SIML

SIML (DEBUG, DYNAM, ENDOG=(*list of endogenous variables*),
METHOD=NEWTON or GAUSSN, PRNDAT, PRNRES, PRNSIM,
STATIC, TAG=*charstring* or NONE, nonlinear options)
list of equation names ;

Function:

SIML solves linear and nonlinear simultaneous equation models using Newton's method with an analytic Jacobian. 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.

SIML is a more powerful and working space-intensive alternative to the SOLVE procedure. Use SIML if your model is highly nonlinear and difficult to solve with the conventional Gauss-Seidel recursive algorithms. SIML is also recommended for any small model (less than about 25 equations) because of its ease of use, requiring less setup cost than SOLVE. If the model is linear, SIML will solve it in one iteration (per time period).

Usage:

To simulate a model with the options set at default values, specify SIML with the ENDOG option to give the list of variables to be solved for and then a list of equations which are to be solved. These equations are specified earlier with FRML or IDENT statements. There is no difference between the two types of equations in SIML - for either one, the model solution tries to make the error as small as possible.

Equations for SIML can be the exact same ones which you estimated using FIML or LSQ. If you want to have linear equations in your model from OLSQ, INST, or AR1 estimation, use FORM to make them after the estimation.

N.B. There must be as many equations as endogenous variables so that the Jacobian of the model is a square matrix.

SIML solves the model specified by the equations 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.

2. If the variables do not exist and this is the first period of the simulation, the value one 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.

Options:

DEBUG/**NODEBUG** prints the endogenous variables and direction vector 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.

ENDOG= (*a list of the endogenous variables in the model*). This is the list of variables for which the model will be solved. They do not have to be predefined, unless you wish to supply starting values, or you are doing a static simulation with lagged endogenous variables.

METHOD= controls the action to be taken if the model becomes singular. For METHOD=NEWTON (the default) iteration stops at a singular point. For METHOD=GAUSSN, a generalized inverse is used (i.e., one or more variables are temporarily excluded from the model and are held constant for the iteration).

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

PRNDAT/**NOPRNDAT** prints all the endogenous and exogenous variables at the beginning of the simulation.

PRNSIM/NOPRNSIM prints a table containing the solved values of the endogenous variables.

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 names.

The default values of the options are DYNAM, METHOD=NEWTON, and TAG=nothing. The print options are all off except PRNSIM, which prints only a one line summary of each iteration and a table of results.

Examples:

This example solves the Illustrative Model described in the User's Manual:

```
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. The input data and the solved series are printed.

The next example shows how to set up the well-known Klein Model I for simulation:

```
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 CX+I+G-(TX+W+P) ;
IDENT PPROD E-P-TX-W1 ;
IDENT CAPSTK K=K(-1)+I ;
SIML (PRNRES,TAG=S,ENDOG=(CX,I,W1,W,E,P,K))
     CONS INV WAGES WAGE BALANCE PPROD CAPSTK ;
```

This model solves for CX (consumption), I (investment), W1 (wages in the private sector), W (total wage bill), E (production of the private sector), P (profits), and K (capital stock) using TM (time), W2 (government wage bill), TX (taxes), and G (government expenditures) as exogenous variables.

Output:

If no options are specified, the normal output from SIML begins with a title and listing of options. This is followed by a table of the data series if the PRNDAT option is on.

Next is the iteration output. If PRINT has not been specified, only one line per iteration is printed, showing the starting value of the objective function, the ending value, and the value of the stepsize for this iteration. Even this information is not printed if you have specified the SILENT option.

If PRINT has been specified, considerably more output is produced, showing the values of the endogenous variables and the vector of changes at each iteration. If PRNRES is on, the residual error from each equation is printed when convergence is achieved or the maximum number of iterations is reached.

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. To suppress the table, use the NOPRNSIM option. @IFCONV is stored as a series with ones if the simulation for the observation converged, and zeroes otherwise. This may be useful for a convergence check, since this information may otherwise be hidden in a large output file full of iteration information.

Method:

If a simultaneous model is linear, it can be written as $Ax = b$, where A is a matrix of coefficients, x is the set of endogenous variables, and b is a vector of numbers (which may include functions of exogenous variables). This model can be solved directly by inverting A and multiplying b by it.

Newton's method applies this idea to the iterative solution of nonlinear models in the following way: 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. The resulting set of new values is treated as a direction vector for a linear search for a "better" set of values.

The criterion function for SIML is the sum of squared deviations of the equations. At the solution, all the 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 SIML at each iteration.

TSP's implementation of Newton's method uses an analytic Jacobian evaluated at the current variable values as the matrix A; this is an extremely powerful method for finding the solution of a nonlinear model, but it can still run into trouble,

primarily because of (near)-singularity of the Jacobian. If this happens, a message is printed, and you may wish to try the GAUSSN method, which uses a generalized inverse to try to get past a locally singular point.

References:

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.

Saaty, T. L. and J. Bram, **Nonlinear Mathematics,** McGraw-Hill Book Co., New York, 1964.

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