MATHEMATICS OF COMPUTATION
VOLUME 44, NUMBER 170
APRIL 1985, PAGES 391-404

Variable Step Size Predictor-Corrector Schemes for Second Kind Volterra Integral Equations

By H. M. Jones* and S. McKee

Abstract. In this paper a family of implicit multistep methods for the solution of Volterra integral equations is derived. These methods together with an explicit Euler predictor permit the use of a variable step size when solving integral equations. Means of controlling the error and stability by varying the step size and the order of the method are described. Extensive numerical results are presented.

1. Introduction. There are many different numerical methods for solving the second kind Volterra integral equation

$$y(t) + \int_0^t K(t, s, y(s))\, ds = g(t). \tag{1.1}$$

For instance, a selection can be found in Delves and Walsh [3] or Baker [1]. As with ordinary differential equations explicit linear multistep methods for integral equations tend to be less numerically stable than implicit ones, a point noted by the authors in a previous paper (Jones and McKee [6]). The main disadvantage of using implicit linear multistep methods is that it is necessary to solve a nonlinear equation at each grid point. Predictor-corrector methods have the merit of being essentially explicit, whilst retaining stability properties somewhere between those of the component explicit and implicit methods.

The purpose of this paper is to derive and develop a family of predictor-corrector methods for solving Volterra integral equations of the second kind. These methods may have variable step size and each member of the family a different order of accuracy. A divided difference notation is used. This allows a higher-order method to be calculated from that of a lower-order by the addition of appropriate differences. The coefficients of the divided differences are calculated from recurrence relations. Each method consists of the sum of two different types of quadrature rule, each of which approximate

$$\int_{t_i}^{t_{i+1}} K(t_k, s, y(s))\, ds \tag{1.2}$$

to the required order. To minimize the number of starting values required, a forward quadrature rule based on the points

$$\{t_i, t_{i+1}, \ldots, t_{i+l}\}$$

Received November 19, 1980; revised June 21, 1982 and November 14, 1983.
1980 *Mathematics Subject Classification*. Primary 65R20.
*Née Williams.

0025-5718/85 $1.00 + $.25 per page

is employed to approximate (1.2) for small i. This is used to integrate up to some point t_ν, which is decided by stability considerations. Then a backward rule based on the points

$$\{t_{i+1}, t_i, \ldots, t_{i+1-l}\}$$

is employed. These rules are derived in a divided difference formulation which allows complete freedom in the choice of step size within a given range. The order of the rule used to integrate between t_i and t_{i+1} is permitted to vary with i and k. We use these methods to obtain a solution to the integral equation to within a user-specified tolerance. This requires the use of an error estimate which is given in Section 4. The techniques for changing the order and step size are also described in this section.

A package based on the methods of this paper was used to solve some 20 test equations (see Williams [14]), 15 of which are given here. The user is required to supply only the problem and the tolerance; then the program automatically changes step size and method in a manner similar to that used by Shampine and Gordon [9]. The package is written in ANSI (1966) FORTRAN and can be found in Jones [5] or Williams [14].

2. Preliminaries. The Volterra integral equation of the second kind

$$y(t) + \int_0^t K(t, s, y(s))\, ds = g(t), \qquad t \in [0, T], \tag{2.1}$$

is assumed to satisfy the conditions for a unique solution (see, for example, Tricomi [11] or Smithies [10]). The grid and explicit and implicit methods are now defined in general terms; in subsequent sections particular values of the coefficients are derived. The interval $[0, T]$ is divided into N subintervals. The grid is not defined a priori and so N is not known initially. The grid points are chosen to satisfy

$$0 = t_0 < t_1 < t_2 < \cdots < t_{N-1} < t_N = T$$

with $t_k - t_{k-1} = h_k = \gamma_k h$, where $0 < \gamma^* \leqslant \gamma_k \leqslant 1$. Thus, each step size $h_k \in [h_{\min}, h]$, where $h_{\min} = \gamma^* h$. At each point t_k (2.1) becomes

$$y(t_k) + \sum_{i=0}^{k-1} \int_{t_i}^{t_{i+1}} K(t_k, s, y(s))\, ds = g(t_k), \qquad 0 < k \leqslant N, \tag{2.2}$$

with $y(t_0) = y_0 = g(t_0)$. Each integral $\int_{t_i}^{t_{i+1}} K(t_k, s, y(s))\, ds$ in (2.2) can be approximated by

$$\sum_{j=\mu}^{\mu+l} \alpha_{kj} K\big(t_k, t_{i+j}, y(t_{i+j})\big),$$

where μ and l are integers such that $i + \mu \geqslant 0$ and $i + \mu + l \leqslant k$, and the α_{kj} are some appropriate quadrature weights. The weights α_{kj} and the integers μ and l will, in general, depend upon i and k. Thus (2.2) can be approximated by

$$y_k + \sum_{i=0}^{k-1} \sum_{j=\mu}^{\mu+l} \alpha_{kj} K_{ki+j} = g_k, \tag{2.3}$$

where

$$y_k \simeq y(t_k), \quad K_{ki+j} = K(t_k, t_{i+j}, y_{i+j}) \quad \text{and} \quad g_k = g(t_k).$$

The formula (2.3) is implicit if $i + \mu + l = k$ for some i and explicit otherwise. When the formula is implicit it can be used as a corrector

$$y_k^c = g_k - \sum_{i=0}^{k-1} a_{ki} K_{ki} - a_{kk} K_{kk}^p,$$

where $K_{kk}^p = K(t_k, t_k, y_k^p)$. Here a_{ki} denotes the coefficients of the corrector obtained by simplifying (2.3).

