Solving Ordinary Differential Equations with Discontinuities

Charles William Gear, Ole Østerby

Solving Ordinary Differential Equations with Discontinuities

C. W. GEAR
University of Illinois at Urbana-Champaign
and
O. ØSTERBY
Aarhus University, Denmark

Automatic codes for differential equations can be inadequate when the solutions have discontinuities. If the user provides an external indicator for discontinuities (e.g., a switching function whose sign changes indicate discontinuities), a code can be more efficient. A technique for detection and location of a discontinuity is discussed which can be implemented when such indicators are not practical. It estimates the order and magnitude of the discontinuity and hence the stepsize which keeps the error under control while stepping over the discontinuity.

Categories and Subject Descriptors: G.1.7 [**Numerical Analysis**]: Ordinary Differential Equations

General Terms: none

Additional Key Words and Phrases: Multistep, differential equations, singularities, discontinuities

1. INTRODUCTION

Many problems in simulation and control give rise to systems of ordinary differential equations (ODEs)

$$y' = f(x, y),$$

in which the right-hand-side function f contains discontinuities in the form of finite jumps either in components of f itself or in some derivatives of f. Numerical software for solving ODEs will often behave very inefficiently in the presence of such singularities and it is the aim of this paper to show how these shortcomings can be remedied.

We distinguish between four computational stages and treat each of them in detail in subsequent sections.

This work was supported in part by the U.S. Department of Energy under grant DOE DEAC02 76ERO2383, and in part by the Danish Science Research Council.
Authors' addresses: C.W. Gear, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801; O. Østerby, Computer Science Department, Aarhus University, Aarhus, Denmark.

Stage 1. *Detecting the possible presence of a discontinuity.* This should be a very inexpensive set of tests because it must be activated at every rejected step of the integration routine.

Stage 2. *Locating the discontinuity.* After a discontinuity is detected, we must locate it as quickly and efficiently as possible.

Stage 3. *Passing the discontinuity.* When we know where the discontinuity is we can step right up to it and step across it with due regard to the size of the local error involved in the crossing.

Stage 4. *Restarting.* Since the previous values of y and f no longer correspond to smooth functions, we must be careful about using such information. The restarting procedure must be chosen according to the nature of the discontinuity.

In many cases these discontinuities do not occur completely out of the blue. We may know the x-value where f changes its nature or we may be given a switching function which, when reaching a certain known value, triggers the discontinuity. It is very important to use such extra information whenever available as this will facilitate the tasks of Stage 1 and especially the otherwise very time-consuming tasks of Stage 2. Methods for locating discontinuities triggered by switching functions have been discussed in the literature and range from interpolation-type methods [2, 5, 10] to fractional step methods [7, 9] and methods which regard the discontinuity conditions as additional differential equations [1]. Here, however, we concentrate on the harder problem of dealing with discontinuities which appear without other warning, as, for instance, when the right-hand-side function is supplied by some black-box code which hides the switching from the user.

A typical variable-order, variable-step code will check a local error estimate at every step and decide whether to accept or reject the step and whether to try a different stepsize and/or order in the next step. The presence of a discontinuity is usually signaled by a very large value of the local error estimate (cf. eq. (2.14, p. 28), resulting in the rejection of the step and a drastic reduction of the stepsize and possibly also of the order for the next try. Back in the smooth region to the left of the discontinuity there will often be time enough to build up the stepsize (and order) until the code again attempts to step over the discontinuity. The stepsize is reduced again and this process may be repeated several times before the code successfully passes the trouble spot.

Figure 1 illustrates this for the Hindmarsh code [6] as used on the initial value problem

$$y' = \begin{cases} 0 & x < 40.33, \\ 100 & x \geq 40.33, \end{cases} \qquad y(0) = 40.33.$$

The ordinate is the number of function evaluations and is thus a measure of the work involved in passing the discontinuity. Failed steps are marked with an x and successful steps with an o.

From the time the code first attempts to overstep the discontinuity until it finally succeeds it uses 118 function evaluations and takes 97 steps, 18 of which are rejected. The local error tolerance is 10^{-5} relative to y, corresponding to

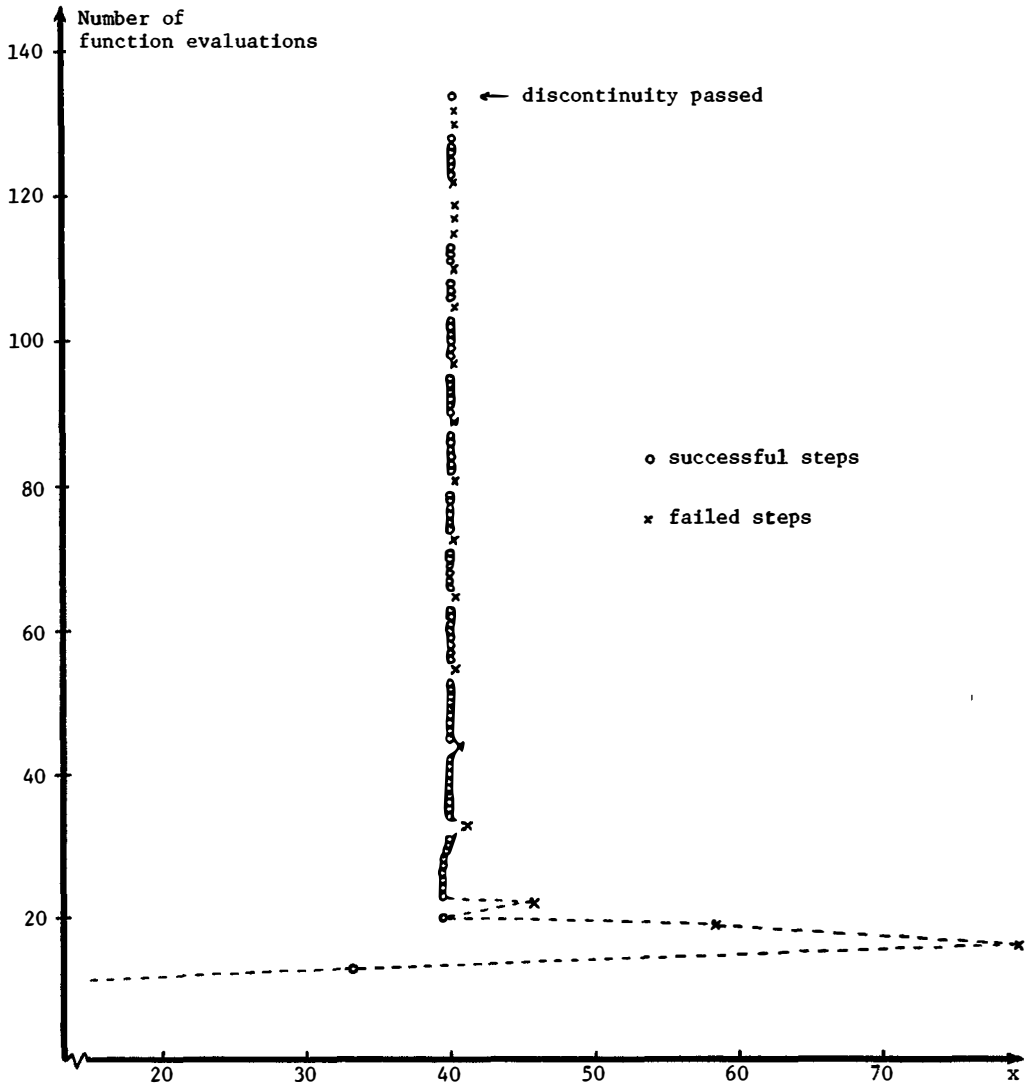

Fig. 1. The behavior of an ODE code at a discontinuity.

