Mathematics and Computers in Simulation XX (1978) 190-196

EFFICIENT INTEGRATION OVER DISCONTINUITIES IN ORDINARY DIFFERENTIAL EQUATION SIMULATIONS

M.B. Carver

Atomic Energy of Canada Limited
Chalk River Nuclear Laboratories
Chalk River, Ontario, Canada

Abstract

This paper introduces a new method of detecting and handling discontinuities in arbitrary functions which form part of an ordinary differential equation set. The method has been implemented in conjunction with the Gear [6] integration algorithm for stiff equation sets, published by Hindmarsh [9], but the philosophy applies to any predictor corrector integration algorithm with Nordsieck [10] step size control.

1. INTRODUCTION

1.1 Discontinuities

Many dynamic systems may be simulated by ordinary differential equations written as a first order set:

$$\frac{dy_i}{dt} = f_i(t,y_1,y_2,\ldots,y_n) \quad i=1,n \qquad (1)$$

$$y_i(0) = y_{0i}$$

Such a set is most efficiently integrated on a digital computer using an error controlled variable step algorithm such as Runge-Kutta-Merson, Bulirsch-Stoer, Adams, Gear, etc.

In applied simulations, however, the system of equations frequently contains discontinuities in the form of switches, which are thrown when certain conditions are fulfilled.

Equation set (1) then becomes

$$\frac{dy_i}{dt} = f_{ij}(t,y_1,y_2,\ldots,y_n) \quad \begin{matrix} i=1,n \\ j=1,m \end{matrix} \qquad (2)$$

where the m states of the equation system are determined by a set of arbitrary algebraic discontinuity functions $\phi_k(t,y_1,y_2,\ldots,y_n)$, a switch or change of state occurring at a critical point defined by one of the ϕ_k passing through zero.

As the step size chosen by the integration algorithm is controlled by the current estimate of truncation error, a well-written algorithm will detect the discontinuity and adjust its step size sufficiently to overcome the discontinuity. However, in all cases, unless special measures are taken to detect the discontinuity in advance, before it affects the smooth operation of the integration, the discontinuity will cause a loss of time and accuracy arising in three possible ways.

(1) The algorithm will hunt around the discontinuity, using a large number of unsuccessful function evaluations before a step size is found which crosses the discontinuity with acceptable error.

(2) The actual time at which the discontinuity occurs is critical for accuracy, but will probably not be used as an integration point because of the above hunting.

(3) A very short-lived state switching on and then off, in an otherwise well-behaved system can be missed out entirely, if the current step includes its entire life span.

Routines for the detection and handling of discontinuities have been built into Runge-Kutta algorithms with Merson step control by O'Regan [11], and Kay, et al. [7,5]. These involve an interpolation of the discontinuity function within the integration step which spans the discontinuity, but the fact that the interpolation formula is not related to the integration algorithm, can cause problems in accuracy. Cellier [4] has tackled the discontinuity problem in a combined discrete event/continuous process simulator. Here events analogous to discontinuities are triggered by critical relationships between state variables. Cellier presents an interpolative iteration scheme to determine event occurrence times during integration by fifth- and eight-order Runge-Kutta algorithms.

Halin [8], using Lie series methods, expands the functions f_i as power series for integration, and points out that the discontinuity functions can also be expanded, thus discontinuities can be located much more accurately by using exactly the same method that the integration algorithm employs. He also points out that this cannot be done with classical multi-step methods, such as Adams or Hammings predictor correctors because of the difficulty in changing step size and the necessity of restarting.

However, the Nordsieck step control algorithm used in the Gear algorithm has largely overcome these problems. Here functions g of scaled derivatives are stored up to the current order q such that the prediction of y at time t+h is given simply by

$$y_{t+h} = y_t + \sum_{i=1}^{q} g_i(y^{(i)}) \qquad (3)$$

and at any other step size h_1 by

$$y_{t+h_1} = y_t + \sum_{i=1}^{q} g_i(y^{(i)})(h_1/h)^i \qquad (4)$$

It is shown in this paper that if the discontinuity functions, instead of being treated algebraically, are regarded merely as additional differential equations, equation 4 provides a natural and accurate prediction of discontinuities, which can be used to control the integration algorithm and ensure that an integration point is taken at the discontinuity without hunting.

The method is outlined and example of discontinuous systems investigated. Results are shown using Runge-Kutta, standard Hindmarsh-Gear, and Hindmarsh-Gear with discontinuity detection by the new method. The new method can give over 90% reduction in the number of function evaluations in the neighbourhood of a discontinuity. In large systems of equations such as those described in Baudouin and Carver[1], the time saving is impressive as the introduction of additional equations for the small number of discontinuity functions causes negligible penalty in matrix manipulations.

