SIAM J. SCI. STAT. COMPUT.
Vol. 6, No. 3, July 1985

003

FAST NUMERICAL SOLUTION OF NONLINEAR VOLTERRA CONVOLUTION EQUATIONS*

E. HAIRER†, CH. LUBICH‡ AND M. SCHLICHTE†

Abstract. Numerical methods for general Volterra integral equations of the second kind need $O(n^2)$ kernel evaluations and $O(n^2)$ additions and multiplications. Here it is shown how the effort can be reduced for nonlinear convolution equations. Exploiting the convolution structure, most numerical methods need only $O(n)$ kernel evaluations. With the use of Fast Fourier Transform techniques only $O(n(\log n)^2)$ additions and multiplications are necessary. The paper closes with numerical examples and comparisons.

Key words. Volterra integral equation, convolution, fast Fourier transform, Runge-Kutta method

1. Introduction. We consider nonlinear second kind Volterra integral equations of convolution type

$$y(x)=f(x)+\int_{x_0}^{x} k(x-s)g(s, y(s))\,ds, \qquad x_0 \leqq x \leqq \bar{x}. \tag{1}$$

The kernel k and the functions f, g are assumed to be sufficiently smooth on $[x_0, \bar{x}]$, so that the solution $y(x)$ is smooth, too. Problems of this type appear in biology, e.g. neurophysiology (an der Heiden [7]), epidemiology (Hethcote-Tudor [9]), and in the treatment of special hyperbolic differential equations (Friedlander [5]). Further applications are given in Corduneanu [3].

Usually the starting point for numerical methods is the more general equation

$$y(x)=f(x)+\int_{x_0}^{x} K(x, s, y(s))\,ds, \qquad x_0 \leqq x \leqq \bar{x}. \tag{2}$$

If the integration interval is discretized with n gridpoints, then algorithms for (2) need $O(n^2)$ evaluations of K. For the special equation (1), which appears most often in applications, the number of function evaluations can be reduced to $O(n)$ k- and g-evaluations for suitably chosen methods, e.g. extended Runge-Kutta methods and linear multistep methods.

In § 2 we describe the extended classical Runge-Kutta method. If this method is implemented straightforwardly, it requires still $O(n^2)$ additions and multiplications. It is shown in § 3, the central part of this paper, that this overhead can be reduced to $O(n(\log n)^2)$. In § 4 the asymptotic expansion of the global error is used for improving the accuracy of the numerical solution and estimating the global error. Some numerical results are given in § 5. Comparisons of our code VOLCON with existing codes are presented.

The ideas of this article are not restricted to Runge-Kutta methods, they are also applicable to multistep methods. Although we present the theory only for the scalar equation (1), it also pertains to systems of equations (1), to integrodifferential equations and to weakly singular integral equations of convolution type.

* Received by the editors June 7, 1983, and in final form January 15, 1984.

† Institut für Angewandte Mathematik, Universität Heidelberg, Im Neuenheimer Feld 293, D-6900 Heidelberg 1, Germany.

‡ Institut für Mathematik und Geometrie, Universität Innsbruck, Technikerstr. 13, A-6020 Innsbruck, Austria.

2. The extended classical Runge–Kutta method. Extended Runge-Kutta methods have been introduced by Pouzet [11]. The classical 4th order method, applied to (2), is given by

$$
\begin{aligned}
&y_n = \tilde{F}_n(x_n),\\
&\tilde{F}_n(x) = f(x) + \frac{h}{6}\sum_{j=0}^{n-1}\left\{K(x, x_j, Y_1^{(j)}) + 2K\left(x, x_j+\frac{h}{2}, Y_2^{(j)}\right)\right.\\
&\qquad\qquad \left. + 2K\left(x, x_j+\frac{h}{2}, Y_3^{(j)}\right) + K(x, x_j+h, Y_4^{(j)})\right\},\\
&Y_1^{(j)} = \tilde{F}_j(x_j),\\
&Y_2^{(j)} = \tilde{F}_j\left(x_j+\frac{h}{2}\right) + \frac{h}{2}K\left(x_j+\frac{h}{2}, x_j, Y_1^{(j)}\right),\\
&Y_3^{(j)} = \tilde{F}_j\left(x_j+\frac{h}{2}\right) + \frac{h}{2}K\left(x_j+\frac{h}{2}, x_j+\frac{h}{2}, Y_2^{(j)}\right),\\
&Y_4^{(j)} = \tilde{F}_j(x_j+h) + hK\left(x_j+h, x_j+\frac{h}{2}, Y_3^{(j)}\right).
\end{aligned}
\tag{3}
$$