4×10^{-4} in absolute measure. If the location of this very simple discontinuity is known to the code, it is clear it need use no more than one extra step. If the discontinuity has to be located by bisection (which will be seen to be the best option in this case), 23 step halvings are adequate to get over the discontinuity in the worst case.

In Section 2 we introduce notation and perform a theoretical analysis of what happens at and around a discontinuity, paying particular attention to the behavior of local errors and error estimates. We develop the theory for a predictor–corrector method in PEC or PECE mode, but many of our results carry over or are easily adapted to other modes or methods.

Sections 3–6 deal with each of the four computational stages in the process of crossing a discontinuity. In Section 7 we look at what can go wrong when we misinterpret information and when things do not turn out quite the way we expected.

This paper is a revised and shortened version of [4] which also contained formulas for error estimates with variable stepsize and where we suggested methods for determining new stepsizes which are different from and theoretically more satisfying than the ones currently being used. Reference [4] also contains some of the messy algebra the results of which are used in Section 4 of this paper to determine the order of the discontinuity.

2. NOTATION AND THEORY

The numerical method we have in mind is a predictor–corrector method in PEC or PECE mode where both the predictor and corrector have order p. The notation used for the predictor is

$$\sum_{j=0}^{k} \alpha_j^* y_{n+j} = h \sum_{j=0}^{k-1} \beta_j^* f_{n+j}; \qquad f_{n+j} = f(x_{n+j}, y_{n+j}); \quad \alpha_k^* = 1, \tag{2.1}$$

and for the corrector

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j}; \qquad \alpha_k = 1. \tag{2.2}$$

The local truncation error of such a method can be written in the form

$$C_{p+1} h^{p+1} y^{(p+1)}(x_n) + O(h^{p+2}), \tag{2.3}$$

when f has continuous partial derivatives of order through $p + 1$ in a strip around the solution y. The error constant of the corrector is C_{p+1}, and the stepsize is h.

Together with the numerical method we also have a method for estimating the local error. Such a local error estimate is typically proportional to the difference between the predicted and the corrected y-value (e.g., Milne's device).

We shall first see how discontinuity can affect the local truncation error and the local error estimate. The former is important for the error control and the latter because many decisions in the program will be based on the estimate.

Assume that f has a discontinuity along a border line λ, but that it also has a smooth continuation, $\tilde{f}$ across λ with continuous partial derivatives of order $p + 1$, giving rise to a smooth solution $\tilde{y}$ (see Figure 2).

If the discontinuity appears between x_{n+k-1} and x_{n+k}, then the predicted value y_{n+k}^* will not be affected by it: y_{n+k}^* is actually an approximation to $\tilde{y}(x_{n+k})$. The corrector uses the function value

$$f_{n+k}^* = f(x_{n+k}, y_{n+k}^*) = \tilde{f}_{n+k}^* + (f_{n+k}^* - \tilde{f}_{n+k}^*), \tag{2.4}$$

and the corrected value is thus

$$\begin{aligned} y_{n+k} = {} & \sum_{j=0}^{k-1} (-\alpha_j y_{n+j} + h\beta_j f_{n+j}) \\ & + h\beta_k \tilde{f}_{n+k}^* + h\beta_k (f_{n+k}^* - \tilde{f}_{n+k}^*). \end{aligned} \tag{2.5}$$

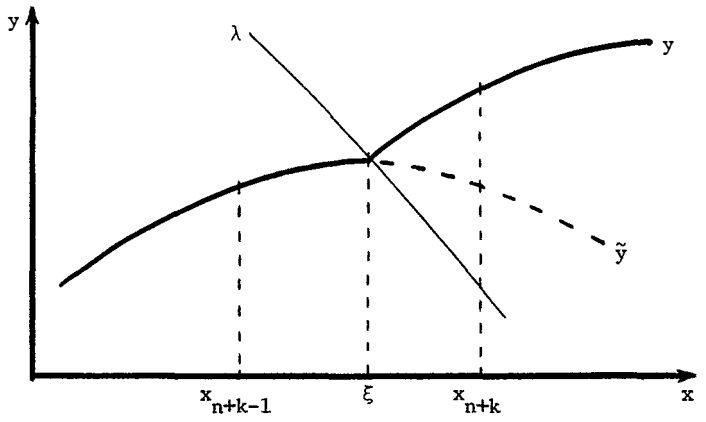

Fig. 2. The solution y, and the smooth solution $\tilde{y}$ corresponding to a smooth continuation $\tilde{f}$ of f.

The local error estimate is a multiple of the corrector–predictor difference:

$$\begin{aligned}\gamma \times (y_{n+k} - y^*_{n+k}) &= \gamma \times \left\{ \sum_{j=0}^{k-1} (\alpha_j^* - \alpha_j) y_{n+j} + h(\beta_j - \beta_j^*) f_{n+j} \right\} \\ &\quad + \gamma h \beta_k \tilde{f}^*_{n+k} + \gamma h \beta_k (f^*_{n+k} - \tilde{f}^*_{n+k}) \\ &= C_{p+1} h^{p+1} \tilde{y}^{(p+1)}(x_n) + O(h^{p+2}) \\ &\quad + \gamma h \beta_k (f^*_{n+k} - \tilde{f}^*_{n+k}).\end{aligned} \tag{2.6}$$

If Milne's device is used then $\gamma = C_{p+1}/(C^*_{p+1} - C_{p+1})$, where C^*_{p+1} is the error constant of the predictor, and in this case we should have $\alpha_j = \alpha_j^*$, $j = 0, 1, \ldots, k$, for the error estimate to be asymptotically correct.

In eq. (2.6) we have isolated the effect of the discontinuity on the local error estimate, and we now take a closer look at the types of discontinuities which we treat: jump discontinuities in f or in the partial derivatives of f.

The discontinuity is said to be *of order* q (≥ 1) if f has continuous partial derivatives through order $q - 2$ and there is a finite jump discontinuity in at least one of the partial derivatives of f of order $q - 1$. This gives rise to a jump discontinuity in $y^{(q)}$ at the point ξ and to a term of order $O(h^q)$ in the local error estimate.

The size of the jump in $y^{(q)}$ is denoted K_q:

$$y^{(q)}_+(\xi) = \tilde{y}^{(q)}(\xi) + K_q. \tag{2.7}$$

The + in $y^{(q)}_+$ indicates that the derivative is taken from the right.

If the location of the discontinuity depends on the value of y, then we may not know the exact value of ξ. For practical purposes we can use the value corresponding to the local solution through (x_{n+k-1}, y_{n+k-1}).

The jump in $y^{(q)}$ at ξ can be simply related to the jump in the $(q - 1)$th partial derivative of f. Considering the autonomous form $y' = f(y)$ for notational

simplicity, we have

$$y^{(q)} = \frac{\partial^{q-1} f}{\partial y^{q-1}} f^{q-1} + \text{combinations of lower derivatives.}$$

The only jump is in the $(q-1)$th partial, so we have

$$\frac{\partial^{q-1}}{\partial y^{q-1}} [f_+ - \tilde{f}] \tilde{f}^{q-1} = K_q. \tag{2.8}$$

We want to measure the effect of a discontinuity on the error estimate (eq. (2.6)). To do this we use Taylor's series to evalute $f^*_{n+k} - \tilde{f}^*_{n+k}$. For notational simplicity, the argument (ξ, η) of the discontinuity will be omitted from values of f, $\tilde{f}$, and their partial derivatives; and f and its derivatives will refer to values to the right of the discontinuity. We get

$$\tilde{f}^*_{n+k} = \tilde{f} + \tilde{f}_y(y^*_{n+k} - \eta) + 1/2\, \tilde{f}_{yy}(y^*_{n+k} - \eta)^2 + \cdots \tag{2.9}$$

$$f^*_{n+k} = f + f_y(y^*_{n+k} - \eta) + 1/2\, f_{yy}(y^*_{n+k} - \eta)^2 + \cdots \tag{2.10}$$

