Solution of integral equations by hybrid computation

by G. A. BEKEY
and J. C. MALONEY
University of Southern California
Los Angeles, California
and
R. TOMOVIC*
Belgrade University
Belgrade, Yugoslavia

INTRODUCTION

The mathematical description of many problems of engineering interest contains integral equations. Typical of a large class of such problems is the Fredholm integral equation of the second kind,

$$y(x) = f(x) + \lambda \int_a^b K(x,t)\ y(t)\ dt \qquad (1)$$

where f(x) and the kernel K(x,t) are given functions, a and b are constants, λ is a parameter and y(x) is to be found. From a computational point of view, equations of this type may be considered as problems in two dimensions, where one dimension (t) is the dummy variable of integration. For digital computer solution, both variables must be discretized. For analog computer solution, it is possible to perform continuous integration with respect to the variable t for a fixed value of x and perform a scanning process to obtain step changes in the second variable. In either case, the solution is iterative and results in a sequence of functions $\{y_n(x)\}$, $n=1,2,\ldots$ which, under certain conditions, converge to the true solution y(x) as n increases.

It is evident that such a sequential solution, with a two dimensional array K(x,t), may be extremely time consuming for pure digital solution. On the other hand, the scanning and iteration procedures, which require storage of the successive approximation to the solution, do not lend themselves to pure analog computation. Rather, the problems require a hybrid combination of high speed, repetitive integration, memory, and flexible control logic.

The advantage of using such hybrid computational methods for the solution of integral equations was realized quite early. A special purpose computer for solution of integral equations was proposed by Wallman in 1950[1]. Basically, the computer technique consisted of replacing the integration with respect to two independent variables by scanning at two different rates. These original proposals for iterative solution of integral equations were based on the classical or Neumann method.[2] In 1957 M. E. Fisher proposed an iteration technique for high-speed analog computers equipped with a supplementary memory capacity which resulted in considerably faster convergence than the classical technique.[3] However, little practical experience with his method is available due to the complexity of the function storage and playback apparatus.[4] One test of Fisher's method was made using a repetitive analog computer in which the computer operator manually adjusted a set of potentiometers in a special function generator at the end of each iteration cycle.[5]

The purpose of this paper is to examine hybrid computer solution of integral equations, by both the Neumann and Fisher methods.

* This work was initiated during 1964-65, while the author was an NSF Senior Foreign Scholar and Visiting Professor at the University of Southern California, Los Angeles. It was supported in part by the Office of Scientific Research U. S. Air Force under Grant No. AF-AFOSR 1018-67.

The Neumann iteration method

The classical or Neumann iteration procedure for the solution of (1) is specified by

$$y_{n+1}(x) = f(x) + \lambda \int_a^b K(x,t)\, y_n(t)\, dt \qquad (2)$$

with $y_o(t) = 0$. Under conditions discussed in Reference 4, the process converges to a limit $y_\infty(x)$ which is the solution of (1).

From a computer point of view, an integration over the whole range of t must be made for each particular selected value of x. If the range of x is divided arbitrarily into I segments of length Δx, the function is represented by the values of y(x)) at the midpoint of each segment, i.e., $y(x_i)$, $i = 1,2,3,\ldots I$. It is clear that a total of I integrations in the t domain must be made before a single change in $y_n(t)$ is made in equation (2). Such an integration in t for a single value of $x = x_i$ will be called a *minor cycle*. In order to increase the index n, i.e., to derive the next approximating function $y_{n+1}(t)$, one has to complete I minor cycles. This group of minor cycles will be called a *major cycle*. The complete solution theoretically requires an infinite number of major cycles. However, practical experience[4] has demonstrated that accuracies of the order of 1% are attainable in about 20 major cycles using Neumann's method.

It should be noted that digital computer implementation of the strategy defined by (2) requires that the variable t also be discretized and that an appropriate numerical integration formula be used. For example, if Euler or rectangular integration is used, Equation (2) becomes

$$y_{n+1}(x_i) = f(x_i) + \lambda \sum_{j=1}^{J} K(x_i,t_j)\, y_n(t_j)\, \Delta t \qquad (3)$$

$$i = 1,2,\ldots,I$$

where Δt = constant is the integration step size and J is the total number of steps in the interval (b - a). Since t is only a dummy variable, the total range in t must equal the range of x and it is possible to choose the number of steps in t and x to be equal, i.e., let I = J. More sophisticated numerical procedures do not change the need for minor and major integration cycles. Equation (3) requires only the algebraic operations of addition and multiplication and is well suited to digital computation.

