# An Extrapolation Step-Size Monitor for Solving Ordinary Differential Equations

N. L. Schryer
Bell Laboratories
Murray Hill, New Jersey 07974

Differential Equations, Integration, Step-Control, Order-Control

## ABSTRACT

A step-size monitor is presented for use in numerically solving ordinary differential equations by extrapolation methods. The monitor uses the information present in the extrapolation lozenge to determine the "optimal" step-size and order. This allows the monitor to adjust both the order and step-size to the local behavior of the solution in a reasonably "optimal" fashion. The monitor is particularly useful when obtaining low-precision solutions which require radical step-size changes. The results of this monitor are compared quite favorably with previous proposals.

## 1. Introduction

Bulirsch and Stoer [2] and Gragg [6] have considered the numerical solution of non-stiff ordinary differential equations by extrapolation methods. These techniques have been found (Fox [4], Hull [7] and Yu [11]) to be reliable, robust and competitive with other (Gear [5]) methods.

The problem studied in [2] and [6] is the numerical solution of the ordinary differential equation initial value problem

$$\frac{dx}{dt} = f(t,x) \quad a \leq t$$
$$x(a) = x_a \tag{1.1}$$

where $f(t,x)$ is some smooth vector valued function of $x$ and $t$. A brief outline of the ideas developed in those papers follows.

There are many basic differencing schemes for solving (1.1), such as Gragg's modified mid-point rule [6] and Backwards-Euler methods. Most of these methods have the property [10] that if they take $N$ time-steps to go from $t_0$ to $t_1$ and result in an approximation to $x(t_1)$ which we shall denote by $T(h)$ where $h = (t_1-t_0)/N$, then

$$T(h) = T + \sum_{j=1}^{\infty} T_j h^{j\gamma} \tag{1.2}$$

where $T = x(t_1)$, $\gamma$ is a positive constant depending on the basic difference scheme used, and the $T_j$ are unknown constants independent of $h$. For Gragg's modified mid-point rule $\gamma = 2$ for Backwards-Euler methods $\gamma = 1$.

Let a sequence of h's be defined by

$$h_i = h_0/N_i \quad i = 1,2,3,... \tag{1.3}$$

where $h_0 = t_1-t_0$ and the $N_i$ form a monotone increasing sequence of positive integers. Bulirsch and Stoer showed in [1] that given an operator $T(h)$ satisfying (1.2), and such

sequence $h_i$, then the value at h = 0 of the polynomial of degree m which interpolates $T(h_i)$ for i = 0,...,m, is given by $T_m^0$ which is determined from the recursion

$$T_0^i = T(h_i) \text{ for } 0 \le i \le m$$

(1.4)

$$T_j^i = T_{j-1}^{i+1} + (T_{j-1}^{i+1} - T_{j-1}^i)/\{(h_i/h_{i+j})^\gamma - 1\}$$

$$\text{for } 0 \le i \le m-j,\ 1 \le j \le m.$$

If the $T_j^i$ are organized into a lozenge of the form

$$\begin{array}{lccccc} T(h_0) = T_0^0 & & & & & \\ & T_1^0 & & & & \\ T(h_1) = T_0^1 & & T_2^0 & & & \\ & T_1^1 & & T_3^0 & & \\ T(h_2) = T_0^2 & & T_2^1 & & T_4^0 & \\ & T_1^2 & & T_3^1 & & T_5^0 \\ T(h_3) = T_0^3 & & T_2^2 & & T_4^1 & \\ & T_1^3 & & T_3^2 & & \\ T(h_4) = T_0^4 & & T_2^3 & & & \\ & T_1^4 & & & & \\ T(h_5) = T_0^5 & & & & & \end{array}$$

then (1.4) expresses each element of the j-th column (j>0) in terms of its two neighbors in column j-1. They also showed that the error obeys

$$(1.5) \quad |T_j^i - T| \le M_{j+1} (h_i \cdots h_{i+j})^\gamma$$

for some constants $M_{j+1}$. A similar result is established for interpolation by rational functions [1]. In (1.4) m is called the level of extrapolation, while from (1.5) we see that the order in column j is $(j+1)\gamma$. The value $h_0 = t_1 - t_0$ is referred to as the time-step while the $h_i$ are called sub-steps. Extrapolation approximates the $x(t_1)$ values accurately, but does not accurately approximate $x(t_0+nh_i)$ for $0<n<N_i$. Thus, by extrapolating the results of a basic ordinary differential equation solver, a process of arbitrarily high order can be obtained.

