JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS 92, 552–563 (1983)

A Finite Algorithm for the Exact Evaluation of Higher Order Partial Derivatives of Functions of Many Variables

R. KALABA*, L. TESFATSION*, AND J.-L. WANG

Department of Economics, University of Southern California, Los Angeles, California 90007

Submitted by W. Welsh

A formal algorithm is given for the systematic exact evaluation of higher order partial derivatives of functions of many variables. The algorithm improves upon Wengert's method in two key respects. Applications are envisioned wherever gradients, Jacobians, Hessians, and power series expansions could be employed.

1. INTRODUCTION

In recent work [3, 4] on the development of a complete differential system for the local and nonlocal sensitivity analysis of parameterized systems, it quickly became apparent that recourse had to be made to an automatic method of evaluating needed partial derivatives if a practical numerical tool was to be obtained. For example, the local parameter sensitivity analysis of an n-equation system of n variables requires the evaluation of $n(n+1)$ first-order partial derivatives.

Use was first made of Wengert's method [1, 6] for sequentially evaluating higher order partial derivatives. Wengert's key idea was to decompose the evaluation of complicated functions of many variables into a sequence of simpler evaluations of special functions of one and two variables. Total differentials of the special functions could be automatically evaluated along with the special function values. Partial derivatives could then be recovered from the total differentials by solving certain associated sets of linear algebraic equations.

Although programs were successfully written for implementing Wengert's method for first- and second-order partial derivatives, two principal difficulties arose in attempting to extend Wengert's method to higher-order

* The work of R. Kalaba and L. Tesfatsion was partially supported by the National Science Foundation under Grant ENG 77-28432 and the National Institutes of Health under Grant GM 23732-03.

0022-247X/83 $3.00

partial derivatives. First, Wengert's method requires the repeated evaluation of certain identical functional forms as each individual partial derivative is separately recovered from a total differential, resulting in significant computational inefficiency. Second, Wengert's method requires the formation and solution of a distinct set of linear algebraic equations for each successively higher order partial differentiation, and it does not seem possible to provide a systematic rule for how this is to be done.

The present paper develops a general finite "table algorithm" for the systematic exact evaluation of higher order partial derivatives that overcomes both of these difficulties. Wengert's key idea of sequential function evaluation is retained, but total differentials and linear algebraic equations play no role. An additional advantage of the table algorithm is that memory and arithmetic requirements can be determined prior to any calculations.

The table algorithm is so conceptually and computationally straightforward, it is difficult to believe that previous researchers have not already discovered it. Yet, current computer method sources (e.g., [2, pp. 170–171; 5, Sect. 14]) still discuss the desirability of avoiding derivative evaluations, and no presentation of such an algorithm has been found in the journal literature. Even Wengert's method is rarely cited.

The table algorithm is illustrated in Section 2 by means of a simple example. The general table algorithm is presented in Section 3. Calculus subroutines needed for certain special functions are given in Section 4, and Section 5 contains concluding comments.

2. An Illustrative Example

As will be clarified in Section 3, the table algorithm can in principle be used to evaluate the rth order partial derivatives of a broad class of n-variable functions $z = f(x_1, ..., x_n)$, for arbitrary integers r and n. For expositional simplicity, however, attention will here be focused on the evaluation of the first few partial derivatives of a certain two-variable function.

Thus, consider the function $F: R_{++} \times R \to R$ defined by

$$z = F(x, y) = (1 + \exp(xy))/\log(x). \tag{1}$$

Suppose evaluations are needed for the function value z and the partial derivatives z_x, z_y, z_{xx} at some particular point (x, y) in $R_{++} \times R$.

As in Wengert [6], we first note that z can be sequentially evaluated using certain special functions of one and two variables, as follows:

$$u = x, \tag{2a}$$

$$v = y, \tag{2b}$$

$$w = uv, \tag{2c}$$

$$q = \exp(w), \tag{2d}$$

$$r = 1, \tag{2e}$$

$$s = q + r, \tag{2f}$$

$$t = \log(u), \tag{2g}$$

$$z = s/t. \tag{2h}$$

All right-hand terms in (2) involve operations on x, on y, on a known constant, or on previously calculated variables. For any given x and y, the sequential evaluation of steps (2a)–(2h) yields the corresponding value z for the function $F(x, y)$ defined in (1).

