

Nonlocal sensitivity analysis, automatic derivative evaluation, and sequential nonlinear estimation

Robert KALABA and Leigh TEFATSION

*Modelling Research Group, Department of Economics, University of Southern California,
Los Angeles, CA 90089-0035, USA*

Received 6 March 1986

Abstract: The present paper summarizes recent work by the authors on computational methods for nonlinear processes. Section 2 develops a complete set of ordinary differential equations for generating solutions to parameterized systems of nonlinear equations over parameter intervals of interest. Section 3 presents a simple finite algorithm for the systematic exact evaluation of higher-order partial derivatives. Section 4 obtains an exact sequential characterization for the solution to a general nonlinear least-squares estimation problem as the duration of the process increases and additional observations are made.

Keywords: Nonlocal sensitivity analysis, automatic derivative evaluation, sequential nonlinear estimation

1. Introduction

In statistics, as in other disciplines, linear models have been extensively relied upon to provide analytically tractable representations for physical processes. However, cause and effect are not always proportional, and nonlinearities have to be dealt with. Recent advances in computer hardware and software suggest that dramatic changes lie ahead, that the traditional intense focus on linear models will be replaced by an increased emphasis on nonlinear models as practical methods are evolved for the routine analysis of nonlinearities.

The present paper summarizes recent work by the authors on computational methods for nonlinear processes. Section 2 focuses on parameterized systems of nonlinear equations, e.g., the first-order necessary conditions for maximum likelihood estimation. Often it is of interest to understand how solutions for such systems vary with changes in a parameter. It is shown how one can obtain a complete system of ordinary differential equations for tracking solution branches over parameter intervals. As a by-product, a complete system of ordinary differential equations is obtained for tracking the adjoint, the determinant, the

eigenvalues, and the right and left eigenvectors of parameterized matrices over parameter intervals.

The nonlocal sensitivity techniques discussed in section 2 require numerous evaluations of partial derivatives. Thus, recourse to automatic derivative evaluation is needed to ensure their practicality. In section 3 a conceptually and computationally straightforward algorithm (FEED) is presented for the systematic exact evaluation of higher-order partial derivatives of functions of many variables.

Finally, section 3 focuses on model specification and estimation for nonlinear processes. Recognizing the difficulty of obtaining distributional properties for estimators in such contexts, an alternative approach is proposed.

Prior theoretical specifications are combined with sample data to form a complete model specification. The prior theoretical specifications may take many forms: e.g., nonlinear dynamic relationships among variables, measurement error postulates, and/or stochastic restrictions on error terms. The reasonableness of the complete model specification is examined by means of an 'incompatibility' measure, a weighted sum of squared residual modelling errors whose minimized value converges almost surely to zero if the prior theoretical specifications are correct. Although it is not possible to say with precision how small is small, it is at least possible to make incompatibility comparisons for competing model specifications. An algorithm is presented for the exact sequential minimization of the incompatibility measure as the duration of the process increases and additional observations are obtained.

2. Nonlocal sensitivity analysis

Sensitivity studies in many fields typically reduce to determining the response of a vector $x^0 = (x_1^0, \dots, x_n^0)$ to changes in a scalar α^0 , where x^0 and α^0 are required to satisfy an n -dimensional system of nonlinear equations of the form

$$\mathbf{0} = \psi(x, \alpha) \equiv \begin{bmatrix} \psi^1(x, \alpha) \\ \vdots \\ \psi^n(x, \alpha) \end{bmatrix}. \quad (1)$$

Given well-known regularity conditions, the implicit function theorem guarantees the existence of a continuously differentiable function $x(\alpha)$ taking some open neighborhood $N(\alpha^0)$ of α^0 into R^n such that

$$\mathbf{0} = \psi(x(\alpha), \alpha), \quad \alpha \in N(\alpha^0). \quad (2)$$

From (2) one obtains the well-known fundamental equation of sensitivity analysis,

$$dx(\alpha)/d\alpha = -\psi_x(x(\alpha), \alpha)^{-1} \psi_\alpha(x(\alpha), \alpha), \quad \alpha \in N(\alpha^0). \quad (3)$$

