Basic Paper: February 1997
(Updates of references listed as "forthcoming," January 2004)

Nonlocal Sensitivity Analysis with Automatic Differentiation [1]

Leigh Tesfatsion
Professor of Economics and Mathematics
Iowa State University
Ames, IA 50011-1070
http://www.econ.iastate.edu/tesfatsi/
Email: tesfatsi@iastate.edu

Keywords and Phrases: nonlocal sensitivity analysis, the Nasa program, automatic differentiation, computational differentiation, the Feed algorithm, adaptive homotopy, adaptive computation.

1 Basic Problem Formulation

Sensitivity analysis problems typically reduce to determining the response of a vector $x^* = (x_1^*, \ldots, x_n^*)$ to changes in a scalar α^*, where x^* and α^* are required to satisfy an n-dimensional system of nonlinear equations of the form[2]

$$0 \quad = \quad \psi(x, \alpha) \quad = \quad (\psi^1(x, \alpha), \ldots, \psi^n(x, \alpha))^T \ . \tag{1}$$

Assuming ψ:$R^{n+1} \rightarrow R^n$ is twice continuously differentiable and has a nonsingular Jacobian matrix $\psi_x(x^*, \alpha^*)$, the implicit function theorem guarantees the existence of a continuously differentiable function $x(\alpha)$ taking some neighborhood $N(\alpha^*)$ of α^* into R^n such that

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

[1] The published version of this article appears as pp. 92-97 in C. A. Floudas and P. M. Pardalos (eds.), *Encyclopedia of Optimization*, Volume 4, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001. See http://www.econ.iastate.edu/tesfatsi/nasahome.htm for annotated pointers to related articles and to the Nasa program (available on-line as freeware).

[2] This problem formulation, with a scalar paramater α, is more general than it might first appear. For example, suppose an analyst wishes to investigate the surface of function values $x = f(z)$ taken on by some function f:$R^m \rightarrow R^n$ as z ranges over a specified region Z in R^m. One approach is to consider a suitably smooth curve s:$[0, 1] \rightarrow Z$ which roughly fills this region, of the form $z = s(\alpha)$, and to define a new function of the form $\psi(x, \alpha) \equiv x - f(s(\alpha))$. Solving the system of equations $\psi(x, \alpha) = 0$ for x as a function of α as α ranges from 0 to 1 then yields a curve of points $x(\alpha)$ on the function surface which gives some idea of the shape of this surface over the region Z.

with $x(\alpha^*) = x^*$. From (2) one obtains the fundamental equation for sensitivity analysis,

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

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

Why not simply incorporate a linear equation solver to accomplish the needed matrix inversions? Two reasons can be given. First, the Jacobian matrix might have one or more eigenvalues which are small in absolute value. Consequently, as can be seen using a singular value decomposition, the inverse matrix can be highly ill-conditioned in the sense that its elements have large absolute values and take on both positive and negative values. In this case, small round-off and truncation errors can cause large errors in the resulting numerically determined component values of the sensitivity vector $dx(\alpha)/d\alpha$. Second, there exists an alternative approach [9] that has proven its reliability and efficiency in numerous contexts over the past twenty years: replace the algebraic operation of matrix inversion by an initial value problem highly suited for modern digital computers.

The latter approach is taken in [10]. The differential system (3) is extended by the incorporation of ordinary differential equations for the Jacobian inverse. More precisely, letting $A(\alpha)$ and $\delta(\alpha)$ denote the adjoint 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 to its determinant, the following differential system is validated for $x(\alpha)$, $A(\alpha)$, and $\delta(\alpha)$:

$$dx(\alpha)/d\alpha \quad = \quad -A(\alpha)\psi_\alpha(x(\alpha),\alpha)/\delta(\alpha) \ ; \tag{4}$$

$$dA(\alpha)/d\alpha \quad = \quad [A(\alpha)\mathrm{Trace}(A(\alpha)B(\alpha)) - A(\alpha)B(\alpha)A(\alpha)]/\delta(\alpha) \ ; \tag{5}$$

$$d\delta(\alpha)/d\alpha \quad = \quad \mathrm{Trace}(A(\alpha)B(\alpha)) \ . \tag{6}$$

