Annals of Economic and Social Measurement, 6/5, 1978

AN OPTIMIZATION CODE FOR NONLINEAR ECONOMETRIC MODELS BASED ON SPARSE MATRIX TECHNIQUES AND REDUCED GRADIENTS*

BY ARNE DRUD

The paper describes an implementation of a reduced gradient code for optimizing nonlinear econometric models. The theory of reduced gradient algorithms is presented, and special attention is given to 1) the representation of information in the computer, where sparse matrix techniques are recommended, 2) a recursion for computing the reduced gradient, and 3) the solution of the econometric model for prescribed controls where a pseudo-Newton method based on matrices computed in the reduced gradient step is described. Efficient handling of linear model equations, computation of analytic derivatives, and computational results with a small Danish model are also mentioned.

1. INTRODUCTION

In the past decade the interest in using optimization methods on macroeconomic models has grown rapidly. The ability of this new approach has been demonstrated in a number of cases. Most of the employed models had some special structure, e.g. were linear models or small recursive nonlinear models. Methods for handling more general optimization problems have also been presented by e.g. Fair [7]. Due to rather high computational costs with current methods more efficient computational methods must be developed before optimization techniques may be applied in the economic planning.

The paper describes the main theoretical background of an optimization code for nonlinear econometric models developed and implemented at the Technical University of Denmark. The paper reviews some techniques from mathematical programming and shows how the techniques may be applied to econometric models.

Recently, many authors have suggested, that sparse matrix techniques may be used for the many matrix manipulations in an optimization, see e.g. J. R. Bird, [4], J. Mantell, [14], or A. L. Norman, M. R. Norman, & C. J. Palash, [17]. Following a short problem formulation in section 2, the next section describes the basic ideas of sparse matrix techniques, giving special attention to the topics relevant to nonlinear econometric models.

The application of reduced gradients has been suggested by several

*This research project has partly been supported by the Danish Social Science Research Council through grant no. 514-5201. A part of the research was performed while the author was at the Department of Mathematics, The Technical University of Denmark.

authors, e.g. J. R. Bird, [4], and J. Mantell, [14]. An algorithm for the computation of the reduced gradient for econometric models with many lags is presented in section 4. After the reduced gradient has been computed the control variables are changed and a simulation is performed yielding the new state variables. The simulation is performed using a pseudo-Newton method which applies an inverse matrix computed in the reduced gradient step. The repeated use of the same inverse matrix, originally suggested by J. Abadie & J. Carpentier, [1], is one of the major advantages of reduced gradient methods. The simulations are treated in section 5.

In section 6 the handling of additional constraints such as mixed constraints and simple lower and upper bounds is discussed.

The current code employs the analytical derivatives of both objective function and constraints. Some arguments in support of this approach are given in section 7. Finally, section 8 gives some computational experiences from the application of the code.

2. Model Formulation

In the sequel all vectors are column vectors. x' is the transpose of x. $\partial q_t/\partial x_{t-s}$ is the matrix of partial derivatives of the components of the vector function q_t with respect to the variables in the vector x_{t-s}. The matrix has one row per function in g_t and one column per variable in x_{t-s}.

In the following sections the model below is solved:

(1) $$\min \quad z = f(x, u)$$

subject to $g_t(x_t, u_t, x_{t-1}, u_{t-1}, \dots, x_{t-\tau}, u_{t-\tau}) = 0,$

(2) $$t = 1, \dots, T$$

and

(3) $$\alpha' \leq (x', u') \leq \beta'$$

where $x' = (x_1', x_2', \dots, x_T')$ is a vector of endogenous variables and $u' = (u_1', u_2', \dots, u_T')$ is a vector of control or policy variables. x and u are divided into T subvectors, x_t, u_t, one for each time interval within the planning horizon. Each subvector x_t is of length n, and each subvector u_t is of length m. g_t is a vector-valued function of length n representing the structural equations of the econometric model. g_t is assumed to incorporate exogenous variables. Furthermore, it is assumed, that g_t can be computed from the same set of formulas for all t. Occassionally the constraints (2) are summerized as $G(x, u) = 0$, where G has $n \cdot T$ components. g_t depends on variables from $\tau + 1$ time intervals, and $x_0, x_{-1}, \dots, x_{1-\tau}$ as well as $u_0, u_{-1}, \dots, u_{1-\tau}$ are assumed to be known.

$f(x,u)$ in (1) is a scalar-valued objective function where exogenous variables are built into the function f. This form of the objective function is preferred to the widely used separable form

$$z = \sum_{t=1}^{T} f_t(x_t, u_t),$$

since it is more general and allows for terms with arguments from several time intervals. We assume that the partial derivatives of the objective function, $\partial f/\partial x$ and $\partial f/\partial u$, and of the model equations, $\partial g_t/\partial x_{t-s}$ and $\partial g_t/\partial u_{t-s}$, $t = 1, \dots, T$; $s = 0, \dots, \min(S, t - 1)$, are known.

In the first sections the constraints (3) will be omitted corresponding to α = [a vector of $-\infty$] and β = [a vector of $+\infty$]. In section 6 it is discussed how these constraints can be taken into account.

3. Sparse Matrix Techniques

A sparse matrix is a matrix with few nonzero elements. Sparse matrix techniques are computational techniques taking advantages of the many zeros by storing only nonzero elements in the computer, and by only performing multiplications in which both factors are nonzeros. Sparse matrix techniques can save a large amount of core storage and a considerable amount of computing time, the time saving usually being the most important.

