

"MINIMUM COST DIFFERENTIATION METHODS AND THEIR USES
IN UNDERSTANDING, DESIGNING AND OPTIMIZING COMPLEX SYSTEMS"

by Paul J. Werbos, Quality Assurance Division, 3/20/82

CONTENTS

ABSTRACT	1
(I) INTRODUCTION	1
(II) OVERVIEW OF DIFFERENTIATION METHODS TO BE GIVEN	3
(III) SENSITIVITY ANALYSIS METHODS FOR DYNAMIC MODELS	9
(IV) SUMMARY OF DIFFERENTIATION METHODS	12
Methods to Compute First Order Derivatives	12
Methods to Compute Second Order Derivatives (Two Initial Values)	13
Methods to Compute Second Order Derivatives (Variable-Parameter)	15
(V) DERIVATION OF BACKWARDS SWEEPS: DYNAMIC FEEDBACK	18
Basics of Dynamic Feedback	18
Second-Order Backwards Sweeps	19
Extensions and Applications	20
(VI) DERIVATION OF METHODS FROM PERTURBATION METHODS OF PHYSICS	22
(VII) APPLICATION TO STOCHASTIC OPTIMIZATION AND INTELLIGENT SYSTEMS	25
GDHP: An Approach to Optimizing Complex, Nonlinear Stochastic Systems	26
Application of Differentiation Methods	29
Applications to Modelling, Economics and Artificial Intelligence	35
Parallels to the Human Brain	38
Major Features of the Brain and Parallels to GDHP	38
Major Apparent Discrepancies	42
REFERENCES	45
APPENDIX A: THE THEOREM BEHIND DYNAMIC FEEDBACK	48

"MINIMUM COST DIFFERENTIATION METHODS AND THEIR USES
IN UNDERSTANDING, DESIGNING AND OPTIMIZING COMPLEX SYSTEMS"

by Paul J. Werbos, Quality Assurance Division, 3/20/82

ABSTRACT

This paper derives low-cost methods for computing "ordered derivatives," which are required in numerous applications. Applications to statistical estimation, optimization and equation solving are pointed out. Applications to the sensitivity analysis of nonlinear dynamic models are discussed in detail. To illustrate the possibilities for adapting these methods to deal with large "network" systems, it is shown how to compute the derivatives required by a previously proposed method for decision-making over time under uncertainty. This example leads to an approach to the design of generalized, adaptive decision-making systems; applications to artificial intelligence and model evaluation concepts are discussed, along with structural analogies to the human brain.

(I) INTRODUCTION

It is well known that the derivative and partial derivative of elementary calculus have applications throughout the natural and social sciences. This paper is concerned with a related mathematical concept, the "ordered derivative," which also has many applications, but which goes by different names in different fields of study. The purpose of this paper is two-fold: to discuss the applications of ordered derivatives, and to show how they may be calculated at minimum cost either directly or as part of a "network design." Roughly speaking, it will be shown that all of the first or second derivatives needed for most applications can be obtained exactly for a cost comparable to that of exercising the original model or system itself.

Section (II), below, will explain what an ordered derivative is, and summarize the advantages of the alternative methods to be described. It will also mention applications to statistical estimation, deterministic optimization and certain equation-solving techniques, of particular importance to large systems. Related methods and literature will be mentioned. Section (III) will discuss "sensitivity analysis," which attempts to help one understand complex systems such as energy models by pinpointing the key inputs and providing related information. Section (IV) will list the equations for the methods discussed in section (II), for the case of a simple nonlinear dynamic system. Section (V) will derive and generalize the "backwards" methods of section (IV), by using a concept called "dynamic feedback." Section (VI) will use perturbation approaches taken from physics to derive the remaining methods.

Section (VII) will illustrate how these methods may be adapted to handle very difficult problems, by spelling out the calculations required to implement a method for dynamic optimization under uncertainty, while taking full advantage of the power of "parallel" or "vector" computers. It will also show that a combination of this and related work may ultimately lead to a generalized, adaptive artificial intelligence, without some of the limitations of those now being developed; structural analogies between this design and the human brain will be discussed. In the mathematical framework provided by this section, model development may be analyzed as one of several analysis activities required to support rational decision making.