Here h denotes the stepsize, and y_n approximates the solution at $x_n = x_0 + nh$. This method is convergent with a global error of $O(h^4)$. For a proof see Pouzet [11] or Hairer-Lubich-Nørsett [6]. The positions, where the kernel $K(x, s, y)$ has to be evaluated, lie very regularly in the (x, s)-plane. They are indicated with crosses in Fig. 1.

Since these points lie on lines parallel to the diagonal and the x-axis, the number of function evaluations can be reduced for the convolution equation (1). Here the method (3) reads ($\tilde{F}_0(x) = f(x)$)

$$y_n = \tilde{F}_n(x_n), \tag{4a}$$

$$
\begin{aligned}
\tilde{F}_n(x) = f(x) &+ \frac{h}{6}k(x-x_0)g(x_0, Y_1^{(0)})\\
&+ \frac{h}{3}\sum_{j=0}^{n-1} k\left(x-x_j-\frac{h}{2}\right)\left[g\left(x_j+\frac{h}{2}, Y_2^{(j)}\right) + g\left(x_j+\frac{h}{2}, Y_3^{(j)}\right)\right]\\
&+ \frac{h}{6}\sum_{j=1}^{n-1} k(x-x_j)[g(x_j, Y_4^{(j-1)}) + g(x_j, Y_1^{(j)})]\\
&+ \frac{h}{6}k(x-x_n)g(x_n, Y_4^{(n-1)}),
\end{aligned}
\tag{4b}
$$

$$
\begin{aligned}
&Y_1^{(j)} = \tilde{F}_j(x_j),\\
&Y_2^{(j)} = \tilde{F}_j\left(x_j+\frac{h}{2}\right) + \frac{h}{2}k\left(\frac{h}{2}\right)g(x_j, Y_1^{(j)}),\\
&Y_3^{(j)} = \tilde{F}_j\left(x_j+\frac{h}{2}\right) + \frac{h}{2}k(0)g\left(x_j+\frac{h}{2}, Y_2^{(j)}\right),\\
&Y_4^{(j)} = \tilde{F}_j(x_j+h) + hk\left(\frac{h}{2}\right)g\left(x_j+\frac{h}{2}, Y_3^{(j)}\right).
\end{aligned}
\tag{4c}
$$

It is seen that for the computation of y_n only $2n+1$ k- and f-evaluations and $4n$

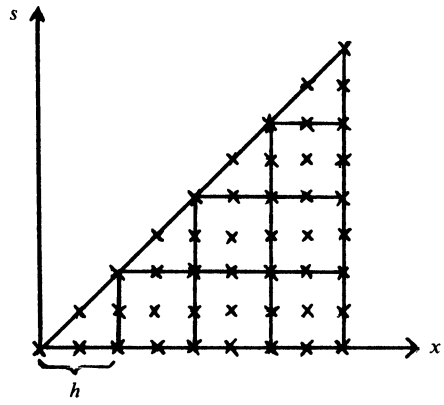

FIG. 1

g-evaluations are necessary. If the formulas (4) are implemented straightforwardly, $O(n^2)$ additions and multiplications still are needed.

3. Fast computation of the lag-terms. We describe in this section, how the overhead for method (4) can be reduced using FFT-techniques.

