MATHEMATICS OF COMPUTATION, VOLUME 31, NUMBER 139
JULY 1977, PAGES 691–707

Taylor Series Methods for the Solution of Volterra Integral and Integro-Differential Equations

By Alan Goldfine

Abstract. Algorithms based on the use of Taylor series are developed for the numerical solution of Volterra integral and integro-differential equations of arbitrary order. It is shown that these algorithms are uniformly convergent, bounds are obtained for the truncation error, and an asymptotic error analysis is provided for the integral equation case. The various problems of computer implementation are discussed, and the results of certain experiments suggested by the theory are presented.

I. Introduction. Consider the equation

$$y^{(p)}(x) = f(x, y(x), y^1(x), \ldots, y^{(p-1)}(x)) + \int_a^x F(x, y(x), y^1(x), \ldots, y^{(p-1)}(x), s, y(s), y^1(s), \ldots, y^{(p)}(s))\,ds, \tag{1}$$

$$y^{(i)}(a) = y_0^{(i)}, \qquad i = 0, 1, \ldots, p-1, \; a \leqslant x \leqslant b,$$

which defines a Volterra integral or integro-differential equation of the second kind, according as $p = 0$ or $p \geqslant 1$.

In this paper, we discuss the numerical solution of (1) by a class of algorithms based on the Taylor expansion. The techniques are inspired by the Taylor series method of solution of the initial value problem of ordinary differential equations, and have been discussed in the context of a first order version of (1) by Feldstein and Sopka [6].

If $y(x), y^1(x), \ldots, y^{(p)}(x)$ are expanded in qth order Taylor series, and the derivatives $y^{(p+1)}(x), \ldots, y^{(p+q)}(x)$ are known, an approximate method of solution of (1) can be defined.

Traditionally, methods of this type were usually regarded as impractical, because the necessary derivatives of the right-hand side of (1) are often complicated and difficult to obtain by hand. Henrici [8, p. 66] discusses this problem in connection with the solution of ordinary differential equations. In recent years, however, attention has been given to the use of symbolic manipulation for the computation of derivatives and Taylor coefficients [1], [2]. Even more recently, a technique has been developed that allows the differentiation of functions defined by a FORTRAN program [9]. This very interesting concept involves the use of a specially designed pre-compiler.

Received January 20, 1974; revised December 30, 1976.

AMS (MOS) subject classifications (1970). Primary 65R05, 45L10, 68A15; Secondary 45D05, 45J05.

Key words and phrases. Volterra integral equation, Volterra integro-differential equation, Taylor series method, initial value problems, symbolic differentiation.

The technique used in this study for the automation of the differentiation problem uses a program written in FORMAC [14]. In comparison with hand calculation, the use of this program greatly enlarges the class of equations for which it is practical to solve using our Taylor series based methods.

In what follows, $\overrightarrow{y(u)}$ (or $\overrightarrow{y(u)}$, if ambiguous), stands for the sequence $y(u)$, $y^1(u), \ldots, y^{(v)}(u)$. Therefore, $F(x, \overrightarrow{y(x)}, s, \overrightarrow{y(s)})$ would represent the integrand in (1). In contexts where there is no ambiguity, we often abbreviate still further: f for $f(x, \overrightarrow{y(x)})$ and F for $F(x, \overrightarrow{y(x)}, s, \overrightarrow{y(s)})$.

II. Definition of the Algorithms. Two variations of a Taylor series algorithm of order q, $q \geqslant 1$, will be discussed. It is assumed that both f and F are $q + 1$ times continuously differentiable with respect to all their arguments.

Suppose we expand y, and each of the first p derivatives of y, in qth order Taylor series around $x_n = a + nh$:

$$y^{(r)}(x_{n+1}) = y^{(r)}(x_n) + hy^{(r+1)}(x_n) + \cdots + \frac{h^q}{q!} y^{(r+q)}(x_n) + \frac{h^{q+1}}{(q+1)!} y^{(r+q+1)}(\xi_n^{(r)}), \tag{2}$$

with $x_n < \xi_n^{(r)} < x_{n+1}$, for $r = 0, 1, \ldots, p$. By successively differentiating (1), using Leibnitz's rule, we obtain

$$y^{(p+i)}(x) = \frac{d^i}{dx^i} f + \sum_{j=0}^{i-1} \frac{d^{i-j-1}}{dx^{i-j-1}} \left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right] + \int_{x_0}^{x} \frac{d^i}{dx^i} F \, ds \tag{3}$$

for $i = 1, 2, \ldots, q$. From our assumptions it is easily seen that each of the functions $y^{(p+i)}$ are continuously differentiable, and that $y^{(p+q+1)}$ is continuous, all of these with respect to all arguments. It is assumed that we have convergent integration formulas which, for any continuous function g, have the form

$$\int_{x_0}^{x_n} g(x)dx = h \sum_{k=0}^{n} w_{nk} g(x_k) + \bar{E}(h, g) \tag{4}$$

with bounded weights $|w_{nk}| \leqslant W$, for $k \leqslant n$, and $w_{00} = 0$.

Now, using (3) and (4), the following approximations $y_n^{(p+i)}$ to $y^{(p+i)}(x_n)$, $i = 0, 1, \ldots, q$, are defined:

$$y_n^{(p+i)} = \left[\frac{d^i}{dx^i} f\right]_{x=x_n, \overrightarrow{y(x_n)}=\vec{y}_n} + \sum_{j=0}^{i-1} \left[\frac{d^i}{dx^i} \left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]\right]_{x=x_n, \overrightarrow{y(x_n)}=\vec{y}_n} + h \sum_{k=0}^{n} w_{nk}^{(i)} \left[\left[\frac{d^i}{dx^i} F\right]_{s=x_k, x=x_n, \overrightarrow{y(x_k)}=\vec{y}_k}\right] + O(h^\alpha). \tag{5}$$

Putting these together, an algorithm of order q can be defined as follows:

Algorithm 1. At step n, $n = 0, 1, \ldots, N - 1$,

1. $y_n, y_n^{(1)}, \ldots, y_n^{(p)}$ are known.
2. Using (5), calculate $y_n^{(p+1)}, y_n^{(p+2)}, \ldots, y_n^{(p+q)}$.
3. For $r = 0, 1, \ldots, p$, calculate

(6)
$$y_{n+1}^{(r)} = y_n^{(r)} + hy_n^{(r+1)} + \cdots + \frac{h^q}{q!} y_n^{(r+q)} + O(h^\alpha).$$

The $O(h^\alpha)$ terms in (5) and (6.3) can be thought of as representing the total of various corrections, including perhaps roundoff error. It is assumed that $\alpha \geqslant 1$.