In large econometric models the matrices $\partial g_t/\partial x_{t-s}$ will usually have few nonzero elements. The purpose of this section is to describe the basic ideas of sparse matrix techniques and, especially, how these techniques may be applied in econometric modelling.

The following small matrix will be used for illustrative purposes:

$$\underline{A} = \begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 2 & & & 6 & \\ 2 & 4 & & 3 & & -2 \\ 3 & & -1 & & 1 & \\ 4 & & 1 & & & 2 \\ 5 & 1 & & -2 & & 1 \\ \hline \end{array}$$

The matrix may be stored as a sparse matrix in many different ways. In the following, however, the matrix will always be used column by column. Consequently, the following scheme is suggested: The nonzero elements are stored in a one-dimensional vector, A, and the corresponding row numbers are stored in a parallel vector R. A third parallel vector, L, contains the index of the next nonzero element in the column. A -1 in L indicates the end of the column. A fourth shorter vector, S, contains the index of the first element in each column.

For the above example the storage scheme might be:

Index	1	2	3	4	5	6	7	8	9	10	11	12
A	-1	3	6	1	-2	2	1	2	1	4	1	-2
R	3	2	1	3	2	4	5	1	5	2	4	5
L	11	12	4	-1	6	7	-1	10	-1	9	-1	-1
S	8	1	2	3	5							

Matrix-vector multiplications are simple. A more difficult, but also more important sparse matrix operation, is the solution of a set of linear equations, $\underline{\underline{A}}\, x = b$, with a sparse coefficient matrix. This operation is usually called an "inversion." From basic linear algebra it is known, that a set of linear equations may be solved through a sequence of row-operations gradually changing the coefficient matrix into the unit matrix, $\underline{\underline{I}}$. The same row-operations applied to the right hand side yields the solution to the set of equations.

If $\underline{\underline{A}}$ is a sparse matrix only few row-operations have to be performed since most of the matrix elements, which would otherwise have to be eliminated, are already zero. The idea of sparse matrix inversion is to store the few remaining row-operations. The stored row-operations are applied to b each time $\underline{\underline{A}}\, x = b$ is solved for a new vector b.

A row-operation may be: "add α times row k to row i." As a result of this row-operation row i of the coefficient matrix changes according to the formula $a_{ij} := a_{ij} + \alpha \cdot a_{kj}$, $j = 1, \ldots, n$. A very unpleasant thing happens when a_{ij} was zero and a_{kj} was nonzero, because suddenly an additional nonzero element appears. This so-called "fill-in" must be eliminated later on. The number of fill-ins depends on the ordering of the rows and columns of the matrix $\underline{\underline{A}}$. Many heuristic methods for ordering rows and columns minimizing the number of fill-ins have been suggested see e.g. H. M. Markowitz, [15], J. K. Reid, [18], or D. J. Rose & R. A. Willoughby, [19].

A valuable observation is, that when row k is added to other rows, fill-ins will only be created in those columns where row k has nonzero elements. If row k has one nonzero element, no fill-ins will be created. In a lower-triangular matrix the elements below the diagonal can be eliminated from the left, and in each step the row added to other rows will have precisely one element. A method recently suggested by E. Hellerman & D. C. Rarick, [10] and [11], utilize these observations. The rows and columns

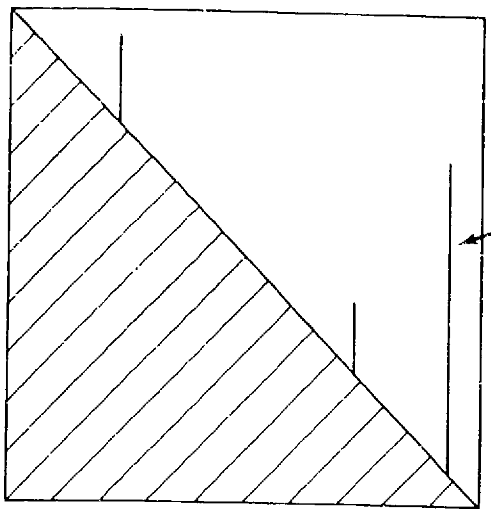

Figure 1 After rearranging rows and columns the matrix will be almost lower-triangular with only a few spike-columns.

are rearranged such that most of the elements are in the lower-triangular part of the matrix. If it is not possible to create a completely lower-triangular matrix, the elements above the diagonal are arranged to form as few columns as possible. The columns with nonzero elements in the upper-triangular part of the matrix are called spike-columns or spikes, see fig. 1. The fill-ins will occur in the spike columns only. Hellerman and Rarick's procedure applied to the exemplification matrix yields a matrix where only the original column 2 is a spike column:

$$\begin{array}{c|ccccc|}
 & 4 & 1 & 5 & 3 & 2 \\
\hline
3 & 1 & & & & -1 \\
1 & 6 & 2 & & & \\
4 & & & 2 & & 1 \\
2 & & 4 & -2 & 3 & \\
5 & & 1 & 1 & -2 & \\
\hline
\end{array}$$

If the set of equations is regarded as representing a flow of information from of-diagonal to diagonal elements as in D. V. Steward, [20], the lower-triangular matrix-elements correspond to a feed-forward of information while the upper-triangular elements correspond to a feed-back. The idea is to concentrate the feed-back into a small number of variables, the spike variables. If the values of the spike-variables are known, the rest of the matrix is lower-triangular and the set of equations may be solved recursively. This recursion has been used by P. Nepomiastchy [16], to obtain a solution to a set of nonlinear equations.