Define θ by

$$x_{n+k} - \xi = \theta h, \qquad 0 < \theta \leq 1, \tag{2.11}$$

and note that

$$y^*_{n+k} - \eta = \theta h \tilde{f} + O(h^2). \tag{2.12}$$

Hence we have from eqs. (2.8), (2.9), (2.10) and (2.12)

$$\begin{aligned} f^*_{n+k} - \tilde{f}^*_{n+k} &= \frac{1}{(q-1)!} \frac{\partial^{q-1}}{\partial y^{q-1}} [f - \tilde{f}](\theta h \tilde{f}^{q-1} + O(h^q) \\ &= \frac{(\theta h)^{q-1}}{(q-1)!} K_q + O(h^q), \end{aligned} \tag{2.13}$$

and the local error estimate becomes

$$\begin{aligned} \gamma(y_{n+k} - y^*_{n+k}) = \gamma\beta_k \times \frac{\theta^{q-1}}{(q-1)!} h^q K_q + O(h^{q+1}) \\ + C_{p+1} h^{p+1} \tilde{y}^{(p+1)}(x_n) + O(h^{p+2}). \end{aligned} \tag{2.14}$$

When $q \geq p+1$ the method will remain of order p and we may not even notice the discontinuity, whereas for $q \leq p$ the dominant term in the local error estimate is due to the discontinuity. A variable step code which determines the stepsize on the basis of the local error estimate will therefore often respond to a discontinuity by reducing the stepsize drastically.

To find out what the local error actually is, note that

$$y^{(q)}(x) = \tilde{y}^{(q)}(x) + K_q + O(h) \tag{2.15}$$

throughout the interval (ξ, x_{n+k}), and

$$y(x_{n+k}) = \tilde{y}(x_{n+k}) + \frac{(\theta h)^q}{q!} \times K_q + O(h^{q+1}). \tag{2.16}$$

An explicit method (predictor) will produce a good approximation to $\tilde{y}(x_{n+k})$:

$$y^*_{n+k} = \tilde{y}(x_{n+k}) + O(h^{p+1}), \tag{2.17}$$

whereas the implicit method (corrector) will take the new f into account. From eqs. (2.5) and (2.13) we deduce that

$$y_{n+k} = \tilde{y}(x_{n+k}) + \frac{\beta_k \theta^{q-1}}{(q-1)!} h^q K_q + O(h^{q+1} + h^{p+1}). \tag{2.18}$$

The local error is thus

$$\left(\frac{\theta}{q} - \beta_k\right) \times \frac{\theta^{q-1}}{(q-1)!} h^q K_q + O(h^{q+1} + h^{p+1}), \tag{2.19}$$

and, when $q \leq p$, an upper bound for the leading term of the local error is

$$\frac{h^q}{(q-1)!} K_q, \tag{2.20}$$

since $0 < \beta_k \leq 1$. When we know K_q this expression can be used to ensure that the local error will be less than a specified error tolerance ϵ when taking a step across the discontinuity. If we define the *passing stepsize*

$$h_{\text{pass}} = \left(\frac{(q-1)!}{K_q} \times \epsilon\right)^{1/q}, \tag{2.21}$$

then we know whenever $h < h_{\text{pass}}$ that the local error in passing the discontinuity will be less than ϵ.

In particular, for $q = 1$ we have (as in [6])

$$h_{\text{pass}} = \epsilon / K_1. \tag{2.22}$$

We note in passing that the error estimate is very realistic in this case. From eqs. (2.19) and (2.22) the local error is seen to be approximately $(\theta - \beta_k) \times \epsilon$, with an expected value as θ varies of at least $\epsilon/4$.

The formulas for the local error estimate (2.14) and for the local error (2.19) are very similar, which makes a comparison very easy. The only difference in the leading term is the factor $\gamma\beta_k$ in (2.14) versus $\theta/q - \beta_k$ in (2.19). If Milne's device is used, then γ is negative and ≤ 0.1 in magnitude for Adams' methods of orders greater than 2. The local error estimate will therefore often be as much as an order of magnitude smaller than the local error. This means that if we rely on the local error estimate (which is all we have) then we might commit an error which is an order of magnitude greater than the tolerance when we pass over the discontinuity. In order to avoid serious consequences from this, we might choose to either lower the error tolerance when a discontinuity is detected or use the corrector–predictor difference directly, as this provides a safer estimate of the local error when a discontinuity is present.

3. STAGE 1—DETECTING A DISCONTINUITY

This must be a very low cost set of tests that fit well into an existing code, for in principle it should be activated at every rejected step of the integration routine.

As already mentioned the presence of a discontinuity is usually indicated by a large value of the local error estimate. This in turn will prompt a reduction in the stepsize according to a formula like

$$h_{\text{new}} = \left| \frac{\epsilon}{\text{error estimate}} \right|^{1/(p+1)} \times h_{\text{old}}$$

if an error-per-step strategy is used and the order of the numerical method is p. An easy check for trouble is thus that $h_{\text{new}} \ll h_{\text{old}}$ or that the local error estimate is much greater than ϵ.

As a matter of fact most variable-order variable-step codes already check the local error estimate at every step in order to determine whether to accept or reject the current step. It is therefore very easy to fit the discontinuity detection into the code at little extra expense. Only when a step is rejected should we do anything at all and in this case we should just check h_{new} against h_{old}.

It is a matter of discussion how small h_{new} must be, but we feel that if $h_{\text{new}} < h_{\text{old}}/2$, then we shall probably be better off assuming that the change in f is a discontinuity (possibly in a derivative) and is best handled by a special procedure. On the other hand, if $h_{\text{new}} \geq h_{\text{old}}/2$, then the code might be able to force its way through the trouble—whether it is a discontinuity or just a rapidly varying f—in possibly a couple of attempts, and we would not have to bother with activating the special code. The value 1/2 has no particular significance, but in terms of the discussion in the next section it is a very natural choice.

It should be mentioned that some discontinuities might avoid detection. These would probably be very easy to pass without the need to invoke a special routine. On the other hand, tests may be activated when f is not really discontinuous but just varies rapidly in a particular region. In this case it might just be sensible to assume that f is discontinuous and to act accordingly; but more about this in Section 7.

4. STAGE 2—DETERMINING THE ORDER, SIZE AND LOCATION OF THE DISCONTINUITY

Once we have detected what we believe to be a discontinuity, we must find its approximate location. If we know the value of ξ, then we should of course use it. If, as is often the case, we have knowledge of a steering function which triggers the discontinuity upon reaching a certain value, then we should use an inverse interpolation procedure [2, 5, 10] or some other method [1, 7, 9] to locate the discontinuity approximately. In most cases a few extra calculations will suffice to determine the location with sufficient accuracy that we can step right up to the discontinuity and pass it with an error less than the prescribed tolerance. We shall not go into more details with this problem but instead concentrate on the more difficult case where we have no extra information available, but must base our decisions on computed values of y and f exclusively.