In the above, we could have defined

(7)
$$y_{n+1}^{(p)} = f(x_{n+1}, \overrightarrow{y_{n+1}}) + h \sum_{k=0}^{n+1} w_{n+1,k}^{(p)} F(x_{n+1}, \overrightarrow{y_{n+1}}, x_k, \vec{y}_k) + O(h^\alpha),$$

instead of defining $y_{n+1}^{(p)}$ by means of a Taylor expansion. Consideration of the resulting algorithm, which eliminates the need to calculate the qth derivative of (1) and leads to the possibility of an implicit definition of $y_{n+1}^{(p)}$, is necessary to the subsequent theory. We omit a discussion of the precise nature of any iteration, and define:

Algorithm 2. At step n, $n = 0, 1, \ldots, N-1$,

1. $y_n, y_n^{(1)}, \ldots, y_n^{(p)}$ are known.
2. Using (5), calculate $y_n^{(p+1)}, y_n^{(p+2)}, \ldots, y_n^{(p+q-1)}$.
3. For $r = 0, 1, \ldots, p-1$, calculate

(8)
$$y_{n+1}^{(r)} = y_n^{(r)} + hy_n^{(r+1)} + \cdots + \frac{h^q}{q!} y_n^{(r+q)} + O(h^\alpha).$$

4. Use (7) to calculate a value for $y_{n+1}^{(p)}$.

III. Analysis of Convergence. In order to bound the truncation error, the quantities $e_n^{(r)} = y_n^{(r)} - y^{(r)}(x_n)$, $r = 0, 1, \ldots, p+q$, $n = 0, 1, \ldots, N-1$, are examined. Algorithm 1 is considered first. For $r = 0, 1, \ldots, p$, we subtract (2) from (6.3) and take absolute values to obtain

$$|y_{n+1}^{(r)} - y^{(r)}(x_{n+1})| \leqslant |y_n^{(r)} - y^{(r)}(x_n)| + h|y_n^{(r+1)} - y_n^{(r+1)}(x_n)|$$
$$+ \cdots + \frac{h^q}{q!}|y_n^{(r+q)} - y^{(r+q)}(x_n)| + \frac{h^{q+1}}{(q+1)!}|y^{(r+q+1)}(\xi_n^{(r)})|$$
$$+ O(h^\alpha)$$

or

(9)
$$|e_{n+1}^{(r)}| \leqslant \sum_{j=0}^{q} \frac{h^j}{j!}|e_n^{(r+j)}| + O(h^{q+1}) + O(h^\alpha),$$

since $y^{(r+q+1)}(x)$ is continuous on the closed interval $a \leqslant x \leqslant b$, and so is bounded.

Now, for $e_{n+1}^{(p+i)}$, $i = 1, 2, \ldots, q$, we subtract (3), evaluated at $x = x_n$, from (5), and take absolute values to obtain

$$|e_{n+1}^{(p+i)}| \leqslant \left| \left[\frac{d^i}{dx^i} f\right]_{x=x_n, \overrightarrow{y(x)}=\vec{y}_n} - \left[\frac{d^i}{dx^i} f\right]_{x=x_n} \right|$$

$$+ \sum_{j=0}^{i-1} \left| \left[\frac{d^{i-j-1}}{dx^{i-j-1}} \left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]_{x=x_n, \overrightarrow{y(x)}=\vec{y}_n}\right] \right.$$

(10)
$$\left. - \left[\frac{d^{i-j-1}}{dx^{i-j-1}} \left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]\right]_{x=x_n} \right|$$

$$+ \left| h \sum_{k=0}^{n} w_{nk}^{(i)} \left[\left[\frac{d^i}{dx^i} F\right]_{s=x_k, x=x_n, \overrightarrow{y(s)}=\vec{y}_k}\right] \right.$$

$$\left. - \left[\int_{x_0}^{x} \frac{d^i}{dx^i} F\, ds\right]_{x=x_n} \right| + O(h^\alpha).$$

Now, $d^i f/dx^i$ is a function of the $p + i + 1$ functions x, $\overrightarrow{y(x)}^{p+i-1}$. By expanding the approximate version of this in (10) in a Taylor series around the point $(x_n, \overrightarrow{y(x_n)})$, we get

$$\sum_{m=0}^{p+i-1} \left| (y_n^{(m)} - y^{(m)}(x_n)) \left[\left[\frac{\partial}{\partial y^{(m)}(x)} \left[\frac{d^i}{dx^i} f\right]\right]_{x=x_n, y^{(m)}(x)=\mu_n^{(m)}}\right] \right|$$

with $\mu_n^{(m)}$ between $y_n^{(m)}$ and $y^{(m)}(x_n)$, for the first term of (10). The absolute values of the partial derivative factors are bounded by, say K_f, yielding $K_f \Sigma_{m=0}^{p+i-1} |e_n^{(m)}|$ as an upper bound.

Suppose we expand, in a similar manner,

$$\left[\frac{d^{i-j-1}}{dx^{i-j-1}} \left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]_{x=x_n, \overrightarrow{y(x)}=\vec{y}_n}\right] \quad \text{and} \quad \left[\left[\frac{d^i}{dx^i} F\right]_{s=x_k, x=x_n, \overrightarrow{y(s)}=\vec{y}_k}\right]$$

in Taylor series around the points $(x_n, \overrightarrow{y(x_n)})$ and $(x_n, \overrightarrow{y(x_n)}, x_k, \overrightarrow{y(x_k)})$, respectively. If we then bound the resulting partial derivative factors by K_F and K_I, respectively, an upper bound for the first sum in (10) would be

$$iK_F \sum_{m=0}^{p+i-1} |e_n^{(m)}|$$

and a bound for the integral term would be

$$2(b-a)K_I W \sum_{m=0}^{p+i-1} |e_n^{(m)}| + hK_I W \sum_{k=0}^{n-1} \sum_{m=0}^{p} |e_k^{(m)}| + E(h, i),$$

where $E(h, i) = \bar{E}(h, d^i F/dx^i)$.

