

Event Location for Ordinary Differential Equations

L.F. Shampine
Mathematics Department
Southern Methodist University
Dallas, TX 75275
lshampin@mail.smu.edu

S. Thompson
Department of Mathematics & Statistics
Radford University
Radford, VA 24142
thompson@runet.edu

February 9, 2000

Abstract

An initial value problem for $y' = f(t, y)$ may have an associated event function $g(t, y)$. An event is said to occur at t^* when $g(t^*, y(t^*)) = 0$. We consider problems for which the definition of $f(t, y)$ changes at the time of an event. A number of solvers locate events and restart the integration there so as to deal with the changes in f , but there is little theoretical support for what is done. Here we prove that with reasonable assumptions about the problem and the solver, the error of the numerical solution is qualitatively the same whether or not events occur. Numerical results obtained with a wide range of solvers confirm the theory developed here.

1 Introduction

There are many effective solvers based on a variety of methods and implementations for approximating the solution $y_L(t)$ of the initial value problem (IVP) for a system of ordinary differential equations (ODEs),

$$y' = f_L(t, y) \tag{1}$$

with initial value $y(a) = A$. A few solvers have an event location capability. By this is meant that associated with the IVP is an event function $g(t, y)$. At the first occurrence t^* of an event, i.e., the first solution of $g(t^*, y_L(t^*)) = 0$, the

integration is terminated and restarted with a new set of differential equations. Specifically, $y_R(t)$ is defined as the solution of

$$y' = f_R(t, y) \tag{2}$$

with initial value $y(t^*) = y_L(t^*)$. If there is no other event prior to $t = b$, the solution $y(t)$ of the IVP on $[a, b]$ is defined as $y_L(t)$ for $[a, t^*]$ and $y_R(t)$ for $[t^*, b]$. A situation easily imagined is a set of ODEs describing the temperature in an office. When it gets too hot, a thermostat causes an air conditioning unit to switch on. The temperature is then described by a different set of ODEs until the office cools to the point that the unit is switched off and the cycle repeats. Problem 3 of [6] is a simple model of this kind:

$$y' = \begin{cases} y & \text{for sw} = 1, \\ -y/2 & \text{for sw} = 0 \end{cases} \tag{3}$$

with $y(0) = 1$ to be solved on $[0, 10]$. Here sw is set initially to 1 and is reset to 0 when $g_1(t, y) = y - 2 = 0$. The switch sw is reset from 0 to 1 when $g_2(t, y) = y - 1 = 0$. The solution is a sawtoothed function of period $3 \ln 2$ with two discontinuities per period. Enright et alia [6] solve such problems by integrating the current ODEs until it is found that the current event function has a change of sign in the course of a step from t_n to t_{n+1} . They consider solvers that produce a piecewise polynomial solution $y^{**}(t)$ that is accurate throughout $[t_n, t_{n+1}]$. The event is located in this interval by solving very accurately $g(t, y^{**}(t)) = 0$ for its first root t^{**} . The integration is then restarted from $(t^{**}, y^{**}(t^{**}))$ with the new definition of the ODEs. Enright et alia discuss other ways of proceeding, but this one is typical of a modern solver with event location capability. Numerical results for this problem presented in Table 5 of [6] show that the IVP is solved accurately at a cost scarcely greater than when no events occur. It is striking that the behavior of the error as a function of the tolerance τ is qualitatively the same as when no events occur.

Event location arises naturally in many models as a way of dealing with discontinuous behavior. A nice survey of the task and methods that have been proposed for dealing with it form Chapter 6 of the monograph [5] on the dynamics of multibody problems. The task also arises in the solution of delay-differential equations because of discontinuities that are induced by a lack of smoothness and propagated by the effects of delays, c.f. [4, 13]. Despite a good deal of attention given to the task, little has been done to show that the numerical procedures proposed actually “work”. The most general results are those of Mannshardt [12] who investigates one-step methods. He shows that with suitable assumptions, if a one-step method of order p is used to integrate the IVP with constant step size h and events are located accurately enough, the numerical approximation converges uniformly of order p over the whole interval of integration. We prove here similar convergence results with assumptions that describe most modern IVP solvers. Specifically we assume that the solver produces a piecewise polynomial solution $y^{**}(t)$ so that it is practical to locate events as accurately as possible in the precision available. We assume that when

given a tolerance τ , the solver produces a solution $y^{**}(t)$ that is uniformly accurate of order $r(\tau)$. We assume only that $r(\tau)$ is monotone and $r(\tau) \rightarrow 0$ as $\tau \rightarrow 0$. To be concrete, $r(\tau) = \tau$ for some popular methods and implementations. Also, for a fixed order p , other implementations have $r(\tau) = \tau^{(p+1)/p}$. Whatever the rate of convergence, we prove that the same rate is achieved in the presence of isolated events. For some methods and implementations it is known that the solver produces a solution with an error at t that to leading order is equal to $e(t)r(\tau)$ when there are no events. We go on to prove that with such a solver, the error is everywhere proportional to $r(\tau)$ except in intervals of length $O(r(\tau))$ containing the events. These theoretical results are in accord with the numerical results just cited from [6], and we provide numerical results obtained with a wide range of solvers that confirm this behavior.

