

CHAPTER I.

INTRODUCTION

There have been two divergent developments for the evaluation of the derivatives of analytic functions. One is based on the manipulation of the function formula by classical differential calculus techniques. The results are formulae of ever increasing lengths. The other method is based on the replacement of the differentiation operation by simple sums and multiplications. The results are constants that represent the value of derivatives at the point of evaluation. Unlike the classical method, the second method yields only values of the derivatives at a specified point of the independent variable. This limitation can be overcome by analytic continuation -- the evaluation of the derivatives at a neighbor point by repeating the entire process. This book is concerned with the second method. We shall begin by discussing the classical methods.

Recent results have been published for the analytical manipulation of functional formulae to find the derivatives. Hanson, et al(1), developed some symbol manipulation algorithms for the analytical differentiation of complicated functions. The method involves scanning the function formula to be differentiated and produces an algebraic expression for its derivative. One obtains a numerical value for the derivative by including the resultant algebraic expression in one's computer program. Since the resultant algebraic expression is progressively longer and more involved for higher-order derivatives, this method of differentiation has not seen much use. Wendert(2) discussed an automatic numerical evaluation of derivatives that avoids this problem. Given any algebraic expression for the function, the Wendert method will automatically yield the numerical value of the derivative of the function at a specified point. The key to the method is the decomposition of the given function into a sequence of elementary functional steps. A library of elementary function derivatives is then accessed and used in the numerical evaluation of the derivative of the complicated function. Applications of the Wendert method have been reported(3-5), which showed promise for this numerical technique. However, its scope is limited; it is not fully automatic in the differentiation of functions.

Both the Hanson and the Wengert methods are semi-automatic in that, given an analytic function, these methods will produce only the first derivative of the input function. Hanson's method produces an algebraic expression, and the Wengert method produces a numerical value. In the Hanson method, the output can be fed back as an input; thus generating a higher-order derivative. There is however a limit to what one can do with this method since the output expression will very quickly build up to more complicated expressions that may become too long to be stored or written in a reasonable amount of space. In Wengert's method, the output cannot be used as an input for the next derivative. To obtain a higher-order derivative, one must have the algebraic expression for the previous derivative. These methods cannot produce an unending sequence of constants representing the derivatives in a Taylor series. A fully automatic differentiation of analytic functions is capable of yielding the numerical values of all the higher-order derivatives in an unending sequence. The Automatic Taylor Series (ATS) method is based on a fully automatic differentiation technique.

The foundation of the Automatic Taylor Series (ATS) method is the reduction of differentiation of analytic functions to simple arithmetic operations. First publication on this simplification was by Wilson(6), who showed that the solution of certain simple linear differential equations can be obtained by a continuous series of multiplications and additions. His work, published in 1949, did not receive much notice because it was concerned with a small class of problems, where its full potentials were not apparent to the casual reader. In 1960, Gibbons(7) obtained the recursive relations for the fully automatic differentiation of some simple functions, and he indicated the possibility for the automatic differentiation of more complicated functions. Somehow, the full significance of Gibbons' work has not been recognized, and development of his method for the solution of complicated problems was not carried out. There are only two references, Fox(8) and Moore(9), to this inherently powerful analysis technique. Our discussion in Chapter II parallels this work of Gibbons but differs in the manner of bookkeeping. Richtmyer(10) and Kitelman(11) have worked on the automatic differentiation of complicated functions. Leavitt(12) has published a summary on the power series method, and Barton, et al.(13) have reported on the automatic solution of ordinary differential equations.

In this book, we shall discuss the ultimate possibility: automatic differentiation of complicated functions including functions of two and three variables, automatic solution of ordinary differential equations, and solution of linear and nonlinear partial differential equations. Our interest in the automatic differentiation of analytic functions arose in 1962 from a dissatisfaction with the then standard numerical method for the solution of ordinary differential equations such as the Runge-Kutta method or a predictor-corrector method. The drawback in the old methods is the lack of adequate control of the accuracy of the results, the local error due to truncation. While the large-scale digital computer can handle 14-figure decimal numbers matter-of-factly, results from standard methods may lack even 4-figure accuracy. The reason for this lack of accuracy in standard numerical integration methods is the short lengths of the equivalent power series. For example, a fourth-order predictor-corrector method is equivalent to a Taylor series with only five terms.