The set of row-operations used to eliminate nonzeros above and below the diagonal in a given column can be represented by the following information: The pivot element and its row number, and for the remaining nonzeros in the column, the values of the elinated elements and their row numbers. This information, however, is exactly the content of the column before it is affected by the elimination. Hence, the set of all row-operations in the inversion can be represented by the content of the columns just before they are eliminated. Since only the spike-columns are changed before they are eliminated, the information requiring storage is limited to:

1) The original matrix $\underline{\underline{A}}$.
2) The spike-columns just before they are eliminated (called updated spike-columns) added as extra columns to $\underline{\underline{A}}$ using the standard matrix storage scheme.
3) A list of the order in which columns, including updated spike-columns, are eliminated, and the row numbers for corresponding pivot elements.

The storage scheme described above was originally suggested by J. E. Kalan, [12], for use in large scale linear programming. Sharing core storage between the inverse and the matrix itself, as it is done in the present storage scheme, is very space saving. Furthermore, the scheme holds a very important advantage over most other methods for representing the inverse. The matrix $\underline{\underline{A}}$ will usually be the Jacobian $\partial q_t/\partial x_t$. Assuming the model to be nonlinear, changes in x_t during the iterations will produce changes in the Jacobian. However, it is easy to find a representation of the new $(\partial q_t/\partial x_t)^{-1}$. When the new $\partial q_t/\partial x_t$ is computed, the information in 1) above is automatically updated. And the pivot pattern in 3) can be used unchanged. In 2) the nonzero pattern of the updated spike-columns will remain unchanged, only the numerical values of the elements of the updated spikes must be recomputed. Since most sparse matrices will have few spike columns, this reinversion is very fast.

Although only the operations for $\underline{\underline{A}}^{-1} \cdot b$ have been described, the information concerning row-operations may also be applied to compute $c' \cdot \underline{\underline{A}}^{-1}$ see e.g. A. Drud, [6].

3.1 *Linear Models and Sparse Matrices*

A linear model may be written in structural form:

$$\sum_{s=0}^{\vdots} \{\underline{\underline{A}}_{t,t-s} \cdot x_{t-s} + \underline{\underline{B}}_{t,t-s} \cdot u_{t-s}\} = b_t \tag{4}$$

$\Updownarrow$

(5) $$\underline{\underline{A}}_{t,t}\cdot x_t = b_t - \underline{\underline{B}}_{t,t}\cdot u_t - \sum_{s=1}^{\ell}\{\underline{\underline{A}}_{t,t-s}\cdot x_{t-s} + \underline{\underline{B}}_{t,t-s}\cdot u_{t-s}\}$$

where the nonzero pattern of the matrices represent the structure of the economy, or in reduced form:

(6) $$x_t = b_t^1 - \underline{\underline{B}}^1_{t,t}u_t - \sum_{s=1}^{\ell}\{\underline{\underline{A}}^1_{t,t-s}\cdot x_{t-s} + \underline{\underline{B}}^1_{t,t-s}u_{t-s}\}$$

where

$$b_t^1 = \underline{\underline{A}}^{-1}_{t,t}\cdot b_t, \underline{\underline{B}}^1_{t,t-s} = \underline{\underline{A}}^{-1}_{t,t}\cdot \underline{\underline{B}}_{t,t-s}, s = 0,\ldots,\ell,$$

and

$$\underline{\underline{A}}^1_{t,t-s} = \underline{\underline{A}}^{-1}_{t,t}\cdot \underline{\underline{A}}_{t,t-s}, s = 1,\ldots,\ell$$

have been introduced.

For most purposes the reduced form is used because x_t can be computed recursively. $\underline{\underline{B}}^1_{t,t}$ and $\underline{\underline{A}}^1_{t,t-1}$ are usually completely dense, and thus the number of multiplications needed to compute x_t is at least $n\cdot(n + m)$.

However, it is often faster to use the structural form (5). $\underline{\underline{A}}^{-1}_{t,t}$ is computed once. $\underline{\underline{A}}_{t,t}$ rarely has more than 3–4 $\cdot\, n$ elements, and $\underline{\underline{A}}^{-1}_{t,t}$ can be represented by 5–10 $\cdot\, n$ row operations. The right-handside of (5) may usually be computed in less than $5\cdot n$ multiplications. Even with the additional time for data-administration, the computing time with the structural form is much smaller than the time for the approximately $n\cdot(n + m)$ multiplications in the reduced form, whenever the model dimensions are medium or large.

The example above clearly demonstrates the computational power of sparse matrix techniques. Similarily, other computations in linear models may be performed much faster using sparse matrices and the structural form of the model instead of the reduced form.

4. The Reduced Gradient

The concept of a reduced gradient was originally introduced by P. Wolfe, [21], for linear models and later generalized to nonlinear models by J. Abadie & J. Carpentier, [1]. J. Abadie, [2], describes a method for computing the reduced gradient for dynamic, recursive models, and the method is applied to recursive economic models by J. Abadie & M. Bichara, [3]. Recently J. R. Bird, [4], has applied the reduced gradient to simultaneous econometric models. The following is a slight generalization of Bird's approach.