However, the original proposal in [2] had a number of built-in inefficiencies. First, the method attempted to use a fourteenth order process, no matter what error tolerance was prescribed. This resulted in gross inefficiency when very little, say only 1%, accuracy was desired. A second problem [4] was that if the initial step-size chosen was far too small or large, then the number of function calls needed to solve a problem often rose dramatically.

Thus, a step-size monitor was needed which

(a) is stable against an initially too large or small choice for the time-step

and

(b) uses the smallest time step (lowest order) consistent with "minimal" cost.

Requirement (b), at first glance, appears to be inconsistent in that small time-steps usually mean excessive cost. However, as we shall see later, it is not inconsistent at all. It merely tries to balance the need to accurately approximate the solution at as many $t_1$ points as possible, in order to get a decent picture or plot of it, and the need to conserve dollars.

In [9] Stoer presented a step-size monitor which meets both objectives (a) and (b) above admirably well for a very wide range of problems. By making a reasonable assumption about the growth of the derivatives of the solution being obtained, he developed a formula for obtaining the "optimal" level (order) of extrapolation to be used, as a function of the relative accuracy desired. Thus, for a given relative accuracy tolerance, a fixed-order technique was used. The resulting fixed-order, variable step-size algorithm was a

great improvement over the original proposal in [2]. However, there are situations in which the growth assumptions of [9] are not satisfied and for which that monitor gives less than satisfactory results. The reason for this is that when the solution being obtained has both "smooth" and "steep" regions, then the optimal order in these regions is typically "high" and "low", respectively. Thus, when entering a "steep" region, a fixed order method can not decrease its step-size as rapidly as it ought to. This wastes function evaluations when a "steep" region is approached.

In [9] Stoer uses two different error estimation procedures. The first is the very reliable scheme used in [2] and the second is unreliable. The first error estimation scheme of [9] estimates, roughly, the error in $T_j^i$ to be $|T_j^{i+1}-T_j^i|$. The second one estimates the error in $T_j^{i+1}$ to be $|T_j^{i+1}-T_j^i| * f_{ij}$ where $f_{ij} > 0$ can be very small numbers. So even though $|T_j^{i+1}-T_j^i|$ may be large, the second scheme may erroneously think the error in $T_j^{i+1}$ is quite small. The reliable error estimation procedure of [2] will be used in this paper. All comparisons made to the results of [9] will be made to the results using that same error estimation procedure. Work is continuing on making the second error estimator of [9] more reliable while retaining its ability to detect convergence earlier than the procedure used here. This may reduce the cost of extrapolation methods to 60% of their current cost.

This paper describes a simple step-size and order monitoring scheme which usually performs slightly better than the fixed-order step-size monitor given in [9], and in some cases much better. Section 2 describes the proposed monitor. Rather than make the growth assumption of [9], the proposal is to let the current extrapolation lozenge tell us what size lozenge is actually optimal. Section 3 considers two examples. The first example arises in the numerical solution of a coupled system of two parabolic partial differential equations and is exceedingly "stiff". The process extrapolated there is a Backwards-Euler time differencing scheme. The second is a non-stiff moon-earth-spaceship problem taken from [2]. In this example, the process extrapolated is Gragg's modified mid-point rule. For the first example, the present monitor uses one third the time and function evaluations as that used by the monitor of [9], when obtaining the solution to 1%. For the second example, the results of the two monitors are essentially identical when solving to a relative error of $10^{-11}$. However, when solving to $10^{-3}$ the present monitor uses 639 function calls compared with 1100 for [9].