Letting $K_1 = K_f + qK_F + 2(b-a)K_I W$ and $K_2 = K_I W$, we can combine terms to obtain

$$|e_n^{(p+i)}| \leqslant K_1 \sum_{m=0}^{p+i-1} |e_n^{(m)}| + hK_2 \sum_{k=0}^{n-1} \sum_{m=0}^{p} |e_k^{(m)}| + E(h, i) + O(h^\alpha). \tag{11}$$

We would now like to express the $e_n^{(p+i)}$ in terms of the $e_j^{(r)}$, $r = 0, 1, \ldots, p$, $j = 0, 1, \ldots, n$. The following lemma can be proved directly by induction:

LEMMA 1. *Suppose that for some sequence of nonnegative real numbers* $\{z_j\}$, *we have that* $Z_{j+1} \leqslant A \sum_{u=0}^{j} Z_u + g_{j+1}$, *for* $j = t, t+1, \ldots$. *Then*

$$Z_{j+1} \leqslant A(A+1)^{j-t} \sum_{u=0}^{t} Z_u + A \sum_{u=t+1}^{j} (A+1)^{j-u} g_u + g_{j+1} \quad \text{for } j = t, t+1, \ldots. \tag{12}$$

Lemma 1 is applied to (11), with $j = p + i - 1$, $t = p$, $Z_u = |e_n^{(u)}|$, $A = K_1$ and

$$g_u = hK_2 \sum_{k=0}^{n-1} \sum_{m=0}^{p} |e_k^{(m)}| + E(h, u-p) + O(h^\alpha).$$

If we let $K_3 = K_1(K_1+1)^{q+1}$, $K_4 = K_2 + K_2K_1(K_1+1)^{q+1} \cdot (q+1)$ and $K_5 = 1 + K_1(K_1+1)^{q-1}$, the result is that

$$|e_n^{(p+i)}| \leqslant K_3 \sum_{u=0}^{p} |e_n^{(u)}| + hK_4 \sum_{k=0}^{n-1} \sum_{m=0}^{p} |e_k^{(m)}| + K_5 \sum_{u=1}^{i} E(h, u) + O(h^\alpha). \tag{13}$$

At this point we separate our analysis and restrict ourselves, for the time being, to Algorithm 1. By using (13), we can eliminate all occurrences of error expressions of order $> p$ from (9). Let $\epsilon_k = \max_{0 \leqslant r \leqslant p; 0 \leqslant j \leqslant k} |e_j^{(r)}|$. Then by (13),

$$|e_n^{(p+i)}| \leqslant K_3(p+1)\epsilon_n + hK_4(p+1)n\epsilon_n + K_5 \sum_{u=1}^{i} E(h, u) + O(h^\alpha).$$

Letting $K_6 = K_3(p+1) + (b-a)K_4(p+1)$, we have

$$|e_n^{(p+i)}| \leqslant K_6\epsilon_n + K_5 \sum_{u=1}^{i} E(h, u) + O(h^\alpha). \tag{14}$$

Assume that $h \leqslant h_0$, for some constant $h_0 < 1$. Let $H = 1 + h_0/2! + \cdots + h_0^{q-1}/q!$. Since those terms in (9) for which $j \geqslant \min\{q, p-r\}$ are the terms replaceable by (14), we have, for $r = 0, 1, \ldots p$,

$$|e_n^{(r)}| \leqslant \sum_{j=0}^{\min\{q,p-r\}} \frac{h^j}{j!}\epsilon_n + \sum_{j=p-r+1}^{q} \frac{h^j}{j!}\left[K_6\epsilon_n + K_5 \sum_{t=1}^{r-p+j} E(h,t) + O(h^\alpha)\right]$$

$$+ O(h^\alpha) + O(h^{q+1})$$

$$\leqslant \epsilon_n(1 + hHK_6) + K_5 \sum_{t=1}^{r-p+q} E(h,t) \sum_{j=p-r+1}^{q} \frac{h^j}{j!} + O(h^\alpha) + O(h^{q+1}).$$

Since the second term is at a maximum when $r = p$, we have

$$\epsilon_{n+1} \leqslant \epsilon_n(1 + hHK_6) + hK_5 \sum_{t=1}^{q} E(h,t) \cdot O(h^{t+1}) + O(h^\alpha) + O(h^{q+1}). \tag{15}$$

We are now ready for our first major result.

THEOREM 2. *Let a numerical algorithm for solving* (1) *be defined by Algorithm* 1. *Then, using the notation we have developed,*

$$\text{(a)}\quad \epsilon_n \leqslant \exp\{(x_n - a)HK_6\}\epsilon_0$$

$$+ \frac{\exp\{(x_n - a)HK_6\} - 1}{HK_6}\left[K_5 \sum_{t=1}^{q} E(h,t) \cdot O(h^{t-1})\right] \tag{16}$$

$$+ O(h^{\alpha-1}) + O(h^q),$$

$$\text{(b)}\quad \lim_{h\to 0;N\to\infty;hN=constant} \max_{0\leqslant n\leqslant N} |y(x_n) - y_n| = 0.$$

Proof. We apply Henrici's Lemma [8, p. 18] to (15). This yields

$$\epsilon_n \leqslant \exp\{nhHK_6\}\epsilon_0$$

$$+ \frac{\exp\{nhHK_6\} - 1}{hHK_6}\left[hK_5 \sum_{t=1}^{q} E(h,t)O(h^{t-1}) + O(h^\alpha) + O(h^{q+1})\right].$$

By observing that $nh = x_n - a$ and simplifying, (16) is obtained. Now, $\max_{0\leqslant n\leqslant N}|y(x_n) - y_n| \leqslant \epsilon_n$ by definition, and $\epsilon_0 = O(h^\alpha)$, so if we assume that $E(h, 1)$ is at least $O(h)$ and that the other $E(h, t)$ are bounded, we see that

$$\lim_{h\to 0;N\to\infty;hN=\text{constant}} \max_{0\leqslant n\leqslant N} |y(x_n) - y_n| \leqslant \lim_{h\to 0;N\to\infty;hN=\text{constant}} \epsilon_n = 0.$$

With respect to the order of convergence, we have that

$$\exp\{(x_n - a)HK_6\} \quad \text{and} \quad \frac{\exp\{(x_n - a)HK\} - 1}{HK_6}K_5$$

are constants, and that $\epsilon_0 = O(h^\alpha)$, so

$$\max_{0\leqslant n\leqslant N} |y(x_n) - y_n| \leqslant \epsilon_n \leqslant \sum_{t=1}^{q} E(h,t) \cdot O(h^{t-1}) + O(h^{\alpha-1}) + O(h^q).$$

Therefore, we can state

COROLLARY 3. *In order to insure $O(h^q)$ convergence (the maximum fixed by the order of the Taylor expansion) in the application of Algorithm* 1, *we must have*

(a) $O(E(h, t)) \geqslant q - t + 1$, *and*

(b) $\alpha \geqslant q + 1$.

In other words, the quadrature rule for the integral of the first derivative of F must be of order q, but rules for successively higher derivatives can be successively less precise. In addition, the order of any additional error made during a given step must be no smaller than $\alpha = q + 1$.

