Volume Title: Annals of Economic and Social Measurement, Volume 6, number 5

Volume Author/Editor: NBER

Volume Publisher:'P DGT

Volume URL: http://www.nber.org/books/aesm77-5

Publication Date: December 1977

Chapter Title: A GRG Ani qtkj o hqt Eeqpqo gvke CqpvqnPtqdrgo u

Chapter Author: J. B. Mantell, L.S. Lasdon

Chapter URL: http://www.nber.org/chapters/c10545

Chapter pages in book: (p. 581 - 597)

A GRG ALGORITHM FOR ECONOMETRIC CONTROL PROBLEMS*

By J. B. MANTELL AND L. S. LASDON

A software system for solving large deterministic econometric control problems is presented. The system, consisting of FORTRAN subroutines, implements a generalized reduced gradient (GRG) method.

1. INTRODUCTION

This paper describes a software system for solving large deterministic econometric control problems. The system is a collection of FORTRAN subroutines. implementing a generalized reduced gradient (GRG) method. Its distinguishing features are:

(1) general, yet easy to use input formats and a range of output options,

(2) the ability to solve problems with hundreds of equations

(3) dynamic storage allocation, so problems of any size may be attempted by changing only one dimension statement

(4) a minimum of machine dependent statements, and

(5) well documented**

The GRG algorithm uses a pseudo-Newton method, implemented using sparse matrix techniques, to solve the model equations, and a choice of conjugate gradient or variable metric methods to generate search directions for the controls. The step along the search direction is chosen by a modification of an algorithm due to Shanno and Phua [24]. It allows the user to choose either a modified "step-length" procedure, which tends to use fewer function evaluations per search, or a more accurate cubic interpolation procedure.

The system has been tested thus far on three models: Ando-mini (5 equations) Klein-Norman (27 equations) and Klein (1950) (7 equations). Results, presented in section 8, are most encouraging. Further tests on larger models are planned.

2. PROBLEM FORMULATION

Let $y(t)$ be an n-dimensional state vector at time t, and $u(t)$ an m-dimensional control vector. The class of problems considered has the form

**User and system documentation is currently in preparation.

$$(1) \qquad \text{minimize} \sum_{t=1}^{T} f_t(y(t), \ldots, y(t - s'), u(t), \ldots, u(t - c'))$$

subject to

$$(2) \qquad g(y(t), \ldots, y(t - s), u(t), \ldots, u(t - c)) = 0, \quad t = 1, T$$

$$(3) \qquad lb(t) \leq u(t) \leq ub(t), \quad t = 1, T$$

For simplicity, all exogenous variables have been suppressed, although the software has capabilities to handle them. The objective (1) permits the term f_t to be different in different time periods and allows state and control lags of length s' and c' respectively. These lags may differ from the lags s and c in the model equations (2). The n dimensional vector of functions g and the functions f_t may all be nonlinear, and are assumed to be at least once continuously differentiable. The recursive equations (2) are assumed to have a unique solution $y(1), \ldots, y(T)$ for any set of vectors $u(1), \ldots, U(T)$ satisfying (3) and for any initial conditions and exogenous values under consideration. Only control bounds, (3), are treated exactly. Bounds on the state variables y, or other inequality constraints, may be dealt with by penalty or Augmented Lagrangian methods [2]. This simplifies the algorithm considerably, since no basis changes are required. However, penalty or Lagrangian methods may not be as efficient as approaches which deal with state variable bounds directly. For a description of such an algorithm, see the paper by A. Drud in this issue.

All values of $y(t)$ and $u(t)$ for $t < 1$ which are needed to define the problem over the interval $t = 1, T$ are assumed known. For a description of data input and required user-supplied subroutines, see the appendix.

3. A Generalized Reduced Gradient Algorithm

Let

$$(4) \qquad y = (y(1), y(2), \ldots, y(T))$$

and

$$(5) \qquad u = (u(1), u(2), \ldots, u(T))$$

Then the model equations (2) may be written

$$(6) \qquad G(y, u) = 0$$

where G is an nT-dimensional vector consisting of the T vectors g with arguments as indicated in (2). Then the problem (1)-(3) may be written

$$(7) \qquad \text{minimize} \quad f(y, u)$$

(8) $$\text{subject to} \quad G(y, u) = 0$$

and

(9) $$lb \leq u \leq ub$$

We have assumed that, given u, the system (8) may be solved for a unique y, $y(u)$. This function may be used to eliminate y in the objective yielding a new function.

