An Adaptive Method for the Numerical Solution of Volterra Integral Equations

JOSHUA H. GORDIS
Department of Mechanical Engineering
Naval Postgraduate School
Code ME/Go
Monterey, CA 93943
U.S.A.
BENY NETA
Department of Mathematics
Naval Postgraduate School
Code MA/Nd
Monterey, CA 93943
U.S.A.
bneta@nps.navy.mil http://www.math.nps.navy.mil/ bneta

Abstract

In this paper we introduce an adaptive method for the solution of Volterra integral equations of the second kind. We demonstrate the benefit of adaptivity and apply the idea to the nonlinear equation arises in transient structural synthesis.

Keywords: Volterra integral equations, adaptive, earthquake.

1 Introduction

In this paper we extend our work on the adaptive solution of Fredholm integral equations of the second kind [1, 2] to Volterra type integral equations. In [1] , Neta and Nelson have discussed the adaptive solution of

$$x(t) = \int_0^1 k(t,\tau)\, x(\tau)\, d\tau + g(t), \quad t\epsilon\,[0,\,1], \tag{1}$$

where $k(t,\tau)$ is a regular kernel. The method is based on the trapezoidal rule for obtaining the numerical solution of (1). The idea is to start with a given number of equally spaced points (or a given mesh). The solution at this stage is obtained by solving a linear system of algebraic equations. The program then decides if the mesh needs to be refined and where. This is done in such a way that both the change in the approximate solution and its gradient are equidistributed. Based upon the experience described by Dwyer, Smooke and Kee [3] for boundary-value problems, one perhaps can expect this often to be more efficient than the alternative of equidistributing arc-length or local truncation error.

This idea was extended to singular kernels. Neta [2] considered the two most common types of singularities

$$\text{i.} \qquad k(t,\,\tau) = R(t,\,\tau)\,\log|\,t-\tau\,|\,, \tag{2}$$

$$\text{ii.} \quad k(t,\,\tau) = R(t,\,\tau)\,|\,t-\tau\,|^{-\alpha}\,,\ \alpha < 1, \tag{3}$$

where R$(t,\,\tau)$ is non singular. The adaptive method is based on a product integration rule (see Atkinson [5]) for obtaining the numerical solution of (1) with kernel (2) or (3).

2 Solution of Volterra Equations

Volterra integral equations, given by

$$x(t) = \int_0^t k(t,\tau)\, x(\tau)\, d\tau + g(t), \quad t\epsilon\,[0,\,1], \tag{4}$$

are slightly different. Here the upper limit of the integral term is not a constant. This difference manifests itself in the numerical approximation. In the case of Fredholm integral equations of the second kind, the coefficient matrix is dense. For Volterra integral equations the matrix becomes lower triangular. In fact, in the case of convolution kernels, i.e. $k(s,t) = k(s-t)$, the matrix is called semicirculant. This means that the system can be solved directly without the need to factor. In fact, this means that one doesn't even have to write a system of equations, but solve the problem by marching in time. The approximation of (4) is given by

$$x_j = \sum_{i=0}^{j-1} \int_{t_i}^{t_{i+1}} k(t_j, \tau) x(\tau) d\tau + g_j \qquad (5)$$

Using the trapezoidal rule,

$$x_j = \sum_{i=0}^{j-1} \frac{1}{2} h_{i+1} \left(k_{j\,i+1} x_{i+1} + k_{j\,i} x_i \right) + g_j, \qquad j = 1, 2, \ldots, N \qquad (6)$$

$$x_j = \sum_{i=0}^{j-1} \frac{1}{2} h_{i+1} k_{j\,i} x_i + \sum_{i=1}^{j} \frac{1}{2} h_i k_{j\,i} x_i + g_j, \qquad j = 1, 2, \ldots, N \qquad (7)$$

$$x_j \left(1 - \frac{1}{2} h_j k_{j\,j} \right) = \sum_{i=0}^{j-1} \frac{1}{2} \left(h_{i+1} + h_i \right) k_{j\,i} x_i + g_j, \qquad (8)$$

$$j = 1, 2, \ldots, N$$

with the understanding that $h_0 = 0$. Clearly from (4), the value of $x(0)$ is $g(0)$, so

$$x_0 = g_0 \qquad (9)$$

In general

$$x_j \left(1 - \frac{1}{2} h_j k_{j\,j} \right) = \frac{1}{2} \sum_{i=0}^{j-1} \left(h_{i+1} + h_i \right) k_{j\,i} x_i + g_j \qquad (10)$$

The sum on the right have values of x_i already computed, since $i \leq j-1$.

