Chapter 14
FORECASTING AND MODEL SIMULATION

Often the primary goal of an econometric study is to produce a model for forecasting an economy's future behavior, a sector in an economy, or even the behavior of an individual firm. Two common approaches to this problem are the structural model method and the time series (ARIMA or VAR) methods. Both approaches can be used in TSP. A discussion of the pros and cons of these two methods is beyond the scope of this manual; an excellent reference that describes both methods, emphasizing their forecasting aspects is the Pindyck and Rubinfeld text.

Chapter 11 introduced the time series methods of forecasting variables using ARIMA or VAR models. In this chapter, we discuss how to use the structural model forecasting methods in TSP. First we give an overview of the steps in constructing a forecasting model of any type, then we describe methods for forecasting a single variable, and finally we describe the model solution procedures used for multi-equation models.

There are usually three major steps in producing a forecast:

1. Define a model with one or more equations giving a reasonably good description of the variables of interest over a recent time period. The model can come from any source, although multi-equation models must be expressible in TSP equations (FRMLs). In particular, it may arise from a series of estimations within TSP, both nonlinear and linear. The dimensions and quality of the model are your choice. If you want to use ARIMA models, use BJIDENT to help you choose the model.

2. Choose a forecast period and make some assumptions about the behavior of the exogenous variables in the model over this period. In an ARIMA forecast, there are no exogenous variables in the model, so you can skip this step.

3. Using projected values of the exogenous variables and the model you have chosen, choose one of TSP's forecasting or simulation procedures to compute the endogenous variables of the model one period at a time. Lagged endogenous variables may be determined from either the previous forecasts (dynamic simulation -- this method will always be used for ARIMA or VAR forecasts), or may be actual realizations of the variable (static or historical simulation).

TSP provides several procedures and techniques to compute the forecast and display the results. These techniques can be divided into two major groups, single equation (single variable) forecasts, and the solution of simultaneous equations models. We describe the former first, and then describe the model solution procedures, SIML and SOLVE.

14.1. Creating equations: FRML, FORM

Chapter 7 described how FRML can be used to make TSP equations. Any equation thus created can be used in forecasting with the proviso that equations for SOLVE must be normalized (have a left-hand side variable). Unlike earlier versions of TSP, logical expressions may be included in the equations of a simulation model.

It is also important to remember that equations used in simulation must have unknown parameter values defined in some way beforehand. This can be done by using a SET or PARAM statement to supply a value, for example,

```
FRML GNPGROW GNP = (1+ALPHA)*GNP(-1) ;
SET ALPHA = .15 ;
```

You can also estimate the equation earlier in your TSP program; in this case, the parameter will retain its estimated value when the simulation is performed. For example, using the same equation for GNP growth as above,

```
PARAM ALPHA ;
LSQ GNPGROW ;
```

Databanks are also useful for saving parameter values after an estimation. If the estimated value of ALPHA were 0.123, the equation GNPGROW would now be interpreted by GENR, SIML, or SOLVE as

```
FRML GNP = 1.123*GNP(-1) ;
```

FRML works well when estimating nonlinear models, but many equations will be estimated by linear methods, so it may be convenient to construct linear equations automatically. FORM does this easily:

```
OLSQ CX C W P P(-1) ;
FORM CONS ;
```

This example estimates a consumption equation by ordinary least squares and forms an equation named CONS as though you had entered the following FRML statement:

```
FRML CONS CX=16.6+.81*W+.017*P+.216*P(-1);
```

[assuming that the coefficient vector estimated in the OLSQ procedure was (16.6, .81, .017, .216)].

FORM may be used after any linear estimation procedure: OLSQ, 2SLS, LIML, LAD, or AR1. If it is used following an AR1 estimation, the term involving $\hat{\rho}$ times the lagged residual is automatically added (see Section 14.3). FORM can also be used to create unnormalized equations and to create equations with parameter names rather than values. See the *Reference Manual* for details.

14.2. Forecasting with an explicit equation: GENR

The simplest way to forecast a single variable is with GENR, if an explicit equation has already been defined. For example, the following two sets of statements are equivalent in their result:

```
SMPL 83 90 ;
GENR GNPFIT = A0 + A1*POP + A2*IMPT ;
```

```
SMPL 83 90 ;
FRML GNPEQ GNP = A0 + A1*POP + A2*IMPT ;
GENR GNPEQ GNPFIT;
```

This method even works when the variable to be forecast depends on its own lagged values (which is described below), because GENR will operate dynamically. The alternate method for computing single equation forecasts, FORCST, can be used for either static or dynamic simulation. In addition, FORCST does not require that you specify the equation beforehand; it can be computed directly from a previous linear estimation.

14.3. Forecasting linear models: FORCST

FORCST will do a forecast immediately following any single equation linear estimation command in TSP (OLSQ, 2SLS, LIML, LAD, or AR1) without requiring FORM or a FRML. This also works on PROBIT and TOBIT models, although you may wish to transform the predicted latent variable.

Here are some examples:

```
FREQ A ; SMPL 48 56 ;
OLSQ IMPT C GNP RELP ;
FORCST(PRINT) IMPFIT ;
```

In this example, the forecast is computed over the estimation sample and therefore IMPFIT is simply the fitted values of IMPT (equal to the automatically stored @FIT series). It is generally true that if you compute a static forecast over the same sample as the regression, the output of FORCST will be the fitted values from the regression.

```
FREQ A ; SMPL 48 82 ;
  AR1 (METHOD=CORC,PRINT) IMPT C GNP RELP ;
  SMPL 83 90 ;
  FORCST(PRINT) IMPFOR ;
```

This example illustrates the computation of a dynamic forecast when the errors are serially correlated. Projections of GNP and RELP must exist for the period 1983 to 1990; the simulation uses the last historical period (1982) to calculate a presample residual. For this kind of model, the forecasting equation is

$$y_t = X_t b + \hat{\rho} e_{t-1}$$

where $e_{t-1} = \hat{\rho} e_{t-2}$; the initial condition for e is the presample residual.

```
FREQ A ; SMPL 48 82 ;
OLSQ IMPT C IMPT(-1) GNP ;
SMPL 83 90 ;
FORCST IMPDYN ;
```

This example shows forecasting when there is a lagged dependent variable in the model. Since IMPT has been referred to explicitly with a lag on the right-hand side of the original equation, the dynamic forecast will automatically use the predicted values of IMPT in the next period's forecast. To use this feature, you must refer to the lagged endogenous variables as IMPT(-1); use of a computed variable, such as IMPTL1 = IMPT(-1), will not have the same effect.