3. The Family of Methods. The aim of this section is to derive two specific families of quadrature formulae to approximate

$$\int_0^{t_k} f(s)\, ds = \sum_{i=0}^{k-1} \int_{t_i}^{t_{i+1}} f(s)\, ds.$$

We shall approximate $f(s)$ for $s \in [t_i, t_{i+1}]$ by interpolation using either

(i) the 'forward' points $\{t_i, t_{i+1}, \ldots, t_{i+j}\}$, or

(ii) the 'backward' points $\{t_{i+1}, t_i, \ldots, t_{i+1-j}\}$.

In the first case we obtain

$$\int_{t_i}^{t_{i+1}} f(s)\, ds \simeq \sum_{j=0}^{l} w_{ij} f[t_i, t_{i+1}, \ldots, t_{i+j}], \qquad l \in \{1, 2, \ldots, 10\}, \tag{3.1}$$

where

$$f[t_i, t_{i+1}, \ldots, t_{i+j}] = \frac{f[t_i, t_{i+1}, \ldots, t_{i+j-1}] - f[t_{i+1}, t_{i+2}, \ldots, t_{i+j}]}{(t_i - t_{i+j})}, \tag{3.2}$$

$$f[t_i] = f(t_i),$$

and

$$w_{ij} = \begin{cases} h_{i+1}, & j = 0, \\ \displaystyle\int_{t_i}^{t_{i+1}} \prod_{n=0}^{j-1} (s - t_{i+n})\, ds, & j \geqslant 1. \end{cases}$$

The coefficients $w_{ij}, j \geqslant 1$, can be expressed recursively:

$$w_{i0} = h_{i+1}, \quad w_{ij} = h_{i+1} C_{ij-1}^1, \qquad j \geqslant 1,$$

where

$$C_{i0}^q = h_{i+1}/(q+1)!,$$

$$C_{ij}^q = \left(h_{i+1} - \sum_{n=1}^{j} h_{i+n} \right) C_{ij-1}^q - q h_{i+1} C_{ij-1}^{q+1}, \qquad j \geqslant 1. \tag{3.3}$$

In the second case, we obtain

$$\int_{t_i}^{t_{i+1}} f(s)\, ds \simeq \sum_{j=0}^{l} w_{ij}^* f[t_{i+1}, t_i, \ldots, t_{i+1-j}], \qquad l \in \{1, 2, \ldots, 10\}, \tag{3.4}$$

where $f[t_{i+1}, t_i, \ldots, t_{i+1-j}]$ is defined by (3.2) with i and $i + j$ replaced by $i + 1$ and

$i + 1 - j$, respectively. Similarly, we find that

$$w_{i0}^* = h_{i+1}, \quad w_{ij}^* = -h_{i+1}C_{ij-1}^{*1}, \qquad j \geqslant 1,$$

and

$$C_{i0}^{*q} = (-1)^{q+1}h_{i+1}/(q+1)!,$$

$$C_{ij}^{*q} = \left(-h_{i+1} + \sum_{n=1}^{j} h_{i+2-n}\right)C_{ij-1}^{*q} - qh_{i+1}C_{ij-1}^{*q+1}, \qquad j \geqslant 1. \tag{3.5}$$

The recursions (3.3) and (3.5) permit efficient generation of the coefficients of the forward and backward implicit formulae, respectively. A proof of these relations can be found in Williams [14].

Now since we do not wish to employ the points $t_{-1}, t_{-2}, \ldots$ we shall approximate

$$\int_0^{t_k} f(s)\,ds = \sum_{i=0}^{k-1} \int_{t_i}^{t_{i+1}} f(s)\,ds$$

by the forward rule up to some point, say t_ν, and thereafter by the backward rule. Here the choice of ν will be made on numerical stability considerations (see Section 4). Thus, we shall write

$$\int_0^{t_k} f(s)\,ds \simeq \sum_{i=0}^{\nu} \sum_{j=0}^{l} w_{ij} f\left[t_i, t_{i+1}, \ldots, t_{i+j}\right] + \sum_{i=\nu+1}^{k-1} \sum_{j=0}^{l} w_{ij}^* f\left[t_{i+1}, t_i, \ldots, t_{i+1-j}\right]. \tag{3.6}$$

For notational simplicity we write l rather than l_i; in general l will be different at different step numbers.

Examples. I. In the case $l = 1$ (for all i) (3.5) reduces to the composite trapezoidal rule.