2. THE DISCONTINUITY FUNCTIONS

Generally in the representation of a physical system by a simulation model, the discontinuities occur when certain variables pass through known critical points, for example a safety valve can activate when pressure reaches a given level, thus changing the system. The condition at which this occurs are readily defined, but the exact time at which this happens depends on the evolution of the equations being integrated, and is not known in advance.

Thus equations (2) must be governed by a statement such as

IF $\alpha(y_1) \leq \beta(y_2)$ j=1, ELSE j=2

where α and β are arbitrary functions. Obviously the discontinuity function here is

$$\phi(t,y) = \alpha(y_1) - \beta(y_2) \qquad (5)$$

This can be turned into an additional differential equation.

$$\frac{d\phi}{dt} = \frac{dy_{n+1}}{dt} = \alpha'(y_1) - \beta'(y_2), \qquad (6)$$

$$y_{n+1,0} = \alpha(y_{10}) - \beta(y_{20})$$

For a number of discontinuity functions, the logic of defining the functions ϕ_k and choosing the state j can be quite involved, depending on whether the functions affect the equations in an independent or interdependent manner. (Example 3 shows a system governed by interdependent discontinuity functions.) However, this definition of the ϕ_k, and choice of state j, must be done whatever method is used, the only difference being that the new method uses equations 6 to define ϕ, rather than equation 5, and this permits the integration to proceed efficiently.

This, of course, increases the size of the equation set, but if n>>m as is usually so in simulation applications, the penalty is small. Furthermore, discontinuity functions $\phi_1,\phi_2,\ldots\phi_k$ can often be combined into one function $\phi = \phi_1,\phi_2,\ldots\phi_k$ having k zeros, but increasing the equation set by only one. Finally, if one is evaluating the system Jacobian to accelerate predictor corrector convergence, the new method prevents the wasted effort of unnecessarily evaluating several Jacobians in the neighbourhood of the discontinuity, and this can greatly reduce the number of necessary function evaluations, especially for large sets of equations.

3. DETECTION OF THE CRITICAL POINTS

The point at which any ϕ passes through zero is determined by applying equations 4 in the neighborhood of ϕ_t, i.e. to find h_1 for which

$$\phi_{t+h_1} = \phi_t + \sum_{i=1}^{q} g_i(y^{(i)})(h_1/h)^i = 0 \qquad (7)$$

if ϕ is predicted to pass through zero during the current step size h. For most cases, a test for negative sign in the product

$$s_1 = \phi_t * \phi_{t+h} \qquad (8)$$

will be a sufficient prediction criterion. To cover a short-lived switch in which ϕ passes through zero in both directions within h, the product

$$s_2 = \phi'_t * \phi'_{t+h} \qquad (9)$$

must also be examined. ϕ_{t+h} and ϕ'_{t+h} are predicted values obtained from equations similar to (7). Thus the following strategy applies:

Condition	S1	S2	Critical points within h	Action (a)
(a)	>0	>0	0	Continue integration.
(b)	>0	<0	2	Reduce h and repeat to attain condition a or c.
(c)	<0	>0	1	Solve 7 for h_1.

4. CONTROL OF THE INTEGRATION ALGORITHM NEAR A DISCONTINUITY

Once h_1 has been determined, the integration algorithm is directed to integrate to time $t+h_1$ and then restart at this time. Hindmarsh provides facilities for doing these latter two operations in reference (9).

It is necessary to check the values s_1 and s_2 for every ϕ following completion of each integration step. This can be programmed into the integration algorithm, but it is preferable to merely modify the algorithm to return control to the user at this point, allowing him to call a separately provided detection routine which performs the functions described above. This requires a four statement modification to the integration algorithm.

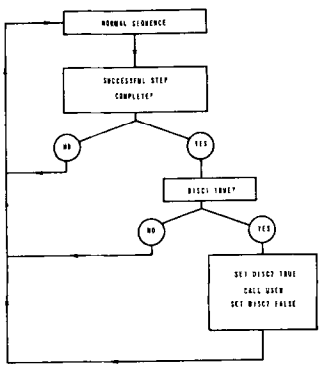
5. IMPLEMENTATION

The method as outlined above has been implemented in the FORSIM partial differential equation package[3], which uses the sparse matrix version of the Hindmarsh-Gear algorithm, GEARZ, as described[1]. This algorithm stores the N variables Y and the scaled derivatives $g_i(y^{(i)})$ up to the current order q, as required by equation 3, in an array Y(N,13), $q \leq 12$. The discontinuity detection routine DISCO is passed this array and an array of the indices ND_k, $ND_k \leq N$, of the discontinuity functions $y_k = \phi_k$. The return to the user routine is made conditional on a flag DISC1 which can be set true always, or for better efficiency, only for the neighbourhood in which a discontinuity is expected to occur. The logic coupling is shown in Figure 1, the FORTRAN code and documentation are available[3], but not included here, as interaction naturally depends on the particular version of integration algorithm being used.