As it stands, (3) is an analytically incomplete system of ordinary differential equations. That is, a closed form representation for the Jacobian inverse $J(\alpha)^{-1}$

$\equiv \psi_x(x(\alpha), \alpha)^{-1}$ as a function of α is often not obtainable for $n \geq 3$. Thus, the integration of (3) from initial conditions would typically require the supplementary algebraic determination of the Jacobian inverse $J(\alpha)^{-1}$ at each step in the integration process. In any case, system (3) is interpreted by most analysts as a purely algebraic relation of the form $z = A^{-1}b$ which can be used, qualitatively, to sign or partially sign the components of $dx(\alpha)/d\alpha$ at the point α^0 ; numerical integration using specific functional forms is generally not attempted.

In Kalaba and Tesfatsion [1] the differential system (3) is completed by the incorporation of ordinary differential equations for the Jacobian inverse. More precisely, letting $A(\alpha)$ and $\delta(\alpha)$ denote the adjoint matrix and the determinant of the Jacobian matrix $J(\alpha)$, and recalling that the inverse of any nonsingular matrix can be represented as the ratio of its adjoint matrix to its determinant, the following differential system is validated for $x(\alpha)$, $A(\alpha)$, and $\delta(\alpha)$:

$$dx(\alpha)/d\alpha = -A(\alpha)\psi_\alpha(x(\alpha), \alpha)/\delta(\alpha), \quad (4a)$$

$$dA(\alpha)/d\alpha = [A(\alpha)\text{Trace}(A(\alpha)B(\alpha)) - A(\alpha)B(\alpha)A(\alpha)]/\delta(\alpha), \quad (4b)$$

$$d\delta(\alpha)/d\alpha = \text{Trace}(A(\alpha)B(\alpha)), \quad (4c)$$

where $B(\alpha) \equiv dJ(\alpha)/d\alpha$ is expressible as a known function of $x(\alpha)$, $A(\alpha)$, $\delta(\alpha)$, and α . Initial conditions for system (4) must be provided at a parameter point α^0 by specifying values for $x(\alpha^0)$, and $\delta(\alpha^0)$ satisfying

$$\psi(x(\alpha^0), \alpha^0) = 0, \quad (4d)$$

$$A(\alpha^0) = \text{Adj}(J(\alpha^0)), \quad (4e)$$

$$\delta(\alpha^0) = \det(J(\alpha^0)) \neq 0. \quad (4f)$$

If initialization at α^0 is difficult, a homotopy continuation method might be used to simplify the process. For example, consider the modified system of equations

$$0 = \psi^*(x, \beta) \equiv \beta\psi(x, \alpha^0) + [1 - \beta]Ix. \quad (5)$$

In principle, the differential system (4) with $\psi^*(x, \beta)$ in place of $\psi(x, \alpha)$ can be integrated over the β interval $[0,1]$ to yield the needed initial conditions at α^0 .

In summary, (4) provides an analytically complete system of ordinary differential equations for tracking the solution vector $x(\alpha)$ together with the adjoint and determinant of the Jacobian matrix $J(\alpha)$ over α intervals of interest. A wide variety of specific functional forms for $\psi(\cdot)$ can be tested with minimal effort. In this sense, system (4) makes possible nonlocal sensitivity analysis.

The differential system (4) has been integrated on an IBM 370/Model 158 for various illustrative examples using a single precision FORTRAN program incorporating a fourth order Adams–Moulton integration method with a Runge–Kutta start. As reported in Kalaba and Tesfatsion [1] and Kalaba et al. [2], high numerical accuracy was obtained, even near critical points c where the determinant $\delta(\alpha)$ became singular. In addition, system (4) has been employed by several researchers at the University of Southern California to analyze certain nonlinear

models arising in development economics. See, for example, Datta et al. [3] and Datta et al. [4].