The convergence property of short power series cannot be determined by any means. There is insufficient information in the few terms of a short power series for a reliable analysis of the truncation error. The short-series methods are forced into taking small incremental steps. This is because the short power series does not properly approximate the function at the point of expansion. The many small steps give rise to the propagation of machine roundoff errors. Thus, on a difficult problem the short-series methods are bound to produce results with accuracy much poorer than machine accuracy. While large steps give rise to truncation errors, small steps give rise to propagation of roundoff error.

The Automatic Taylor Series (ATS) method maintains control over the accuracy of the computations. When the order of the method is high, say about thirty, it is possible to find the exact positions of poles in the solution function. Norman(14) had used this to factor out the pole. It will be shown in Chapter III that there is a direct relationship between the behavior of Taylor series terms and the locations of singularities. A singularity is the main source of the truncation error. When a computation step taken is larger than an allowable fraction of the radius of convergence -- the distance to the nearest singularity -- there will be appreciable error. We will then reduce the step size.

The accuracy of the Automatic Taylor Series (ATS) method is totally dependent on the convergence analysis of the long series. Not only does a long series allow for good error analysis, it allows for an integration stepsize much larger than that for a typical fourth-order method. When the order of the Taylor series is sufficiently high, it is possible to calculate or estimate the radius of absolute convergence. This information allows the integration step to be adjusted to keep the local error below a specified limit. In the ATS method, solutions of problems are calculated only after obtaining both the calculation for the radius of convergence and adjusting the integration step. When a singularity is in or near the path of integration, the ATS method will adjust the integration step to be very small.

The ATS analysis can be applied to linear and nonlinear problems alike. The bases of the ATS analysis are the recursive relation obtainable from the equations of the problem, and the Leibnitz rule(15), also called a Cauchy product, for the derivative of a product of functions. The solutions to all problems, linear and nonlinear, are in the form of Taylor series. Linear problems in ordinary differential equations have simple Taylor series solutions that may be evaluated without computers. In Chapter VII, we will show that the series solution to linear ordinary differential equations can often be resolved by transcendental functions. Nonlinear problems and partial differential equations, when solved by the ATS method have solutions that require meticulous bookkeeping. The ATS method is a mathematical analytical tool in the case of linear problems, and it is a numerical method in the case of nonlinear problems.

Before going into the main subjects of this book, we will pause for a while to point out some of the conventions used in the notations and equations. We will use x as the independent variable in all functions of one variable. We will use x as the first independent variable and t as the second independent variable in all functions of two variables. We will use y as the first dependent variable and z as the second dependent variable. We will also use z instead of x when the independent variable is a complex quantity. Throughout this book, $f(x)$ is used to represent all possible functions of x . So, at times $f(x)$ may represent more than one function of x . When this occurs, we will write

the function with an equation number like this, $f(x;15)$, so as to remove ambiguity of expression. Then, the reader can refer to the equation and see the exact function under discussion.

We will use $g, u, v, w,$ and z as auxiliary functions where needed. We will use $A, B,$ and $C,$ as Taylor arrays in FORTRAN subroutines. We will use $a, b, c, d, e, f, g, h,$ and s as real or floating-point constants, and $i, j, k, l, m,$ and n as integer or fixed-point constants. These constants may appear either in lower-case or upper-case letters.

There is one word that will be over-used in this book. The word is "order". There is the order of a pole. There is the order of a derivative. There is the order of a Taylor series term. There is the order of a numerical integration method. We will attempt to keep order among all these orders, but it could be a rather tall order to do so.