(10) $$F(u) = f(y(u), u)$$

By the implicit function theorem, $y(u)$ is continuously differentiable, so F is a differentiable function of u. It is called the reduced objective function, and its gradient, $\partial F/\partial u$, is called the reduced gradient. This gradient may be computed as follows [13]:

(1) Solve

(11) $$\Pi(\partial G/\partial y) = \partial f/\partial y$$

for the Lagrange multiplier (row) vector Π

(2) Evaluate $\partial F/\partial u$ by

(12) $$\partial F/\partial u = \partial f/\partial u - \Pi(\partial G/\partial u)$$

In the above, all partial derivatives are evaluated at some known point $(\overline{y}, \overline{u})$, and (12) yields $\partial F/\partial u$ evaluated at $\overline{u}$.

Many authors (e.g. see [1], [3], [14] and [17]) have shown that, because of the dynamic structure of G and the nature of f, (11)–(12) have a recursive structure. Let Π be partitioned as $(\Pi(1), \ldots, \Pi(T))$, and define

(13) $$\partial g(t)/\partial y(\tau) = \partial g(y(t), \ldots, y(t - s); u(t), \ldots, u(t - c))/\partial y(\tau)$$

and similarly for $\partial g(t)/\partial u(\tau)$. Then, since $\partial G/\partial y$ is lower block triangular, (11) may be written

(14) $$\Pi(t)B(t) = \sum_{i=0}^{\ell} \partial f_{t+i}/\partial y(t)$$
$$- \sum_{j=1}^{p} \Pi(t + j)\partial g(t + j)/\partial y(t) \quad t = T, T - 1, \ldots, 1$$

where $B(t) \equiv \partial g(t)/\partial y(t)$, $\ell = \min(T - t, s')$, and $p = \min(T - t, s)$. Assuming that the matrices $B(t)$ are nonsingular*, these equations may be solved sequentially for $\Pi(T), \Pi(T - 1), \ldots, \Pi(1)$. Then the reduced gradient subvectors $\partial F/\partial u(t)$ are evaluated using

---

*This, plus the fact that the problem functions are continuously differentiable are sufficient conditions for existence of the function $y(u)$ for all points in some neighborhood of the current one.

(15)
$$\partial F/\partial u(t) = \sum_{i=0}^{r'} \partial f_{t+i}/\partial u(t) - \sum_{j=0}^{p'} H(t+j)\,\partial g(t+j)/\partial u(t) \quad t = 1, T$$

where $r' = \min(T-t, c')$ and $p' = \min(T-t, c)$.

A GRG algorithm for the problem (1)–(3) may now be stated in general terms

Step 0 Given:

Initial control vector $= u^{(0)}$

all exogenous variables plus all initial values of lagged states and controls

set $i = 0$

Step 1 Simulate the system with $u = u^{(0)}$ to determine all state variables and the objective value $F(u^{(i)})$

Step 2 Compute $\nabla F(u^{(i)})$ from (14) and (15).

Step 3 Check for convergence. If the convergence criteria are satisfied, stop. Otherwise go to Step 4.

Step 4 Compute the search direction $d^{(i)}$ using an unconstrained minimization algorithm. This step must be modified to account for bounds on the variables.

Step 5 Perform a one dimensional search along $d^{(i)}$ to find $\alpha = \alpha^{(i)}$ such that $F(u^{(i)} + \alpha d^{(i)})$ is minimized subject to $\alpha > 0$, and $lb \le u^{(i)} + \alpha d^{(i)} \le ub$. At each value for α in the search it is necessary to simulate the system, compute the objective, and (perhaps) compute the reduced gradient.

Step 6 Set $u^{(i+1)} = u^{(i)} + \alpha^{(i)} d^{(i)}$

Step 7 Replace i by $i + 1$ and return to Step 3 (to Step 2 if the reduced gradient is not computed in the one dimensional search).

To transform this general algorithm into a computer code, the methods used in the various steps must be specified, and they must be implemented in a numerically reliable way which exploits sparsity. The following sections describe our choices and implementations.