In optimization and equilibrium investigations, the definiteness and stability properties of the Jacobian matrix $J(\alpha)$ are often of major concern. In a companion paper, Kalaba et al. [5], it is shown how the eigenvalues and the right and left eigenvectors of a parameterized matrix such as $J(\alpha)$ can be tracked as functions of α by integrating a system of ordinary differential equations from initial conditions. In Kalaba et al. [6] an analogous differential system is developed for tracking a single eigenvalue of a parameterized matrix together with one of its corresponding right or left eigenvectors over parameter intervals of interest. Finally, in Kalaba et al. [7] it is shown how the differential system developed in [6] can be modified to obtain an initial value problem for determining the Perron root and a corresponding unit normalized right eigenvector for any positive $n \times n$ matrix.

3. Automatic derivative evaluation

In the process of developing methods for nonlocal sensitivity analysis [1,2], 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.

Use was first made of a method developed by Wengert [13] 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 or 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-order and second-order partial derivatives, two principal difficulties arose in attempting to extend Wengert's method to higher-order 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.

In Kalaba et al. [8] a 'table algorithm' is developed 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, or FEED ('fast efficient evaluation of derivatives') as we now refer to it, is conceptually and computationally straightforward, and it is difficult to believe that previous researchers have not already discovered it. However, no presentations of such an algorithm have been found by us in the previous journal literature, or conveyed to us by readers of [8]. Even Wengert's method is rarely cited. (See, e.g. Rall [12].)

FEED has now been successfully tested in a variety of applications, including nonlinear least squares, optimal control, system identification, nonlinear two-point boundary value problems, and the numerical solution of nonlinear integral equations. Discussion of these applications can be found in Kalaba and Tishler [10], and especially in Kalaba et al. [11].

In Kalaba and Tesfatsion [9] it is shown that FEED can be applied to a much broader class of functions than originally envisioned in [8]. Specifically, FEED can be used to evaluate the higher-order partial derivatives of functions which are defined in terms of the derivatives of other functions, a task required in many applications.

A simple application of FEED appearing in [11, pp. 5-8] will now be given for illustration.

An illustrative example of FEED

Consider the function $F: R_{++}^2 \rightarrow R$, defined by

$$z = F(x, y) \equiv x + \log(xy). \tag{6}$$

Suppose one wishes to evaluate the function value z , the first-order partial derivatives z_x and z_y , and the second-order partial derivative z_{xx} , at a given domain point (x, y) . Consider Table 1.

The first column of the table sequentially evaluates the function value $z = x + \log(xy)$ at the given domain point (x, y) . The second, third, and fourth entries in each row give the indicated derivative evaluations of the first entry in the row, using only algebraic operations. The first two rows initialize the algorithm, one row being required for each function variable. The only input required for the first two rows is the domain point (x, y) . Each subsequent row outputs a one-dimensional array of the form (p, p_x, p_y, p_{xx}) , using the arrays obtained

Table 1

Variable	$\partial/\partial x$	$\partial/\partial y$	$\partial^2/\partial x^2$
$a = x$	$a_x = 1$	$a_y = 0$	$a_{xx} = 0$
$b = y$	$b_x = 0$	$b_y = 1$	$b_{xx} = 0$
$c = ab$	$c_x = a_x b + ab_x$	$c_y = a_y b + ab_y$	$c_{xx} = a_{xx} b + 2a_x b_x + ab_{xx}$
$d = \log(c)$	$d_x = c^{-1}c_x$	$d_y = c^{-1}c_y$	$d_{xx} = -c^{-2}c_x^2 + c^{-1}c_{xx}$
$z = a + d$	$z_x = a_x + d_x$	$z_y = a_y + d_y$	$z_{xx} = a_{xx} + d_{xx}$

from previous row calculations as inputs. The final row yields the desired evaluations (z, z_x, z_y, z_{xx}) . These evaluations are exact up to round-off error.

It will now be shown how the table rows can be sequentially evaluated by means of FORTRAN calculus subroutines at any given domain point $(x, y) = (X, Y)$.