II. If $l = 2$ (for all i) and $h_i = h$ for all i, then the following quadrature method results:

$$\begin{aligned}\int_0^{t_k} f(s)\,ds \simeq h \sum_{i=0}^{\nu} &\{ f(t_i) - \tfrac{1}{2}(f(t_i) - f(t_{i+1})) \\ &- \tfrac{1}{12}(f(t_i) - 2f(t_{i+1}) + f(t_{i+2}))\} \\ &+ \sum_{i=\nu+1}^{k-1} \{ f(t_{i+1}) - \tfrac{1}{2}(f(t_{i+1}) + f(t_i)) \\ &- \tfrac{1}{12}(f(t_{i+1}) - 2f(t_i) + f(t_{i-1}))\} \\ = h \{ &\tfrac{5}{12}f_0 + \tfrac{13}{12}f_1 + f_2 + \cdots + f_{\nu-2} + \tfrac{11}{12}f_{\nu-1} + \tfrac{14}{12}f_\nu \\ &+ \tfrac{11}{12}f_{\nu+1} + f_{\nu+2} + \cdots + f_{k-2} + \tfrac{13}{12}f_{k-1} + \tfrac{5}{12}f_k \},\end{aligned}$$

where $f_n = f(t_n)$, $n = 0, 1, \ldots, k$. A predictor is obtained through the Euler method

$$y_k^0 + \sum_{i=0}^{k-1} h_{i+1}K_k[t_i] = g_k \tag{3.7}$$

and the corrector can be obtained from the quadrature formula (3.6)

$$
\begin{aligned}
&y_k^l + \sum_{i=0}^{\nu} \sum_{j=0}^{l} w_{ij} K_k\left[t_i, t_{i+1}, \ldots, t_{i+j}\right] \\
&\quad + \sum_{i=\nu+1}^{k-2} \sum_{j=0}^{l} w_{ij}^* K_k\left[t_{i+1}, t_i, \ldots, t_{i+1-j}\right] \\
&\quad + \sum_{j=0}^{l} w_{k-1j}^* K_k^{l-1}\left[t_k, t_{k-1}, \ldots, t_{k-j}\right] = g_k, \qquad l \in \{1,2,\ldots,10\},
\end{aligned} \tag{3.8}
$$

where

$$
\begin{aligned}
K_k[t_i] &= K(t_k, t_i, y_i), \\
K_k\left[t_i, t_{i+1}, \ldots, t_{i+j}\right] &= K_k\left[t_i, t_{i+1}, \ldots, t_{i+j}; y_i, y_{i+1}, \ldots, y_{i+j}\right] \\
&= \frac{K_k\left[t_i, t_{i+1}, \ldots, t_{i+j-1}\right] - K_k\left[t_{i+1}, t_{i+2}, \ldots, t_{i+j}\right]}{(t_i - t_{i+j})}
\end{aligned}
$$

and

$$
K_k^{l-1}\left[t_k, t_{k-1}, \ldots, t_{k-j}\right] = K_k\left[t_k, t_{k-1}, \ldots, t_{k-j}; y_k^{l-1}, y_{k-1}, \ldots, y_{k-j}\right],
$$

where y_i denotes y_i^m, that is, the approximation obtained after m ($\leqslant 10$) corrections at a previous time step.

These methods are used in a predictor-corrector mode which closely parallels the method of Shampine and Gordon [9] for ordinary differential equations. In each case the next correction is obtained by updating the previous formula with the next highest divided difference, the appropriate coefficients being calculated from the recursion formulae (3.3) and (3.5).

4. Application of the Methods to Integral Equations. We can solve a second kind integral equation by first obtaining a prediction of the solution using the explicit Euler method, and then obtaining a series of corrections using the methods derived in Section 3. It can be proved (see Williams, [14]) that if $h_i = h$ for all i, then the methods will be convergent to $O(h^{m+1})$ as $h \to 0$, where m (a fixed positive integer in this case) denotes the number of corrections employed, each correction using the corrector with one more term than the previous corrector. However to use m corrections we require m starting values, also convergent of order $m + 1$.

In practice we aim to use only the naturally occurring starting value $y(t_0) = g(t_0)$ and to generate values $y_i \simeq y(t_i)$, $i = 1,2,\ldots,N$, such that the required range is covered and $|y_i - y(t_i)|$ is less than some specified tolerance. In order to do this we control the error by changing the step size and order of the method so that some error estimate is kept suitably small.

In allowing the step size to vary we may obtain accurate solutions with small step sizes and low-order methods when we have few values of y_i, but larger step sizes and higher-order methods can be employed when we have more values. We therefore seek some error estimate which will ensure that the approximations to the solution

are for the most part within the required tolerance without the use of unreasonably small step sizes or high-order methods.

Asymptotic error bounds for the global error can be calculated but these are of little value as they are usually so pessimistic, often by several orders of magnitude, that they force the use of unrealistically small step size and unnecessarily high order of method. We therefore seek some other means of error control which depends on the local truncation error, but also takes account of characteristics of the problem which may lead to instability or inaccuracy for other reasons.

First we consider e_i^l, an approximation to the local truncation error involved in using the formula (3.1) or (3.4) to calculate

$$\int_{t_i}^{t_{i+1}} K(t_k, s, y(s))\, ds.$$

Thus (Henrici [4, p. 248]),

$$e_i^l \simeq \begin{cases} w_{il} K_k[t_i, t_{i+1}, \ldots, t_{i+l}], \\ w_{il}^* K_k[t_{i+1}, t_i, \ldots, t_{i+1-l}], \end{cases} \tag{4.1}$$

as appropriate. Further, if we define E_k^l to be the local truncation error of (3.6) after l corrections, we have

$$E_k^l \simeq \sum_{i=0}^{\nu} w_{il} K_k[t_i, t_{i+1}, \ldots, t_{i+l}] + \sum_{i=\nu+1}^{k-1} w_{il}^* K_k[t_{i+1}, t_i, \ldots, t_{i+1-l}]. \tag{4.2}$$

