OPTIMAL CONTROL OF NONLINEAR SYSTEMS PROGRAM USER'S GUIDE

by

Gregory C. Chow
Ettie H. Butters

Econometric Research Program
Princeton University

Research Memo #209

April 1977

Econometric Research Program
PRINCETON UNIVERSITY
207 Dickinson Hall
Princeton, New Jersey

OPTIMAL CONTROL OF NONLINEAR SYSTEMS PROGRAM

USER'S GUIDE

Ettie H. Butters
Gregory C. Chow

Table of Contents

The optimal control of nonlinear systems program (OPTNL) computes the optimal control policy and the associated welfare cost using a quadratic loss function for an econometric model whose parameters are assumed to be known. The algorithm used is described in Sections 12.1 and 12.4 of G.C. Chow, Analysis and Control of Dynamic Economic Systems (John Wiley and Sons, 1975).

I. Eliminating Second and Higher Order Lags in the Model

Let the model be written as a system of simultaneous structural equations.

$$y_t = \phi(y_t, y_{t-1}, x_t, x_{t-1}, w_t) + u_t \qquad (1)$$

where y_t = vector of 'ny' endogenous variables

x_t = vector of 'nx' control variables

w_t = vector of 'nw' exogenous variables not subject to control

npd = number of time periods in the planning horizon

u_t = vector of random error terms, and

where ϕ is a vector of possible nonlinear functions.

Since the program does not accept lagged endogenous variables of order higher than the first, the user should eliminate the variables with lags of two or more periods by introducing identities. For example, let there be 50 endogenous variables in the model to begin with. The number of simultaneous equations 'ns' is 50. If the model consists of $y_{6,t-2}$, an identity $y_{51,t} = y_{6,t-1}$ should be introduced. This identity permits the user to write $y_{6,t-2}$ as $y_{51,t-1}$ and get rid of the second-order lag. If $y_{6,t-3}$ is also present, another identity $y_{52,t} = y_{51,t-1}$ can be used, permitting the user to write $y_{6,t-3}$ as $y_{52,t-1}$. Let 40 additional identities of this kind be required in our example to eliminate all endogenous variables with lags of two or more periods. There will then be 90 endogenous variables in the model, to be included in the vector y_t.

If the model consists of control variables lagged two or more periods, more identities and endogenous variables will be required. Let there be $nx = 4$ control variables in our example, $x_{1t},\dots,x_{4,t}$. To eliminate the variable $x_{1,t-2}$, introduce the identity $y_{91,t} = x_{1,t-1}$ and write $x_{1,t-2}$ as $y_{91,t-1}$. Similarly, to eliminate $x_{1,t-3}$, introduce the identity $y_{92,t} = y_{91,t-1}$ and write $x_{1,t-3}$ as $y_{92,t-1}$. If 10 identities of this type are required, there will be all together 100 variables in the vector y_t. 'ny', the number of endogenous variables in y_t, will be set equal to 100.

The computer program will automatically make up a vector consisting of the 104 elements of y_t and x_t. This augmented vector will serve as the argument in the welfare function, its last 4 elements in our example being included to serve the possible need to penalize variations in the instruments or control variables. The expanded model takesthe form

$$\left.\begin{array}{l} y_{1,t} = \phi_1(y_t,y_{t-1},x_t,x_{t-1},w_t) + u_{1,t} \\ \vdots \qquad\quad \vdots \qquad\qquad\qquad\qquad\qquad \vdots \\ y_{50,t} = \phi_{ns}(y_t,y_{t-1},x_t,x_{t-1},w_t) + u_{50,t} \end{array}\right\} \text{ns structural equations}$$

$$\left.\begin{array}{l} y_{51,t} = y_{6,t-1} \\ \vdots \qquad\quad \vdots \\ y_{100,t} = y_{99,t-1} \end{array}\right\} \text{nid identities of lagged values} \qquad (2)$$

$$\left.\begin{array}{l} y_{101,t} = x_{1,t} \\ \vdots \qquad\quad \vdots \\ y_{104,t} = x_{4,t} \end{array}\right\} \text{nx identities of control variables}$$

There are ns simultaneous structural equations, nid identities of lagged values, and nx identities for the control variables; altogether the augmented y vector has p elements, with $p = ny + nx$, and $ny = ns + nid$. u_t is a random vector with mean zero and covariance matrix V, to be supplied by the user. The program calculates a linear approximation of the model explaining the augmented y vector shown in equation (2)

$$y_t = A_t y_{t-1} + C_t x_t + b_t \ ,$$

from which it derives a feedback control equation

$$x_t = G_t y_{t-1} + g_t$$

so as to minimize the expectation of the welfare loss

$$E_o W = E_o \sum_{t=1}^{npd} (y_t - z_t)' K_t (y_t - z_t) \ ,$$

where npd is the number of periods, z_t is n vector of targets, and $K_t = K \cdot EXKCAP^t$ is the weighting matrix. npd, z_t, EXKCAP and K are to be specified by the user.

The model in the form shown in equation (2) is coded in FORTRAN statements in two subroutines provided by the user. The first subroutine, named MODELS, consists of FORTRAN statements of the ns simultaneous structural equations; the second, named MODELI, consists of FORTRAN statements of the nid identities of the lagged values. The last nx identities of the control variables are not coded by the user, but are 'remembered' by the program. Some knowledge of basic FORTRAN is therefore required by the user. A description of the two user-supplied subroutines is contained in section IV.

II. Description of the Program

The general logic of the program, represented in the flow diagram of Figure 1, is taken directly from section 12.1 of Chow's text. A description for some of the steps follows.

1. Each iteration starts at A of the diagram, and depends on the optimal path computed in the previous iteration. To provide a tentative path of y_t for the first iteration, the program starts with the given trial values of x_t, $(t=1,\ldots,npd)$ and solves the nonlinear system using the Gauss-Seidel method which is described in section 6.6 of Chow. For the model of equation (2), let $y_{it}^{(k)}$ be the values of y_{it} at the kth iteration of the Gauss-Seidel process. The iterating process continues until y_{it} converges

$$\left| \frac{y_{it}^{(k+1)} - y_{it}^{(k)}}{y_{it}^{(k)}} \right| < \varepsilon \text{ for } i = 1,\ldots,ns.$$

   The method requires the user to supply an initial y_o vector, trial y_1 and x_1 vectors in period 1 for the first Gauss-Seidel iteration, the maximum number, r, of iterations permitted, and the convergence criterion ε.