The first two rows of the table can be evaluated by calling calculus subroutines LIN1 and LIN2, defined as follows:

<pre> SUBROUTINE LIN1(X, A) DIMENSION A(4) A(1) = X A(2) = 1.0 A(3) = 0.0 A(4) = 0.0 RETURN END </pre>	<pre> SUBROUTINE LIN2(Y, B) DIMENSION B(4) B(1) = Y B(2) = 0.0 B(3) = 1.0 B(4) = 0.0 RETURN END </pre>
--	--

(7)

Subroutine LIN1 uses standard calculus formulas to obtain the value and first three partial derivatives $A = (a, a_x, a_y, a_{xx}) = (X, 1, 0, 0)$ of the function a of x and y defined by $a(x, y) = x$, given any particular value X for x . Thus, the output array A evaluates the first row of the table at $x = X$. Similarly, subroutine LIN2 obtains the value and first three partial derivatives $B = (b, b_x, b_y, b_{xx}) = (Y, 0, 1, 0)$ of the function b of x and y defined by $b(x, y) = y$, given any particular value Y for y . Thus, the output array B evaluates the second row of the table at $y = Y$.

Row three can be evaluated by calling the following calculus subroutine for multiplication:

```

SUBROUTINE MULT(A, B, C)
DIMENSION A(4), B(4), C(4)
C(1) = A(1) * B(1)
C(2) = A(2) * B(1) + A(1) * B(2)
C(3) = A(3) * B(1) + A(1) * B(3)
C(4) = A(4) * B(1) + 2.0 * A(2) * B(2) + A(1) * B(4)
RETURN
END

```

(8)

Subroutine MULT uses standard calculus formulas to obtain the value and first three partial derivatives $C = (c, c_x, c_y, c_{xx})$ of the function $c = ab$, where a and b are any two real-valued twice-differentiable functions of x and y , and the input arrays $A = (a, a_x, a_y, a_{xx})$ and $B = (b, b_x, b_y, b_{xx})$ contain the value and first three partial derivatives of a and b at (X, Y) . Thus, $c_x = a_x b + a b_x$, $c_y = a_y b + a b_y$, and so forth. For the case at hand, the input arrays A and B are the outputs of table rows one and two.

Row four can be evaluated by calling the following calculus subroutine for the log function:

```

SUBROUTINE LOGG(C, D)
DIMENSION C(4), D(4)
D(1) = ALOG(C(1))
D(2) = C(2)/C(1)
D(3) = C(3)/C(1)
D(4) = -D(2)* *2 + C(4)/C(1)
RETURN
END

```

(9)

Subroutine LOGG uses standard calculus formulas to obtain the value and first three partial derivatives $D = (d, d_x, d_y, d_{xx})$ of the function $d = \log(c)$, where c is any real-valued, positive, twice-differentiable function of x and y , and the input array $C = (c, c_x, c_y, c_{xx})$ contains the value and first three partial derivatives of c at (X, Y) . For the case at hand, the input array C is the output of table row three. Note that the calculus subroutine LOGG calls the FORTRAN library function subroutine ALOG for the log function.

Finally, row five can be evaluated by calling the following calculus subroutine for addition:

```

SUBROUTINE ADD (A, D, Z)
DIMENSION A(4), D(4), Z(4)
DO 100 I = 1,4
100 Z(I) = A(I) + D(I)
RETURN
END

```

(10)

Subroutine ADD uses standard calculus formulas to obtain the value and first three partial derivatives $Z = (z, z_x, z_y, z_{xx})$ of the function $z = a + d$, where a and d are any two real-valued twice-differentiable functions of x and y , and the input arrays $A = (a, a_x, a_y, a_{xx})$ and $D = (d, d_x, d_y, d_{xx})$ contain the value and first three partial derivatives of a and d at (X, Y) . For the case at hand, the input arrays A and D are the outputs of table rows one and four.

The complete sequential evaluation of the table rows is accomplished by means of the following master subroutine:

```

SUBROUTINE FUN(X, Y, Z)
DIMENSION A(4), B(4), C(4), D(4), Z(4)
CALL LIN1(X, A)
CALL LIN2(Y, B)
CALL MULT(A, B, C)
CALL LOGG(C, D)
CALL ADD(A, D, Z)
RETURN
END

```

(11)

For any given domain point (X, Y) , subroutine FUN obtains the value and first three partial derivatives (z, z_x, z_y, z_{xx}) of the function $z = x + \log(xy)$ at (X, Y) by means of the indicated sequence of calls to the calculus subroutines (7) through (10). The evaluations (z, z_x, z_y, z_{xx}) are returned in the array Z.