The model is (cf. section 2)

(7) $$\min z = f(x, u)$$

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

G represents an econometric model and thus it is possible for each choice of control variables, u, to compute the endogenous variables, x, i.e. $G(x,u) = 0 <=> x = x(u)$. Introducing this implicitly defined function $x(u)$ in (7) yields $z = f(x(u),u) = F(u)$, i.e. the objective function depends on the control variables u alone. The reduced gradient is defined as the vector $\partial F/\partial u$, i.e. the reduced gradient is equivalent to the gradient that R. C. Fair, [7], computed using a finite difference method. The chain rule provides a formula for the reduced gradient:

(9) $$\frac{\partial F'}{\partial u} = \frac{\partial f'}{\partial u} + \frac{\partial f'}{\partial x} \cdot \frac{\partial x}{\partial u}$$

and a formula for the matrix $\partial x/\partial u$ may be derived from the implicit function theorem. The intuitive idea is:

(10) $$\begin{aligned} &G(x(u),u) = 0 \quad \text{for all } u \\ &\Downarrow \\ &\frac{\partial G}{\partial u} + \frac{\partial G}{\partial x} \cdot \frac{\partial x}{\partial u} = 0 \\ &\Downarrow \\ &\frac{\partial x}{\partial u} = -\left(\frac{\partial G}{\partial x}\right)^{-1} \cdot \frac{\partial G}{\partial u} \end{aligned}$$

Combining (9) and (10), the reduced gradient becomes

(11) $$\begin{aligned} \frac{\partial F'}{\partial u} &= \frac{\partial f'}{\partial u} - \left\{ \left[\frac{\partial f'}{\partial x} \cdot \left(\frac{\partial G}{\partial x}\right)^{-1} \right] \cdot \frac{\partial G}{\partial u} \right\} \\ &= \frac{\partial f'}{\partial u} - \Pi' \cdot \frac{\partial G}{\partial u} \end{aligned}$$

where the content of the large brackets, the vector Π, is computed first and then multiplied by the second matrix. This is cheaper than computing the matrix-matrix product in (10) directly. The computation of Π involves solving the set of linear equations:

$$(\Pi'_1, \Pi'_2, \ldots, \Pi'_T) \cdot \left[\begin{array}{|c|c|c|c|} \hline \frac{\partial g_1}{\partial x_1} & & & \\ \hline \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & & \\ \hline \cdot & \cdot & \cdot & \\ \hline \cdot & \cdot & \frac{\partial g_T}{\partial x_{T-1}} & \frac{\partial g_T}{\partial x_T} \\ \hline \end{array}\right] = \left(\frac{\partial f'}{\partial x_1}, \frac{\partial f'}{\partial x_2}, \ldots, \frac{\partial f'}{\partial x_T}\right).$$

where Π, $\partial f/\partial x$, and $\partial G/\partial x$ have been decomposed into time-components. Since $\partial G/\partial x$ is lower-block-triangular, the solution may be found by a recursion, starting at $t = T$:

$$(12) \qquad \Pi_t' = \left(\frac{\partial f'}{\partial x_t} - \sum_{s=1}^{\min(\ell, T-t)} \Pi_{t+s}' \cdot \frac{\partial g_{t+s}}{\partial x_t}\right) \cdot \left(\frac{\partial g_t}{\partial x_t}\right)^{-1}, t = T, \ldots, 1$$

Note, that only the blocks on the diagonal are inverted. If the model is valid, the blocks are always regular. For large models $\partial g_t/\partial x_t$ will be sparse, and the inverse diagonal blocks should be represented by some sparse form as described in section 3. Further note, that apart from the inversions, only vector-matrix products are computed, a fairly inexpensive operation when the matrices $\partial g_{t+s}/\partial x_t$ are stored as sparse matrices.

After the multipliers, Π, (which in the optimal solution are equal to the Lagrange multipliers) have been computed, the reduced gradient may be computed by

$$(13) \qquad \frac{\partial F'}{\partial u_t} = \frac{\partial f'}{\partial u_t} - \sum_{s=0}^{\min(\ell, T-t)} \Pi_{t+s}' \cdot \frac{\partial g_{t+s}}{\partial u_t}, t = T, \ldots, 1$$

where again all operations are inexpensive vector-matrix multiplications.

Many lags in the model or a large number of lagged variables does not affect the size of the inverted matrices. It is not necessary to introduce additional model variables and model equations to reduce the number of lags in the model as it is done in G. C. Chow, [5], and in J. R. Bird, [4]. Neither is it necessary that the objective function is separable like $\sum_{t=1}^{T} f_t(x_t, u_t)$ as long as the vectors $\partial f/\partial x_t$ and $\partial f/\partial u_t$ may be computed.

5. Simulations

The reduced gradient, $\partial F/\partial u$, may now be used in combination with any gradient, conjugate gradient, or variable metric method to minimize $F(u)$. All the methods mentioned include a one-dimensional search in which u is changed into $u + \gamma \cdot \Delta u$ and $F(u + \gamma \cdot \Delta u)$ is computed for various γ-values. However, the function $F(u)$ is not known explicitly. Consequently, the model must be solved with respect to $x(u)$, and the explicitly known objective function $f(x, u)$ may then be applied.

