Annals of Economic and Social Measurement, 3/1, 1974

METHODS FOR COMPUTING OPTIMAL CONTROL SOLUTIONS

ON THE SOLUTION OF OPTIMAL CONTROL PROBLEMS AS MAXIMIZATION PROBLEMS

BY RAY C. FAIR*

In this paper the problem of obtaining optimal controls for econometric models is treated as a simple unconstrained nonlinear maximization problem. Various maximization algorithms are tested, and the results indicate that quite large problems can be solved. For deterministic problems it appears feasible to compute optimal controls for most econometric models encountered in practice. Stochastic problems can also be solved by the approach of this paper by means of stochastic simulation.

1. INTRODUCTION

There appears to be among many economists the view that the computation of optimal controls for moderate- to large-scale nonlinear econometric models is not feasible. Pindyck [19], for example, has questioned whether "nonlinear optimization [is] worth all of the computational difficulty that it entails,"[1] and Shupp [24] has stated that "the size and complexity of these models preclude formal optimization."[2] The results presented in this paper indicate that this view is not correct, even for models of up to 100 or 200 equations. The results suggest that it is feasible to compute optimal controls for most econometric models encountered in practice.[3]

Historically, optimal control problems have been formulated in continuous time and have been looked upon as problems in choosing *functions* of time to maximize an objective function. Fairly advanced mathematical techniques are required to solve these problems. For discrete-time models, however, which include virtually all large-scale econometric models, optimal control problems can also be looked upon as problems in choosing *variables* to maximize an objective function. The number of variables to be determined is equal to the number of control variables times the number of time periods chosen for the problem. From this perspective, optimal control problems are straightforward maximization problems, and in attempting to solve problems in this way, one can take advantage of the recent advances that have been made in computational algorithms for maximizing nonlinear functions of variables. This approach, of treating optimal control problems as problems of maximizing a nonlinear function of variables, is the approach taken in this paper.

* Department of Economics, Princeton University. I would like to thank Gregory C. Chow, Kenneth D. Garbade, Stephen M. Goldfeld, and Richard E. Quandt for many helpful comments.

[1] Pindyck [19], p. 388.

[2] Shupp [24], p. 94.

[3] See also Holbrook [13] for a method of controlling a nonlinear system with a quadratic objective function.

2. The General Method of Solution

Assume that the model under consideration is deterministic[4] and has g equations. Write each equation for each period of time as

$$f_{it}(y_t, z_t, x_t, \alpha_{it}) = 0: \quad \begin{array}{l} i = 1, \ldots, g; \\ t = 1, \ldots, T; \end{array} \tag{1}$$

where y_t is a vector of observations for period t on the g endogenous variables in the model, z_t is a vector of observations for period t on the noncontrol, predetermined variables in the model, x_t is a vector of observations for period t on the control variables in the model, and α_{it} is a vector of nonzero parameters that are included in equation i for period t. The t subscripts in α_{it} and f_{it} allow for the possibility that some parameters and some functional forms are changing over time.[5] Lagged endogenous variables are included in the z_t vector. T is the total number of periods to be considered in the control problem.

The model in (1) is assumed to be such that, for each t, given values for z_t, x_t, and α_{it} $(i = 1, \ldots, g)$, one can solve numerically for y_t. In practice, most large-scale econometric models are solved each period by some version of the Seidel method.[6] Further, one can frequently isolate each component of the y_t vector on one side of one equation, which greatly aids in the solution of the model. If the model is solved for more than one period, then the solution values of the endogenous variables for previous periods are used, when appropriate, as values for the lagged endogenous variables in the z_t vector. For linear models, of course, values of y_t are merely obtained from reduced form equations.

For a time horizon of T periods, the objective function, h, is taken to be a function of y_t, z_t, and x_t $(t = 1, \ldots, T)$:

$$W = h(y_1, \ldots, y_T; z_1, \ldots, z_T; x_1, \ldots, x_T), \tag{2}$$

where W, a scalar, is the value of the objective function corresponding to values of y_t, z_t, and x_t $(t = 1, \ldots, T)$.

The optimal control problem for this discrete-time, deterministic model is to choose values of $x_1, \ldots, x_T$ so as to maximize W subject to the equation-constraints in (1). The givens of the problem are the value of each α_{it}, the values for each period of the purely exogenous variables, and initial values for the lagged endogenous variables. Assume that x_t is of dimension k, so that there are kT control values to determine. Let x be a kT-component vector denoting these values: $x = (x_1, \ldots, x_T)$. Now, for each value of x, one can compute a value of W by first solving the model in (1) for $y_1, \ldots, y_T$ and then using these values along with the values for $z_1, \ldots, z_T$ and x to compute W in (2). The optimal control problem can thus be looked upon as a problem in choosing variables (the elements of x) to maximize an *unconstrained* nonlinear function. By substitution, the constrained maximization problem is transformed into the problem of maximizing

[4] Stochastic models are discussed in Section 7.

[5] It is assumed throughout this paper that the values of α_{it} and the values of the exogenous variables in the z_t vector are known with certainty.

[6] See, for example, Fromm and Klein [10], pp. 373–382.

an unconstrained function of the control variables:

$$W = \phi(x),$$

where ϕ stands for the mapping $x \to x, y_1, \ldots, y_T, z_1, \ldots, z_T \to W$. In general it will not be possible to express y_t explicitly in terms of z_t, x_t, and α_{it}, so that in general it will not be possible to write W in (2) explicitly as a function of z_t, x_t, and α_{it} ($t = 1, \ldots, T$). Nevertheless, given values for z_t and α_{it} ($t = 1, \ldots, T$), values of W can be obtained numerically for different values of x.

There are many algorithms available for maximizing (or minimizing) nonlinear functions of variables. Since W cannot in general be written as an explicit function of x, it will in general be difficult to obtain analytically the partial derivatives of h with respect to the elements of x. Therefore, in attempting to solve optimal control problems by treating them as problems in maximizing a nonlinear function of variables one will usually be required either to use algorithms that do not require derivatives or else to compute derivatives numerically. Both approaches have been followed for the results in Sections 4 and 5.

Algorithms that do not require derivatives and algorithms for which derivatives are obtained numerically spend most of their time doing function evaluations. For the results in Sections 4 and 5, over 75 percent of the time was spent doing function evaluations for all algorithms tried except in two cases, where the figures were 52 and 53 percent. One function evaluation in the present context corresponds to the solution of a g-equation model for T periods (plus the rather trivial computation, once $y_1, \ldots, y_T$ are determined, of W in (2)). It is therefore quite important to solve a model in the most efficient way possible, since for one solution of the optimal control problem a model will usually be solved hundreds or thousands of times. Some suggestions are presented in Section 6 for efficient ways of solving models.