In principle, a FUN subroutine can be constructed to calculate the value and partial derivatives through order k , $k \geq 1$, of any real-valued multi-variable function F which can be sequentially evaluated by means of the special two-variable functions

$$w = u + v, \quad w = u - v, \quad w = uv, \quad w = u/v, \quad w = u^v, \quad (12a)$$

and arbitrary, one-variable, k -th-order differentiable functions such as

$$\sin(u), \quad \cos(u), \quad \exp(u), \quad \log(u), \quad \text{and} \quad au^b + c \quad (12b)$$

for arbitrary constants, a , b , and c . Systematic rules for constructing k -th-order calculus subroutines for the special functions (12) are presented in Kalaba et al. [8].

A recent paper by Wexler [14] proposes an important improvement in the implementation of FEED. Rather than explicitly listing a sequence of call statements in a subroutine FUN to evaluate a function and its partial derivatives, as illustrated above in (11), the user simply writes an assignment statement which closely resembles the form of the function to be evaluated. For example, to obtain the value z and partial derivatives z_x , z_y , and z_{xx} of the function $z = x + \log(xy)$, the user writes an assignment statement of the form

$$K = \text{IADD}(\text{IX}, \text{ILOGG}(\text{IMULT}(\text{IX}, \text{IY}))). \quad (13)$$

The FORTRAN compiler then evaluates the expression (13), automatically calling the needed calculus subroutines in correct sequence. The integer K is a pointer which indicates where in memory the function and derivative values (z, z_x, z_y, z_{xx}) are stored.

4. Sequential nonlinear estimation

Consider a dynamical process described by an n -dimensional system of nonlinear difference equations

$$x_{t+1} = F(x_t) + \varepsilon_t, \quad t = 0, \dots, T-1, \quad (14a)$$

where x_t is an n -dimensional column vector of state variables and ε_t is an n -dimensional column vector of unknown dynamical errors, $n \geq 1$. Observations on the state vectors are obtained in the nonlinear form

$$y_t = H(x_t) + \eta_t, \quad t = 0, \dots, T, \quad (14b)$$

where y_t is an m -dimensional column vector of observations and η_t is an

m -dimensional column vector of unknown observational errors, $m \geq 1$. The dynamical and observational errors are believed a priori to be close to zero, i.e.,

$$\varepsilon_t \approx \mathbf{0}, \quad t = 0, \dots, T-1; \tag{14c}$$

$$\eta_t \approx \mathbf{0}, \quad t = 0, \dots, T. \tag{14d}$$

The basic problem at hand is assumed to be the reconciliation of theory with observations. Specifically, at each time T the modeller must determine whether or not there exists *any* sequence of state vectors x_0, \dots, x_T which satisfies the theoretical specifications (14) in an acceptable approximate sense for the realized sequence of observation vectors y_0, \dots, y_T . For simplicity, we suppose that the incompatibility of the theoretical specifications (14) with the observation vectors y_0, \dots, y_T is measured by minimizing a weighted sum of squared residual error terms of the form

$$\sum_{t=0}^T |y_t - H(x_t)|^2 + k \sum_{t=0}^{T-1} |x_{t+1} - F(x_t)|^2 \tag{15}$$

with respect to x_0, \dots, x_T , where k is a fixed positive scalar weight and $| \cdot |$ denotes the usual Euclidean norm.

In Kalaba and Tesfatsion [15] two algorithms are developed for the exact sequential minimization of the cost function (15) as the duration of the process increases and additional observation vectors are obtained. The first algorithm proceeds via an imbedding on the process length and the final state vector. The second algorithm proceeds via an imbedding on the process length and the final observation vector. Each algorithm generates least-cost filtered and smoothed state estimates, together with least-cost one-step-ahead state predictions.