For hybrid computer solution each minor cycle may be performed continuously (using analog integration) and the resulting values stored. Assignment of other specific computational functions to the analog or digital portions of a system may significantly affect overall accuracy, as discussed in a later section of the paper. Thus, the generation of the functions f(x) and K(x,t) as well as the multiplication under the integral sign in (2) may be performed either in the analog or digital computer. A flow chart illustrating the programming of Neumann's method is shown in Figure 1. A stopping criterion given in the flow chart is based on reducing the difference between successive approximations to a sufficiently small value.

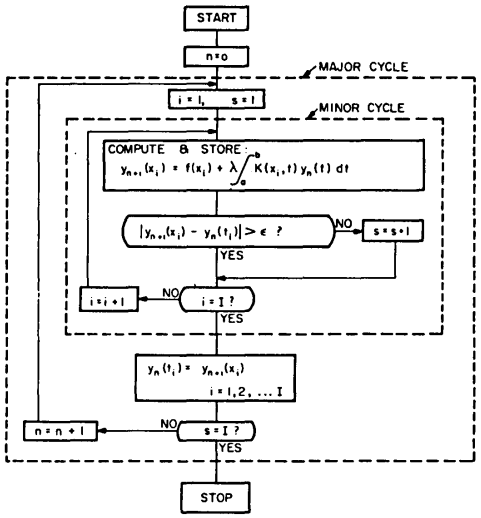


Figure 1—Flow diagram of Neumann's method

The Fisher iteration method

An examination of Equation (2) and Figure 1 reveals that Neumann's method requires the storage of $y_{n+1}(x_i)$ at the end of each minor cycle (i.e. for $i = 1,2,\ldots I$). At the end of I minor cycles the entire vector $[y_n(t)]$ under the integral sign is replaced and a major cycle has been completed.

Fisher's method[3,4] for the solution of the same problem requires that one element of the vector $[y_n(t)]$ be updated at the end of each minor cycle. Consequently, the Fisher version of the digital process of equation (3) becomes

$$y_{n+1}(x_i) = f(x_i) + \lambda \sum_{j=1}^{J} K(x_i,t_j)\, y_{n,i-1}(t_j)\, \Delta t \qquad (4)$$

$$i = 1,2,\ldots I,$$
$$j = 1,2,\ldots,J, \qquad I = J$$

Note that y(t) under the summation sign now carries a double subscript. The idea is to replace at the end of each minor cycle the existing value of $y_n(x_i)$ in the memory with the newly obtained value of $y_{n+1}(x_i)$. The notation (n,i)i=1,2,. . , I implies that during each major cycle the unknown function $y(x_i)$ is gradually adjusted as the index i is increased, not waiting, as in

the Neumann method, until all minor cycles are completed. In other words, Fisher's method is based on using each piece of new information as soon as it is available so that the adjustment from $y_n(x)$ to $y_{n+1}(x)$ proceeds gradually, rather than being performed all at once after I minor cycles.

The hybrid computer version of Fisher's method takes the form

$$y_{n+1}(x_i) = f(x_i) + \lambda \int_a^{x_i} K(x_i,t)\, y_{n+1}(t)\,dt + \lambda \int_{x_i}^{b} K(x_i,t)\, y_n(t)\,dt \qquad i=1,2,\ldots I \tag{5}$$

The first integral on right hand side of (5) contains the results which have been obtained during the previous minor cycles of the present major cycle. A flow chart showing the hybrid computer implementation of this strategy is shown in Figure 2.