2. Instead of estimating the tentative path by the Gauss-Seidel method, the program can, if so directed by the user, accept a set of y_t and x_t from an output device as the tentative path to be used for the first iteration. These y_t and x_t are optimal solutions from a previous job, and are being used now to continue the iteration process. At the end of each iteration cycle, the program automatically saves the solutions y_t and x_t in an output device, regardless of whether or not y_t has converged. The purpose is to safeguard against

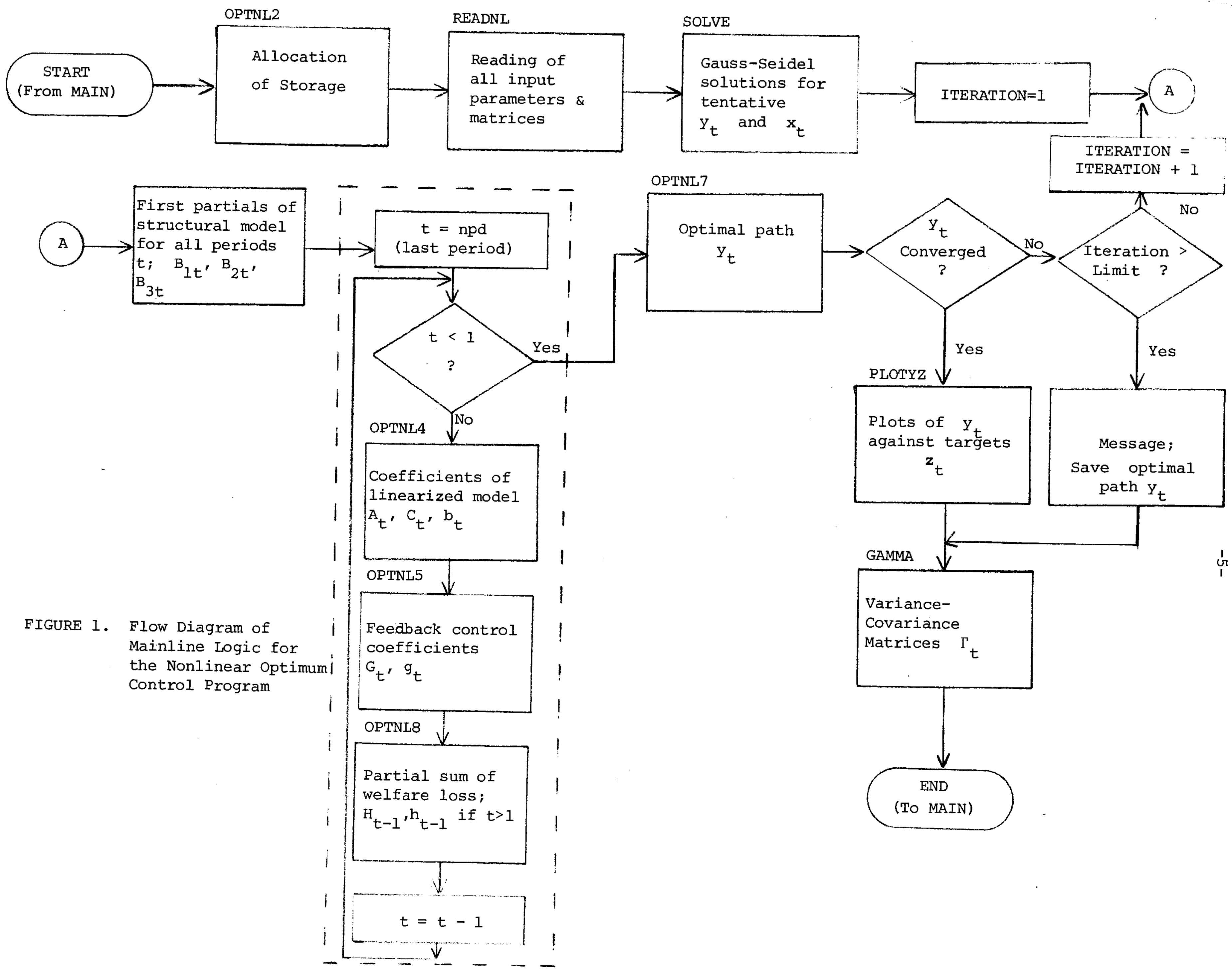

START (From MAIN)
OPTNL2
Allocation of Storage
READNL
Reading of all input parameters & matrices
SOLVE
Gauss-Seidel solutions for tentative y_t and x_t
ITERATION=1
A
ITERATION = ITERATION + 1
A
First partials of structural model for all periods t; B_{1t}, B_{2t}, B_{3t}
t = npd (last period)
t < 1 ?
Yes
No
OPTNL4
Coefficients of linearized model A_t, C_t, b_t
OPTNL5
Feedback control coefficients G_t, g_t
OPTNL8
Partial sum of welfare loss; H_{t-1}, h_{t-1} if t>1
t = t - 1
OPTNL7
Optimal path y_t
y_t Converged ?
No
Iteration > Limit ?
No
Yes
Yes
PLOTYZ
Plots of y_t against targets z_t
Message; Save optimal path y_t
GAMMA
Variance-Covariance Matrices Γ_t
END (To MAIN)

FIGURE 1. Flow Diagram of Mainline Logic for the Nonlinear Optimum Control Program

losing all the results of the iterative process before the job is completed such as when the job is terminated because the estimated time is exceeded or the system crashes. What has been saved can then be used for the next iteration in a later job.

3. The program computes the first partial derivatives of ϕ_j numerically. For example, the derivative of ϕ_j with respect to the variable y_{it} is

$$\frac{\partial \phi_j}{\partial y_{it}} = \frac{y^1_{it}-y^2_{it}}{2dy_i} ,$$

where

$$y^1_{it} = \phi_j(y_{1t},\ldots,y_{it}+dy_i,\ldots,y_{ny,t};y_{t-1};x_{t-1};x_t;w_t) + u_t$$

$$y^2_{it} = \phi_j(y_{1t},\ldots,y_{it}-dy_i,\ldots,y_{ny,t};y_{t-1};x_{t-1},x_t;w_t) + u_t$$

$$dy_i = \max(|FDFRAC \cdot y_{it}|,FDMIN) .$$

FDFRAC, FDMIN are provided by the user.

III. Description of Time and Space Saving Procedures

The nonlinear model is constructed in the particular form shown in (2) in order to save both computation time and core space. In that form the matrices B_{1t}, B_{2t}, and B_{3t} of the model's first partial derivatives where, following the notation in Chow,

$$B'_{1t} = \left(\frac{\partial\phi_1}{\partial y_t} \cdots\cdots \frac{\partial\phi_p}{\partial y_t}\right)$$

$$B'_{2t} = \left(\frac{\partial\phi_1}{\partial y_{t-1}} \cdots\cdots \frac{\partial\phi_p}{\partial y_{t-1}}\right)$$

$$B'_{3t} = \left(\frac{\partial\phi_1}{\partial x_t} \cdots\cdots \frac{\partial\phi_p}{\partial x_t}\right)$$

take the following form:

$$B'_{1t} = \left(\frac{\partial\phi_1}{\partial y_t} \cdots \frac{\partial\phi_{ns}}{\partial y_t}\; 0 \cdots\cdots 0\right)$$

$$B'_{2t} = \left(\frac{\partial\phi_1}{\partial y_{t-1}} \cdots \frac{\partial\phi_{ns}}{\partial y_{t-1}}\; d_1 \cdots d_{nid}\, 0 \cdots 0\right)$$

$$B'_{3t} = \left(\frac{\partial\phi_1}{\partial x_t} \cdots \frac{\partial\phi_{ns}}{\partial x_t}\; 0 \cdots 0\; I\right)$$

where d_k is a vector of all zeroes except a single element of one, and where I is an identity submatrix of dimension nx. Since all except the first ns columns have only either zeroes or ones, the program computes the partial derivatives only for the ns structural equations. The location of the element one in each d_k column of B'_{2t} is informed by an input vector named IDENTVEC, where IDENTVEC(k) = j indicates that the d_k column has an element one in its jth row, the rest of the column being zero.