A. Reduced Derivatives and Modified Leibnitz Rule.

The Taylor series expansion of a function, $f(x)$, at the point $x=a$ for the value of the function at $x=b$ is given by

$$f_b = f_a + f'_a * h + f''_a * h^2 / 2! + \dots + f^{(m)}_a * h^m / m! + \dots \quad [11]$$

where $h = (b-a)$, and the m superscript on f signifies the m -th derivative of $f(x)$ with respect to x . The subscripts a and b indicate evaluation of the function at $x=a$ and $x=b$. The ATS-array for the function $f(x)$ at $x=a$ with an increment of h is defined to be just the Taylor series given in Eq.[11]. The terms of the ATS-array for $f(x)$ are the terms of the Taylor series; also referred to as the reduced derivatives of $f(x)$. The reduced derivatives are defined with upper-case letters for the name of the function concerned.

$$F(m+1) := f^{(m)}_a * h^m / m! \quad [12]$$

In this work, all normal functions will be denoted by lower-case letters (or as they would appear normally), and the corresponding ATS terms or reduced derivatives will be denoted by upper-case letters in the name of

the functions. The need for $(m+1)$ in Eq.[2] is due to FORTRAN limitation, since an array may not begin at zero in FORTRAN. The Taylor series expansion for $f(x)$ in terms of the reduced derivatives is

$$f_b = F(1) + F(2) + F(3) + \dots = \sum_{i=1}^{\infty} F(i) \quad [3]$$

where i is defined as the index of the F -array elements. The index of the Taylor series element is one more than the order of the differentiation, as shown in Eq.[2].

The automatic sequential numerical evaluation of reduced derivatives for simple analytic functions had been discussed by Gibbons in 1968. We shall restate his results in Chapter II, and we shall extend the development to complicated functions, inverse functions, and two-dimensional functions. The foundation for the automatic sequential differentiation of functions is the Leibnitz rule for the differentiation of a product of functions. Consider the function $f(x) = v(x)*w(x)$. According to the Leibnitz rule, the m -th derivative is given by

$$\frac{d^m f}{dx^m} = \sum_{i=0}^m \frac{v^{(i)} w^{(m-i)}}{i! (m-i)!} \quad [4]$$

where $v^{(i)} := \frac{d^i v}{dx^i}$, and $w^{(m-i)} := \frac{d^{m-i} w}{dx^{m-i}}$.

With Eq.[2] in mind, we rearrange Eq.[4] to be

$$f^{(m)} * \frac{h^m}{m!} = \sum_{i=0}^m v^{(i)} * \frac{h^i}{i!} * w^{(m-i)} * \frac{h^{m-i}}{(m-i)!} \quad [5]$$

Then, substitution of Eq.[2] into Eq.[5] yields

$$F(m+1) = \sum_{i=0}^m U(i+1)*W(m-i+1) \quad [6]$$

Let $n:=m+1$ and $J:=i+1$. This yields the modified Leibnitz rule.

$$F(n) = \sum_{J=1}^n U(J)*W(n-J+1) \quad [7]$$

This is the foundation for the automatic sequential differentiation of complicated functions. Note that the modified Leibnitz rule, Eq.[7], does NOT contain FACTORIALS. This one fact simplifies all the developments to follow.

The automatic sequential differentiations of two-dimensional functions is needed in many of the more difficult types of problems. Consider a function $f(x,t)$ with Taylor expansion at $x=a$ with increment h and at $t=c$ with increment k . The Taylor expansion is as follows

$$\begin{aligned}
 f(a+h,c+k) = & f(a,c) + \frac{\partial f}{\partial x} h + \frac{\partial^2 f}{\partial x^2} \frac{h^2}{2!} + \frac{\partial^3 f}{\partial x^3} \frac{h^3}{3!} + \dots \\
 & + \frac{\partial f}{\partial t} k + \frac{\partial^2 f}{\partial x \partial t} h k + \frac{\partial^3 f}{\partial x^2 \partial t} \frac{h^2 k}{2!} + \dots \\
 & + \frac{\partial^2 f}{\partial t^2} \frac{k^2}{2!} + \frac{\partial^3 f}{\partial x \partial t^2} \frac{h k^2}{2!} + \frac{\partial^4 f}{\partial x^2 \partial t^2} \frac{h^2 k^2}{2!2!} + \dots \\
 & + \frac{\partial^3 f}{\partial t^3} \frac{k^3}{3!} + \frac{\partial^4 f}{\partial x \partial t^3} \frac{h k^3}{3!} + \frac{\partial^5 f}{\partial x^2 \partial t^3} \frac{h^2 k^3}{2!3!} + \dots \\
 & + \dots + \frac{\partial^{m+n} f}{\partial x^m \partial t^n} \frac{h^m k^n}{m! n!} + \dots \quad [8]
 \end{aligned}$$