Second, in a departure from Wengert [6], we show that (2) can be interpreted as the first column of an 8×4 table whose sequential evaluation, row by row, yields the required values (z, z_x, z_y, z_{xx}) in the final row.

Proceeding in stages, the first two rows of the table are obtained by formally differentiating each side of the equalities (2a) and (2b) with respect to x, with respect to y, and twice with respect to x, as indicated in (3):

Function	$\partial/\partial x$	$\partial/\partial y$	$\partial^2/\partial x^2$
$u = x$	$u_x = 1$	$u_y = 0$	$u_{xx} = 0$
$v = y$	$v_x = 0$	$v_y = 1$	$v_{xx} = 0$

(3)

The third row of the table is obtained by similarly differentiating each term in (2c), as indicated in (4):

Function	$\partial/\partial x$	$\partial/\partial y$	$\partial^2/\partial x^2$
$w = uv$	$w_x = u_x v + u v_x$	$w_y = u_y v + u v_y$	$w_{xx} = u_{xx} v + 2u_x v_x + u v_{xx}$

(4)

Note that row three transforms the two input arrays

$$(u, u_x, u_y, u_x), \qquad (v, v_x, v_y, v_{xx}), \tag{5}$$

previously calculated in rows one and two, into the output array

$$(w, w_x, w_y, w_{xx}). \tag{6}$$

The fourth row appears as

$$\begin{array}{lll} \text{Function} & \partial/\partial x & \partial/\partial y \qquad\qquad \partial^2/\partial x^2 \\ q = \exp(w) & q_x = q w_x & q_y = q w_y \qquad\quad q_{xx} = q_x w_x + q w_{xx} \end{array} . \tag{7}$$

The input array (w, w_x, w_y, w_{xx}), obtained from previous row calculations, is transformed into the output array (q, q_x, q_y, q_{xx}). It is important to note that the transcendental function $\exp(\cdot)$ is evaluated only once, in the first cell of (7). Only the simple algebraic operations *add* and *multiply* are used in remaining cells.

The evaluation of the remaining three rows proceeds analogously. Each successive row outputs a one-dimensional array of the form (p, p_x, p_y, p_{xx}), using the outputs of previous row calculations as inputs. In all columns except the first, only algebraic operations are used. The output of the final row yields the desired evaluations (z, z_x, z_y, z_{xx}).

The complete table is depicted in Table I. The sequential row-by-row evaluation of Table I constitutes the proposed table algorithm for calculating the value and first three partial derivatives (z, z_x, z_y, z_{xx}) of the function $z = F(x, y)$ defined in (1).

The table algorithm is exact if the evaluation of Table I is carried out exactly. In practice, of course, round-off and truncation errors will be introduced. Upper bounds for the memory and arithmetic requirements of this evaluation are easily determined. A maximum of thirty-two memory locations are needed for the 8×4 table evaluation. Estimating that a maximum of ten multiplications apiece are used for the (approximate)

TABLE I

Table Algorithm for the Sequential Evaluation of z, z_x, z_y, z_{xx}

Function	$\frac{\partial}{\partial x}$	$\frac{\partial}{\partial y}$	$\frac{\partial^2}{\partial x^2}$
$u = x$	$u_x = 1$	$u_y = 0$	$u_{xx} = 0$
$v = y$	$v_x = 0$	$v_y = 1$	$v_{xx} = 0$
$w = uv$	$w_x = u_x v + u v_x$	$w_y = u_y v + u v_y$	$w_{xx} = u_{xx} v + 2u_x v_x + u v_{xx}$
$q = \exp(w)$	$q_x = q w_x$	$q_y = q w_y$	$q_{xx} = q_x w_x + q w_{xx}$
$r = 1$	$r_x = 0$	$r_y = 0$	$r_{xx} = 0$
$s = q + r$	$s_x = q_x + r_x$	$s_y = q_y + r_y$	$s_{xx} = q_{xx} + r_{xx}$
$t = \log(u)$	$t_x = u_x/u$	$t_y = u_y/u$	$t_{xx} = \frac{u u_{xx} - u_x^2}{u^2}$
$z = s/t$	$z_x = \frac{[t s_x - s t_x]}{t^2}$	$z_y = \frac{[t s_y - s t_y]}{t^2}$	$z_{xx} = [t s_{xx} - s t_{xx}]\, t^{-2} - 2t^{-1} t_x z_x$

evaluation of $\exp(w)$ and $\log(u)$, the evaluation of Table I will require a maximum of fifty-nine multiplication–division operations, twenty-two for the first column and thirty-seven for the remaining three columns.