The two algorithms developed in [15] for the sequential minimization of (15) are generalizations of the sequential estimation procedure developed in Kalaba and Tesfatsion [16] and numerically tested in Kalaba et al. [17]. Derivation of the first algorithm is outlined below.

Let $C_{t-1}(x_0, \dots, x_t)$ denote the cost of the estimation process starting at time 0 and extending through time $t-1$, conditioned on the state estimates x_0, \dots, x_t . Thus,

$$C_{t-1}(x_0, \dots, x_t) \equiv \sum_{s=0}^{t-1} |y_s - H(x_s)|^2 + k \sum_{s=0}^{t-1} |x_{s+1} - F(x_s)|^2 \tag{16}$$

for $t \geq 1$. Also, let $\phi_{t-1}(x_t)$ denote the smallest cost of the estimation process starting at time 0 and extending through time $t-1$, conditioned on the state estimate x_t . Thus,

$$\phi_{t-1}(x_t) \equiv \inf_{x_0, \dots, x_{t-1}} C_{t-1}(x_0, \dots, x_t) \tag{17}$$

for $t \geq 1$.

By construction, each cost function C_{t-1} in (16) is bounded below over its domain $R^{n(t+1)}$. It follows (see [15]) that the functions ϕ_{t-1} in (17) satisfy the recurrence relation

$$\phi_t(x_{t+1}) = \inf_{x_t} \left\{ |y_t - H(x_t)|^2 + k |x_{t+1} - F(x_t)|^2 + \phi_{t-1}(x_t) \right\} \tag{18a}$$

for all x_{t+1} in R^n and $t \geq 0$. The recurrence relation (18a) is initialized by assigning a prior cost-of-estimation value $\phi_{-1}(x_0)$ to each x_0 in R^n , for example,

$$\phi_{-1}(x_0) \equiv 0. \quad (18b)$$

Thus, the functions $\phi_0(x_1)$, $\phi_1(x_2)$, ... are determined one after the other. In practice, a grid could be introduced for x_1, x_2, \dots , and the infima could be found by direct search. Alternatively, use might be made of the first-order necessary conditions associated with the successive minimizations. (See [15, Section IV].)

The recurrence relation (18a) can be given a dynamic programming interpretation. The bracketed term on the right-hand side of (18a) gives the total cost of the estimation process at time t , conditioned on the state estimates x_0, \dots, x_{t+1} . This cost is decomposed into three parts. The first part, $|y_t - H(x_t)|^2$, denotes the penalty imposed if the time- t state estimate x_t fails to satisfy the basic observational constraints (14b) and (14d). The second part, $k|x_{t+1} - F(x_t)|^2$, denotes the penalty imposed if the time- t state estimate x_t and the time- $(t+1)$ state estimate x_{t+1} fail to satisfy the basic dynamic constraints (14a) and (14c). The third part, $\phi_{t-1}(x_t)$, denotes the minimum total penalty incurred over times 0 through $t-1$ for all deviations from constraints (14), given that the time- t state estimate is x_t . Thus, $\phi_t(x_{t+1})$, the infimum of the bracketed term with respect to x_t , yields the minimum total penalty incurred over times 0 through t for all deviations from constraints (14), given that the time- $(t+1)$ state estimate is x_{t+1} .

Suppose the cost function (15) is minimized by a unique sequence of state vectors for each process length $T \geq 0$. Then the following exact sequential procedure can be used to generate the optimal (cost-minimizing) solution $\hat{x}(T|T)$ for the state vector x_T at time T for each process length $T \geq 0$.

In storage at time $T=0$ is the function $\phi_{-1}(x_0)$, defined for all x_0 in R^n . A first observation vector y_0 is obtained. The optimal estimate $\hat{x}(0|0)$ for the state vector x_0 at time 0, based on the single observation vector y_0 , is found by minimizing the expression

$$\{|y_0 - H(x_0)|^2 + \phi_{-1}(x_0)\} \quad (19)$$

with respect to x_0 . Formally,

$$\hat{x}(0|0) = \arg \left[\min_{x_0} \{|y_0 - H(x_0)|^2 + \phi_{-1}(x_0)\} \right]. \quad (20)$$