This is, of course, not the only way to approximate Volterra integral equations. See, for example, the excellent book by Peter Linz [7] and references there. Westreich and Cahlon [6] have analyzed five different methods for the numerical solution of nonlinear Volterra integral equations of the form

$$x(t) = \int_a^t k(t,\tau) h(t, \tau, x(\tau)) d\tau + g(t), \qquad a \leq t \leq b \qquad (11)$$

One of the two methods recommended is based on piecewise quadratic polynomial interpolation of $h(t, \tau, x(\tau))$ for all points except the first where we must use a linear interpolation.

3 Adaptive Solution

The adaptive solution method for Fredholm integral equations can be applied to Volterra type. The difference is that we have a lower triangular matrix to solve at each stage. This is cheaper since there is no need for factorization.

Since the solution of the problem is done by marching in time (similar to initial value problems), it makes more sense to use variable step instead of adaptivity suggested above. We approximate the solution at time t by using a time step to ensure that the local error is not to exceed a given tolerance. In order to implement variable step, we need to have a strategy for step doubling and step halving. This will depend on the local truncation error. Thus it is necessary to estimate the local truncation error from a knowledge of the numerical solution, and use this information to control the step size (see Gear [8]). We measure the error by the absolute difference between the coarse and fine solutions at the current point.

It is suggested that the errors be computed and divided by the maximum value of $x(t)$ so far. The worst case scaled error is restricted to be less than a given tolerance. In fact it is advised to take 99% of that, so that one wouldn't have to change the step very often.

To compute x_j^f, we only subdivide the last

interval. Thus

$$\begin{aligned} x_j^f &= \sum_{i=0}^{j-2} \int_{t_i}^{t_{i+1}} k(t_j-\tau)x(\tau)d\tau \\ &+ \int_{t_{j-1}}^{t_{j-1/2}} k(t_j-\tau)x(\tau)d\tau \\ &+ \int_{t_{j-1/2}}^{t_j} k(t_j-\tau)x(\tau)d\tau + g_j \end{aligned} \tag{12}$$

Using the trapezoidal rule,

$$\begin{aligned} x_j^f &= \sum_{i=0}^{j-2} \frac{1}{2}\left(k_{j\,i+1}x_{i+1} + k_{j\,i}x_i\right)h_{i+1} \\ &+ \frac{1}{2}\left(k_{j\,j-1/2}x_{j-1/2} + k_{j\,j-1}x_{j-1}\right)\frac{1}{2}h_j \\ &+ \frac{1}{2}\left(k_{j\,j}x_j + k_{j\,j-1/2}x_{j-1/2}\right)\frac{1}{2}h_j + g_j \end{aligned} \tag{13}$$

$$\begin{aligned} x_j^f &= \sum_{i=0}^{j-2} \frac{1}{2}h_{i+1}k_{j\,i}x_i + \sum_{i=1}^{j-1} \frac{1}{2}h_i k_{j\,i}x_i \\ &+ \frac{1}{2}\left(k_{j\,j-1}x_{j-1} + 2k_{j\,j-1/2}x_{j-1/2}\right. \\ &+ \left. k_{j\,j}x_j\right)\tfrac{1}{2}h_j + g_j \end{aligned} \tag{14}$$

$$\begin{aligned} x_j^f\left(1-\frac{1}{4}h_j k_{j\,j}\right) &= \sum_{i=0}^{j-2} \frac{1}{2}\left(h_i+h_{i+1}\right)k_{j\,i}x_i \\ &+ \frac{1}{4}\left(h_j+2h_{j-1}\right)k_{j\,j-1}x_{j-1} \\ &+ \frac{1}{2}h_j k_{j\,j-1/2}x_{j-1/2} + g_j \end{aligned} \tag{15}$$

