SOLVE

SOLVE (CONV2=*secondary convergence criterion*, DEBUG, DYNAM, KILL,
MAXPRT=*iterations to be printed*, METHOD=GAUSS or FLPOW or JACOBI,
PRNDAT, PRNRES, PRNSIM, STATIC, TAG=*charstring* or NONE, nonlinear options)
name of collected model ;

Function:

SOLVE solves linear and nonlinear simultaneous equation models using the Gauss-Seidel method or the Fletcher-Powell method for minimization. The model is solved period by period; if a dynamic simulation (the default) is specified, the solved values for lagged endogenous variables are fed forward to later periods.

SOLVE is suitable for large, mostly linear, loosely structured models. Since the algorithms are designed to operate on the model in blocks or groups of equations, you can obtain cost savings by using SOLVE instead of SIML when your model is large but not very interrelated - for example, it includes several different sectors.

The Gauss-Seidel algorithm, which is fundamentally a recursive loop through the equations, is the least powerful of the algorithms available in TSP for model simulation. The Fletcher-Powell algorithm, which solves simultaneous blocks by minimizing the sum of squared residuals from each equation, is somewhat more powerful, but not as good as the Newton's method implementation in SIML, since it does not use an analytic Jacobian.

Usage:

To simulate a model with SOLVE, first specify all the equations of the model in normalized form, that is, with each endogenous variable appearing once and only once on the left hand side of an equation. These equations may be specified with FRML or IDENT statements. There is no difference between the two types of equations in SOLVE - for either one, the model solution tries to make the error as small as possible. There must be as many equations as endogenous variables.

After the equations are specified, form and order the model with the MODEL procedure. This procedure takes the list of equations and endogenous variables in the model and produces a collected and ordered model which is stored under a name which you supply. This is the name which should appear on the SOLVE statement.

SOLVE solves the model specified over the current SMPL, one period at a time. The starting values for the variables are chosen as follows:

1. If the variables already exist, the actual values for the current period are used as starting values, unless they are missing values.

2. If the variables do not exist and it is the first period of the simulation, the value zero is used as a starting value.

3. If the variables do not exist and this is not the first period of the simulation, the values of the last period solution are used as starting values.

SOLVE

Options:

CONV2= specifies a secondary convergence criterion which is applied to the sum of squared residuals for each block of equation after the variables have passed the standard TOL convergence test.

DEBUG/**NODEBUG** prints the endogenous variables at each iteration--for debugging recalcitrant models.

DYNAM/NODYNAM specifies dynamic simulation. Earlier solved values of lagged endogenous variables are used in place of actual values. STATIC is the alternative to DYNAM. DYNAM is the default, unless there are no lagged endogenous variables.

KILL/**NOKILL** specifies whether the simulation is to be stopped if convergence fails for any block or period. If you use the DYNAM option, you may wish to specify the KILL option since the simulated results will feed forward.

MAXPRT= the number of iterations for which printout of the residuals is to be produced. This has no effect unless PRNRES is on. In that case, the default value is 5.

METHOD= GAUSS or JACOBI or FLPOW specifies the iteration method to be used. The JACOBI method is a variation of the Gauss-Seidel method in which the endogenous variables for any simultaneous block are all updated at once at the beginning of the iteration (see Ortega and Rheinboldt, pp.217-220).

PRNRES/**NOPRNRES** prints the residuals when solution is complete for each time period. All of the residuals will be small enough to satisfy the convergence criterion.

PRNDAT/**NOPRNDAT** prints the starting values for the endogenous variables at each time period.

PRNSIM/NOPRNSIM prints a table of the solved values of the endogenous variables at the completion of the simulation.

STATIC/**NOSTATIC** specifies static simulation. Actual values of lagged endogenous variables are used, not earlier solved values.

TAG= *charstring* specifies that the solved values of all endogenous variables should be stored as series with names created by adding *charstring* to the variable names; *charstring* should be a single character, or perhaps two, to avoid creating excessively long names. (Names larger than the allowed length of a TSP name will be truncated). TAG=NONE stores under the original endogenous variable names.

The default values of the options are DYNAM, CONV2=.001, MAXPRT=5, METHOD=GAUSS, and TAG=nothing. The print options are off except PRNSIM- so a one line summary of each iteration and a table of results will be printed.