In preparation for the next time 1, determine and store for each x_1 in R^n the function value

$$\phi_0(x_1) \equiv \min_{x_0} \{|y_0 - H(x_0)|^2 + k|x_1 - F(x_0)|^2 + \phi_{-1}(x_0)\}. \quad (21)$$

In storage at time $T \geq 0$ is the function $\phi_{T-1}(x_T)$, defined for all x_T in R^n . An additional observation vector y_T is obtained. The optimal estimate $\hat{x}(T|T)$ for the state vector x_T at time T , based on the observation vectors y_0, \dots, y_T , is then given by

$$\hat{x}(T|T) = \arg \left[\min_{x_T} \{|y_T - H(x_T)|^2 + \phi_{T-1}(x_T)\} \right]. \quad (22)$$

In preparation for the next time $T + 1$, determine and store for each x_{T+1} in R^n the function value

$$\phi_T(x_{T+1}) \equiv \min_{x_T} \left\{ |y_T - H(x_T)|^2 + k|x_{T+1} - F(x_T)|^2 + \phi_{T-1}(x_T) \right\}. \quad (23)$$

A proof that (22) does yield the optimal (cost-minimizing) solution for the time- T state vector x_T based on the observations y_0, \dots, y_T is given in [15]. It is also shown that the recurrence relation (23) can be used to generate the optimal solution $\hat{x}(0|T), \dots, \hat{x}(T-1|T)$ for the state vectors x_0, \dots, x_{T-1} based on the observations y_0, \dots, y_T .

References [15] through [17] stress exact sequential estimation procedures. Nevertheless, the basic conceptual idea underlying these papers concerns model specification. It is proposed that a modeller form a cost function which imposes a penalty for any deviation away from his theoretical model specifications, whatever form these specifications might take. As elaborated in Kalaba and Tesfatsion [18], this basic conceptual idea can be applied in principle to any model specification problem for which the model specifications are representable as a system of equality and inequality constraints. In particular, stochastic restrictions on error terms can be incorporated into the cost function along with dynamical and observational constraints.

For example, a prior belief that the error terms ϵ_t are independent drawings from a distribution with zero mean might be handled by replacing restriction (14c) with a restriction on the sample mean of ϵ_t of the form

$$\left[\sum_{t=0}^{T-1} \epsilon_t \right] / T \approx 0. \quad (24)$$

The cost function (15) should then be modified to take into account the new model specification (24) together with the previous model specifications (14a), (14b), and (14d). For example the modified cost function might take the form

$$k_0 \sum_{t=0}^T |y_t - H(x_t)|^2 + k_1 \sum_{t=0}^{T-1} |x_{t+1} - F(x_t) - \epsilon_t|^2 + k_2 \left| \left[\sum_{t=0}^{T-1} \epsilon_t \right] / T \right|^2, \quad (25)$$

where k_0 , k_1 , and k_2 are positive scalar weights, normalized to sum to one.

The cost function (25) is then minimized with respect to both the state variables (x_0, \dots, x_T) and the error terms ($\epsilon_0, \dots, \epsilon_{T-1}$). How this minimization to be accomplished, whether by sequential or other methods, is a secondary issue. If all of the model specifications are correct, and the error term components have finite variance, then, by the strong law of large numbers, the minimized value of the cost function (25) will almost surely be close to zero for sufficiently large T . Although it is not possible to say with precision how small is small, one can at least make cost comparisons for competing model specifications. Once the basic plausibility of a model has been established, statistical techniques can be used to further refine estimates.

In another series of studies (Tsfatsion [19–21]), the direct sequential updating of criterion functions ('criterion filtering') is stressed as a conceptually and

computationally attractive alternative to the updating of probability distributions for certain classes of adaptive control problems. In contrast to Kalaba and Tesfatsion [15], the criterion filters proposed in these studies are designed to generate asymptotically optimal controls while by-passing explicit state estimation altogether. Analytical and computer simulation studies for the convergence and optimality properties of several specific criterion filters are presented in [19,20], and additional papers cited in [20]. The analogy between use of criterion filters for criterion function updating and use of Bayes' rule for probability distribution updating is investigated in [21].