Much of the engineering literature on optimal control is concerned with continuous-time models and so is not of direct concern here. Polak [20], however, does present a good discussion of the discrete optimal control problem in engineering.[7] The discrete-time model considered by Polak differs from the standard econometric model considered in this paper in that his model is already in reduced form. In the notation of this paper, each component of y_t would be written as an explicit function of z_t, x_t, and α_{it} for Polak's model. The fact that the derivatives of y_t with respect to z_t and x_t can be directly obtained for Polak's model allows Polak to obtain fairly easily the derivatives of the objective function with respect to the values of the control variables. Polak also reports that the time horizon for the problems he is considering may be as large as 1,000 periods,[8] which is much larger than the time horizon for most problems in economics, where the horizon is likely to be much less than even 100 periods. The discrete optimal control problem in economics is thus on the one hand easier than the corresponding problem in engineering in that the time horizon appears to be much smaller and on the other hand more difficult in that analytic derivatives of the objective

[7] See especially pp. 66–71. See also Athans [1] for a discussion of the linear–quadratic–Gaussian stochastic control problem for discrete-time models.

[8] Polak [20], p. 67. Polak does not, however, report on any actual solutions of problems of this sort in his book.

function with respect to the values of the control variables are not easy to obtain because of the non-reduced-form nature of most econometric models.

3. The Computational Algorthims Used

Three basic algorithms were used for the results in Sections 4 and 5. The first is the 1964 algorithm of Powell [21], which does not require any derivatives. The second is a gradient algorithm, which requires first derivatives. The third is the quadratic hill-climbing algorithm of Goldfeld, Quandt, and Trotter [12], which requires both first and second derivatives. The gradient algorithm that was used in this study is a member of the class of algorithms considered by Huang [15].[9] The algorithms within this class basically differ from each other in how the approximation to the inverse of the matrix of second partial derivatives is updated after each iteration. One member of this class is the well-known DFP variable metric algorithm.[10] Some results using the DFP algorithm are reported below, but the main gradient algorithm that was used in this study is the one that updates by means of the "rank one correction formula."[11] This algorithm appears to give the best results. Some results using one other member of the class of algorithms considered by Huang are also reported below.[12] All three of the gradient algorithms considered in this study use linear searches on each iteration.

All of the computer programs were compiled in FORTRAN-H and were run on an IBM 360-91 computer at Princeton University.[13] All derivatives for the gradient and quadratic hill-climbing algorithms were computed numerically. For the gradient algorithms the derivatives were computed in two ways. For one set of runs derivatives were obtained for each iteration by computing two function evaluations per variable, each variable being perturbed by equal amounts around the value available from the previous iteration. For the other set of runs derivatives were obtained for each iteration by computing only one function evaluation per variable. The percentage amount by which variables were perturbed (0.01 percent) was not varied from iteration to iteration.[14] Stewart [25] has proposed a more sophisticated way of computing numeric derivatives when using gradient algorithms, but his method was not tried in this study. For the quadratic hill-climbing algorithm first derivatives were always obtained by computing two function evaluations per variable, as these computations had to be made anyway to obtain the own second derivatives, but the cross partial derivatives were computed in two ways. For one set of runs the cross partial derivatives were obtained by computing four extra function evaluations per set of two variables, and for the other set of runs the derivatives were obtained by computing only one extra

[9] See Powell [23] for an excellent summary of Huang's theory.

[10] See Davidon [7] and Fletcher and Powell [9].

[11] See Powell [23], p. 41.

[12] See Powell [23], equations (31) and (32), p. 41, for a presentation of this algorithm.

[13] The Powell and quadratic-hill-climbing algorithms were programmed by Stephen M. Goldfeld and Richard E. Quandt. The three gradient algorithms were programmed by Thomas Russell.

[14] Let $f(a, b)$ be a function of two variables. Then the formulas that were used to obtain the partial derivative of f with respect to a for the two runs are $(f(a + \varepsilon, b) - f(a - \varepsilon, b))/2\varepsilon$ and $(f(a + \varepsilon, b) - f(a, b))/\varepsilon$, where $\varepsilon = 0.0001a$ or 0.000001, whichever is larger. For all of the runs the problems were set up so that the solution values of the variables would be between about 0.1 and 10.0.

function evaluation per set of two variables.[15] The reason two methods were used to obtain derivatives for the gradient and quadratic hill-climbing algorithms —one more expensive but likely to be more accurate and one less expensive but likely to be less accurate—was to see how sensitive the results were to the way in which the derivatives were obtained. Box, Davies, and Swann [5], for example, report that their experience is that "gradient methods employing numerical differentiation are (with the exception of Stewart, 1967) usually inferior to the best direct search methods, and therefore not recommended."[16] The results in this study do not confirm this view.

In the programs, the algorithms were taken to have converged when the absolute value of the difference between the value of each variable on successive iterations was within a prescribed tolerance level. The Powell algorithm was generally more sensitive to the particular tolerance level used than were the gradient and quadratic hill-climbing algorithms, and for the results in Section 4 two sets of runs were obtained using the Powell algorithm, corresponding to two different tolerance levels.

Studies that have been done comparing different computational algorithms have tended to limit the size of the problems considered to 20 variables or less. This is true, for example, of the comparisons in Bard [3], Box [4], Goldfeld and Quandt [11], Kowalik and Osborne [16], Murtagh and Sargent [17], Pearson [18], and Stewart [25]. Powell [22] reports that the DFP algorithm using analytic derivatives has been successful for problems of size 100 and that his 1964 algorithm and the DFP algorithm using numeric derivatives in the manner proposed by Stewart have solved problems of size 20.[17] Wolfe [26] states that the upper limit to the size of problems that can be solved in which derivatives can be calculated analytically is around 100. For problems in which derivatives cannot be calculated, Wolfe's diagram indicates that the upper limit is about 10.[18] The results reported below indicate that the upper limit to the size of problems that can be solved when derivatives are not calculated analytically is much larger than 10 or 20. The largest problem solved below was of size 239, and a number of problems between size 59 and 100 were solved. In fact, one of the main reasons why the method proposed in this paper appears feasible for most econometric models is the ease in which algorithms appear to be able to solve large problems even when analytic derivatives are not calculated.

4. An Example Using a Linear Model with a Quadratic Objective Function

The method proposed in Section 2 was first used to solve one of the optimal control problems solved by Chow [6] for his nine-equation, linear econometric

[15] Using the notation in footnote 14, the formula used for the own second derivatives is $(f(a+\varepsilon, b) - 2f(a,b) + f(a-\varepsilon, b))/\varepsilon^2$. The two formulas used for the cross partial derivatives are $(f(a+\varepsilon, b+\eta) - f(a-\varepsilon, b+\eta) - f(a+\varepsilon, b-\eta) + f(a-\varepsilon, b-\eta))/4\varepsilon\eta$ and $(f(a+\varepsilon, b+\eta) - f(a, b+\eta) - f(a+\varepsilon, b) + f(a,b))/\varepsilon\eta$, where $\eta = 0.0001b$ or 0.000001, whichever is larger. In the second formula, values for $f(a, b+\eta)$ and $f(a+\varepsilon, b)$ are available from the own second derivative calculations.