Example:

This example shows how to set up the well-known Klein Model I for simulation. At the end of this simulation, the solved variables are stored under the names CXS, IS, etc.

```
INST CX C P P(-1) W INVR C P(-1) K(-1) E(-1) TM W2 G TX ;    FORM CONS ;
INST I C P P(-1) K(-1) INVR C P(-1) K(-1) E(-1) TM W2 G TX ;    FORM INV ;
INST W1 C E E(-1) TM INVR C P(-1) K(-1) E(-1) TM W2 G TX ;    FORM WAGES ;
IDENT WAGE W = W1+W2 ;     IDENT BALANCE E=E+CX+I+G-(TX+W+P) ;
IDENT PPROD P = E-TX-W1 ;     IDENT CAPSTK K=K(-1)+I ;
```

```
LIST KENDOG I W E P CS W1 K ;
LIST KLEIN CONS WAGES BALANCE PPROD INV WAGE CAPSTK ;
MODEL KLEIN KENDOG KLEINC ;
SOLVE (TAG=S,TOL=.0001,METHOD=FLPOW) KLEINC ;
```

Output:

If no options are specified, the normal output from SOLVE begins with a title and listing of options. This is followed by a table of the data series if the PRNDAT option is on. Then comes the iteration output. If PRINT has not been specified, only convergence or non-convergence messages are printed for each block of the model as they are solved. If the PRINT option is on, a one line message for each iteration is printed, showing the iteration number, the block number, the objective function (the sum of squared residuals), and the value of the stepsize for this iteration.

If PRNRES is on, the residual error from each equation is printed for the first MAXPRT iterations on each block, and also at convergence of the block. This can help in identifying problem equations if your model is difficult to solve.

After solution of the model over the whole sample, a message is printed if the variables are being saved in data storage. Following this message a table of the results of the simulation is printed, labelled by the observation ids. This table can be suppressed by specifying the NOPRNSIM option. @IFCONV is stored as a series which contains ones if the simulation for the observation converged, and zeroes otherwise. This is useful to check the convergence. The sum of @IFCONV should be equal to @NOB if all periods converged.

Method:

The model to be solved is broken into a series of blocks by the procedure MODEL. These blocks are alternately recursive and simultaneous. A recursive block is one which can be solved simply by specifying the value(s) of a set of previously determined endogenous variables and then computing each equation in the block in turn. Each equation must depend only on endogenous variables which are input to the block or computed previously within the block.

Obviously, a recursive block is easily solved on the conditional values of the input endogenous variables. A simultaneous block, on the other hand, does not have a triangular Jacobian and thus requires either the inversion of the Jacobian or some sort of iterative technique. The two methods available in SOLVE are the Gauss-Seidel and the Fletcher-Powell. The first is simply a generalization of the method for computing recursive blocks: the equations are computed in order, each endogenous variable being evaluated in turn. Then the new values of the endogenous variables are used to start the process over again until convergence (no change in the variables) is achieved. This process works best on mildly simultaneous and fairly linear models; it does not guarantee convergence.

The criterion function for SOLVE is the sum of squared deviations of the equations. At the solution, all deviations will be zero. Away from the solution, the deviations are computed by substituting the current values of the variables into the equations and evaluating them. This sum of squared deviations is the objective function printed out by SOLVE at each iteration.

The Fletcher-Powell algorithm solves simultaneous blocks by minimizing this criterion function with respect to the endogenous variables in the block. The method uses numerical first derivatives of the objective function and a rank one updating technique to build up the second derivative matrix. See the references for further information on this method.

References:

Fletcher, R. and M. J. D. Powell, "A Rapidly Converging Descent Method for Minimization," **Comput. J.** 6 (1963), pp.163-168.

Maddala, G.S., **Econometrics,** McGraw-Hill Book Company, New York, 1977, pp.237-242.

Ortega, J.M., and W. C. Rheinboldt, **Iterative Solution of Nonlinear Equations in Several Variables,** Academic Press, New York, 1970, Chapter 7.

Pindyck, Robert S., and Daniel L. Rubinfeld, **Econometric Models and Economic Forecasts,** McGraw-Hill Book Company, New York, 1976, Chapters 10,11,12.

Theil, Henri, **Principles of Econometrics,** Wiley, New York, 1971, pp. 432-439.