The problem is the value of $x_{j-1/2}$. This is done in the same fashion.

$$\begin{aligned} x_{j-1/2} &= \sum_{i=0}^{j-2} \int_{t_i}^{t_{i+1}} k(t_{j-1/2}-\tau)x(\tau)d\tau \\ &+ \int_{t_{j-1}}^{t_{j-1/2}} k(t_{j-1/2}-\tau)x(\tau)d\tau \\ &+ g_{j-1/2} \end{aligned} \tag{16}$$

Using the trapezoidal rule,

$$\begin{aligned} x_{j-1/2} &= \sum_{i=0}^{j-2} \frac{1}{2}\left(k_{j-1/2\;i+1}x_{i+1} + k_{j-1/2\;i}x_i\right)h_{i+1} \\ &+ \frac{1}{2}\left(k_{j-1/2\;j-1/2}x_{j-1/2} + k_{j-1/2\;j-1}x_{j-1}\right)\frac{1}{2}h_j \\ &+ g_{j-1/2} \end{aligned} \tag{17}$$

$$\begin{aligned} x_{j-1/2} &= \sum_{i=0}^{j-2} \frac{1}{2}h_{i+1}k_{j-1/2\;i}x_i \\ &+ \sum_{i=1}^{j-1} \frac{1}{2}h_i k_{j-1/2\;i}x_i \\ &+ \frac{1}{4}\left(k_{j-1/2\;j-1}x_{j-1} + k_{j-1/2\;j-1/2}x_{j-1/2}\right)h_j \\ &+ g_{j-1/2} \end{aligned} \tag{18}$$

$$\begin{aligned} x_{j-1/2}\left(1-\frac{1}{4}h_j k_{j-1/2\;j-1/2}\right) &= \sum_{i=0}^{j-2} \frac{1}{2}\left(h_i+h_{i+1}\right)k_{j-1/2\;i}x_i \\ &+ \frac{1}{2}\left(\frac{1}{2}h_j + h_{j-1}\right)k_{j-1/2\;j-1}x_{j-1} \\ &+ g_{j-1/2} \end{aligned} \tag{19}$$

The procedure is then to evaluate x_j^c from (10), then substitute (19) into (15) to get x_j^f.

Algorithm:

- Initialize
  - Set $t_0 = 0$, choose Δt
  - Set error tolerance and minimum time step
  - Compute $g(t_0)$ and $x(t_0) = g(t_0)$
- First time step
  - Evaluate coarse solution at $t_0 + \Delta t$ using (10)
  - Evaluate fine solution using half the time step using (15)

- Compare the coarse and fine solutions

  While the error greater than the tolerance and Δt greater than some minimum value we refine

- Save results

- Next time step

  Here we loop

  - Get coarse solution using (10),
  - Get fine solution using (15) and (19).
  - If error is less than tolerance double the time step and recompute coarse and fine solutions
  - If error is greater than tolerance and time step larger than a minimum step refine
  - Save results

- Plot numerical and analytic solutions

4 Nonlinear Equations

The Nonlinear Volterra equation considered her is

$$x(t) = \int_0^t k(t,\tau) f(t,\tau,x(\tau),\dot{x}(\tau)) d\tau + g(t) \tag{20}$$

A discussion of this and more general nonlinear equations can be for example in [6]. They consider the convergence and compare several numerical schemes. Our interest lie in the transient structural synthesis, so the kernel is $k(t-\tau)$ with $k(0) = 0$. To get the coarse solution we have to solve

$$x_j = \frac{1}{2} \sum_{i=0}^{j-1} (h_{i+1} + h_i) k_{j-i} f_{ji}(x_i, \dot{x}_i) + g_j \tag{21}$$

where $f_{ji}(x_i, \dot{x}_i) = f(t_j, t_i, x_i, \dot{x}_i)$ Notice that even though the problem is nonlinear we don't have to solve a nonlinear equation at each step since $k(0) = 0$.