If the discontinuity is of order 1 (as shown in Figure 3), then it is impossible to locate it on the basis of just a few values of f. The jump can occur anywhere in the last interval and the only way we can find it is by evaluating f at various points in this interval. The most efficient way of narrowing down the search is thus some kind of bisection procedure (which is also suggested in [8]). Also, when

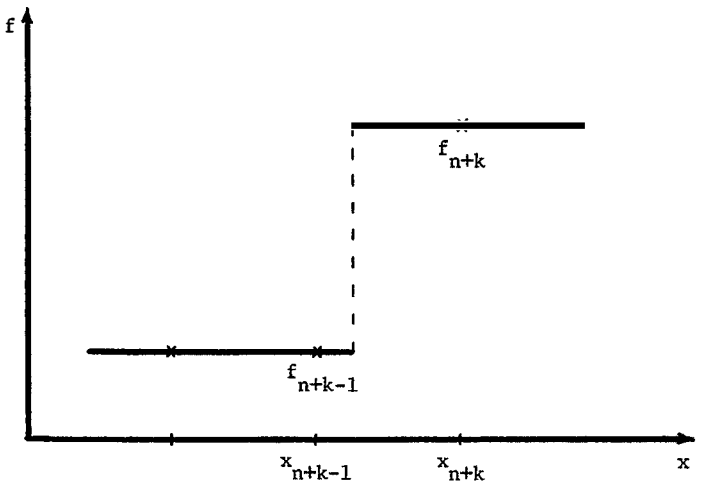

Fig. 3. A first-order discontinuity.

we first detect the discontinuity, we have no way of deciding the order so we must be prepared for the worst case (i.e., $q = 1$). As our basic method we therefore propose to do as follows.

(a) Halve the stepsize and take a step;
(b) if the step fails, then go to a;
(c) if we succeed, then advance the solution and go to a.

Since we assume the function to be smooth to the left of the discontinuity, we should stay with the same order method, but since we reduce the stepsize, we expect no stability problems and can save function evaluations by switching to PEC mode.

If a step is successful as indicated by the local error estimate being smaller than the tolerance, then we should not attempt a second step with the same stepsize until this has become smaller than h_{pass}. In most cases this second step is rejected and it gives us little extra information on the location of the discontinuity, so we might as well save the work and just go ahead with stepsize halving. If a step is rejected, we just divide h by two and try again.

After the first failed step in which we first detect the presence of a discontinuity we have no way of finding its order, but after several tries we may be able to estimate q and K_q using information from divided differences of f-values. To see how this can be done in a simple example let us first assume that $q = 1$ and that

$$f(x, y) = \begin{cases} 0 & x < \xi, \\ K_1 > 0 & x \geq \xi. \end{cases} \tag{4.1}$$

Let FR and FL denote the values of f at the right and left endpoints of the smallest step taken containing the discontinuity. In our example we have thus $FR = K_1$ and $FL = 0$. Define

$$d_1 = \frac{FR - FL}{h}, \tag{4.2}$$

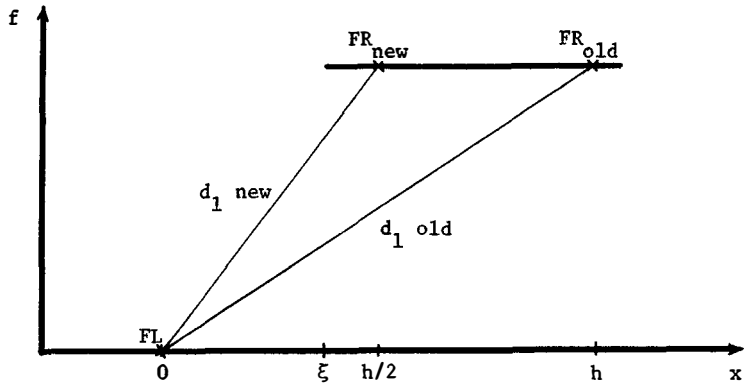

Fig. 4. A first-order discontinuity.

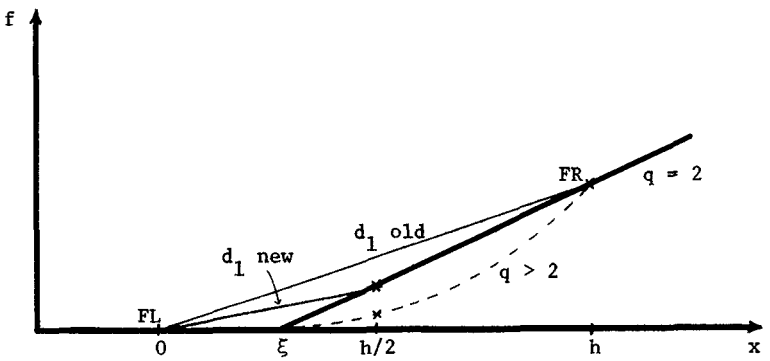

Fig. 5. A second- (or higher-) order discontinuity.

where h is the length of the step. We can compute d_1 as soon as a discontinuity is detected. It is the slope of the chord as shown in Figure 4. Now take a step of size $h/2$ and notice that the new value of d_1 is twice the old value independent of where in the interval the discontinuity is located. (In Figure 4 the discontinuity is located in the first half of the interval. If it had been located in the second half, the point marked *FL* would have moved to the right rather than *FR* moving to the left, as shown.)

If instead we have $q \geq 2$, then we notice that the new value of d_1 is smaller than the old value if the discontinuity is located in the left half of the interval, that is, if the halved step is a failure. This is shown in Figure 5. It is thus possible to distinguish between first- and higher-order singularities after a failed step.

After a successful step, d_1 is always doubled no matter what q is (*FR* stays the same and h is halved), so we get no new information on q. In case of doubt, we stay with an assumption of a lower-order discontinuity because this enforces a tighter stepsize control (cf. eq. (2.21)).

In the more realistic case when $f \neq 0$ before the discontinuity, we can assume

$$f = \tilde{f} + K_1 \times f_1, \tag{4.3}$$

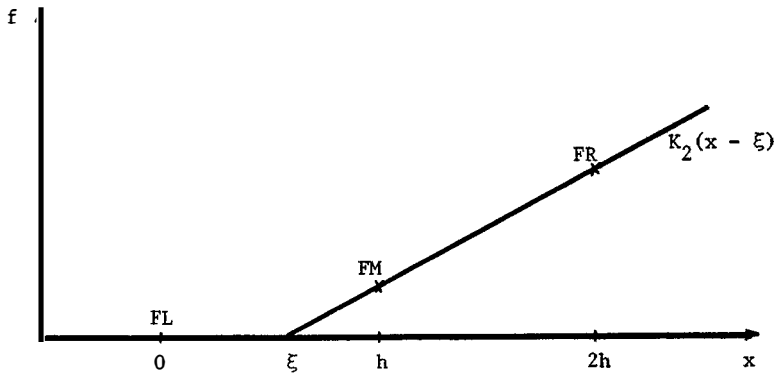

Fig. 6. A second-order discontinuity.

where $\tilde{f}$ is a smooth function and f_1 is a jump function:

$$f_1(x, y) = \begin{cases} 0 & x < \xi, \\ \text{continuously differentiable for} & x > \xi. \end{cases} \tag{4.4}$$

Since we are interested in the jump function, we must subtract out the smooth part by extrapolating from earlier values of f. We are then back to our example cases (show in Figures 4 and 5), with the exception that the function is not constant for $x > \xi$. We can therefore not expect exact doublings in d_1 but must allow for (small) changes in f_1. Still the difference between the behavior of d_1 after a failed step when $q = 1$ and when $q > 1$ is big enough for us to be able to decide with some confidence.

When dealing with a system of equations, d_1 should be taken to be a suitable norm of the vector difference of the (extrapolated) function values. Which norm we use is probably not important since the value will be dominated by the large variations in one (or a few) of the components whereas the smooth components will contribute very little.

We can now distinguish the case where $q = 1$ from that where $q > 1$. We can also go further and distinguish between $q = 2$ and $q > 2$ if we introduce a second divided difference

$$d_2 = \left\| \frac{\dfrac{FR - FM}{h} - \dfrac{FM - FL}{h}}{2h} \right\| = \left\| \frac{FR - 2 \times FM + FL}{2h^2} \right\|. \tag{4.5}$$

FM denotes the most recently computed f-value which is halfway between the points corresponding to FL and FR (see Figure 6). We can compute d_2 as soon as h has been halved once. A new value of d_2 can be computed after the second halving.