Assume that $Y_i^{(j)}$ for $i=1,\cdots,4$ and $j=0,\cdots,r-1$ are computed directly by (4) (step I of the algorithm). With the notation

$$\kappa_j = k((j+1)h/2), \qquad j=0,1,\cdots,4r-1,$$

$$\gamma_0 = \frac{h}{6} g(x_0, Y_1^{(0)}),$$

$$\gamma_{2j+1} = \frac{h}{3}\left[g\left(x_j+\frac{h}{2}, Y_2^{(j)}\right) + g\left(x_j+\frac{h}{2}, Y_3^{(j)}\right)\right], \qquad j=0,\cdots,r-1,$$

$$\gamma_{2j} = \frac{h}{6}[g(x_j, Y_4^{(j-1)}) + g(x_j, Y_1^{(j)})], \qquad j=1,\cdots,r-1,$$

$$\gamma_{2r} = \frac{h}{6} g(x_r, Y_4^{(r-1)}),$$

$$\gamma_j = 0 \quad \text{for } j=2r+1,\cdots,4r-1,$$

the lag-term $\tilde{F}_r(x)$ becomes

$$\tilde{F}_r(x) = f(x) + (\kappa * \gamma)_{2r+j} \quad \text{for } x = x_r + (j+1)h/2 \text{ and } j=0,1,\cdots,2r-1.$$

Here the convolution of the two $4r$-dimensional sequences $\kappa=(\kappa_j)$ and $\gamma=(\gamma_j)$ is given by

$$(\kappa * \gamma)_m = \sum_{i=0}^{4r-1} \kappa_{m-i}\gamma_i.$$

This convolution can be computed efficiently using the fast Fourier transform (FFT). For a description see Henrici [8] and the references given there. These computations are illustrated in Fig. 2.

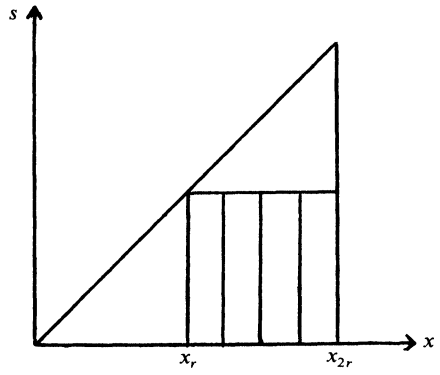

FIG. 2

Each vertical line in the square of Fig. 2 represents the lag-term $\tilde{F}_r(x)$ as given by (4b) at the corresponding x-value. Formula (7) permits to compute the lag-terms of this square simultaneously with FFT. We have thus obtained step II in Fig. 3.

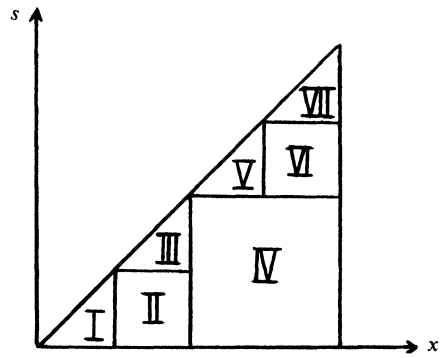

FIG. 3

For the computation of step III we observe that method (4) applied to

$$y(x) = \tilde{F}_n(x) + \int_{x_n}^{x} k(x-s)g(s, y(s))\, ds, \qquad x \geqq x_n \tag{8}$$

(with $n = r$) yields the same numerical solution for $x \geqq x_r$ as when applied to (1). This permits us to compute $Y_i^{(j)}$ for $j = r, \cdots, 2r-1$ since the required $\tilde{F}_r(x)$-values are known by (7).

In step IV we employ the same arguments as in step II with r replaced by $2r$ and compute

$$\tilde{F}_{2r}(x) \quad \text{for } x = x_{2r} + (j+1)h/2,\ j = 0, 1, \cdots, 4r-1$$

simultaneously by FFT.

The steps V, VI and VII are now performed by applying the steps I, II and III to the integral equation (8) with $n = 2r$.