It is also possible to use FORCST independently of any estimation command; in this case you must explicitly supply the procedure with all the information it needs to compute the forecast:

```
FORCST(DYNAM,COEF=BETA,RHO=R,DEPVAR=IMPT)
 IMPFIT C GNP RELP ;
```

In this example, BETA is the vector of estimated coefficients of the independent variables, R the estimated value of the serial correlation coefficient, IMPT the name of the original dependent variable (needed for dynamic forecasts), IMPFIT the name to be given to the forecasted variable, and C, GNP, and RELP are the independent variables in the forecasting equation. The coefficients BETA must be supplied in the same order as the list of independent variables.

14.4. Solving simultaneous equation models

TSP provides two quite different procedures for the solution of simultaneous equation models. The methods differ in their speed of convergence, use of computer storage and time, and ability to handle highly nonlinear or very simultaneous models.

The most general and powerful procedure is SIML, which uses Newton's method applied to nonlinear equation solution. The model is specified just like FIML. SIML does not require normalized equations nor that the model be ordered in a particular way, and uses analytic derivatives of the model in the solution process. For linear models, SIML converges in one iteration, and is very fast for multiperiod models. For highly nonlinear models, Newton's method (Gauss-Newton) may be the only method in TSP that can provide a solution. The cost of this power is considerable use of computer storage compared with other model solution techniques. Consequently, SIML may not be the method of choice for more than 50 equations. TSP does not explicitly limit the number of equations, but you may not have enough memory available. You may also be limited by TSP's (relatively large) limits on the number of unique variable names/arguments in the collected model and on the number of variables in a TSP session. Use the SHOW command to see what these limits are in your copy of TSP.

For large economic models, particularly those with some sort of block structure, SOLVE will be more suitable. To use this procedure, first collect the equations of the model using MODEL to determine the best order for solution. The equations must be "normalized", i.e., each endogenous variable must appear once and only once on the left-hand side of an equation. MODEL determines how the equations are to be arranged into recursive and simultaneous blocks for solution. SOLVE will evaluate the recursive blocks (blocks in which every equation uses only previously computed

endogenous variables as input) and will solve the simultaneous blocks either by the Gauss-Seidel method or by the more powerful Fletcher-Powell method. This method of model solution is suitable for most large economic models, which tend to be sparse, fairly linear, and separable into blocks. (It has been used on models with more than 400 equations.) However, convergence of the simultaneous blocks is not guaranteed, since it does not compute any analytic derivatives.

These two procedures are described in greater detail below.

14.4.1. Small nonlinear models: SIML

SIML invokes Newton's method; it has the same form as FIML described in Chapter 7: SIML, followed by options including ENDOG=(list of endogenous variables), followed by a list of equations, including identities. Unlike FIML, SIML treats IDENTs the same as FRMLs (see Section 7.1). For example, the solution of the illustrative model is specified by

```
SIML(ENDOG=(GNP,CONS,I,R,LP)) CONSEQ,INVEQ,INTRSTEQ,GNPID,PRICEQ;
```

The default values of the options are Newton's method, no storage of the results, and a dynamic simulation. A static simulation is often used when simulating over a historical period: it uses actual realized values of the lagged endogenous variables rather than obtaining them from the simulation of the previous period. The default for printing is to print a one line summary of each iteration and a table of the solved variables at the end.

An example of SIML that stores the results and prints the input data is

```
SIML(PRNDAT,TAG=S,ENDOG=(GNP,CONS,I,R,LP)) CONSEQ,INVEQ,INTRSTEQ,GNPID,PRICEQ;
```

After this model has been solved, the solved series are stored under the names GNPS, CONSS, IS, etc. Further details on the options available with SIML are given in the *Reference Manual.*

14.4.1.1. Newton's Method

The method used by SIML for solution of nonlinear simultaneous equation models is a variant of Newton's method. Obviously, if a simultaneous model is linear, it can be solved directly by matrix inversion. That is, the solution to the matrix equation

$$A\,x = b$$

is

$$x = A^{-1}b.$$

Newton's method applies this idea to the iterative solution of nonlinear models. At each iteration, the model is linearized in its variables around the values from the previous iteration. The linearized model is solved by matrix inversion. In terms of the equation above, A is the Jacobian of the model with respect to the variables, b the vector of model values at the previous iteration and x the direction vector of changes in the variables to be computed.

The resulting vector of changes is used as a direction vector in a linear search for better values of the variables, as in the other iterative procedures such as LSQ. A deviation is computed for each equation by substituting the current values of the variables; at the solution, all the deviations will be zero. The criterion for the search is the sum of squared deviations of the equations; it is printed as F = and FNEW = in SIML's output. Unless a solution has been achieved, there will be a better set of values somewhere along the direction vector. The actual choice of the new set of variable values is made by the same methods used in nonlinear estimation and is described in Chapter 10. For a further description of Newton's method, see Saaty and Bram (1964) or Ortega and Rheinboldt (1970).

The POS function can be used to constrain the solved values to be non-negative.