When analyzing the change in d_2 from one step to the next, we have to consider the success or failure of two consecutive steps. We have thus four different cases which we will denote FF, FS, SF and SS—F meaning fail, S meaning success—corresponding to the discontinuity being located in the first, second, third or fourth quarter of the large interval in Figure 6.

Table I. The Behavior of d_2 When $q = 2$ and $q \geq 3$

		d_2 is multiplied by		
	position of ξ	$q = 2$	$q \geq 3$	identification
FF	$(0, h/2)$	4	$(0, 1)$	+
FS	$(h/2, h)$	$(0, 4)$	$(0, 4/7)$	(+)
SF	$(h, 3h/2)$	$(0, 4)$	$(2, 4)$	(+)
SS	$(3h/2, 2h)$	4	4	—

We shall not burden the reader with unnecessary computational detail here, but refer the curious to [4]. Instead, we summarize the results in Table I which gives the multiplication factors for d_2 as a function of q and the success or failure of the last two steps.

When an interval is given for the multiplier, it means that d_2 is multiplied by some number in this interval depending on the exact location of the discontinuity and the value of q. In the last column of Table 1 a simple plus sign + indicates that we can determine whether $q = 2$ or $q > 2$ with confidence since d_2 is either quadrupled or reduced. A plus sign in parentheses (+) in the last column of Table I indicates that we can do a partial identification in the following sense:

- If, in case FS, we observe that d_2 is multiplied by more than 4/7, then we can conclude that $q = 2$ (and at the same time that $\xi \in (h/2, 4h/5)$).
- If, in case SF, we observe that d_2 is multiplied by less than 2, then we can again conclude that $q = 2$ (and that $\xi \in (h, 6h/5)$).

On the other hand, if $q \geq 3$ and/or if $\xi \in (4h/5, h) \cup (6h/5, 3h/2)$, then we cannot conclude anything as to the value of q.

After two successful steps in a row (i.e., $\xi \in (3h/2, 2h)$), we have no information on q from the behavior of d_2.

It is not advisable to use the above numerical values of 4/7 and 2 in practical computations since we should allow for continuous variations in f. Therefore 4/7 should be replaced by a slightly larger constant, such as 0.65, and 2 should be replaced by a slightly smaller constant, such as 1.9, to prevent misinterpretation of results.

In principle it is also possible to distinguish between $q = 3$ and $q > 3$ by introducing third divided differences, but the analysis is more complicated and probably not worthwhile since discontinuities of orders greater than 2 are usually rather easy to pass because of a reasonably large value of h_{pass}.

As can be seen from Figure 6, it is possible to locate the discontinuity quickly if we know that $q = 2$ and we have had at least two failed steps (which we must have had for we cannot decide that $q = 2$ on the basis of successes only). A linear extrapolation through FR and FM yields

$$\xi \approx \alpha = x_M - FM \times \frac{X_R - X_M}{FR - FM} = x_M - FM \times \frac{h}{FR - FM}, \tag{4.6}$$

and we can step right up to the discontinuity by choosing the next stepsize equal to $h = \alpha - x_L$. Of course we must be reasonably sure that the order of the discontinuity is actually 2 before we use this estimate of ξ, so we wait until we

have two consecutive indications of $q = 2$ together with a check that the estimate of ξ did not change appreciably (i.e., by more than $h_{\text{pass}}/2$).

In principle it is also possible to locate ξ when $q > 2$, but first of all it would require us to determine the correct order, and even if we knew q, the intersection between the axis and a parabola or a higher order curve will be very sensitive to small errors in the determination of FM and FR. We shall therefore not try to estimate ξ when $q > 2$ but rely on the observation that such discontinuities can be crossed with moderately sized steps (cf. eq. (2.21)).

Out of these considerations a strategy emerges for locating the discontinuity, determining the order q and estimating the size of the jump. In the following, FR and FM indicate function values of the (vector-valued) function f with the smooth part—as determined by (componentwise) extrapolation from earlier values of f—subtracted out. When calculating K_1, K_2, d_1 and d_2, a suitable norm should be used. Below, d_1' and d_2' denote previous values of d_1 and d_2.

(1) When a discontinuity is suspected because $h_{\text{new}} < h_{\text{old}}/2$, assume the worst case; that is, set $q = 1$ and compute d_1, K_1 and h_{pass} accordingly:

$$K_1 = \| FR - FL \|; \qquad d_1 = K_1/h; \qquad h_{\text{pass}} = \epsilon/K_1. \tag{4.7}$$

Halve the stepsize, but keep the same order. A switch to PEC mode is made to give fewer function evaluations.

(2) Take a step and compute d_1 and d_2. If the step fails and $d_1 \approx 2d_1'$, then note that $q = 1$ and recompute K_1 and h_{pass}. If $d_1 < d_1'$, then note that $q > 1$. Assume the worst case and set $q = 2$. Compute

$$K_2 = \frac{\| FR - FM \|}{h}; \qquad h_{\text{pass}} = \sqrt{\frac{\epsilon}{K_2}}; \quad \alpha = x_M - \frac{\| FM \|}{K_2}. \tag{4.8}$$

(3) In either case halve h and try another step (from x_L or from x_M, depending on whether the previous step was a failure or a success).

(4) In the general situation when previous values of d_1 and d_2 are available, we do the following:

If the step fails and $d_1 > d_1'$, then
  If $q \neq 1$ then set $q = 1$, compute K_1 and h_{pass},
  otherwise confirm $q = 1$, compute K_1 and h_{pass}.
  Do not look at d_2 at all.
If the step fails and $d_1 < d_1'$, then
  If the previous step failed, we are in case FF.
    If $d_2 \approx 4d_2'$ and $q \neq 2$, then set $q = 2$.
      If q was already 2, then confirm $q = 2$.
    Compute K_2, h_{pass} and α according to (4.8).
  If $d_2 < d_2'$, then set $q = 3$.
      If q was already 3, then confirm $q = 3$.
    Compute $K_3 = 2d_2$; $h_{\text{pass}} = \sqrt[3]{2\epsilon/K_3}$
    ($q = 3$ here means $q \geq 3$).
  If the previous step was a success (case SF)
    and $d_2 < 2d_2'$, then set or confirm $q = 2$ (see (*)).

(*)

If the step succeeds and the previous step failed and $d_2 > 4/7d'_2$, then set or confirm $q = 2$ (see (*)).

(5) Go to 3.

In addition to the actual value of q as it appears in a computer program, we need a set of flags to indicate the degree of confidence we have in the value. This is done in our code by means of an extra integer variable which is given the value 1 the first time the value of q is determined and increased by 1 each time this value of q is confirmed. The value 0 is used to indicate that the value of q has not yet been determined, as is the case when we first detect the discontinuity (when we set $q = 1$ but actually mean $q \geq 1$) or when on the basis of d_1 and d'_1 we determine that $q \geq 2$ (and we set $q = 2$).

5. STAGE 3—PASSING THE DISCONTINUITY

The above procedure for approaching and locating the discontinuity should now be augmented so that it can also pass the discontinuity. The basic rule is that if $h \leq h_{\text{pass}}$ and the last step was successful, then we should attempt a second step of the same size. If this is also successful, then we assume that we have safely passed the discontinuity and we can go to the restarting procedure (Section 6).

If we have confirmed that $q = 2$, then we can take a step right up to the estimated location of the discontinuity, α, and, if this step is a success, go to the restarting procedure.