[16] Box, Davies, and Swann [5], p. 32.

[17] Powell [22], p. 95

[18] Wolfe [36], pp. xi–xii. It should be noted, however, that it is not clear from Wolfe's notes whether for these particular figures Wolfe is also including problems in which there are inequality constraints.

model. The model has two control variables. Chow solved various 10-period optimal control problems corresponding to different quadratic objective functions (to be minimized). The problem chosen to solve in this study is the second problem in Table 3 of Chow [6]. Two control variables and ten periods means that there are 20 variables to be determined. The initial values for the 20 variables were chosen to be zero, although in practice one could obviously choose better initial values than these. The results of solving this problem are presented in the first row of Table 1. Two runs for the Powell algorithm are reported, one which used a tolerance level of 0.0005 and one which used a tolerance level of 0.00001. Two runs each for the gradient and quadratic hill-climbing algorithm are also reported, corresponding to the two ways of computing derivatives. The latter two algorithms used a tolerance level of 0.00001.

Powell's no-derivative algorithm required 1687 function evaluations to attain the optimum using a tolerance level of 0.0005 and 2,633 function evaluations using a tolerance level of 0.00001. The value of the objective function at the stopping point was smaller for the smaller tolerance level, but only by a very small amount. The corresponding variable values for the two runs agreed to three significant digits, with the largest difference being 0.00015 (0.70272 vs. 0.70287). The gradient algorithm required 614 function evaluations to attain the optimum using one function evaluation per derivative per variable and 1,033 function evaluations using two. The value of the objective function at the stopping point was smaller for the second run, but again by only a very small amount. The corresponding variable values for these two runs also agreed to three significant digits. The quadratic hill-climbing algorithm required 929 function evaluations to attain the optimum using one function evaluation per cross derivative and 3,209 function evaluations using four. For these two runs the values of the objective function at the stopping point were the same. The time per function evaluation for the Chow-model, 10-period problem was 0.0018 of a second. The optimum obtained for this problem was the same as Chow had obtained.

The optimal control problem for the Chow model was next made progressively larger by increasing the time horizon. The largest problem considered was a time horizon of 50 periods, which meant that there were 100 variables to estimate. The results for 40, 60, 80, and 100 variables are presented in rows 2 through 5 in Table 1 respectively. For the various problems the gradient algorithm clearly dominated Powell's in terms of speed of convergence. The use of the smaller tolerance level for the Powell algorithm increased the number of function evaluations considerably, and the values of the objective functions at the stopping points were only slightly larger for the larger tolerance level. Likewise, for the gradient algorithm the values of the objective functions at the stopping points were only slightly larger for the runs using one function evaluation per derivative. For the quadratic hill-climbing algorithm no accuracy at all was lost using one function evaluation per cross derivative. The quadratic hill-climbing algorithm was not tried after 40 parameters, although the use of the algorithm for problems of, say, size 100 is not completely out of the question. Using the less expensive way of obtaining cross derivatives, it requires $0.5N^2 + 1.5N$ function evaluations to compute the vector of first derivatives and the matrix of second partial derivatives per iteration (where N is the number of variables). If four iterations are required

TABLE 1

RESULTS FOR CHOW MODEL

						Powell				Gradient				Hill-Climbing				
	No. of Control Vars.	No. of Periods	No. of Variables	Starting Point	Time per Funct. Eval. (sec.)	Time (sec.)	No. of Funct. Evals.	No. of Iterations	Value of Obj. Funct. $(10)^9$	Time (sec.)	No. of Funct. Evals.	No. of Iterations	Value of Obj. Funct. $(10)^9$	Time (sec.)	No. of Funct. Evals.	No. of Iterations	Value of Obj. Funct. $(10)^9$	Value of Obj. Funct. at Starting Point $(10)^9$
1	2	10	20	0's	0.0018	3.27	1,687	14	0.12494578	1.33	614	21	0.12494570	3.21	929	4	0.12494558	10.868368
						4.70	2,633	23	0.12494558	1.96	1,033	21	0.12494558	7.07	3,209	4	0.12494558	
2	2	20	40	0's	0.0034	17.38	4,752	20	0.47827735	6.44	1,575	32	0.47827742	22.21	3,450	4	0.47827668	44.715432
						23.51	6,762	30	0.47827668	10.42	2,854	32	0.47827668	55.41	12,810	4	0.47827668	
3	2	30	60	0's	0.0050	41.78	7,820	22	1.4724707	17.81	2,945	42	1.4724725			nc		149.58890
						54.37	10,830	32	1.4724700	29.95	5,465	42	1.4724700					
4	2	40	80	0's	0.0067	81.29	10,960	23	4.2400571	34.27	4,432	49	4.2400653			nc		463.37069
						102.83	15,371	34	4.2400560	63.42	8,517	50	4.2400560					
5	2	50	100	0's	0.0085	129.94	14,883	25	11.875411	57.54	5,854	53	11.875442			nc		1,378.3241
						167.62	19,893	35	11.875411	104.55	11,156	53	11.875411					
6	2	40	80	a	0.0067	42.60	6,253	15	4.2400580	12.04	1,396	16	4.2400573			nc		4.5209593
						42.60	6,253	15	4.2400580	20.22	2,842	17	4.2400560					
*7	2	30	60	0's	0.0050			—		26.14	4,499	64	1.4724720					149.58890
										40.28	7,455	57	1.4726138					
†8	2	30	60	0's	0.0050			—		27.69	4,836	68	1.4724713					149.58890
										47.57	8,790	67	1.4724700					

Notes:
Powell = Powell [21].
Gradient = Gradient algorithm that uses rank one correction formula. See Powell [23], p. 41.
Hill-climbing = Goldfeld, Quandt, and Trotter [12].
nc = not computed.
[a] Answer to 3 plus trend values otherwise.
* Gradient = DFP variable metric algorithm. See Powell [23], p. 37.
† Gradient = Algorithm presented in equations (31) and (32) in Powell [23], p. 41.
Chow model: 9 equations; linear; lags of first order; welfare function is quadratic.
First run for Powell for each problem used tolerance level of 0.0005. Second run used 0.00001.
First run for Gradient for each problem used less expensive way of obtaining first derivatives. Second run used more expensive way.
First run for Hill-climbing for each problem used less expensive way of obtaining cross partial derivatives. Second run used more expensive way.
Tolerance level for Gradient and Hill-climbing was 0.00001.

to attain convergence, then roughly 20,600 function evaluations would be required to solve the 100-variable problem.