14.4.2. Large models

Large models require two steps (procedures) for solution -- MODEL and SOLVE. MODEL is used to group the equations into smaller blocks, and SOLVE seeks the solution using this structural information. Several SOLVE commands can use the same MODEL (using different scenarios of exogenous variables or parameter values, for example). SOLVE uses the same iteration and TAG options as SIML.

14.4.2.1. Ordering equations: MODEL

MODEL groups the equations of a model into blocks and saves this structure under a model name. We use the well-known Klein Model I as a simple example. This model consists of three behavioral equations and four identities. There are seven endogenous variables (CX, I, W1, P, K, W and E), three lagged endogenous variables (P(-1), K(-1), and E(-1)), and four exogenous variables (G, TIME, TX, and W2). The equations may be estimated by a single equation method, using FORM to construct a TSP equation, or they may be specified with FRMLs and estimated with multi-equation methods. We assume that we will use two-stage least squares, and specify the model as follows:

```
LIST IVK C P(-1) E(-1) K(-1) G TIME TX W2;
2SLS(INST=IVK) CX C,W,P,P(-1);
FORM CONS ;
2SLS(INST=IVK) I C,P,P(-1),K(-1);
FORM INV ;
2SLS(INST=IVK) W1 C,E,E(-1),TIME;
FORM WAGES ;
IDENT WAGE W=W1+W2;
IDENT BALANCE P = CX+I+G - (TX+W) ;
IDENT PROFIT E = P+TX+W1 ;
IDENT CAPSTCK K=K(-1)+I ;

LIST KLEIN CONS INV WAGES WAGE BALANCE PROFIT CAPSTCK ;
LIST KENDOG CX I W1 W E P K ;
MODEL KLEIN,KENDOG,KLEINC ;
```

MODEL takes the model specified in the list of equations KLEIN and the list of endogenous variables KENDOG, and orders the equations into a recursive block structure. An ordering of this kind provides increased understanding of the model structure and provides efficient solution. The recursive block ordering for the system is formed by operations on the "adjacency" matrix of the system, a matrix of ones and zeroes relating the dependent variables in the system to the equations in which they appear.

See Steward (1962) for further explanation of this procedure and for details of the specific algorithm employed in obtaining the ordering. The method involves viewing the adjacency matrix as a network and systematically seeking closed loops that define systems of simultaneous equations.

The output for the MODEL example above would be the following:

```
Block structure of KLEINC

Blk Eq# Equation Dep.Var. 1234567
 1S   1 INV      I         X  X
 1S   2 WAGE     W          X   X
 1S   3 PROFIT   E            XX X
 1S   4 BALANCE  P         XX XX
 1S   5 CONS     CX         X XX
 1S   6 WAGES    W1           X  X
 2R   7 CAPSTCK  K         X     X
```

The column labeled Blk shows the number of each block of equations and the type (S for simultaneous and R for recursive). The X's mark the equations in which each dependent variable appears.

Because this model is completely simultaneous except for the capital stock equation, there is one block with six equations, followed by the equation that determines the new level of capital stock from investment. This last equation could be left out of the model without affecting the results in any way, since the level of the capital stock has no impact on output in this version of the model.

Having ordered this model, the procedure then puts the ordered list of equations and variables under the name KLEINC.

14.4.2.2. Solution: SOLVE

SOLVE causes the model saved by a MODEL statement to be solved. The only required argument is the name of the ordered model. For each time period in the sample, each block is solved separately in the order determined by MODEL. The recursive blocks are always solved in one iteration via the equivalent of GENR. Three methods are provided for solving the simultaneous blocks: the Gauss-Seidel, Jacobi, and Fletcher-Powell algorithms.

14.4.2.3. Gauss-Seidel and Jacobi methods

Solution by the Gauss-Seidel method is the default, and it could be invoked for the Klein model in the following way (TAG is optional):

```
SOLVE(TAG=S) KLEINC;
```

The option METHOD=JACOBI specifies that a variant of the Gauss-Seidel method called the Jacobi method (see pp. 217-220 of Ortega and Rheinboldt) is to be used. This method does not update the endogenous variables immediately while the simultaneous block is being computed, but waits until the beginning of the next iteration and updates them all at once.

14.4.2.4. Fletcher-Powell method

To use the Fletcher-Powell method to solve the simultaneous blocks of a model, include the METHOD=FLPOW option in SOLVE:

```
SOLVE(METHOD=FLPOW) KLEINC ;
```

The Fletcher-Powell method is described in Fletcher and Powell (1963); it is useful when the Gauss-Seidel method does not converge. It uses numeric derivatives with respect to the endogenous variables (as opposed to SIML, which uses analytic derivatives).

14.4.2.5. Example: a 33-equation model

Example 4.1 shows a sample TSP job for the solution of a rather complex 33-equation model with 6 structural equations and 27 identities. The first three pages show the TSP input for the model:

The data (10 observations in the example) are read from an external file, 6 variables per record, 14 records per observation. Some normalizing constants are defined.

The identities are specified; note that the equations are normalized, i.e., there is only one endogenous variable on the left-hand side of each equation, and each endogenous variable appears only once on the left-hand side. The six behavioral equations are defined, together with parameter values previously estimated by FIML over the whole sample. The endogenous variables and equation names are put into LISTs.

MODEL is executed to produce a collected model, TRADEC, to be input to SOLVE. We show the output of MODEL on the following page. This output shows the best order of solution of the model: the procedure has determined that the first 9 equations are recursive, the next 15 are a simultaneous block, and the last 9 are recursive once the previous

24 have been solved.

An inspection of the model confirms this: the first 7 equations involve only exogenous and lagged endogenous variables, equation IDFIO involves V and N, already determined by IDV and IDF9, equation IDRK involves RP, already determined by IDRP. Then we start the simultaneous block. Following this block is a series of equations that are essentially superfluous to the model: the whole thing could be solved without the last recursive section and then these equations could be computed using simple GENRs (in the order listed).