When $q = 1$ we actually do not need to wait until $h \leq h_{\text{pass}}$ if we instead look a little more closely at the local error estimate which tells us whether a given step was accepted or not. Since we use the same order method as before the discontinuity was detected, and have halved the stepsize a number of times, say m, we expect the local error estimate to be well below the tolerance as long as we are to the left of the discontinuity. Even at first order, the local error should not exceed $4^{-m}\epsilon$. Therefore, if a step barely passes our test after at least two step halvings, say if

$$\epsilon/10 < \text{error estimate} < \epsilon, \tag{5.1}$$

then we have passed the discontinuity in this step and can go right on to the restarting procedure (Section 6).

When $q = 1$ we have another way of checking whether we have actually passed the discontinuity. The accepted f-value must then correspond to the value after the discontinuity so FM must be much closer to FR than to FL. It is rather important for the accuracy of the restarting procedure that this be true.

When $q = 2$ and we use our estimate for the location of the discontinuity, we are usually well within a distance h_{pass} from the discontinuity so that a step of size h_{pass} in most cases will be sufficient to cross it. Should we not actually cross it, we should now be very close to it. Although in the restart process f may be evaluated on the wrong side of the discontinuity, the error thus introduced is small because f is continuous.

When $q > 2$ or when $q = 2$ but has not been confirmed, we might either check the local error estimate or $h < h_{\text{pass}}$. Usually h_{pass} is so large that either test will be satisfied rather quickly.

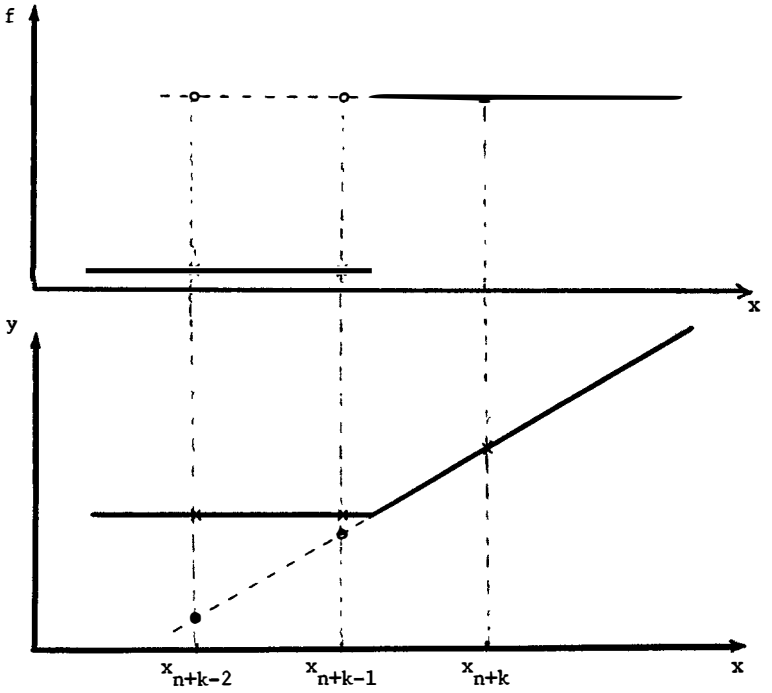

Fig. 7. A discontinuity of order 1.

6. STAGE 4—RESTARTING

When we have passed the discontinuity, we should start up the usual integration routine again using as much of the previously obtained information as is profitable. We first discuss the problems encountered when using a self-starting variable-order variable-step code based on linear multistep formulas.

Because of the discontinuity, back values of y and f do not correspond to those of smooth functions but differ by terms of the order $O(h^q)$ and $O(h^{q-1})$, respectively, when the discontinuity is of order q. See Figure 7 for an illustration of this when $q = 1$. To include such values in our linear multistep formulas would in turn give rise to terms in the local truncation error of order $O(h^q)$. It is therefore not advisable to use any back values when $q = 1$.

Since Euler's method does not use back values, it is always safe to use as a predictor. For the corrector we can use either backward Euler or the trapezoidal rule. The resulting local truncation error will be of order $O(h^2)$ or $O(h^3)$, respectively.

If $q = 1$ then we have already committed an error of order $O(h)$ in the passage of the discontinuity, although our step control mechanism has been designed to keep the size of the error less than the local error tolerance. It is essential for our results, however, that we pass the discontinuity and that the last value of f correspond to the right branch of that function.

If $q = 2$ we shall gain very little by raising the order of the restarting method, so we still recommend Euler's method combined with backward Euler or the

trapezoidal rule. Since f varies continuously, it is difficult to ascertain whether the last value is on the right branch. We must therefore be prepared for the case where we actually have not passed the discontinuity but are going to pass it in the "restarting" step. The effect will be similar to that of using back values: the local truncation error will be of order $O(h^2)$.

For $q \geq 3$ we should restart with a method of order no more than $q - 1$, since the local error is going to be $O(h^q)$ anyway. Since we do not distinguish between values of q greater than or equal to 3, we therefore recommend a second order restarting method unless the value of q is known to us from other considerations.

It is more difficult to suggest a stepsize for restarting, but since the passing stepsize according to eq. (2.21) satisfies

$$h_{\text{pass}}^q = \frac{(q-1)!}{K_q} \times \epsilon, \tag{6.1}$$

it probably has the right order of magnitude to be used with a method of order $q - 1$. For a first order discontinuity the value of h_{pass} is probably much smaller than needed and we suggest using a restarting stepsize closer to $(h_{\text{pass}})^{1/2}$.

If our method includes a Runge-Kutta starter [2], then we can make use of previous values of p and h. If we assume that our equations (or rather the function f) have not changed character but have only jumped to a different level, then it is a good guess that the order and stepsize which were used when we first detected the discontinuity can be used again after the discontinuity has been passed. We therefore recommend that the values of p and h be stored away at that point and used now. We can thus avoid the usual dilemma of starting an initial value problem from scratch when we often have little idea of an appropriate order and stepsize to use.

7. ROBUSTNESS

In practice things may not turn out quite as we have assumed which may lead to a misinterpretation of results and to wrong decisions about what to do. It is to be expected that a general program will keep functioning in a reasonable manner, giving results within the specified error tolerance with only a moderate loss of efficiency.

Among the things that can go wrong, we consider what happens when

(1) q is underestimated,
(2) q is overestimated,
(3) q is taken to be 2 and the (incorrect) estimate of ξ is used,
(4) the discontinuity is passed inadvertently, or
(5) there is actually no discontinuity but (the derivative of) f varies rapidly enough to suggest one to the code.

Since the estimated values of K_q and h_{pass} depend critically on the value of q, these will be determined incorrectly in cases 1 and 2. In particular, when we first detect the presence of a discontinuity, we have no information on the order and by setting $q = 1$ we often have a case 1 situation right away. If the next couple of steps are successful (because the discontinuity is located far to the right in the

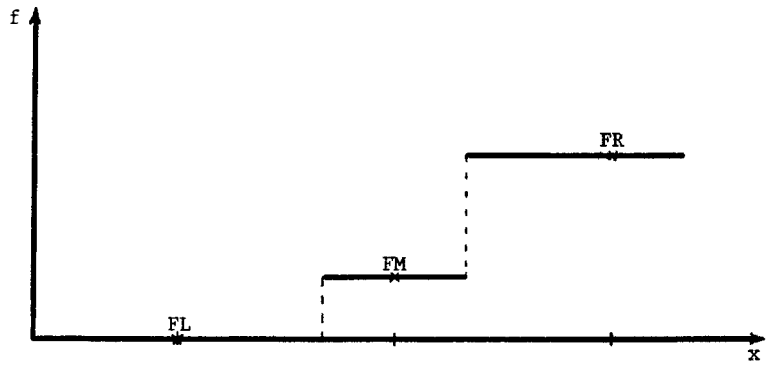

Fig. 8.

interval), we stay in case 1 for a while. The behavior of the algorithm in each of the five cases is discussed below.