To implement the table algorithm for $F(x, y)$ on a computer, each row of Table I is replaced by a call statement for a calculus subroutine designed to generate the needed values for the row. Specifically, the table algorithm embodied in Table I can be implemented by the following dimension statement and sequence of calculus subroutines:

$$
\begin{aligned}
&\text{DIMENSION } U(4), V(4), W(4), Q(4),\\
&\qquad\qquad R(4), S(4), T(4), Z(4)\\
&\text{CALL } L1(U, X) \qquad \text{(row 1)}\\
&\text{CALL } L2(V, Y) \qquad \text{(row 2)}\\
&\text{CALL MUL}(W, U, V) \qquad \text{(row 3)}\\
&\text{CALL EXPP}(Q, W) \qquad \text{(row 4)}\\
&D = 1.0\\
&\text{CALL CON}(R, D) \qquad \text{(row 5)}\\
&\text{CALL ADD}(S, Q, R) \qquad \text{(row 6)}\\
&\text{CALL LOG}(T, U) \qquad \text{(row 7)}\\
&\text{CALL Div}(Z, S, T) \qquad \text{(row 8).}
\end{aligned} \tag{8}
$$

The calculus subroutine $L1$ used for row one takes the following general form:

$$
\begin{aligned}
&\text{SUBROUTINE } L1(B, A)\\
&\text{DIMENSION } B(4)\\
&B(1) = A\\
&B(2) = 1.0\\
&B(3) = 0.0\\
&B(4) = 0.0\\
&\text{RETURN}\\
&\text{END.}
\end{aligned} \tag{9}
$$

For each scalar input A, subroutine $L1$ outputs the one-dimensional array $B = (A, 1, 0, 0)$. Thus, subroutine $L1$ embodies the standard calculus formulas for obtaining the value and first three partial derivatives $B = (g, g_x, g_y, g_{xx})$ of the function $g: R^2 \to R$ defined by $g(x, y) = x$, given any particular value for $A = x$. The subroutine $L2$ used for row two is analogously specified for the function $h: R^2 \to R$ defined by $h(x, y) = y$.

The calculus subroutine used for row three takes the general form

```
SUBROUTINE MUL(C, B, A)
DIMENSION C(4), B(4), A(4)
C(1) = B(1) * A(1)
C(2) = B(2) * A(1) + B(1) * A(2)
C(3) = B(3) * A(1) + B(1) * A(3)
C(4) = B(4) * A(1) + 2.0 * B(2) * A(2)
     + B(1) * A(4)
RETURN
END.                                    (10)
```

Subroutine MUL embodies the standard calculus formulas for obtaining the value and first three partial derivatives $C = (h, h_x, h_y, h_{xx})$ of a function h defined as the product fg of two functions, $f: R^2 \to R$ and $g: R^2 \to R$, given any particular values for $B = (f, f_x, f_y, f_{xx})$ and $A = (g, g_x, g_y, g_{xx})$.

The calculus subroutine EXPP used for row four is

```
SUBROUTINE EXPP(B, A)
DIMENSION B(4), A(4)
R = A(1)
B(1) = EXP(R)
B(2) = B(1) * A(2)
B(3) = B(1) * A(3)
B(4) = B(2) * A(2) + B(1) * A(4)
RETURN
END.                                    (11)
```

Subroutine EXPP embodies the standard calculus formulas for obtaining the value and first three partial derivatives $B = (h, h_x, h_y, h_{xx})$ of a function h defined by $h = \exp(f)$ for some function $f: R^2 \to R$, given any particular value for $A = (f, f_x, f_y, f_{xx})$. Note that subroutine EXPP calls the standard library function subroutine EXP for the exponential function.

Similar interpretations can be given for the remaining calculus subroutines in (8).