4. SIMULATING THE SYSTEM (STEP 1)

We have chosen to simulate the system using a pseudo-Newton algorithm. Assume that, for given initial conditions, exogenous, and control variables, the model equations (2) have been solved for times $1, 2, \ldots, t-1$, and the solution for $y(t)$ is desired. The pseudo-Newton method is

(16) $$y^0(t) \text{ given}$$

(17) $$B(t)\delta^k(t) = g(y^{k-1}(t), y(t-1), \ldots, y(t-s); u(t), \ldots, u(t-c))$$

584

(18) $$y^k(t) = y^{k-1}(t) + \delta^k(t), \quad k = 1, 2, \ldots$$

In the above. $y^k(t)$ is the kth estimate of $y(t)$. The method is called pseudo-Newton because the matrix $B(t)$ is evaluated only once at the starting point. In our code. this point is the y and u vector from the last simulation. The classical Newton's method reevaluates $B(t)$ anew at each point $y^k(t)$.

Ortega and Rheinboldt [18] show that this method will converge if the solution of (2) is sufficiently close to the starting point. and that the rate of convergence is linear.

The key to implementing this Newton's method successfully for large models is exploiting the sparsity of $B(t)$. For large econometric models (more than, say, 100 equations), the nonzero density of $B(t)$ (i.e.. the percentage of its elements which are nonzero) will be on the order of a few percent. Hence only the nonzero elements are stored, and the linear equations (17) are solved by finding lower and upper triangular matrices $L(t)$ and $U(t)$ such that

(19) $$B(t) = L(t)U(t)$$

Then (17) is easily solved by forward elimination and back substitution in two triangular systems.* Much work exists on how to perform the factorization (19) so that, if $B(t)$ is sparse, $L(t)$ and $U(t)$ will be sparse also [9] [22]. Our software uses slightly modified versions of subroutines by Curtis and Reid [2] and Duff and Reid [4] to perform the factorization in a way which preserves both numerical stability and sparsity. Figure 1 shows the result of applying these subroutines to the $B(t)$ arising from Klein's model [10]. In the figure, X denotes a nonzero element, and F denotes a new nonzero created in the course of the factorization. The densities of L and U are only slightly higher than that of the original matrix, while B^{-1} is 100% dense.

Actually, the structure of $B(t)$ may be exploited even more fully. For most models, the rows and columns of $B(t)$ may be rearranged to create a block triangular matrix with nonsingular blocks. Figure 2 shows this for Pindyck's model [20]. The blocks may then be solved sequentially. In solving each block, the pseudo-Newton method (16)–(18) is used, and each block is factorized rather than the entire matrix $B(t)$. This yields significant savings in efficiency, storage, and accuracy.

When implemented as described above, simulation using Newton's method is often faster than current Gauss-Seidel techniques. Newton's method converges in one step for linear equations, its storage demands

---

*LU decomposition is only one of several techniques which have been developed for solving sparse linear systems. It is widely used in linear programming. For other approaches, see [25].

Klein's Model of U.S. Economy [1950]

$$\frac{\partial g(t)}{\partial y(t)} \equiv B =$$

(34.7% dense)

$$LU =$$

(42.9% dense)

$$B^{-1} =$$

(100% dense)

Figure 1   *LU* decomposition.

are modest, and it is likely to be more accurate as well. The software implementation is much more complex than for Gauss-Seidel, but many of the subroutines are needed anyway for the reduced gradient computations. Little more is required of the user, since the partial derivatives in $B(t)$ can be computed by finite differencing of the model equations. Of course, only nonconstant elements of $B(t)$ are recomputed,* and any blocks of $B(t)$ which are constant are factorized only once, at the start of the algorithm. In addition, the pattern of nonzero elements in $B(t)$ is independent of t. This means that the subroutine which determines the se-

---

*To permit this, the user must indicate which variables appear nonlinearly in each equation - see the appendix for details.

586

Pindyck's Quarterly Model of the U.S. Economy (1970)

	1	2	3	4	5	6	7	8	9	10	11	12
1	X				X	X	X					
2	X	X		X								
3		X	X									
4		X	X							X		
5		X		X								
6		X			X							
7		X	X				X					
8		X						X		X		
9		X						X	X			
10		X							X			
11		X									X	
12		X										X

After Partitioning:

	1	2	3	4	5	6	7	10	12	11	8	9	
1	X				X	X	X						
2	X	X		X									
3		X	X										
4		X	X				X						Block #1
5		X		X									
6		X			X								
7		X	X				X						
10		X					X						
12		X						X					Block #2
11		X							X				Block #3
8		X					X			X			Block #4
9		X								X	X		Block #5

Figure 2   Block Triangularization

587

quence of pivot elements in $B(t)$ need be invoked only once at the start of the algorithm, and recalled only if some pivot element is too small.