## 2. The Proposed Step-Size Monitor

In [9] Stoer attempts to minimize the "cost per unit time step" used by the extrapolation procedures. The work, or number of function evaluations, needed to compute $T(h_i)$ is proportional to $N_i$. So the work needed to compute the first column of a lozenge based on $h_0,\ldots,h_k$ is proportional to

$$W_k = N_0 + \ldots + N_k. \tag{2.1}$$

Let $h_0^{(k)}$ be the largest value of $h_0$ such that convergence will be attained in the lozenge based on $h_0,\ldots,h_k$. Then the cost per unit time step for the k level lozenge is

$$C_k = W_k/h_0^{(k)}. \tag{2.2}$$

In this section we show how to compute the optimal step-sizes $h_0^{(k)}$ given the elements of the current lozenge. We shall then "optimize" the cost per unit time-step, using the information provided by the lozenge. This allows the order and time-step to change as the ongoing numerical solution process dictates it should.

It should be noted that both the original proposal [2] and the updated one in [9] assume that the most accurate value in the lozenge is at its tip. Thus, they only check the tip for convergence, ignoring all lower order columns. In many applications, especially those where the step-size varies radically, the best value in the lozenge is often not anywhere near the tip. So we test all columns of the lozenge for convergence, not just the last. This certainly does not slow the convergence process down, and in many cases results in a substantial improvement over just testing the last column.

Before proceeding, we need some error estimates and information on the rates of convergence of columns in the lozenge. Assume that we have a lozenge $T_j^i$, $i=0,\ldots,M-j$, $j=0,\ldots,M$, with $M>0$. Let $T$ be the true solution, Then [3] shows that for sufficiently small $h_0$,

$$(2.3)\quad |T_j^i - T| \cong \left[1+\frac{1}{\left(\frac{h_i}{h_{i+j+1}}\right)^{\gamma}-1}\right] |T_j^{i+1}-T_j^i|$$

and we can estimate the error $\epsilon_j^i = |T_j^i - T|$ in $T_j^i$. We also know from [3] and [9] that for sufficiently small $h_0$,

$$(2.4)\quad \epsilon_j^i = h_0^{\beta} D_j (h_i \ldots h_{i+j})^{\gamma}$$

where the $D_j$ are constants, $\gamma$ is the order of the basic process being extrapolated and $\beta$ is a constant. When extrapolating Gragg's modified mid-point rule, $\gamma=2$ and $\beta=1$. When extrapolating a Backwards-Euler time differencing process, $\beta=1=\gamma$. Thus, we can both estimate the accuracy of each element in the lozenge and tell how rapidly each column in the lozenge is converging.

Given a lozenge $T_j^i$ we can now determine the optimal sub-lozenge and step-size for the next time-step. This argument will assume that the constants $D_j$ in (2.4) do not change from the current time step to the next. If we want convergence to within an absolute error tolerance $\epsilon$ in column $j$, with $k$ levels of extrapolation, then by (2.4) we must have an initial step-size, call it $h_0^{(k,j)}$, so that

$$\epsilon = \left(h_0^{(k,j)}\right)^{\beta} D_j \left(h_{k-j}^{(k,j)} \ldots h_k^{(k,j)}\right)^{\gamma}$$

But by (2.3) and (2.4) we have from the current lozenge that

$$\epsilon_j^{M-1-j} = h_0^{\beta} D_j \left(h_{M-1-j} \cdots h_{M-1}\right)^{\gamma}$$

Inserting (1.3) into these equations we obtain

$$(2.5)\quad h_0^{(k,j)} = h_0 \left[\frac{\epsilon}{\epsilon_j^{M-1-j}}\right]^{\frac{1}{\beta+(j+1)\gamma}} \cdot \left[\frac{N_{k-j}\cdots N_k}{N_{M-1-j}\cdots N_{M-1}}\right]^{\frac{1}{\beta+(j+1)\gamma}}$$

and from this we can look for the largest $h_0^{(k,j)}$. This gives

$$(2.6)\quad h_0^{(k)} = \max_{0\le j\le k} h_0^{(k,j)},\quad k=0,\ldots,M-1.$$

$h_0^{(k)}$ is the largest $h_0$ that will guarantee convergence in the lozenge based on $h_0,\ldots,h_k$. Since the $W_k$ are given by (2.1), (2.6) enables us to compute the actual cost per unit time step, $C_k$, $k=0,\ldots,M-1$, if we are given the lozenge $T_j^i$.