Case 1. As an illustration assume that the order actually is 2 and that the size of the jump in y'' is K_2. The passing stepsize is therefore

$$h_{\text{pass}} = \sqrt{\epsilon/K_2}, \tag{7.1}$$

where ϵ is the local error tolerance. Since we believe that $q = 1$ we actually compute

$$K_1^c = |FR - FL| \approx K_2 \times (h - \xi) \tag{7.2}$$

and

$$h_{\text{pass}}^c = \frac{\epsilon}{K_1^c} \approx \frac{\epsilon}{K_2(h - \xi)} = h_{\text{pass}} \times \frac{h_{\text{pass}}}{h - \xi}. \tag{7.3}$$

The superscript c indicates that these are computed values; h is the current stepsize and ξ the location of the discontinuity.

We see from eq. (7.3) that h_{pass}^c is a conservative value ($h_{\text{pass}}^c < h_{\text{pass}}$) when $h_{\text{pass}} \ll h$ and ξ is not too far to the right in the interval.

If $h - \xi$ is actually very small, we shall have several successful steps and not be able to correct the value of q, in which case we might reach the point where the stepsize is less than h_{pass}^c and attempt a "second step" prematurely. Since FR corresponds to a failed step, and we are basically recomputing that value, this "second step" will also fail. This second failure will not give us any more information on the order of the discontinuity, but it will at least force us to halve the stepsize and try again. We may thus lose efficiency either by failing steps or by using a too small h_{pass}, but we shall not lose much and not at the expense of accuracy.

Case 2. If we overestimate the value of q, as might happen with the discontinuous function shown in Figure 8, then we usually overestimate h_{pass} also. This could cause us to try to step over the discontinuity too soon, which would result in an unnecessary failure. This cannot happen more than once per step halving,

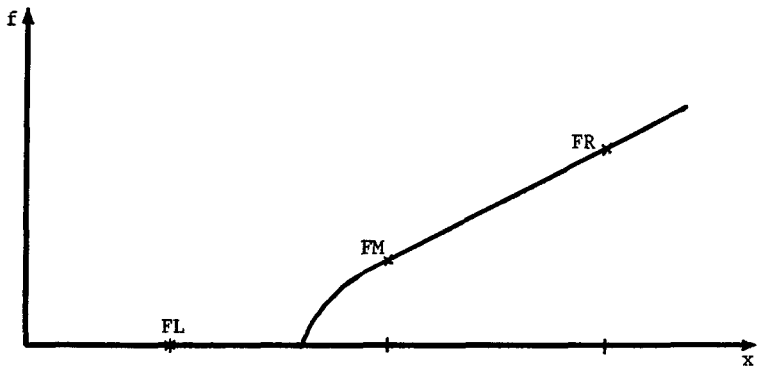

Fig. 9.

so, even in the unlikely event that we never realize the true nature of the discontinuity, we shall not use more than twice the optimal number of steps and should still be able to perform better than a standard code.

Case 3. A more serious by-product of the erroneous determination of q shows up when q is determined to be 2 (as in Figure 8) and we therefore use the estimate of eq. (4.6) for the location of the discontinuity. This is the reason why we should be careful with this estimate and only use it when we have confirmed the determination of q at least once and have assured ourselves that the last two estimates have not differed by much (say, by $h_{\text{pass}}/2$).

Still the estimate might be so large that the next step fails, in which case we should resort to the old step-halving strategy (so we must remember the old stepsize such that we can carry on as if we never had an estimate of the discontinuity).

If the estimate is too small, as could happen if the discontinuity looks like Figure 9, then we are still to the left of the discontinuity and, if we do not pass it in the next step of size h_{pass}, we shall have to repeat the whole process, from detecting the discontinuity (which is still the same one), to locating it, etc.

When worst comes to worst, we might even misjudge the location again, even though we still have the correct value of q (= 2), and we may have to switch the linear extrapolation (eq. (4.6)) off completely and just rely on stephalving. (This has not been implemented for the test code.)

Case 4. If during the process of locating the discontinuity we actually pass it without realizing it, we have two kinds of problems.

First, the next step or steps will be taken with a method of the same order, which is probably too high for restarting (cf. Section 6), leading to larger errors than expected. But we have just passed the discontinuity with a successful step and the next step will probably be taken with half the stepsize: although the new error will have the same order, it will have smaller magnitude. But once again we have to note that the usual error estimate might be too optimistic (cf. Section 2).

Second, we will almost certainly have successful steps from now on, so we shall get no new information on q, K_q or h_{pass}. If we are using a too small value of q and therefore of h_{pass}, we might halve the stepsize many times, just to build it up again after finally having decided to restart.

It is therefore important to check the local error estimate carefully and distinguish between two types of successful steps: those with a very small local error, indicating that we are to the left of the discontinuity, and those that are barely less than the tolerance, indicating that the discontinuity was located in the last interval, which allows us to go straight to the restarting procedure.

Another strategy which could alleviate some efficiency problems (but no accuracy problems), would be to try a "second step" once in a while, just to see if the discontinuity is still there. (By a "second step" we mean that after a successful step we would not do the automatic halving in the location algorithm.) This strategy might be especially useful in Case 5.

Case 5. Some problems (e.g., in the modeling of diodes), involve rapidly varying functions which actually do not contain any discontinuities (mathematically speaking) even though our procedures mistakenly detect a discontinuity. This often reflects the fact that these functions can be approximated quite well by discontinuous ones. Unfortunately, as the stepsize is reduced the apparent discontinuity often disappears (in the sense that we keep making successful steps), although the independent variable is not being advanced very much.

The best solution to this problem seems to be to allow "second steps" quite often, so that we can quickly realize when the stepsize is small enough for successful integration. Since there are no discontinuities, there is also no reason to use the special restarting procedures of Section 6, but this kind of information is difficult to build into a general program. The step-halving strategy, however, is probably more efficient than traditional step selection schemes which can be expected to reduce the stepsize drastically when they first encounter variations resembling discontinuities.

8. NUMERICAL RESULTS

The algorithm described in the previous sections has been added to an experimental code we have developed, and several problems have been run, with and without the discontinuity checker turned on. The code uses a PECE Adams method with a Nordsieck representation and interpolatory step changing. Although the code has no known bugs, it has neither been extensively tested nor tuned, so it is possible that a production quality code would perform differently and change the results slightly. It is also probable that the use of a modified divided difference representation with a variable step mechanism would change the results. (We hope to have a similar code based on divided differences soon and will attempt such comparisons.) It is also likely that a different stepsize/order selection strategy would change the results. However, in spite of these reservations, we believe that the numerical results presented below give a qualitative validation of the ideas presented in earlier sections.