For the fine solution, we need the intermediate value of $x_{j-1/2}$. We can generalize (19) and (15). We now get

$$\begin{aligned} x_j^f &= \sum_{i=0}^{j-2} \frac{1}{2} (h_i + h_{i+1}) k_{j-i} f_{ji}(x_i, \dot{x}_i) \\ &+ \frac{1}{4} (h_j + 2h_{j-1}) k_1 f_{j\,j-1}(x_{j-1}, \dot{x}_{j-1}) \\ &+ \frac{1}{2} h_j k_{1/2} f_{j\,j-1/2}(x_{j-1/2}, \dot{x}_{j-1/2}) + g_j \end{aligned} \tag{22}$$

$$\begin{aligned} x_{j-1/2} &= \sum_{i=0}^{j-2} \frac{1}{2} (h_i + h_{i+1}) k_{j-1/2-i} f_{j-1/2\ i}(x_i, \dot{x}_i) \\ &+ \frac{1}{2} \left(\frac{1}{2} h_j + h_{j-1} \right) k_{1/2} f_{j-1/2\ j-1}(x_{j-1}, \dot{x}_{j-1}) \\ &+ g_{j-1/2} \end{aligned} \tag{23}$$

5 Numerical Experiments

In this section we describe some of the numerical experiments performed in solving the linear Volerra integral equation. In all cases, we chose the right hand side $g(t)$ in such a way that we know the exact solution. This exact solution is used only to show that the numerical solution obtained with our method is correct.

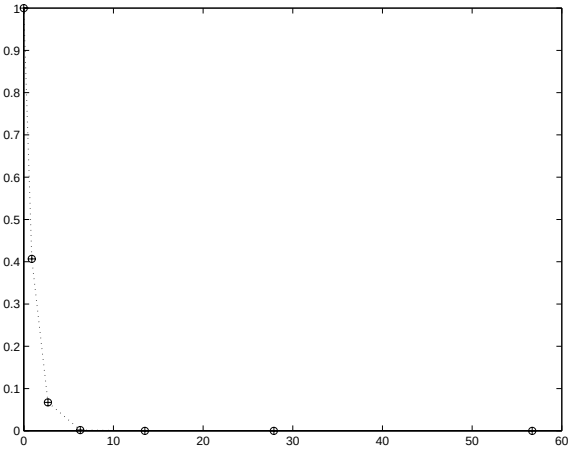

Figure 1: Numerical and Analytic Solutions for Example 1

The first example has an exact solution $x(t) = e^{-t}$ and kernel $k(t) = e^{-t}$. It can be

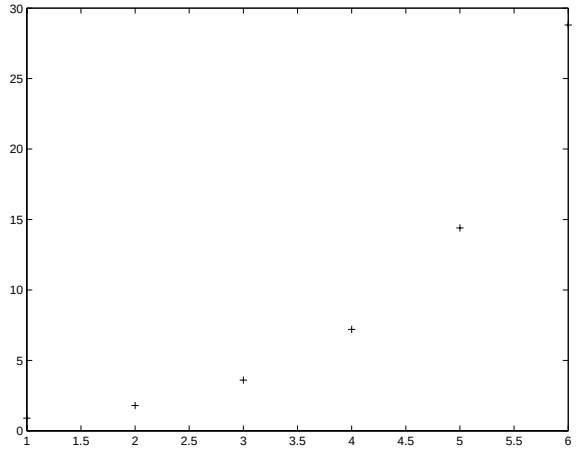

Figure 2: Step Sizes for Example 1

shown that in that case $g(t) = e^{-t}(1-t)$. We ran our code with error tolerance of 10^{-4} for $0 < t < 50$. Since the solution decays exponentially, we found that the step size was doubled at every step, see Figure 2. As can be seen in Figure 1 the numerical and analytic solutions are identical.