The analysis of Algorithm 2 is similar. The major difference is in the bound for $|e_n^{(p)}|$, which now resembles in structure those for the $|e_n^{(p+i)}|$, $i = 1, 2, \ldots$. If (1), evaluated at x_n is subtracted from (7), for $y_n^{(p)}$, and absolute values are taken, then

$$|e_n^{(p)}| \leqslant K_1 \sum_{m=0}^{p-1} |e_n^{(m)}| + hK_2 \sum_{k=0}^{n} \sum_{m=0}^{p} |e_k^{(m)}| + E(h, 0) + O(h^\alpha),$$

by analogy to the argument leading up to (11). $|e_n^{(p)}|$ now appears on both sides of the inequality. Solving for $|e_n^{(p)}|$, and introducing $\tilde{\epsilon}_k = \max_{0 \leqslant r \leqslant p-1;\, 0 \leqslant j \leqslant k} |e_j^{(r)}|$, we get

$$|e_n^{(p)}| \leqslant hK_7 \sum_{k=0}^{n-1} |e_k^{(p)}| + K_8 \tilde{\epsilon}_n + \frac{E(h, 0)}{1 - h_0 K_8} + O(h^\alpha),$$

where

$$K_7 = \frac{K_2}{1 - h_0 K_2} \quad \text{and} \quad K_8 = \frac{K_1 p}{1 - h_0 K_2}, \qquad h \leqslant h_0 \leqslant K_2.$$

If we apply Lemma 1, and observe that $1 + hK_7 \leqslant e^{hK_7}$, $h(n-1) \leqslant b - a$, and $|e_0^{(p)}| = O(h^\alpha)$, we obtain

$$|e_n^{(p)}| \leqslant (b - a)K_7 K_8 e^{(b-a)K_7} \tilde{\epsilon}_{n-1} + \frac{(b - a)K_7 e^{(b-a)K_7}}{1 - h_0 K_2} E(h, 0)$$

$$+ K_{10} \tilde{\epsilon}_n + \frac{E(h, 0)}{1 - h_0 K_2}.$$

Now, $\tilde{\epsilon}_{n-1} \leqslant \tilde{\epsilon}_n$, so letting

$$K_9 = (K_8 + K_8(b - a)K_7 e^{(b-a)K_7}) \quad \text{and} \quad K_{10} = \frac{K_7(b - a)e^{(b-a)K_7} + 1}{1 - h_0 K_2},$$

it follows that

$$|e_n^{(p)}| \leqslant K_9 \tilde{\epsilon}_n + K_{10} E(h, 0) + O(h^\alpha). \tag{17}$$

Using (17) and the definition of $\tilde{\epsilon}_k$ in (13), we obtain, for $i = 1, 2, \ldots, q - 1$,

$$|e_n^{(p+i)}| \leqslant [pK_3 + K_3 K_9 + (b - a)K_3 K_4 K_9 + p(b - a)K_4] \tilde{\epsilon}_n$$

$$+ [K_3 K_{10} + (b - a)K_4 K_{10}] E(h, 0) + K_5 \sum_{j=1}^{i} E(h, j) + O(h^\alpha).$$

If we define

$$K_{11} = \max\{K_9,\ pK_3 + K_3K_9 + (b-a)K_3K_4K_9 + p(b-a)K_4\} \text{ and}$$

$$K_{12} = \max\{K_{10},\ K_3K_{10} + (b-a)K_4K_{10} + K_5\},$$

it follows that for $i = 0, 1, \ldots, q-1$,

$$|e_n^{(p+i)}| \leqslant K_{11}\tilde{\epsilon}_n + K_{12}\sum_{j=0}^{i} E(h, j) + O(h^\alpha).$$

This expression can be used to eliminate from (9) those terms for which $j > \min\{q, p-r-1\}$:

$$|e_{n+1}^{(r)}| \leqslant \tilde{\epsilon}_n(1 + hHK_{11}) + hK_{12}\sum_{t=0}^{q-1} E(h, t)\cdot O(h^t) + O(h^\alpha) + O(h^{q+1}),$$

for $r = 0, 1, \ldots, p-1$. Thus we have the analogue of Theorem 2:

THEOREM 4. *Let Algorithm 2 be used to solve* (1). *Then, using the notation we have developed,*

$$\tilde{\epsilon}_n \leqslant \exp\{(x_n - a)HK_{11}\}\tilde{\epsilon}_0$$

(a)
$$+ \frac{\exp\{(x_n-a)HK_{11}\} - 1}{HK_6}\left[K_{12}\sum_{t=0}^{q-1} E(h, t)\cdot O(h^t)\right]$$

$$+ O(h^{\alpha+1}) + O(h^{q+1}),$$

(b)
$$\lim_{h\to 0;N\to\infty;hN=\text{constant}} \max_{0\leqslant n\leqslant N} |y(x_n) - y_n| = 0.$$

COROLLARY 5. *In order to insure $O(h^q)$ convergence for Algorithm 2, it is necessary to have*

(a) $O(E(h, t)) \geqslant q - t$, *and*

(b) $\alpha \geqslant q + 1$.

IV. An Asymptotic Error Expansion. We would now like to derive an asymptotic expansion for the errors incurred in the application of Algorithm 1, in the case $p = 0$. This is the case of the classical Volterra integral equation

(18)
$$y(x) = f(x) + \int_{x_0}^{x} F(x, s, y(s))\,ds.$$

It is assumed that the functions f and F are now $q+2$ times continuously differentiable. The following assumption is also made, concerning the integration rules we are using: For each i, $i = 1, 2, \ldots, q$, there exists a function Q_i of two variables, such that

$$\left[\int_{x_0}^{x} \frac{d^i}{dx^i} F\,ds\right]_{x=x_n} - h\sum_{k=0}^{n} w_{nk}^{(i)}\left[\frac{d^i}{dx^i} F\right]_{x=x_n, s=x_k}$$

$$= Q_i(x_n, y(x_n))h^v + O(h^{v+1}).$$

In terms of Corollary 3,

$$E(h, i) = Q_i(x_n, y(x_n))h^{q-i+1} + O(h^{q-i+2}).$$

Our goal is to find an expression for e_{n+1} that does not involve any errors of order $\geqslant 1$. To do this, we reconsider our analysis of $e_n^{(p+i)}$, $i = 1, 2, \ldots, q$. Subtracting (3) from (5) yields

$$e_n^{(i)} = \left\{\left[\frac{d^i}{dx^i} f\right]_{x=x_n} - \left[\frac{d^i}{dx^i} f\right]_{x=x_n}\right\}$$

$$+ \sum_{j=0}^{i-1} \left\{\left[\left[\frac{d^{i-j-1}}{dx^{i-j-1}}\left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]\right]_{x=x_n, \overrightarrow{y(x)}=\overrightarrow{y_n}^{i-1}}^{i-1}\right]\right.$$

(19)

$$\left. - \left[\left[\frac{d^{i-j-1}}{dx^{i-j-1}}\left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]\right]_{x=x_n}\right]\right\}$$

$$+ \left\{h \sum_{k=0}^{n} w_{nk}^{(i)} \left[\left[\frac{d^i}{dx^i} F\right]_{x=x_n, s=x_k, y(s)=y_k}\right]\right.$$

$$\left. - \left[\int_{x_0}^{x} \frac{d^i}{dx^i} F ds\right]_{x=x_n}\right\} + O(h^\alpha).$$