The ijth component of the matrix $B(\alpha) = dJ(\alpha)/d\alpha$ appearing in equations (5) and (6) is

$$\sum_{k=1}^{n}(\psi^i_{jk}(x(\alpha),\alpha)dx_k(\alpha)/d\alpha) + \psi^i_{j,n+1}(x(\alpha),\alpha) \ , \tag{7}$$

where ψ^i_{jk} denotes the second partial of ψ^i with respect to x_j and x_k, and $\psi^i_{j,n+1}$ denotes the second partial of ψ^i with respect to x_j and α. Given (4), note that each of the components (7) is expressible as a known function of $x(\alpha)$, $A(\alpha)$, $\delta(\alpha)$, and α. Initial conditions for equations (4) through (6) must be provided at a parameter point α^* by specifying values for $x(\alpha^*)$, $A(\alpha^*)$, and $\delta(\alpha^*)$ satisfying $0 = \psi(x(\alpha^*),\alpha^*)$, $A(\alpha^*) = \mathrm{Adj}(J(\alpha^*)$, and $\delta(\alpha^*) = \mathrm{Det}(J(\alpha^*)) \neq 0$.

In summary, the system of equations (4) through (6) provides an analytically complete system of ordinary differential equations for the *nonlocal sensitivity analysis* of the original system of interest, $0 = \psi(x, \alpha)$. That is, it permits the tracking of the solution vector $x(\alpha)$ and the sensitivity vector $dx(\alpha)/d\alpha$, together with the adjoint $A(\alpha)$ and the determinant $\delta(\alpha)$ of the Jacobian matrix $J(\alpha)$, over any α-interval $[\alpha^*, \alpha^{**}]$ where the determinant remains nonzero.

2 Fully Automated Implementation

The complete differential system (4) through (6) was initially implemented in [10] by means of a fortran program incorporating a fourth-order Adams-Moulton integration method with a Runge-Kutta start and hand-coded partial derivatives. High numerical accuracy was obtained in illustrative applications, even near critical points α where the determinant $\delta(\alpha)$ became zero. Nevertheless, hand-coding of partial derivatives was clearly an undesirable feature of the program. The partial derivative expressions in (7) involve the second-order partial derivatives of $\psi(\cdot)$; and $\psi(\cdot)$ in turn could involve the partial derivatives of some still more basic function, such as the criterion function for an optimization problem. This is indeed the typical case for economic problems (e.g., the profit maximization problem handled in [10]), since such problems invariably incorporate the decision-making processes of various types of economic agents.

In consequence, a more fully automated fortran program for nonlocal sensitivity analysis was eventually developed in [11]. This program, referred to as *Nasa*,[3] incorporates a fairly substantial library for the forward-mode automatic evaluation of partial derivatives through order three [13] as well as an adaptive homotopy method [12] for automatically obtaining all required initial conditions. The following sections briefly describe these features. An example of how Nasa has been applied to an applied general equilibrium problem in economics is detailed in [2].

3 Incorporation of Automatic Differentiation

Four basic approaches[4] can be used to obtain computer-generated numerical values for derivatives: hand-coding; numerical differentiation; symbolic differentiation; and automatic derivative evaluation, or *automatic differentiation* for short.[5] Numerical differentiation methods substitute discrete approximate forms for derivative expressions. For example, finite difference methods involve the approximation of derivatives by ratios of discrete increments; e.g., $f'(t) \approx [f(t + h) - f(t)]/h$

[3] Nasa is an acronym for *N*onlocal *A*utomated *S*ensitivity *A*nalysis. Nasa is available for downloading as freeware from the Web site http://www.econ.iastate.edu/tesfatsi/.

[4] See Jerrell [7] for an interesting comparative discussion of these four alternative approaches.

[5] Recently, *computational differentiation* has come to be the preferred term for automatic differentiation; see [1]. To avoid confusion, the more traditional term is used here.

for some suitably small h. Symbolic differentiation methods generate exact symbolic expressions for derivatives that can be manipulated algebraically as well as evaluated numerically. In contrast, automatic differentiation methods do not generate explicit derivative expressions, either approximate or symbolic. Rather, these methods focus on the generation of derivative evaluations by breaking down the evaluation of a derivative at a given point into a sequence of simpler evaluations for functions of at most one or two variables. These evaluations are exact up to round-off and truncation error.