To deal with the possibility of the pseudo-Newton method not converging (within a prespecified iteration limit), we have added the option of reevaluating and refactorizing $B(t)$ every r iterations, where r may be user controlled. The classical Newton's method corresponds to $r = 1$. If this option is unsuccessful, the step α in the one dimensional search (see Section 7) is reduced and the pseudo-Newton method is tried again. If α is small enough, Newton's method must converge.

5. COMPUTING THE REDUCED GRADIENT (STEP 2)

Computing $\partial F/\partial u$ by (14)-(15) again requires an LU factorization of each $B(t)$. The factorizations used in simulating the system cannot be used for two reasons. First, the matrices in (14) must be evaluated at the point obtained by simulating the system in step 1, while the $B(t)$ used in the simulation were evaluated at the previously simulated point, since the new state values are unknown prior to simulation. Second, to reduce storage requirements, the matrices $B(t)$ and their factorizations are overwritten when t is changed, so they are not available for use in (14).

For $t = T, T - 1, \ldots, 1$, the matrix $B(t)$ is reevaluated and refactorized, (14) is solved for $\Pi(t)$, and $\partial F/\partial u(t)$ is evaluated using (15). These computations are considerably simplified by the fact that the Jacobians with respect to the lagged states and controls ($\partial g(t + j)/\partial y(t)$ in (14) and $\partial g(t + j)/\partial u(t)$ in (15)) grow increasingly sparse as j increases. Hence the summations in (14)-(15) may be evaluated very rapidly.

6. CONVERGENCE CRITERIA (STEP 3)

Let u_j be the jth component of u. Then u satisfies the Kuhn-Tucker necessary conditions for optimality in the problem (1)-(3) if

$$(20) \qquad u_j = lb_j \implies \partial F/\partial u_j \geq 0$$

$$(21) \qquad u_j = ub_j \implies \partial F/\partial u_j \leq 0$$

$$(22) \qquad lb_j < u_j < ub_j \implies \partial F/\partial u_j = 0$$

See [12] for proof. In this software system the algorithm stops if these conditions hold to within a tolerance ϵ, with default value 10^{-3}. It also stops if the fractional change in the objective value is less than ϵ^1 for NSTOP consecutive iterations, where ϵ^1 and NSTOP have default values of 10^{-3} and 3 respectively.

7. COMPUTING THE SEARCH DIRECTION (STEP 4)

In this software system, the user has a choice of a variable metric method, or one of 6 conjugate gradient methods for choosing the search

direction, d. The variable metric method chooses d by solving the equations

$$R^T R d = -\partial f / \partial u$$

where $R^T R$ is an approximation to the Hessian matrix $\partial^2 f / \partial u^2$. The square upper triangular matrix R and the vectors d and $\partial f / \partial u$ have dimension equal to the number of control variables which are free (not at a bound) at the current point. Following Murtagh and Saunders [15], these free controls are called "superbasic." $R^T R$ is updated using the complementary DFP formula

$$R^T R = R^T R + \frac{1}{\alpha h^T d} h h^T + \frac{1}{g^T d} g g^T$$

In the above, R and $\bar{R}$ are the previous and updated matrices respectively, h is the change in $\partial f / \partial u$ in the last one dimensional search, α is the step size determined by the search, and g is $\partial f / \partial u$ at the start of the search.

To insure stability, all modifications to R are implemented using elementary orthogonal matrices (plane rotations). The two subroutines used, obtained through the courtesy of Michael Saunders, are

$$\text{R1ADD} \quad \bar{R}^T \bar{R} = R^T R + z z^T$$

$$\text{R1SUB} \quad \bar{R}^T \bar{R} = R^T R - z z^T$$

These use forward and backward sweeps of plane rotations respectively, as described in [7].