It has been found from extensive computational experiments that the following is a reasonable estimate of the error after l corrections at step k:

$$\tilde{h}|E_k^l|\big(1 + \tilde{h}\big(|C_1| + |C_2|\big)\big),$$

where $\tilde{h}$ is the largest step size used so far and where

$$C_1 = \begin{cases} \dfrac{K(t_k, 0, y_1) - K(t_k, 0, y_2)}{y_1 - y_2}, & y_1 \neq y_2, \\ 0, \quad \text{otherwise}, \end{cases}$$

$$C_2 = \begin{cases} \dfrac{K(t_k, t_k, y_k) - K(t_k, t_k, y_{k-1})}{y_k - y_{k-1}}, & y_k \neq y_{k-1}, \\ 0, \quad \text{otherwise}. \end{cases}$$

We do not attempt to justify the validity of this test theoretically. However, it involves very little extra computation since the divided differences and the quadrature weights w_{il} and w_{il}^* have already been calculated.

The stability of numerical methods for the solution of integral equations is still really an open problem (however, see Lubich [7]). Just as obtaining a realistic error estimate is considerably more difficult for integral equations than it is for ordinary differential equations, so is the analysis of numerical stability. It was Mayers [8] (but see also Baker and Keech [2]) who originally suggested the simple test equation

$$y(t) + \lambda \int_0^t y(s)\, ds = g(t)$$

through analogy with ordinary differential equations. Although this can give some indication as to the stability of the method applied to certain integral equations, it can often be wide of the mark for the same method used to solve other integral equations. Other papers (e.g. Van der Houwen and Wolkenfelt [12] and Jones and McKee [6]) have suggested other more sophisticated test equations but it is probably fair to say that none have been completely satisfactory. Also, even if one did obtain the "true" region of stability, any algorithm which included a test for stability at every step (and possibly every correction) would be very time-consuming. The alternative approach used here is to examine an estimate of the local truncation error at each correction. This estimate E_k^l is obtained easily as shown earlier, (4.2). If E_k^{l+1} is greater than E_k^l this is taken as a sign of instability and no further correction is made.

The numerical stability of composite schemes obtained by using different values of ν and a fixed step size h is compared using the test equation

$$y(t) + \int_0^t (\lambda_0 + \lambda_1(t-s))\, y(s)\, ds = g(t).$$

The largest stability regions occur when ν is as small as possible (see Williams [14]). Thus ν is chosen to be

$$\min\left\{\frac{k+1}{2}, 9\right\},$$

where k is the step number and 9 is the number of back points needed in the case of the order 10 corrector. The user must supply the tolerance, denoted by TOL, within which he requires the solution. Additionally, he may supply the range of mesh spacing and starting value for step size. However, if he does not, the step size range is taken to be $[10^{-5}, 10^{-1}]$, and the initial step size to be $10 \times h_{\min}$. It is necessary to scale the user supplied tolerance in cases when the solution is increasing. We therefore use

$$\text{TOL} = \text{TOL} * (t_k - t_0)/(t_N - t_0)$$

whenever $y_{k-1} > y_{k-2}$. This forces the solution to be more accurate at the beginning of the interval whenever we have an increasing solution.

We note that if $|e_i^l| \leqslant \text{TOL}/k$, then

$$|E_k^l| \leqslant \sum_{i=0}^{k-1} |e_i^l| \leqslant \text{TOL}.$$

The approach is therefore to correct each component of the integral for up to 10 corrections, until either

(i) $|e_i^l| < \text{TOL}/k$ for all i and some l such that $|e_i^{l-1}| > \text{TOL}/k$ for at least one i. (Indeed, if $|e_i^l| < \text{TOL}/k$, then no further corrections are made to the approximation of $\int_{t_i}^{t_{i+1}} K(t_k, s, y(s))\, ds$.)

or

(ii) a stability limit is reached (i.e., $|E_k^l| > |E_k^{l-1}|$).

Suppose that the correction procedure is stopped after q corrections. Then the

estimate is tested to check that

$$\tilde{h}|E_k^q|\big(1 + \tilde{h}\big(|C_1| + |C_2|\big)\big) \leqslant \text{TOL}. \tag{4.3}$$

If this condition is satisfied, then the method proceeds to the next step with h_{k+1} equal to h_k unless q is less than half the allowable order for that step, in which case h_{k+1} is chosen to be $2h_k$ provided this does not exceed the largest allowable step size. If

$$\tilde{h}|E_k^q|\big(1 + \tilde{h}\big(|C_1| + |C_2|\big)\big) > \text{TOL}$$

then h_k is reduced to $h_k/2$ and this step is repeated.

If the minimum allowable step is reached and (4.3) is not satisfied, then the tolerance is increased by multiplying by a factor of 5; the user is of course informed of this in his output.