2 Well-posed Problems

We assume that the function f_L of (1) is as smooth as necessary for the arguments that follow. It is then a standard result that the IVP consisting of (1) and $y(a) = A$ is well-posed. If the first event occurs at t^* , $y_R(t)$ is defined as the solution of (2) with initial value $y(t^*) = y_L(t^*)$. Here f_R is another smooth function. It is worth remark that the initial value for the second integration might be related to $y_L(t^*)$ in a more complicated way. The classic bouncing ball problem is an example for which the sign of a component is changed and the magnitude is reduced by a coefficient of restitution at the event of the ball rebounding from the ground. Our analysis applies to such problems, but for simplicity we restrict ourselves to initial values obtained by continuity.

We study the solution $y(t)$ of the initial value problem on an interval $[a, b]$ for which there is only one event at t^* with $a < t^* < b$ by defining $y(t) = y_L(t)$ for $t \leq t^*$ and $y(t) = y_R(t)$ otherwise. The assumption that the event is isolated has implications that we take up below. It is necessary that both f_L and f_R are defined and smooth on an interval containing t^* so that $y_L(t)$ is defined for some $t > t^*$ and $y_R(t)$ for some $t < t^*$.

Example 4 of [16, p. 74] presents a concrete example of an ill-posed problem with an event function depending on y . It goes on to give an informal proof that the task is well-posed when the event function depends only on t . Here we discuss similarly the general problem. The IVP for $y_L(t)$ is well-posed on an interval containing $[a, t^*]$, so small changes to the initial conditions and to f_L lead to a small change $\delta(t)$ in $y_L(t)$. This induces a small change ϵ in the location t^* of the event that we approximate by expanding about $(t^*, y_L(t^*))$ the expression

$$0 = g(t^* + \epsilon, y_L(t^* + \epsilon) + \delta(t^* + \epsilon)) = \frac{\partial g}{\partial y} \delta(t^*) + \epsilon \left[\frac{\partial g}{\partial t} + \frac{\partial g}{\partial y} \frac{dy_L}{dt} \right] + h.o.t.$$

where *h.o.t.* are higher order terms and each term on the right is evaluated at

$(t^*, y_L(t^*))$. After recognizing that the quantity in brackets is g' , we see that

$$\epsilon = -\frac{\partial g}{\partial y}(t^*, y_L(t^*)) \frac{\delta(t^*)}{g'(t^*, y_L(t^*))} + h.o.t.$$

Key to a well-posed problem is the assumption that t^* is a simple root, or equivalently, that $g'(t^*, y_L(t^*))$ is not zero. For the sake of definiteness, let us assume that $g(a, A) < 0$ so that this derivative is positive. For an event that arises as a simple root, this argument shows that the change ϵ in the location t^* of the event is of the same order as the change $\delta(t)$ in the solution $y_L(t)$.

This last assumption is a transversality condition to the effect that the solution $y_L(t)$ crosses the hypersurface $g(t, y) = 0$ at t^* . It is illuminating to write the condition in terms of f_L :

$$g'(t^*, y_L(t^*)) = \left(\frac{\partial g}{\partial t} + \frac{\partial g}{\partial y} f_L \right) (t^*, y_L(t^*)) > 0$$

In this form it is clear that there must be a corresponding condition involving f_R if the solution $y_R(t)$ is to leave the hypersurface at t^* , namely

$$g'(t^*, y_R(t^*)) = \left(\frac{\partial g}{\partial t} + \frac{\partial g}{\partial y} f_R \right) (t^*, y_R(t^*)) > 0$$

There are practical problems that involve solutions that slide along the hypersurface, but here we investigate only problems for which the solution crosses the hypersurface. As we discuss below, in practice events must be clearly separated. To this end (and for technical reasons), Mannshardt [12] assumes that $g'(t^*, y_L(t^*)) \geq G > 0$ and $g'(t^*, y_R(t^*)) \geq G > 0$. Rather than make a formal assumption of this kind, we make the equivalent informal assumption that events are clearly separated.

In the unperturbed problem $y_R(t)$ is defined for $t \geq t^*$ as the solution of an IVP with initial data $(t^*, y_L(t^*))$ and smooth function f_R . In the present situation the initial point is perturbed to $t^* + \epsilon$ and the initial value to

$$y_L(t^* + \epsilon) + \delta(t^* + \epsilon) = y_L(t^*) + \epsilon f_L(t^*, y_L(t^*)) + \delta(t^*) + h.o.t.$$

With our assumptions small changes to the data of the first part of the integration induce small changes in the location of the initial point and in the initial value for the second part of the integration. It is shown in [3] that the solution of an IVP with smooth function f_R is well-posed with respect to variation of the initial point t^* as well as the initial value $y_L(t^*)$. Accordingly, these changes induce only small changes in the solution in the second part of the integration. We conclude that if there is a single event in $[a, b]$ where $y(t)$ crosses the hypersurface $g(t, y) = 0$ and all the functions of the problem are sufficiently smooth, then the IVP is well-posed on $[a, b]$. Repetition of the argument shows that the same is true when there are several well-separated, simple events in the interval.

There are some practical difficulties related to well-posedness that merit discussion. It is not unusual that there be more than one event function and

sometimes there are many. In both theory and practice the event functions are treated independently and the first event is the one that determines the change in the differential equations. The difficulty is that the task might be well-posed for each event function, but not for the group. Clearly the task might be mathematically ill-posed if an event occurs simultaneously for more than one event function. This seems an unlikely situation, but events are located using approximate solutions $y^{**}(t)$ and it is quite possible that close events occur in a different order when $y^{**}(t)$ is used in place of $y(t)$. In practice we have to assume that events are sufficiently well separated for the tolerance used in their computation that their numerical approximations occur in the correct order.