Proceeding by induction, we arrive at Fig. 4, where each square symbolizes the computation of one convolution by FFT.

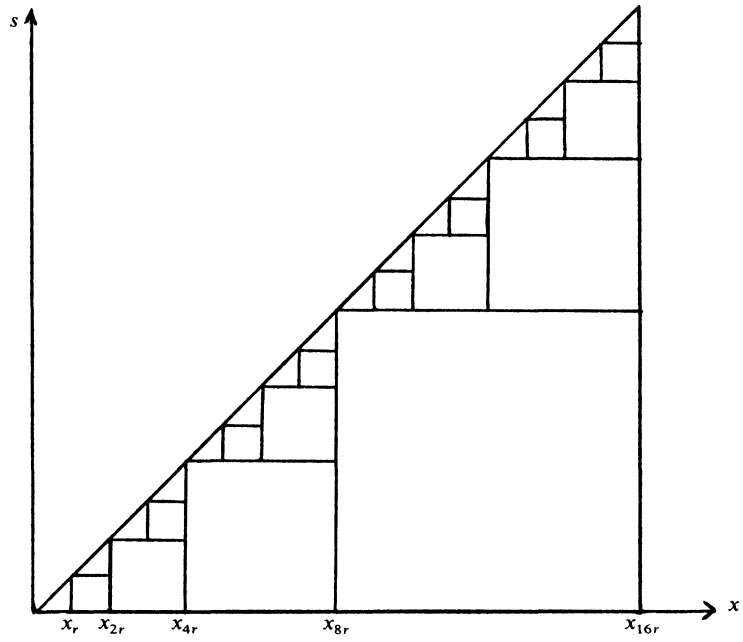

FIG. 4

Since the computation of the convolution of two n-vectors requires $O(n \log n)$ additions and multiplications, our algorithm needs only

$$O\left(n \log n + 2\left(\frac{n}{2}\log\frac{n}{2}\right) + 4\left(\frac{n}{4}\log\frac{n}{4}\right) + \cdots\right) + O(n) = O(n(\log n)^2)$$

additions and multiplications.

In Fig. 5 we compare the straightforward computation of (4) with our algorithm.

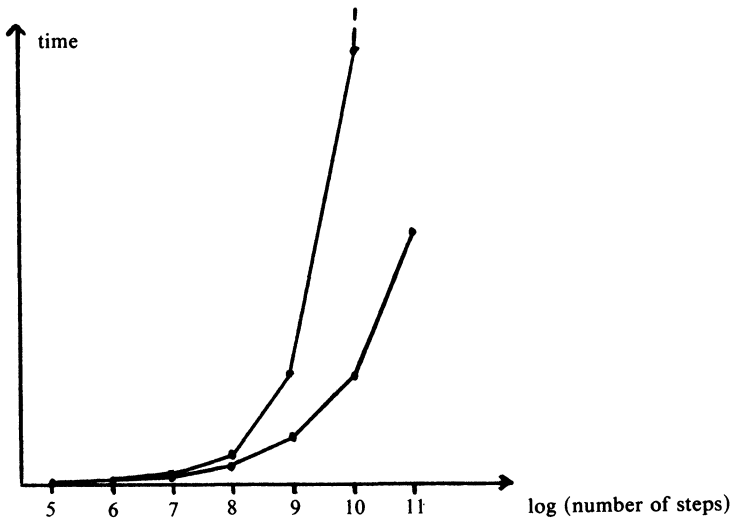

FIG. 5

Here we have chosen $r = 2^5$, which turned out to be optimal in numerical experiments. The difference in computer time is independent of the integral equation to be solved.

Although both algorithms are mathematically equivalent, that one, which uses the FFT, causes in general less rounding errors, since it needs fewer additions and multiplications. This is confirmed by numerical computations.