There is a problem in deciding which size sub-lozenge is "optimal". Most step sequences $h_i$ have $N_i \cong b^i$ where $b>1$ is some constant. Typically $b=\sqrt{2}$ or 2. This means that $W_{k+1} \cong b*W_k$. Also, if a lozenge

based on $h_0,\ldots,h_k$ converges, then, roughly, so does one based on $b*h_0,\ldots,b\,h_{k+1}$. This follows since if we let $h_0 \to b*h$ and $i \to i+1$ in (2.4), then the error goes from $\epsilon$ to $b^{\beta}\epsilon \cong \epsilon$. Thus, $h_0^{(k+1)} \cong b*h_0^{(k)}$ and $C_{k+1} \cong C_k$. So, in general, since all sufficiently large k give about the same cost $C_k$, there is no k so that $C_k$ is minimal. This leads to the definition that a lozenge of size k is sub-optimal if the largest $h_0^{(k,j)}$ for that lozenge has j=k. That is, if no column below column k can be used to obtain $h_0^{(k)}$, then column k is needed to make the step-size for the lozenge as large as possible and thus minimize $C_k$. The largest sub-optimal lozenge can then be called optimal. That is, any larger lozenge would use the same columns as the optimal one in order to produce the same cost per unit time-step.

The above definition of optimality leads to the following algorithm, which assumes that a lozenge is given, for obtaining the optimal lozenge size (order) and time-step.

(a) Compute the $\epsilon_j^{M-1-j}$ using (2.3).
(b) Compute the costs $C_k$, k=0,...,M-1 using (2.5) and (2.6).
(c) Find the optimal sub-lozenge size, $k_{opt}$.

This gives the optimal time-step $h_0^{(k_{opt})}$ and lozenge size $M_{opt} = k_{opt} + 1$. This value of $M_{opt}$ is needed in order to use (2.3) in estimating the error in column $k_{opt}$.

Another problem arises here. So far, the size of the first lozenge computed will dominate all successive lozenge sizes because $M_{opt} = k_{opt} + 1 \leq M$. So some way must be found to let M increase, if it ought to. The only way to do that is to sample the error in column $k_{opt} + 1$. This requires $\tilde{M}_{opt} = k_{opt} + 2$ levels of extrapolation and, using (2.5), a step-size

$$(2.7)\quad \tilde{h}_0^{(k_{opt})} = \max_{0\leq j\leq k_{opt}} h_0^{(k_{opt}+1,j)}.$$

will give convergence in the lozenge based on $\tilde{h}_0^{(k_{opt})},\ldots,\tilde{h}_{\tilde{M}_{opt}}^{(k_{opt})}$. This choice of $\tilde{h}_0^{(k_{opt})}$ for the next $h_0$ should, if $M_{opt}=M$, require M+1 levels of extrapolation for the next time step. Specifically, this mechanism will increase M by 1 until k=M-1 is no longer optimal and then stop increasing M.

A final heuristic modification of the $\tilde{h}_0^{(k_{opt})}$ is adopted. In the same spirit as the assumption that the constants $D_j$ in (2.4) will not change radically from the current time step to the next, we assume that if $L=\min(k_{opt}^{new}, M_{opt}^{old}-1)$ and $C_L^{old} < C_L^{new}$ then the same thing will happen on the next time-step. Thus we choose

$$(2.8)\quad H_0^{(k_{opt})} = \tilde{h}_0^{(k_{opt})} \min(1, C_L^{old}/C_L^{new})$$

This tries to make $h_0 = H_0^{(k_{opt})}$ more nearly optimal for the next time step.