where all the partial derivatives are evaluated at (a,c) . This is a two dimensional array for the function $f(x,t)$ with derivatives with respect to x running horizontally and with derivatives with respect to t running vertically. It is important for the reader to fix this picture in his mind when we discuss the application of two-dimensional Taylor series to partial differential equations and other complicated problems. It is merely a bookkeeping device, but one would be lost without it. We will define two-dimensional reduced derivatives as

$$F(m+1,n+1) := \frac{\partial^{m+n} f}{\partial x^m \partial t^n} \frac{h^m k^n}{m! n!} \quad [9]$$

The value of the function $f(x,t)$ at $x = a+h$ and $t = c+k$ is given by the double sum

$$f(a+h,c+k) = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} F(i,j) . \quad [10]$$

By development similar to that for Eq.[7], the reduced derivatives of a product of functions of two variables, $f(x,t) = u(x,t)*w(x,t)$, is given by the modified Leibnitz rule,

$$F(m,n) = \sum_{i=1}^m \sum_{j=1}^n U(i,j)*W(m-i+1,n-j+1) . \quad [11]$$

This is used in the solutions of partial differential equations and many other difficult problems. This completes the preliminary discussion of the ATS method of analysis. In the Chapters to follow, we will describe in detail the development of the ATS method and applications to a large variety of problems.

REFERENCES

1. J. W. Hanson, Jane Caviness, and Camilla Joseph, "Analytic Differentiation by Computer", *Comm. ACM*, vol. 5, #349; 1962.
2. R. E. Wensert, "A Simple Automatic Derivative Evaluation Program", *Comm ACM*, vol. 7, #463; 1964.
3. R. D. Wilkins, "Investigation of a New Analytical Method for Numerical Derivative Evaluation", *Comm. ACM*, vol. 7, #465; 1964.
4. R. E. Bellman, H. Kasaiwada, and R. E. Kalaba, "Wensert's Numerical Method for Partial Derivatives, Orbit Determination, and Quasi-Linearization", *Comm. ACM*, vol. 8, #231; 1965.
5. R. D. Wilkins, "General Time-Varying System Error Sensitivities Program", *Comm. ACM*, vol. 9, #855; 1966.
6. E. M. Wilson, "A Note on the Numerical Integration of Differential Equations", *Quart. J. Mech.*, vol. 2, #208; 1949.
7. A. Gibbons, "A Program for the Automatic Integration of Differential Equations using the Method of Taylor Series", *Computer J.*, vol. 3, #108; 1960.
8. L. Fox, 'NUMERICAL SOLUTION OF ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS', Pergamon Press, London, 1962, pp 18-20.
9. R. E. Moore, 'INTERVAL ANALYSIS', Prentice-Hall, Englewood Cliffs, N. J.; 1966.
10. R. D. Richtmyer, "Detached-Shock Calculations by Power Series", A.E.C. R&D Report, NYU-7973, New York University; 1957.
11. A. L. Kitzelman, Private Communications; 1967.
12. J. A. Leavitt, "Methods and Applications of Power Series", *Math. Comp.* vol. 20, #46; 1966.
13. D. Barton, I. M. Willers, and R. U. M. Zahar, "The Automatic Solution of Systems of Ordinary Differential Equations by the Method of Taylor Series", *Computer J.*, vol. 14, #243; 1971.
14. A. C. Norman, "Expanding the Solutions of Implicit Sets of Ordinary Differential Equations in Power Series", *Computer J.* vol. 19, #63; 1974.
15. H. Griffin, 'ELEMENTARY THEORY OF NUMBERS', McGraw-Hill, New York; 1954, #78.

CHAPTER II.

AUTOMATIC SEQUENTIAL DIFFERENTIATION

In the first section of this Chapter, we shall discuss the automatic sequential differentiation of simple analytic functions and arithmetic operations. We shall also introduce two subprograms that will be useful in the evaluation of Taylor series for analytic functions composed of combinations of simple analytic functions and operations. In the second section, we will discuss the method for differentiating complicated functions. Examples include the Bessel function, the Error function, and the Sine-integral. In the third section, the resolution of indeterminate forms and the generalized L'Hospital rule will be discussed. We will develop the automatic differentiation of inverse functions in the fourth section. It will become necessary to develop the sequential differentiation of functions of two variables. In the fifth section, the automatic sequential evaluation of partial derivatives will be discussed. In the final section, we will develop the differentiation of implicit functions.