The following page shows the start of the solution procedure. Recursive blocks always take only one iteration, so no message is printed; the convergence messages shown correspond to the simultaneous blocks for the two periods. If we had used the PRINT option, there would also be a printout of SSR (sum of squared residuals) showing the golden section stepsize search method searching for the optional stepsize for each Fletcher-Powell iteration on the simultaneous block. Unlike the quasi-Newton minimization methods used elsewhere in TSP, the Fletcher-Powell algorithm does not have a natural stepsize of approximately unity.

14.5. Displaying and evaluating a forecast: ACTFIT

The first step in evaluating a forecast is probably to print or plot it. If you are using the single equation FORCST, this is easily accomplished by including the PRINT option on the statement:

```
FORCST(PRINT) SALESF ;
```

This option prints information about the equation it is using for the forecast and a time series plot of the forecasted variable over the time period in question. The plot also shows the values of the variable on the right-hand side. On a PC, to see a high-resolution graphics plot use the command :

```
PLOT SALES SALESF ;
```

If you use SOLVE or SIML to obtain the forecast, plotting will not be automatically available, although you can obtain a table of the solved values by use of the PRNSIM option. To get plots, save the simulated values with a TAG= option (for example, CXS, IS, etc.) and use the PLOT procedure described in Chapter 6:

```
PLOT CX A CXS S ;          ? hardcopy
PLOT CX CXS ;              ? high-resolution graphics
```

This example assumes that the simulation has been done over a historical period, so that both actual (CX) and solved (CXS) values of consumption are available and may be plotted on the same scale for comparison. Of course, if you did not know CX for the forecasting period, you could simply plot CXS.

Standard references for the evaluation of forecasts are: Theil (1961 and 1966) and Pindyck and Rubinfeld (1976). Some of the measures discussed by Theil in Chapter 2 of his 1966 book have been incorporated into TSP in the ACTFIT procedure. You can use this procedure to compute a set of statistics such as the root mean square error, the inequality coefficient (U), and a decomposition of the sources of forecast error. The command is

```
ACTFIT CX CXS ;
```

Once again, the historical values of the variable are required to make this comparison. If the option PLOTS has been turned on, ACTFIT also plots the two variables and their difference.

> **Note:** The 1961 and 1966 definitions of U differ; Pindyck and Rubinfeld use the 1961 definition. TSP prints both versions of U.

```
         PROGRAM
LINE  ***************************************************************
1  OPTIONS CRT NODATE;
2  NAME TRADESOL '33 EQ TRADE MODEL - SOLVE WITH FLETCHER-POWELL METHOD' ;
2  ?
2  ?    READ IN THE DATA FOR THE TRADE MODEL FROM THE FILE TRADSOL.DAT
2  ?
2  FREQ A ; SMPL 64 73 ;
4       READ (FILE='TRADESOL.DAT',FORMAT='(6G12.5)')
4          ACDP AK AKD AKL CC CE
4          CG DG DLDT DP DR E
4          EI EJ EL ER ET EX
4          G HLT HR I IG IM
4          K KD L LD LE LG
4          LH LJ LR LU M N
4          NRE NW P PC PCE PCG
4          PDP PEX PF PG PI PIG
4          PIM PKD PL PLD PLE PLG
4          PLR PR R RDP RE RIM RK
4          RL RP RT RV RW SHRC
4          SHRDP SHREX SHRFC SHRIM SHRKD T
4          TDP TIM TK TL TP TV
4          TW V VCR VIR W ;
5  ?    SET THE CONSTANTS AND PARAMETERS FOR THE RUN.
5  ?
5  SET NKD=166.92656 ;
6  SET NLD=252.81388 ;
7  SET NIM= 20.331 ;
8  SET NDP=419.56444 ;
9  SET NEX= 20.507 ;
10  SET  NT=  0.0 ;
11  SET  NL=  0.8464  ;
12  ?
12  ?    EQUATIONS OF THE 33 EQUATION MODEL.
12  ?
12  IDENT IDSDP  DP=SHRDP*PLD*LD/PDP ;
13  IDENT IDSIM  PL=PL+SHRIM+PIM*IM/(PLD*LD) ;
14  IDENT IDSKD   L=L+SHRKD+PKD*KD/(PLD*LD) ;
15  IDENT IDSC   PC=(PL*LJ*SHRC)/(CC*(1-SHRC)) ;
16  IDENT IDSFC  CC=(SHRFC*(1+NW)*W(-1)-PL*LJ)/PC ;
17  IDENT IDF7  PDP=PC/ACDP ;
18  IDENT IDF8   LJ=LH-L ;
19  IDENT IDF9    N=(M*PI*AKL+PI(-1)*AK(-1)-PI*AKL
19                  -((1-TK)*((PKD*AKD)-TP*PI(-1)
19              *AK(-1))))/(-PI(-1)*AK(-1)) ;
20  IDENT IDF10  NW=(-V-N*PI(-1)*AK(-1)*K(-1) + (PI*AKL-PI(-1)*AK(-1))*K(-1)
20              -(1-TV)*(EI+VCR+VIR+NRE)-EJ+RV+TW*W(-1))/(-W(-1)) ;
21  IDENT IDF11 PEX=-(PDP*DP-PLD*LD-PKD*KD-PIM*IM)/EX ;
22  IDENT IDF12 IM = ((1+TDP)*PDP*DP-PC*CC+PCE*CE-PCG*CG-PIG*IG-PI*I)/
22                    (-TIM*PIM)  ;
23  IDENT IDF13 PLD=(PL*L/(1-TL)-PLE*LE-PLG*LG-PLR*LR)/LD ;
24  IDENT IDF14  LD=L-(LE+LG+LR+LU) ;
25  IDENT IDF16   R=(PEX*EX-PIM*IM+VCR+VIR+PLR*LR-ER-HR+NRE-ET)/PR+R(-1) ;
26  IDENT IDDG DG=E+EL+ER+EI+EJ-(RDP+RIM+RP+RK+RL+RW+RT+RE+RV);
27  IDENT IDDR DR=PEX*EX-PIM*IM+VCR+VIR-ER-HR+PLR*LR+NRE ;
28  IDENT IDE E=PCG*CG+PIG*IG+PLG*LG ;
29  IDENT IDG G=(DG+ET)/PG+G(-1) ;
30  IDENT IDKD KD=AKD*K(-1);
31  IDENT IDPF PF=PF(-1)*EXP(0.5*(
31       ((PC*CC)/(PC*CC+PL*LJ)+(PC(-1)*CC(-1))/
31       (PC(-1)*CC(-1)+PL(-1)*LJ(-1)))
31      *LOG(PC/PC(-1))
31      +((PL*LJ)/(PC*CC+PL*LJ)+(PL(-1)*LJ(-1))/
31      (PC(-1)*CC(-1)+PL(-1)*LJ(-1)))
31      *LOG(PL/PL(-1)))) ;
32  IDENT IDRDP RDP=TDP*PDP*DP ;
33  IDENT IDRE RE=PCE*CE-PLE*LE ;
34  IDENT IDRIM RIM=TIM*PIM*IM ;
35  IDENT IDRK RK=TK*(PKD*KD-RP)+TV*(EI+VCR+VIR+NRE);
```