The above algorithms allow the lozenge from one time step to predict the optimal M and $h_0$, namely $\tilde{M}_{opt}$ and $H_0^{(k_{opt})}$, for the next time step. This assumes that the solutions's structure, the constants $D_j$ in (2.4), do not change too radically from this time step to the next. In practice of course, this is not always the case. Sometimes the predicted $h_0$ will turn out to be far too large for convergence to occur in a reasonably finite lozenge. Some way must be found to discover this fact before the lozenge gets very large, and very expensive, and we are forced to throw away all this dearly obtained information, lower the step-size $h_0$ and start all over again. Such restarts can add a significant overhead cost to a run. The following mechanism is proposed to handle this problem. A similar method was used in [9] to predict restarts. When in the next time step

$M = \tilde{M}_{opt}$, the predicted optimal level, and convergence has not yet occurred, then compute the $M_{opt}$, call it M*, and $\tilde{h}_0^{(k^*)}$, $k^* = M^*-2$, based on the current lozenge. Then, if the solution can be carried out more efficiently over a time interval of length $h_0$ by using a step-size $\tilde{h}_0^{(k^*)}$, rather than $h_0$, restart the process using $h_0 = \tilde{h}_0^{(k^*)}$. This can be stated as: If

$$(2.9) \qquad W_M + W_{M^*} \frac{h_0}{\tilde{h}_0^{(k^*)}} < W_{M'} \cdot$$

where $M' > M$ is the smallest integer which will guarantee convergence in the first M-1 columns of the lozenge based on $h_0,\ldots,h_{M'}$, then restart. The value of M' may be found using the relations

$$\epsilon_j^{M-1-j} = h_0^{\beta} D_j(h_{M-1-j}\cdots h_{M-1})^{\gamma}$$

and

$$\epsilon \geq h_0^{\beta} D_j(h_{M'-1-j}\cdots h_{M'-1})^{\gamma}$$

giving

$$(2.10) \qquad \frac{\epsilon}{\epsilon_j^{M-1-j}} \geq \left[\frac{N_{M-1-j}\cdots N_{M-1}}{N_{M'-1-j}\cdots N_{M'-1}}\right]^{\gamma}$$

M' is simply increased from M' = M+1 until (2.10) holds for some j.

To start things off, when there is no previous lozenge to predict the proper value of $h_0$, we simply force convergence to occur in columns 0 or 1 by setting $M_{opt}^{old}$ to 2. For $\gamma$=1, this uses a method of order no greater than 2 for the first time step. For $\gamma$=2, the order used is no greater than 4 for the first time step. This results in no great expense when $h_0$ is far too large to start with.

The entire step-size and order monitor is then

(a) Set $\tilde{M}_{opt}^{old} = 2$, take $h_0$ as given by the user.

(b) Set M = 0.

(c) Compute the level M extrapolation lozenge.

(d) If have convergence, compute $k_{opt}$ and $\tilde{M}_{opt}$, set $h_0 = H_0^{(k_{opt})}$, $\tilde{M}_{opt}^{old} = \tilde{M}_{opt}$ and go to (b).

(2.11)

(e) If $M < \tilde{M}_{opt}^{old}$ then go to (h).

(f) Compute M' from (2.10) and $M_{opt}$ based on the current lozenge.

(g) If (2.9) holds, set $h_0 = \tilde{h}_0^{(k_{opt})}$ and go to (b).

(h) Set M = M+1 and go to (c).

## 3. Examples

All examples were carried out on a Honeywell HIS 6070 computer in double precision (18 decimal digits). The first example considered is the numerical solution of a large (144 components) and very stiff system of ordinary differential equations which arises from the spatial discretization of a coupled system of two parabolic partial differential equations, in one space dimension. These equations model the motion of magnetic domain walls in uniaxial materials [8]. The basic algorithm used in [8] was a fully implicit Backwards-Euler differencing scheme in time. Thus $\beta$=1=$\gamma$. The sequence $\{N_0,N_1,\ldots\} = \{1,2,3,4,6,\ldots\}$ was used, without rounding error problems. The error tolerance used was a local 1% error in all variables.

When an initial time step of $h_0 = 10^{-4}$ was used, the second time step chosen by (2.11) was 0.13, which required 2 levels of extrapolation to converge. When an initial step size of $h_0 = 20$ was chosen, in 6 function evaluations (out of a total of 540 needed to solve the problem) the extrapolation was successfully restarted with $h_0 = 0.18$. This shows the ability of the monitor (2.11) to recover nicely from

too small or too large an initial step-size.