For the nonlocal sensitivity analysis problem outlined in Section 1, the primary requirement is for partial derivative evaluations through order three to be obtained in a reliable and efficient manner. The use of numerical differentiation methods such as finite difference introduces systematic approximation errors into applications that can be reduced but not eliminated entirely due to the risk of catastrophic floating point error. Symbolic differentiation software packages such as Macsyma, Mathematica, and Maple[6] produce analytical expressions for derivatives but are notorious for "expression swell"—that is, for the great many lines of code they produce for the derivative expressions of even relatively simple functional forms despite repeated use of reduction routines; see [5] for explicit examples. Thus, an automatic derivative evaluation routine would seem to be the preferred alternative for the application at hand.

Automatic differentiation appears to have been independently developed by Moore [15] and Wengert [20]. The key idea of Moore and Wengert 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, referred to below as a "function list." Total differentials of the special functions could be automatically evaluated along with the special function values, and partial derivatives could then be recovered from the total differentials by solving certain associated sets of linear algebraic equations.

As detailed in [1] and [4], great strides have been made over the past thirty years in developing fast and reliable automatic differentiation algorithms. The Nasa program incorporates one such algorithm, originally developed in [13], that is now referred to as Feed.[7] Total differentials are replaced by derivative arrays in order to avoid repeated function evaluations and the need to recover partial derivatives from total differentials for each successively higher-order level of differentiation.

As a simple illustration of Feed, consider the function $F{:}R^2_{++} \rightarrow R$ defined by

$$z = F(x, y) = x + \log(xy) \ . \tag{8}$$

[6] Automatic differentiation has recently been introduced into Maple; see Heck [6].

[7] *Feed* is an acronym for *F*ast *E*fficient *E*valuation of *D*erivatives. A detailed discussion of the use of this automatic differentiation algorithm for both optimization and sensitivity analysis can be found in [8].

Suppose one wishes to evaluate the function value z and the partial derivatives z_x, z_y, z_{xx} and z_{xxx} at a given domain point (x, y). Consider Table 3.1.

Table 3.1: An Illustrative Application of the Feed Algorithm

Function List	$\partial/\partial x$	$\partial/\partial y$	$\partial^2/\partial x^2$	$\partial^3/\partial x^3$
$a = x$	1	0	0	0
$b = y$	0	1	0	0
$c = ab$	$a_x b + a b_x$	$a_y b + a b_y$	$a_{xx} b + 2 a_x b_x + a b_{xx}$	$a_{xxx} b + 3 a_{xx} b_x + 3 a_x b_{xx} + a b_{xxx}$
$d = \log(c)$	$c^{-1} c_x$	$c^{-1} c_y$	$-c^{-2} c_x^2 + c^{-1} c_{xx}$	$2 c^{-3} c_x^3 - 3 c^{-2} c_x c_{xx} + c^{-1} c_{xxx}$
$z = a + d$	$a_x + d_x$	$a_y + d_y$	$a_{xx} + d_{xx}$	$a_{xxx} + d_{xxx}$

The first column of Table 3.1 constitutes the function list for the function (8); it sequentially evaluates the function value $z = x + \log(xy)$ at the given domain point (x, y). The remaining 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 independent 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 $(p, p_x, p_y, p_{xx}, p_{xxx})$, using the arrays obtained from previous row calculations as inputs. The final row yields the desired evaluations $(z, z_x, z_y, z_{xx}, z_{xxx})$.[8]

The elements in each of the rows in Table 3.1 can be numerically evaluated by means of sequential calls to Feed calculus subroutines. These evaluations are exact up to round-off and truncation error. For expositional simplicity, Table 3.1 only depicts evaluations for partial derivatives through order three. However, Feed calculus subroutines can in principle be constructed to evaluate the function value and the distinct partial derivatives through order k of any real-valued multi-variable function that can be sequentially evaluated in a finite number of steps by means of the two-variable functions

$$w = u + v, \quad w = u - v, \quad w = uv, \quad w = u/v, \quad w = u^v \tag{9}$$

and arbitrary nonlinear one-variable kth-order differentiable functions such as

$$\cos(u), \quad \sin(u), \quad \exp(u), \quad c^u, \quad \log(u), \quad \text{and} \quad au^b + c \tag{10}$$

for arbitary constants a, b, and c. Systematic rules for constructing general kth-order calculus subroutines for special functions such as (10) are derived in [13]. References to other work focusing