To understand other difficulties, let us imagine a solution $y(t)$ that is a parabolic arc with maximum value 2 at $t = 1$. If the event function is $g(t, y) = y - \gamma$, there are two events near $t = 1$ for γ a little less than 2. These events come together and merge as $\gamma \rightarrow 2$. For $\gamma = 2$ the task is not in the class we treat because the event is not a simple root; it is ill-posed because there is no event for γ a little bigger than 2. It is well-posed in principle for $\gamma < 2$, but in practice it becomes increasing ill-conditioned as $\gamma \rightarrow 2$ because events are located using numerical approximations $y^{**}(t)$ to $y(t)$; for a given tolerance it is not possible to distinguish the two events when γ is sufficiently close to 2. It is also ill-conditioned in practice even when $y(t)$ is approximated well for a reason we now take up.

An issue of great practical importance is whether the solver “notices” an event. A variety of methods have been proposed for locating events [5], but the typical modern code monitors the sign of the event function and searches for an event in the span of a step from an approximation $y^{**}(t_n)$ to an approximation at $t_{n+1} = t_n + h$ only when $g(t_n, y^{**}(t_n))g(t_{n+1}, y^{**}(t_{n+1})) \leq 0$. It is very important to appreciate that solvers choose the step size so as to compute $y(t)$ accurately, not to resolve changes in the event function. If the step size is large enough that two simple events occur in $[t_n, t_{n+1}]$, the solver will not notice that an event has occurred because there is no sign change. Because of this the assumption that events are isolated is a more serious matter in practice than it might at first seem. Carver [2] suggests an approach to event location that responds to this difficulty. Assuming that the event function can be differentiated conveniently, he adjoins a differential equation for the event function to the system defining $y(t)$. In this way the step size is selected not only to resolve changes in $y(t)$, but also in $g(t, y(t))$.

A difficult software issue arises when restarting after an event. Solvers use a variety of approaches to avoid reporting the same event more than once. Among them are not checking for the event at the initial point, not checking for events during the first step, and not reporting events which are within a prescribed minimum distance from the initial point. In our experimentation with a wide range of solvers reported in §4, we encountered some failures for this reason. For a wide range of tolerances, DDRIV2 [9] fails at one or another of the events of the F-2 test problem of [19] by repeatedly reporting the same event. It is a striking demonstration of the care that must be exercised when restarting the integration at an event.

3 Convergence

There are few results that assert convergence of numerical solutions of IVPs with event functions. The most general is Mannshardt's investigation [12] of one-step methods. In his approach to the task a one-step method is used to form approximations y_n to $y_L(t_n)$ at mesh points $t_n = a + nh$. The integration continues until a change in the sign of the event function indicates that an event has occurred in the span of the step from t_n . The event is located by finding a step size for which a step from (t_n, y_n) to $t_{n+\sigma} = t_n + \sigma h$ results in $y_{n+\sigma}$ for which $g(t_{n+\sigma}, y_{n+\sigma}) \approx 0$. This completes the first part of the integration. In the second part, the function f_L is changed to f_R and a step of size $(1 - \sigma)h$ is taken from $(t_{n+\sigma}, y_{n+\sigma})$ to (t_{n+1}, y_{n+1}) . The integration then continues with constant step size h . With assumptions essentially the same as ours, he shows that if a Runge-Kutta method of order p is used for the integration and the event is located accurately enough, the numerical solution is convergent of order p on the whole interval.

At the time Mannshardt investigated the task, codes based on one-step methods produced answers only at mesh points. Indeed, this is why most codes with an event location capability were (and still are) based on Adams formulas and backward differentiation formulas (BDFs). These methods represent the solution by polynomials in the span of a step and are properly viewed as producing a piecewise polynomial approximation $y^{**}(t)$ on the whole interval of integration rather than merely approximations at mesh points. In Mannshardt's iterative procedure for locating an event, he actually takes steps of size σh with the one-step method to compute $y_{n+\sigma}$. With a polynomial $y^{**}(t)$ that is accurate throughout $[t_n, t_n + h]$, an approximation to $y_{n+\sigma}$ is obtained by merely evaluating a polynomial. This is so much cheaper that it becomes practical to locate an event about as accurately as possible in the precision available. Recent codes based on one-step methods use continuous extensions to obtain the same kind of approximation that is natural with Adams formulas and BDFs.