Example 14.1: Simulation of a Medium-sized Trade Model

```
36  IDENT IDRL RL=TL*(PLD*LD+PLE*LE+PLG*LG+PLR*LR);
37  IDENT IDRP RP=TP*PI(-1)*AK(-1)*K(-1);
38  IDENT IDRW RW=TW*W(-1);
39  IDENT IDV V=(PI*AKL-PI(-1)*AK(-1))*K(-1)+(PG-PG(-1))*G(-1)
39               +(PR-PR(-1))*R(-1);
40  ?
40  ?     BEHAVIORAL EQUATIONS FOR THE CONSUMPTION SIDE OF THE MODEL AND THEIR
PARAMETER ESTIMATES.
40  ?     SPECIFICATION: SYMMETRIC, CONVEXITY IMPOSED
40  ?
40  FRML INTER1   SHRFC
40       =((AXO+BON*LOG((PF*P)/
40             ((1+NW)*W(-1)+LDA*(EL-HR-RT)+DEL*PL*LH))
40             +BOT*(T-NT))/
40       (-1+BNN*LOG((PF*P)/
40             ((1+NW)*W(-1)+LDA*(EL-HR-RT)+DEL*PL*LH))
40             +BNT*(T-NT)))*
40       (1+(LDA*(EL-HR-RT)+DEL*PL*LH)/((1+NW)*W(-1)))$
41  FRML INTRA1 SHRC=AC+BCC*LOG(PC/(PL/NL)) ;
42  PARAM AXO -.122007 BON -.0242706 LDA 7.85130 DEL 7.85130
42     BNN -.197991 BNT .00143592 BOT .00014993G1 ;
43  PARAM AC .148803 BCC .0155448 ;
44  ?
44  ?     BEHAVIORAL EQUATIONS FOR THE PRODUCTION SIDE OF THE MODEL AND
44  ?   THEIR PARAMETERS.
44  ?
44  FRML CODP SHRDP=ADP+(LDPKD*DKD-AKS*ADP)*LOG((KD*NEX)/(NKD*EX))
44            +(LDPKD*LDPKD*DKD+LDPIM*LDPIM*DIM+DDP
44            -ADP*(ADP-1))*LOG((DP*NEX)/(NDP*EX))
44            +BDPT*(T-NT);
45  FRML COKD SHRKD=AKS+(DKD-AKS*(AKS-1))*LOG((KD*NEX)/(NKD*EX))
45            +(LIMKD*DKD-AKS*AIM)*LOG((IM*NEX)/(NIM*EX))
45            +(LDPKD*DKD-AKS*ADP)*LOG((DP*NEX) /(NDP*EX))+BKDT*(T-NT);
46  FRML COIM SHRIM=AIM+(LIMKD*DKD-AKS*AIM)*LOG((KD*NEX)/(NKD*EX))
46            +(LIMKD*LIMKD*DKD+DIM-AIM*(AIM-1))*LOG((IM*NEX)/(NIM*EX))
46            +(LDPKD*LIMKD*DKD+LDPIM*DIM-AIM*ADP)*LOG((DP*NEX)/
46            (NDP*EX))+BIMT*(T-NT) ;
47  FRML COT DLDT=AT+BKDT*LOG((KD*NEX)/(NKD*EX))+
47            BIMT*LOG((IM*NEX)/(NIM*EX))
47            +BDPT*LOG((DP*NEX)/(NDP*EX))+BTT*(T-NT);
48  PARAM AT -.02217 BTT .0001963 ;
49  PARAM AKS -.6373 DKD 1.260 LIMKD .0881 AIM -.0794
49        LDPKD -1.020 ADP 1.638 ;
50  PARAM BKDT -.0046 DIM -.0262 LDPIM -.4053 BIMT .0006
50        DDP .0320 BDPT .0048 ;
51  ?
51  ?      THIS IS THE LIST OF ENDOGENOUS VARIABLES IN THE MODEL.
51  ?
51  LIST ENDOGL  E KD RE RP RW V N NW RK
51               LJ LD PDP SHRC CC PF PLD    SHRDP SHRKD SHRIM
51               L PL DP SHRFC PC
51               PEX R DLDT DR RDP RIM RL DG G ;
52  ?
52  ?      THIS IS THE LIST OF EQUATIONS IN THE MODEL; THE ORDER CORRESPONDS
52  ?   TO THE ORDER OF SOLUTION DESIRED.
52  LIST TRADEM IDE IDKD IDRE IDRP IDRW IDV IDF9 IDF10 IDRK
 52               IDF8 IDF14 IDF7 INTRA1 IDSFC IDPF IDF13
 52               CODP COKD COIM
 52                    IDSKD IDSIM IDSDP INTER1 IDSC
 52               IDF11 IDF16 COT IDDR IDRDP IDRIM IDRL IDDG IDG ;
 53  ?
 53  ?      THIS COLLECTION OF THE MODEL IS SUPERFLUOUS SINCE WE HAVE ALREADY
 53  ?      ACHIEVED THE DESIRED ORDERING.
 53  ?
 53  COLECT TRADEM ENDOGL TRADEC ;
 54  SMPL 65 66 ;
 55  SOLVE(STATIC,METHOD=FLPOW,TAG=F,MAXIT=50) TRADEC ;
 56  STOP ; END ;
       EXECUTION
```