For a smooth continuous solution, one would expect a good approximation to the solution using a fixed step size and constant order method assuming that these are both suitably chosen. Since the algorithm starts by means of an order h^2 method and small step sizes, one expects that once a suitable step size and order have been reached, then these will remain reasonably constant over the whole of the rest of the interval. However, if instability begins to take effect, then there may be a rapid decrease in step size. Thus a rapid decrease in step size will be permitted, but a rapid increase in order will be used to identify a possible discontinuity. An increase of order greater than four is considered sufficiently significant. If there has been this rapid increase in order from (t_{k-1}, t_k) to (t_k, t_{k+1}), then a discontinuity is suspected between t_{k-1} and t_{k+1}. It is assumed to occur at t_k (i.e., $t_D = t_k$) and the solution is recomputed on (t_{k-1}, t_D). As t_D is approached, small step sizes and lower-order methods are used; these, in effect, assume less continuity in the underlying function. Having reached t_D, a restart is made incorporating an approximation to $\int_{t_0}^{t_D} K(t_k, s, y(s))\, ds$ into $g(t_k)$. Of course, it is then necessary to redetermine ν. On the whole, this procedure works well, although occasionally a discontinuity is identified when it is in fact a rapidly changing continuous solution.

A flow chart (see Figure 1) illustrates the calculation of a typical value y_k. We assume we start with a step size h_k from the previous step, and TOL the tolerance (scaled if necessary). We first make the k kernel evaluations $K(t_k, t_j, y_j)$ for $j = 0, 1, \ldots, k-1$. From these we calculate the prediction, y_k^p, and then $K(t_k, t_k, y_k^p)$. The first correction is automatically made using the first-order divided-differences which will overwrite the kernel values. We then enter the correction loop, correcting each component integral until one of the exit criteria is met. Note that after each correction the whole of the approximation to $\int_{t_{k-1}}^{t_k} K(t_k, s, y(s))\, ds$ is recalculated using the latest value of y_k. After exit from the correction loop we either set up the values for the next step or repeat this step depending on whether (4.3) is satisfied.

For clarity, only the essentials in the flow chart have been included. Both the scaled tolerance and code's ability to change the tolerance should the minimum allowable step size be reached have been omitted.

This section has been concerned with an outline description of stability, error and discontinuity control. More details including suggestions for fine tuning can be found in Williams [14]. The documented package written in ANSI (1966) FORTRAN can be found in Williams [14] or Jones [5].

FLOWCHART

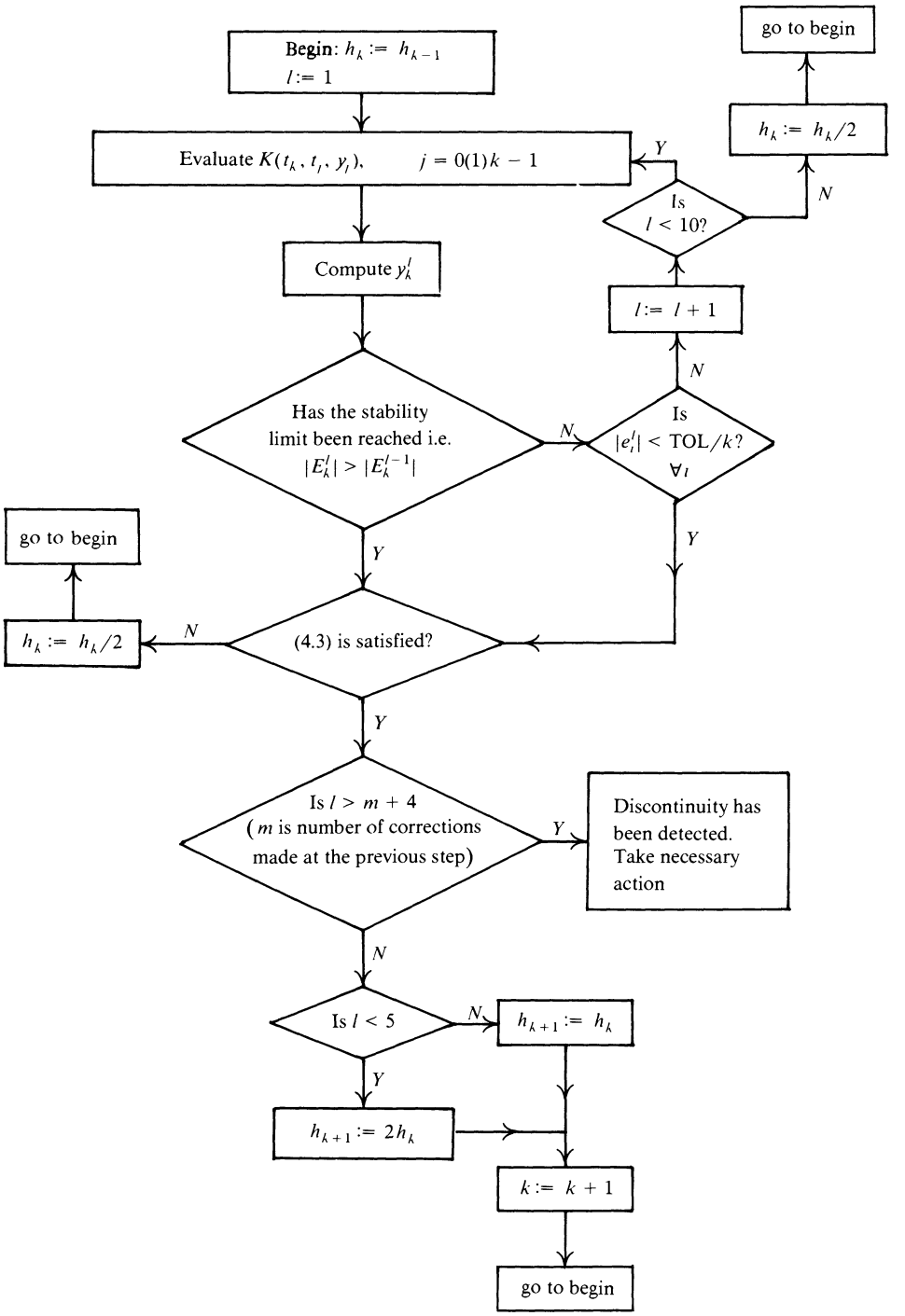

FIGURE 1

5. Numerical Results. A package using these methods applied as described in Sections 4 and 5 has been written, see Jones [5]. This package was tested using a number of different test equations obtained primarily from te Riele [13]. These results are presented here in a condensed form.