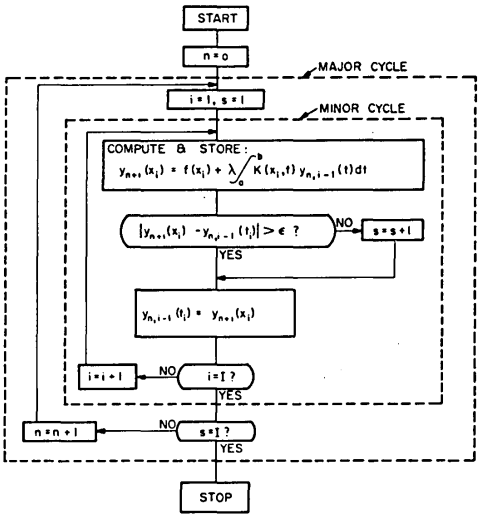


Figure 2—Flow diagram of Fisher's method

Illustrative examples

Fisher has shown[4] that for symmetric kernels his algorithm converges whenever the Neumann algorithm does, and in certain cases also when the classical method fails. (A symmetric kernel is characterized by $K(x,t) = K(t,x)$). Furthermore, using a simple example, Fisher has demonstrated that his method may speed up convergence significantly. In order to obtain practical results concerning this comparison, several problems were solved using a small hybrid computer (IBM 1620 digital computer and Beckman 2132 analog computer).

In order to facilitate the evaluation of the two methods, solutions, $y_n(t)$ were compared with known exact analytical solutions, $z(t)$, by means of a root-sum square criterion, defined by

$$F_n = \sqrt{\sum_{i=1}^{I} [\, z(x_i) - y_n(x_i)\,]^2} \tag{6}$$

Example 1. The following equation was solved:

$$y(x) = \frac{2}{3} + 2\int_0^1 (x-t)\, y(t)\,dt \tag{7}$$

with the initial approximation $y_o(t) = \frac{2}{3}$. The step size was chosen as $\Delta x = 0.1$ and the multiplication was performed on the analog computer. The solutions obtained by the Neumann and Fisher methods are compared in Figure 3 using the criterion F_n defined in (6). The considerably faster convergence of Fisher's method is clearly illustrated.

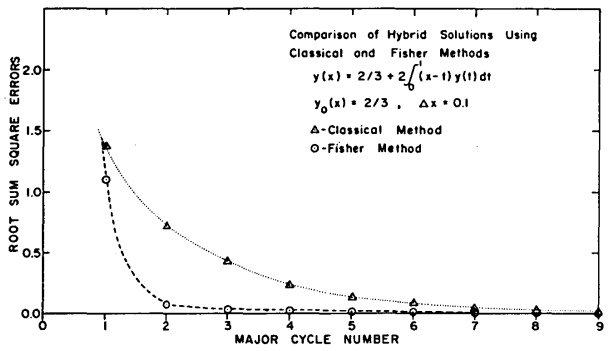


Figure 3—Comparison of hybrid solutions using classical and Fisher methods

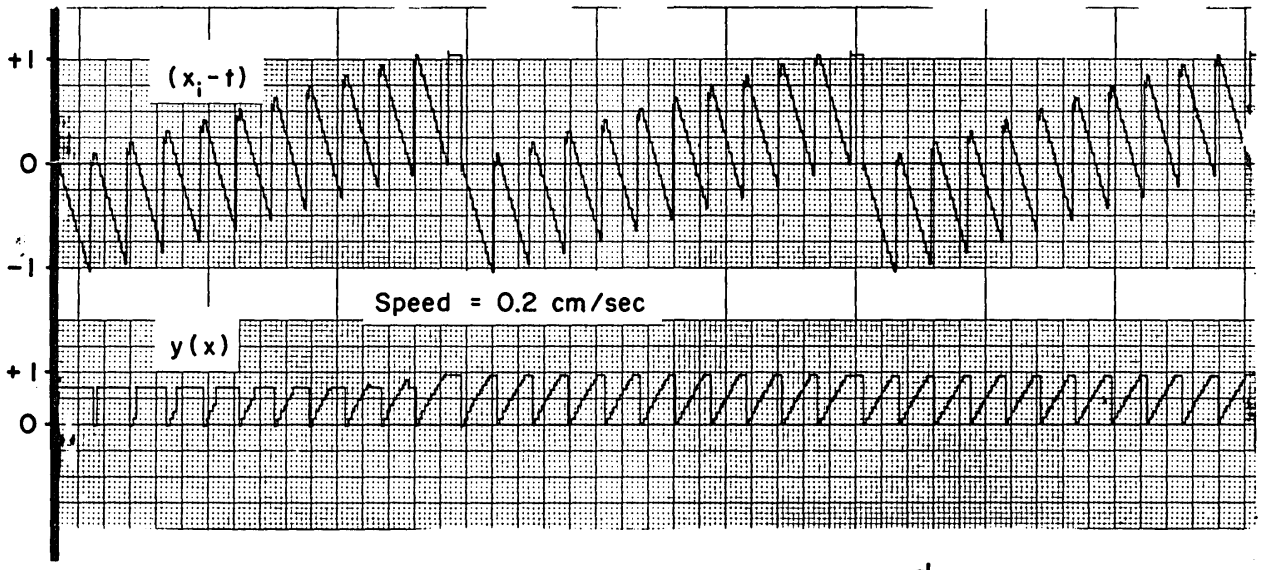
Time histories of the kernel $K(x,t) = x-t$ and the functions $y_n(x_i)$ for several major cycles are shown in Figure 4.