Some more time may be saved by the program in computing the partial derivatives for the ns structural equations $\phi_1,\ldots,\phi_{ns}$. If a variable such as $y_{8,t-1}$ is not in any of the equations $\phi_1,\ldots,\phi_{ns}$, the program will not waste time in computing the numerical derivative with respect to that variable, provided that it is known in advance that the derivative is zero. Such variables are identified by an input vector named VARBLVEC, supplied by the user. VARBLVEC has $ns + p + nx$ entries, each being either zero or nonzero, corresponding to the following variables:

$$y_{1,t},\ldots,y_{ns,t};\ y_{1,t-1},\ldots,y_{ny,t-1},x_{1,t-1},\ldots,x_{nx,t-1};\ x_{1,t},\ldots,x_{nx,t}\ .$$

Derivatives of $\phi_1,\ldots,\phi_{ns}$ with respect to these variables are computed to form the matrices B_{1t}, B_{2t} and B_{3t}. If a variable in the above list appears nowhere in the model $\phi_1,\ldots,\phi_{ns}$, the user enters the value zero in its entry in VARBLVEC; otherwise he or she enters a nonzero value as described in section V.4. The program computes and utilizes the partial derivatives with respect to only those variables with nonzero values in their corresponding entries in VARBLVEC; it automatically treats the remaining columns in B_{1t}, B_{2t} and B_{3t} as columns of zeroes.

Another procedure used by the program to save time and core space is by "compressing" the linearized approximation of the first ns equations of the model. The user does not need to be concerned with this process, since it is done entirely within the program without the user's intervention. The program rearranges the endogenous variables in y_t^s, the first ns elements of y_t, into two catagories which we shall call y_t^a and y_t^b. y_t^b consists of variables that have the following characteristics:

1. they are not targetted in the weighting matrix K;
2. they have no lagged values in the model.

Because of these two characteristics, it is unnecessary to compute the optimal path for y_t^b. y_t^a consists of all the remaining variables of y_t^s. In this arrangement the linearized reduced form equations become

$$\begin{bmatrix} y_t^a \\ \hline y_t^b \end{bmatrix} = \begin{bmatrix} A_t^a & 0 \\ \hline A_t^b & 0 \end{bmatrix} \begin{bmatrix} y_{t-1}^a \\ \hline y_{t-1}^b \end{bmatrix} + \begin{bmatrix} C_t^a \\ \hline C_t^b \end{bmatrix} \begin{bmatrix} x_t \end{bmatrix} + \begin{bmatrix} b_t^a \\ \hline b_t^b \end{bmatrix}$$

The program computes only A_t^a, C_t^a and b_t^a of the "compressed" model; it computes the feedback control coefficients G_t and g_t and the welfare loss based on the compresed model, as shown in the flow diagram of Figure 1 where these steps are enclosed by the dotted rectangle.

IV. User-Supplied Subroutines

To run the nonlinear optimal control program three subroutines must be provided by the user. They are the main program, MODELS and MODELI.

1. Main Program

The main program has to be coded by the user in order to allocate the right amount of space for his model. It calls in the optimal control program package and should be coded as follows:

```
      DIMENSION MAINAR(m)
      REAL*8 DWORD
      COMMON/IOBLK/INPUT,LIST,LCNT,NPG,INOUT2,INOUT3,
     1     INOUT4,INOUT5
      EQUIVALENCE (DWORD,MAINAR(1))
      INPUT=5
      LIST=6
      INOUT2=n2
      INOUT3=n3
      INOUT4=n4
      INOUT5=n5
      LCNT=99
      NPG=0
      NDIM=m
      MAINAR(2)=2
      MAINAR(1)=NDIM
      CALL OPTNL1(MAINAR,NDIM)
      STOP
      END
```

All capital letters and numbers must be coded as shown; the lower case letters represent variables whose values are to be supplied by the user. The following explains some of the symbols:

MAINAR	is an array from which the program assigns all storage space.
m	is the dimension of MAINAR; its formula is given in section VI; note that the number m appears in two places: DIMENSION MAINAR(m) and NDIM=m.
INPUT=5	device unit 5, which usually refers to the card-reader, is assigned to the input data set INPUT. If the input data is on tape or disk instead of in cards, another unit number should be assigned to INPUT.
LIST=6	device unit 6, which usually refers to the printer, is assigned to the output data set LIST. If the printed output is to be first stored on tape or disk, another device unit should be assigned.

INOUT2	the program writes in this data set the optimal solutions y_t and x_t at the end of each iteration; each new set of y_t and x_t replaces the one from the previous iteration. This data set should be kept by the user in case the iteration process is to be continued in a later job.

Three temporary data sets are used by the program to store intermediate results which are to be read by a later part of the program; they are INOUT3, INOUT4 and INOUT5 and they should be deleted at the end of each job.

INOUT3	stores the first partial derivatives B_{1t}, B_{2t} and B_{3t}
INOUT4	stores G_t, g_t for each period t to be used in the feedback control equation $x_t = G_t y_{t-1} + g_t$.
INOUT5	stores $A_t + C_t G_t$ and V_t for each period to be used for the computation of the covariance matrices Γ_t.
n2,n3,n4,n5	are device unit numbers for the temporary data sets assigned by the user; they should be single digit numbers other than 5, 6 and 7.

LCNT=99
NPG=0
	initialize line count and page number

NDIM	the dimension of the main array MAINAR.
OPTNL1	the mainline logic subroutine of the nonlinear optimal control package.

2. MODELS

As described in section I the nonlinear model is coded in two subroutines MODELS and MODELI. MODELS contains the ns simultaneous equations of the model, and it should be coded as follows:

```
SUBROUTINE MODELS(NY,NX,NW,D,Y,YL,XL,X,W,*)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION D(NY),Y(NY),YL(NY),XL(NX),X(NX),W(NW)
D(1) = φ1 (Y,YL,XL,X,W)  )
D(2) = φ2 (Y,YL,XL,X,W)  )
  .                      )
  .                      } ns simultaneous equations
  .                      )
RETURN
END
```

where NY, NX and NW are defined in section V.1, and where

D	is the array that contains the new values of the endogenous variables resulting from the calculations,
Y	is the vector of endogenous variables,
YL	is the vector of lagged endogenous variables,
X	is the vector of control variables,
XL	is the vector of lagged control variables, and
W	is the vector of exogenous variables; if there are no exogenous variables in the model, W and NW should nevertheless be coded as dummy parameters.