Test Examples.

1. $y(t) + \int_0^t \frac{1+t}{1+s} y^2(s)\, ds = \left(1 + (1+t)e^{-10t}\right)^{1/2} + \frac{1+t}{10}\left(10 \log(1+t) + 1 - e^{-10t}\right)$

   Solution: $y(t) = \left(1 + (1+t)e^{-10t}\right)^{1/2}, \quad t \in [0,5]$

2. $y(t) + \int_0^t y^3(s)\, ds = (\sin t + e^{-t})^{1/3} + 2 - \cos t - e^{-t}$

   Solution: $y(t) = (\sin t + e^{-t})^{1/3}, \quad t \in [0, \pi]$

3. $y(t) + \int_0^t (3 + 2(t-s))\, y(s)\, ds = 2t + 3$

   Solution: $y(t) = 4e^{-2t} - e^{-t}, \quad t \in [0,5]$

4. $y(t) - \frac{1}{2}\int_0^t (t-s)^2 e^{-(t-s)} y(s)\, ds = \frac{1}{2} t^2 e^{-t}$

   Solution: $y(t) = \frac{1}{3}\left(1 - e^{-3/2t}\left(\cos \frac{\sqrt{3}}{2} t + \sqrt{3} \sin \frac{\sqrt{3}}{2} t\right)\right), \quad t \in [0,5]$

5. $y(t) + \int_0^t \cos(t-s)\, y(s)\, ds = t + 1 - \cos t$

   Solution: $y(t) = t, \quad t \in [0,5]$

6. $y(t) - \int_0^t t^2 e^{-st} y(s)\, ds = t - 1 + (1+t^2)e^{-t^2}$

   Solution: $y(t) = t, \quad t \in [0,5]$

7. $y(t) + \int_0^t ts\sqrt{y(s)}\, ds = e^{t^2} + t\left(e^{t^2/2} - 1\right)$

   Solution: $y(t) = e^{t^2}, \quad t \in [0,5]$

8. $y(t) - \int_0^t (t-s)\, y(s)\, ds = \sin t$

   Solution: $y(t) = \frac{1}{2}(\sin t + \sinh t), \quad t \in [0,5]$

9. $y(t) - 2\int_0^t \cos(t-s)\, y(s)\, ds = e^t$

   Solution: $y(t) = e^t(1+t)^2, \quad t \in [0,5]$

10. $y(t) - \int_0^t y(s)\, ds = \cos t$

    Solution: $y(t) = \frac{1}{2}(e^t + \sin t + \cos t), \quad t \in [0,5]$

11. $y(t) + \int_0^t \cosh(t-s)\, y(s)\, ds = \sinh t$

    Solution: $y(t) = \frac{2}{\sqrt{5}} \sinh\left(\frac{\sqrt{5}}{2} t\right) e^{-t/2}, \quad t \in [0,5]$

12. $y(t) + \int_0^t \max\{s, y(s)\}\, ds = 0$

Solution: $y(t) = \begin{cases} \frac{1}{2}t^2, & t \in [0,2] \\ 2e^{t-2}, & t \in (2,5] \end{cases}$

13. $y(t) + \int_0^t y(s)\, ds = \begin{cases} 1 + t, & t \in [0,1] \\ \frac{1}{2}(1 + t^2) + t, & t \in (1,5] \end{cases}$

Solution: $y(t) = \begin{cases} 1, & t \in [0,1] \\ t, & t \in [1,5] \end{cases}$

14. $y(t) + \int_0^t (t - s - y(s))\, ds = \begin{cases} \frac{1}{2}t^2, & t \in [0,1] \\ \frac{1}{2}t^2 + t, & t \in (1,5] \end{cases}$

Solution: $y(t) = \begin{cases} 0, & t \in [0,1] \\ 1, & t \in (1,5] \end{cases}$

15. $y(t) + \int_0^t e^{y(s)}\, ds = \begin{cases} te + 1, & t \in [0,1] \\ e + t - 1, & t \in (1,5] \end{cases}$

Solution: $y(t) = \begin{cases} 1, & t \in [0,1] \\ 0, & t \in (1,5] \end{cases}$

The results of solving these equations using the package described in this paper are summarized in Table 1. More detailed results including 5 more test equations can be found in Williams [14]. These results were obtained using the Oxford University ICL 2900 computer.