The expression in the first braces of (19) equals zero. The approximation terms in the second braces are again expanded in a Taylor series around the point $(x_n, \overrightarrow{y(x_n)}^{i-1})$. This time however, they are expanded through an additional term:

$$\left[\frac{d^{i-j-1}}{dx^{i-j-1}}\left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]\right]_{x=x_n, \overrightarrow{y(x)}=\overrightarrow{y_n}}$$

$$= \left[\frac{d^{i-j-1}}{dx^{i-j-1}}\left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]\right]_{x=x_n}$$

(20)

$$+ \sum_{m=0}^{i-1} e_n^{(m)} \left[\frac{\partial}{\partial y^{(m)}(x)} \frac{d^{i-j-1}}{dx^{i-j-1}}\left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]\right]_{x=x_n}$$

$$+ \frac{1}{2}\left[\left(\sum_{m=0}^{i-1} e_n^{(m)} \frac{\partial}{\partial y^{(m)}(x)}\right)^2 \left[\frac{d^{i-j-1}}{dx^{i-j-1}}\left[\left[\frac{d^j}{dx^j} F\right]_{s=x}\right]\right]_{x=x_n, \overrightarrow{y(x)}=\overrightarrow{\xi_n}^{i-1}}^{i-1}\right]$$

with $\overrightarrow{\xi_n}^{i-1}$ between $\overrightarrow{y_n}^{i-1}$ and $\overrightarrow{y(x_n)}^{i-1}$. All of the partial derivatives are bounded, so

using (14) and Corollary 3 we get for the expression in the second braces of (19):

$$e_n^{(0)}\left[\frac{\partial}{\partial y(x)}\,[F_{s=x}]\right]_{x=x_n} + O(\epsilon_n)^2 \quad \text{if } i = 1, \text{ and}$$

$$\text{(21)}\quad \sum_{j=0}^{i-1}\sum_{m=0}^{i-1}\left[\frac{\partial}{\partial y^{(m)}(x)}\,\frac{d^{i-j-1}}{dx^{i-j-1}}\left[\left[\frac{d^j}{dx^j}\,F\right]_{s=x}\right]\right]_{x=x_n}\cdot e_n^{(m)} + O(\epsilon_n)^2$$

$$+ O(E(h, i-1))^2 + O(h^{2\alpha}) + O(\epsilon_n)\cdot O(E(h, i-1)) + O(\epsilon_n)O(h^{\alpha})$$

$$+ O(E(h, i-1))O(h^{\alpha}),$$

if $i = 2, 3, \ldots, q$. For the integration expression in the third braces of (19), we expand the function into a Taylor series around $(x_n, x_k, y(x_k))$

$$\left[\frac{d^i}{dx^i}F\right]_{x=x_n, s=x_k, y(x_k)=y_k}$$

$$\text{(22)}\quad = \left[\frac{d^i}{dx^i}F\right]_{x=x_n, s=x_k} + e_k^{(0)}\,\frac{\partial}{\partial y(x_k)}\left[\left[\frac{d^i}{dx^i}F\right]_{x=x_n, s=x_k}\right]$$

$$+ \frac{1}{2}\left(e_k^{(0)}\,\frac{\partial}{\partial y(x_k)}\right)^2\left[\left[\frac{d^i}{dx^i}F\right]_{x=x_n, s=x_k, y(x_k)=\mu_k}\right],$$

with μ_k between $y(x_k)$ and y_k. Since $e_k^{(0)} \leqslant \epsilon_k = O(\epsilon_n)$, the error term in (22) is simply $O(\epsilon_n)^2$, and the last braces of (19) equals

$$\text{(23)}\quad h\sum_{k=0}^{n} w_{nk}^{(i)}\left[\frac{\partial}{\partial y(x_k)}\left[\left[\frac{d^i}{dx^i}F\right]_{x=x_n, s=x_k}\right]\right]e_k^{(0)} + O(\epsilon_n)^2 + E(h, i).$$

By inserting (21) and (23) into (19), and using the previously obtained expression,

$$e_n^{(1)} = \left[\frac{\partial}{\partial y(x)}\,[F_{s=x}]\right]_{x=x_n}\cdot e_n^{(0)}$$

$$\text{(24)}\quad + h\sum_{k=0}^{n} w_{nk}^{(i)}\left[\frac{\partial}{\partial y(x_k)}\left[\left[\frac{d}{dx}\,F\right]_{x=x_n, s=x_k}\right]\right]e_k^{(0)} + Q_1(x_n, y(x_n))h^q$$

$$+ O(h^{q+1}) + O(h^{\alpha})$$

and, for $i = 2, 3, \ldots, q$,

$$\text{(25)}\quad e_n^{(i)} = Q_i(x_n, y(x_n))h^{q-i+1} + O(h^{q-i+2}) + O(h^{\alpha}).$$

At this point, if an exact Taylor series expansion for $y(x_{n+1})$, taken out to one more term than (2), is subtracted from (6.3), and (24) and (25) are substituted, we arrive at the representation

$$e_{n+1} = e_n + h\left[\frac{\partial}{\partial y(x)}[F_{s=x}]\right]_{x=x_n} e_n$$

$$+ h^2 \sum_{k=0}^{n} w_{nk}^{(i)} \left[\frac{\partial}{\partial y(x_k)}\left[\left[\frac{d}{dx}F\right]_{x=x_n, s=x_k}\right]\right] e_k + h^{q+1} \sum_{j=1}^{q} \frac{Q_j}{j!} \tag{26}$$

$$+ h^{q+1} \frac{y^{(q+1)}(x_n)}{(q+1)!} + O(h^{q+2}) + O(h^{\alpha}).$$

We now come to the asymptotic expansion itself:

THEOREM 6. *Let y satisfy the Volterra integral equation* (18), *with f and F $q+2$ times continuously differentiable on $[a, b]$ for some fixed q. Then, if $\alpha \geqslant q+2$,*

$$e_n = y_n - y(x_n) = h^q e(x_n) + O(h^{q+1}),$$

where e is the solution of the first order ($p = 1$) Volterra integro-differential equation

$$e^{(1)}(x) = \sum_{j=1}^{q} \frac{Q_j(x, y(x))}{j!} - \frac{y^{(q+1)}(x)}{(q+1)!}$$

$$+ \left[\frac{\partial}{\partial y(x)}\left[\left[\frac{d}{dx}F(x, s, y(s))\right]_{s=x}\right]\right] e(x) \tag{27}$$

$$+ \int_{x_0}^{x} \left[\frac{\partial}{\partial y(s)} \frac{d}{dx} F(x, s, y(s))\right] e(s)\, ds,$$

with $e(x_0) = 0$.