Adding extra periods for the Chow model in general had little effect on the optimal variable values of previous periods, so that, for example, the answer to the 60-variable problem was close to the answer to the 80- or 100-variable problem for the first 60 variables. In view of this, the answer to smaller problems should be a good starting point for larger problems, and so to test this, the answer to the 60-variable problem was used as a starting point for the first 60 variables of the 80-variable problem. Starting points for the other 20 variables were obtained by letting the values of the two control variables grow by 6 and 5 percent respectively, these figures being obtained by observing how the control variables were growing in the answer to the 60-variable problem. The results of this test are presented in row 6 of Table 1. For the gradient algorithm the number of function evaluations was cut by about a factor of 3 (from 4,432 to 1,396 and from 8,517 to 2,842), a substantial savings. For the Powell algorithm the number of function evaluations was cut from 10,960 to 6,253 using the larger tolerance level and from 15,371 to 6,253 using the smaller tolerance level. In both cases for the Powell algorithm, a slightly smaller value of the objective function was obtained by starting the variable values from zero.

As a final test using the Chow model, two other gradient algorithms were tried for the 60-variable problem. The results are reported in rows 7 and 8 of Table 1. Neither algorithm worked as well as the rank one algorithm. The DFP algorithm required about 1,554 more function evaluations than did the rank-one algorithm for the run using one function evaluation per derivative. For the run using two function evaluations per derivative, the DFP algorithm did not quite attain the optimum.

5. An Example Using a Nonlinear Model with a Non Quadratic Objective Function

The method of Section 2 was next used to solve a more complicated optimal control problem. The model used was the Fair model [8], less the monthly housing starts sector. The model used consists of 19 equations, is nonlinear, has lags of up to eighth order, and was estimated under the assumption of first-order serial correlation of most of the error terms.[19] The initial period was taken to be 1962III and the horizon for the various runs was either 10, 20, 25, or 60 quarters. The number of control variables was varied between one and four. Government spending was always taken to be a control variable. The other three variables that were sometimes used as control variables were the level of consumer sentiment, plant and equipment investment expectations, and nonfarm quarterly housing starts. These latter three variables are clearly not variables under the direct control of the government, but for purposes of illustrating the method of solution, there is no harm in treating them as if they were. The objective function was deliberately chosen to be non-quadratic in the variables of the model. The objective function

[19] The coefficients were taken from Table 11-4 in [8].

(to be minimized) was:

$$W = \sum_{t=1}^{T} \left\{ 10(g_{\mathrm{PD}_t})^2 + 10/\mathrm{UR}_t - 0.030/^2 + \left(\frac{\mathrm{CD}_t}{\mathrm{GNP}_t} - 0.094\right)^2 + \left(\frac{\mathrm{CN}_t}{\mathrm{GNP}_t} - 0.275\right)^2 + \left(\frac{\mathrm{CS}_t}{\mathrm{GNP}_t} - 0.257\right)^2 + \left(\frac{\mathrm{IP}_t}{\mathrm{GNP}_t} - 0.101\right)^2 + \left(\frac{\mathrm{IH}_t}{\mathrm{GNP}_t} - 0.038\right)^2 \right\},$$

where g_{PD_t} is the rate of growth (at an annual rate) of the private output deflator, UR_t is the unemployment rate, and the five ratios are the ratios of durable consumption, non-durable consumption, service consumption, plant and equipment investment, and housing investment to gross national product respectively. The slashes around $\mathrm{UR}_t - 0.030$ denote the fact that $/\mathrm{UR}_t - 0.030/$ was taken to be equal to $\mathrm{UR}_t - 0.030$ if $\mathrm{UR}_t \geq 0.030$ and zero otherwise. In other words, welfare was not improved for an unemployment rate below 0.030, but it was not decreased either, as a straight quadratic function would imply. The objective function is non-quadratic in this respect, as well as in targeting *ratios* of the various components of GNP to GNP itself. The rate of inflation and the unemployment rate were weighted ten times more heavily in the objective function than were the ratios. It should be noted that the welfare function is not differentiable at $\mathrm{UR}_t = 0.030$. In the present case, however, the optimum values of UR_t were always greater than 0.030, and the lack of differentiability at $\mathrm{UR}_t = 0.030$ did not appear to be a problem for the algorithms for which numeric derivatives had to be computed. In general, if the lack of differentiability of either the model or the welfare function appears to be important (as it might be, for example, for models in which capacity ceilings play an important role), then algorithms that do not require the computation of derivatives may be better choices than those that do.

The results for the various runs using the Fair model are presented in Table 2. The second control variable, the level of consumer sentiment, does not enter the model currently, but only with lags of one or more periods, so when this variable was used as a control variable, the number of values of this variable to be determined was one less than the number of periods. Except for lines 7 and 8, historic values were used as starting points for the values of the control variables. Again, two runs each for the gradient and quadratic hill-climbing algorithms are reported, corresponding to the two ways of computing derivatives. The tolerance level used for these two algorithms was 0.00005. The tolerance level used for the Powell algorithms was 0.000005.

From the results in Table 2, it can be seen that the gradient algorithm worked better than Powell's. The number of function evaluations was usually less for the gradient algorithm, and for the problems of greater than 20 variables the Powell algorithm did not quite attain the optima that the gradient algorithm did. For the 39- through 99-variable problems, the largest differences between the variable values computed by the Powell algorithm and the corresponding variable values computed by the gradient algorithm were 26, 8, 34, and 88 percent respectively. An even smaller tolerance level was tried for some of the runs using the Powell