The right-hand side of the equations, represented above by $\phi_1, \phi_2, \ldots,$ are the algebraic FORTRAN statements; for example,

```
D(1) = Y(5) + DLOG(YL(1) + XL(3)) -  100.*W(4)/Y(6) + .085
```

The following rules should be observed in coding these statements:

(a) All mathematical functions should be written in double precision since the program is written in double precision.

(b) The order of all lags should be one.

(c) If the lagged endogenous variable $y_{6,t-3}$ is in the original structural equations, then as illustrated in the example of section I, two variables of lagged values are defined to eliminate this higher order lag: $y_{51,t} = y_{6,t-1}$ and $y_{52,t} = y_{51,t-1}$. $y_{6,t-3}$ should then be coded in the subroutine as YL(52) which stands for $y_{52,t-1}$.

(d) If $y_{6,t-2}$ is in the original structural model, it should be coded as YL(51) and <u>not</u> as Y(52). In the right-hand side of the structural equations the subscripts of the Y vector should be confined to the range from 1 to ns; in our example, the range is 1 to 50.

The '*' as it appears in the last argument of SUBROUTINE MODELS is an optinal facility for non-standard return, and may be used in the following way. For example, let 10^{-6} be a poor value for the variable D(2), and if it is reached then the trial solution is considered too far out. If we would like to terminate the program at this point, we can add the statements below the listing of all the equations in the SUBROUTINE MODELS:

```
      .
      .
      .
      .
      IF (D(2).LT.1.D-6) GO TO 31
      RETURN
   31 RETURN1
      END
```

MODELS is called by two subroutines in the nonlinear program: SOLVE, which solves for y_i by the Gauss-Seidel method, and FP, which takes the first partial derivatives of ϕ. SOLVE requires that new values of y_i be used for the calculation of y_j, $j > i$, thus it provides one array, Y, for the two dummy variables D and Y in MODEL. FP, however, solves for each y_i after a variable on the right hand side of a structural equation is perturbed, using the old Y vector, thus providing separate arrays for the dummy variables D and Y in MODELS, and it expects the solution from D. To accomplish the double purposes required by SOLVE and FP the MODELS subroutine must be compiled by the FORTRAN G compiler.

3. MODELI

The second subroutine for the nonlinear model, named MODELI, consists of all the identities of lagged values as shown in equation (2); it should be coded as follows:

```
SUBROUTINE MODELI (NY,NX,NW,D,Y,YL,XL,X,W,*)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION D(NY), Y(NY),Y(NY),YL(NY),XL(NX),X(NX),W(NW)
D(ns+1)=YL(i_1)
D(ns+2)=YL(i_2)
   .
   .
   .
D(ny)=YL(i_nid)
RETURN
END
```

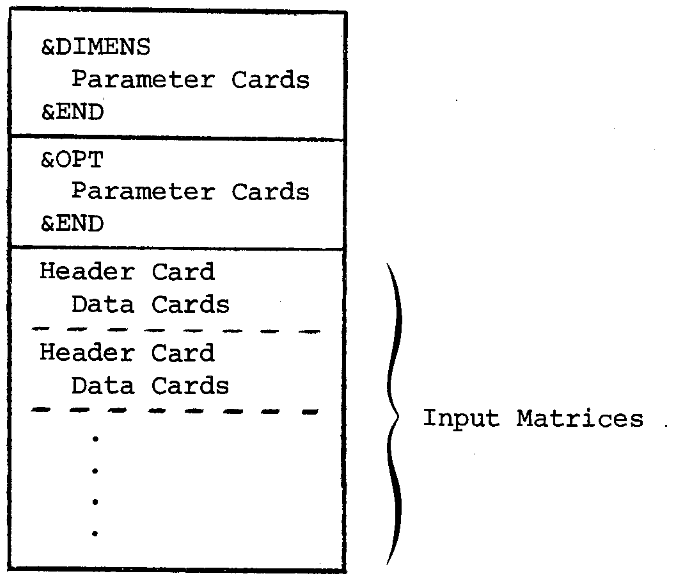
The actual values for ns+1,...,ny and for $i_1,\ldots,i_{nid}$ will be supplied by the user. For example, using the illustration from section I,

```
D(51)=YL(6)
D(52)=YL(51)
   .
   .
   .
```

All parameters are the same as defined in MODELS, as is the use of the '*' feature. MODELI is used by only one subroutine of the program, SOLVE. As was explained earlier for MODELS, MODELI should be compiled by the FORTRAN G compiler.

V. Data Requirements

The program accepts data in the sequence represented by the diagram below:



&DIMENS and &OPT are names for two blocks of parameters to be read in by the NAMELIST feature of FORTRAN. To illustrate the use of this feature, consider the following example, taken from the model discussed in section I:

```
_&DIMENS_NS=50,NY=100,NX=4
_        NW=40,NPD=5
_END
```

All imput cards for a NAMELIST block must start at a column to the right of column 1. The first card must start with the name, in this case &DIMENS, followed by the parameters for that list. The parameters are separated by commas; spaces are allowed between parameters, but no space is allowed on either side of the '=' sign. A parameter not included in the input is given its default value by the program, therefore it may not be necessary to list all the parameters for a NAMELIST block. The block ends with an &END card. Note in the example '_' means one or more blanks.

The input matrices are read in blocks, each being preceded by a header card which has the following format:

cols 1-8:	name of the matrix, left justified. The name must be spelled exactly as given in subsection 3 below.
cols 9-40:	optional format in FORTRAN convention for the data following this header card. The default format is 8F10.4.
cols 41-80:	optional comments describing the matrix.

All data matrices except V and KCAP are to be read in row-wise, following the format specified in their respective header cards. Detailed description for each input matrix will be given in subsection 3 below. The following example illustrates an input matrix:

```
 ↙col. 1      ↙col. 9          ↙col. 41
ZRATE        (5F5.2/2F5.2)    GROWTH RATE FOR TARGETS
1.0    1.0   1.5    1.0   1.35
1.7    1.0
1.62   1.2   1.0    1.0   1.0
1.0    1.0
```

1. <u>&DIMENS</u>

The NAMELIST &DIMENS consists of the following parameters:

NS = number of structural equations in the model; default = 1.

NY = number of endogenous variables of the vector y_t in equation (2); i.e., NY = NS + the number of identities of lagged values; default = 1.

NX = number of control variables in the vector x_t in equation (2); default = 1.

NW = number of uncontrollable exogenous variables in the vector w_t in (2); default = 0.

NPD = number of time periods for the plan; default = 1.