Example 14.1: (continued, page 2).

```
 Current sample:  1964 to 1973

 Block structure of TRADEC

                                        11111111111222222222223333
 Blk Eq# Equation Dep.Var. 12345678901234567890123456789012345678901123
  1R   1 IDE      E         X
  1R   2 IDKD     KD         X
  1R   3 IDRE     RE          X
  1R   4 IDRP     RP           X
  1R   5 IDRW     RW            X
  1R   6 IDV      V              X
  1R   7 IDF9     N               X
  1R   8 IDF10    NW              XXX
  1R   9 IDRK     RK         X X    X
  2S  10 IDF8     LJ                  X          X
  2S  11 IDF14    LD                   X         X
  2S  12 IDF7     PDP                   X           X
  2S  13 INTRA1   SHRC                   X        X  X
  2S  14 IDSFC    CC                X X   X       X XX
  2S  15 IDPF     PF                  X   XX      X  X
  2S  16 IDF13    PLD                  X    X   XX
  2S  17 C0DP     SHRDP      X                X    X
  2S  18 C0KD     SHRKD      X                  X   X
  2S  19 C0IM     SHRIM      X                    X  X
  2S  20 IDSKD    L          X         X     X X X
  2S  21 IDSIM    PL                    X     X  X X
  2S  22 IDSDP    DP                    XX    XX    X
  2S  23 INTER1   SHRFC             X       X      X X
  2S  24 IDSC     PC                  X  XX       X  X
  3R  25 IDF11    PEX        X         XX   X      X  X
  3R  26 IDF16    R                                  XX
  3R  27 C0T      DLDT       X                      X    X
  3R  28 IDDR     DR                                    X  X
  3R  29 IDRDP    RDP                    X          X       X
  3R  30 IDRIM    RIM                                       X
  3R  31 IDRL     RL                     X    X               X
  3R  32 IDDG     DG         X XXX   X                      XXXX
  3R  33 IDG      G                                             XX


Current sample:  1965 to 1966

                        SIMULATION OF THE MODEL TRADEC
                        ==============================

                           OPTIONS FOR THIS ROUTINE
                           ========================

  CONV1    = 0.01000000      CONV2    = 0.001000000     DEBUG    = FALSE
  DYNAM    = FALSE           ITERMX   = 50              KILL     = FALSE

  MAXIT    = 50              MAXPRT   = 5               METHOD   = FLPOW
  PRINT    = FALSE           PRNDAT   = FALSE           PRNRES   = FALSE
  PRNSIM   = TRUE            RESIDU   = FALSE           SOLNAM   = F
  STATIC   = TRUE            TAG      = F               TOL      = 0.01000000

NUMBER OF EQUATIONS IN THE MODEL =  33
NUMBER OF BLOCKS IN THE MODEL = 3

          BLOCK #   NUMBER OF EQUATIONS

                1          9
                2         15
                3          9
```