We analyze the effects of events on the numerical solution of IVPs in a way applicable to most of the solvers in wide use. To this end we assume that when given a tolerance τ , the solver produces a numerical solution $y^{**}(t)$ that is piecewise smooth and uniformly accurate of order $r(\tau)$. We assume of the rate of convergence $r(\tau)$ only that it is monotone and $r(\tau) \rightarrow 0$ as $\tau \rightarrow 0$. The text [16] discusses a number of convergence results for various methods and controls on the local error. For example, methods implemented with an error per unit step (EPUS) control typically have $r(\tau) = \tau$, and the same is true with error per step and local extrapolation (XEPS). Such codes are common. A method of fixed order p based on error per step (EPS) would have $r(\tau) = \tau^{(p+1)/p}$. The matter is blurred with popular codes of this kind that also vary the order p , but our convergence assumption is sufficiently general that it is an acceptable description of the behavior of most of the popular IVP solvers. We further assume that events are located about as accurately as possible in the precision available. We shall prove that whatever the rate of convergence $r(\tau)$, the same rate is observed in the presence of well-separated, simple events. Event location

has a remarkably small effect on the solution of IVPs: Computation of an event, even as accurately as possible, is relatively inexpensive when a continuous extension is used. The only extra steps due to events are those that arise in restarting the equation after an event. Clearly this cost is necessary because the ODEs change at events. The last step size used before an event is often on-scale for the new integration and this reduces the cost of a restart. If there are few events, all modern implementations start efficiently enough that restarting is a small portion of the overall cost. However, if there are many events, it might be noticeably faster to use one-step methods because they start faster.

For a tolerance τ , suppose that an integrator returns $y_L^{**}(t)$ as the numerical solution of (1) with initial value $y(a) = A$ and that an event is found at t^{**} . Specifically, t^{**} is the first root of the equation

$$g(t^{**}, y_L^{**}(t^{**})) = 0 \quad (4)$$

It is assumed that this root is simple. Let $u(t)$ be the (mathematical) solution of (2) with initial value $u(t^{**}) = y_L^{**}(t^{**})$. Note that both the initial point and initial value of $u(t)$ are different from those of $y_R(t)$. Using the same tolerance and integrator, we compute $y_R^{**}(t)$ as an approximation to $u(t)$. Assuming that no events occur in the integration to b , the numerical solution $y^{**}(t)$ is defined to be $y_L^{**}(t)$ on $[a, t^{**}]$ and $y_R^{**}(t)$ on $[t^{**}, b]$. Our assumption about the integrator is that there is a constant C_1 for which

$$\|y_L(t) - y_L^{**}(t)\| \leq C_1 r(\tau) \quad (5)$$

for $a \leq t \leq t^{**}$ and a constant C_2 for which

$$\|u(t) - y_R^{**}(t)\| \leq C_2 r(\tau) \quad (6)$$

for $t^{**} \leq t \leq b$.

We shall prove that $y^{**}(t) = y(t) + O(r(\tau))$ on all of $[a, b]$. In the first part of the integration this is just our assumption (5) on the integrator, but what constitutes the first part exposes a complication. For the sake of definiteness, let us suppose that $t^{**} < t^*$. Then the first part of the integration is $a \leq t \leq t^{**}$. The remainder of the interval breaks naturally into two parts, $[t^{**}, t^*]$ and $[t^*, b]$, that have to be treated differently because $y(t)$ is $y_L(t)$ on the first and $y_R(t)$ on the second.

How well the location t^* is approximated by t^{**} is of both practical and theoretical importance. The point t^{**} is defined by the equation (4). We are assuming that the integrator yields an approximation

$$\begin{aligned} y_L^{**}(t^{**}) &= y_L(t^{**}) + O(r(\tau)) \\ &= y_L(t^*) + (t^{**} - t^*)y_L'(t^*) + O((t^{**} - t^*)^2) + O(r(\tau)) \end{aligned}$$

Substituting this into (4) and some further expansion leads to

$$0 = (t^{**} - t^*)g'(t^*, y_L(t^*)) + O(r(\tau)) + O((t^{**} - t^*)^2)$$

This equation and our assumption that t^* is a simple root imply that $t^{**} - t^*$ is $O(r(\tau))$. For definiteness we write this as

$$|t^{**} - t^*| \leq C_3 r(\tau) \quad (7)$$

On the interval $[t^{**}, t^*]$,

$$\begin{aligned} \|y(t) - y^{**}(t)\| &= \|y_L(t) - y_R^{**}(t)\| \\ &\leq \|y_L(t) - y_R(t)\| + \|y_R(t) - u(t)\| + \|u(t) - y_R^{**}(t)\| \end{aligned} \quad (8)$$

We begin with the first term on the right in (8). Let M be a bound on both $\|f_L(t, y)\|$ and $\|f_R(t, y)\|$ in a ball about $(t^*, y_L(t^*))$. By continuity and (7), $y_L(t)$ lies in this ball for $t^{**} \leq t \leq t^*$ for all sufficiently small τ . The solution $y_R(t)$ also passes through $(t^*, y_L(t^*))$, so it also lies in the ball for $t^{**} \leq t \leq t^*$ for all sufficiently small τ . Using the integrated form of (1) and (2) and the fact that the solutions have the same initial value at t^* , we obtain

$$\|y_L(t) - y_R(t)\| = \left\| \int_{t^*}^t [f_L(\zeta, y_L(\zeta)) - f_R(\zeta, y_R(\zeta))] d\zeta \right\|$$