Proof. Suppose we divide (26) by h^q and define $\bar{e}_n = e_n/h^q$, $\bar{e}_n^{(1)} = e_n^{(1)}/h^q$. Then

$$\bar{e}_{n+1} = \bar{e}_n + h\left[\frac{\partial}{\partial y(x)}[F_{s=x}]\right]_{x=x_n} \bar{e}_n$$

$$+ h^2 \sum_{k=0}^{n} w_{nk}^{(1)} \left[\frac{\partial}{\partial y(x_k)}\left[\left[\frac{d}{dx}F\right]_{x=x_n, s=x_k}\right]\right] \bar{e}_k \tag{28}$$

$$+ h \sum_{j=0}^{q} \frac{Q_j}{j!} - h \frac{y^{(q+1)}(x_n)}{(q+1)!} + O(h^2).$$

Now, consider the first order integro-differential equation defined by (27).

Suppose Algorithm 2 is applied to this equation, with the order of the algorithm being 1, the approximations to $e(x_k)$ and $e^{(1)}(x_k)$ called $\bar{e}_k$ and $\bar{e}_k^{(1)}$, respectively, and the additional error made at each step no greater than $O(h^2)$. We then have

$$\bar{e}_{n+1} = \bar{e}_n + h\bar{e}_n^{(1)} + O(h^2), \tag{29}$$

$$\bar{e}_{n+1}^{(1)} = \sum_{j=1}^{q} \frac{Q_j}{j!} - \frac{y^{(q+1)}(x_{n+1})}{(q+1)!} + \left[\frac{\partial}{\partial y(x)}\,[F_{s=x}]\right]_{x=x_{n+1}} \bar{e}_{n+1}$$

$$+ h \sum_{k=0}^{n+1} w_{n+1,k}^{(1)} \left[\frac{\partial}{\partial y(x_k)}\left[\left[\frac{d}{dx}\,F\right]_{x=x_{n+1},s=x_k}\right]\right] \bar{e}_k + O(h^2). \tag{30}$$

If we write out the equation for $\bar{e}_n^{(1)}$, analogous to (30), and substitute into (29), then (29) is precisely (28). Therefore, by Corollary 5,

$$\left|\frac{e_n}{h^q} - e(x_n)\right| = |\bar{e}_n - e(x_n)| \leqslant \tilde{\epsilon}_n = O(h), \qquad |e_n - h^q e(x_n)| = O(h^{q+1})$$

so $y_n = y(x_n) + h^q e(x_n) + O(h^{q+1})$, which was to be proved.

For asymptotic results on related problems, see Linz [11] and Feldstein and Sopka [6].

V. Implementation. The most difficult part of the two algorithms to implement, indeed the factor that has usually led to the dismissal of Taylor series methods from practical consideration in several areas, is the necessary differentiation of perhaps complicated functions.

A PL/1-FORMAC program was written to perform this differentation. The program symbolically computed the functions

$$\frac{d^i}{dx^i}F \quad \text{and} \quad \frac{d^i}{dx^i}f + \sum_{j=0}^{i-1} \frac{d^{i-j-1}}{dx^{i-j-1}}\left[\left[\frac{d^j}{dx^j}\,F\right]_{s=x}\right],$$

for arbitrary values of i. The FORMAC statements defining these functions were then punched out as legal FORTRAN assignment statements with the aid of the FORMAC utility program PUNCHE [10]. The numerical results were then calculated on an IBM 370/165 using FORTRAN IV-G double precision.

For the purposes of this study, the actual FORTRAN programs handled orders $q = 1, 2, 3$, and 4. The standard composite Newton-Cotes rules were used for the various integrations. Where the Simpson rule was used, the difficulty involved in having an even number of points was resolved (for $n \geqslant 3$) by using the standard rule for $\int_{x_0}^{x_{n-3}}$, and the Simpson 3/8 rule for $\int_{x_{n-3}}^{x_n}$.

For $n = 3$, this will not work. We need an alternate means of computing $y_1^{(3)}$ and $y_1^{(4)}$. Using FORMAC, we can easily differentiate (1) $q + 1$ times instead of q times, finding the $O(h^\alpha)$ accurate values $y_0^{(p+q+j)}$, $j = 1, 2, \ldots, i$. If we let

$$y_1^{(p+i)} = y_0^{(p+i)} + hy_0^{(p+i+1)} + \cdots + \frac{h^q}{q!} y_0^{(p+i+q)},$$

the error is seen to be

$$e_1^{(p+i)} \leqslant O(h^\alpha) + \frac{h^{q+1}}{(q+1)!} y_0^{(p+q+i+1)}(\xi), \qquad x_0 < \xi < x_1,$$

$$= O(h^\alpha) + O(h^{(q+1)}).$$

Thus, we retain our qth order accuracy.

In our implementation of Algorithm 2, in order to obtain a value for the $y_{n+1}^{(p)}$ appearing on the right-hand side of (7), $y_n^{(p)}$ is first evaluated by means of the equivalent formula in Algorithm 1. Therefore, we essentially have a predictor-corrector arrangement in the approximation of $y^{(p)}(x_n)$. All the other values are computed directly, without iteration.

VI. Computational Efficiency. In order to provide some estimates for the amount of time required to execute each of the proposed algorithms, we make the following assumptions:

–The major part of the computational effort in both cases lies in the repeated evaluations of the functions f and F (and their derivatives).

–The necessary derivatives will have been obtained previously.

– f, F, $\dfrac{d^i}{dx^i} f$, $\dfrac{d^{i-j-1}}{dx^{i-j-1}} \left[\left[\dfrac{d^j}{dx^j} F \right]_{s=x} \right]$ and $\dfrac{d^i}{dx^i} F$,

for all values of i, j, and n, can each be evaluated (numerically) in roughly the same time.

–Each approximate integration requires $n + 1$ function evaluations.

For Algorithm 1, the functional count is as follows: At x_n, we must evaluate $y_n^{(p+i)}$ for $i = 1, 2, \ldots, q$. $y_n^{(p+i)}$ requires one evaluation of $d^i f/dx^i$, i evaluations of

$$\frac{d^{i-j-1}}{dx^{i-j-1}} \left[\left[\frac{d^j}{dx^j} F \right]_{s=x} \right],$$

and, in the numerical integration, $n + 1$ evaluations of $d^i F/dx^i$, for a total of $n + i + 2$. Therefore, the total functional count would be

$$\sum_{n=0}^{N} \sum_{i=1}^{q} (n + i + 2) = \frac{q}{2} N^2 + \left(\frac{q^2}{2} + 3q \right) N + \frac{q^2 + 5q}{2}.$$

For large N, therefore, the dominant term would be $qN^2/2$.