Maintaining the approximate Hessian in factorized form has several advantages. During updating, tests can be made to insure that a new non-singular matrix $R^T R$ with real elements exists. If not, the updating is skipped. Hence the approximate Hessian is always positive definite, even in the presence of numerical error. Also, the square of the ratio of largest to smallest diagonal elements of R is a lower bound on the condition number of $R^T R$. When this bound becomes too large, $R^T R$ can be reset to the identity matrix. Resetting is not necessary except under these circumstances.

The alternative of using a conjugate gradient (CG) method was added to deal with problems with many superbasic variables, where the dimension of the (dense) matrix R may require excessive storage. CG methods require storage of at most a few vectors of dimension equal to the number of superbasics. Six variants have been included in a subroutine CG (implemented by Michael Saunders). These include formulas due to Fletcher and Reeves [6], Polak and Ribiere [21], Perry [19], and two derived as 1-step version of the DFP and complementary DFP variable metric formulas. If the number of superbasic variables exceeds a user defined limit

(default value 50), the system will switch from the variable metric method to whichever CG method the user selects, switching back again if possible.

The direction finding subroutine must deal with the possibility that a superbasic variable may have hit a bound during the previous one dimensional search. If so, and a CG method is being used, the CG method is restarted (d is set to $-\partial F/\partial u$). If the variable metric method is being used, R can be modified by deleting the appropriate column, and transforming the resulting matrix back to upper triangular form (see [15] for details). If a new superbasic variable is added by releasing a nonbasic variable from its bound, a unit vector column is added to R. The logic for releasing a nonbasic variable from its bound is as prescribed by Goldfarb [8].

Note that both the CG and the variable metric methods minimize quadratic functions in as many one dimensional searches as there are variables, if the one dimensional search is done exactly.

8. THE ONE DIMENSIONAL SEARCH (STEP 5)

The one dimensional search subroutine is an adaptation of one due to Shanno and Phua [24]. It uses quadratic and cubic interpolation and extrapolation to estimate the value of α^i. Both F and $\partial F/\partial u$ are evaluated at each point in the search, and the polynomials are fitted to the F values and the values of $dF/d\alpha = (\partial F/\partial u)^T d$. The decision to use $dF/d\alpha$ was made because, once the system is simulated to evaluate F, $\partial F/\partial u$ can be evaluated with less additional effort than a complete simulation, i.e., the gradient of F is easier to evaluate than F itself.

The search operates in either of 2 modes. In mode 2 the search is terminated whenever the decrease in F is deemed sufficiently large, i.e., whenever the conditions

$$F_{new} < F_{old} + 10^{-4}(dF/d\alpha)_{\alpha=0}$$

and

$$(dF/d\alpha)_{\alpha=0} < (dF/d\alpha)_{new}$$

are satisfied [24]. In mode 1, the minimum must be bracketed and at least one interpolation must be performed before the search will stop. This usually yields a more accurate estimate of α^i at the expense of more effort in the search. Some experience with these options is described in section 9.

For problems with linear model equations and a quadratic objective, the function $F(u + \alpha d)$ is quadratic in α for any u and d. Hence, by using quadratic and cubic approximating functions, this one dimensional search will converge in at most 2 evaluations of F and $dF/d\alpha$. This fact, the 1 step convergence of Newton's method, and the finiteness of the conjugate direction methods of section 7 on quadratic functions, together imply that this GRG algorithm will converge finitely for linear-quadratic problems.

9. COMPUTATIONAL RESULTS

This software system has been tested on three different econometric models. These are Klein's Model (1950) [11], the Ando-Mini Model and the Klein-Norman Model [10], [16]. The problems solved were chosen to test the computational efficiency of the software system rather than to evaluate economic policies.

Results are summarized in table 1. In the methods column of the table, VM denotes the Complementary DFP variable metric method, FR the Fletcher-Reeves Conjugate Gradient Method, and SH a scaled conjugate gradient algorithm proposed by Shanno [23]. G, DFP, and SA are Gradient, Davidon-Fletcher-Powell, and Successive Approximation algorithms tested by A. L. Norman [17]. A simulation is T solutions of the model equations for $t = 1, 2, \ldots, T$ and time is in computer resource units.*

All problems were solved using the FORTRAN V compiler on the UNIVAC 1108 computer at Case Western Reserve University, with double precision floating-point arithmetic (machine accuracy of approximately 10^{-15}).