The bound on the functions and (7) then imply that for $t^{**} \leq t \leq t^*$,

$$\|y_L(t) - y_R(t)\| \leq 2M|t - t^*| \leq 2MC_3 r(\tau) \quad (9)$$

Thus, the first term of (8) is $O(r(\tau))$ on the interval $[t^{**}, t^*]$. To show this of the second term, we use the fact that both $u(t)$ and $y_R(t)$ are solutions of the same equation (2) so that we can apply Gronwall's inequality to obtain

$$\|u(t) - y_R(t)\| \leq e^{\mathcal{L}(t-t^{**})} \|u(t^{**}) - y_R(t^{**})\|$$

for $t^{**} \leq t \leq b$. Here \mathcal{L} is a Lipschitz constant for $f_R(t, y)$. For $t = t^{**}$, inequality (9) states that

$$\|u(t^{**}) - y_R(t^{**})\| = \|y_L(t^{**}) - y_R(t^{**})\| \leq 2MC_3 r(\tau)$$

from which we deduce that

$$\|u(t) - y_R(t)\| \leq 2MC_3 e^{\mathcal{L}(t-t^{**})} r(\tau) \quad (10)$$

for $t^{**} \leq t \leq b$. Obviously $\|u(t) - y_R(t)\|$ is $O(r(\tau))$ for the subinterval $t^{**} \leq t \leq t^*$. The third term in (8) is $O(r(\tau))$ by the assumption (6), so the proof that $y^{**}(t) = y(t) + O(r(\tau))$ on $[t^{**}, t^*]$ is complete.

On the interval $[t^*, b]$,

$$\begin{aligned} \|y(t) - y^{**}(t)\| &= \|y_R(t) - y_R^{**}(t)\| \\ &\leq \|y_R(t) - u(t)\| + \|u(t) - y_R^{**}(t)\| \end{aligned}$$

The inequality (10) shows that the first term on the right is $O(r(\tau))$ on this interval and the same is true of the second because of our assumption (6) about

the integrator. Thus $y^{**}(t) = y(t) + O(r(\tau))$ on $[t^*, b]$ and the proof that this relationship holds on all of $[a, b]$ is complete. In the course of the proof we assumed that $t^{**} < t^*$, but the proof is essentially the same when $t^* < t^{**}$.

With some additional assumptions, Mannshardt [12] extends his convergence result to an asymptotic expression for the error. It is not difficult to do the same in the present circumstances. Instead of merely assuming that the integrator produces a solution $y_L^{**}(t)$ with an error that is $O(r(\tau))$ as in (5), let us assume now that

$$y_L^{**}(t) = y_L(t) + e_L(t)r(\tau) + h.o.t. \quad (11)$$

for a differentiable function $e_L(t)$. We argue that the error is proportional to $r(\tau)$ even in the presence of events. As in the convergence proof, we suppose that $t^{**} < t^*$. It is not difficult to sort out the leading term in the error on the interval $[t^{**}, t^*]$, but it is not proportional to $r(\tau)$, so we do not provide the details. Despite this, we shall see that the error *is* proportional for $t^* < t$. Accordingly, we shall prove that the error is proportional to $r(\tau)$ except in an interval of length $O(r(\tau))$ that contains t^* . When $t^{**} < t^*$, this interval is $[t^{**}, t^*]$ and otherwise, $[t^*, t^{**}]$.

We must sharpen the result (7) by using the asymptotic expression (11) instead of the bound (5). Proceeding as before we find that

$$0 = (t^{**} - t^*)g'(t^*, y_L(t^*)) + \frac{\partial g}{\partial y}(t^*, y_L(t^*)) e_L(t^*)r(\tau) + O((t^{**} - t^*)^2)$$

from which it follows that

$$t^{**} - t^* = C_3 r(\tau) + h.o.t. \quad (12)$$

with appropriate definition of the constant C_3 .

For $t^* \leq t \leq b$,

$$\begin{aligned} y^{**}(t) - y(t) &= y_R^{**}(t) - y_R(t) \\ &= [y_R^{**}(t) - u(t)] + [u(t) - y_R(t)] \end{aligned} \quad (13)$$

The first term has the desired behavior by assumption:

$$y_R^{**}(t) - u(t) = e_R(t)r(\tau) + h.o.t.$$

We use the equation of first variation

$$\Theta' = \frac{\partial f_R}{\partial y}(t, y_R(t)) \Theta, \quad \Theta(t^*) = I$$

to write the second term as

$$u(t) - y_R(t) = \Theta(t) [u(t^*) - y_R(t^*)] + h.o.t$$

and direct our attention to the term in brackets.

From (2) we have

$$\begin{aligned} u(t^*) &= u(t^{**}) + (t^* - t^{**})f_R(t^{**}, u(t^{**})) + h.o.t. \\ y_R(t^*) &= y_R(t^{**}) + (t^* - t^{**})f_R(t^{**}, y_R(t^{**})) + h.o.t. \end{aligned}$$

Because $t^* - t^{**}$ and $u(t^{**}) - y_R(t^{**})$ are both $O(r(\tau))$, this implies that

$$u(t^*) - y_R(t^*) = u(t^{**}) - y_R(t^{**}) + h.o.t.$$

Using the fact that $u(t^{**}) = y_L^{**}(t^{**})$, we then have

$$u(t^*) - y_R(t^*) = [y_L^{**}(t^{**}) - y_L(t^{**})] - [y_R(t^{**}) - y_L(t^{**})] + h.o.t.$$

The assumption (11), the differentiability of $e_L(t)$, and (12) show that the first term

$$y_L^{**}(t^{**}) - y_L(t^{**}) = e_L(t^*)r(\tau) + h.o.t.$$

has the desired behavior. From (2) we have

$$\begin{aligned} y_R(t^{**}) &= y_R(t^*) + (t^{**} - t^*)f_R(t^*, y_R(t^*)) + h.o.t. \\ y_L(t^{**}) &= y_L(t^*) + (t^{**} - t^*)f_L(t^*, y_L(t^*)) + h.o.t. \end{aligned}$$