2. <u>&OPT</u>

The &OPT list consists of parameters that are options for the program; they are:

OSUP	= 0	for full print-out of A_t, C_t, b_t, G_t, H_t, g_t, h_t at every time period. (These symbols are defined in Chow's book as referenced.)
	= 1	for print-out of G_t, H_t, g_t, h_t only at every time period.
	= 2	for print-out of G_1, H_1, g_1, h_1 at time period 1 only; all other time periods have no print out.
	= 3	same as in OSUP=2 except that the program will also omit printing H_1 and h_1.
GAUSS	= 0	will omit the printing of the Gauss-Seidel solution for y_t used for the first linearization of the model (default value).
	= 1	will print the above Gauss-Seidel solution.
	= 2	will print and save the Gauss-Seidel solution and the program will terminate at that point.
GAMMA	= 0	will omit the calculation of the covariance matrices Γ_t (default value).
	= 1	when the optimal solution y_t converges* the program will calculate the covariance matrices Γ_t of the variables y_t. The program will not print the entire Γ_t but will print only those rows of Γ_t that correspond to the variables targetted in the weighting matrix K .
	= n	where $2 \leq n \leq ns+nx$. As in GAMMA=1 the program will calculate Γ_t when the optimal solution y_t converges*; the print-out of Γ_t is controlled by the input vector GAMVAR, which is to be supplied by the user; n indicates the number of entries in GAMVAR.
PLOT	= 0	will plot the optimal solution values of the meansof y_t against target values z_t only for those variables that have nonzero diagonal weights in the K matrix; the means of the control variables x_t are also plotted (default value).
	= 1	will plot the means of all the variables of the optimal solution y_t as well as the control variables x_t against their targets z_t.
UGZ	= T	if the targets z_t are to grow at constant percentage rates; in this case the user will supply the initial z_o, the ZO vector, and the rate of growth, the ZRATE vector, and the program will compute z_t = ZO • ZRATE.
	= F	if the targets z_t for all periods are to be supplied by the user in the input matrix Z (default value).

* The program will also compute Γ_t when the iteratim limit ITERL1 is reached.

OFDIAV	= T	if the covariance matrix V of the random vector u_t in equation (2) have nonzero off-diagonal elements; in this case the user must supply the lower triangle of the symmetric V.
	= F	if the off-diagonal elements of V are zero; only the diagonal elements of V are required for input (default value).
OFDIAG	= T	if the K matrix in the loss function have nonzero off-diagonal elements; in this case the user must supply the lower traingle of the symmetric K matrix.
	= F	if the off-diagonal elements of K are zero; only the diagonal elements of K are required for input (default value).
EXKCAP	=	discount factor for modifying the K matrix for each time period t according to the formula $K_t = K \cdot (\text{EXKCAP})^t$, where K is supplied by the user; default = 1.0.
NROUND	= 1	if the tentative path for y_t required in the first iteration is to be estimated by the Gauss-Seidel method; this marks the first job run in the iterative process to calculate the optimal path y_t which might require more than one job run (default value).
	= 2	(or 3, 4, etc.) marks the second (or third, fourth, etc.) job run in the iterative process to find the optimal path y_t; the solution for y_t and x_t from the previous job will be read from an output device named INOUT2, to serve as the tentative path for the first iteration of the current run. All other input data remain the same as in the previous job. (See section VII on the use of more than one job run.)
ITERL1	=	the maximum number of times the model will be linearized for calculating optimal control path; default = 1. Our method starts with a guess for $x_1,\ldots,x_{npd}$, which, by the use of the Gauss-Seidel method, if NROUND=1, implies a solution for $y_1,\ldots,y_{npd}$. Around this tentative path the model is linearized and an optimal control path is obtained for the linearized model. This optimal path becomes the initial guess of $x_1,\ldots,x_{npd}$ in the next linearization. ITERL1 refers to the maximum number of times the model will be linearized.
ITERL2	=	the maximum number of iterations allowed for solving the model by Gauss-Seidel; default = 10.
EPS1	=	the convergence criterion for optimal policy; default = .001.

When $\left|\dfrac{y_i^{(k)} - y_i^{(k-1)}}{y_i^{(k-1)}}\right| \leq \text{EPS1}$, for each $i = 1,\dots,ns$ that corresponds to a nonzero diagonal element of the K matrix, k being the iteration count, the program terminates. The solution y_{it} will be plotted against their targets z_{it}. If the convergence criterion is not met another iteration will be performed.

EPS2 = convergence criterion for the solution of the nonlinear model by the Gauss-Seidel method; default = .001. Gauss-Seidel solves the system of equations (1) until

$$\frac{y_i^{(k+1)} - y_i^{(k)}}{y_i^{(k)}} \le \text{EPS2}$$

for $i = 1,\dots,ny$, k being the iteration count.

FDFRAC = parameter for computing step sizes in evaluating the first derivatives; the step size for the variable y_{it} is $dy_i = \max(|\text{FDFRAC}\cdot y_{it}|, \text{FDMIN})$; default = .001.

FDMIN = minimum step size allowed in the above formula; default = .001.

DAMP = a factor used to dampen the changes in successive iterations of the Gauss-Seidel solution.

$$y_t^{(k+1)} = y_t^{(k)} + \text{DAMP} \cdot (y_t^{'(k+1)} - y_t^{(k)})$$

where $y_t^{'(k+1)}$ = solution obtained at iteration k+1, and

$y_t^{(k+1)}$ = solution actually used to compute y_t in the (k+1)th iteration.

'DAMP' may be made small if solution tends to oscillate from iteration to iteration; default = 1.0.

DAMPX = a factor used to dampen the changes in successive iterations of the control variables x_t.

$$x_t^{(k+1)} = x_t^{(k)} + \text{DAMPX} \cdot (x_t^{'(k+1)} - x_t^{(k)})$$

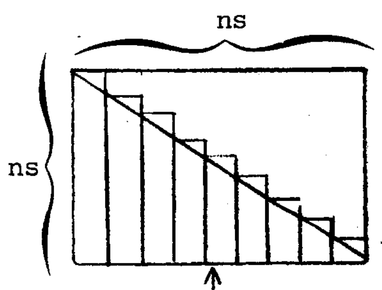
where $x_t^{'(k+1)}$ = solution obtained at iteration k+1, and

$x_t^{(k+1)}$ = solution actually used to compute the optimal path y_t for the (k+1)th iteration.

'DAMPX' may be made small if the optimal path y_t tends to oscillate from iteration to iteration; default = 1.0.

3. Input Matrices

The input matrices described below must appear in the input deck in the order given; the name of the matrix must be spelled correctly in the header card. IDENTVEC, VARBLVEC and GAMVAR are the only vectors whose elements are integers; all other vectors have floating-point numbers as elements. The dimension p is defined as p = ny + nx.

NAME	DIMENSION	DESCRIPTION
IDENTVEC	(nid)	A vector where elements are the variables that appear in the identities of lagged values in the model; its construction is described in subsection 4 below.
VARBLVEC	(ns+p+nx)	A vector that identifies those variables with respect to which the first partial derivatives are computed by the program; its construction is described in subsection 4 below.
V	(ns) if OFDIAV=F (ns(ns+1)/2 if OFDIAV=T  The lower triangle is entered as data if ODFIAV=T	Variance-covariance matrix of u_t in equation (2); if OFDIAV=F, only the diagonal of V is entered in the input; if OFDIAV=T, the lower triangle of V is entered by columns, starting with the left-most column of ns elements; each succeeding column has one less element than the previous one.
KCAP	(p) if OFDIAG=F (p(p+1)/2) if OFDIAG=T	The weighting matrix K in the welfare function; if OFDIAG=F, only the diagonal of K is entered as input; if OFDIAG=T, the lower triangle is entered by columns, as described for V above.
Z0	(p)	Initial values of targets for y_t and x_t; required only if UGZ=T.
ZRATE	(p)	Growth rate for targets which the program will compute based on Z0 and ZRATE; required only if UGZ=T.

