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

The Fair-Parke program allows all the techniques discussed in this book to be used automatically once the necessary information on a model has been read by the program. The necessary information consists of a few FORTRAN subroutines and a few data sets to present the stochastic equations and the identities. Once the program has this information, almost all the techniques can be used with no further programming. (In a few cases, such as FIML estimation, the user must supply additional information. In the FIML case, for example, information on the Jacobian must be supplied. These exceptions are discussed later in this appendix.) This has the obvious advantage of allowing many things to be done with only one setup, and it also means that the model only needs to be debugged once. It is quite easy, as will be seen, to check coding errors, and once these errors have been corrected, one need not worry about further coding errors for any of the techniques.

The model is represented by (6.1), which is repeated here:

$$(6.1) \qquad f_i(y_t, x_t, \alpha_i) = u_{it}, \qquad i = 1, \ldots, n.$$

The program requires that the stochastic equations of the model be rewritten,

$$(C.1) \qquad f_i(z_t, \alpha_i) = u_{it}, \qquad i = 1, \ldots, m,$$

where z_t is a vector of variables that are transformations (generally nonlinear) of the variables in y_t and x_t. If, for example, one of the variables in an equation is $\log(y_{2t}/x_{3t-1})$, then one of the variables in z_t would equal this. A variable in z_t can simply be a variable in y_t or x_t, and thus no generality is lost in going from (6.1) to (C.1). In this notation the stochastic equations are assumed to come first in the model, although the program does not require this.

The heart of the program consists of four subroutines: ZFYX, YFZX, IDENT, and RESID. RESID is internal to the program, and the other three are user-supplied. ZFYX calculates the z variables as a function of the y and x variables. It consists of statements like Z(J,1) = DLOG(Y(J,2)/X(J − 1,3)),

where J is the time index. YFZX contains the reverse transformations from the z and x variables to the y variables that are matched to the stochastic equations. If, say, $z_{1t} = \log(y_{2t}/x_{3t-1})$ is the LHS variable of equation 1 and if y_{2t} is matched to this equation, then YFZX would contain the expression $y_{2t} = e^{z_{1t}} \cdot x_{3t-1}$, which in FORTRAN is Y(J,2) = DEXP(Z(J,1))*X(J − 1,3). YFZX contains m such statements. The z variables pertain only to the stochastic equations; they are not needed and are not used for the identities. IDENT calculates the identities. It contains the code for the identities in terms of the y and x variables, such as Y(J,6) = Y(J,5) + Y(J,4) + X(J,3). IDENT contains $n - m$ such statements.

RESID calculates the LHS z variable for each stochastic equation. If an equation is linear in coefficients except for the possible presence of serial correlation coefficients, RESID only needs to know which z variables appear in the equation. These variables can simply be listed in a data set, and therefore in this case RESID does not have to be touched by the user. If an equation is nonlinear in coefficients, RESID has to be modified for the equation. Since most equations in macroeconometric models are linear in coefficients, RESID seldom needs to be adjusted.

The reason these four subroutines are the heart of the model is that they are used by the Gauss-Seidel technique to solve the model. In the solution of the model for a given period, ZFYX is first called to get initial values for the z variables. The problem is then turned over to the Gauss-Seidel technique. One iteration (that is, one "pass" through the model) consists of successive calls to RESID, YFZX, IDENT, and ZFYX. RESID computes the LHS z variables in the stochastic equations; YFZX computes the y variables corresponding to these z variables; IDENT computes the y variables that are determined by the identities; and ZFYX computes the z variables that were not computed by RESID. The four calls are then repeated, and the process continues until convergence is reached or there is an abnormal termination. (With respect to the call to ZFYX, it does not make any difference if ZFYX computes the z variables that were already computed by RESID. Given that YFZX is called right after RESID, ZFYX merely computes the values computed by RESID back again. It is, of course, wasteful of computer time to do this, and ZFYX has an option for the relevant z variables to be skipped.) The order of the equations matters for solution purposes in that once a z or y variable is computed, this value is used in any subsequent calculations involving the variable on the RHS of the equations. The order is determined by the user in the coding of subroutines YFZX, IDENT, and ZFYX and in the numbering of the stochastic equations.

Since the techniques discussed in this book require little, if anything, more from the user than a way of solving the model, once the four subroutines are available, the rest of the programming for a technique requires little or no user intervention. One of the advantages of this feature is that one can move automatically from estimation to the use of other techniques. It is easy in the program to modify a stochastic equation (or to create a new one) and then estimate it, and the program always stores the last estimate of each equation. This means that one can modify a model, reestimate it, and then go immediately to the solution of the modified version with no extra programming.