There are two standard methods for numerically solving a nonlinear simultaneous econometric model, Gauss-Seidel's method and Newton's method, see e.g. G. From & L. R. Klein, [8]. Until recently, Newton's method has not been widely used with econometric models due to the matrix inversions. However, when the matrices involved are sparse matrices, and when the reduced gradient is computed as shown in (12)-(13), a relatively efficient pseudo-Newton method may be devised:

Consider time interval t. It is assumed that x_s has already been computed for $s = 1, \ldots, t-1$, and the problem is to find x_t such that g_t. $(x_t, u_t, \ldots, x_{t-\tau}, y_{t-\tau}) = h(x_t) = 0$, where h is used to indicate that at the moment only x_t is unknown.

Newton's method is based on the formula

$$h(x_t + \Delta x_t) = h(x_t) + \frac{\partial h(x_t + \underline{\underline{\theta}} \cdot \Delta x_t)}{\partial x_t} \cdot \Delta x_t$$

where $\underline{\underline{\theta}} = \operatorname{diag}(\theta_1, \ldots, \theta_n)$ is a diagonal matrix with $0 \leq \theta_i \leq 1$. Setting the right-hand-side equal to 0 yields

$$\Delta x_t = -\left\{\frac{\partial h(x_t + \underline{\underline{\theta}} \cdot \Delta x_t)}{\partial x_t}\right\}^{-1} \cdot h(x_t). \tag{14}$$

Unfortunately, $x_t + \underline{\underline{\theta}} \cdot \Delta x_t$ is not known. Otherwise (14) could have been used to compute the correct $x_t + \Delta x_t$ in one iteration. In the standard Newton method, the inverse Jacobian

$$\left\{\frac{\partial h(x_t + \underline{\underline{\theta}} \cdot \Delta x_t)}{\partial x_t}\right\}^{-1}$$

is approximated by the inverse Jacobian computed at the current point x_t,

$$\left\{\frac{\partial h(x_t)}{\partial x_t}\right\}^{-1},$$

and as x_t converges to the solution,

$$\left\{\frac{\partial h(x_t)}{\partial x_t}\right\}^{-1}$$

converges to

$$\left\{\frac{\partial h(x_t + \underline{\underline{\theta}} \cdot \Delta x_t)}{\partial x_t}\right\}^{-1}.$$

In an optimization several x_t-vectors are available: Let x_t^0 represent the point from which the one-dimensional search is started and where the inverse Jacobian

$$\left\{\frac{\partial h(x_t^0)}{\partial x_t}\right\}^{-1} = \left\{\frac{\partial g_t}{\partial x_t}\right\}^{-1}$$

is computed for use in (12), and let x_t represent the current trial-solution to $h(x_t) = 0$ on which improvement is desired. The pseudo-Newton method applied in connection with reduced gradient algorithms applies to known

$$\frac{\partial h(x_t^0)}{\partial x_t}^{-1}$$

instead of

$$\left\{\frac{\partial h(x_t)}{\partial x_t}\right\}^{-1}$$

as in the standard Newton method saving both the computation of the Jacobian and the inversion. The pseudo-Newton method converges as long as

$$\left\{\frac{\partial h(x_t^0)}{\partial x}\right\}^{-1}$$

is sufficiently close to

$$\left\{\frac{\partial h(x_t + \underline{\underline{\theta}} \cdot \Delta x_t)}{\partial x_t}\right\}^{-1},$$

and the limiting rate of convergence is

$$\rho\left\{\underline{\underline{I}} - \frac{\partial h(x_t)}{\partial x_t} \cdot \left(\frac{\partial h(x_t^0)}{\partial x_t}\right)^{-1}\right\},$$

where x_t is the true solution and ρ indicates the spectral radius, i.e. the absolute value of the numerically largest eigenvalue of the matrix. Note. that the rate of convergence is linear and not quadratic as for the standard Newton method. Also, the rate of convergence depends on the steplength γ in the one-dimensional search. For a small steplength x_t^0 will be close to x_t implying a good rate of convergence.

A first trial-solution x_t for the pseudo-Newton method may be obtained in various ways, e.g. by a first order approximation $x = x^0 + \gamma \cdot \Delta x$:

$$G(x^0 + \gamma \cdot \Delta x, u^0 + \gamma \cdot \Delta u) \sim \gamma \cdot \left\{\frac{\partial G}{\partial x} \cdot \Delta x + \frac{\partial G}{\partial u} \cdot \Delta u\right\} = 0$$

$$\Downarrow$$

$$\Delta x = -\left(\frac{\partial G}{\partial x}\right)^{-1} \cdot \left(\frac{\partial G}{\partial u} \cdot \Delta u\right)$$

which may be solved recursively by:

$$\Delta x_t = -\left(\frac{\partial g_t}{\partial x_t}\right)^{-1}\left\{\frac{\partial g_t}{\partial u_t} \cdot \Delta u_t + \sum_{s=1}^{\min(\ell, t-1)} \left(\frac{\partial g_t}{\partial x_{t-s}} \cdot \Delta x_{t-s} + \frac{\partial g_t}{\partial u_{t-s}} \cdot \Delta u_{t-s}\right)\right\},$$

$$t = 1, \ldots, T.$$

where only the inverse matrices computed earlier are used.