(II) OVERVIEW OF DIFFERENTIATION METHODS TO BE GIVEN

Figure 1 shows a simple example of the kind of "derivative" we are trying to compute. Suppose that we have a nonlinear system, with a vector \underline{x} of N endogenous variables and a vector \underline{u} of exogenous variables. Suppose that the system is governed by the equation shown in Figure 1. The cost of simulating the model over the whole time range is mNT , because in each of the T time periods we compute a forecast for each of the N variables in \underline{x} , and each such forecast involves m terms. Please note that N is often much larger than m . Given a small change in the variable x_i in time period 0, we want to know how large the resulting change in x_i is in the final time period T .

$$\underline{x}(t+1) = \underline{f}(\underline{x}(t), \underline{u}(t))$$

- N components of \underline{x}
- m terms per equation f_i
- T time periods ($t = 0$ to $T-1$)
- cost of simulation = mNT
- not a "simultaneous" (implicit) model

$$\frac{\partial^+ x_i(T)}{\partial x_j(0)}$$

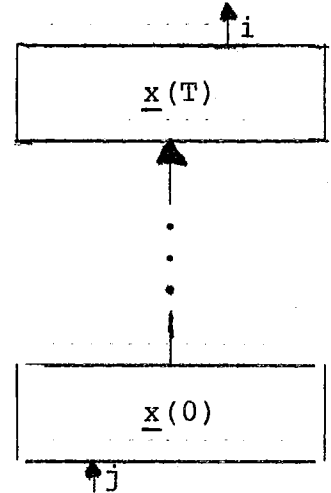


Figure 1: A Simple Example

The change in $x_i(T)$ per change in $x_j(0)$, holding the rest of $\underline{x}(0)$ constant, is a fundamental quantity of the system. It goes by many different names. In modelling, it is often called a "sensitivity coefficient." In economics, it is traditionally called an "impact multiplier." Electrical engineers often call it a "transient response," or "constrained derivative." Nuclear engineers sometimes use the term "adjoint." Here we will call it an "ordered derivative," using the notation shown in Figure 1, for two reasons: (1) the notation is somewhat more explicit than what is usually used; and (2) the concept of ordered derivative is somewhat more general rigorous, as will be seen. Some of the methods to be discussed below were published independently by control engineers⁽¹⁾ and by nuclear engineers⁽²⁾ at about the same time (mid 1970's) as the ordered derivative concept was developed, and related methods for linear models were available even earlier. For differential equation systems (which this paper does not address) the nuclear engineering literature is perhaps the most extensive at present⁽³⁾.

Well-known applications which require the use of such first-order derivatives are sensitivity analysis, maximization of a system result (i.e., "deterministic optimization"), and statistical estimation. In the last two cases, one actually is concerned with the derivative of a function of $\underline{x}(T)$ or of $\underline{x}(t < T)$ rather than the derivatives of $x_j(T)$ for some j , but it is easy to make this extension of the methods; for example, the function to be differentiated or a running total for it may be added to the list of system variables.

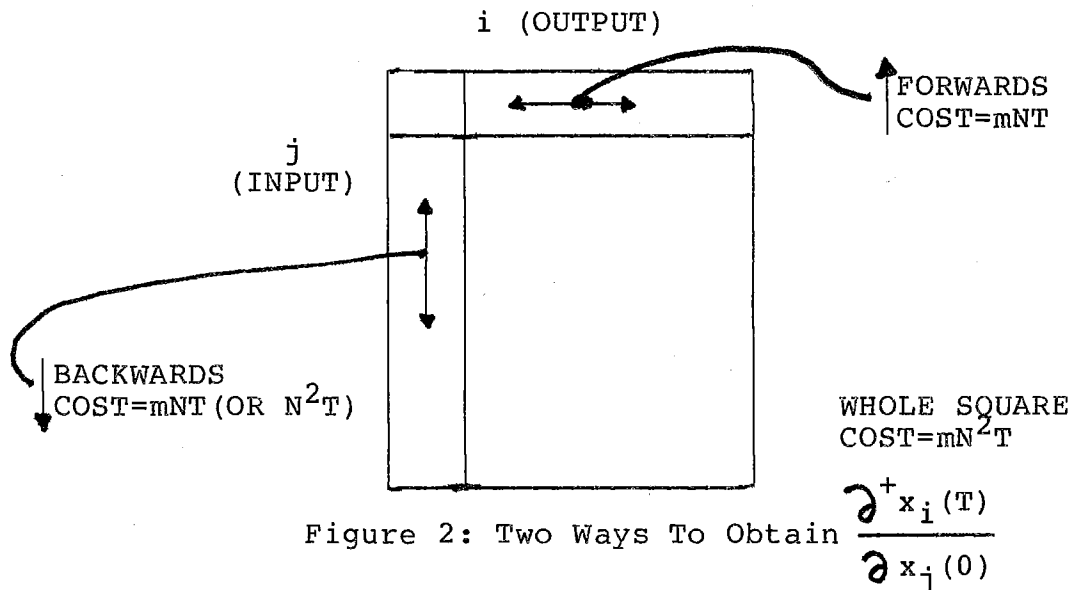


Figure 2 describes two methods for computing ordered derivatives exactly in the example above. The corresponding equations are:

$$\begin{aligned}
 \uparrow : \quad \underline{z}(t) &= \frac{\partial^+ \underline{x}(t)}{\partial x_j(0)} ; \quad \underline{z}(t+1) = F(t)\underline{z}(t) \\
 \downarrow : \quad \underline{z}'(t) &= \frac{\partial^+ x_i(T)}{\partial \underline{x}(t+1)} ; \quad \underline{z}'(t) = F^T(t)\underline{z}'(t+1), \text{ (or transpose)}
 \end{aligned}
 \tag{2.1}$$

where $F(t)$ is the matrix of derivatives of $\underline{f}(\underline{x}(t))$. The large square in Figure 2 represents the entire matrix of ordered derivatives of all $x_i(T)$ with respect to all $x_j(0)$. The conventional or "forwards" method (indicated by an arrow pointing upwards) is based on perturbing one of the initial values $x_j(0)$, and observing the impact on all the final results, i.e., on the vector $\underline{x}(T)$. Each time we apply this method, we perturb only one of the initial values; thus we obtain only one row of the matrix of ordered derivatives, as shown in Figure 2. This costs us mNT calculations, as shown. Often the initial value $x_j(0)$ is actually changed, and the

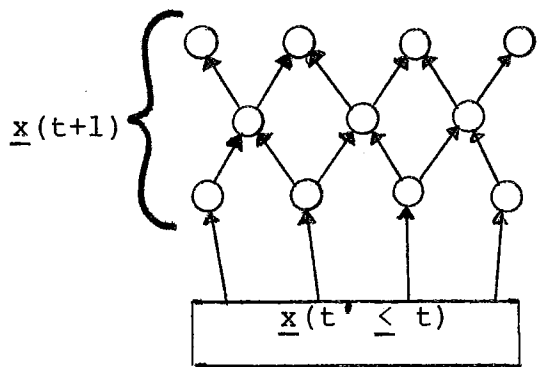
model resimulated. (This costs mNT operations, as did the original run of the model.) However, this leads to problems with the numerical accuracy of the results, because it requires that one compute each derivative by subtracting two numbers very close to each other in size. The forwards closed-form Jacobian formula, shown at the bottom of Figure 2, has the same cost but is more accurate.

The backwards method, shown with a downwards pointing arrow in Figure 2, computes an entire column of the matrix, using only mNT calculations. In control engineering, this sort of method has been used and related to the concept of "constrained derivatives," but has not been applied more generally⁽¹⁾.

The key point about these methods is that the forwards method is often used when the backwards method would be more appropriate. This can multiply costs (by a factor of N) to the point where it becomes infeasible to do what one wants to do. For example, it has long been known that economic data, like engineering measurements, are fraught with many errors, and that these errors invalidate conventional estimation methods. Statisticians^(4,5) observed years ago that white noise converts a simple econometric model (like our example, but linear) into a "vector mixed autoregressive moving average process." In other words, one can account for such errors in data by estimating the corresponding vector ARMA process. However, because of the sheer cost of such estimation, it has rarely been done in economics. Instead, an approximation suggested by Hibbs has become popular of late: a conventional model is estimated by regression, and then simple univariate ARMA ("Box-Jenkins"⁽⁵⁾) modeling is used on the residuals, and the process may be iterated. Yet in statistical estimation, one only needs a single column of the derivative matrix (i.e., the derivatives of L), not the whole matrix; using the backwards method, one can compute all the derivatives needed in an iteration at the cost of only mNT , which is what it takes to exercise the model. This method was applied to vector ARMA estimation in the early 1970's, and inserted into a user-oriented software package at MIT⁽⁶⁾ (TSP), but has yet to receive wide application in economics. It now appears that vector ARMA estimation (and thus Kalman filtering estimation, which is formally equivalent to it) may have less value in social science than other more robust methods based on a generalization of Hartley's simulation path approach^(6,7,8); however, those methods, too, require a set of derivatives, as part of minimizing a complicated loss function.

Likewise, in sensitivity analysis, a user often wants to know the sensitivity of a few key results to all the initial values, or to be sure he knows the largest of these sensitivity coefficients. Again, only a few columns of the matrix are required; it is wasteful to pay for the whole matrix.