TABLE 1

Problem number	Step size range		Number of steps	Maximum number of corrections	Tolerance at beginning	Tolerance at end	|Maximum error|
1	10^{-5}	10^{-1}	94	9	10^{-5}	10^{-5}	2.0×10^{-7}
2	10^{-5}	10^{-1}	73	7	10^{-5}	2.5×10^{-4}	3.8×10^{-5}
2	10^{-5}	5×10^{-2}	86	6	10^{-5}	10^{-5}	1.9×10^{-6}
3	10^{-5}	10^{-1}	182	9	10^{-5}	10^{-5}	4.4×10^{-7}
4*	10^{-5}	10^{-1}	112	7	10^{-5}	10^{-5}	2.5×10^{-5}
5	10^{-5}	10^{-1}	120	6	10^{-5}	10^{-5}	1.3×10^{-6}
6*	10^{-5}	10^{-1}	76	9	10^{-5}	1.25×10^{-3}	1.8×10^{-3}
6	10^{-5}	5×10^{-2}	131	7	10^{-5}	2.5×10^{-4}	8.7×10^{-5}
7*	10^{-5}	10^{-1}	145	10	10^{-5}	10^{-5}	6.9×10^{-3}
8*	10^{-5}	10^{-1}	393	5	10^{-4}	10^{-4}	2.2×10^{-4}
9*	10^{-5}	10^{-1}	141	10	10^{-5}	10^{-5}	1.4×10^{-4}
10	10^{-5}	10^{-1}	71	8	10^{-5}	10^{-5}	3.5×10^{-7}
11	10^{-5}	10^{-1}	104	10	10^{-5}	6.25×10^{-3}	1.2×10^{-3}
11	10^{-5}	10^{-1}	91	8	10^{-4}	1.25×10^{-2}	2.0×10^{-3}
12	10^{-5}	10^{-1}	144	7	10^{-5}	10^{-5}	9.8×10^{-6}
13	10^{-5}	10^{-1}	98	5	10^{-5}	10^{-5}	6.7×10^{-6}
13	10^{-5}	10^{-1}	93	9	10^{-6}	10^{-6}	2.0×10^{-7}
14*	10^{-5}	10^{-1}	246	8	10^{-5}	10^{-5}	1.3×10^{-5}
15*	10^{-5}	10^{-1}	89	8	10^{-5}	10^{-5}	4.2×10^{-5}

Except for those test equations marked by * the maximum error was always less than the tolerance, although rarely less by more than an order of magnitude and often considerably closer. In problem 4 and 6 there was only one step where |error| > TOL. In problem 7, 53 steps had |error| > TOL and for all those steps $y(t) > 1.7 \times 10^{+6}$. In problem 8 there were 23 steps with |error| > TOL and for all these $y(t) > 15$. In problem 9 there were 36 steps with |error| > TOL in which case $y(t) > 380$. In problems 14 and 15 all the steps with |error| > TOL were near the discontinuities.

A detailed history of problem 12 is give for tolerances of 10^{-4}, 10^{-5} and 10^{-6}.

Problem 12.

$$y(t) + \int_0^t \max(s, y(s))\, ds = 0$$

$$y(t) = \begin{cases} t^2/2, & \text{for } t \in [0,2] \\ 2e^{t-2}, & \text{for } t \in [2,5] \end{cases}$$

$K(t, t, y) > 0$, for all t.

TOLERANCE = 10^{-4}
HMIN = 10^{-4} HMAX = 10^{-1}

t	solution	step size	order	|error|
0.539	0.1453	0.1	4	5.0×10^{-7}
1.039	0.5397	0.1	4	5.0×10^{-7}
1.539	1.1843	0.1	4	5.0×10^{-7}
2.009	2.0181	0.01	4	9.6×10^{-6}
2.541	3.4362	0.1	6	6.8×10^{-6}
3.041	5.6655	0.1	7	1.2×10^{-5}
3.541	9.3407	0.1	8	2.0×10^{-5}
4.041	15.4004	0.1	8	3.4×10^{-5}
4.541	25.3909	0.1	8	6.7×10^{-5}
5.0	40.1711	0.059	8	1.1×10^{-4}

TOLERANCE = 10^{-5}
HMIN = 10^{-5} HMAX = 10^{-1}

t	solution	step size	order	|error|
0.5035	0.1268	0.1	4	2.0×10^{-8}
1.0035	0.5035	0.1	4	3.5×10^{-8}
1.5035	1.1303	0.1	4	4.5×10^{-8}
2.0010	2.0020	0.0025	4	5.2×10^{-7}
2.4950	3.2810	0.0410	6	7.8×10^{-7}
2.9865	5.3638	0.0410	6	1.3×10^{-6}
3.5190	9.1353	0.0410	6	2.2×10^{-6}
4.0105	14.9344	0.0410	6	3.6×10^{-6}
4.5020	24.4148	0.0410	7	5.9×10^{-6}
5.000	40.1711	0.0410	7	9.8×10^{-6}

TOLERANCE = 10^{-6}
HMIN = 10^{-5} HMAX = 10^{-1}

t	solution	step size	order	|error|
0.5035	0.1268	0.1	4	5.0×10^{-9}
1.0035	0.5035	0.1	4	5.0×10^{-9}
1.5035	1.1303	0.1	4	5.0×10^{-9}
2.0010	2.0020	0.0025	4	5.3×10^{-7}
2.4950	3.2810	0.0205	10	6.8×10^{-7}
3.0070	5.4748	0.0205	10	1.1×10^{-6}
3.4985	8.9501	0.0205	10	1.8×10^{-6}
3.9900	14.6317	0.0205	10	3.1×10^{-6}
4.5020	24.4148	0.0205	10	5.0×10^{-6}
5..000	40.1711	0.0064	10	8.2×10^{-6}

TOL	largest |error|	no. points |error| > TOL
10^{-4}	1.1×10^{-6}	2/92
10^{-5}	9.8×10^{-6}	0/144
10^{-6}	8.2×10^{-6}	103/212

N. B. For the last set of results the maximum order (10th) method had been used for 132/212 of the steps—this would be unlikely for a well-behaved problem—the calculations were therefore repeated with a lower range of step sizes:

$$\text{HMIN} = 10^{-6} \quad \text{and} \quad \text{HMAX} = 0.5 \times 10^{-1}$$

The two sets of solutions agree to 5 decimal places.