Example 2. The following equation was solved.

$$y(x) = 1.5x - \frac{7}{6} + \int_0^1 (x-t)y(t)\,dt \tag{8}$$

To illustrate the effect of choice of initial conditions, the equation was solved once with $y_o(t) = 0$ and once with $y_o(t) = 2/3$. The results are shown in Figure 5 for $\Delta x = 0.01$. It is evident that a poor choice of initial approximation may lengthen the convergence process.

Example 3. Examples 1 and 2 used kernels which



Time History of Fishers Solution to $y(x) = 2/3 + 2\int_0^1 (x-t)y(t)\,dt$

$y_0(x) = 2/3$

Figure 4—Time history of Fisher's solution

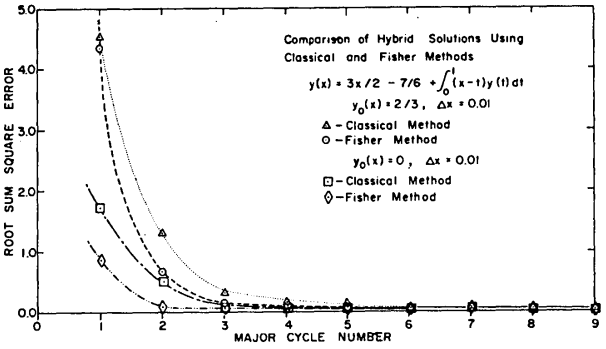


Figure 5—Comparison of hybrid solutions using classical and Fisher methods

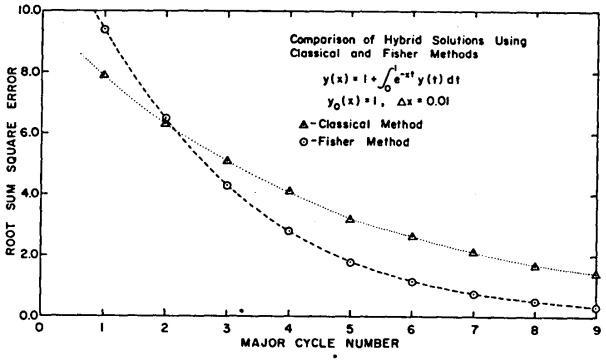


Figure 6—Comparison of hybrid solutions using classical and Fisher methods

could be considered functions of a single variable. Consider now the equation

$$y(x) = 1 + \int_0^1 e^{-xt}\, y(t)\, dt \qquad (9)$$

$$y_0(t) = 0$$

The results are shown in Figure 6. Once again the superiority of Fisher's method is evident.

Discussion of errors

The major errors which enter into the hybrid solutions of integral equations are the following: (a) truncation errors (due to the fact that the functions have been quantized), (b) A/D and D/A conversion errors, (c) other analog computer errors, and (d) phase shifts due to digital execution time.