NAME	DIMENSION	DESCRIPTION
Z	(npd,p)	Targets to be read in row-wise for all periods; required only if UGZ=F.
Y0	(p)	Initial values for the vectors y_t and x_t in period $t = 0$.
W	(npd,nw)	Exogenous variables for all periods, to be read in row-wise; required only if $nw > 0$.
X	(npd,nx)	A tentative policy for all periods, to be read in row-wise. This policy will be used to compute the tentative path y_t by Gauss-Seidel.
Y	(ny)	A trial solution of the y vector for the first period, to be used in the first iteration of the Gauss-Seidel solution for period 1; this vector may take the same value as Y0.
GAMVAR	(n)	A vector whose elements are the row/column numbers of Γ_t to be printed; required only if GAMMA=n, where $2 \leq n \leq ns+nx$ and is the dimension of GAMVAR.

4. Construction of IDENTVEC and VARBLVEC

The use of IDENTVEC and VARBLVEC is discussed in section IV. Before constructing IDENTVEC, the user must first eliminate all higher order lags in the nonlinear model by adding identities of lagged values to the model, as discussed in section I. To illustrate, we will use the example from section I. In this example the model has 50 simultaneous structural equations, 50 identities of lagged values, and 4 control variables; that is, ns=50, nid=50, ny=ns+nid = 100, and nx=4, and the identities of lagged values are:

$y_{51} = y_{6,t-1}$	6
$y_{52} = y_{51,t-1}$	51
$y_{53} = y_{52,t-1}$	52
.	.
.	.
.	.
$y_{91} = x_{1,t-1}$	101
$y_{92} = y_{91,t-1}$	91
$y_{93} = x_{3,t-1}$	103
.	.
.	.
.	.
$y_{100} = y_{99,t-1}$	99

The numbers listed on the right are the variable numbers that will appear in IDENTVEC: for $y_{k,t-1}$ in an identity, the number listed is simply k; for $x_{k,t-1}$, the number listed is $ny+k$. The IDENTVEC for this model will look like this:

```
                        col. 1        col. 9
                       ↙             ↙
Header
Card      ——>    IDENTVEC          2014
Data
Card 1    ——>       6     51    52 . . .
                 .    .    .    .    .    .    .
Data
Card 3    ——>    101    91   103 . . . 99
```

Note that the data entered are 'right-justified' because the format is I, and that the dimension of IDENTVEC is nid, the number of identities of lagged values.

The construction of VARBLVEC is a bit more involved, but the user should not have too much trouble if the steps listed below are followed:

(a) List the integers from 1 to ns+p+ns. In our example from Section I, the numbers would be from 1 to 158.

(b) Above this list of numbers write the following variables in order, with one variable corresponding to one number:

$$y_{1,t},\dots,y_{ns,t}$$

$$y_{1,t-1},\dots,y_{ny,t-1},x_{1,t-1},\dots,x_{nx,t-1}$$

$$x_{1,t},\dots,x_{nx,t}$$

In our example, we would have:

$$\begin{array}{cccc} y_{1t} & y_{2t} & \cdots & y_{50,t} \\ 1 & 2 & \dots & 50 \end{array}$$

$$\begin{array}{cccccc} y_{1,t-1} & y_{2,t-1} & \cdots & y_{100,t-1}, & x_{1,t-1},\dots, & x_{4,t-1} \\ 51 & 52 & \dots & 150 & 151 \quad \dots & 154 \end{array}$$

$$\begin{array}{ccc} x_{1t} & \dots & x_{4t} \\ 155 & \dots & 158 \end{array}$$

(c) Look through the right-hand side of each structural equation of the model. Whenever one of the variables listed in (b) appears, circle the number underneath the variable in the list. In our example, we might obtain

$$\begin{array}{lllll} \textcircled{1} & \textcircled{2} & 3 & \textcircled{4} & \dots 50 \\ 51 & \textcircled{52} & \textcircled{53} & \dots & \\ \vdots & & & & \\ \textcircled{155} & 156 & \textcircled{157} & \textcircled{158} & \end{array}$$

Any number in the list not circled means the variable above it appears <u>nowhere</u> in the right-hand side of the structural equations; any variable of the list in (b), that appears <u>somewhere</u> in the right-hand side of the structural equations should have its corresponding number circled. In our example, y_{3t}, y_{50t}, $y_{1,t-1}$, $x_{2,t}$ are some of the variables not in the structural model.

(d) VARBLVEC is simply the list of integers from (b), with the uncircled numbers replaced by 0; thus VARBLVEC has ns+p+nx entries. In our example, VARBLVEC would look like this:

```
                 col. 1    col. 9                      col. 40
                   ↓         ↓                            ↓
Header                                                    |
Card             VARBLVEC  10I4                           |
                                                          |
Data                                                      |
Card 1             1   2     0   4 . .. .  ....  .  .     |
                   .                                      |
                   .                                      ↓
Data               .
Card 5             .  .  .  .  .   .  .  .  .  .  .  .  . 0

Data
Card 6             0  52    53 .   .  .  .  .  .  .  .  .
                   .
                   .
Data               .
Card 16            .  .  .  .  .   .  . 155   0  157  158
```

It is vitally important that IDENTVEC and VARBLVEC be constructed correctly since the program depends on them to calculate the optimal solution y_t. If the econometric model is not very large or the saving of computer space and time is not essential, the user may simply fill in the vector VARBLVEC with wll nonzero elements.

VI. Space Requirements

1. The Dimension of MAINAR

The program parcels out all its work space from the array MAINAR. Since the amount of work space depends entirely on the size of the model and the number of periods in the plan, the dimension of MAINAR is provided by the user in the main program. Following is a formula for calculating the dimension, NDIM, of MAINAR, assuming OFDIAV=F and OFDIAG=F:

$$\mathrm{NDIM} = 100 + 2 \times \left\{ \begin{array}{l} 5p + 2ns + ns(p+nx) + nx + nx^2 + nid \\ + p^*(nx+1+nxp) + 2p^{*2} - p^*(p^*-1) \\ + npd(3p+nw) + 1 \end{array} \right\}$$

where p	=	$ny + nx = ns + nid + nx$
ns	=	number of structural equations in MODELS
nx	=	number of control variables
nid	=	number of identities in MODELI
npd	=	number of periods
nw	=	number of exogenous variables
nxp	=	$\max(2,nx)-1$
p^*	=	dimension of the 'compressed' model which is discussed in section III
	=	number of entries in TABL3, as printed.

The user may find it difficult to determine p^*. p^* should take the same value as p in the very first time the model is run, which will produce a print-out of TABL3; the number of entries of TABL3 (always $\leq p$) can then be used for the value of p^* in the later runs of the model.

Add to the expression within the brackets the following:

$(ns^2-ns)/2$ if OFDIAV=T, and

$(p^2-p)/2$ if OFDIAG=T .