The count for Algorithm 2 is similar. In our implementation, $y_{n+1}^{(p)}$ is predicted by Algorithm 1, then corrected by (7), so there are an additional $N + 2$ evaluations per step:

$$\sum_{n=0}^{N} \left[n + 2 + \sum_{i=1}^{q} (n + i + 2) \right] = \frac{q+1}{2} N^2 + \frac{(q+1)(q+5)}{2} N + \frac{(q+1)(q+4)}{2}.$$

For large N, the $(q + 1)N^2/2$ term dominates.

VII. Numerical Comparisons and Experiments. The following three equations were among those to which our algorithms were applied:

$$(1) \qquad y(x) = 1 + x - \cos(x) - \int_{x_0}^{x} y(s)\cos(x - s)\,ds, \qquad 0 \leqslant x \leqslant 1,$$

with solution $y(x) = x$.

$$(2)\ y'(x) = 1 + 2x - y(x) + \int_{x_0}^{x} y(s)x(1 + 2x)e^{s(x-s)}\,ds, \qquad 0 \leqslant x \leqslant 1, y(0) = 1,$$

with solution $y(x) = e^{x^2}$.

$$(3)\ y''(x) = y'(x) - \frac{x^2 y(x)}{2} - 1 + \int_{x_0}^{x} [sy(x) - y'(s) - y''(s)]\,ds, \qquad 0 \leqslant x \leqslant 1, y(0) = 0, y'(0) = 1,$$

with exact solution $y(x) = x$.

The numerical results for the two algorithms are summarized in Tables 1, 2, and 3, which present the absolute errors at the point $x = 1.0$ for $q = 1, 2, 3, 4$, and $h = .1, .05, .025$.

Equations (1) and (2) have been considered previously in the literature. Our $q = 4$ results will be used as standards for comparison.

For Eq. (1), Algorithm 2 compares favorably with the block-by-block technique of Linz [13]; his results are slightly better than those of Algorithm 1. Campbell and Day [3] get more accurate results using their $O(h^6)$, 15 iteration technique.

For Eq. (2), both our algorithms give uniformly better results than those of Day [4], [5], Linz [12], and a previous method by the author [7]. They are also more accurate than four out of the five methods proposed by Wolfe and Phillips [15]. Their fifth technique, an $O(h^4)$ Runge-Kutta type method, gives better results for $h = .1$, but our algorithms are more accurate for $h = .025$.

Corollary 3 tells us that for qth order accuracy in the application of Algorithm 1, it is sufficient to use a quadrature rule of order $q - i + 1$ to approximate

$$\int_{x_0}^{x} \frac{d^i}{dx^i} F\,ds, \qquad i = 1, 2, \ldots, q.$$

Likewise, Corollary 5 implies that for Algorithm 2, rules of order $q - i$ can be used. To verify these results experimentally, the two algorithms were each modified such that for a qth order method, all integrations were performed using qth order quadrature rules exclusively. Equations (1), (2), and (3) were then solved with these modified algorithms. The results are presented in Tables 4, 5, and 6. The data strongly indicates that no loss in accuracy occurs when the minimal order integration rule is used. In fact, the standard algorithm sometimes gives better results, as can be observed by comparing Table 3 with Table 6.

In Section IV, it was demonstrated that the error term in the application of Algorithm 1 to an equation in which $p = 0$ can be expressed in the form $y_n = y(x_n) + h^q e(x_n) + O(h^{q+1})$. This result implies that the techniques of Richardson's extrapolation can be used to combine two $O(h^q)$ values for $y(x_n)$, yielding an $O(h^{q+1})$ value. If we assume that the $O(h^{q+1})$ term has an asymptotic expansion of

the form $h^{q+1}d(x) + O(h^{q+2})$, the process can be repeated. In order to illustrate the use of this technique, we solved Eq. (1), the only one of our examples to which Theorem 6 can be applied directly, by Algorithm 1 for values of h equal to .2, .1, .05, .025, and .0125. For a given value of q, four successive extrapolations can be performed. Table 7 presents typical results, namely for the case $q = 4$. The entries in the column representing the first extrapolation should be approximations to $y(1)$ of an order of h one greater than the corresponding initial approximation, and an examination of the table bears this out. In addition, each succeeding extrapolation seems to increase the order of approximation. Similar extrapolations performed upon the results of applying the Algorithm to Eqs. (2) and (3) have yielded similar increases in accuracy. This would indicate that an expansion analogous to that in Theorem 6 exists in those cases ($p \geqslant 1$) for which we do not give a proof.

TABLE 1. *Errors for Eq.* (1) *at* $x = 1.0$ ($y(1) = 1.0$)

	Algorithm 1			Algorithm 2		
q	h = .1	h = .05	h = .025	h = .1	h = .05	h = .025
1	5.64×10^{-4}	1.45×10^{-4}	3.67×10^{-5}	3.38×10^{-2}	1.67×10^{-2}	8.29×10^{-3}
2	1.43×10^{-3}	3.71×10^{-4}	9.42×10^{-5}	2.73×10^{-4}	7.03×10^{-5}	1.77×10^{-5}
3	6.51×10^{-6}	7.33×10^{-7}	8.68×10^{-8}	7.48×10^{-6}	5.11×10^{-7}	3.35×10^{-8}
4	2.58×10^{-6}	1.61×10^{-7}	1.00×10^{-8}	5.17×10^{-7}	3.48×10^{-8}	2.21×10^{-9}

TABLE 2. *Errors for Eq.* (2) *at* $x = 1.0$ ($y(1) = e$)

	Algorithm 1			Algorithm 2		
q	h = .1	h = .05	h = .025	h = .1	h = .05	h = .025
1	4.15×10^{-1}	2.28×10^{-1}	1.20×10^{-1}	2.60×10^{-1}	1.36×10^{-1}	6.98×10^{-2}
2	4.22×10^{-2}	1.18×10^{-2}	3.10×10^{-3}	2.54×10^{-2}	6.77×10^{-3}	1.75×10^{-3}
3	3.52×10^{-3}	4.90×10^{-4}	6.48×10^{-5}	2.34×10^{-3}	3.10×10^{-4}	4.00×10^{-5}
4	2.51×10^{-4}	1.77×10^{-5}	1.18×10^{-6}	1.42×10^{-4}	9.53×10^{-6}	6.17×10^{-7}

TABLE 3. *Errors for Eq.* (3) *at* $x = 1.0$ ($y(1) = 8.41 \times 10^{-1}$)