A. Differentiation of Simple Functions and Operations.

Generally, simple analytic functions are composed of transcendental functions and algebraic operations. In order to get a clear understanding of the automatic sequential differentiation of simple analytic function we shall examine transcendental functions and algebraic operations as isolated cases. Later, we shall study the effects of combining these simple functions and operations to obtain the automatic differentiation of complex functions. The first function we shall look at is the sine function, $f(x) = \sin(ax)$. The sine function and first two derivatives are given by

$$f(x) = \sin(ax) , \quad f'(x) = a \cdot \cos(ax) , \quad f''(x) = -a^2 \cdot \sin(ax) . \quad [1]$$

Clearly, the recursive relation is

$$\frac{d^m f}{dx^m} = -a^2 * \frac{d^{m-2} f}{dx^{m-2}} . \quad [2]$$

In terms of the reduced derivatives (see Eq.[2] in Chapter 1), the recursive relation becomes

$$F(m) = -F(m-2) * \frac{a^2 h^2}{(m-1)(m-2)} , \quad [3]$$

where h is the increment in x . This reduced-derivative recursive relation can be used to evaluate the Taylor series terms for $f(x) = \sin(ax)$ in an unending sequence. This same recursive relation can be used for the function $f(x) = \cos(ax)$. The use of Eq.[3] is preferred over Eq.[2] because of the absence of the factorial function when Eq.[3] is used in Taylor series. The factorial function grows in magnitude too rapidly for any finite digital computer to handle reasonably. The presence of h in Eq.[3] permits control on overflow and underflow.

The exponential function, $f(x) = \exp(ax)$, can be differentiated in a similar manner. The exponential function and its first two derivatives are

$$f(x) = \exp(ax) , \quad f'(x) = a * \exp(ax) , \quad f''(x) = a^2 * \exp(ax) . \quad [4]$$

The recursive relation is

$$\frac{d^m f}{dx^m} = a * \frac{d^{m-1} f}{dx^{m-1}} . \quad [5]$$

In terms of reduced derivatives, the recursive relation becomes

$$F(m) = F(m-1) * \frac{a * h}{m-1} . \quad [6]$$

This recursive relation will be used later in this section for analyzing the function $f(x) = \exp(y)$, where y is another function of x .

A third example is the natural logarithm function, $f(x) = a * \ln(x)$. The natural logarithm function and first two derivatives are

$$f(x) = a \ln(x), \quad f'(x) = \frac{a}{x}, \quad f''(x) = -\frac{a}{x^2}. \quad [7]$$

The recursive relation for this function is

$$\frac{d^m f}{dx^m} = -\frac{d^{m-1} f}{dx^{m-1}} * \frac{m-1}{x}. \quad [8]$$

In terms of reduced derivatives, the recursive relation becomes

$$F(m) = -F(m-1) * \frac{h*(m-2)}{x*(m-1)}. \quad [9]$$

Of course, it is recognized that

$$F(m) = (-1)^m \frac{a h^{m-1}}{(m-1) x^{m-1}}. \quad [10]$$

However, Eq.[9] is to be preferred because the argument of the logarithm could be another function of x . For example, we will consider the function $f(x) = \ln(y)$, where $y(x) = \sin(ax)$, after we complete the discussion on arithmetic operations.

Arithmetic Operations

The automatic differentiation of functions with arithmetic operation proceed with ease when the operations are either addition or subtraction. The derivative of the sum of two functions is the sum of two individual derivatives. Under operation of multiplication, the derivative is found by the use of Leibnitz rule. Consider the product $f(x) = u(x)*w(x)$, the m -th term of the Taylor series for $f(x)$ is given by Eq.[7] in Chapter I.

$$F(m) = \sum_{i=1}^m U(i) W(m-i+1). \quad [11]$$

This is the foundation for the automatic sequential differentiation of complex functions. It has been written into a small FORTRAN function subprogram called the ATS function.