TABLE 2

RESULTS FOR FAIR MODEL

						Powell				Gradient				Hill-Climbing				
	No. of Control Vars.	No. of Periods	No. of Variables	Starting Point	Time per Funct. Eval. (sec.)	Time (sec).	No. of Funct. Evals.	No. of Iterations	Value of Obj. Funct.	Time (sec.)	No. of Funct. Evals.	No. of Iterations	Value of Obj. Funct.	Time (sec.)	No. of Funct. Evals.	No. of Iterations	Value of Obj. Funct.	Value of Obj. Funct. at Starting Point
1	1	10	10	hist.	0.0072	8.40	1,123	12	0.076917053	1.35	163	9	0.076917048	2.33	269	4	0.076917047	
				val.						1.80	225	8	0.076917048	6.43	809	4	0.076917047	0.1051447
2	1	20	20	hist.	0.0148	32.67	2,152	12	0.16427133	4.78	307	11	0.16427053	15.45	929	4	0.16427052	
				val.						8.71	573	12	0.16427052	52.16	3,209	4	0.16427052	0.2008097
3	2	20	39	hist.	0.0148	48.16	3,142	10	0.16380396	20.83	1,371	29	0.16373695	63.26	3,284	4	0.16373693	
				val.						40.39	2,756	32	0.16373693	198.52	12,176	4	0.16373693	0.2008097
4	3	20	59	hist.	0.0148	84.36	5,580	10	0.16290711	35.82	2,344	35	0.16288105			nc		
				val.						73.84	4,787	38	0.16288104					0.2008097
5	4	20	79	hist.	0.0148	156.20	10,513	15	0.16250254	113.82	7,314	84	0.16234170			nc		
				val.						197.26	12,807	77	0.16234169					0.2008097
6	4	25	99	hist.	0.0181	231.40	12,822	14	0.20624995	204.47	10,181	95	0.20511032			nc		
				val.						384.73	19,631	95	0.20511030					0.3035085
7	4	60	239	a	0.0439			nc		1189.51	25,661	104	0.58885958			nc		
												nc						0.8073080
8	4	20	79	b	0.0148	39.11	2,615	4	0.16249898	110.60	7,047	81	0.16234170			nc		
										196.45	12,793	77	0.16234169					0.1628810
*9	3	20	59	hist.	0.0148			—		35.12	2,305	34	0.16288106			—		
				val.						73.98	4,821	38	0.16288104					0.2008097
†10	3	20	59	hist.	0.0148			—		35.53	2,336	35	0.16288105			—		
				val.						72.55	4,777	38	0.16288104					0.2008097

Notes:
See Table 1.
Tolerance level for Powell was 0.000005.
Tolerance level for Gradient and Hill-climbing was 0.00005.
* Gradient = DFP variable metric algorithm.
† Gradient = Algorithm presented in equations (31) and (32) in Powell [23], p. 41.
[a] Answer to 6 plus trend values otherwise.
[b] Answer to 4 plus historical values otherwise.
Fair model: 19 equations; nonlinear; lags of up to eighth order; first-order serial correlation of the error terms in most of the equations; welfare function is not quadratic.

algorithm (0.0000001 vs. 0.000005) to see if this resulted in a smaller value of the objective function, but the results were not improved using the smaller tolerance levels. For the gradient algorithm the use of the less expensive way of obtaining derivatives resulted in virtually no loss in accuracy for any of the runs. For the quadratic hill-climbing algorithm the use of the less expensive way of computing cross partial derivatives resulted in no loss in accuracy at all and, of course, substantial savings on cost. For the problem of 4 control variables and 25 periods (99 variables), the gradient algorithm using the less expensive way of computing derivatives required 10,181 function evaluations and took about 3.4 minutes to attain the optimum.

When the 79-variable problem was started from the answer to the 59-variable problem plus historical values otherwise (line 8), the speed of convergence was only slightly increased for the gradient algorithm. The number of function evaluations fell from 7,314 to 7,047 for the one run and from 12,807 to 12,793 for the other. The number of function evaluations fell substantially for the Powell algorithm, but the optimum was still not attained.

When the other two gradient algorithms were tried for the 59-variable problem (lines 9 and 10), the results were virtually the same as for the rank one algorithm. For this problem there is nothing to choose among the three algorithms.

The largest problem tried for the Fair model was four control variables and 60 periods (1962III–1977II) for a total of 239 variables. The answer to the 99-variable problem was used as a starting point plus historical or extrapolated values otherwise. Only the gradient algorithm using the less expensive way of obtaining derivatives was tried for this problem. The program was allowed to run for approximately 20 minutes. At the end of 20 minutes and 104 iterations, the value of the objective function was changing only in the eighth decimal place between iterations and the largest difference between any corresponding parameter values on the last two iterations was 0.0007. The value of the objective function at the starting point was 0.80730797 and the value after 104 iterations was 0.58885958. The starting point turned out to be fairly far away from the stopping point, with unemployment rates of about 7 percent near the end of the horizon compared with the stopping-point values of around 5 percent. The stopping-point values for the 239-variable problem appeared to be in line with what would be expected from observing the answers to the smaller problems. The Powell algorithm was started from the values attained by the gradient algorithm on the 53rd iteration (an objective-function value of 0.58890611) to see if it would go anywhere. A tolerance level of 0.000005 was used. The algorithm went one iteration, lowered the objective function to 0.58890571, and stopped (the convergence criterion having been met for all parameters), a clear failure in view of the value obtained by the gradient algorithm. One other result is also of interest to note here. The gradient algorithm was also started from the values attained on the 53rd iteration. A tolerance level of 0.00005 was used. The algorithm went one iteration, lowered the objective function to 0.58890575, and stopped (the convergence criterion having been met), also a clear failure. By starting the gradient algorithm over on the 53rd iteration, one lost the approximation to the inverse of the matrix of second partial derivatives that had been developed over 53 iterations, which in the present case was obviously quite important. A similar result occurred when experimenting with the 99-variable problem. These results suggest that if one contemplates having to restart the gradient algorithm for one reason or another (like running out of time on the computer), one ought to save the latest approximation to the inverse of the matrix of second partial derivatives to be used when the algorithm is restarted. The results also suggest, oddly enough, that when using the gradient algorithm one ought not to start the algorithm too close to the (presumed) optimum for fear that the algorithm will get stuck before it has a chance to build up a good approximation to the inverse of the matrix of second partial derivatives.

The answers to the problems for the Fair model were characterized by a large value of government spending in the first period (compared with the historical value) and large values near the end of the time horizon. In the model employment responds faster to government spending than does the price level, and so the relatively large values of government spending for the last few periods of the horizon are taking advantage of this fact and lowering the unemployment rate without having too much effect on the price level.[20] The large value of spending in the first period is apparently designed to lower the unemployment rate quickly from its relatively high historic level. Excluding beginning and ending effects, the particular objective function used resulted in an unemployment rate of about 5.0 percent and an annual rate of inflation of about 2.2 percent. The IP_t/GNP_t and IH_t/GNP_t ratios were met almost exactly when plant and equipment investment expectations and housing starts were used as control variables, as would be expected. The three consumption ratios were not met as exactly when consumer sentiment was used as a control variable since in this case there was, in effect, only one main control variable influencing three ratios.

In Table 3 are presented estimates for each run in Tables 1 and 2 of the percentage of time that was spent doing function evaluations. The estimates were obtained by multiplying the time per function evaluation by the number of function evaluations and dividing this figure by the total time for the job. For the Fair model abnormal exits sometimes occurred from the function-evaluation program (before all of the computations were performed), which causes some of the percentages for the Fair model in Table 3 to be too high. Abnormal exits occur when variable values imply that the logarithm of a negative number should be taken. The estimates in Table 3 are also subject to error for reasons that have to do with the way that computation time in the computer is estimated. In general, the percentages are quite high in Table 3, indicating the importance of writing efficient programs for evaluating functions.