	Algorithm 1			Algorithm 2		
q	h = .1	h = .05	h = .025	h = .1	h = .05	h = .025
1	4.01×10^{-2}	2.01×10^{-2}	1.01×10^{-2}	3.85×10^{-2}	1.92×10^{-2}	9.60×10^{-3}
2	1.03×10^{-3}	2.47×10^{-4}	6.01×10^{-5}	9.69×10^{-4}	2.17×10^{-4}	5.06×10^{-5}
3	3.14×10^{-5}	3.96×10^{-6}	4.98×10^{-7}	3.47×10^{-5}	4.40×10^{-6}	5.53×10^{-7}
4	5.26×10^{-7}	3.15×10^{-8}	1.93×10^{-9}	4.42×10^{-7}	2.20×10^{-8}	1.18×10^{-9}

TABLE 4. *Errors for Eq.* (1) *at* $x = 1.0$

	Modified Algorithm 1			Modified Algorithm 2		
q	h = .1	h = .05	h = .025	h = .1	h = .05	h = .025
1	5.64×10^{-4}	1.45×10^{-4}	3.67×10^{-5}	3.38×10^{-2}	1.67×10^{-2}	8.29×10^{-3}
2	5.39×10^{-4}	1.42×10^{-4}	3.63×10^{-5}	2.81×10^{-4}	7.07×10^{-5}	1.77×10^{-5}
3	9.36×10^{-7}	5.42×10^{-8}	3.22×10^{-9}	5.60×10^{-7}	3.61×10^{-8}	2.25×10^{-9}
4	9.36×10^{-7}	5.42×10^{-8}	3.22×10^{-9}	5.60×10^{-7}	3.61×10^{-8}	2.25×10^{-9}

TABLE 5. *Errors for Eq.* (2) *at* $x = 1.0$

	Modified Algorithm 1			Modified Algorithm 2		
q	h = .1	h = .05	h = .025	h = .1	h = .05	h = .025
1	4.51×10^{-1}	2.28×10^{-1}	1.20×10^{-1}	2.60×10^{-1}	1.36×10^{-1}	6.98×10^{-2}
2	3.94×10^{-2}	1.08×10^{-2}	2.84×10^{-3}	1.94×10^{-3}	5.05×10^{-3}	1.28×10^{-3}
3	3.40×10^{-3}	4.69×10^{-4}	6.18×10^{-5}	1.89×10^{-3}	2.46×10^{-4}	3.13×10^{-5}
4	2.55×10^{-4}	1.79×10^{-5}	1.19×10^{-6}	1.30×10^{-4}	8.63×10^{-6}	5.55×10^{-7}

TABLE 6. *Errors for Eq.* (3) *at* $x = 1.0$

	Modified Algorithm 1			Modified Algorithm 2		
q	h = .1	h = .05	h = .025	h = .1	h = .05	h = .025
1	4.01×10^{-2}	2.01×10^{-2}	1.01×10^{-2}	3.85×10^{-2}	1.92×10^{-2}	9.60×10^{-3}
2	1.00×10^{-3}	2.38×10^{-4}	5.79×10^{-5}	1.26×10^{-3}	3.05×10^{-4}	7.50×10^{-5}
3	3.31×10^{-5}	4.27×10^{-6}	5.41×10^{-7}	3.91×10^{-5}	5.13×10^{-6}	6.57×10^{-7}
4	5.07×10^{-7}	2.99×10^{-8}	1.82×10^{-9}	6.56×10^{-7}	3.97×10^{-8}	2.43×10^{-9}

TABLE 7. *Extrapolation Errors for Algorithm* 1, *Example* 1, $x = 1.0, q = 4$

h	Initial	Extrap 1	Extrap 2	Extrap 3	Extrap 4
.2	3.62×10^{-5}				
.1	2.58×10^{-6}	3.40×10^{-7}			
.05	1.61×10^{-7}	1.57×10^{-11}	1.10×10^{-8}		
.025	1.00×10^{-8}	6.19×10^{-11}	6.45×10^{-11}	1.08×10^{-10}	
.0125	6.25×10^{-10}	2.79×10^{-12}	8.80×10^{-13}	1.29×10^{-13}	7.24×10^{-13}

United States Bureau of the Census
Systems Software Division
Washington, D.C. 20233

1. D. BARTON, I. M. WILLER & R. V. ZAHAR, "Taylor series methods for ordinary differential equations," *Mathematical Software*, Academic Press, New York, 1971.

2. J. A. BRAUN & R. E. MOORE, *A Program for the Solution of Differential Equations Using Interval Arithmetic (DIFEQ) for the* CDC 3600 *and* 1604, MRC Report No. 901, Math. Research Center, Univ. of Wisconsin, Madison, 1968.

3. G. M. CAMPBELL & J. T. DAY, "A block by block method for the numerical solution of Volterra integral equations," *BIT*, v. 11, 1971, pp. 120–124. MR 43 #7093.

4. J. T. DAY, "A note on the numerical solution of integro-differential equations," *Comput. J.*, v. 9, 1967, pp. 394–395.

5. J. T. DAY, "On the numerical solution of integro-differential equations," *BIT*, v. 10, 1970, pp. 511–514. MR 43 #5749.

6. A. FELDSTEIN & J. SOPKA, "Numerical methods for nonlinear Volterra integro-differential equations," unpublished (October, 1968).

7. A. GOLDFINE, "An algorithm for the numerical solution of integro-differential equations," *BIT*, v. 12, 1972, pp. 578–580. MR 48 #3281.

8. P. HENRICI, *Discrete Variable Methods in Ordinary Differential Equations*, Wiley, New York, 1962. MR 24 #B1772.

9. G. KEDEM, *Automatic Differentiation of Computer Programs*, MRC Report, Math. Res. Center, Univ. of Wisconsin, Madison. (To appear.)

10. K. KLINGEN & B. GEISLER, *Punching FORMAC Statements*, Inst. for Applied Math., 517 Julich, Germany, 1971.

11. P. LINZ, *The Numerical Solution of Volterra Integral Equations by Finite Difference Methods,* MRC Report No. 825, Math. Res. Center, Univ. of Wisconsin, Madison, 1967.

12. P. LINZ, "Linear multistep methods for Volterra integro-differential equations," *J. Assoc. Comput. Mach.*, v. 16, 1969, pp. 293–301. MR 39 #1143.

13. P. LINZ, "A method for solving nonlinear Volterra integral equations of the second kind," *Math. Comp.*, v. 23, 1969, pp. 595–599. MR 40 #1055.

14. R. TOBEY, J. BAKER, R. CREWS, P. MARKS & K. VICTOR, *PL/1-Interpreter User's Reference Manual,* IBM #360-D-03.33004, 1967.

15. M. A. WOLFE & G. M. PHILLIPS, "Some methods for the solution of non-singular Volterra integro-differential equations," *Comput. J.*, v. 11, 1968/69, pp. 334–336. MR 38 #4065.