Debugging subroutines is always a problem for large models, but the program allows this to be done fairly easily. First, given the actual data for the y and x variables, a call to IDENT should result in the predicted values of the identity-determined y variables being equal to the actual values. Since, as noted earlier, order matters in this subroutine, if an error has been made in one equation so that the predicted value of the y variable corresponding to the equation is not equal to the actual value, this error will affect the calculations of subsequent identities that use this variable. This sometimes makes it difficult to determine if an error is a coding error or the result of a previous error, and the easiest thing to do is to correct the obvious errors and run the test again. Debugging of IDENT can usually be accomplished with two or three sets of corrections.

Second, RESID can be tested in the following way. There is an option in RESID to compute either the LHS z variables in the stochastic equations or the residuals. In other words, RESID will compute either the LHS variable in equation i in (C.1) or the error term u_{it}. If the residuals are computed over the estimation period and if the actual values of the z variables are used for these calculations, then the sum of these residuals squared for each equation should equal the sum of squared residuals computed by the estimation technique at the time of estimation. This latter sum is printed by the program at the time of estimation, and thus one can check to see if the two sums are the same for each equation. To some extent this check is unnecessary, since RESID does not have to be debugged, but it is useful to make sure that the set of coefficients being used is what the user thinks it is and that no changes have been made between estimation and solution time that affect these calculations.

Finally, the entire solution process can be tested as follows. If RESID is called and if the residual computation option is used, the program computes and stores the residuals. A second call to RESID to compute the z variables will then result in the computed values of the z variables being equal to the

actual values. A call to YFZX should then produce actual values of the y variables corresponding to the stochastic equations, and a call to IDENT should produce the actual values of the y variables determined by the identities. In short, the solution values should equal the actual values when the estimated residuals are used in RESID for the solution of the z variables. If not, and if RESID and IDENT have been checked previously, then YFZX must contain one or more errors.

These three tests do not catch all errors. There may, for example, be an error in ZFYX in computing a z variable that is not a LHS variable of a stochastic equation, and this will not necessarily be caught. The tests do, however, catch most errors, so once these tests are passed, one can have some confidence that no coding errors are involved in the use of the various techniques.

Note with respect to the third test that because RESID computes residuals as well as z variables, perfect tracking solutions are easy to create and then use as a base for other experiments. This is accomplished by one call to RESID using the residual option and the actual values of the z variables. The residuals are stored and treated as exogenous for any future experiments.

The extra subroutines are that needed for some of the techniques will now be discussed. For FIML estimation, one must supply information on the Jacobian. This is done by creating a data set that consists of FORTRAN code for the nonzero derivatives (in any order). A program that accompanies the main Fair-Parke program reads this data set and creates two FORTRAN subroutines, which are then added to the main program. The program automatically takes account of the sparse structure of the Jacobian, so the user need not worry about this. Debugging the Jacobian code is a serious problem, however. If errors have been made in the code, it may still be the case that the subroutines compile and the determinants of the Jacobian are computed with no error messages. There is no obvious way to test that all the derivatives are correct. It is easy to make small errors in the code, and my suggestion is to have two people each take and code the derivatives. Two separate setup jobs can then be run, and two separate initial values of the likelihood function can be computed. If the two values are not the same, then at least one error has been made, which then requires checking the two sets of code line by line.

If there are constraints on the coefficients, such as $\alpha_{18} = \alpha_{17}\alpha_{25}$, a subroutine must be supplied that codes these constraints. The constrained coefficients are not estimated, but they are used in RESID in computing the z variables and residuals. Given the subroutine for the constraints, RESID does not have to be modified to account for them.

If an equation is nonlinear in coefficients for reasons other than because of serial correlation problems and if the equation is to be estimated by OLS or 2SLS, a subroutine must be supplied that computes the residuals for a given set of coefficients. The program uses the residuals to compute $u_i'D_iu_i$ in (6.5), which is then turned over to the DFP algorithm.

For the solution of optimal control problems, a subroutine must be supplied that computes the value of the objective function for a given set of values of the y and x variables. In other words, the user must supply a subroutine that computes W in (10.2).

Two additional subroutines are needed if the model is a rational expectations model. One subroutine creates the expectations variables from the Y and X variables. The user-supplied part of this subroutine consists merely of one line of code per expectation variable, so it requires very little work to construct. The other subroutine creates the expectations of the exogenous variables, where the assumptions that are used for this are left to the user. This subroutine does not have to be supplied if the expectations of the exogenous variables are assumed to be equal to the actual values for all variables.

These are the main additional subroutines. A few others are required for some of the options, but they are not of general interest here. A final point to emphasize about the program is that it allows successive reestimation and stochastic simulation to be done with virtually no extra work on the part of the user. One number indicates how many times the estimation or simulation is to be done. Because of the emphasis in this book on the comparison method in Chapter 8, which requires successive reestimation and stochastic simulation, the program was written to make this as easy as possible.