The second example has a quadratic polynomial solution $x(t) = t^2 + 2t + 1$ and the same decaying exponential kernel. The right hand side is then $g(t) = e^{-t} + 2t$. This problem is slightly more complicated, but the solution is very accurate, as can be seen in Figure 3. Notice that the step size was doubled once at $t = 1$ (see Figure 4.)

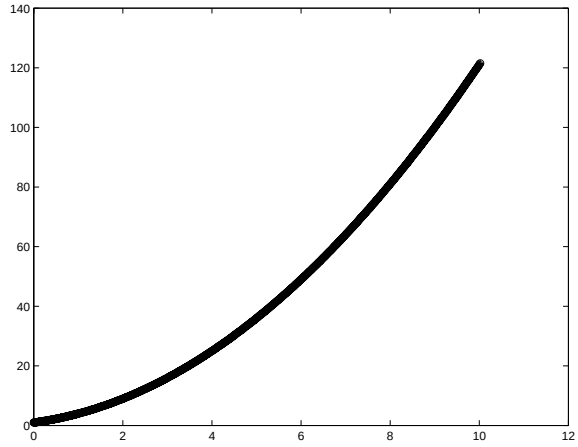

Figure 3: Numerical and Analytic Solutions for Example 2

In the next example, we took the same kernel, but $x(t) = \cos t$. Thus the problem has

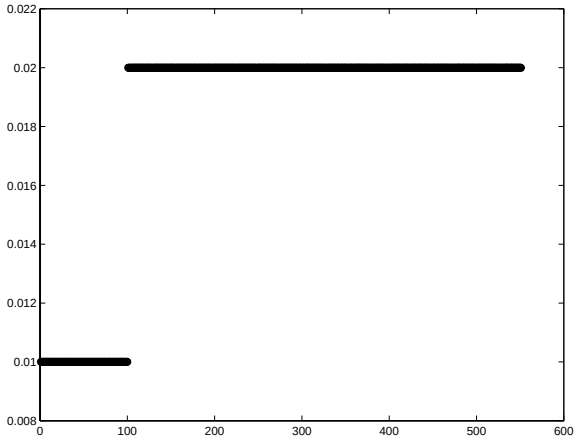

Figure 4: Step Sizes for Example 2

$g(t) = \frac{1}{2}\left(\cos t - \sin t + e^{-t}\right)$. In this example the agreement is excellent (see Figure 5) for all t (we ran the problem for $t \leq 10$.) It is interesting to see that the code manages to double the step in cases and halve the step for other t, see Figure 6. The smallest step size is 0.01 and the largest is 0.08.

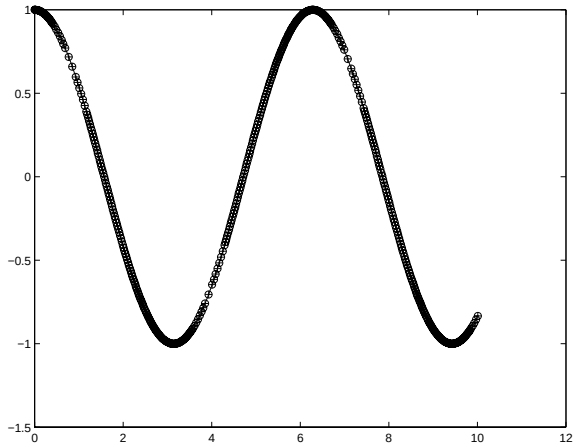

Figure 5: Numerical and Analytic Solutions for Example 3

In our fourth example we changed the kernel from a decaying exponential to a bounded trigonometric function. We have $k(t) = \sin t$ and the exact solution $x(t) = t + 1$. The right hand side is then $g(t) = \sin t + \cos t$. The solution agrees very well with the analytic.