4. Global error estimation. Consider the method (4) and denote its numerical solution by $y(x, h) = y_n$ if $x = x_0 + nh$, in order to indicate its dependence on the stepsize. It has been shown in [6] that the global error has an asymptotic expansion of the form

$$y(x) - y(x, h) = e_4(x)h^4 + e_5(x)h^5 + \cdots + e_N(x)h^N + O(h^{N+1}).$$

The numerical solution at x is computed for the stepsizes h, $h/2$, $h/4$, $h/8, \cdots$ and is denoted by $T_{i0} = y(x, h/2^i)$. Then the extrapolation tableau (cf. [4])

$$\begin{array}{lll} T_{00} & & \\ T_{10} & T_{11} & \\ T_{20} & T_{21} & T_{22} \\ \cdot & \cdot & \cdot \end{array}$$

is calculated according to the formula (Aitken-Neville algorithm)

$$T_{ik} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{2^{k+3} - 1}, \qquad i \geqq k \geqq 1.$$

Since by this procedure the leading term in the asymptotic expansion of $T_{i,k-1}$ is cancelled, we have

$$y(x) - T_{ik} = \gamma_{ik} e_{k+4}(x) h^{k+4} + O(h^{k+5}). \tag{9}$$

Observe that the leading term in (9) equals that of $T_{i,k+1} - T_{ik}$, which is a numerically available estimation of (9).

This motivates the following strategy: For a prescribed tolerance TOL, calculate the extrapolation tableau until we have for some indices i, k

$$|T_{i,k+1} - T_{ik}| \leqq \mathrm{TOL}. \tag{10}$$

By the above considerations this difference estimates the error of T_{ik}. The more accurate value $T_{i,k+1}$ is then accepted as a numerical approximation to $y(x)$.

Observe that the k- and f- values, which are needed for the computation of $y(x, h)$, can be used again for the computation of $y(x, h/2)$. Furthermore, the Fourier transforms of the kernel-values, which were computed before, can be used to reduce the effort for the computation of the kernel Fourier transforms for the stepsize $h/2$.

5. Numerical experiments and comparisons. Based on the theoretical considerations of §§ 2-4 the authors have written a FORTRAN subroutine VOLCON for the numerical solution of a scalar equation (1). This program can be obtained on request from the authors.

In this section we give some numerical results and compare them to those of the codes VE1 (due to Bownds [2]) and ORION (due to Bader-Kunkel [1]). We omit comparisons with the codes VOLTEX (due to Hock [10]) and INTSOL (due to Williams-McKee [12]), since ORION turned out to be competitive or even superior to these codes in extensive numerical tests (see [1]).

Our numerical experiments have been run on the IBM 370/168 of the University of Heidelberg in FORTRAN double precision (with about 16 decimal digits). The numerical examples to be presented are documented in terms of the quantities

TOL-prescribed absolute precision
TIME-execution time of the subroutine on the IBM 370/168
NFEV-number of f-evaluations
NKEV-number of kernel evaluations
NGEV-number of g-evaluations
ERRACT-actual absolute error

Problem 1. As a first test example we have taken the equation

$$y(x) = \cos x - 2\int_0^x (x-s+2)^{-2}(y(s)+y^3(s))\,ds$$

on the intervals [0, 10] and [0, 40]. The exact solution at the endpoints is $y(10) = -0.4718905296$ and $y(40) = -0.6501311013$. These values have been obtained numerically using different codes with very stringent tolerances.

A linear version of this equation is formula (8b) in Friedlander [4]. We have introduced a nonlinearity, since for linear convolution equations the FFT can be used directly without applying the techniques of § 3.

In Tables 1 and 2 the numerical results of VOLCON are presented. ERREST denotes the difference of the best to the second-best approximation in the extrapolation tableau (cf. § 4 formula (10)).

TABLE 1
Results of VOLCON *for Problem* 1 (XEND = 10).