During a one-dimensional search x is an implicitly defined function of the steplength γ, $x = x(\gamma)$, with $\partial x/\partial_{\gamma|\gamma=0} = \Delta x$. The trial-solution mentioned above is based on a first order approximation to the implicit function. L. S. Lasdon et al., [13], suggest that a second order approximation be used after the first simulation. The quadratic approximation may e.g. be fitted to the points $x(0)$ and $x(\gamma_0)$ with derivative Δx in $x(0)$, where γ_0 is the steplength of the first simulation.

In the current code, CONOPT, none of the two above methods for computing a first trial-solution have yet been implemented. x is initialized as the x corresponding to the best objective function found so far. Even with this unsophisticated start, the simulations usually converges with approx. 3 evaluations of the residuals g_t and 2 multiplications by the inverse $(\partial g_t/\partial x_t)^{-1}$ per time interval.

5.1. *Linear Equations*

A linear function is always equal to its first order approximation evaluated at any point. Thus, during Newton or pseudo-Newton iterations, where a solution satisfying a first order approximation to the set of equations is computed, the linear equations are solved in one iteration. When the initial trial-solution is computed using either the linear or the quadratic approximation mentioned above, the linear equations will even be satisfied by the trial-solution.

When an equation is known to be satisfied, there is no reason to compute its residual. Hence, the subroutine that computes residuals should contain an input argument, which controls whether all residuals or only residuals of nonlinear equations should be computed. Most large econometric models contain a large percentage of linear equations, and when they only have to be considered in a few calls of the constraint subroutine, the saving will be considerable. Furthermore, during the inversion, i.e. during the row-operations, more savings will be obtained, since all row-operations with a zero in the right-hand-side of the pivot-row are not performed.

5.2. *Stop Criteria for Simulations*

The question to be answered in this section is: "How accurately should the simulations be performed?" The purpose of a simulations is, at least during the optimization, to find an x-vector which can be used in the objective function subroutine for evaluating $f(x, u)$. Hence the accuracy of the simulation should be determined such that the accuracy of $f(x, u)$ is sufficient.

The error e in the objective function due to an error Δx in x may be estimated as follows:

(15)

$$e = f(x + \Delta x, u) - f(x,u) \sim \frac{\partial f'}{\partial x} \cdot \Delta x \sim -\frac{\partial f'}{\partial x} \cdot \left(\frac{\partial G}{\partial x}\right)^{-1} \cdot G = -H' \cdot G$$

where (14) and (11) have been used. Hence, the error in the objective function is approximately equal to the scalar product of the multiplier vector with the residual vector.

Consequently, the following stop-criteria are used: Compute z, the sum of squared residuals divided by the number of constraints. If z is less than epsmin (e.g. 10^{-28}), then stop — it is not possible to find a more accurate solution due to round-off errors. If z is less than epsmax (e.g. 10^{-6}), and if the estimated error $|e_i| = |H_i' \cdot h_i|$ is less than f_{eps}/I, then stop. f_{eps} may e.g. be between 10^{-3} to 10^{-4} times the change in the objective function in the previous one-dimensional search. The last stop criterion is: If a solution satisfying one of the first two stop criteria has not been found within a certain number of iterations (e.g. 10), the pseudo-Newton method will probably not converge. Consequently, it is stopped and a shorter steplength is used in the one-dimensional search.

The stop criteria suggested here have the important advantage, that the solutions become gradually more accurate as we approach the optimum and the changes in the objective function are smaller, while at the same time computer-time is not wasted on accurate solutions when u is far from the optimum.

6. Additional Constraints

This section contains some ideas on how to treat simple bounds on the variables. A more detailed description is given by A. Drud, [6].

The problem formulation of section 2 included the simple bounds:

$$\alpha' \leq (x', u') \leq \beta'$$

When only the u-vector has finite simple bounds the basic procedure is as described earlier. The only difference is that $F(u)$ is minimized subject to $\alpha_u \leq u \leq \beta_u$. However, the simple lower and upper bounds are easily taken into account by using e.g. D. Goldfarb's, [9], approach.

When simple bounds are active on x the following observation is useful: An optimal solution is a set of variables (x, u) that satisfies the constraints and minimizes the objective function — whether some variables are control variables and others are endogenous variables is without interest, at least from the optimization point of view. This leads to the following proposal: Since lower and upper bounds on the control variables can easily be dealt with, use the variables that are close to a bound as pseudo-control variables, u_p, and use the variables far from the bounds as pseudo-endogenous variables. x_p. x_p should also be chosen such that $g_t(x_{p,t}, u_{p,t}, \ldots) = 0$ can be solved with respect to $x_{p,t}, t = 1, \ldots, T$, or such that $x_{p,t}$ has n components and $\partial g_t/\partial x_{p,t}$ is regular. In the CONOPT-code the set of pseudo-endogenous variables $x_{p,t}$ is chosen from the set of candidates (i.e. variables far from bound) such that the matrix $\partial g_t/\partial x_{p,t}$ is easy to invert, i.e. such that $\partial g_t/\partial x_{p,t}$ has few nonzero elements above the diagonal.

It should be noted, that mixed inequality constraints like

$$e(x_t, u_t, x_{t-1} \ldots) \leq 0$$

may be handled as well: Add a slack variable with lower bound zero and transform the inequality into an equality which is added to the model with the slack variable as the corresponding endogenous variable.