Using the definition $y_R(t^*) = y_L(t^*)$ and the expression (12), we find that

$$y_R(t^{**}) - y_L(t^{**}) = C_4 r(\tau) + h.o.t.$$

where

$$C_4 = C_3 [f_R(t^*, y_L(t^*)) - f_L(t^*, y_L(t^*))]$$

Inserting all these results into (13) we find that

$$y^{**}(t) - y(t) = r(\tau) \{e_R(t) + \Theta(t) [e_L(t^*) + C_4]\} + h.o.t.$$

for $t^* \leq t \leq b$. This completes the proof that if the integrator produces an approximate solution with an error that is proportional to $r(\tau)$, then the same is true when events occur except in an interval of length $O(r(\tau))$ about each event.

4 Numerical Experiments

Our theoretical results apply to a wide variety of methods and implementations. This is illustrated by the MATLAB ODE Suite [17]. Because event location was a design objective for the Suite, all the methods implemented produce piecewise polynomial approximate solutions. These methods include explicit and implicit Runge-Kutta, Rosenbrock, Adams, and BDF. Events are located about as accurately as possible in the precision available. Because all the solvers of this Suite have exactly the same user interface, changing the method used in an experiment is accomplished by simply changing the name of the solver. Our

experiments with this wide range of methods support the theory developed in this article. However, because this Suite is so homogeneous, we have preferred to report here only results obtained with a wide variety of FORTRAN codes popular in general scientific computation.

First we note that all the FORTRAN codes we used are described by our theory because they all produce piecewise polynomial solutions that are used with a root finder to approximate event times as accurately as possible in the precision available. As to the root finder, one (DDRIV2) uses a variant of that described in [18], two (DRDEAM and DRDEBD) use the approach described in [20, 21], and the rest (as well as all the solvers of the MATLAB ODE Suite) use variants of the scheme described in [7]. The solvers differ greatly in detail, but the user interfaces are sufficiently similar that we could solve the IVPs of our experiments in the same way. All computations were done in IEEE double precision arithmetic.

We are not aware of a widely-available, general-purpose, explicit Runge-Kutta code in FORTRAN with event location capability. However, two programs for solving delay-differential equations (DDEs), DKLAG5 [13] and DKLAG6 [4], can be used to solve IVPs with event location. These programs are based on (4,5) and (5,6) explicit Runge-Kutta pairs due to Sarafyan and have continuous extensions. DRDEAM [20] is a variant of the well-known DEABM implementation of Adams-Bashforth-Moulton formulas. It supplements the standard scheme of recognizing the occurrence of an event by a change of sign with a second, more sophisticated scheme. DDRIV2 [9] (and its lower level subroutine DDRIV3) has options to use either Adams-Moulton formulas or BDFs and we experimented with both. LSODAR [8, 14] is a variant of the well-known LSODE code. In addition to event location, it is unusual in that it switches automatically between Adams-Moulton formulas and BDFs, depending on whether the IVP appears to be nonstiff or stiff. DRDEBD [21] is a variant of the well-known DEBBF implementation of the BDFs. Like DRDEAM, it has an exceptionally strong scheme for recognizing that an event has occurred. DDASRT is a variant of the well-known DASSL [1] program for solving differential-algebraic equations (DAEs). It is based on the BDFs. As with the DDE solvers, this DAE solver can be used to solve IVPs with event location.

Experiment 1

In this experiment our theoretical results are illustrated with the numerical solution of (3). We obtained quite similar results with all the codes cited, so report here results for just a few. A mixed error test was used with both relative and absolute tolerances equal to EPS. The accuracy required is indicated in the tables by $ITOL = -\log_{10}(\text{EPS})$. The maximum error overrun, ERO1, is the maximum ratio of the error observed to the error allowed by the tolerances. The cost of the integration is measured by the number, NFE1, of evaluations of the functions defining the ODEs. The maximum error in the computed event times is reported as ERT.

The most difficult aspect of designing the experiment was to show clearly

what might be expected of the solver when there is no event location. We did this by solving a collection of ten IVPs that approximates (3). Let $t_0 = 0$ and let t_1, t_2, \dots, t_9 be the times of the nine events as determined from the analytical solution of (3). On each $[t_j, t_{j+1}]$ we solved an IVP with the ODE of (3) for this interval and with initial value given by the analytical solution of (3) evaluated at t_j . Solving this collection of IVPs approximates well the cost of integrating (3) because the same ODEs are integrated over nearly the same subintervals with nearly the same starting values, but there is no event location. Solving the collection does include the cost of repeatedly restarting to integrate a different ODE. The maximum error overrun for solving all these IVPs is reported as ERO2 and the cost as NFE2.

It is observed in the tables that event location has almost no effect on the cost as measured by function evaluations. The discrepancy seen in Table 1 is only apparent: DKLAGE5 does 5 function evaluations at each of the 9 events because of bookkeeping associated with the solution of delay-differential equations. Accordingly, the cost of the integration itself is actually the same with and without event location.

The tables show that the error is controlled when events are present just as it is when no events are present. DKLAGE5 is a fixed order code for which the theory predicts that the error is proportional to the tolerance and this is seen in Table 1. (This is also true of the computations with DKLAGE6 which we do not report.) We have noted that the matter is blurred when the order (formula) is varied. All the other solvers do vary their order and LSODAR varies the formula, too, but it is seen that in all cases the behavior of the error is qualitatively the same whether or not events are present.

