.DAT',FORMAT=FREE) var-1,var-2,.....,var-k;
```

As an illustration of the LOAD command consider the following sequence of commands

```
FREQ A;
SMPL 70,72;
LOAD E,F;
100 200
110  210
120 220 ;
PRINT E,F;
```

which generates the following output

```
            E         F

1970        100       200
1971        110       210
1972        120       220
```

The LOAD command also allows to read data according to the format rules of FORTRAN. Suppose data for the variables YEAR, GDP, CON, INVEST, IMP and EXP for 1945 to 1988 are stored in the file, say, USDATA.DAT in the format 1X,F4.0,5(2X,F10.4). The data can then be read into TSP by the following sequence of commands:

```
FREQ A;
SMPL 45,88;
LOAD (FILE='USDATA.DAT',FORMAT='(1X,F4.0,5(2X,F10.4))') YEAR GDP CON
     INVEST IMP EXP;
```

TSP can also read data directly from Lotus files. To read data from a Lotus file use the option FORMAT=LOTUS. For details please consult the TSP User's Guide and the TSP Reference Manual.

By default the LOAD command assumes that the data are organized by observation (i.e. the first observation for all series, the second observation for all series,...). If the data are organized by series use the option BYSER. As an illustration, the sequence of commands

```
FREQ A;
SMPL 70,72;
LOAD (BYSER) E,F;
100 110 120;
200 210 220;
PRINT E,F;
```

creates the same series E and F as in the (corresponding) example above.

4.2 PRINT Command

The PRINT command can be used to display data in tabular form on the screen. The PRINT command has the following structure:

```
PRINT var-1,var-2
```

where 'var-i' stands for the name of the i-th variable . The sample for which the data are displayed is governed by the SMPL command. For example,

```
FREQ A;
SMPL 71,75;
LOAD A;
3 2 5 10 1;
LOAD B;
151 89 17 1012 11;
SMPL 72,74;
PRINT A B;
```

will produce the following output:

```
                A             B

1972            2            89
1973            5            17
1974           10          1012
```

PLOT Command

The PLOT command can be used to display data in graphical form. The PLOT command has the following structure:

```
PLOT var-1,symbol-1,var-2,symbol-2
```

where 'var-i' stands for the name of the i-th variable and 'symbol-i' stands for the symbol that is to be used in plotting the i-th variable. For example, the commands

```
SMPL 3 5;
LOAD C;
10 11 12;
LOAD D;
20 21 22;
PLOT C,*,D,+;
```

will produce a plot as illustrated below:

```
          10        15        20        25
3         *                   +
4                               +
5              *                  +
```

WRITE Command

The WRITE command allows the user to write data and/or text in a pre-specified output format to the screen or to an external file. The format of the WRITE command is analogous to the LOAD command. The simplest form of the WRITE command is

```
WRITE var-1,var-2,    ..,var-n;
```

where 'var-i' stands for the i-th variable name. In this form the write command acts in the same way as the PRINT command. To write data in free format to a file use the WRITE command with the following options:

```
WRITE (FILE='filename',FORMAT=FREE) var-1,var-2,..  .,var-n;
```

To write the data in a certain format the user can, analogously to the LOAD command, specify the desired format according to the rules of FORTRAN. For example, suppose data for the variables YEAR, GDP, CON, INVEST, IMP and EXP for 1945 to 1988 are to be stored in the file, say, USBANK.DAT in the format 1X,F4.0,5(2X,F10.4). The data can then be written to that file by the following sequence of commands:

```
FREQ A;
SMPL 45,88;
WRITE (FILE='USBANK.DAT',FORMAT='(1X,F4.0,5(2X,F10.4)    YEAR GDP CON
      INVEST IMP EXP;
```

4.5 Basic Descriptive Statistics

Basic descriptive statistics of a set of time series variables can be computed with the MSD command, CORR command and COVA command. The basic structure of those commands is:

```
MSD var-1,var-2,.
CORR var-1,var-2,
COVA var-1,var-2,
```

where 'var-i' stands for the name of the i-th variable. The MSD command computes, stores and prints the means, standard deviations, minima, maxima, sums and variances of the variables named as arguments to the command. The CORR command computes, stores and prints a correlation matrix and the COVA command computes, stores and prints a covariance matrix for the variables named as arguments to the command. As discussed later, it is possible to retrieve the vector of means, the vector of standard deviations as well as the covariance matrix and correlation matrix for further use. The command

```
MSD(CORR,COVA) var-1,var-2,
```

enables you to get several forms of descriptive statistics on the variables at once, saving on computation time.

5. PARAMETERS, CONSTANTS AND MATRICES: BASIC DEFINITION

Parameters, constants and matrices differ from series in that they are not subject to the SMPL command. Parameters and constants differ from each other essentially only in that during estimation constants maintain their values while parameters are assigned new values (depending on the estimation) Matrices represent two-dimensional arrays.

The PARAM and CONST commands create and assign values to a parameter or constant. For example, the commands

```
PARAM ALPHA 0.6 BETA 1.0
CONST RHO 0.9 DELTA 2.5;
```

create parameters ALPHA and BETA and constants RHO and DELTA and assign to them the values 0.6, 1.0, 0.9 and 2.5, respectively.

The MMAKE command is used to create a matrix from a set of series, or a vector (i.e. a matrix consisting of one column) from a set of scalars, and to assign values to its elements. The structure of the MMAKE command is:

```
MMAKE matrix name  list of series;
                or
MMAKE vector name  list of scalars;
```

For example, the following sequence of commands

```
SMPL 1,3;
LOAD A;
2 3 4;
LOAD B;
4 5 6;
MMAKE AMAT A B;
PRINT AMAT;
```

generates the following output

MATRIX AMAT

1	2	4
2	3	5
3	4	6

As a further example,

```
MMAKE COL 10 9.5 8 6 9
```

creates the 5x1 vector COL containing the values 10, 9.5, 8, 6, 9. An alternative way to define and assign values to a matrix is with the LOAD command. An example of 'loading' a matrix is:

```
LOAD(NROW=4,NCOL=3) AMAT;
1.0 1.0 1.0
2.0 2.0 2.0
3.0 3.0 3.0
4.0 4.0 4.0;
```

This creates a 4x3 matrix named AMAT with elements AMAT(i,j) = i for j=1,2,3 and i=1,2,3,4. By default the type of a matrix is GENERAL. The type of a matrix can be explicitly specified with the TYPE option. Other types are SYMETRIC, TRIANG or DIAG. For example,

```
LOAD(NROW=3,NCOL=3,TYPE=DIAG) BAND;
1 2 3;
```

defines a 3x3 diagonal matrix named BAND with BAND(i,i) = i for i=1,2,3

Parameters, constants, matrices (and as a special case vectors) can be printed with the PRINT command. Note, however, that when printing a vector the SMPL command does not govern what part of the vector will be printed. The entire vector will be printed in all cases.

6. BASIC OPERATIONS WITH TIME SERIES, PARAMETERS AND CONSTANTS

6.1 GENR Command

TSP's basic data transformation command is the GENR command. The basic structure of the GENR command is as follows:

```
GENR var = formula;
```

where 'var' stands for variable name. The GENR command creates a new variable and stores values in it according to the formula. For example, the sequence of commands