Figure 1 shows a plot of the velocity of the wall as a function of time. The behavior of the velocity as a function of time is typical of the general behavior of the solution of this problem. However, the velocity is not one of the variables obtained directly by extrapolation. Rather, it is derived from the results produced by the extrapolation process. As can be seen, the monitor samples the solution nicely. This is a particularly nasty problem in that the optimal $h_0$ varies by a factor of more than 100 from one place to another and $k_{opt}$ varies from 0 through 3. The lower order and step-sizes occurred near the bottom of the spiky regions of the velocity and the larger ones occurred near the middle of the smoother regions.

The results of [9] were generalized to cover the case of $\gamma$=1 and were coded and tested on exactly the same problem solved above. The result was that it used more than 3 times as many (1782) function evaluations as the present one did. When the program was altered to check each column for convergence, rather than just the last, this ratio dropped to roughly 2 (1276).

The present monitor only failed (restarted) twice during the run. It automatically lowered both the order and the step-size as it stepped into the spiky regions. But the monitor of [9] could not, being of fixed order, thus always choosing too large a step-size as it entered the spikes. This reflects the fact that (2.11) actually looks at the entries in the lozenge and adapts to whatever it finds is going on there, whereas [9] makes some assumptions about what is going on there.

The second example is the non-stiff equation for the restricted three body problem given in [2]. The system of equations is

$$\ddot{x} = x+2\dot{y}-\mu\frac{x+\mu}{[(x+\mu)^2+y^2]^{3/2}} -\mu\frac{x-\mu'}{[(x-\mu')^2+y^2]^{3/2}}$$

$$\ddot{y} = y-2\dot{x}-\mu'\frac{y}{[(x+\mu)^2+y^2]^{3/2}}-\mu\frac{y}{[(x-\mu')^2+y^2]^{3/2}}$$

where $\mu = 0.012128562765312$ and the initial conditions are $x(0) = 1.2$, $dx(0)/dt = 0$, $y(0) = 0$ and $dy(0)/dt = -1.04935750983$. The motion is periodic with period $T = 6.192169331396$ and the solution is obtained over one period. When the $\epsilon = 10^{-11}$ relative error convergence criterion of that paper was used, the results of both (2.11) and [9] were basically identical (4144 vs. 4320 function evaluations). Here extrapolation of Gragg's modified mid-point rule was used, so $\gamma$=2 and $\beta$=1, and the sequence $\{N_0,N_1,\ldots\} = \{1,2,3,4,6,\ldots\}$ was used. When the same problem was solved with $\epsilon = 10^{-3}$ the result was that (2.11) required 639 function calls and [9] required 1100.

These examples, taken from widely diverse backgrounds and solved to substantially different error tolerances, are typical of the observed relative behavior of (2.11) and [9]. For smooth problems they behave in almost exactly the same manner (the assumption of [9] holds quite well). For non-smooth problems solved to low precision, (2.11) may be substantially faster. For non-smooth problems solved to high precision, the two monitors behave very similarly.

The last two observations are easily understood. When a solution is desired to low precision, but for which the optimal order varies over the "low" order range (say from 1 to 4), then varying the order results in a substantial change in the procedure. However, when the same solution is obtained to "high" precision, the optimal order may vary from 10 to 14, and the relative savings are not as pronounced.

In all cases run, the execution-time overhead of (2.11) has never exceeded 6%,

and for the very large problem of [8] (200 variables) was only 0.4%. The low overhead of the present monitor allows its increased efficiency in terms of function evaluations to also make it more efficient in terms of execution time.

References


[1] R. Bulirsch and J. Stoer, "Fehlerabschatzungen und Extrapolation mit rationalen Funktionen bei Verfahren vom Richardson-Typus", Num. Math. 6, 413-427 (1964).

[2] R. Bulirsch and J. Stoer, "Numerical Treatment of Ordinary Differential Equations by Extrapolation Methods", Num. Math. 8, 1-13 (1966).

[3] R. Bulirsch and J. Stoer, "Asymptotic Upper and Lower Bounds for Results of Extrapolation Methods", Num. Math. 8, 93-104 (1966).

[4] P. A. Fox, "A Comparative Study of Computer Programs for Integrating Differential Equations", Comm. ACM 15, 941-948 (1972).