Experiment 2

Our theory can be used to prove a convergence result rather like Mannshardt's that we illustrate with DKLAGE5 and DKLAGE6. Though not ideal for the present purpose, both solvers use a fixed order Runge-Kutta pair and it is possible to make them use a constant step size. We integrated with constant step size, but like Mannshardt, when the code stepped over an event, we located the event, stepped to it, and restarted. The problem (3) was solved for a sequence of step sizes 2^{-m} and the maximum error measured. For a fifth order Runge-Kutta formula as in DKLAGE5, the theory predicts that the ratio of errors in successive integrations will be approximately 5, and for a sixth order formula as in DKLAGE6, approximately 6. The results given in Table 5 show this to be the case.

Experiment 3

Modern IVP solvers are so robust that it is often possible for users to ignore events, and many do. In some circumstances [5, 6] this is permissible, but as this experiment shows, even when a quality IVP solver reports success, it can be very expensive to ignore events and the answers can be unacceptable. Also,

this experiment furnishes examples of two difficulties discussed in §2, viz. events located in the wrong order and trouble restarting.

The quench front problem of [10] models a cooled liquid rising on a hot metal rod by $u_t = u_{xx} + g(u)$ for $0 \leq x \leq 1$ and $0 < t$. Here

$$g(u) = \begin{cases} -Au & \text{if } u \leq u_c, \\ 0 & \text{if } u_c < u \end{cases}$$

with $A = 200,000$ and $u_c = 1/2$. The boundary conditions are $u(0, t) = 0$ and $u_x(1, t) = 0$. The initial condition is

$$u(x, 0) = \begin{cases} 0 & \text{if } 0 \leq x \leq 0.1, \\ (x - 0.1)/0.15 & \text{if } 0.1 < x < 0.25, \\ 1 & \text{if } 0.25 \leq x \leq 1. \end{cases}$$

The method of lines [15] is used to approximate $u(x, t)$ at grid points x_i . Specifically, PDEONE [11] is used to approximate u_{xx} and the boundary conditions at grid points. This results in a system of ODEs for $u_i(t) \approx u(x_i, t)$.

Event functions $g_i(t, u_1, \dots, u_n) = u_i(t) - u_c$ are used to deal with the discontinuity in the definition of $g(u)$. Note that there are as many event functions as ODEs. Initially a switch is set for each component larger than u_c ; the switch signals the derivative subroutine to use $g(u) = -Au$ for each such component. When the time at which the corresponding u_i drops to u_c is found, the switch is reset to signal the derivative subroutine to use $g(u) = 0$ thereafter for this component. In this way, the integrator locates the events without encountering derivative discontinuities and restarts the integration easily. The situation is quite different if event location is not used: the discontinuities in the definition of $g(u)$ result in derivative discontinuities with strong effects on both the efficiency and reliability of the integrator.

Table 6 presents the cost of solving this problem with three of the BDF solvers using banded Jacobians and a local error tolerance of 10^{-6} . The results correspond to $n = 50$, but the same behavior was observed for other tolerances and other n . NFE1 is the number of function evaluations required when event location is used and NFE2, the number when the events are ignored. As the table shows, ignoring events is very expensive, but what is worse is that when events are ignored, the solution has the wrong shape and is negative at some points.

As discussed in §2, special precautions must be taken so that DDRIV3 starts properly following an event and to insure that events are located in the correct (in this case, successive) order. Without these precautions, sometimes DDRIV3 repeatedly reports the same event and other times, does not locate events in the correct order. In the latter situation, the final solution has the wrong shape.

References

- [1] K.E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Elsevier, New

York, 1989.

- [2] M.B. Carver, Efficient integration over discontinuities in ordinary differential equations, *Math. Comp. Simulation*, 20 (1978) 190–196.
- [3] E.A. Coddington and N. Levinson, *Theory of Ordinary Differential Equations*, McGraw-Hill, New York, 1955.
- [4] S.P. Corwin, D. Sarafyan, and S. Thompson, DKL6G6: a code based on continuously imbedded sixth order Runge–Kutta methods for the solution of state dependent functional differential equations, *Appl. Num. Math.*, 24, (1997) 319–333.
- [5] E. Eich-Soellner and C. Führer, *Numerical Methods in Multibody Dynamics*, Teubner, Stuttgart, 1998.
- [6] W.H. Enright, K.R. Jackson, S.P. Nørsett, and P.G. Thomsen, Effective solution of discontinuous IVPs using a Runge-Kutta formula pair with interpolants, *Appl. Math. Comp.*, 27 (1988) 313–335.
- [7] K.L. Hiebert and L. F. Shampine, Implicitly Defined Output Points for Solutions of ODEs, Report SAND80–0180, Sandia National Laboratories, Albuquerque, NM, 1980.
- [8] A.C. Hindmarsh ODEPACK: A Systematized Collection of ODE Solvers, pp. 55–64 in *Scientific Computing*, R. S. Stepleman *et al.* (eds.), North-Holland, Amsterdam, 1983.
- [9] D. Kahaner, C. Moler and S. Nash, *Numerical Methods and Software*, Prentice–Hall, Englewood Cliffs, NJ, 1989.
- [10] H.T. Laquer and B. Wendroff, Bounds for the Model Quench Front, *SIAM J. Numer. Anal.*, 18 (1981) 225–241.
- [11] N.K. Madsen and R.F. Sincovec, Software for Nonlinear Partial Differential Equations, *ACM Trans. Math. Soft.*, 1 (1975) 232–260.
- [12] R. Mannshardt, One-step methods of any order for ordinary differential equations with discontinuous right-hand sides, *Numer. Math.*, 31 (1978) 131–152.
- [13] K.W. Neves and S. Thompson, Software for the Numerical Solution of Systems of Functional Differential Equations With State Dependent Delays, *J. Appl. Num. Math.*, 9 (1992), 385–401.
- [14] L.R. Petzold, Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations, *SIAM J. Sci. Stat. Comput.*, 4 (1983) 136–148.
- [15] W.E. Schiesser, *The Numerical Method of Lines*, Academic Press, New York, 1991.