```
SMPL 1 3;
LOAD X1;
10 20 15;
LOAD X2;
100 400 600;
GENR X3 = 2*X2/X1;
PRINT X1,X2,X3;
```

generates the following output

	X1	X2	X3
1	10	100	20
2	20	400	40
3	15	600	80

The GENR command name can be omitted, i.e. instead of GENR X3 = 2*X2/X1 one can also simply write X3 = 2*X2/X1. The rules for writing formulae are standard. Typical operators are:

+	Add
	Subtract
*	Multiply
/	Divide
	Raise to power

Open and closed parenthesis are denoted by, respectively, (and). Furthermore, not only the names of time series but also the names of constants and parameters can be used in formulae. Also, TSP has various built in functions. For example, the following functions are available:

LOG	Natural logarithm
EXP	Exponential function
ABS	Absolute value
SQRT	Square root

NORM	Standard normal density
CNORM	Standard normal cumulative distribution function

As an illustration, GENR X4 = LOG(X1); would create a new time series X4 whose elements equal the natural logarithm of the respective elements of X1

It is often of interest to calculate the growth rate of a variable, say, GDP. The command

```
GENR GDPG = (GDP - GDP(-1))/GDP(-1)*100;
```

creates a new time series, called GDPG, that contains the growth rates of the times series GDP (in percentage points).

6.2 REPL Command

In the course of economic research it is often necessary to revise data series or to splice different series. In such cases the REPL command is used. REPL is like the GENR command. The only difference in execution is that the variable being revised must have been previously defined and observations which are temporarily deactivated due to the SMPL command are preserved rather than lost. For example, the sequence of commands

```
SMPL 1 3;
LOAD X1;
1 2 3;
LOAD X2;
11 12 13;
SMPL 3 4;
REPL;
GENR X2 = 100;
NOREPL;
GENR X1 = 100;
REPL;
SMPL 1 4;
PRINT X1,X2;
```

generates the following output:

	X1	X2
1	MISSING	11
2	MISSING	12
3	100	100
4	100	100

Note: REPL is the default mode. Hence series are always updated rather than being completely lost when the current sample under which they are being computed does not cover the complete series. It is only when a NOREPL has been executed, that the REPL command assumes significance.

6.3 SET Command

The SET command, rather than the GENR command, is used to perform operations that result in a scalar value. The basic structure of the SET command is as follows:

```
SET var = formula;
```

where 'var' stands for the name of a parameter, a constant, an element of a series or the element of a matrix. In the SET command you refer to, say, the fifth element of the series GNP as GNP(5). (Note, in the GENR command GNP(5) would refer to the series GNP lead by five periods.) Furthermore, you can refer to the, say (1,2)-element of the matrix VMAT as VMAT(1,2). The following examples illustrate the SET command:

```
SMPL 1 5;
LOAD Y;
11 12 13 14 15;
LOAD (NROW=2,NCOL=2) VMAT;
100 200
300 400;
CONST ALPHA 2.0;
PARAM BETA 3.0;
SET GAMMA = ALPHA*BETA;
SET DELTA = VMAT(1,2);
SET RHO = Y(2);
SMPL 3 3;
SET LAMBDA = Y;
SMPL 1 5;
PRINT ALPHA,BETA,GAMMA,DELTA,RHO,LAMBDA;
```

generate the following output

```
ALPHA      2.0
BETA       3.0
GAMMA      6.0
DELTA    200.0
RHO       12.0
LAMBDA    13.0
```

6.4 TREND Command

The **TREND** command can be used to generate a time trend variable. The time trend variable will be one in the first period (as indicated by the SMPL command) and then increases by one every period. For example, the commands

```
FREQ Q;
SMPL 30:1,31:4;
TREND T; PRINT T
```

generate the following output

```
          T

1930Q1    1.0
1930Q2    2.0


1931Q4    8.0
```

By adding the option FREQ, the trend will be restarted every year. I.e., if instead of TREND T; you stated in the above example TREND(FREQ) T; the series T would equal 1,2,3,4,1,2,3,4.

7. BASIC OPERATIONS WITH MATRICES

The MATRIX command is used to evaluate matrix expressions. It is the matrix equivalent of the GENR and SET commands. The basic structure of the MATRIX command is as follows:

```
MATRIX name = formula;
```

where 'name' stands for the name of a constant, parameter, vector, or matrix. For example, let X and Y be matrices of appropriate dimensions, and let S be a constant or parameter, let V be a vector (i.e., a matrix consisting of one column), and let Z be a general matrix on which the result of the matrix operation is to be stored, then

```
MATRIX Z = X*S;          multiplies matrix X with scalar S
MATRIX Z = X*Y;          multiplies matrix X and Y
MATRIX Z = X + Y;        add matrix X and Y
MATRIX Z = X - Y;        subtracts matrix Y from X
MATRIX Z = X';           computes the transpose of X
MATRIX Z = X";           computes the inverse of X
MATRIX S = DET(X);       computes the determinant of X
MATRIX Z = X'X;          computes the cross product of X, i.e.
                         transpose(X)*X
MATRIX S = TR(X);        computes the trace of X
MATRIX Z = X#Y;          computes the Kronecker product
                         between X and Y
MATRIX V = EIGVAL(X);    computes the eigenvalues of X, given
                         X is symmetric, pos. semi-definite
MATRIX Z = EIGVEC(X);    computes the eigenvectors of X, given
                         X is symmetric, pos. semi-definite
MATRIX Z = IDENT(m)      creates a mxm identity matrix
MATRIX Z = DIAG(X)       creates a matrix of type DIAG from
                         the diagonal elements of X
```

MATRIX Z = DIAG(V)	creates a matrix of type DIAG from the elements of the vector V
MATRIX Z = GEN(X)	creates a matrix of type GENERAL from a matrix of type SYM or DIAG
MATRIX Z = SYM(X)	creates a matrix of type SYM from a matrix of type DIAG or GENERAL (using the lower triangular elements)

The MATRIX command may be abbreviated to MAT. (There is a bug in TSP in that the command MATRIX Y = X does not work. TSP requires some oparation on the right hand side. For example, the command MATRIX Y = 1.0*X works.)

8. EQUATIONS

The command FRML allows the user to name and define a formula. The arguments in the formula may be both time series variables, constants (or parameters) and matrices. The formula can be any legal TSP transformation as discussed above within the context of the GENR and MATRIX command. The FRML command allows for the definition of either implicit or explicit equations. The corresponding structure of the FRML command is:

```
FRML equ form;
    or
FRML equ var = form;
```

where 'equ' stands for the name of the equation, 'var' stands for the name of a variable and 'form' stands for some formula. For example, let Y, K, L, and T have the interpretation of time series variables containing data on output, capital, labor and time. Then

```
FRML COB Y = A0*(K**ALPHA)*(L**BETA)*EXP(LAMBDA*T);
```

defines the formula for a Cobb-Douglas production function. A0, ALPHA, BETA and LAMBDA denote the parameters of the production function. COB is the name of the equation.

FRMLs can be executed with the GENR, SET or MATRIX commands. For example, the commands

```
SMPL 1 3;
LOAD A;
1 2 3;
FRML EB B = A + 10;
GENR EB;
GENR EB S;
PRINT A B S;
```

generate the following output:

	A	B	S
1	1	11	11
2	2	12	12
3	3	13	13

The IDENT command performs the same function as the FRML command, except that it indicates that the equation being specified is an identity rather than a stochastic relationship. For example