Example 14.1: (continued, page 3).

```
F=  0.35338E-01 FNEW=  0.35975E-02 ISQZ=  5 STEP= 0.93253E-04 CRIT=  661.22
F=  0.35975E-02 FNEW=  0.32056E-02 ISQZ=  1 STEP= 0.46627E-04 CRIT=  18.170
F=  0.32056E-02 FNEW=  0.31779E-02 ISQZ=  1 STEP= 0.23313E-04 CRIT=  1.9218
F=  0.31779E-02 FNEW=  0.31748E-02 ISQZ=  1 STEP= 0.23313E-04 CRIT= 0.36206
F=  0.31748E-02 FNEW=  0.31735E-02 ISQZ=  1 STEP= 0.58283E-05 CRIT= 0.53940
F=  0.31735E-02 FNEW=  0.31731E-02 ISQZ=  1 STEP= 0.58283E-05 CRIT= 0.12414
F=  0.31731E-02 FNEW=  0.31723E-02 ISQZ=  1 STEP= 0.23313E-04 CRIT= 0.10275
F=  0.31723E-02 FNEW=  0.31714E-02 ISQZ=  1 STEP= 0.58283E-05 CRIT= 0.28931
F=  0.31714E-02 FNEW=  0.31711E-02 ISQZ=  1 STEP= 0.11657E-04 CRIT= 0.52775E-01
F=  0.31711E-02 FNEW=  0.31683E-02 ISQZ=  1 STEP= 0.93253E-04 CRIT= 0.89781E-01
F=  0.31683E-02 FNEW=  0.31640E-02 ISQZ=  1 STEP= 0.23313E-04 CRIT= 0.33477
F=  0.31640E-02 FNEW=  0.31625E-02 ISQZ=  1 STEP= 0.46627E-04 CRIT= 0.60929E-01
F=  0.31625E-02 FNEW=  0.30717E-02 ISQZ=  6 STEP= 0.32532E-02 CRIT= 0.57482E-01
F=  0.30717E-02 FNEW=  0.20818E-02 ISQZ= 10 STEP= 0.28686E-01 CRIT= 0.66896E-01
F=  0.20818E-02 FNEW=  0.14164E-02 ISQZ= 10 STEP= 0.18745E-01 CRIT= 0.70349E-01
F=  0.14164E-02 FNEW=  0.10492E-02 ISQZ= 10 STEP= 0.18745E-01 CRIT= 0.39951E-01
F=  0.10492E-02 FNEW=  0.99427E-03 ISQZ=  7 STEP= 0.45593E-02 CRIT= 0.24638E-01

CONVERGENCE ACHIEVED AFTER   17 ITERATIONS.

F=   329.30     FNEW=  0.24536E-02 ISQZ= 15 STEP= 0.93269E-02 CRIT=  70635.
F=  0.24536E-02 FNEW=  0.23947E-02 ISQZ=  1 STEP= 0.23313E-04 CRIT=  7.6644
F=  0.23947E-02 FNEW=  0.23872E-02 ISQZ=  1 STEP= 0.23313E-04 CRIT=  1.1213
F=  0.23872E-02 FNEW=  0.23841E-02 ISQZ=  1 STEP= 0.11657E-04 CRIT= 0.64235
F=  0.23841E-02 FNEW=  0.23820E-02 ISQZ=  1 STEP= 0.23313E-04 CRIT= 0.19101
F=  0.23820E-02 FNEW=  0.23525E-02 ISQZ=  1 STEP= 0.37301E-03 CRIT= 0.13292
F=  0.23525E-02 FNEW=  0.23253E-02 ISQZ=  3 STEP= 0.74603E-03 CRIT= 0.79214E-01
F=  0.23253E-02 FNEW=  0.23142E-02 ISQZ=  1 STEP= 0.37301E-03 CRIT= 0.10094
F=  0.23142E-02 FNEW=  0.23056E-02 ISQZ=  1 STEP= 0.93253E-04 CRIT= 0.18896
F=  0.23056E-02 FNEW=  0.22694E-02 ISQZ=  4 STEP= 0.18443E-02 CRIT= 0.41723E-01
F=  0.22694E-02 FNEW=  0.21749E-02 ISQZ=  6 STEP= 0.36886E-02 CRIT= 0.53054E-01
F=  0.21749E-02 FNEW=  0.20761E-02 ISQZ=  7 STEP= 0.56356E-02 CRIT= 0.34541E-01
F=  0.20761E-02 FNEW=  0.17175E-02 ISQZ= 10 STEP= 0.16907E-01 CRIT= 0.39828E-01
F=  0.17175E-02 FNEW=  0.11663E-02 ISQZ= 10 STEP= 0.23679E-01 CRIT= 0.46441E-01
F=  0.11663E-02 FNEW=  0.84419E-03 ISQZ=  9 STEP= 0.14754E-01 CRIT= 0.46001E-01
F=  0.84419E-03 FNEW=  0.82742E-03 ISQZ=  6 STEP= 0.39576E-02 CRIT= 0.82962E-02

CONVERGENCE ACHIEVED AFTER   16 ITERATIONS.
THE SOLVED VARIABLES  ARE STORED WITH A TAG:    F

                            SIMULATION RESULTS

                     E            KD            RE            RP            RW
1965         137.00119     207.99859       1.44707      30.97627       3.63086
1966         156.81016     219.27492       1.22126      32.79372       3.90878

                     V             N            NW            RK            LJ
1965          10.00131      0.054672      0.050904      44.88508    2361.87817
1966          76.73501      0.095167      0.084888      49.95992    2415.59923

                    LD           PDP          SHRC            CC            PF
1965         296.87295       1.09249       0.14670     386.97003       1.25840
1966         308.72938       1.12410       0.14643     407.14032       1.31754

                   PLD         SHRDP         SHRKD         SHRIM             L
1965           1.25097       1.69216      -0.69388     -0.086272     372.73465
1966           1.31518       1.69956      -0.69774     -0.094543     388.01917

                    PL            DP         SHRFC            PC           PEX
1965           1.08487     583.68877       1.33431       1.11572       1.15193
1966           1.13741     624.94808       1.33769       1.15322       1.23089

                     R          DLDT            DR           RDP           RIM
1965          33.48336     -0.020346       8.35776      29.09586       1.64702
1966          38.22776     -0.020047       8.97031      29.49516       1.93012

                    RL            DG             G
1965          41.85911       3.40792     317.38777
1966          48.54232       3.99788     327.75846
```

Example 14.1: (continued, page 4).

14.6. Monte Carlo Simulation: RANDOM

Random variables are useful in a variety of situations, from checking the statistical properties of an econometric or simulation model to computing bootstrap standard errors when analytic formulas are not available. This type of procedure is often referred to as Monte Carlo analysis, and is easily programmed in TSP. For example, to draw a series E of independent standard normal variates, use the following command:

```
RANDOM E ;
```

To draw 3 series X1, X2, X3 that have a multivariate normal distribution with mean XMEAN (a length 3 vector) and variance matrix XVAR (a symmetric matrix of order 3), use this command:

```
RANDOM (VMEAN=XMEAN,VCOV=XVAR) X1-X3 ;
```

Random can also be used to generate series from uniform, Poisson, Cauchy, exponentail, Laplace, student's t, gamma, negative binomial, and arbitrary empirical distributions. See the *Reference Manual* for details on the options necessary.