In all runs, convergence was assumed if the Kuhn-Tucker conditions, (20)-(22), were satisfied to within 10^{-3} or if the relative change in the objective function was less than 10^{-3} for three consecutive iterations. The convergence criterion used for the Newton method was that the left hand side of each equation was less than 10^{-4} in absolute value. All runs used finite difference derivatives for the model equations and analytic objective derivatives.

The first two test problems use Klein's Model with government spending and business taxes chosen as controls. The objective function is the sum of the squared deviations of the state variables from desired values. These values were obtained by simulating the system at the historical values of the controls. Thus, the optimal values of the objective and the controls were known to be zero and the historical values respectively. One and three time period versions of this problem were solved using starting values of zero for all controls.

Since these problems are linear-quadratic, convergence using variable metric or conjugate gradient methods should theoretically occur in M iterations or less (where M is the total number of controls) if exact one dimensional searches are used. This behavior was observed for the one time period problem using the complementary DFP variable metric formula and using the Fletcher-Reeves conjugate gradient formula. Both algorithms converged in two iterations generating identical solutions at

---

*This is execution time plus a small charge for input/output and file handling. Cost is computed on this basis.

591

TABLE 1

OPTIMAL CONTROL TEST PROBLEMS

Model	States Per Period	Controls Per Period	Periods	Method	Search Mode	Objective Function	Line Searches	Simulations	Model Equation Evaluations	Time
Klein (1950)	7	2	1	VM	1	1.08×10^{-7}	2	5	102	1.23
			1	F-R	1	1.08×10^{-7}	2	5	102	1.24
			3	VM	1	1.14×10^{-7}	6	13	578	3.27
			3	F-R	1	1.14×10^{-7}	7	15	662	3.73
			3	SH	1	1.14×10^{-7}	7	15	662	3.48
Ando-Mini	5	1	10	VM	1	35.82	12	34	8246	8.39
				VM	2	35.86	35	36	8961	10.03
				F-R	1	35.83	16	33	8458	8.57
				SH	1	35.82	14	37	9232	9.37
				SH	2	36.66	15	16	4084	5.14
				SA*		35.82	3 4	17 22		
Klein-Norman	27	1	13	VM	1	−160.49	9	19	36823	29.34
				VM	2	−160.56	11	12	24418	20.56
				SH	2	−160.47	10	11	22601	18.69
				F-R	1	−160.49	9	19	36823	27.69
				G*		−159.79	9	50		
				CG*		−160.50	9	50		
				DFP*		−160.50	9	49		
				SA*		−160.59	2	17		
Klein-Norman**	27	1	13	VM	1	−210.54	17	35	85425	58.95
				SA*		−210.55	3	20		

*Results obtained by A. L. Norman.
**A different starting point was used.

each iteration. Similar results were obtained for the three period problem. The quasi-Newton method converged in the theoretical limit of six iterations. Two conjugate gradient algorithms. Fletcher-Reeves and Shanno. performed almost as well. They reduced the objective from 505.0 to approximately 10^{-5} in six iterations. However. one more iteration was required by both to satisfy the Kuhn-Tucker conditions to within 10^{-3}.

The second model was the Ando-Mini model. It has five equations and is nonlinear. The policy problem considered used unborrowed reserves as a control variable and has been considered by other researchers. Computational results for this problem obtained by A. L. Norman using the successive approximation algorithm [16] are listed in Table 1. For this problem. the successive approximation method gave the best results in terms of the number of simulations required.* Of course. the simulations require different amounts of effort. since different methods were used. The results of Table 2 suggest that the simulations done in [21]). which used a Gauss-Seidel algorithm, required more time than ours. which used the pseudo-Newton algorithm.

All Ando-Mini test runs achieved optimality except the Shanno conjugate gradient with search mode 2 which terminated prematurely on the fractional change criterion. This method converged when the search mode was set to 1.

The last model considered is the Klein-Norman model. This is a nonlinear 27-equation model of the U.S. economy. A policy problem previously considered by Normal et. al.** was solved. Results obtained using the variable metric algorithm compare very favorably with the previous results.

Great savings are obtained in this problem by using the pseudo-Newton method for simulation rather than Gauss-Seidel. A comparison between the two methods for the test problems considered is given in Table 2. The Gauss-Seidel method was found to be highly sensitive to the choice of the damping factor and thus requires a skilled analyst for efficient use. The Newton method. on the other hand. is very easy to use and is significantly more efficient for these problems.