```
IDENT NIPA GDP = CONP + CONG + INV + EXP - IMP;
```

defines the identity NIPA The equation is assumed to hold exactly

Execution time can be reduced by simplifying the formulae as much as possible, and by eliminating repeated terms. Repeated terms are best handled with the EQSUB command, which substitutes on equation into another and has to the effect that the repeated term will be evaluated only once. The basic structure of the EQSUB command is as follows:

EQSUB(PRINT,NAME=*equ-out* *equ-in*,*equ-m*1, .,*equ-ms*;

where '*equ-out*' stands for the name of the output equation, '*equ-in*' stands for the name of the input equation and '*equ-m*1',..., *equ-ms* stand for the names of the macro equations. The macro equation(s) and the input equation are defined with FRML or IDENT statements. For example,

```
FRML EQ1 Y = A + XB;
FRML XB X1*B1 + X2*B2;
EQSUB(PRINT,NAME=EQ2) EQ1 XB;
```

results in the output

```
FRML EQ2 Y = A + X1*B1 + X2*B2
```

(If the NAME=*equ-out* option is not present, the input equation is replaced. In the above example, excluding the NAME option would have resulted in the output FRML EQ1 Y = A + X1*B1 + X2*B2).

9. ESTIMATION TECHNIQUES

9.1 Ordinary Least Squares

The TSP command for ordinary least squares (OLS) is the OLSQ command. The structure of the OLSQ command is as follows:

```
OLSQ depvar,indepvar-1,indepvar-2,..  ,indepvar-k;
```

where 'depvar' stands for the name of the dependent variable and 'indepvar-i' stands for the name of the i-th independent variable. TSP does not include an intercept term. If you wish to have an intercept term in the regression, include the special variable C or CONSTANT in your list of independent variables. For example,

```
SMPL 50,88;
OLSQ CON C,INCOME,CON(-1);
```

provides OLS estimates from regressing the variable CON against an intercept, the variable INCOME and the first order lag of CON. The sample period for the regression is 1950 to 1988. White's heteroscedasticity robust estimates of the variance covariance matrix can be obtained by using the option ROBUSTSE. For example,

```
OLSQ (ROBUSTSE) CON C,INCOME,CON(-1);
```

Output from the regression can be further controlled by entering the command REGOPT with customized options before the OLSQ command. Examples for the use of the REGOPT command are:

```
REGOPT ALL;
```

which provides the maximal output (on all statistics that can be computed without additional user input) including residual plots and the variance covariance matrix of the estimated parameters. Of course, to be able to compute some statistics the user has to provide additional information. For example, the Breusch-Pagan heteroscedasticity test requires a list of variables which the user hypothesizes affect the residual variances. This list of variables, say *var1* to *varl*, can be specified as an option of the REGOPT command as follows:

```
REGOPT (BPLIST=(var1,var2,...,varl   ALL;
```

Please consult the Reference Manual, pp. 298-307, for details on other options and a description of how the various test statistics are computed. The command

```
REGOPT ;
```

restores the output to the default values

Nonlinear Least Squares

The LSQ command can be used in connection with the FRML command to calculate nonlinear least squares estimates. In its simplest form, the structure of the LSQ command is as follows:

```
LSQ equ;
```

'equ' stands for the equation name. Suppose the user sets the maximum

number of iterations with respect to the parameter estimates to 100 and the convergence criterion for the parameter estimates to 0.001. The structure of the LSQ command is then as follows:

```
LSQ (MAXIT=100,TOL=0.001  equ;
```

Clearly, the equation must be defined beforehand. Also, the PARAM command must be used beforehand to tell TSP which arguments in the equation are the parameters to be estimated and to assign initial guesses to those parameters. For example, the Cobb-Douglas production function used to illustrate the FRML command above can be estimated as follows:

```
FRML COB Y = A0*(K**ALPHA)*(L**BETA)*EXP(LAMBDA*T);
PARAM A0 1 ALPHA 0.5 BETA 0.5 LAMBDA 0.03;
LSQ COB;
```

The initial parameter values are overwritten with the actual parameter estimates. Of course, the above form of the LSQ command can also be used to estimate linear equations, i.e., ordinary least squares (or OLSQ) is a special case of nonlinear least squares (or LSQ).

9.3 Seemingly Unrelated Regression

Zellner's seemingly unrelated regression techniques was developed for systems of equation where all explanatory variables are exogenous. The seemingly unrelated regression estimator is a GLS estimator applied to a system of equations. The TSP command for the seemingly unrelated regression estimator is SUR. (It is programmed as a special case of the general LSQ command.) The SUR command can be used to estimate linear and nonlinear systems of equations. The same parameter may be contained in more than one equation. Suppose the user sets the maximum number of iterations with respect to the parameter estimates to 100 and the convergence criterion for the parameter estimates to 0.001. The structure of the SUR command is then as follows:

```
SUR (MAXIT=100,TOL=0.001  equ-1,equ-2,     ,equ-g;
```

where 'equ-i' stands for the name of the i-th equation. By using the option ITERU the SUR estimator will also iterate on the residual variance covariance matrix. The default is NOITERU. Upon convergence the seemingly unrelated regression estimator is numerically equivalent to the full information maximum likelihood estimator, given iterations are also performed over the residual variance covariance matrix.

NOTE The last statement remains correct, even if some of the regressors are endogenous variables, as long as the system is triangular. However, if some of the regressors are current endogenous variables, then the estimates of the variance covariance matrix printed by the SUR routine will in general not be consistent; cf. Prucha (1987) for details.

Prucha, I.R., The Variance-Covariance Matrix of the Full Information Maximum

Likelihood Estimator in Triangular Structural Systems Consistent Estimation, *Econometrica*, 1987.

Two Stage Least Squares

The two stage least squares (2SLS) procedure can be used to estimate a single equation of a system of equations consistently. (The 2SLS command is programmed as a special case of the general LSQ command.) In the following we distinguish between endogenous variables and exogenous variables, or more precisely, predetermined variables. To apply the 2SLS procedure we have to tell TSP which variables are predetermined. The list of predetermined variables is specified with the INST option. Suppose the user sets the maximum number of iterations with respect to the parameter estimates to 100 and the convergence criterion for the parameter estimates to 0.001. The structure of the 2SLS command is then as follows:

```
2SLS (INST=(inst-1,...,inst-s),MAXIT=100,TOL=0.001) equ;
```

where 'inst-j' stands for the j-th instrument and 'equ' stands for the name of the equation.

As an illustration, suppose the 2SLS procedure is to be applied to the equation

$$Y2 = \alpha + \beta*Y1 + \gamma*Z3 + \delta*Z7 + u$$

where Y1 and Y2 are endogenous variables; Z1, ..., Z8 are the predetermined variables in the system; α, β, γ, δ are the parameters to be estimated and u is the disturbance term. 2SLS estimates can then be obtained by submitting the following commands:

```
2SLS  INST=(Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8)) Y2 C,Y1,Z3,Z7;
```

It is sometimes convenient to first define the set of instruments as a list. The 2SLS estimates can then be obtained by submitting the following commands

```
LIST IVARS Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8;
2SLS (INST=IVARS) Y2 C,Y1,Z3,Z7;
```

Analogously to the LSQ command, an alternative form of the 2SLS command that allows for nonlinear two stage least squares is:

```
LSQ (INST=IVARS) equ;
```

where 'equ' stands for the name of the equation to be estimated by 2SLS.

Three Stage Least Squares

The three stage least squares (3SLS) procedure can be used to jointly estimate systems of equations consistently. (The 3SLS command is programmed

as a special case of the general LSQ command.) As with the 2SLS estimator we need to distinguish between endogenous variables and exogenous variables, or more precisely, predetermined variables. To apply the 3SLS procedure we have to tell TSP which variables are predetermined in the entire system. The list of predetermined variables is specified with the INST option. Suppose the user sets the maximum number of iterations with respect to the parameter estimates to 100 and the convergence criterion for the parameter estimates to 0.001. The structure of the 3SLS command is then as follows:

```
3SLS (INST=(inst-1,...,inst-s),MAXIT=100,TOL=0.001)
equ-1, equ-2, ...., equ-g;
```

where 'inst-j' stands for the j-th instrument and 'equ-i' stands for the name of the i-th equation. (As with the 2SLS command it is possible to first define a list for the set of instruments.)

By using the option ITERU the 3SLS estimator will also iterate on the residual variance covariance matrix. The default is NOITERU.

As an example, consider Klein's Model I as given in Theil (1971), *Principles of Econometrics*, pp. 432. Let CN, P, W, I, K and X denote, respectively, the endogenous variables corresponding to consumption, profits, wage bill, net investment, capital stock, and total production. Let WG, T and G denote the exogenous variables corresponding to the government wage bill, taxes and government nonwage expenditures. The following sequence of commands illustrates how you can estimate Klein's Model I by 3SLS:

```
FRML CNEQ CN = A0 + A1*P + A2*P(-1) + A3*(W+WG);
FRML IEQ I = B0 + B1*P + B2*P(-1) + B3*K(-1);
FRML WEQ W = C0 + C1*X + C2*X(-1) + C3*TIME;
PARAM A0 18.0 A1 0.80 A2 -0.20 A3 0.30;
PARAM B0 27.0 B1 -0.80 B2 1.00 B3 -0.15;
PARAM C0 5.00 C1 0.20 C2 0.30 C3 0.20;
3SLS (INST=(P(-1),K(-1),X(-1),TIME,T,WG,G),MAXIT=100,TOL=0.001)
CNEQ,IEQ,WEQ;
```

9.6 Full Information Maximum Likelihood

The full information maximum likelihood (FIML) estimator can alternatively be used to jointly estimate systems of equations. As with the 3SLS estimator we need to distinguish between endogenous variables and exogenous variables, or more precisely, predetermined variables. To apply the FIML procedure you have to specify the entire system including all identities and you have to tell TSP which variables are endogenous in the entire system. The list of endogenous variables is specified with the ENDOG option. The basic structure of the FIML command is analogous to that of the 3SLS command. Suppose the user sets the maximum number of iterations with respect to the parameter estimates to 100 and the convergence criterion for the parameter estimates to 0.001. The structure of the FIML command is then as follows:

```
FIML (ENDOG=(end-1,...,end-g),MAXIT=100,TOL=0.001)
equ-1,equ-2, ....,equ-g;
```

where 'end-i' and 'equ-i' stands for the name of the i-th endogenous variable and equation.

As an example, consider again Klein's Model I as given in Theil (1971), *Principles of Econometrics*, pp. 432. Let CN, P, W, I, K and X denote, respectively, the endogenous variables corresponding to consumption, profits, wage bill, net investment, capital stock, and total production. Let WG, T and G denote the exogenous variables corresponding to the government wage bill, taxes and government nonwage expenditures. The following sequence of commands illustrates how you can estimate Klein's Model I by FIML:

```
FRML CNEQ CN = A0 + A1*P + A2*P(-1) + A3*(W+WG);
FRML IEQ I = B0 + B1*P + B2*P(-1) + B3*K(-1);
FRML WEQ W = C0 + C1*X + C2*X(-1) + C3*TIME;
IDENT XEQ X = CN + I + G;
IDENT PEQ P = X - W - T;
IDENT KEQ K = K(-1) + I;
PARAM A0 18.0 A1 0.80 A2 -0.20 A3 0.30;
PARAM B0 27.0 B1 -0.80 B2 1.00 B3 -0.15;
PARAM C0 5.00 C1 0.20 C2 0.30 C3 0.20;
FIML (ENDOG=(C,I,W,X,P,K),MAXIT=100,TOL=0.001) CNEQ,IEQ,WEQ,XEQ,PEQ,KEQ;
```

9.7 Generalized Method of Moments Estimation

The GMM command permits the computation of the GMM (Generalized Method of Moments) estimator for the parameters of systems of equations. (The GMM command is programmed as a special case of the general LSQ command.) To be able to explain the meaning of the various options in an unambiguous way we provide a brief review of the relevant estimation theory in the Appendix. Suppose the user sets the maximum number of iterations with respect to the parameter estimates to 100 and the convergence criterion for the parameter estimates to 0.001. Basic structures of the GMM command are then as follows:

```
GMM (INST=(inst-1,...,inst-p),MAXIT=100,TOL=0.001)
equ-1, equ-2, ...., equ-g;
```

or

```
GMM (INST=(inst-1,...,inst-p),HETERO,NMA=maxlag,MAXIT=100,TOL=0.001)
equ-1, equ-2, ...., equ-g;
```

where 'inst-j' stands for the j-th instrument and 'equ-i' stands for the name of the i-th equation. (As with the 2SLS command it is possible to first define a list for the set of instruments.)

Both GMM commands optimize formula (2) in the Appendix with $P(\hat{\tau}) = \hat{\Psi}^{-1}$, where $n^{-1}\hat{\Psi}$ can be interpreted as an estimator for the variance covariance matrix of orthogonality conditions. The first command computes the GMM estimator using formula (9) in the Appendix for $\hat{\Psi}$. By default the program estimates the residual covariance matrix Σ from 3SLS residuals. Thus the first command yields the 3SLS estimator with one iteration on the residual variance covariance matrix. If the ITERU option is added, then the procedure will continue iterating on the residual variance covariance matrix.

The second command computes the GMM estimator using formula (5) in the Appendix for $\hat{\Psi}$, where all weights are taken to equal zero for lags greater than 'maxlag'. That is, 'maxlag' specifies the number of autocorrelation terms that are to be used in computing the variance covariance matrix of the orthogonality conditions. By default the weight are computed from the Bartlett kernel. (If the option KERNEL=PARZEN is used the weight are computed from the Parzen kernel.) The estimator for the variance covariance matrix of the orthogonality conditions is by default based on 3SLS residuals. If the ITEROC option is added the procedure iterates on this variance covariance matrix of the orthogonality conditions.

As an illustration consider e.g. the following set of orthogonality conditions implied by a (greatly simplified) version of dynamic factor demand model considered in Prucha and Nadiri (1986):

$$E_t\left[-AKKD*K_{t+1} + (AKK+(2+I_t)*AKKD)*K_t - (1+I_t)*AKKD*K_{t-1} + AK + AKY*Y_t + UK_t\right]Z_{tj} = 0\ ,$$

$$E_t\left[-ARRD*R_{t+1} + (ARR+(2+I_t)*ARRD)*R_t - (1+I_t)*ARRD*R_{t-1} + AR + ARY*Y_t + UR_t\right]Z_{tj} = 0\ , \quad j=1,\ldots,p,$$

where K and R denote the end of period stocks of capital and R&D, respectively, I is the interest rate, Y denotes output, AND UK and UR denote the rental price of capital and R&D, respectively. The instruments Z are assumed to be in the firm's information set in period t. The following sequence of commands illustrates how we would estimate the parameters by GMM, allowing for heteroscedasticity and temporal dependence (choosing NMA=2):