[5] C. W. Gear, "The Automatic Integration of Ordinary Differential Equations", Comm. ACM 14, 176-179 (1971).

[6] W. B. Gragg, "Repeated Extrapolation to the Limit in the Numerical Solution of Ordinary Differential Equations", Thesis, UCLA (1963).

[7] T. E. Hull, W. H. Enright, B. M. Fellen and A. E. Sedgwick, "Comparing Numerical Methods for Ordinary Differential Equations", Technical Report 29, 1971, Department of Computer Sciences, University of Toronto.

[8] N. L. Schryer and L. R. Walker, "On the Motion of 180° Domain Walls", submitted to the J. of Appl. Physics.

[9] J. Stoer, "Extrapolation Methods for the Solution of Initial Value Problems and their Practical Realization", Conference on the Numerical Solution of Ordinary Differential Equations, University of Texas at Austin, 1972.

[10] H. J. Stetter, "Asymptotic Expansions for the Error of Discretization Algorithms for Non-Linear Functional Equations", Num. Math. 7, 18-31 (1965).

[11] T. Yu, "Comparison of Numerical Methods for Ordinary Differential Equations", Tech. Report CNA-73, 1973, University of Texas at Austin.

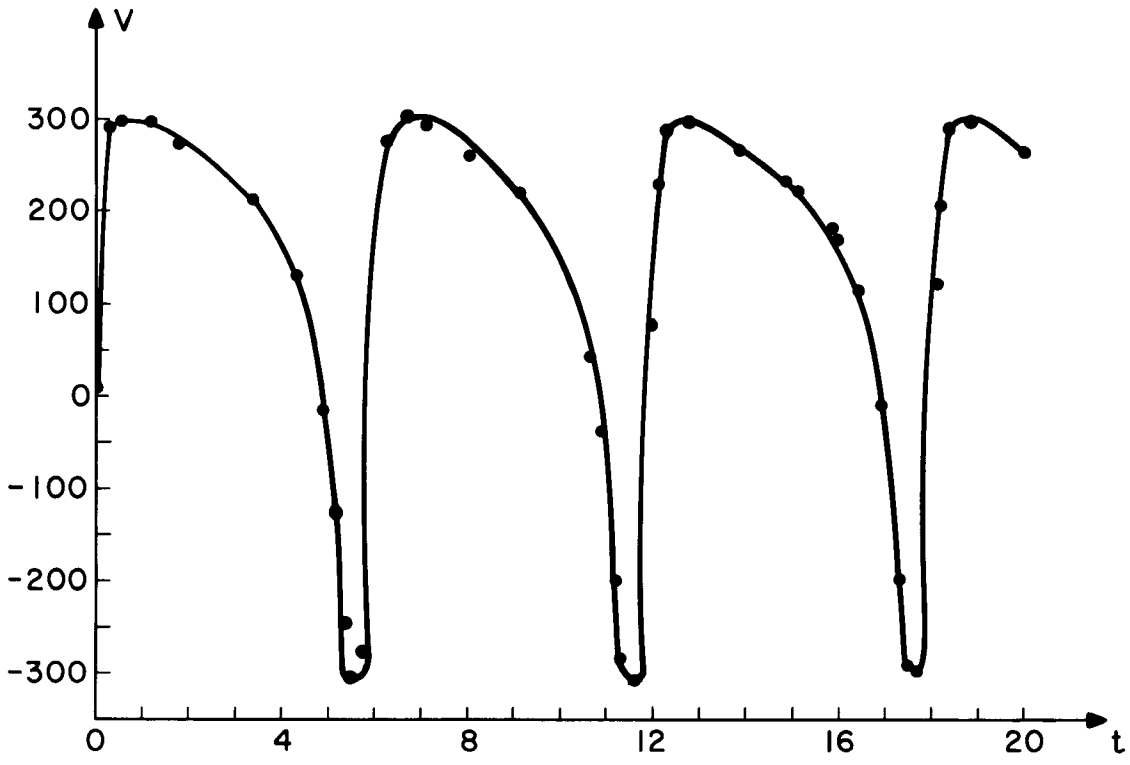


FIG. 1