2. Region Size

The region(or core) size needed to execute the optimal control program is determined by the following factors:

(a) size of the optimal control program $\simeq$ 146K;

(b) size of I/O buffers $\simeq$ 16K;

(c) size of MODELS and MODELI subroutines = MOD('bytes');

(d) dimension of MAINAR = NDIM('words')

The formula to obtain the value for the REG parameter is

$$\text{REG} = \left[\frac{4 \times \text{NDIM} + \text{MOD}}{1024} + 146 + 16\right] \text{K}$$

To obtain MOD in 'bytes,' the user should first compile MODELS and MODELI (to debug coding errors as well as to determine the size), and the size of each subroutine can be found at the end of its compilation listing, printed after the words 'PROGRAM SIZE.'

VII. The Use of Two or More Job Runs.

Figure 2 is a schematic diagram showing the input and output of the optimal control program and the relationship between two 'continuous' job runs. Recall from section II that if a model requires a large number of iterations in the optimal control calculation before the optimal solution path y_t is reached, the user may break up the iterating process into two or more job runs. At the end of the first job run, where NROUND=1 and assuming ITERL1=3, the program writes the solution path y_t and x_t obtained from the third iteration onto the data set INOUT2. If the printed output from this job shows that the solution path y_t obtained from the third iteration did not converge, the user may then 'continue' the iterating process by feeding into the program the same input deck with one change -- setting NROUND=2. The program will read y_t and x_t from INOUT2 and use them as the tentative path for the fourth iteration. A third or a fourth job run may be required before the solution path y_t will converge. One may of course obtain the optimal solution y_t in one job run by setting ITERL1 (and the estimated computer time) to a high value, because the program will not terminate until the solution path converges or until the iteration limit is reached (see Figure 1). However, the user is recommended to break up a possibly long iterating process into several job runs, each with a small iteration limit ITERL1. The advantage is that the user may examine the printed output at the end of each 'small' job, to determine if any parameter, such as DAMPX, should be changed to speed up the convergence or to generate more efficient runs; whereas by giving a high value to ITERL1 the user may look at a printed output only after a lot of computer time has been used, and possibly wasted.

During the first few times that a model is run, many of the input parameters such as ITERL2, EPS1, EPS2 and DAMPX are probably given values on the

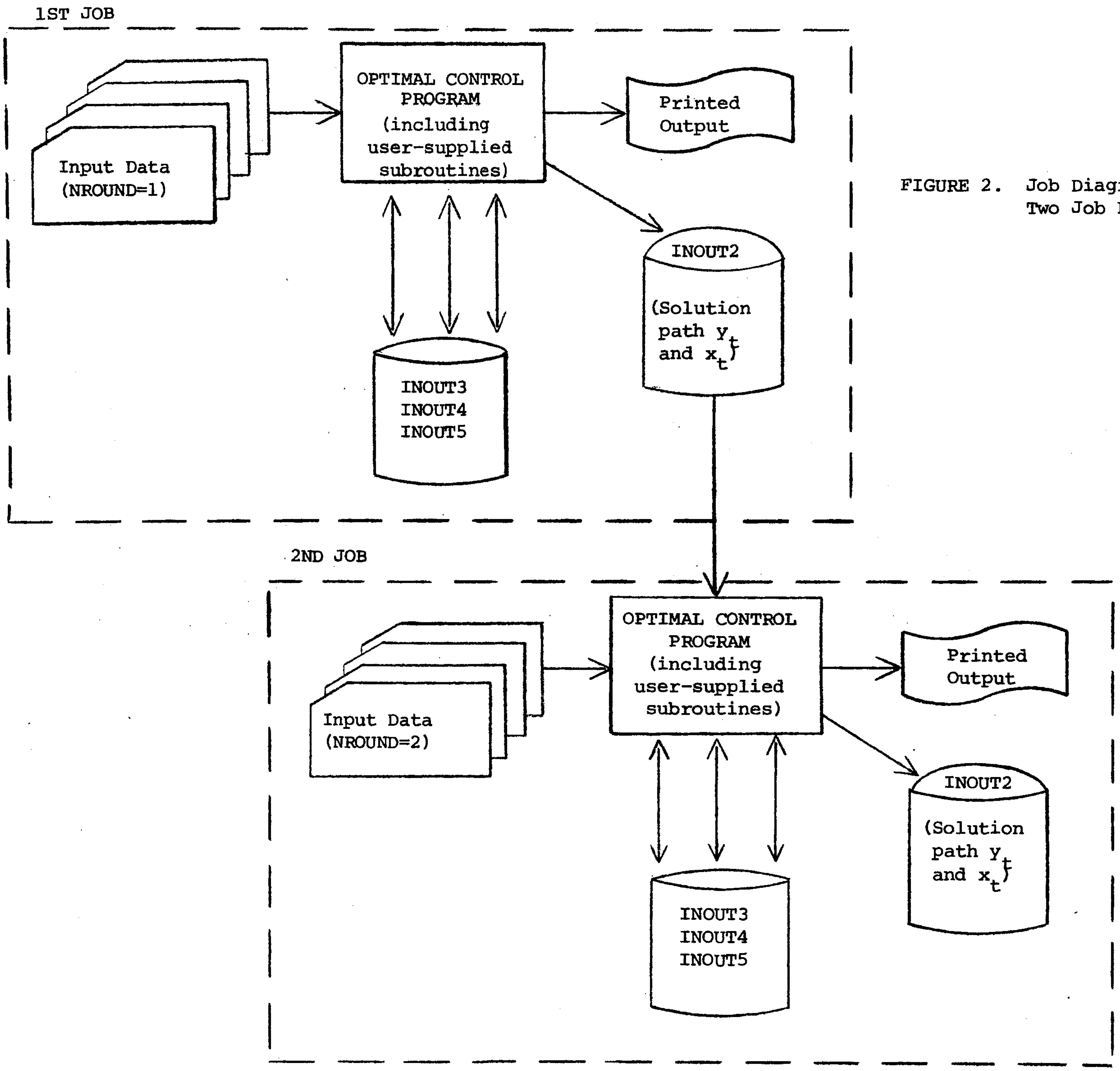

1ST JOB
Input Data (NROUND=1)
OPTIMAL CONTROL PROGRAM (including user-supplied subroutines)
Printed Output
INOUT2
(Solution path y_t and x_t)
INOUT3 INOUT4 INOUT5
2ND JOB
Input Data (NROUND=2)
OPTIMAL CONTROL PROGRAM (including user-supplied subroutines)
Printed Output
INOUT2
(Solution path y_t and x_t)
INOUT3 INOUT4 INOUT5

FIGURE 2. Job Diagram Showing Two Job Runs

basis of trial and error. It then makes sense to keep these runs 'small' in order to determine what better values to give these parameters without wasting too much computer time. Moreover, the coding of MODELS and MODELI and the construction of the vectors IDENTVEC and VARBLVEC are subject to errors that can be very easily committed. By setting GAUSS=2, the user terminates the program immediately after the Gauss-Seidel solution for y_t is obtained, printed, and saved on INOUT2, and he/she may then decide what might be 'wrong' with the model subroutines or with the input data. A good set of Gauss-Seidel solutions can be used as the tentative path for the first iteration in a later run, with NROUND=2. By running 'small' jobs one does not lose except possible a small amount of labor.