6. An Evaluation of the Practical Usefulness of the Method

The results in Sections 4 and 5 are very encouraging as to the feasibility of using the method proposed in Section 2 even for large-scale models. For a 20-period problem the 19-equation Fair model takes 0.0148 of a second per function evaluation on the IBM 360-91 computer. The Fair model can be solved without

[20] To avoid undesirable end-point effects in practice, one can always extend the horizon a few periods beyond the actual horizon of interest. For the Fair model it appeared that the horizon should be lengthened by about 5 quarters. Because of the end-point effects, the last few answers to the 99-variable problem for each control variable were not used as starting points for the 239-variable problem.

TABLE 3

ESTIMATES OF PERCENTAGE OF TIME SPENT DOING FUNCTION EVALUATIONS

From Table 1

	Powell		Gradient		Hill-Climbing	
Row	(1)	(2)	(1)	(2)	(1)	(2)
1	93	97	83	95	52	82
2	93	95	83	93	53	79
3	94	90	83	91	—	—
4	90	92	87	90	—	—
5	97	95	86	91	—	—
6	98	98	78	94	—	—
7	—	—	86	93	—	—
8	—	—	87	92	—	—

From Table 2

	Powell	Gradient		Hill-Climbing	
Row		(1)	(2)	(1)	(2)
1	96	87	90	83	91
2	97	95	97	89	91
3	97	97	101	77	91
4	98	97	96	—	—
5	100	95	96	—	—
6	100	90	92	—	—
7	—	95	—	—	—
8	99	94	96	—	—
9	—	97	96	—	—
10	—	97	97	—	—

the use of the Seidel method since the nonlinear part of the model is recursive. If a 100-equation model could be solved in the same way, it should take only about five times longer to solve this model than it takes to solve the Fair model since the number of computations per equation is not likely to vary much from model to model. Econometric models tend to be larger because of more equations and not because of more variables per equation. If the Seidel method must be used to solve a model and if for each iteration for each period the entire model must be passed through, then the cost per solution of the model is increased in proportion to the number of iterations that are required to solve the model each period. If, for example, it takes five iterations to solve a 100-equation model each period, it should take about 25 times longer to solve this model than it takes to solve the Fair model. Since algorithms that do not require derivatives or for which derivatives are computed numerically spend most of their time doing function evaluations, the total time that it takes to solve a control problem for a 100-equation model that requires five iterations per solution of the model should be about 25 times greater for the same problem than the corresponding time in Table 2 for the Fair model. A 20-period problem with one control variable should thus take about 2.0 minutes using the gradient algorithm and the less expensive way of obtaining derivatives (25 × 4.78 seconds). A 20-period problem with two control variables should take about 8.7 minutes (25 × 20.83 seconds). The problem of four control variables and 25 periods should take about 85.2 minutes (25 × 204.47 seconds).

Although the times just mentioned are not completely out of the range of practicality, it is possible that in practice the times can be substantially cut down. First, good starting points can be quite important, and significant time may be saved by first solving a small problem (say one control variable), using the answer to this problem as a starting point for a somewhat larger problem (say two control variables), and so on, building up to the largest problem that one wants to consider. Also, once one has solved a particular optimal control problem once, the answer to this problem may be a good starting point for a slightly different problem (say, a slight change in the objective function). In other words, it may not be too costly to experiment with different objective functions or a slightly different specification of the model once one solution to a particular problem has been obtained. It may also be the case that from a welfare point of view or from the point of view of feasibility one wants to keep the control variables within certain bounds. This can be done by including control variables in the objective function and penalizing deviations of the values of the control variables from target values. If this is done, one has a natural starting point for the control variables—the target values—and this may significantly increase the speed of convergence of the algorithm being used, in addition perhaps to decreasing the likelihood that the algorithm goes to a local but not the global optimum.

A second way in which much time might be saved by models that need to be solved by the Seidel method is by choosing good initial values of the endogenous variables to begin the solution of the model each period. Since most algorithms perturb the variables (in the presence case, the values of the control variables) only a slight amount between function evaluations, particularly when derivatives are being computed, a good choice for the initial values of the endogenous variables is likely to be the solution values obtained in the process of computing the previous function evaluation. It is possible that this choice can cut the number of iterations needed per solution of the model per period to two or three, which would greatly save on cost.

A third way in which time can be saved is to write the program that does function evaluations in such a way that no computations are performed other than those that are absolutely needed in going from values of the control variables to the value of the objective function. For example, any sets of calculations using exogenous variables that are not changed as a result of changes in the values of the control variables should not be done in the function-evaluation program, but only once before the solution of the optimal control problem begins. This kind of efficient programming was not done for the results in Tables 1 and 2.

If for a 100-equation model one could, by following the above suggestions, cut the number of iterations using the Seidel method to an average of 2.5 and could further cut the time per function evaluation by 25 percent, then the times quoted above (2.0, 8.7, and 85.2 minutes) would be cut to 0.75, 3.3, and 32.0 minutes respectively. These times may be further cut by a factor of 2 or more by better choices of initial parameter values than those used for the results in Table 2.[21]

In terms of the size of the problems that the method proposed in this paper can handle, there is an obvious tradeoff between the size of the model, the number of control variables, and the length of the decision horizon. It is hard to establish any precise rules as to what problems are practical to solve and what are not because no two models and problems are the same. Furthermore, for some problems one algorithm may work best and for others another may work best. Each person must to some extent determine for oneself through experimentation the practical limits to the size of problems that one can solve. Nevertheless, the results in this study can give some indication of the likely cost of various problems. One important question in this regard is how rapidly the number of function evaluations increases as the number of variables to be estimated increases. From the results in Tables 1 and 2 one can compute the extra number of function evaluations required per additional variable ($\Delta\text{FE}/\Delta N$, where FE is the number of function evaluations and N is the number of variables) and observe how this quantity varies as the total number of variables varies. These computations are presented in Table 4. For the quadratic hill-climbing algorithm, $\Delta\text{FE}/\Delta N$ clearly increases as N increases since the number of function evaluations required to compute first and second derivatives per iteration increases as the square of N. From the results for the Chow model there is only a slight tendency for $\Delta\text{FE}/\Delta N$ to increase as N increases for the Powell and gradient algorithms. From the results for the Fair model there is somewhat more of a tendency in this direction for the two algorithms, but this tendency is far from being uniform. In general, the results in Table 4 indicate that there is only a slight tendency for $\Delta\text{FE}/\Delta N$ to increase as N increases for the Powell and gradient algorithms.