TOL	TIME	NFEV	NKEV	NGEV	ERREST	ERRACT
10^{-2}	0.016	70	70	204	0.159×10^{-4}	0.440×10^{-5}
10^{-4}	0.017	74	74	220	0.159×10^{-4}	0.440×10^{-5}
10^{-6}	0.049	146	146	508	0.138×10^{-6}	0.493×10^{-8}
10^{-8}	0.144	322	322	1,208	0.770×10^{-10}	0.127×10^{-9}

TABLE 2
Results of VOLCON *for Problem* 1 (XEND = 40).

TOL	TIME	NFEV	NKEV	NGEV	ERREST	ERRACT
10^{-2}	0.121	262	262	780	0.902×10^{-4}	0.212×10^{-5}
10^{-4}	0.122	266	266	796	0.902×10^{-4}	0.212×10^{-5}
10^{-6}	0.334	530	530	1,852	0.733×10^{-7}	0.230×10^{-6}
10^{-8}	0.860	1,090	1,090	4,088	0.361×10^{-8}	0.153×10^{-8}

In Tables 3 and 4 the results of the code VE1 are presented. This code approximates the kernel k by a degenerate one and solves the resulting system of differential equations by an ODE-solver. The parameter DIM, which has to be specified by the user, denotes the dimension of the ODE. It has been chosen as the minimal number such that ERREST, which represents the error caused by approximating the kernel is less than TOL.

TABLE 3
Results of VE1 *for Problem* 1 (XEND = 10).

TOL	TIME	NFEV	NKEV	NGEV	DIM	ERREST	ERRACT
10^{-2}	0.13	104	58	408	8	0.331×10^{-2}	0.359×10^{-3}
10^{-4}	0.34	204	112	808	11	0.856×10^{-4}	0.128×10^{-4}
10^{-6}	0.92	387	242	1,540	16	0.377×10^{-6}	0.335×10^{-6}
10^{-8}	1.96	646	382	2,576	20	0.896×10^{-8}	0.418×10^{-8}

TABLE 4
Results of VE1 *for Problem* 1 (XEND = 40).

TOL	TIME	NFEV	NKEV	NGEV	DIM	ERREST	ERRACT
10^{-2}	0.49	320	112	1,272	11	0.835×10^{-2}	0.256×10^{-1}
10^{-4}	1.45	615	242	2,452	16	0.735×10^{-4}	0.427×10^{-2}
10^{-6}				fail			
10^{-8}				fail			

We observe that for small integration intervals VE1 gives correct results, but the computer time is significantly higher than for VOLCON. For large intervals the kernel cannot be easily approximated by polynomials, so that VE1 produces incorrect results for TOL = 10^{-2} and 10^{-4}. No value DIM ($\leqq 25$) could be found such that ERREST is smaller than TOL for TOL $\leqq 10^{-6}$.

In order to demonstrate that it is worthwhile to exploit the convolution structure in (1), we give in Tables 5 and 6 the results of ORION. This code is written to solve general Volterra integral equations (2).

TABLE 5
Results of ORION *for Problem* 1 (XEND = 10).

TOL	TIME	NFEV	NKEV	ERRACT
10^{-2}	0.17	37	1,562	0.156×10^{-3}
10^{-4}	0.46	61	4,277	0.142×10^{-6}
10^{-6}	1.25	100	11,012	0.261×10^{-7}
10^{-8}	2.92	142	23,904	0.533×10^{-9}

TABLE 6
Results of ORION *for Problem* 1 (XEND = 40).

TOL	TIME	NFEV	NKEV	ERRACT
10^{-2}	1.09	128	12,593	0.383×10^{-3}
10^{-4}	4.25	242	45,813	0.128×10^{-4}
10^{-6}	11.03	345	107,017	0.616×10^{-7}
10^{-8}	29.04	503	270,117	0.221×10^{-8}

ORION gives the correct results. The number of K-evaluations, and therefore also the computer time, is very high, since no use of the convolution structure is made.