To compare the efficiencies of Gauss-Seidel and Newton methods for optimal control problems. the Klein-Norman model was used as a test. along with a Gauss-Seidel routine provided by A. L. Norman. With the same initial values for the controls. and the same initial conditions as were used in the runs of Table 1. the Gauss-Seidel algorithm required an average of 110 iterations per time period to solve the model at the initial point. In the 13 period. 27 equation problem. this leads to over 33,000

*Comparison between the number of iterations is misleading since the computational effort per iteration varies greatly.
**Policy problem 1 in [17].

TABLE 2

COMPARISON OF GAUSS-SEIDEL AND PSEUDO-NEWTON METHODS

Problem	Average Number of Gauss-Seidel Iterations	Average Number of pseudo-Newton* Iterations
Klein (3-period)	22.0 $(w = 1)$**	1.00
Ando-Mini	7.66 $(w = .2)$	
	29.63 $(w = .05)$	2.59
Klein-Norman	69.23	2.88

*The Newton method used reevaluated and refactorized the matrix $B(t)$ every 5 and 3 iterations for Ando-Mini and Klein-Norman respectively.
**w is the relaxation parameter or damping factor.

model equation evaluations to perform a single simulation over all 13 periods. In contrast, a complete optimization of this model using our code, with the same initial conditions required only 24.418 function evaluations (see VM with mode = 2 in Table 1). During this optimization, 11 simulations were performed (each over 13 time periods), using the pseudo-Newton algorithm.

The ratio of simulations to line searches is a measure of the efficiency of the one dimensional search routine. In mode 2 (less exact mode), this ratio is exactly 1 (an extra simulation is required at the start). In mode 1 the ratio is 2 except for 1 problem (Ando-Mini with method SH). Since the total number of searches and simulations is at most a small multiple of the number of variables, both modes are operating well. Neither has a clear superiority in these tests.

In closing this section, we note that computation times on larger models should be considerably reduced if analytic partial derivatives are used. This is because the derivatives are usually much simpler than the equations themselves. Since all runs here used finite difference derivatives, some of these times could be reduced, perhaps significantly. Whether the tradeoff of coding for execution time is worthwhile or not depends mainly on how often the model is run and on how often its structure is changed.

10. CONCLUSION

The software system described here promises to be an efficient, robust tool for solving large optimal control problems. The sparse matrix implementation of the pseudo-Newton method, the variable metric and conjugate gradient methods, and the inexact one dimensional search all have performed well in initial tests. Further testing, on larger models, is planned. Our objectives in future work are to develop a system which is

portable, well documented, easy to use, efficient and robust, and to disseminate such a system as widely as possible.

II. ACKNOWLEDGEMENTS

REFERENCES

[1] J. Abadie, "Application of the GRG-Algorithm to Optimal Control Problems," in "Integer and Nonlinear Programming," J. Abadie (ed.), North Holland, 1970.

[2] Curtis, A. R. and Reid, J. K., "The Solution of Large Sparse Unsymmetric Systems of Linear Equations," *J. Inst. Maths. Applic.*, Vol. 8, 1971, pp. 344 353.

[3] Drud, A., "Optimization of Large Scale Dynamic Systems Using the GRG-Algorithm," TIMS XXIII International Meeting in Athens, July 1977.

[4] Duff, I. S., and Reid, J. K., "An Implementation of Tarjan's Algorithm for the Block Triangularization of a Matrix," *AERE Report* C.S.S. 29 (1976), Harwell England.

[5] Fletcher, R., "An Ideal Penalty Function for Constrained Optimization," *J. Inst. Maths Applics*, 15, 1975, pp. 319 342.

[6] Fletcher, R., and Reeves, C. M., "Function Minimization by Conjugate Gradients," *British Computer J.* 7 1964, 149 154.

[7] Gill, P. E., Golub, G. H., Murray, W., and Saunders, M. A., "Methods for Modifying Matrix Factorizations," *Math. Comp.* 28 (1974) 505 535.

[8] Goldfarb, D., "Extension of Davidons Variable Metric Method to Maximization Under Linear Inequality and Equality Constraints," *SIAM J. Appl. Math.* 17, No. 4, July 1969.

[9] Hellerman, E., and Rarick, D. C., "Reinversion with the Preassigned Pivot Procedure," *Math. Prog.* 1 (1971) 195 216.