References

Nonlocal sensitivity analysis

- [1] R. Kalaba and L. Tesfatsion, Complete comparative static differential equations, *Nonlinear Anal.* **5** (1981) 821–833.
- [2] R. Kalaba, L. Tesfatsion, and J.-L. Wang, Local and nonlocal comparative static analysis of economic systems, *Appl. Math. Comput.* **9** (1981) 227–234.
- [3] S. Datta, J. Nugent, and A. Tishler, Determination of the contractual mix between cash and kind wages for developing agrarian economies, MRG Working Paper No. 8313, Department of Economics, University of Southern California (1983).
- [4] S. Datta, J. Nugent, and A. Tishler, Seasonality, differential access, and interlinking of labor and credit, MRG Working Paper, Department of Economics, University of Southern California (June 1985).
- [5] R. Kalaba, K. Spingarn, and L. Tesfatsion, Variational equations for the eigenvalues and eigenvectors of nonsymmetric matrices, *J. Optim. Theory Appl.* **33** (1981) 1–8.
- [6] R. Kalaba, K. Spingarn, and L. Tesfatsion, Individual tracking of an eigenvalue and eigenvector of a parameterized matrix, *Nonlinear Anal.* **5** (1981) 337–340.
- [7] R. Kalaba, K. Spingarn, and L. Tesfatsion, A new differential equations method for finding the Perron root of a positive matrix, *Appl. Math. Comput.* **7** (1980) 187–193.

Automatic derivative evaluation

- [8] R. Kalaba, L. Tesfatsion, and J.-L. Wang, A finite algorithm for the exact evaluation of higher-order partial derivatives of functions of many variables, *J. Math. Anal. Appl.* **12** (1983) 181–191.
- [9] R. Kalaba and L. Tesfatsion, Automatic differentiation of functions of derivatives, MRG Working Paper No. 8529, Department of Economics, University of Southern California (November 1985), to appear in *Comp. Math. Appl.*
- [10] R. Kalaba and A. Tishler, Automatic derivative evaluation in the optimization of nonlinear models, *Rev. Econom. Statist.* **66** (1984) 653–660.
- [11] H. Kagiwada, R. Kalaba, N. Rasakhoo, and K. Spingarn, *Numerical Derivatives and Nonlinear Analysis* (Plenum, New York, 1986).
- [12] L.B. Rall, *Automatic Differentiation of Functions of Derivatives* (Springer, New York, 1981).
- [13] R. Wengert, A simple automatic derivative evaluation program, *Comm. ACM* **7** (1964) 463–464.
- [14] A. Wexler, Automatic evaluation of derivatives, Working Paper, Department of Physiology and Biophysics, University of Southern California (June 1985).

Sequential nonlinear estimation

- [15] R. Kalaba and L. Tesfatsion, Exact sequential filtering, smoothing, and prediction for nonlinear systems, Modelling Research Group Working Paper No. 8509, Department of Economics, University of Southern California (June 1985).
- [16] R. Kalaba and L. Tesfatsion, Exact sequential solutions for a class of discrete-time nonlinear estimation problems, *IEEE Trans. Automat. Control* **26** (1981) 1144–1149.
- [17] R. Kalaba, K. Spingarn, and L. Tesfatsion, A sequential method for nonlinear filtering: Numerical implementation and comparisons, *J. Optim. Theory Appl.* **34** (1981) 541–559.
- [18] R. Kalaba and L. Tesfatsion, A least-squares model specification test for a class of dynamic nonlinear economic models with systematically varying parameters, *J. Optim. Theory Appl.* **32** (1980) 538–567.
- [19] L. Tesfatsion, A new approach to filtering and adaptive control, *J. Optim. Theory Appl.* **25** (1978) 247–261.
- [20] L. Tesfatsion, Direct updating of intertemporal criterion functions for a class of adaptive control problems, *IEEE Trans. Systems Man Cybernet.* **9** (1979) 143–151.
- [21] L. Tesfatsion, A dual approach to Bayesian inference and adaptive control, *Theory and Decision* **14** (1982) 177–194.