3. The General Table Algorithm

In principle, a table algorithm can be constructed to calculate the value and partial derivatives through order r of any *Wengert* (r, n) *function*, defined as follows:

DEFINITION. A function $F: D \to R$, $D \subseteq R^n$, $n \geqslant 1$, will be referred to as a *Wengert* (r, n) *function* if, given any $x = (x_1,..., x_n)$ in D, the value $F(x)$ can be sequentialy calculated by means of the n initial conditions

$$s_1 = x_1,..., s_n = x_n, \tag{12}$$

the two-variable algebraic special functions $f: R^2 \to R$, given by

$$w = u + v, \qquad w = u - v, \qquad w = uv, \qquad w = u/v, \tag{13}$$

and arbitrary, one-variable, rth-order continuously differentiable special functions[1] of the form

$$h: M \to R, \qquad M \subseteq R. \tag{14}$$

Given any Wengert (r, n) function F with domain point x, the general table algorithm for $F(x)$ is constructed as follows:

Step 1. Form the *Wengert list* for $F(x)$, i.e., the list of initial conditions (12) and special functions (13) and (14) whose sequential evaluation yields $F(x)$. (See, for example, the Wengert list (2) in Section 2 for the function F defined by (1).)

Step 2. Replace the initial condition $s_1 = x_1$ in the Wengert list for $F(x)$ by the calculus subroutine $L1$, defined as follows with $K \equiv \sum_{j=0}^{r} n^j$:

```
      SUBROUTINE L1(S1, X1)
      DIMENSION S1(K)
      S1(1) = X1
      S1(2) = 1.0
      DO 10 I = 3, K
        S1(I) = 0.0
10    CONTINUE
      RETURN
      END.
```
(15)

For each scalar input $X1 = x_1$, subroutine (15) outputs a one-dimensional array containing the value and partial derivatives through order r of the function $g: R^n \to R$ defined by $g(x_1,..., x_n) = x_1$. Specify analogous subroutines $L2,..., LN$ for the functions $g(x_1,..., x_n) = x_i$, $i = 2,..., n$, to

[1] The one-variable special functions (14) may include, for example, the functions $\sin(u)$, $\cos(u)$, $J_m(u)$ (mth Bessel function), $\exp(u)$, $\log(u)$, and $au^b + c$ for arbitrary constants a, b, and c. Note that a Wengert (r, n) function is rth-order continuously differentiable by construction.

replace the remaining initial conditions $s_i = x_i$, $i = 2,..., n$, in the Wengert list for $F(x)$.

Step 3. Replace each special function (13) and (14) in the Wengert list for $F(x)$ by a calculus subroutine[2] for outputting the value of the special function, together with all of its partial derivatives with respect to x through order r. For any two-variable special function $w = f(u, v)$, the input to the calculus subroutine will be the previously calculated one-dimensional arrays $U = (u, u_{x_1}, u_{x_2},...)$ and $V = (v, v_{x_1}, v_{x_2},...)$, and the output will be a one-dimensional array $W = (w, w_{x_1}, w_{x_2},...)$. For any one-variable special function $z = h(y)$, the input to the calculus subroutine will be the previously calculated one-dimensional array $Y = (y, y_{x_1}, y_{x_2},...)$, and the output will be a one-dimensional array $A = (z, z_{x_1}, z_{x_2},...)$.

The list of calculus subroutines resulting from these three steps will be called the *table algorithm for F at x*. The proof that the table algorithm for F at x correctly calculates the value and all partial derivatives through order r of F at x follows by a straightforward complete induction argument.

4. Calculus Subroutines for Special Functions

Aside from initial conditions, the Wengert list for any Wengert (r, n) function F evaluated at a domain point x will consist of two-variable special function statements of the form

$$w = u + v, \qquad w = u - v, \qquad w = uv, \qquad w = u/v, \tag{16}$$

and one-variable special function statements of the form

$$z = h(u), \tag{17}$$

where $h: M \subseteq R$ is rth-order continuously differentiable. (See Section 3.) The table algorithm for F at x requires that each such statement be replaced by a calculus subroutine for generating the value and partial derivatives with respect to x through order r of the indicated special function. The present section will present systematic rules for how this can be done.

The add and subtract statements in (16) are easily and obviously handled, and the division statement in (16) can be handled as a multiplication statement once the rule for the multiplication statement $w = uv$ is known.

[2] Systematic rules for constructing the needed calculus subroutines for general special functions (13) and (14) will be presented in Section 4.