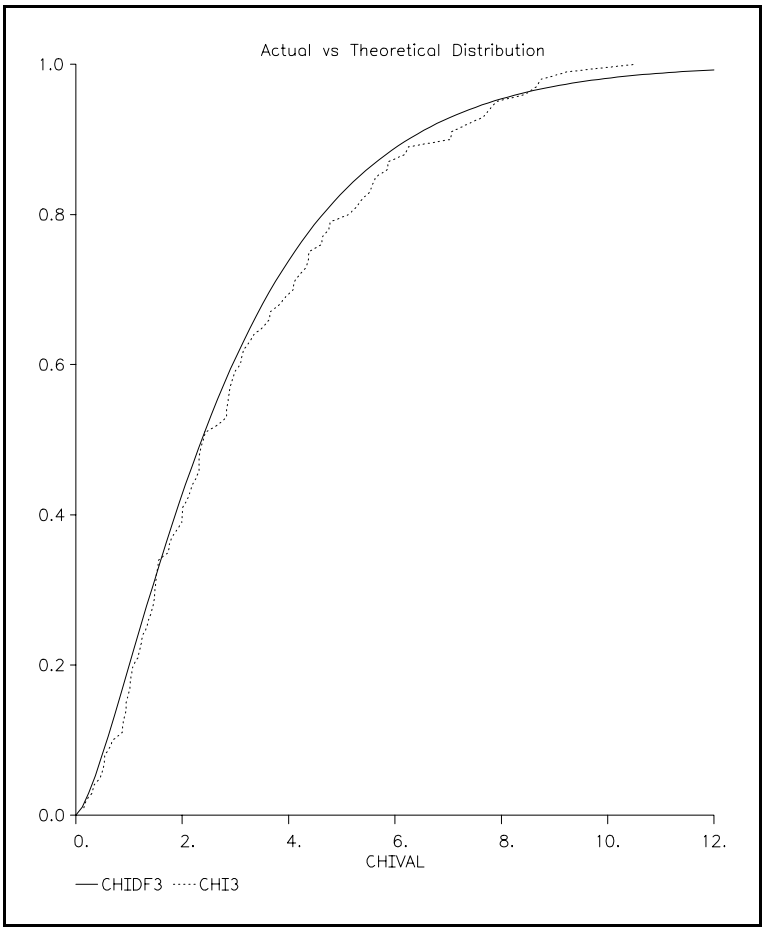
Two useful examples appear in Chapter 9: in section 9.6.3, we simulate an ARCH model using the standard normal distribution and a dynamic GENR. In section 9.6.7, we show how to use uniform random variables and the inverse distribution function to generate random variates from an arbitrary distribution function, in this case, the Type I Extreme Value Distribution [exp(-exp(-u))].

The example below uses the random number generator to make a chi-squared(3) random variable and then plots the "empirical" distribution of this random variable on the same scale as the theoretical distribution (computed by CDF). The plot is shown in **Example 14.2**.

```
SMPL 1 100 ;
RANDOM E1-E3 ;                    ? Make 3 indep. normal Rvs.
CHI3 = E1*E1+E2*E2+E3*E3 ;        ? Chi-squared variable=sum of squares of normal random vars.
SORT CHI3 ;                       ? These statements make the empirical distribution function of chi3
TREND T ;
P = T/@NOB ;                      ? p = 1/n,2/n,etc. is the height at the corresponding value chi3.
SET MAXC = 1+INT(1.1*CHI3(@NOB)); ? maxc is the upper limit of the graph.
?
? Now make the theoretical chi-squared(3) distribution function using CDF.
CHIVAL = MAXC*(T-1)/(@NOB-1) ;
CDF(CHISQ,DF=3,LOWTAIL) CHIVAL CHIDF3 ;
?
GRAPH(PAIR,LINE,TITLE="Reference vs Actual Distribution") CHIVAL,CHIDF3 CHI3,P ;
```

Most Monte Carlo analysis involves making a large loop and accumulating statistics on functions of random variables. As an illustration of how to do this in TSP, the distribution of the OLS regression coefficients can be checked under the standard assumptions of fixed Xs and normal residuals.

```
SET NTRIAL=100;  SET NOB=50;  SET NX=2;
SMPL 1,NOB;
RANDOM(MEAN=5,SEEDIN=49824) X;          ? generate X series; use SEED to reproduce results.
YHAT = 3 + 4*X;                         ? generate true values of Y series.
MFORM(NROW=NX,NCOL=1) BMEAN=0;
MFORM(NROW=NX,TYPE=SYM) BPROD=0;
REGOPT(NOPRINT) @LOGL,@COEF,@SES;       ? suppress all OLS output.
DO TRIAL = 1,NTRIAL;
 RANDOM(STDEV=2) E;                     ? generate disturbances E
 Y = YHAT+E;
 OLSQ(SILENT) Y C X;
 MAT BMEAN = @COEF + BMEAN ;            ? sum coeff. estimates.
```



Example 14.2: Cumulative Probability Plot for the Chi-squared Distribution.

```
  MAT BPROD = BPROD + @COEF*@COEF' ;          ? sum cross products.
ENDDO;
MAT BMEAN = BMEAN/NTRIAL ;
MAT BVAR = BPROD/NTRIAL - BMEAN*BMEAN' ;
REGOPT;                                       ? reset output suppression.
TSTATS(NAMES=@RNMS) BMEAN BVAR;
```

For lengthy Monte Carlo studies, it may be useful to write intermediate results occasionally to a file, so that if the program fails for some reason partial results can be obtained. This would also yield more accurate estimates of the variance (the updating formula used above is not very accurate). That is, revise the example above as follows:

```
...
DO TRIAL = 1,NTRIAL;
  RANDOM(STDEV=2) E;
  Y = YHAT+E;
  OLSQ(SILENT) Y C X;
  WRITE(FILE='monte1.dat') @COEF;
ENDDO;
?
? the second part of the program (below) could be put in a separate file if necessary
SMPL 1,NTRIAL;
READ(FILE='monte1.dat') B0 B1;
MSD(COVA) B0 B1 ;
```

```
REGOPT;                                  ? reset output suppression.
TSTATS(NAMES=(C,X)) @MEAN @COVA;         ? display mean and variance of estimated coefficients
```