[10] Klein, L. R., "Estimation of Interdependent Systems in Macroeconomics," *Econometrica*, Vol. 37, No. 2, April 1969.

[11] Klein, L. R., "Economic Fluctuations in the United States 1921 1941," Cowles Commission Monograph No. 11, John Wiley and Sons, Inc., New York, 1950.

[12] Lasdon, L. S., Fox, R., Ratner, M., "Nonlinear Optimization Using the Generalized Reduced Gradient Method," *Revue Francaise d'Automatique d'Informatique et de Recherche Operationnelle*, Vol. 3, 1974, pp. 73 104.

[13] Lasdon, L. S., Waren, A. D., Jain, A., and Ratner, M., "Design and Testing of a GRG Code for Nonlinear Optimization," to appear in the ACM Transactions on Mathematical Software.

[14] Mantell, J. B., "Optimal Control of Large Nonlinear Planning Models," Ph.D. Dissertation, Department of Operations Research, Case Western Reserve University, June 1977.

[15] Murtagh. B.. and Saunders. M.. "Nonlinear Programming for Large Sparse Systems." Tech. Rept. SOL 76-15. Department of Operations Research. Stanford University, Aug. 1976.

[16] Norman. A. L. and Norman. M. R.. "Behavioral Consistency Tests for Econometric Models." *IEEE Transactions on Automatic Control*. AC-18. No. 5. 1973.

[17] Norman. A. L.. Norman. M. R.. and Palash. C. J.. "On the Computation of Deterministic Optimal Macroeconomic Policy." Internal report no. 74-1 Department of Economics. University of Texas. Austin. Texas 78712.

[18] Ortega. J. M. and Rheinboldt. W. C.. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press. New York. New York. 1970.

[19] Perry. A.. "An Improved Conjugate Gradient Algorithm." Technical Note (March 1976). Department of Decision Sciences. Graduate School of Management. Northwestern University. Evanston. Illinois.

[20] Pindyck. R. S.. "An Application of the Linear Quadratic Tracking Problem to Economic Stabilization Policy." *IEEE Transactions on Automatic Control*. AC-17. No. 3. June. 1972.

[21] Polak. E.. *Computational Methods in Optimization: A Unified Approach* (Academic Press. 1971).

[22] Saunders. M. A.. "A Fast. Stable Implementation of the Simplex Method Using Bartels-Golub Updating." in: J. R. Bunch and D. J. Rose. Eds.. *Sparse Matrix Computations* (Academic Press. New York and London. 1976) pp. 213 226.

[23] Shanno. D. F.. "Conjugate Gradient Methods with Inexact Searches." Working Paper. Management Information Systems. College of Business and Public Administration. University of Arizona. Tucson. Arizona 85721.

[24] Shanno. D. F. and Phua. K. H.. "Minimization of Unconstrained Multivariate Functions." *A.C.M. Transactions on Mathematical Software*. Vol. 2. No. 1. March 1976. pp. 88 94.

[25] Duff. I. S.. "A Survey of Sparse Matrix Research." *Proceedings of the IEEE*. Vol. 65. No. 4. April 1977. pp. 500 535.

APPENDIX

Input Data and Subroutine Descriptions

The user inputs the initial conditions, values of all exogenous variables, the bounds lb and ub and other problem data using a data deck which is divided into sections. Each section begins with a keyword (e.g. LIMITS and METHODS) and ends with the word END. This makes the deck easier to read and check. All internal limits and tolerances have default values, and need not be input. All input data is checked for obvious errors and is echoed back to the user.

The model g and objective terms f_i are specified by user-provided subroutines. The model subroutine must allow computation of any specified model equation for given values of the y's and u's (which requires somewhat more complex coding than evaluating all equations). The solution algorithm requires first derivatives of all problem functions. The user has the option of using a system finite difference subroutine to compute them or coding his own subroutine to evaluate them analytically.

It is important to avoid computing Jacobian elements which are identically zero, and to recompute only nonconstant Jacobian elements. To permit this, one of the larger sections of the data deck contains in-

formation specifying which variables appear in each equation, and which appear nonlinearly. This information, coupled with the ability to evaluate any specified model equation, allows the finite difference derivative subroutine to evaluate only those Jacobian elements needed at a particular stage of the algorithm.