```
LIST IVLIST Z1,Z2,Z3,Z4,Z5,...,ZP;
FRML U1EQ -AKKD*K(+1) + (AKK+(2+I)*AKKD)*K - (1+I)*AKKD*K(-1)
              + AK + AKY*Y + UK;
FRML U2EQ -ARRD*R(+1) + (ARR+(2+I)*ARRD)*R - (1+I)*ARRD*R(-1)
              + AR + ARY*Y + UR;
PARAM AK -0.2 AR -0.1 AKK 0.4 ARR 0.3 AKKD 0.7 ARRD 5.0;
PARAM ARY -0.5 AKY 0.15 ;
GMM(INST=IVLIST,HETERO,NMA=2,MAXIT=100,TOL=0.001) U1EQ,U2EQ;
```

Prucha, I.R., and M.I. Nadiri, A Comparison of Alternative Methods for the Estimation of Dynamic Factor Demand Models under Non-Static Expectations, *Journal of Econometrics*, 1986.

9.8 Tobit Estimation

Consider the following model for y_t (t=1,...,n):

$$\underset{1x1}{y_t^*} = \underset{1xK}{x_{t.}}\underset{Kx1}{\beta} + \underset{1x1}{u_t} ,$$

$$y_t = y_t^* \quad \text{if } y_t^*>0 ,$$

$$y_t = 0 \quad \text{if } y_t^*\leq 0 ,$$

where y^* is unobserved if $y^*\leq 0$, the regressors are observed always, and the disturbances are i.i.d. normal with mean zero and constant variance. This model was introduced by Tobin. It represents a special case of a censored model (as opposed to a truncated model for which also the regressors would be unobserved if $y^*\leq 0$). Amemiya refers to this model as the Standard Tobit Model (or Type 1 Tobit Model). The maximum likelihood estimator for the parameters of the above model can be computed with the TOBIT command. The basic structure of the TOBIT command is analogous to that of the OLSQ command:

```
TOBIT depvar,indepvar-1,indepvar-2,     ,indepvar-k;
```

where 'depvar' stands for the name of the dependent variable and 'indepvar-i' stands for the name of the i-th independent variable. (To estimate a model where y^* is unobserved if it lies above and below threshold values see the SAMSEL command.)

9.9 Binary Probit and Logit Estimation

Consider the following (univariate) binary qualitative response model for y (i=1,...,n)::

$$\underset{1x1}{P(y_i=1)} = F(\underset{1xK}{x_{i.}}\underset{Kx1}{\beta})$$

$$\underset{1x1}{P(y_i=0)} = 1-F(\underset{1xK}{x_{i.}}\underset{Kx1}{\beta})$$

The Probit model corresponds to the case where $F(z) = \Phi(z)$ and $\Phi(.)$ is the cumulative distribution function of a standardized normal random variable. The Logit model corresponds to the case where $F(z) = \Lambda(z)$ and $\Lambda(z) = \exp(z)/[1+\exp(z)]$ represents the cumulative distribution function of the logistic distribution.

The TSP commands to estimate a binary Probit or Logit model are, respectively, PROBIT and LOGIT. The structure of the PROBIT and LOGIT command is analogous to that of the OLSQ command:

```
PROBIT depvar,indepvar-1,indepvar-2,  ..,indepvar-k;
LOGIT  depvar,indepvar-1,indepvar-2,  ..,indepvar-k;
```

where 'depvar' stands for the name of the dependent variable and 'indepvar-i' stands for the name of the i-th independent variable.

9.10 Multinomial and Conditional Logit Estimation

Consider the following (univariate) multinomial qualitative response model for y (i=1,...,n, and j=1,...,m)::

$$P(y_i=j) = \frac{\exp(z_{i.,j}\alpha + x_{i.}\beta_j)}{\sum_{s=1}^{m}\exp(z_{i.,s}\alpha + x_{i.}\beta_s}$$

The above model typically corresponds to an underlying model that involves latent value (utility) equations for each choice and where the disturbances in those equations follow a Generalized Extreme Value distribution. As a normalization rule β_1 is set equal to zero. For α=0 the above model is typically referred to as a multinomial logit model. For the multinomial logit model the data are chooser specific and the parameters choice specific. The variables $x_{i.}$ are typically called the multinomial variables. For β_j=0 the above model is typically referred to as the conditional logit model. For the conditional logit model the data are choice specific and the parameters are the same for all choices. The variables $z_{i.,j}$ are typically called the conditional variables. (The conditional logit model may have a variable number of choices per chooser. Furthermore, the choices need not be labeled as 1,2,...m, but can be any set of distinct integers. Note also that the above model contains the binary Logit model as a special case with m=2, β_1=0, β_2=β and α=0.)

The TSP command to estimate Logit models is the LOGIT command. The structure of the LOGIT command is analogous to that of the OLSQ command. The LOGIT command to estimate a multinomial Logit model with m choices is of the form:

```
LOGIT (NCHOICE=m) depvar,multvar-1,multvar-2,    ,multvar-k;
```

where 'depvar' stands for the name of the dependent variable and 'multvar-r stands for the name of the r-th multinomial variable. The names of the parameters are determined by appending the values of the dependent variable for each choice to the name of the explanatory variable. For example,

```
LOGIT (NCHOICE=3) Y,X1,X2,X3,X4;
```

reports estimates for parameters labeled as X11,X21,X31,X41 for the first choice and X12,X22,X32,X42 for the second choice. (The parameters for the

first choice are normalized to zero.)

The LOGIT command to estimate a conditional Logit model with m choices is of the form:

```
LOGIT (COND,NCHOICE=m) depvar,condvar-1,condvar-2,....,condvar-1;
```

where 'depvar' stands for the name of the dependent variable and 'condvar-i stands for the name of the i-th conditional variable. For example,

```
LOGIT (COND,NCHOICE=2) Y,Z1,Z2,Z3 ;
```

looks for variables Z11, Z21 and Z31 corresponding to the first choice, and for variables Z12, Z22 and Z32 corresponding to the second choice, and reports estimates for the corresponding parameters labeled as Z1, Z2 and Z3.

The LOGIT command to estimate a mixed Logit model with m choices is of the form:

```
LOGIT (COND,NCHOICE=m) depvar,
condvar-1,condvar-2,....,condvar-1   multvar-1,multvar-2,....,multvar-k;
```

where 'depvar' stands for the name of the dependent variable, 'condvar-i' stands for the name of the i-th conditional variable, and 'multvar-r' stands for the name of the r-th multinomial variable.

9.11 General M(aximum) Estimation

The ML command can be used to compute the maximum likelihood estimator from general log-likelihood functions, as long as the log-likelihood function can be written in the form

$$\sum_{t=1}^{n} \ell(z_t, \beta) \ ,$$

where

$$\ell(z_t, \beta)$$

represents the log-likelihood function for period t, z is the data vector, β is the unknown parameter vector and n is the sample size. (Actually, the ML command can be used not only to compute maximum likelihood estimators but M-estimators in general. For ease of presentation we continue to interpret $\ell(.)$ as a log-likelihood function.) To apply the ML command the user must first define the log-likelihood function for period t via a FRML statement, using LOGL as the name of the left hand side variable in the equation. The PARAM command can be used to specify which of the arguments in the LOGL equation are the parameters to be estimated and also to assign starting values to those parameters. The ML command then computes the maximum likelihood estimator by maximizing the sum of the log-likelihood functions for the respective periods

in the sample. The ML command is quite powerful, because TSP can compute analytic first and second derivatives for iteration and standard errors. It can, e.g., be used to estimate a variety of censored, truncated and qualitative response models.

Suppose the user sets the maximum number of iterations with respect to the parameter estimates to 100 and the convergence criterion for the parameter estimates to 0.001. The structure of the ML command is then as follows:

```
ML (MAXIT=100,TOL=0.001) equ-name ;
```

where 'equ-name' stands for the name of the equation that defines the log-likelihood function for one period. As an illustration, the following commands would compute the ordinary least squares estimates for, say, the parameters A and B of the consumption function CP = A + B*YD:

```
FRML EQL LOGL = (CP - A - B*YD)**2;
PARAM A B;
ML EQL;
```

The automatic differentiation in ML is a great advantage over coding the derivatives by hand in a FORTRAN subroutine. The disadvantage however, is that execution time is slower and numerical error handling more difficult, so care should be taken when writing the log-likelihood function to minimize these problems.

9.12 Panel Data Estimation

The full text for this section will be supplied later. Please consult the TSP User's Guide and the TSP Reference Manual for a description of the PANEL command.

The PANEL command allows for the estimation of a single equation of the form

$$y_{ti} = \alpha + \mu_i + \sum_{l=1}^{k} x_{til}\beta_l + u_{ti},$$

where t=1,...,T may refer to the time period and i=1,...,N may refer to the individual. The disturbances u_{ti} are assumed to be i.i.d. If the μ_i are treated as fixed parameters, then the model is called a fixed effect or dummy variable model. If the μ_i are treated as i.i.d random variables, then the model is called an error component model.

Estimators for the parameters of seemingly unrelated and simultaneous equation error component and dummy variable models that allow for both components that vary over individuals and components that vary over time are, e.g., developed in Prucha (1984, 1985). Specification issues for models where also the slope parameters can vary over individuals and time see, e.g., Kelejian (1991a,b).

Prucha, I.R., On the Asymptotic Efficiency of Feasible Aitken Estimators for Seemingly Unrelated Regression Models with Error Components, *Econometrica*, 1984.

Prucha, I.R., Maximum Likelihood and Instrumental Variable Estimation in Simultaneous Equation Systems with Error Components, *International Economic Review*, 1985

Gatto, J.P., Kelejian, H.H., and S.W. Scott, A Note Concerning Specifications of Interactive Random-Coefficient Regression Models, *Journal of Econometrics*, 1991a.

Gatto, J.P., Kelejian, H.H., and S.W. Scott, A Random Coefficient Qualitative Choice Model of Telecommunications Demand, *Economics Letters*, 1991b.

9.13 Retrieval of Internal Estimation Results

It is sometimes of interest to retrieve some of the statistics computed by an estimation routine to, e.g., compute further statistics. For example, the OLSQ command stores the estimated parameters in a vector @COEF, the estimated variance covariance matrix in a symmetric matrix @VCOV, the value of the log-likelihood in the scalar @LOGL, the estimated residuals and the fitted values in the series @RES and @FIT, etc. Those internal statistics can, e.g., be retrieved with the GENR, SET and MAT command. For example,

```
MATRIX BVEC = GEN(@COEF);
MATRIX VC = GEN(@VCOV);
SET LNL = @LOGL;
GENR UHAT = @RES;
GENR YHAT = @FIT;
```

stores the estimated coefficient in a vector BVEC, the variance covariance matrix in a matrix VC, the value of the log-likelihood function in a scalar LNL, and the estimated residuals and fitted values in the series UHAT and YHAT.

Alternatively internal statistics can be retrieved with the COPY command The general structure of the COPY command is:

```
COPY old-TSP-variable new-TSP-variable
```

Please consult the TSP User's Guide and the TSP Reference Manual for a listing of the respective names of internal statistics. You can also obtain a list of all active series, scalars and matrices by issuing the following SHOW commands:

```
SHOW SERIES;
SHOW SCALAR;
SHOW MATRIX;
```

10. MODEL SIMULATION

In order to simulate a model the user must collect respective equations into a model by using the MODEL procedure. The structure of the MODEL command is as follows:

```
MODEL group-1 group-2 model ;
```

where 'group-1' stands for the name of the list of equation names, 'group-2' stands for the list of endogenous variables and 'model' stands for the name of the model. The lists are established with the LIST command:

```
LIST group-j name-1 name-2 .
```

where 'group-j' stands for the name of the list for the j-th group j=1,2) and 'name-i' stands for the i-th name in the list.

As an example, consider again Klein's Model I as given in Theil, *Principles of Econometrics*, pp. 432. Let CN, P, W, I, K and X denote, respectively, the endogenous variables corresponding to consumption, profits, wage bill, net investment, capital stock, and total production. Let WG, T and G denote the exogenous variables corresponding to the government wage bill, taxes and government nonwage expenditures. The model can be built by the following sequence of commands (where, for completeness, we also assign values to respective parameters):

```
PARAM A0 18.341 A1 0.80183 A2 -0.23 A3 0.38447;
PARAM B0 27.268 B1 -0.80067 B2 1.0517 B3 -0.14811;
PARAM C0 5.7939 C1 0.23415 C2 0.28465 C3 0.23483;
FRML CNEQ CN = A0 + A1*P + A2*P(-1) + A3*(W+WG);
FRML IEQ I = B0 + B1*P + B2*P(-1) + B3*K(-1);
FRML WEQ W = C0 + C1*X + C2*X(-1) + C3*TIME;
IDENT XEQ X = C + I + G;
IDENT PEQ P = X - W - T;
IDENT KEQ K = K(-1) + I;
LIST EQUKLEIN CNEQ IEQ WEQ XEQ PEQ KEQ;
LIST ENDKLEIN CN I W X P K;
MODEL EQUKLEIN,ENDKLEIN,KLEIN;
```

The SOLVE command simulates the model over the period specified by the SMPL command. The structure of the SOLVE command in its simplest form is:

```
SOLVE model;
```

where 'model' stands for the model name. Two options are available to specify convergence criteria. Convergence occurs when either the proportional change or the absolute change in every variable is less than a given tolerance. This tolerance is specified with the CONV1 option. The CONV2 option can be used to specify a second convergence criterion, which applies to the sum of squared residuals after the variables have passed the CONV1 test. The user can also specify the maximum number of solution iterations with the MAXIT option. For

example, the command

```
SOLVE (CONV1=0.001,CONV2=0.001,MAXIT=100) model;
```

sets both tolerance levels for convergence to 0.001 and the maximum number of iterations to 100.

The default solution algorithm is the Gauss-Seidel method. Other available solution algorithms are Jacobi and Fletcher-Powell. By default TSP solves a model dynamically, i.e., earlier solved values for lagged endogenous variables are used in place of actual values. If the option STATIC is used, then TSP computes a static solution, i.e., the actual values of lagged endogenous variables are used.

Prior to the simulation appropriate values must be stored in the exogenous variables for every period over which the model is to be simulated It is prudent to do the same for the endogenous variables. By default the SOLVE command overwrites the values of the endogenous variables during simulation. To avoid that the original endogenous variables are being overwritten you can use the TAG option. E.g. the command

```
SOLVE  TAG=A,CONV1=0.001,CONV2=0.001,MAXIT=100) model;
```

simulates the model and stores the solution for the endogenous variables in new series for the endogenous variables. The names of the new series equal those of the old series, but with an A appended; e.g., W becomes WA. The following command simulates the model KLEIN created above:

```
SOLVE  TAG=A,CONV1 = 0.001,CONV2 = 0.001  KLEIN;
```

11. RANDOM NUMBER GENERATORS

TSP allows you to generate random numbers that follow a normal, Poisson, uniform or an empirical distribution; see TSP Reference Manual, pp. 292. The random number generator is invoked by the RANDOM command. For example, the following command generates a series of observations from independent normal random variables with mean 5 and variance 100 and stores the respective observations in a series E:

```
RANDOM (MEAN=5,STDEV=10) E;
```

Please consult the TSP User's Guide and the TSP Reference Manual on how to simulate the other distributions.

Note: TSP randomizes the seed to start every run based on the current time, so you need to specify a fixed seed (using SEEDIN and SEEDOUT options) if you want to reproduce results from run to run.

12. DO LOOPS, IF AND GOTO STATEMENTS

In many cases, you may want to execute a group of TSP statements several times, using a variety of parameter values of making some other changes each time. The simplest way of doing this is via the DO loop. The structure of the DO command is as follows:

DO *index-name* = *start-value* TO *end-value* [BY *increment*];

DO loops can be nested with other DO loops but each loop must be terminated by an ENDDO statement. TSP executes the statements between the DO and ENDDO statement repetitively as many times as specified by the index or counter variable which is set equal to the start-value the first time through, and is changed each time through by the increment (which is one by default) until the end value has been reached or exceeded. Note that this test is done at the end of the loop, so the program always goes through once.

For example, suppose the researcher wanted to conduct a simple Monte Carlo experiment of estimating the parameters of the two variable regression model

$$Y_t = a + bX_t + u_t \ , \qquad t = 1,\ldots,30$$

where X_t=t is a simple time trend. For purposes of the experiment let us assume that the error terms are distributed i.i.d. normal with a mean of zero and variance of 2 and let the Y_t's be generated according to the above equation with a = 10 and b = 1.0. Suppose there were 100 trials. The following commands would simulate the above described Monte Carlo experiment:

```
SMPL 1,30;
? Generate time trend
TREND X;
? Define vector for Monte Carlo means
LOAD (NROW=2,NCOL=1) BVECM;
0.0 0.0;
?
?
? Perform the experiment 100 times
DO I = 1 TO 100 BY 1;
?
?   Generate the U and Y series
    RANDOM (MEAN=0,STDEV=2) U;
    GENR Y = 10 + 1.0*X + U
?   Perform OLS estimation
    OLSQ Y,C,X;
?   Retrieve parameter vector
    MATRIX BVEC = GEN(@COEF);
```