For any variable b, define

$$b_i = \frac{\partial b}{\partial x_i}, \qquad b_{ij} = \frac{\partial^2 b}{\partial x_i \partial x_j}, \ldots, \tag{18a}$$

$$b_A = b_{i \cdots l} \qquad \text{for} \quad A = \{i, \ldots, l\}. \tag{18b}$$

Then, assuming u and v are mth-order continuously differentiable functions of x, $m \geqslant 1$, the general rule for obtaining the $ijk \cdots l$ partial derivative of $w = uv$ for any m indices $i, j, k, \ldots, l$ can be expressed as follows:

$$w_{ijk \cdots l} = \sum_{(A,B) \in P_m^2} [u_A v_B + u_B v_A], \tag{19}$$

where P_m^2 is defined to be the collection of all *distinct* partitions (A, B) of the m-element index set $I_m \equiv \{i, j, k, \ldots, l\}$ into two subsets A and B, one of which may be the empty set $\varnothing$. (We define $u_\varnothing = u$ and $v_\varnothing = v$, and (A, B) is considered to be the same partition as (B, A).) The proof is by induction, and is based on the observation that

$$P_{m+1}^2 = \{(A \cup \{s\}, B), (A, B \cup \{s\}) \mid (A, B) \in P_m^2\} \tag{20}$$

for $I_{m+1} = I_m \cup \{s\}$, $m \geqslant 1$.

Now consider a one-dimensional rth-order continuously differentiable special function $z = h(u)$, and define $h^m = d^m h(u)/du^m$, $1 \leqslant m \leqslant r$. Table II exhibits the partial derivatives through order four of $z = h(u)$ for arbitrary indices i, j, k, l in the special case $4 \leqslant r$.

Assuming u is an rth-order continuously differentiable function of x in R^n,

TABLE II

Partial Derivatives of $z = h(u)$ through Order Four

z_i	z_{ij}	z_{ijk}	z_{ijkl}
			$h^4 u_i u_j u_k u_l$
			+
		$h^3 u_i u_j u_k$	$h^3[u_i u_j (u_{kl}) + u_i u_k (u_{jl})$
			$+ u_j u_k (u_{il}) + u_i u_l (u_{jk})$
		+	$+ u_j u_l (u_{ik}) + u_k u_l (u_{ij})]$
			+
	$h^2 u_i u_j$	$h^2[u_i (u_{jk})$	$h^2[u_i (u_{jkl}) + u_j (u_{ikl})$
		$+ u_j (u_{ik})$	$+ u_k (u_{ijl}) + u_l (u_{ijk})$
	+	$+ u_k (u_{ij})]$	$+ u_{ij} u_{kl} + u_{ik} u_{jl}$
		+	$+ u_{il} u_{jk}]$
			+
$h^1 u_i$	$h^1 u_{ij}$	$h^1 u_{ijk}$	$h^1 u_{ijkl}$

the general rule for obtaining all of the partial derivatives of $z = h(u)$ with respect to x through order r is as follows: Given any m indices $i, j, k, \ldots, l$, with $1 \leqslant m \leqslant r$, the corresponding partial derivative of z is

$$z_{ijk\cdots l} = \sum_{s=1}^{m} h^s b^s, \tag{21}$$

where the coefficient b^s, $1 \leqslant s \leqslant m$, is defined as follows: Let P_m^s denote the set of all *distinct* partitions $(A_1, \ldots, A_s)$ of the m-element set $\{i, j, k, \ldots, l\}$ into s *nonempty* subsets $A_1, \ldots, A_s$. Then

$$b^s = \sum_{(A_1,\ldots,A_s)\in P_m^s} [u_{A_1} u_{A_2} \cdots u_{A_s}]. \tag{22}$$

In practical applications, one would presumably want to reduce the number of cells in each table algorithm row by taking advantage of cross-partial equalities. For example, assuming u is an mth order continuously differentiable function of x in R^n, the *total* number of mth-order partial derivatives with respect to x of a one-variable, mth-order differentiable special function $z = h(u)$ is n^m, but the number Q_m of *distinct* mth order partial derivatives is only

$$Q_m = \sum_{j=1}^{m} \binom{m-1}{j-1}\binom{n}{j} = \binom{n+m-1}{m}, \qquad \binom{n}{j} \equiv 0 \quad \text{if} \quad j > n. \tag{23}$$