Truncation errors arise from the quantization of the variables x and t in equation (1). In order to test the

importance of the quantization interval, Example 1 above was solved using 10 intervals ($\Delta x=0.1$) and 100 intervals ($\Delta x=0.01$). A comparison of the root sum squared errors for both interval sizes using the Fisher method is shown in Figure 7. It can be seen that convergence is speeded up by the choice of a smaller interval. After a sufficiently large number of major cycles, there is no apparent advantage to the small interval size. In fact, an oscillation in the final error criterion values ensues with the small Δx, which may be due to a random compounding of roundoff errors from the large number of minor cycles.

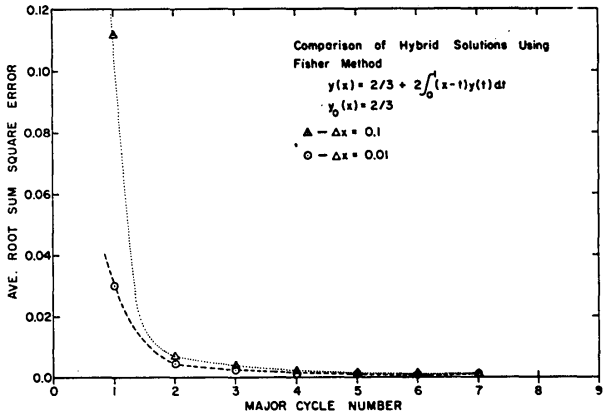


Figure 7—Comparison of hybrid solutions using Fisher method

The effect of quantization also enters into the integration process since the approximating function $y_n(t)$ is reconstructed from the samples $y_n(x_i)$, $i = 1,2, \ldots I$. In the present study this reconstruction was accomplished using simple zero-order holds, i.e.,

$$y_n(t) = y_n(t_i) \text{ for } t_i - \frac{\Delta t}{2} \leq t < t_i + \frac{\Delta t}{2} \qquad (10)$$

It can be shown[4] that with both zero-order and first-order reconstruction the errors are proportional to $(\Delta t)^2$. Second-order interpolation formulas will reduce the error to $0(\Delta t)^3$, but the additional computation required to achieve it may not be justifiable.

Analog-to-digital and digital-to-analog conversion errors were extremely important in the solution of the example problems. However, it should be noted that both the Fisher and Neumann techniques are stable processes as long as the solution to the problem is analytically convergent (see Reference 4). Thus, random errors (which may enter the problem from the converter or multiplier inaccuracies) during any one iteration (major cycle) will be corrected in subsequent iterations. Consequently, random errors delay the convergence process and may also cause a final indeterminacy region in the solution, as evidenced in the oscillation of Figure 7.

Alternate mechanization of kernel generation and multiplication were also investigated. The resulting effects on solution accuracy are clearly a function of the quality and precision of available analog multipliers and function generators, as well as conversion errors. It should be noted however, that analog generation of the kernel K(x,t) has the advantage of producing a program which has a solution time independent of kernel complexity and whose digital portion is universally applicable.

Solution time. While the actual times taken for a given solution clearly depend on the particular digital computer being utilized, comparisons between hybrid and all digital solutions are of interest. A speed up factor of 300 to 1 was obtained by using hybrid computation over digital computation, with comparable final accuracy.

Extension to other types of integral equations

The above discussion has been devoted entirely to equations of the Fredholm type. However, extension of the technique to many other types of equations is possible. Consider, for example, the Volterra equation:

$$y(x) = f(x) + \lambda \int_a^x K(x,t)\, y(t)\, dt \qquad (11)$$

This equation differs from the Fredholm equation in that the upper limit of integration is variable. The algorithms of Figure 1 and 2 are still applicable if the kernel is redefined such that

$$K(x, t) = O \text{ for } t > x \qquad (12)$$

A simple digitally controlled switch (needed to implement (12)) can be used in conjunction with the Fredholm equation program to solve Volterra type equations.

Wallman[1] and Fisher[3] have also indicated possible extension of hybrid techniques to the solution of multidimensional integral equations, integrodifferential equations and certain more general functional equations. Such extensions have yet to be proved in practice.

CONCLUSION

Hybrid computation techniques, involving fast repetitive analog integration and function generation and digital storage and control, are well suited to the solution of integral equations. Hybrid techniques lead to a substantial reduction in solution time when compared to all-digital methods. Further, the examples solved in this study substantiate the faster convergence of Fisher's iteration scheme, when contrasted with the classical