```
?   Compute Monte Carlo mean
    MATRIX BVECM = BVECM + BVEC/100;
?
ENDDO;
?
? Print Monte Carlo mean
PRINT BVECM;
```

APPENDIX: BACKGROUND THEORY ON GENERALIZED METHOD OF MOMENTS ESTIMATION

In the following we provide a brief review of the asymptotic theory for GMM (generalized method of moments) estimators. For more detailed discussions see, e.g., Hansen (1982), Gallant (1987), Gallant and White (1988) and Pötscher and Prucha (1991a,b). We thought that such a review may be helpful for a proper understanding of the GMM command and its options in TSP.

Suppose that under the given model assumptions certain population moments are known to be zero. It then seems intuitively reasonable to define parameter estimators such that the corresponding sample moments are also equal to zero. This is typically possible in case the number of moments equals the number of parameters. This estimation approach is called the method of moments approach. In case the number of moments exceeds the number of parameters it is generally not possible to set all sample moments equal to zero. In this case we may then try to choose our parameter estimates such that they minimize some weighted average of the sample moments or, more generally, some function of the sample moments. Estimators of this kind are called generalized method of moments estimators. Somewhat more formally, using the notation in Pötscher and Prucha (1991a,b), let

$$n^{-1}\sum_{t=1}^{n} q_t(\mathcal{Z}_t, \bar{\tau}_n, \bar{\beta}_n$$

be the vector of sample moments with corresponding population moments equal to zero, let $\mathcal{Z}_t$ denote the data vector, let $\bar{\tau}_n$ be some vector of nuisance parameters and let $\bar{\beta}_n$ be the vector of the "true" parameters of interest (which in general may depend on the sample size). The data vector $\mathcal{Z}_t$ may include current endogenous variables, future endogenous variables, lagged endogenous variables, exogenous variables and outside instruments. Then any estimator $\hat{\beta}_n$ for $\bar{\beta}_n$ that maximizes an objective function of the form

$$(1 \quad Q_n(\mathcal{Z}_1, \ldots, \mathcal{Z}_n, \hat{\tau}_n, \beta) = \vartheta_n\left[n^{-1}\sum_{t=1}^{n} q_t(\mathcal{Z}_t, \hat{\tau}_n, \beta), \hat{\tau}_n, \beta\right]$$

where $\vartheta_n(.)$ is a real valued function and $\hat{\tau}_n$ is an estimator for $\bar{\tau}_n$, is called a generalized method of moments estimator. The class of GMM estimators

programmed in TSP corresponds to objective functions of the form

$$(2)\quad Q_n(z_1,\ldots,z_n,\hat{\tau}_n,\beta) = \left[n^{-1}\sum_{t=1}^{n} q_t(z_t,\beta)\right]' P(\hat{\tau}_n)\left[n^{-1}\sum_{t=1}^{n} q_t(z_t,\beta)\right],$$

i.e., $\vartheta_n(c,\tau,\beta) = c'P(\tau)c$, where P is some symmetric weighting matrix. The vector $q_t(.)$ does not depend on nuisance parameters and is of the form

$$q_t(z_t,\bar{\beta}_n) = u_t \otimes a_t$$

with u_t representing the gx1 the vector of residuals from the model equations in period t and a_t representing the px1 vector of instruments in period t. Let u be the ngx1 vector of residuals stacked equation by equation and let A be the nxp matrix of instruments, then

$$n^{-1}\sum_{t=1}^{n} q_t(z_t,\bar{\beta}_n) = n^{-1}\sum_{t=1}^{n} u_t \otimes a_t = (I \otimes A')u$$

As an illustration, suppose the model under consideration consists of a system of g equations,

$$u_t = f(y_t,\ldots,y_{t-p},x_t,\bar{\beta}_n)$$

where y_t and x_t denote the vectors of endogenous and exogenous variables in the system and u_t is i.i.d. with zero mean and variance covariance matrix Σ and $\bar{\beta}_n \equiv \bar{\beta}$. Let $\tilde{\Sigma}_n$ be the estimator for the residual variance covariance matrix Σ based on 2SLS residuals. The objective function for the 3SLS estimator is then given by

$$n^{-1}u'\left[\tilde{\Sigma}_n \otimes A(A'A)^{-1}A'\right]u = n^{-1}u'(I\otimes A)[\tilde{\Sigma}_n \otimes (A'A)]^{-1}(I\otimes A')u =$$
$$\left[n^{-1}\sum_{t=1}^{n} q_t(z_t,\beta)\right]'\left[\tilde{\Sigma}_n \otimes n^{-1}\sum_{t=1}^{n} a_t a_t'\right]^{-1}\left[n^{-1}\sum_{t=1}^{n} q_t(z_t,\beta)\right] .$$

That is, the 3SLS estimator is a special case of a GMM estimators defined by (2) with

$$P(\hat{\tau}) = \left[\tilde{\Sigma}_n \otimes n^{-1}\sum_{t=1}^{n} a_t a_t'\right]^{-1}$$

Furthermore, for this illustration, $\hat{\tau}_n$ is the vector of the upper triangular

elements of $\tilde{\Sigma}_n \otimes n^{-1}\sum_{t=1}^{n} a_t a_t'$ and $z_t = (y_t, \ldots, y_{t-p}, x_t, a_t$

Hansen (1982) proves consistency and asymptotic normality of GMM estimators under the assumption that the data process is stationary. (Note that various empirical papers refer to Hansen (1982) regrading consistency and asymptotic normality, although the empirical data process does not satisfy the stationarity condition.) Pötscher and Prucha (1991a,b) provide a review of the literature regarding consistency and asymptotic normality of M-estimators in dynamic nonlinear models with temporally dependent and temporally heterogeneous data processes. They give, furthermore, new consistency and asymptotic normality results for least mean distance and GMM estimators for the parameters of such models. Theorem 7.1 in Pötscher and Prucha (1991a) gives conditions for consistency. Theorem 5.5 in Pötscher and Prucha (1991b) gives an asymptotic normality result for GMM estimators corresponding to (1) under the assumption that either $q_t(z_t, \bar{\tau}_n, \bar{\beta}_n)$ with $\bar{\tau}_n \equiv \bar{\tau}$ and $\bar{\beta}_n \equiv \bar{\beta}$ is a martingale difference sequence or that $q_t(z_t, \bar{\tau}_n, \bar{\beta}_n)$ is near epoch dependent on some α-mixing basis process; see Assumptions 5.1,5.6,5.7,5.8 and 5.9 for details. Applying Theorem 5.5 to the subclass of GMM estimators defined by (2) we have

$$(3) \quad n^{1/2}(\hat{\beta}_n - \bar{\beta}_n) \sim N(0, \Omega_n)$$

with

$$\Omega_n = C_n^{-1}[E\nabla_\beta' \underline{S}_n]P(\bar{\tau}_n)\Psi_n P(\bar{\tau}_n)[E\nabla_\beta \underline{S}_n]C_n^{-1},$$

$$C_n = [E\nabla_\beta' \underline{S}_n]P(\bar{\tau}_n)[E\nabla_\beta \underline{S}_n]$$

$$\Psi_n = nE\underline{S}_n \underline{S}_n' ,$$

where $\underline{S}_n = n^{-1}\sum_{t=1}^{n} q_t(z_t, \bar{\beta}_n)$ and $\nabla_\beta = \partial/\partial\beta$. Note that $n^{-1}\Psi_n$ is the variance covariance matrix of the orthogonality conditions. The matrix Ψ_n can be written as

$$(4) \quad \Psi_n = n^{-1}E\left[\sum_{t=1}^{n} v_{t,n}\right]\left[\sum_{t=1}^{n} v_{t,n}'\right] =$$

$$= n^{-1}\sum_{t=1}^{n} Ev_{t,n}v_{t,n}' + \sum_{j=1}^{n-1} n^{-1}\sum_{t=1}^{n-j}[Ev_{t,n}v_{t+j,n}' + Ev_{t+j,n}v_{t,n}']$$

with $v_{t,n} = q_t(z_t, \bar{\beta}_n$ Theorem 6.5 in Pötscher and Prucha (1991b) gives

conditions under which Ω_n can be estimated consistently; cp. also the discussion in section 6 of that paper on estimators for Ψ_n. More specifically, consider the following estimators:

$$\hat{C}_n = \left[n^{-1}\sum_{t=1}^{n}\nabla_{\beta'} q_t(z_t,\hat{\beta}_n)\right] P(\hat{\tau}_n) \left[n^{-1}\sum_{t=1}^{n}\nabla_{\beta} q_t(z_t,\hat{\beta}_n)\right]$$

$$(5)\quad \Psi_n = w(0,n)\left[n^{-1}\sum_{t=1}^{n}\hat{v}_{t,n}\hat{v}'_{t,n}\right] + \sum_{j=1}^{n-1} w(j,n)\left[n^{-1}\sum_{t=1}^{n-j}[\hat{v}_{t,n}\hat{v}'_{t+j,n} + \hat{v}_{t+j,n}\hat{v}'_{t,n}]\right],$$

with $\hat{v}_{t,n} = q_t(z_t,\hat{\beta}_n)$. The weights $w(j,n)$ are taken to be zero for all j greater than some index, say, ℓ_n. Then, under the assumptions of Theorem 6.5, which allow for temporal dependence and temporal heteroscedasticity, $\hat{C}_n$ and $\hat{\Psi}_n$ are consistent estimators for C_n and Ψ_n, respectively, and Ω_n can be estimated consistently by

$$(6)\quad \hat{\Omega}_n = \hat{C}_n^{-1}\left[n^{-1}\sum_{t=1}^{n}\nabla_{\beta'} q_t(z_t,\hat{\beta}_n)\right] P(\hat{\tau}_n)\hat{\Psi}_n P(\hat{\tau}_n)\left[n^{-1}\sum_{t=1}^{n}\nabla_{\beta} q_t(z_t,\hat{\beta}_n)\right]\hat{C}_n^{-1}$$

For $\hat{\beta}_n$ to be efficient within the class defined by (2) we need the weighting matrix $P(\hat{\tau}_n)$ to be a consistent estimator for Ψ_n^{-1}. It then follows that $\Omega_n = C_n^{-1}$. Of course, in this case Ω_n can be estimated consistently by

$$(7\quad \hat{\Omega}_n = \hat{C}_n^{-1}.$$

Note that at this point TSP only reports estimates for Ω_n corresponding to formula (7) and not corresponding to formula (6). Thus if $P(\hat{\tau}_n)$ is not a consistent estimator for Ψ_n^{-1}, the reported estimates for Ω_n may be inconsistent.

If $q_t(z_t,\bar{\beta}_n)$ is a martingale difference sequence (and in particular if $q_t(z_t,\bar{\beta}_n)$ is independent over time), we can estimate Ψ_n consistently by

$$(8)\quad \hat{\Psi}_n = n^{-1}\sum_{t=1}^{n}\hat{v}_{t,n}\hat{v}'_{t,n} = n^{-1}\sum_{t=1}^{n}\hat{u}_t\hat{u}'_t \otimes a_t a'_t$$

If furthermore the u_t are homoscedastic, we can estimate Ψ_n consistently by

(9) $\Psi_n = \overset{v}{\Sigma}_n \otimes n^{-1}\sum_{t=1}^{n} a_t a_t'$ with $\overset{v}{\Sigma}_n = n^{-1}\sum_{t=1}^{n} \overset{v}{u}_t \overset{v}{u}_t'$

with $\overset{v}{u}_t$ representing some consistent estimator for u_t

Gallant, A.R., *Nonlinear Statistical Models*, Wiley, 1987.

Gallant, A.R., and H. White, *A Unified Theory of Estimation and Inference for Nonlinear Dynamic Models*, Basil Blackwell, 1988.

Hansen, L.P., Large Sample Properties of Generalized Method of Moments Estimators, *Econometrica*, 1982.

Pötscher, B.M., and Prucha, I.R., Basic Structure of Asymptotic Theory in Dynamic Nonlinear Models, I. Consistency and Approximation Concepts, *Econometric Reviews*, 1991a.

Pötscher, B.M., and Prucha, I.R., Basic Structure of Asymptotic Theory in Dynamic Nonlinear Models, II. Asymptotic Normality, *Econometric Reviews*, 1991b.