The time required per function evaluation should be roughly proportional to the number of periods times the number of equations in the model times the number of Seidel iterations required to solve the model. The time required to solve a control problem is roughly equal to the time required per function evaluation times the number of function evaluations. If the number of function evaluations varies only in proportion to the number of variables ($\Delta\text{FE}/\Delta N$ not increasing as N increases), then the time required to solve a control problem should be roughly proportional to the square of the number of periods times the number of control variables times the number of equations times the number of Seidel iterations. In this case, if the number of Seidel iterations required to solve a model does not increase as the number of equations of the model increases, then the time

[21] Albert Ando has communicated to the author a "conservative" estimate that for the solution of the 200-equation FMP model it takes about 0.00500 of a second per iteration per period on an IBM 370-165 computer. This figure compares with 0.00072 for the solution of the 19-equation Fair model (divide 0.0072 in Table 2 by 10). Since the FMP model has 10.5 times more equations than the Fair model, one would expect the time per iteration per period to be about 10.5 times greater for the FMP model. The figure supplied by Ando indicates that the time is only 6.9 times greater. Ando's results thus suggest that the times cited in the text above may be too conservative. It should also be noted that Ando's results are for a program that was not written with optimal control problems in mind.

The FMP model usually takes between 10 and 15 iterations to solve per period using the Seidel method. However, the values used as initial values for the endogenous variables are the solution values of the previous quarter, and, as suggested above, in an optimal-control context one should be able to do much better than this.

TABLE 4

VALUES OF $\Delta FE/\Delta N$

From Table 1

		Powell		Gradient		Hill-Climbing	
N	ΔN	(1)	(2)	(1)	(2)	(1)	(2)
20	20	84.4	141.6	30.7	51.7	46.5	160.5
40	20	153.3	216.8	48.1	91.1	126.1	480.1
60	20	153.4	214.5	68.5	130.6	—	—
80	20	157.0	238.1	74.4	152.6	—	—
100	20	196.2	236.1	71.1	132.0	—	—

From Table 2

			Gradient		Hill-Climbing	
N	ΔN	Powell	(1)	(2)	(1)	(2)
10	10	112.3	16.3	22.5	26.9	80.9
20	10	102.9	14.4	34.8	66.0	240.0
39	19	52.1	56.0	114.9	123.9	471.9
59	20	121.9	48.7	101.6	—	—
79	20	246.7	248.5	401.0	—	—
99	20	115.5	143.4	341.2	—	—
239	140	—	110.6[a]	—	—	—

N = number of variables. FE = number of function evaluations.

[a] The 239-variable run was started from a more accurate point than the others and was terminated at a tolerance level of only .0007 versus .00005 for the others.

required to solve a control problem should increase only in proportion to the increase in the number of equations. Otherwise, the time will increase more than in proportion to the increase in the number of equations.[22] The time required to solve a control problem is proportional to the *square* of the number of periods because an increase in the number of periods increases both the number of variables and the time required per function evaluation. If the number of function evaluations increases more than in proportion to the number of variables, then the time required to solve a control problem will increase more than in proportion to the increase in the square of the number of periods times the number of control variables.

Barring further results, some tentative conclusions can be drawn from the results in this study as to the size of problems that it appears feasible to solve using the method discussed in Section 2. For models of about 20 equations, it appears quite practical to solve problems in which the product of the number of control variables and the number of periods is greater than 100. For models of about 100 equations, a product of 100 is probably within the range of practicality. For models of about 200 equations, a product of 60 may be close to the limit of practicality. The use of good starting points and efficient programming may, of course, greatly extend even these limits. Since most econometric models do not

[22] If the objective function to be maximized becomes less well behaved as the number of equations increases, this should also cause the time required to solve a control problem to increase more than in proportion to the increase in the number of equations. Without further experimentation using other models it is not clear how sensitive the shape of the objective function is likely to be to the number of equations in the model.

exceed 200 equations and since the number of control variables in any one model can usually be kept under, say, five without seriously restricting the problem, the method considered in this paper should be able to handle most problems of interest to policy makers who use econometric models in their decision-making process. It should also be noted that the method considered in this paper requires relatively little human effort. All one has to do is write a program to solve the model and compute the objective function. No derivatives are required, no analytic approximations have to be made, and the model does not have to be set up in any special form.

The results in Tables 1 and 2 indicate that the gradient algorithm using the less expensive way of obtaining derivatives is the most efficient. Slightly more accuracy may be obtained by using the more expensive way of obtaining derivatives or by using the quadratic hill-climbing algorithm, but in general this increased accuracy is not likely to be worth the cost. For the quadratic hill-climbing algorithm no accuracy was gained using the more expensive way of computing cross partial derivatives, and so this way is not recommended. The Powell algorithm was generally more expensive than the gradient algorithm, and for the Fair model it had a tendency to get close to but not quite to the optimum. The results in the two tables do, of course, indicate that quite large problems can be solved even when derivatives are obtained numerically. In practice, it may be desirable, after having attained an answer from one algorithm, to start another algorithm from this answer to be more certain that the true optimum has been attained. The quadratic hill-climbing algorithm, while being the most expensive for large problems, is likely to be the most robust to attaining the true optimum.

7. Stochastic Models

In the case of a linear model with additive error terms and a quadratic objective function it is well known that solving the deterministic control problem derived by setting the error terms to their expected values will provide the optimal first-period control values for the stochastic, closed-loop, feedback control problem. Therefore, if one solves the deterministic control problem each period, after observations on the state of the system for the previous period become available, one will over time make the same decisions regarding the current values of the control variables (i.e., the values of the control variables that the decision maker actually sets) as would be made by one who had solved the stochastic, closed-loop, feedback control problem explicitly in terms of feedback equations. To this extent, feedback equations need not be obtained, and one can concentrate on solving deterministic control problems as considered in the previous sections of this paper.[23] For most economic applications sufficient time is usually available to recompute the entire sequence of optimal controls each period.

For nonlinear models the first-period certainty-equivalence property does not hold. One procedure that might be followed in this situation is merely to treat the nonlinear-model case in the same way as one would treat the linear-model case, i.e., setting error terms to their expected values, and solve the deterministic control

[23] Knowledge of feedback equations for a particular model may aid one in understanding the dynamic properties and other characteristics of the model, and for this reason it may be useful to compute feedback equations even though they are not actually needed for the solution of the optimal control problem.

problem each period. This procedure is probably the one most often used in practice for solving nonlinear models, although Howrey and Kelejian [14] have shown that solving a nonlinear model by setting the error terms equal to their expected values is not equivalent to solving the reduced-form equations of the model.