The optimization procedure with bounded variables roughly is:

a) Divide (x, u) into x_p and u_p and compute the inverse matrices $(\partial g_t/\partial x_{t,t})^{-1}$.

b) Compute the reduced gradient $\partial F(u_p)/\partial u_p$ using equations similar to (12)-(13) and use the gradient in the minimization of $F(u_p)$ subject to $\alpha_{up} \leq u_p \leq \beta_{up}$.

$F(u_p)$ is computed indirectly: The model is solved with respect to x_p using the pseudo-Newton method with the inverse matrices $(g_t/\partial x_{p,t})^{-1}$, and $F(u_p) = f(x_p, u_p)$ is computed by the usual objective function subroutine.

During the optimization it may happen that some pseudo-endogenous variables come close to their bounds. In this case redefine x_p and u_p such that the variables near bounds are in u_p. In most practical optimizations such redefinitions of x_p and u_p are rare.

The main differences between the procedure outlined above and the procedure used when there are no constraints on x, are, that in the new procedure it is not known at the beginning which $(n \times n)$ – submatrix of $(\partial g_t/\partial x_t, \partial g_t/\partial u_t)$ is going to be inverted, neither is it known with respect to which variables the model must be solved. With the inversion procedure and storage scheme mentioned in section 3 the inversion is not a difficult problem. Likewise, the pseudo-Newton method described in section 5 does not depend on which set of variables is used as pseudo-endogenous variables. (It would have been very difficult to implement the procedure above using the Gauss-Seidel or related methods for the simulations because these methods depend heavily on a prescribed set of output-variables.)

When the number of active simple bounds in a single time interval is large (i.e. $> m$) e.g. when the model has many terminal conditions it is not possible to choose a set of pseudo-endogenous variables satisfying the necessary requirements. A reduced gradient method may still be applied. However, $\partial G/\partial x_p$ will no longer be lower-block-triangular and the computation of the reduced gradient and the simulation can not be decomposed into time-components as described in section 4 and 5. The best approach seems to be to take care of the additional bounds through a penalty term in the objective function.

7. Input Requirements

The CONOPT-code needs the following input:

1) A subroutine computing the value of the objective function
2) A subroutine computing the gradient of the objective function
3) A subroutine computing the residuals of the constraints
4) A subroutine computing the Jacobian of the constraints
5) The nonzero pattern of the Jacobian
6) Lower bounds, upper bounds, initial solution etc.

Often the objective function and the econometric model are written as a set of expressions/equations with variable-names like C, I, GNP etc. Translating this information into the 4 subroutines mentioned above can be very time consuming. Therefore, a special computer code, TRANSL, has been produced based on the IBM-product FORMAC (*FOR*mula *MA*nipulation Compiler). Input to TRANSL is a set of variable-names defined as endogenous, control, or exogenous variables, a data bank, and a set of equations written as equations of an ordinary econometric model. All variable-names are translated into $X(1), X(2), \ldots, XE(1)$, $XE(2), \ldots$ etc, and replaced in the equations and in the objective function, and equations and objective function are punched as FORTRAN statements for the residual and the objective function subroutines. Simultaneously, all partial derivatives are computed analytically, and it is tested whether they are zero, constant, or variable. The nonzero pattern of the Jacobian as well as the values of the constant Jacobi elements are punched in a data bank, and the variable derivatives are punched as FORTRAN statements for the Jacobian and the gradient subroutines. TRANSL also has a device for scaling equations and variables in order to improve the numerical stability of CONOPT.

The TRANSL code is an experimental code and as such not very efficient or flexible, but it proves that analytic derivatives may be used even for large models. In a better implementation that TRANSL the analytic derivatives should be computed one equation at a time and stored with the equation in an equation-derivative bank with information on nonzero pattern, linearity, etc. Before the optimization the equations, the derivatives, and other data items should be collected into subroutines and data files. This approach will make it very easy and cheap to make changes in the model or in the objective function.

8. Computational Experience

The computer codes have been applied to a small model of the Danish economy with 28 equations, 33 endogenous and control variables, 47 exogenous variables, and 3 lags. In TRANSL a total of 2128 derivatives were computed, 1951 were zero, 120 were constant, and only 57 were variable. The time used by TRANSL was approx. 70 sec. CPU on an IBM 370/165 corresponding to a cost of approx. US $7.50 at the computing center at the Technical University of Denmark. Because only variable elements are computed by the Jacobi-subroutine, the subroutine required less core storage than the constraint subroutine and was also faster to execute. This should be compared to a method based on numerical derivatives, where the computation of the Jacobian would require at least 33 calls of the constraint subroutine.

The following figures relate to the CONOPT-code. They are rather detailed in order to give the reader an impression of the distribution of the computing time on the various steps in the optimization. The model was optimized over a 5 period planning horizon corresponding to a problem with 25 control variables and 140 equations. With one objective function the total optimization time was 24 sec. CPU including some 2 sec. for input/output, initialization etc., leaving 22 sec. for the optimization. 58 iterations, consisting of a computation of the Jacobian, the inverse Jacobian, the reduced gradient, and a one-dimensional search, were performed, i.e. 0.380 sec. per iteration. During these 58 iterations the length of the reduced gradient was changed from 11.4 to $3.3 \cdot 10^{-6}$, a factor of approx. 10^{-7}. Computing the Jacobian and the inverse Jacobian for five periods consumed 0.160 sec., that is 0.032 sec. per time interval, leaving 0.220 sec. per iteration for the one-dimensional search. An average of 4.4 different steplengths were employed by the one-dimensional search which gives some 0.050 sec. per 5-period simulation. With other objective functions the number of iterations has changed a little, but the time per iteration is almost constant.