[8] The limitation to this collection of partial derivative evaluations is for expositional simplicity only. The evaluation of any additional desired partial derivative of z, say z_{xyy} or z_{xxxy}, can be obtained in a similar manner by suitably augmenting Table 3.1 with an additional column of algebraic operations.

on recurrence relations for the derivatives of special functions such as (10) can be found, for example, in [14]. A detailed discussion of the library of Feed calculus subroutines currently incorporated into Nasa is given in [11].

The Feed algorithm thus envisions the successive transformation of arrays of partial derivatives through any specified order k into similarly-configured arrays as one forward sweep is taken through the function list for a specified kth-order differentiable function. A similar approach is proposed in [14] and [17, page 280]. In contrast, the partial derivative evaluation methods proposed in [16, Chapter VI, pages 91-111] and [21] have a tree structure; that is, gradient operations are used to generate evaluations for each successively higher-order collection of partial derivatives using the results of previous gradient operations as inputs. Another approach that has attracted a great deal of interest is reverse-mode differentiation; see [3] and [18].

4 Automatic Initialization via Adaptive Homotopy Continuation

The initial conditions needed to integrate the complete differential system (4)-(6) from a given initial parameter point α^* consist of a solution vector $x(\alpha^*)$ together with evaluations for the adjoint $A(\alpha^*)$ and determinant $\delta(\alpha^*)$ of the Jacobian matrix $\psi_x(x(\alpha^*), \alpha^*)$. For many nonlinear problems, finding an initial solution vector is a difficult matter in and of itself.

Nasa incorporates an adaptive homotopy method [12] for automating these needed initializations. A standard (linear) homotopy method applied to the problem of finding a solution x^* for a system of equations $0 = F(x)$ proceeds by introducing a homotopy of the form

$$0 \;=\; tF(x) + [1-t][x-c] \tag{11}$$

and solving for x as a function of t as t varies from 0 to 1 along the real line, where c represents any initial guess for the solution vector x^*. In contrast, an *adaptive homotopy* is a homotopy for which the usual continuation parameter t varying from 0 to 1 on the real line is replaced by an adaptive continuation "agent" that makes its way by trial and error from $0 + 0i$ to $1 + 0i$ in the complex plane in accordance with certain stated objectives.

Specifically, the continuation agent designed in [12] adaptively selects a path of β values from $0 + 0i$ to $1 + 0i$ in the complex plane for the homotopy

$$0 \;=\; [F(x) - F(c)] + \beta F(c)\ , \tag{12}$$

where c again represents any initial guess for the solution vector x^*. The path for β is selected in accordance with the following multiple objective optimization problem: Reach the point $1 + 0i$ starting from the point $0 + 0i$ by taking as few steps as possible along a spider-web (spoke/hub)

grid centered at $1 + 0i$ in the complex plane, but do so in a way that avoids regions where the Jacobian matrix becomes ill-conditioned.

The adaptive homotopy method introduced in [12] and incorporated into Nasa is thus an example of what might more generally be called an *adaptive computational method*, i.e., a computational method that embodies the following principle important for applied researchers: Let the computational algorithm adapt to the physical problem at hand instead of requiring users to reformulate their physical problems to conform to algorithmic requirements. For sufficiently smooth functions $F(\cdot)$, a properly constructed homotopy—e.g., a probability one homotopy as formulated in [19]—is theoretically guaranteed to have no singular points along the real continuation path from 0 to 1 for almost all initial starting points c. However, successful implementation of such homotopy methods can require a mathematically sophisticated reformulation of the user's original problem.

The homotopy (12) is solved for x as a function of β as β varies from $0 + 0i$ to $1 + 0i$ in the complex plane by making use of a complete system of ordinary differential equations analogous to the system set out in Section 1. At each β point one obtains a solution vector $x^*(\beta)$ together with evaluations $A^*(\beta)$ and $\delta^*(\beta)$ for the adjoint and determinant of the homotopy Jacobian matrix[9] $J^*(\beta) = F_x(x^*(\beta))$. In principle, the solution vector $x^*(1 + 0i)$ obtained for (12) at $\beta = 1 + 0i$ yields a solution vector for the original system of interest, $0 = F(x)$. In particular, letting $F(x) = \psi(x, \alpha^*)$, one obtains complete initial conditions for the original problem of interest, the nonlocal sensitivity analysis of the system $0 = \psi(x, \alpha)$ over an interval of α values starting at α^*.