With large models or network systems, N may range from the hundreds to the millions or more. Thus cutting the cost of computing derivatives by a factor of N is often crucial to feasibility. One may be sure that the cost of exercising the system (mNT) is affordable, or the system would be of no interest; more than this, by a multiple of N , may be unacceptably expensive.



"CHAIN RULE" (DYNAMIC FEEDBACK):

$$\frac{\partial^+ x_i}{\partial x_j} = \sum_{k=j+1}^i \frac{\partial^+ x_i}{\partial x_k} \cdot \frac{\partial f_k}{\partial x_j} \quad i > j$$

CONVENTIONAL PERTURBATION:

$$\frac{\partial^+ x_i}{\partial x_j} = \sum_{k=j}^{i-1} \frac{\partial f_i}{\partial x_k} \cdot \frac{\partial^+ x_k}{\partial x_j} \quad i > j$$

Figure 3: A More General Example: $\underline{x}(t+1) = f(\underline{x}(\text{all } t' \leq t+1), \underline{u}(\text{all } t'))$

The principle of "dynamic feedback" permits one to deal with more general, "network" models such as the one shown in Figure 3. Multisector models, for example, are usually best represented as a network. Because the proof of the "chain rule" for ordered derivatives⁽⁶⁾ is not generally available, it is reproduced in Appendix A.

In Figure 3, the endogenous variables may appear with any nonnegative lag, including zero. However, we still assume here that the model has been reduced to "explicit" form. (In economics, one would call this a recursive model; in mathematics, one calls it a nonrecursive system.) We assume that the functions f_i , which make up \underline{f} , can be ordered in such a way that we can use them one by one to calculate the vector $\underline{x}(t+1)$. Actually, one can apply the methods given in this paper to simultaneous equation models as well, by using substitutions to be described in a forthcoming report from EIA⁽⁹⁾.

Figure 3 illustrates an example where $\underline{x}(t+1)$ has eleven components, each represented by a circle; the arrows flowing into a circle represent inputs required to compute that component of \underline{x} .

The forwards and backwards methods are generalized as shown in Figure 3. The subscripts here refer to an ordered index of all time/variable-number combinations; the formulas are given in more conventional form in the main paper. The key thing to note is that there are only m calculations per time/variable combination. Thus we still only need to make mNT calculations to get a complete row or column of ordered derivatives, as in our earlier example. This has

not previously been published. With conventional matrix methods for constrained derivatives, based on our earlier example, one would have to use N by N matrices f' , which would not usually be sparse; thus the generalization here makes it feasible to differentiate large network systems which would have been too expensive to differentiate with conventional methods.

The methods shown in Figure 3 remain efficient even if one uses "parallel" computers. Parallel computers - based on many processors operating in parallel rather than one CPU - are becoming increasingly common⁽¹⁰⁾. With a conventional computer, it would take roughly 11 calculation times to compute $x(t+1)$ in our example (1 for each component of x). With a parallel computer, it need only take 3: in the first period, 4 processors would calculate the lower tier in parallel, since none of the 4 lower components depends on the others; in the second period, the middle tier would be calculated; etc. The backwards method shown here allows similar economies: one can calculate ordered derivatives of a model result with respect to the top tier in the first period of calculation, then to the middle tier, and then to the bottom tier. The forwards method is similar.

Large scale models or systems typically can be represented as relatively sparse networks, as in this example. Actual physical networks, made up of units operating in parallel, have a similar structure. To optimize such a system (except in unusual special cases) it is essential to know the derivatives of the desired performance measure with respect to all parameters in the system; for this to be feasible, it is essential to use a method such as the generalized backwards method which does not multiply the cost of getting the derivatives far beyond the cost of exercising the system. This reasoning also applies to the problem of minimizing or maximizing a complicated function, where most of the computing time is usually taken up with calculating derivatives⁽¹¹⁾; it also applies to those methods for solving large systems of equations which require less than a full matrix of derivatives⁽¹²⁾.

This overview has discussed derivatives with respect to initial values of the variables only; however, section (IV) will consider parameters, and the case of exogenous variables is a trivial extension of the endogenous variable case⁽⁹⁾. To avoid making a complicated discussion even more complicated, section (IV) will only mention our earlier example when discussing second derivatives; however, it is trivial to substitute the general formulas in Figure 3 for those in Figure 2, whenever they apply in the second derivative calculation, to arrive at more general methods. Section (VII) on stochastic optimization will provide a partial example of these possibilities.