TOLERANCE = 10^{-6}
HMIN = 10^{-6} HMAX = 0.5×10^{-1}

t	solution	step size	order	|error|
0.4820	0.1162	0.05	4	3×10^{-10}
1.0320	0.5325	0.05	4	1×10^{-10}
1.5329	1.1736	0.05	4	1×10^{-10}
2.0008	2.0016	0.0025	4	9.9×10^{-8}
2.4935	3.2792	0.0328	8	1.3×10^{-7}
2.9851	5.3560	0.0328	8	2.2×10^{-7}
3.5094	9.0476	0.0328	8	3.6×10^{-7}
4.0009	14.7911	0.0328	8	6.0×10^{-7}
4.4924	24.1894	0.0328	8	9.8×10^{-7}
5.0000	40.1711	0.0161	8	1.6×10^{-6}

largest error = 1.6×10^{-6}
number of points = 15/185.
|ERROR| > TOL

6. Concluding Remarks. This paper has been concerned with the derivation of variable step size predictor-corrector schemes for nonlinear second kind integral equations. Many of the ideas follow Shampine and Gordon [9] for ordinary differential equations. For instance, recurrence relations are devised for the efficient calculation of the quadrature weights. The implementation of these methods has been discussed. In particular, it has been pointed out how both stability and error are controlled. The code also takes special measures whenever there appears to be a

discontinuity in the solution or its derivatives. Other features include an automatic means of relaxing the user supplied tolerance if either this will not be attained or will take prohibitively long to attain. A package incorporating these methods has been tested using fifteen diverse problems and a summary of the results has been included. The main aim was to achieve reliability in the sense that if the results are not as accurate as requested, then an indication to this effect will be given and less accurate results will be obtained, and with few exceptions, this has been achieved.

Acknowledgments. This work is part of the first author's D. Phil. thesis at the University of Oxford and she would like to acknowledge a grant from the Science and Engineering Research Council. The second author's share of the work was done whilst holding a Central Electricity Generating Board Research Fellowship and he would like to acknowledge their financial support. They would also like to thank the referee and G. F. Miller for their helpful comments.

Division of Information Technology and Computing
National Physical Laboratory
Teddington, Middlesex, England

University Computing Laboratory
8-11 Keble Road
Oxford, England

1. C. T. H. BAKER, *The Numerical Treatment of Integral Equations*, Oxford Univ. Press, Oxford, 1977.

2. C. T. H. BAKER & M. S. KEECH, "Stability regions in the numerical treatment of Volterra integral equations," *SIAM J. Numer. Anal.*, v. 15, 1978, pp. 394–417.

3. L. M. DELVES & J. WALSH (Eds.), *Numerical Solution of Integral Equations*, Oxford Univ. Press, Oxford, 1974.

4. P. HENRICI, *Elements of Numerical Analysis*, Wiley, New York, 1964.

5. H. M. JONES, "A variable step variable order package for solving Volterra integral equations of the second kind," (to be submitted to *ACM Trans. Math. Software*).

6. H. M. JONES & S. MCKEE, "The numerical stability of multistep methods for convolution Volterra integral equations: nonsingular equations," *Comput. Math. Appl.*, v. 8, 1982, pp. 291–303.

7. CH. LUBICH, "On the stability of linear multistep methods for Volterra integral equations of the second kind," in *Treatment of Integral Equations by Numerical Methods* (C. T. H. Baker and G. F. Miller, eds.), Academic Press, London-New York, 1982.

8. D. F. MAYERS, "Equations of Volterra type," in *Numerical Solution of Ordinary and Partial Differential Equations* (L. Fox, ed.), Pergamon Press, New York, 1962, pp. 165–173.

9. L. F. SHAMPINE & M. K. GORDON, *Computer Solution of Ordinary Differential Equations*, Freeman, San Francisco, 1975.

10. F. SMITHIES, *Integral Equations: Cambridge Tracts in Mathematics and Mathematical Physics*, No. 49, Cambridge Univ. Press, New York, 1958.

11. F. G. TRICOMI, *Integral Equations*, Interscience, New York, 1957.

12. P. J. VAN DER HOUWEN & P. H. M. WOLKENFELT, "On the stability of multistep formulas for Volterra integral equations of the second kind," *Computing*, v. 24, 1980, pp. 341–347.

13. H. J. J. TE RIELE, *Proposal for a Test Set for Comparing Accuracy and Efficiency of Algorithms for the Numerical Solution of Volterra Integral Equations of the Second Kind With Nonsingular Kernels*, M/C Rep. Math. Centrum, Amsterdam, 1978.

14. H. M. WILLIAMS, *Variable Step-Size Predictor-Corrector Schemes for Volterra Second Kind Integral Equations*, D.Phil. thesis, University of Oxford, Oxford, 1980.