For a nonlinear model the mean values of the endogenous variables can be obtained by means of stochastic simulation. A number of drawings from the joint probability distribution of the error terms can be taken, and for each drawing one can obtain by solving the model a set of values for the endogenous variables. The mean value for each endogenous variable can then be computed as the average of the values obtained from solving the model for the various drawings. Using the procedure of stochastic simulation, it may be possible for relatively small problems to obtain optimal open-loop controls for nonlinear, stochastic models in a manner similar to that done above for nonlinear, deterministic models. Say the aim were to maximize the expected value of the objective function. For each choice of control values, one could compute by means of stochastic simulation the mean value of W. The computed mean value of W would be the value returned to the maximization algorithm, and the algorithm would be used in the usual way in an attempt to find that set of control values for which the mean value of W were at a maximum. Each function evaluation in the stochastic case would correspond to an entire stochastic simulation. If, for example, 50 drawings from the joint probability distribution of the error terms were needed to obtain an adequate approximation to the expected value of W, then approximately 50 times more time would be needed per function evaluation for the stochastic problem then for the deterministic problem. Even though the cost is high for the stochastic problem, it may be feasible for small problems to carry out the above suggestion. If one did carry out the above suggestion and found the optimum and if one recomputed the entire sequence of optimal controls each period, one would over time make the same decisions regarding the current values of the control variables as would be made by one who had solved the stochastic, open-loop, feedback control problem explicitly in terms of feedback equations.

For the control problem for nonlinear, stochastic models, Athans [1], [2] has suggested first solving the deterministic control problem (the deterministic problem being obtained by setting the error terms equal to their expected values) and then linearizing around the deterministic-control paths to obtain linear feedback equations around the paths. The aim is over time to keep the actual paths close to the deterministic-control paths. While Athans' suggestion may be useful for engineering applications, where reoptimization each period may not be feasible, the suggestion is likely to be of less use for economic applications. If sufficient time is available to reoptimize each period, then it is much more straightforward just to solve the deterministic control problem each period.[24] The results in this paper

[24] These remarks should not be interpreted as meaning that Athans would necessarily disagree with them. For example, Athans [1], p. 449, has stated: "It should be stressed that trends in stochastic control research by engineers has been greatly influenced by two factors: (a) a need to minimize on-line computations, and (b) the requirements in many aerospace applications that the control system be realized by analog hardware.

In economic applications these requirements are not present, since the time period between decisions does allow for extensive digital computer calculations. Thus, one does have the luxury of examining more sophisticated decision and control algorithms, which however have increased computational requirements."

certainly indicate that it is feasible to reoptimize each period when, say, the period is a month or a quarter. The procedure of reoptimizing each period is also somewhat more appealing on intuitive grounds than Athans' procedure. If stochastic simulation is ruled out, then both procedures are based on the incorrect practice of setting error terms equal to their expected values. If one follows Athans' procedure, however, further approximations have to be made that do not have to be made if one reoptimizes each period.

Princeton University

References

[1] Athans, Michael, "The Discrete Time Linear–Quadratic–Gaussian Stochastic Control Problem," *Annals of Economic and Social Measurement*, I (October 1972), 449–491.

[2] Athans, Michael, "The Role and Use of the Linear–Quadratic–Gaussian Problem in Control System Design," *IEEE Transactions on Automatic Control*, AC-16 (December 1971), 529–552.

[3] Bard, Yonathan, "Comparison of Gradient Methods for the Solution of Non-Linear Parameter Estimation Problems," *SIAM Numerical Analysis*, VII (March 1970), 157–186.

[4] Box, M. J., "A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems," *Computer Journal*, IX (May 1966), 67–77.

[5] Box, M. J., D. Davies, and W. H. Swann, *Non-Linear Optimization Techniques*, Oliver and Boyd Ltd., Edinburgh, 1969.

[6] Chow, Gregory C., "How Much Could be Gained by Optimal Stochastic Control Policies," *Annals of Economic and Social Measurements*, I (October 1972), 391–406.

[7] Davidon, W. C., "Variable Metric Method for Minimization," A.E.C. Research and Development Report ANL-5990 (Revised), 1959.

[8] Fair, Ray C., *A Short-Run Forecasting Model of the United States Economy*, D. C. Heath and Co., Lexington, 1970.

[9] Fletcher, R. and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal*, VI (July 1963), 163–168.

[10] Fromm, Gary and Lawrence R. Klein, "Solutions of the Complete System," in Duesenberry, James S., Gary Fromm, Lawrence R. Klein, and Edwin Kuh, *The Brookings Model: Some Further Results*, Rand McNally & Co., Chicago, 1969, 362–421.

[11] Goldfield, Stephen M. and Richard E. Quandt, *Nonlinear Methods in Econometrics*, North-Holland Publishing Co., Amsterdam, 1972.

[12] Goldfeld, Stephen M., Richard E. Quandt, and Hale F. Trotter, "Maximization by Quadratic-Hill-Climbing," *Econometrica*, XXXIV (July 1966), 541–551.

[13] Holbrook, Robert S., "A Practical Method for Controlling a Large Nonlinear Stochastic System," this issue.

[14] Howrey, E. Philip and H. H. Kelejian, "Simulation versus Analytical Solutions: The Case of Econometric Models," Chapter 12, in Naylor, Thomas H., *Computer Simulation Experiments with Models of Economic Systems*, John Wiley & Sons, New York, 1971.

[15] Huang, H. Y., "Unified Approach to Quadratically Convergent Algorithms for Function Minimization," *Journal of Optimization Theory and Applications*, V (June 1970), 405–423.

[16] Kowalik, J. and M. R. Osborne, *Methods for Unconstrained Optimization Problems*, Elsevier Publishing Co., New York, 1968.

[17] Murtagh, B. A. and R. W. H. Sargent, "Computational Experience with Quadratically Convergent Minimization Methods," *Computer Journal*, XIII (May 1970), 185–194.

[18] Pearson, J. D., "On Variable Metric Methods of Minimization," *Computer Journal*, XI (May 1969), 171–178.

[19] Pindyck, Robert S., "Optimal Stabilization Policies via Deterministic Control," *Annals of Economic and Social Measurement*, I (October 1972), 385–389.

[20] Polak, E., *Computational Methods in Optimization*, Academic Press, New York, 1971.

[21] Powell, M. J. D., "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives," *Computer Journal*, VII (July 1964), 155–162.

[22] Powell, M. J. D., "A Survey of Numerical Methods for Unconstrained Optimization," *SIAM Review*, XII (January 1970), 79–97.

[23] Powell, M. J. D., "Recent Advances in Unconstrained Optimization," *Mathematical Programming*, I (October 1971), 26–57.

[24] Shupp, Franklin R., "Uncertainty and Stabilization for a Nonlinear Model," *The Quarterly Journal of Economics*, LXXXVI (February 1972), 94–110.

[25] Stewart, G. W., III, "A Modification of Davidon's Minimization Method to Accept Difference Approximations of Derivatives," *Journal of the Association of Computing Machinery*, XIV (January 1967), 72–83.

[26] Wolfe, Philip, A review of some notes of Philip Wolfe in Fletcher, R., ed., *Optimization*, Academic Press, London, 1969, xi–xv.