For many special functions h, the ordinary derivatives $h^1, \ldots, h^m$ in (21) can be evaluated by means of simple recurrence relations. For example, for $h(u) \equiv \log(u)$ one has

$$h^{s+1} = -su^{-1}h^s, \qquad s \geqslant 1, \tag{24a}$$

$$h^1 = u^{-1}. \tag{24b}$$

In addition, for Wengert $(r, 1)$ functions, all of the needed derivatives for one-variable special functions will generally be obtainable by use of simple recurrence relations based on Leibnitz's rule.[3] For example, consider the one-variable special function $z = e^u$. Defining

$$(e^u)^0 = e^u, \qquad (e^u)^1 = \frac{d(e^u)}{dx}, \qquad (e^u)^2 = \frac{d^2(e^u)}{dx^2}, \ldots, \tag{25a}$$

$$u^0 = u, \qquad u^1 = \frac{du}{dx}, \qquad u^2 = \frac{d^2u}{dx^2}, \ldots; \tag{25b}$$

[3] Leibnitz's rule is as follows: Given the product $h = fg$ of any two mth-order differentiable functions $f: R \to R$ and $g: R \to R$, one has $h^m = \sum_{j=0}^{m} \binom{m}{j} (f)^j (g)^{m-j}$.

note that $(e^u)^1 = zu^1$. Hence, using Leibnitz's rule, one obtains the recurrence relation

$$(e^u)^{m+1} = \sum_{j=0}^{m} \binom{m}{j} (e^u)^j (u^1)^{m-j}, \tag{26a}$$

or, equivalently,

$$\frac{(e^u)^{m+1}}{(m+1)!} = \left(\frac{1}{m+1}\right) \sum_{j=0}^{m} \left[\frac{(e^u)^j}{j!}\right]\left[\frac{(u^1)^{m-j}}{(m-j)!}\right]. \tag{26b}$$

For $z = \log(u)$, note $z^1 u = u^1$. Hence, one can use

$$(u)^{m+1} = \sum_{j=0}^{m} \binom{m}{j} (z^1)^j (u)^{m-j} \tag{27}$$

to obtain the recurrence relation for z^1,

$$(z^1)^m = \frac{1}{u}\left[(u)^{m+1} - \sum_{j=0}^{m-1} \binom{m}{j} (z^1)^j (u)^{m-j}\right], \qquad m \geqslant 1, \tag{28a}$$

$$z^1 = u^1/u. \tag{28b}$$

Finally, for $z = \sin(u)$ and $y = \cos(u)$, one obtains the system of recurrence relations for z and y,

$$(z)^{m+1} = \sum_{j=0}^{m} \binom{m}{j} (y)^j (u^1)^{m-j}, \qquad m \geqslant 0, \tag{29a}$$

$$(y)^{m+1} = \sum_{j=0}^{m} \binom{m}{j} (z)^j (-u^1)^{m-j}, \qquad m \geqslant 0, \tag{29b}$$

$$z^0 = \sin(u), \tag{29c}$$

$$y^0 = \cos(u). \tag{29d}$$

5. Discussion

A finite "table algorithm" has been developed for the systematic exact evaluation of higher order partial derivatives of functions of many variables. For any given application, one would of course want to take advantage of special problem features to increase its efficiency. The general table algorithm is simply a starting point for such investigations.

The main importance of the general table algorithm is that it suggests that higher order partial derivative evaluation can be rendered *routine*. It thus seems time to reconsider, and perhaps even to reverse, the long-term trend towards numerical methods which avoid partial derivative evaluation. Potential applications exist for the table algorithm wherever gradients, Jacobians, Hessians, and power series expansions could be employed. Examples of such applications are optimization, comparative statics, approximation theory, integration of ordinary and partial differential equations, and the use of quasi-linearization and Newton–Raphson methods for system identification.

REFERENCES

1. R. BELLMAN, H. KAGIWADA, AND R. KALABA, Wengert's numerical method for partial derivatives, orbit determination, and quasi-linearization, *Comm. ACM* **8** (1965), 231–232.
2. G. FORSYTHE, M. MALCOLM, AND C. MOLER, "Computer Methods for Mathematical Computations," Prentice–Hall, Englewood Cliffs, N.J., 1977.
3. R. KALABA AND L. TESFATSION, Complete comparative static differential equations, *Nonlinear Anal.* **5** (1981), 821–833.
4. R. KALABA, L. TESFATSION, AND J.-L. WANG, Local and Nonlocal Comparative Static Analysis of Economic Systems, *Appl. Math. Comp.* **9** (1981), 227–234.
5. SAT/ETS "User's Guide," Econometric and Time Series Library, SAS Institute, 1980 ed.
6. R. WENGERT, A simple automatic derivative evaluation program, *Comm. ACM* **7** (1964), 463–464.