Several test problems are reported. The first was the example used in Figure 1 (p. 25), namely

$$y' = \begin{cases} 0 & x < 40.33, \\ 100 & x > 40.33, \end{cases} \qquad y(0) = 40.33. \tag{8.1}$$

This is trivial and was used principally to test the code. It is reported as a basis of comparison with the results shown in Figure 1. The second problem set is

$$y' = \begin{cases} -y & y \geq 0.75, \\ -y[1 + (y - 0.75)^{q-1}] & y < 0.75, \end{cases} \quad y(0) = 1 \tag{8.2}$$

for $q = 1$, 2, and 3. The order of the discontinuity is q. The next four problems have discontinuities of orders 1, 2, 3, and 3, respectively. They are

$$y' = \begin{cases} -y & x \leq 1, \\ +y & x > 1, \end{cases} \quad y(0) = 1, \tag{8.3}$$

$$y' = \begin{cases} 0 & x \leq 1, \\ 10(x - 1) & x > 1, \end{cases} \quad y(0) = 1, \tag{8.4}$$

$$y' = \begin{cases} 0 & x < 1, \\ 100(x - 1)^2 & x > 1, \end{cases} \quad y(0) = 1, \tag{8.5}$$

$$y' = \begin{cases} 0 & x \leq 0.74, \\ 100(x - 0.74)^2 & x > 0.74, \end{cases} \quad y(0) = 1. \tag{8.6}$$

Finally, a problem with a physical basis was chosen. It is a model of a one-dimensional spring/dashpot system with a limit stop such as is found in an automobile suspension. A periodic driving force is applied.
The equations are

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -y_2 - 10(y_1 + \sin x + F(y_1)), \end{aligned} \tag{8.7}$$

where

$$F(y) = \begin{cases} 0 & y \geq -0.1, \\ 10(y + 0.1) & y < -0.1, \end{cases}$$

$$y_1(0) = y_2(0) = 0.$$

The discontinuity is of second order. In all examples a local absolute error per step tolerance of 10^{-5} was used. (Absolute error per step is not necessarily recommended. The choice depends on the problem. Here we just wish to compare two schemes under identical conditions.)

The results are given in narrative fashion as it is difficult to present the information in tabular form. For the first problem, eq. (8.1), the stepsize at $x = 10.5$ was increased to 31.6. The order was still 1, as is best for the trivial problem $y' = 0$. Since the next step crossed the discontinuity, the step was reduced. Without the scheme described, the code took 64 PECE steps, many unsuccessful, until the step was down to 0.627^{-5} and the discontinuity was passed. This represents 128 function evaluations. With detection the step was halved 24 times down to 0.18^{-5}. The passing stepsize was less, 0.1^{-6}, but at that time the step was successful with error estimate $> \epsilon/10$ (see eq. (5.1)), so the integration was restarted at order 1. The new technique used about 80 percent fewer function evaluations, but reduced the step further, making it likely that it will use more steps to get the stepsize large again, but also that it will introduce a local error

that is smaller than the tolerance. The standard method may introduce a larger local error depending on the location of the discontinuity in the step that crosses it.

For $q = 1$, eq. (8.2) gives similar results. Prior to the singularity at $x = 0.2877$, the stepsize was 0.097 and the order was 3. Without detection, the code reduced the stepsize to 0.12^{-3} in 18 steps (36 function evaluations) but had managed to increase the order to 4 (there was a string of successful steps after a large reduction). With detection, 10 step size halvings were used to reduce the step size enough so that the discontinuity could be passed. As in the previous case, the code was certain that q was one (three confirmations) and that it had an estimate of K_1 good to nearly four digits. The savings in function evaluations was about 70 percent. For $q = 2$ and $q = 3$, the discontinuity detection was never invoked because the first step to cross the discontinuity only just barely crossed it. In this case the stepsize reduction is small and there is no reason to believe that anything is wrong. In the case $q = 2$, the first step across failed, the reduced step did not cross but the next did and also failed. One more reduction was sufficient to cross the discontinuity at order 3. In the case $q = 3$ the step crossing the discontinuity was successful but the next step failed. One reduction allowed the code to continue at order 3.

Equation (8.3), a discontinuity of order 1, was passed in 12 steps without detection starting from a stepsize of 0.178 at order 4. With detection, the order, jump and location were determined after 11 reductions: about a 50 percent savings.

Equation (8.4) required 38 steps to pass the second order discontinuity with detection at order 1. Seven halvings were sufficient, by which time the code had computed K_2 and the location of the discontinuity to at least four digits. The value of q had been confirmed three times. At that point it stepped to the discontinuity and restarted.

Equation (8.5) required 20 steps without discontinuity detection, but five halvings were sufficient with detection. By that time the code suspected $q = 2$ but had no confirmation, so no estimate of the location of the discontinuity was made.

Equation (8.6) is just a shift of equation (8.5), but illustrates the importance of luck. Without detection, four steps were sufficient to cross the discontinuity at order 1 because a mesh point landed just to the left of it. With detection, eight halvings were used to reduce the step below the passing size. After six halvings the code had four confirmations that q was equal to 3, but after the next two halvings had reason to suspect that perhaps q was equal to 2.

For the spring system in eq. (8.7), the code was at order 4 when y_1 first dipped below -0.1. Without detection, there were four step failures, but the order remained at 4. With detection, eight halvings were used and $q = 2$ was suspected but never confirmed. The restart at order 1 was more expensive than just continuing, but also more reliable. When y_1 passed -0.1 in the other direction, detection failed: the stepsize was smaller because of the higher speed of the system when $F(y_1)$ was nonzero, and the order was 5 in the case of no detection, and 6 otherwise. (By this point the codes had different step sequences.) In both cases there were six failures caused by the (undetectable) discontinuity. The code

with detection located the next discontinuity, but again failed to find the fourth when y increased past -0.1. By this time the entering stepsizes were sufficiently different that there was little useful comparative information to be gleaned.

9. CONCLUSIONS

An algorithm has been described that can detect and locate some discontinuities and provide information about their size, order and position. However, the success of the algorithm is strongly dependent on the location of the discontinuity with respect to the steps that straddle it. The major advantage of the scheme appears to be that a more reliable error estimate can be used when a discontinuity is present so that codes will be more robust. In some cases significant savings may accrue but it appears that a better restarting procedure than we used will be necessary to realize most of those benefits.

Note added in proof. The use of binary search is also suggested in H. J. Stetter's "Modular Analysis of Numerical Software," in the proceedings of Numerical Analysis, Dundee, 1979, edited by G. A. Watson, and published as volume 773 in Springer-Verlag's Lecture Notes in Computer Science (New York, 1979).

ACKNOWLEDGMENTS

The programs used in the discontinuity detection were written by Mr. Martin Wong, who also ran the tests. The experimental integrator used was written by Dr. Kyle Gallivan.

REFERENCES

1. CARVER, M.B. Efficient integration over discontinuities in ordinary differential equation simulations *Math Comput Simul. 20*, 3 (1978), 190–196
2. ELLISON, D. Efficient automatic integration of ordinary differential equations with discontinuities. *Math. Comput. Simul. 23*, 1 (1981), 12–20.
3. GEAR, C.W. Runge-Kutta starters for multistep methods. *ACM Trans Math. Softw 6*, 3 (1980), 263–279.
4. GEAR, C.W. AND ØSTERBY, O. Solving ordinary differential equations with discontinuities Tech. Rep. R-81-1064, Univ. of Illinois at Urbana-Champaign (1981).
5 HAY, J.L., CROSBIE, R.E AND CHAPLIN, R.I. Integration routines for systems with discontinuities *Comput. J 17*, 3 (1974), 275–278.
6. HINDMARSH, A. GEAR: Ordinary differential equation solver. Tech. Rep UCID-30001, Revision 3, Lawrence Livermore National Laboratories, Livermore, Calif., 1974.
7. MANNSHARDT, R. One-step methods of any order for ordinary differential equations with discontinuous right-hand sides. *Numer. Math. 31*, 2 (1978), 131–152.
8. MUNTER, P. Numeriske metoder til løsing of systemer of saedvanlige differentialligninger indeholdende diskontinuiteter. Rapport F40-78.03, Laboratoriet for Køleteknik, Technical University of Denmark, 1978 (report is in Danish)
9. O'REGAN, P.G Step size adjustment at discontinuities for fourth order Runge-Kutta methods. *Comput. J 13*, 4 (1970), 401–404.
10 THOMSEN, P.G. Numerical simulation in theory and praxis. Mathematics and Computation Tech. Rep. 2/79, Department of Mathematics, University of Trondheim, Norway 1979.

Received October 1981; revised August 1983; accepted August 1983