References

[1] Berz, M., Bischof, C., Corliss, G., and Griewank, G., *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, Philadelphia, Pennsylvania, 1996.

[2] Dakhlia, S., "Testing for a Unique Equilibrium in Applied General Equilibrium Models," *Journal of Economic Dynamics and Control* Volume 23, 1999, 1281–1297.

[3] Griewank, A., "On automatic differentiation," pp. 83–108 in *Mathematical Programming: Recent Developments and Applications*, Iri, M., and K. Tanabe (eds.), Kluwer Academic Publishers, 1989.

[4] Griewank, A., and Corliss, G., *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, Philadelphia, 1991.

[9] Note that the homotopy Jacobian matrix coincides with the Jacobian matrix for the original function of interest $F(\cdot)$, implying that singularities are not artificially induced into the problem by the homotopy method per se.

[5] Hayes, K., Hirschberg, J., and Slottje, D., "Computer algebra: Symbolic and algebraic computation in economic/econometric applications," *Advances in Econometrics*, Volume 6, JAI Press, Inc., 1987, 51–89.

[6] Heck, A., *Introduction to Maple*, Springer-Verlag, New York, 1993.

[7] Jerrell, M. E., "Automatic differentiation and interval arithmetic for estimation of disequilibrium models," *Computational Economics* 10 (August 1997), 295–316.

[8] Kagiwada, H., Kalaba, R., Rasakhoo, N., and Spingarn, K., *Numerical Derivatives and Nonlinear Analysis*, Plenum, New York, N.Y., 1985.

[9] Kalaba, R., Zagustin, E., Holbrow, W., and Huss, R., "A modification of Davidenko's method for nonlinear systems," *Computers and Mathematics with Applications* 3 (1977), 315–319.

[10] Kalaba, R., and Tesfatsion, L., "Complete comparative static differential equations," *Nonlinear Analysis: Theory, Methods, and Applications* 5 (1981), 821–833.

[11] Kalaba, R., and Tesfatsion, L., "Nonlocal automated sensitivity analysis," *Computers and Mathematics With Applications* 20 (1990), 53–65.

[12] Kalaba, R., and Tesfatsion, L., "Solving nonlinear equations by adaptive homotopy continuation," *Applied Mathematics and Computation* 41 (1991), 99–115.

[13] Kalaba, R., Tesfatsion, L., and Wang, J.-L., "A finite algorithm for the exact evaluation of higher-order partial derivatives of functions of many variables," *Journal of Mathematical Analysis and Applications* 92 (1983), 552–563.

[14] Kedem, G., "Automatic differentiation of computer programs," *ACM Transactions on Mathematical Software* 6 (June 1980), 150–165.

[15] Moore, R. E., "Interval arithmetic and automatic error analysis in digital computing," Ph.D. Thesis, Department of Computer Science, Stanford University, 1962.

[16] Rall, L., *Automatic Differentiation: Techniques and Applications*, Springer-Verlag, N.Y., 1981.

[17] Rall, L., "The arithmetic of differentiation," *Mathematics Magazine* 59 (1986), 275–282.

[18] Van Iwaarden, R., "Automatic differentiation applied to unconstrained nonlinear optimization with result verification," *Interval Computations* 4 (1993), 30–41.

[19] Watson, L. T., Sosonkina, M., Melville, R. C. Morgan, A. P., and Walker, H. F., "HOMPACK90: A suite of FORTRAN 90 codes for globally convergent homotopy algorithms," *ACM Trans. Math. Software*, Volume 23, 1997, 514–549.

[20] Wengert, R., "A simple automatic derivative evaluation program," *Communications of the ACM* 7 (1964), 463–464.

[21] Wexler, A., "An algorithm for exact evaluation of multivariate functions and their derivatives to any order," *Computational Statistics and Data Analysis* 6 (1988), 1–6.