VIII. Output and Error Messages

The program prints the following on SYSOUT=A for each job:

(1) A list of the &DIMENS and &OPT NAMELIST options, including the default values if they have not been overridden.

(2) All the input matrices in the order they are read in.

(3) If GAUSS=1 or 2, the values of the Gauss-Seidel solution for y_t used for the first linearization of the model.

(4) The A, b, C, G, g, h, H matrices, starting with the last period. Their frequency of print-out from period to periods depends on the value of the OSUP parameter (see section V.2).

(5) The values of the optimal solution y_t.

(6) Total welfare cost and its deterministic and stochastic components.

(7) If the convergence criterion EPS1 is met before the number of iterations becomes greater than ITERL1, a plot of the values of the optimal solution y_t against the target values z_t. Whether all of the y_t variables or only those that have nonzero weights in the diagonal of the K matrix are plotted depends on the value of the PLOT parameter (see section V.2). The control variables x_t are always plotted against their targets.

(8) TABL1 and TABL3. The only useful information these two tables give to the user is the number of entries in TABL3, which should be used as the value for p^*.

(9) The Γ_t matrices if GAMMA > 0. If GAMMA=1, the rows of Γ_t corresponding to those elements targetted in K are printed; if GAMMA $\geq$ 2, the input vector GAMVAR controls the printing of Γ_t.

The following is a list of messages that may be printed in SYSOUT=A:

"PROGRAM TERMINATING BECAUSE GAUSS-SEIDEL FAILED TO CONVERGE FOR VARIABLE I, PERIOD T, IN K ITERATIONS" -- In calculating the Gauss-Seidel solution for y_t, if any variable $y_{i,t}$ does not converge after ITERL2 iterations, this message appears and the program terminates with a condition code of 29.

"PROGRAM TERMINATING BECAUSE OPTIMAL SOLUTION FAILED TO CONVERGE IN K ITERATIONS, ROUND NO. N" -- If the optimal solution y_t does not converge after ITERL1 iterations, this message appears and the program terminates with a condition code of 29. In addition, the optimal solution y_t and x_t are saved on the output device assigned by INOUT2 of the main program (see section V.2).

"STORAGE ALLOCATION TO THIS POINT=XXXX WORDS OUT OF YYYY (ZZK UNUSED)" -- This message appears throughout the printed output, the user may reduce the value of NDIM in the main program accordingly.

"***INSUFFICIENT MEMORY AREA TO ALLOCATE WORK ARRAYS, XXXX WORDS ALLOCATED TO STORAGE BY MAIN PROGRAM, YYYY WORDS REQUIRED SO FAR, INCLUDING THE FOLLOWING ZZZZ WORDS CURRENTLY REQUESTED AT THE POINT INDICATED BELOW" -- A trace table follows this message showing the subroutine that led to it. The program will continue and allocates more space when required but will eventually terminate after messages like the one following are printed.

"STORAGE ALLOCATION TO THIS POINT=XXXX WORDS OUT OF YYYY (ZZK REQUIRED)" -- When several messages of this type appear before the program terminates, the user should set NDIM in the main program equal to the greatest value of XXXX in these messages for the next job.

"LOOKING FOR 'AAAA' DATA, FOUND THE FOLLOWING: - -   -"

"LOOKING FOR 'AAAA' DATA, FOUND END-OF-FILE"

"END-OF-FILE WHILE READING DATA FOR 'AAAA'"

- These messages appear when the user fails to supply enough data for the matrix named AAAA or when the entire matrix has been omitted from the input deck. The program terminates with a condition code of 130, 131, or 132.

IX. JCL Requirements

The JCL given in this section applies only to the IBM 360 or 370 computers. Users of other computers must make up their own JCL to run the program and to assign device units for the four sequential data sets used by the program, INOUT2, INOUT3, INOUT4, and INOUT5; these data sets are described in section IV.

Suppose the nonlinear optimal control program (excluding the three user-supplied subroutines) has been compiled and linked-edited into a load module named OPTCN as a member of a catalogued partitioned data set named OPTLIB, then the following JCL may be used to compile and link-edit the user-supplied subroutines, link them with OPTCN and execute the program:

```
//_EXEC_FORTGCLG
//FORT.SYSIN_DD_*
      [Main Program]
      [MODELS      ]
      [MODELI      ]
//LKED.SYSLIB_DD_DSN=OPTLIB,DISP=SHR
//            DD_DSN=SYS1.FORTLIB,DISP=SHR
//LKED.SYSIN_DD_*
_INCLUDE_SYSLIB(OPTCN)
_ENTRY_MAIN
//GO.FTOn2F001_DD_DSN=INOUT2,DISP=(NEW,KEEP),
//             UNIT=xxxx,VOL=SER=nnnnnn,
//             DCB=(RECFM=VS,BLKSIZE=1608),
//             SPACE=(CYL,(2,1))
//GO.FTOn3F001_DD_UNIT=xxxx,VOL=SER=nnnnnn,
//             DCB=(RECFM=VS,BLKSIZE=1608),
//             SPACE=(CYL,(5,4))
//GO.FTOn4F001_DD_UNIT=xxxx,VOL=SER=nnnnnn,
//             DCB=(RECFM=VS,BLKSIZE=1608),
//             SPACE=(CYL,(2,1))
//GO.FTOn5F001_DD_UNIT=xxxx,VOL=SER=nnnnnn,
//             DCB=(RECFM=VS,BLKSIZE=1608),
//             SPACE=(CYL,(5,4))
//GO.SYSIN_DD_*
       [Input data]
//
```

Of the four disk data sets, only one, FTOn2F001, is a non-temporary data set;

in this data set the program stores the solution path y_t from each job run. DISP=(NEW,KEEP) is used for the first job run with NROUND=1, and it should be replaced with DISP=OLD in the jobs with NROUND$\geq$2.

The digits n2, n3, n4 and n5 in the ddnames should correspond to the same numbers in the main program provided by the user, where INOUT2=n2, INOUT3=n3, INOUT4=n4 and INOUT5=n5. (See Section IV.1). All four data sets are created by unformetted WRITE statements; therefore they must have RECFM=V or VS. Their SPACE parameters may take different values depending on the model. Tape units may replace disk units for these data sets. If GAMMA=0, the last DD statement, with ddname FTOn5F001, can be eliminated, since no intermediate output associated with computing the Γ_t matrices will be written by the program.

The user might wish to compile the user-supplied model subroutines separately and save the load modules in the same partitioned data set as the one containing the optimal control program (OPTLIB in our JCL example). Suppose MODELS and MODELI are link-edited together as a member named MODELA in OPTLIB, then the following changes should be made in the JCL given above:

(1) Delete the MODELS and MODELI source statements that had been placed after the //FORT.SYSIN_DD_* card.

(2) Replace the INCLUDE card with _INCLUDE_SYSLIB(OPTCN,MODELA).