Some of the techniques described in this paper have not yet been implemented in CONOPT and some savings are expected:

1) All row-operations of the inverse Jacobian are now stored separately and not embedded in the Jacobian itself as described in section 3. After a change the core storage requirement will be reduced a little and the reinversion time is expected to be cut by at least a factor 0.5.
2) A linear or quadratic estimate for the initial x in the pseudo-Newton iterations is expected to decrease the simulation time by a factor 0.5.
3) Dividing the equations in linear and nonlinear equations will probably decrease the simulation time further by a factor 0.6.

9. Conclusions

An optimization code for nonlinear econometric models has been presented. The code is based on sparse matrix techniques and will therefore be able to solve models with 200–300 equations per time interval within a reasonable core-storage. No computational experiments with a really large model has yet been performed. However, the method for computing the reduced gradient is expected to be much faster and more accurate than currently used finite difference methods, and the simulation procedure is also more accurate and probably also faster than current methods.

Institute of Mathematical Statistics and Operations Research
Technical University of Denmark
DK-2800 Lyngby-Denmark

References

[1] J. Abadie & J. Carpentier: "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," in "Optimization," R. Fletcher (ed.), Academic Press, 1969, pp. 37–47.

[2] J. Abadie: "Application of the GRG-Algorithm to Optimal Control Problems," in "Integer and Nonlinear Programming," J. Abadie (ed.), North-Holland, 1970, pp. 191–211.

[3] J. Abadie & M. Bichara: "Resolution Numerique de Certain Problemes de Commande Optimale," in Revue Franciase d'Automatique, Informatique et Recherche Operationelle, 1973, p. 77–105.

[4] J. R. Bird: "An Efficient Method for Computing the Reduced Gradient of an Econometric Model," Working Paper 76-13, School of Business, Queen's University, Kingston, Canada, 1976.

[5] G. C. Chow: "Analysis and Control of Dynamic Economic Systems," John Wiley & Sons, 1975.

[6] A. Drud: "Methods for Control of Complex Dynamic Systems—Illustrated by Econometric Models," Ph.D.-thesis, The Institute of Mathematical Statistics and Operations Research, The Technical University of Denmark, Lyngby, Denmark, 1976.

[7] R. C. Fair: "Methods for Computing Optimal Control Solutions, On the Solution of Optimal Control Problems as Maximization Problems," in Annals of Economic and Social Measurements, vol. 3, 1974, pp. 135–153.

[8] G. From & L. R. Klein: "Solution of the Complete System," in "The Brookings Model: Some Further Results," J. Duesenberry, G. From, L. R. Klein & E. Kuh (eds.), 1969, p. 363–382.

[9] D. Goldfarb: "Extension of Davidon's Variable Metric Method to Maximization under Linear Inequality and Equality Constraints," in SIAM Journal on Applied Mathematics, vol. 17, 1969, pp. 739–764.

[10] E. Hellerman & D. C. Rarick: "Reinversion with the Preas-signed Pivot Procedure," in Mathematical Programming, vol. 1, 1971, 195–216.

[11] E. Hellerman & D. C. Rarick: "The Partitioned Preassigned Pivot Procedure (P^4)," in [19], pp. 67–76.

[12] J. E. Kalan: "Aspects of Large-Scale In-Core Linear Programming," in Proceeding of the 1971 Annual Conference of the Association for Computing Machinery, August 3–5, 1971, Chicago, Illinois, pp. 304–313.

[13] L. S. Lasdon, A. D. Warren, A. Jain, M. W. Ratner: "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Optimization," Technical Mermoran-

dum No. 353, Department of Operations Research, Case Western Reserve University, 1975.

[14] J. Mantell: "Optimal Control of Large Nonlinear Planning Models," Ph.D. Thesis Proposal, Department of Operations Research, Case Western Reserve University, 1976.

[15] H. M. Markowitz: "The Elimination Form of the Inverse and its Application to Linear Programming," in Management Science, vol. 3, 1957, pp. 255–269.

[16] P. Nepomiastchy: "Methode d'Optimisation Adaptees aux Modeles Macroeconomiques," IRIA, Rocquencourt, France, 1976.

[17] A. L. Norman, M. R. Norman, & C. J. Palash: "On the Computation of Deterministic Optimal Macroeconomic Policy," Research Paper No. 7507, Federal Reserve Bank of New York.

[18] J. K. Reid (ed.): "Large Sparse Sets of Linear Equations," Academic Press, 1971.

[19] D. J. Rose & R. A. Willoughby (eds.): "Sparse Matrices and their Applications," Plenum Press, 1972.

[20] D. V. Steward: "On an Approach to Techniques for the Analysis of the Structure of Large Systems of Equations," in SIAM Review, vol. 4, 1962, pp. 321–342.

[21] P. Wolfe: "Methods of Nonlinear Programming," "Recent Advances in Mathematical Programming," R. L. Graves & P. Wolfe (eds.), McGraw-Hill, 1963, pp. 67–86.