Techniques

C. L. McCARTY, JR., Editor

A Simple Automatic Derivative Evaluation Program

R. E. Wengert
General Electric Company, Syracuse, New York*

A procedure for automatic evaluation of total/partial derivatives of arbitrary algebraic functions is presented. The technique permits computation of numerical values of derivatives without developing analytical expressions for the derivatives. The key to the method is the decomposition of the given function, by introduction of intermediate variables, into a series of elementary functional steps. A library of elementary function subroutines is provided for the automatic evaluation and differentiation of these new variables. The final step in this process produces the desired function's derivative.

The main feature of this approach is its simplicity. It can be used as a quick-reaction tool where the derivation of analytical derivatives is laborious and also as a debugging tool for programs which contain derivatives.

Related approaches develop analytical expressions for total or partial derivatives of arbitrary algebraic functions through application of rather elaborate scanning procedures on the entire function. The technique reported here, instead, generates numerical values of derivatives and is made simple by inputting the given complex function as a series of elementary function evaluations.

Proposed Technique

Total Derivatives. To demonstrate the technique for obtaining total derivatives, consider the following example. Compute $\dot{f}$, where

$$f = \frac{x_1}{x_2^2 x_3}.$$

Numerical values for x_1, x_2, x_3, $\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$ are given.

The total derivative $\dot{f}$ is evaluated indirectly. Only the function itself is explicitly programmed. The calculation of the given "complex" expression is decomposed, by introduction of intermediate variables, into a string of elementary functional steps using a predeveloped subroutine library. These subroutines, examples of which may be found in the Appendix, automatically provide derivatives for the intermediate variables. As the computation proceeds, the desired derivative emerges as a by-product of the function evaluation. In the given example, decomposition might proceed as follows.

First call the exponentiation subroutine to evaluate the elementary function $z_1 = x_2^2$ and its derivative $\dot{z}_1 = 2x_2\dot{x}_2$.

Next call the product subroutine to evaluate $z_2 = z_1 x_3$ and its derivative $\dot{z}_2 = z_1\dot{x}_3 + \dot{z}_1 x_3$. Note this uses the previously computed results z_1, $\dot{z}_1$.

Finally, call the division subroutine to evaluate $f = \dfrac{x_1}{z_2}$ and its derivative $\dot{f} = \dfrac{z_2\dot{x}_1 - \dot{z}_2 x_1}{z_2^2}$. This directly uses the previously computed results z_2, $\dot{z}_2$, and implicitly z_1 and $\dot{z}_1$.

The same procedure is used for any function, no matter how complex. Note that we do not attempt to directly evaluate the derivative of the complex function. Instead we proceed in a sequential fashion, evaluating derivatives of elementary functions. The end of the sequence is the desired derivative of the original complex function.

Higher order total derivatives are treated in exactly the same manner. It is only necessary to have library subroutines for evaluating higher order derivatives of the elementary functions.

Partial Derivatives. The proposed method may also be used to compute partial derivatives.

By the chain rule of differentiation, if

$$f = f(x_1, x_2, \cdots, x_n),$$

then f can be expressed as

$$\dot{f} = \frac{\partial f}{\partial x_1}\dot{x}_1 + \frac{\partial f}{\partial x_2}\dot{x}_2 + \cdots + \frac{\partial f}{\partial x_n}\dot{x}_n.$$

By computing, as before, the total derivative $\dot{f}$, but with the input derivatives changed to $\dot{x}_i = 1$,

$$\dot{x}_j = 0, \qquad j \neq i,$$

we will, in effect, have computed $\partial f/\partial x_i$.

Hence a partial derivative subroutine can be constructed to act as a control routine which appropriately sets the input derivatives to zero or one. For each set of input derivatives (one of which is unity and the others are zero), the function subroutine outputs assume the value of the function and one of its partial derivatives.

Higher Order Partial Derivatives. To obtain higher order partial derivatives of the given function, first examine the functional form of higher order total derivatives. To do this, we start by restating the chain rule in a

* Radio Guidance Operation.

more convenient notation, which is used hereafter.

$$f = f(x_1, x_2, \cdots, x_n),$$
$$\dot{f} = f_i \dot{x}_i,$$

where $f_i = \partial f/\partial x_i$, and the subscript-summation convention is used, "products are to be summed over repeated subscripts;" e.g.,

$$f_{ijk} x_i \dot{x}_j = \sum_{ij} f_{ijk} x_i \dot{x}_j .$$

Successive time differentiation then yields the higher order time derivatives:

$$\dot{f} = f_i \dot{x}_i$$
$$\ddot{f} = f_i \ddot{x}_i + f_{ij} \dot{x}_i \dot{x}_j$$
$$\dddot{f} = f_i \dddot{x}_i + 3 f_{ij} \ddot{x}_i \dot{x}_j + f_{ijk} \dot{x}_i \dot{x}_j \dot{x}_k$$

where

$$f_{ijk} = \partial^3 f/\partial x_i \partial x_j \partial x_k .$$

By evaluating $\dot{f}$, $\ddot{f}$, and $\dddot{f}$, with appropriate values of the input derivatives, and equating to the corresponding chain-rule derivative expressions, a sufficient number of independent linear equations can be generated to enable solution for the desired higher order partial derivatives. For example, some alternate solutions for f_{pq} $(p \neq q)$ are:

$$2f_{pq} = \ddot{f}\,|_{\dot{x}_p = \dot{x}_q = 1} - \ddot{f}\,|_{\dot{x}_p = 1} - \ddot{f}\,|_{\dot{x}_q = 1}$$
$$4f_{pq} = \ddot{f}\,|_{\dot{x}_p = \dot{x}_q = 1} - \ddot{f}\,|_{\dot{x}_p = 1,\, \dot{x}_q = -1}$$
$$3f_{pq} = \dddot{f}\,|_{\ddot{x}_p = \dot{x}_q = 1} - \dddot{f}\,|_{\dot{x}_q = 1}$$

where all input derivatives ($\dot{x}$'s, $\ddot{x}$'s, $\dddot{x}$'s) not explicitly indicated to be unity are set to zero.

Partial Derivatives of Time Derivatives. Partial derivatives of time derivatives are readily obtained by combination of the partial derivatives, obtained as above, with the input time derivatives. For instance, suppose we wish to evaluate the first and second partials of the first and second time derivatives of the input function. Straightforward partial differentiation, starting with $\dot{f}$ and $\ddot{f}$ as given previously, yields the following relations for computing the desired partial derivatives.

	First Partials	*Second Partials*
First Time Derivatives	$\dot{f}_p = f_{ip} \dot{x}_i$	$\dot{f}_{pq} = f_{ipq} \dot{x}_i$
	$\dot{f}_{\dot{p}} = f_p$	$\dot{f}_{\dot{p}q} = \dot{f}_{p\dot{q}} = f_{pq}$
Second Time Derivatives	$\ddot{f}_p = f_{ip} \ddot{x}_i + f_{ijp} \dot{x}_i \dot{x}_j$	$\ddot{f}_{pq} = f_{ipq} \ddot{x}_i + f_{ijpq} \dot{x}_i \dot{x}_j$
	$\ddot{f}_{\dot{p}} = 2 f_{ip} \dot{x}_i$	$\ddot{f}_{\dot{p}q} = \ddot{f}_{p\dot{q}} = 2 f_{ipq} \dot{x}_i$
	$\ddot{f}_{\ddot{p}} = f_p$	$\ddot{f}_{\ddot{p}q} = \ddot{f}_{p\ddot{q}} = \frac{1}{2} \ddot{f}_{\dot{p}\dot{q}} = f_{pq}$

where unlisted partials are zero, and e.g., $\ddot{f}_{\ddot{p}q} = \partial^2 \ddot{f}/\partial \ddot{x}_p \partial x_q$.

Conclusion

Programs based upon this technique have been found to be both easy to prepare and easy to use. To differentiate even extremely complex functions, the user need write only a function evaluation subroutine calling sequence.

The main requirement is merely a subroutine library for elementary functions and frequently recurring composite functions.

Acknowledgments. The author appreciates the efforts of Messrs. C. Arabadjis, C. Conover, R. Fagan and R. Wilkins in crystallizing the original concept.

APPENDIX. Example Subroutines

This appendix contains sample Fortran codings of subroutines for: (1) three elementary functions, (2) a sample problem using these elementary functions, (3) a partial derivative generator for the sample problem. Note that "(1)" indicates function value, and "(2)" indicates derivative value.

A main frame I/O routine would also be required, but is omitted for simplicity.

Elementary Functions

```
SUBROUTINE ADD (U, V, W)
DIMENSION U(2), V(2), W(2)
W(1) = U(1) + V(1)
W(2) = U(2) + V(2)
RETURN
END
```

```
SUBROUTINE SINE (U, W)
DIMENSION U(2), W(2)
W(1) = SINF (U(1))
W(2) = U(2) * COSF (U(1))
RETURN
END
```

```
SUBROUTINE PROD (U, V, W)
DIMENSION U(2), V(2), W(2)
W(1) = U(1) * V(1)
W(2) = U(2) * V(1) + U(1) * V(2)
RETURN
END
```

Sample Problem

Find $\dot{y}$ for $y = x_1 x_2 + \sin(x_2 x_3)$.

```
SUBROUTINE FUN (X1, X2, X3, Y)
DIMENSION X1(2), X2(2),
  X3(2), Y(2), Z1(2), Z2(2), Z3(2)
CALL PROD (X1, X2, Z1)
CALL PROD (X2, X3, Z2)
CALL SINE (Z2, Z3)
CALL ADD (Z1, Z3, Y)
RETURN
END
```

Partial Derivative Generator

```
SUBROUTINE PART (X1, X2, X3, F, P)
DIMENSION X1(2), X2(2), X3(2), F(2), P(3)
CALL FUN1 (X1, X2, X3, F)                Evaluate f
X1(2) = 1.0
X2(2) = 0.0
X3(2) = 0.0
CALL FUN1 (X1, X2, X3, F)                Compute ∂f/∂x1
P(1) = F(2)
X1(2) = 0.0
X2(2) = 1.0
CALL FUN1 (X1, X2, X3, F)                Compute ∂f/∂x2
P(2) = F(2)
X2(2) = 0.0
X3(2) = 1.0
CALL FUN1 (X1, X2, X3, F)                Compute ∂f/∂x3
P(3) = F(2)
X3(2) = 0.0
RETURN
END
```

Received February, 1964