In the last example we allowed the kernel to grow linearly, i.e. $k(t) = t$. We chose $x(t) = \cos t$, which means that the right hand side is $g(t) = 2\cos t - 1$. Here the numerical solution

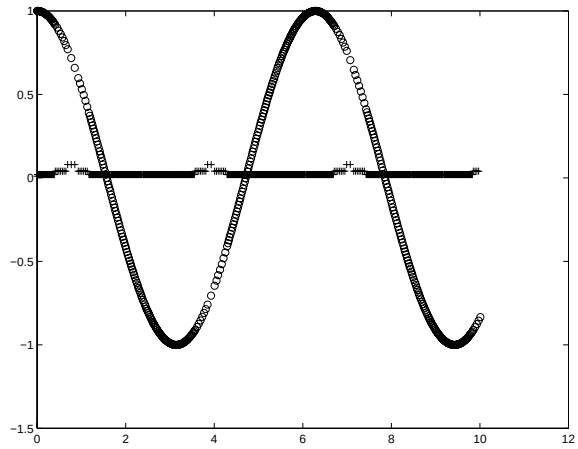

Figure 6: Step Sizes for Example 3 (Min Δt = .01 and Max Δt = .08)

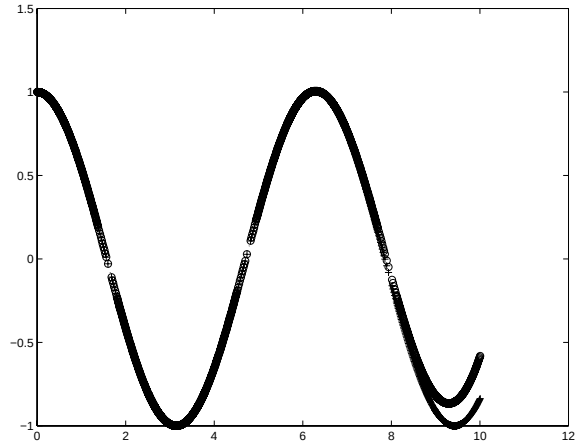

Figure 7: Numerical and Analytic Solutions for Example 5. Notice the divergence of the two curves around $t = 9$.

was able to follow the analytic solution in the beginning, but later they diverge (see Figure 7). When taking error tolerance of 10^{-5}, the agreement was good for t slightly greater than 8. If we reduce the error tolerance to 10^{-4}, then the solutions diverge at t slightly smaller than 6. An increase in the error tolerance cut the execution time by a factor of 15.

6 Conclusions

An adaptive method was developed and tested for the numerical solution of Volterra integral equations of the second kind. The application to a nonlinear Volterra equation arising in earthquake hazards mitigation is discussed.

Acknowledgement

The authors gratefully acknowledge the support of the National Science Foundation, Dr. S. C. Liu, Program Director, Earthquake Hazard Mitigation.

References

[1] B. Neta and P. Nelson, Adaptive method for the numerical solution of Fredholm integral equations of the second kind. Part I: Regular kernels, *Applied Mathematics and Computation*, **21**, (1987), 171-184.

[2] B. Neta, Adaptive method for the numerical solution of Fredholm integral equations of the second kind. Part II: singular kernel, in *Numerical Solution of Singular Integral Equations* (A. Gerasoulis and R. Vichnevetsky, Eds.), 1984, pp. 68-72.

[3] H. A. Dwyer, M. D. Smooke, and R. J. Kee, Adaptive gridding for finite difference solutions to heat and mass transfer problems, *Appl. Math. Comput.*, **10/11**, (1982), 339-356.

[4] C. E. Pearson, A numerical method for ordinary differential equations of boundary-layer type, *J. Math. Phys*, **47**, (1968), 134-154.

[5] K. E. Atkinson, A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind, SIAM, Philadelphia, 1976.

[6] D. Westreich, and B. Cahlon, Numerical solution of Volterra integral equations with continuous or discontinuous terms, *J. Inst. Maths Applics*, **26**, (1980), 175-186.

[7] P. Linz, Analytical and Numerical Methods for Volterra Integral Equations, SIAM, Philadelphia, 1985.

[8] C. W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, NJ, 1971.