Problem 2. Equations of the following type arise in the analysis of neural networks with post-inhibitory rebound. The equation below has been modelled after a qualitative

description in an der Heiden [7] on pages 4, 9 and 10.

$$y(x) = 1 + \int_0^x (x-s)^3(4-x+s)\, e^{-x+s} \frac{y^4(s)}{1+2y^2(s)+2y^4(s)}\, ds.$$

The exact solution at $x = 10$ is $y(10) = 1.25995582337$. This value has again been obtained numerically using different codes with very stringent tolerances. Table 7 gives the results of VOLCON.

TABLE 7
Results of VOLCON *for Problem* 2 (XEND = 10).

TOL	TIME	NFEV	NKEV	NGEV	ERREST	ERRACT
10^{-2}	0.017	70	70	204	0.341×10^{-4}	0.203×10^{-4}
10^{-4}	0.018	74	74	220	0.341×10^{-4}	0.203×10^{-4}
10^{-6}	0.052	162	162	504	0.949×10^{-6}	0.134×10^{-5}
10^{-8}	0.368	578	578	2,168	0.252×10^{-9}	0.476×10^{-10}

The kernel of Problem 2 is exactly decomposable. Problem 2 is therefore equivalent to a 5-dimensional system of ordinary differential equations (cf. [2]). Solving this ODE with the efficient code DIFEX1 (due to Deuflhard [4]), we obtain the results of Table 8.

TABLE 8
Results of DIFEX1 *for Problem* 2 (XEND = 10).

TOL	TIME	NGEV	ERRACT
10^{-2}	0.014	67	0.776×10^{-2}
10^{-4}	0.039	203	0.611×10^{-3}
10^{-6}	0.048	256	0.408×10^{-5}
10^{-8}	0.070	380	0.116×10^{-7}

It is seen that VOLCON and DIFEX1 are competitive for TOL $\geqq 10^{-6}$. For integral equations with degenerate kernel, whose corresponding differential equation has a dimension greater than 5, VOLCON becomes superior. This is due to the fact that the dimension is significant for the overhead of the ODE-solver. For lower dimensions or for TOL $< 10^{-6}$ an ODE-solver like DIFEX1 is to be preferred.

REFERENCES

[1] G. BADER AND P. KUNKEL, *An adaptive multistep method for the solution of second kind Volterra integral equations*, in preparation.

[2] J. M. BOWNDS, *Theory and performance of a subroutine for solving Volterra integral equations*, Computing, 28, (1982), pp. 317-332.

[3] C. CORDUNEANU, *Integral Equations and Stability of Feedback Systems*, Academic Press, New York, 1973.

[4] P. DEUFLHARD, *Order and stepsize control in extrapolation methods*, Numer. Math., 41 (1983), pp. 399-422.

[5] F. G. FRIEDLANDER, *The reflexion of sound pulses by convex parabolic reflectors*, Proc. Cambr. Philos. Soc., 37 (1941), pp. 134-149.

[6] E. HAIRER, CH. LUBICH AND S. P. NØRSETT, *Order of convergence of one-step methods for Volterra integral equations of the second kind*, SIAM J. Numer. Anal., 20 (1983), pp. 569-579.

[7] U. AN DER HEIDEN, *Analysis of Neutral Networks*, Lecture Notes in Biomathematics 35, Springer-Verlag, Berlin-Heidelberg-New York, 1980.

[8] P. HENRICI, *Fast Fourier methods in computational complex analysis*, SIAM Rev., 21 (1979), pp. 481-527.

[9] H. W. HETHCOTE AND D. W. TUDOR, *Integral equation models for endemic infectious diseases*, J. Math. Biol., 9 (1980) pp. 37-47.

[10] W. HOCK, *An extrapolation method with step size control for nonlinear Volterra integral equations*, Numer. Math., 38 (1981), pp. 155-178.

[11] P. POUZET, *Etude en vue de leur traitement numérique des équations intégrales de type Volterra*, Rev. Français Traitement Information (Chiffres) 6 (1963), pp. 79-112.

[12] H. M. WILLIAMS AND S. MCKEE, *Variable step-size predictor-corrector schemes for second kind Volterra integral equations*, Math. Comp., to appear.