- [16] L.F. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, New York, 1994.
- [17] L.F. Shampine and M.W. Reichelt, The MATLAB ODE Suite, SIAM J. Sci. Comput., 18 (1997) 1–22.
- [18] L.F. Shampine and H. A. Watts, ZEROIN, A Root–Solving Routine, Report SC–TM–70–631, Sandia National Laboratories, Albuquerque, NM, 1970.
- [19] S. Thompson, A Collection of Problems for Testing Rootfinding ODE Solvers, Report ORNL–9912, Oak Ridge National Laboratory, Oak Ridge, TN, 1987.
- [20] H.A. Watts, RDEAM – An Adams ODE Code With Root Solving Capability, Report SAND85–1595, Sandia National Laboratories, Albuquerque, NM, 1985.
- [21] H.A. Watts, Backward Differentiation Formulae Revisited: Improvements in DEBF and a New Root Solving Code RDEBD, Report SAND86–2676, Sandia National Laboratories, Albuquerque, NM, 1986.

ITOL	NFE1	NFE2	ERO1	ERO2	ERT
3	215	170	0.577E+00	0.459E-02	0.115E-02
4	239	194	0.105E-01	0.197E-01	0.211E-05
5	329	284	0.863E-02	0.266E-01	0.173E-06
6	413	368	0.111E-01	0.307E-01	0.223E-07
7	587	542	0.114E-01	0.351E-01	0.228E-08
8	857	812	0.104E-01	0.380E-01	0.209E-09
9	1319	1274	0.129E-01	0.368E-01	0.258E-10
10	1991	1946	0.126E-01	0.372E-01	0.252E-11
11	3101	3056	0.171E-01	0.370E-01	0.332E-12

Table 1: DKLAG5: Runge–Kutta DDE solver.

ITOL	NFE1	NFE2	ERO1	ERO2	ERT
3	156	138	0.209E+00	0.593E+00	0.557E-03
4	216	196	0.142E+00	0.463E+00	0.285E-04
5	253	231	0.143E+01	0.270E+01	0.359E-04
6	321	291	0.956E+00	0.701E+00	0.160E-05
7	379	308	0.115E+01	0.395E+01	0.346E-06
8	384	405	0.505E+01	0.288E+01	0.102E-06
9	466	487	0.316E+01	0.158E+01	0.644E-08
10	514	514	0.299E+01	0.161E+01	0.675E-09
11	674	626	0.157E+01	0.305E+01	0.290E-10

Table 2: DRDEAM: Adams–Bashforth–Moulton solver.

ITOL	NFE1	NFE2	ERO1	ERO2	ERT
3	335	365	0.174E+02	0.741E+01	0.265E-01
4	486	496	0.300E+02	0.700E+01	0.451E-02
5	649	670	0.190E+01	0.214E+02	0.380E-04
6	793	802	0.481E+02	0.391E+02	0.721E-04
7	916	904	0.611E+02	0.309E+02	0.917E-05
8	1132	1125	0.239E+02	0.438E+02	0.478E-06
9	1335	1349	0.205E+02	0.608E+02	0.307E-07
10	1638	1576	0.239E+02	0.341E+02	0.358E-08
11	2028	2023	0.303E+02	0.631E+02	0.454E-09

Table 3: DDASRT: BDF DAE solver.

ITOL	NFE1	NFE2	ERO1	ERO2	ERT
3	138	130	0.790E+01	0.129E+02	0.119E-01
4	168	168	0.471E+01	0.209E+02	0.707E-03
5	216	216	0.401E+01	0.236E+02	0.602E-04
6	296	296	0.359E+01	0.193E+02	0.538E-05
7	384	384	0.756E+01	0.209E+02	0.151E-05
8	476	476	0.143E+02	0.320E+02	0.286E-06
9	594	594	0.278E+02	0.186E+02	0.416E-07
10	624	624	0.311E+02	0.276E+02	0.466E-08
11	812	812	0.328E+02	0.242E+02	0.492E-09

Table 4: LSODAR: Adams-Moulton and BDF solver.

m	DKLAG5	DKLAG6
2	4.5	5.5
3	4.8	5.7
4	4.9	5.8
5	4.9	5.9

Table 5: Estimated order with constant step size integration.

Solver	NFE1	NFE2
DDRIV3	1493	11439
DDASRT	2927	13380
DRDEBD	1687	15311

Table 6: Cost of ignoring events in quench model problem.