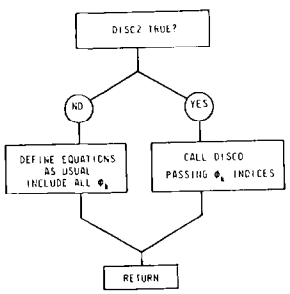
DISCO may handle more than one discontinuity function. Should two critical points be scanned by a proposed time step, each will be detected and handled in turn.

Several options are available in GEARZ for the evaluation and use of the Jacobian matrix. In the examples following, the functional iteration method is used to avoid introducing any confusion concerning function evaluations needed for integration, and those needed for Jacobian assessment. This and the Adams' predictor-corrector formulae option were used for all numbers quoted, but all other options were tested, and the new method found superior in each case, saving a large number of unnecessary Jacobian evaluations.

(a) Integration Algorithm



(b) USER: Routine to Define Equations(2)



(b) DISCO: Discontinuity Detection and Handling Routine

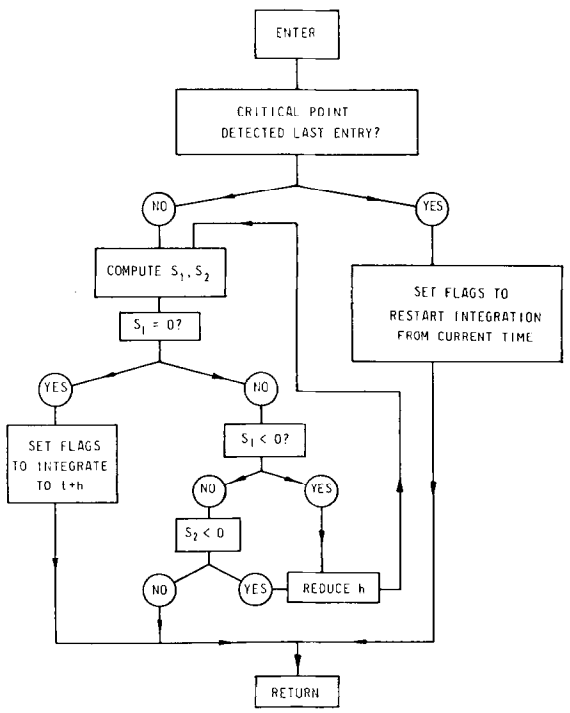


Fig. 1: Logic Structure for Discontinuity Detection

6. EXAMPLES

6.1 Single ODE, Single Discontinuity Function, Two States

Consider the differential equation

$$\frac{dy}{dt} = F_j(y) \quad F_j = y^2, \ \phi(t) > 0 \qquad (10)$$
$$F_j = 0, \ \phi(t) \leq 0$$
$$y(0) = y_0$$
$$0 \leq t \leq t_1$$

where the discontinuity function is given by $\phi = \sin(2\pi ft)$. If (10) is integrated for an integral number n cycles of ϕ, the result can be obtained from the solution of the related equation

$$\frac{dz}{dt} = z^2, \ 0 \leq t \leq t_2, \ t_2 = 0.5\ t_1 \qquad (11)$$

which is

$$z(t) = \frac{1}{1/z_0 - t} \qquad (12)$$

Table 1 shows the performance of the new method for three different interruption frequencies f, given an initial value $y_0 = z_0 = 0.1$, and taking $t_2 = 9.5$ which prevents (12) from going infinite, giving the solution $y(t_1) = z(t_2) = 2.00$. These and the following results were done with a requested local truncation error tolerance of 10^{-5}.

Table 1: Algorithm Performance for Example 1

Number of Discontinuity Function Cycles	0	2	4
	Number of Function Evaluations		
Runge-Kutta-Romberg	1,491	4,780	11,231
Standard Hindmarsh-Gear	150	359	421*
Standard Hindmarsh-Gear (Step Size Limited)	150	400	586
Modified Hindmarsh-Gear	150	293	485

*Indicates grossly inaccurate results due to the failure to detect one or more discontinuities.

The Runge-Kutta algorithm, a fourth-order routine with Romberg step control is obviously unsuited for this type of application. The standard Hindmarsh-Gear routine appears to perform quite well, but unless a step size limit is imposed, $h \leq 0.1/f$, the discontinuities can be missed altogether.

This problem imposes a mild discontinuity, so the savings realized by the new method are modest. However, as shown in Figure 2, the results clearly illustrate the stable behaviour of the new method as opposed to the hunting of the standard method. For severe discontinuities, this hunting becomes extreme and can lead to failure due to numerical overflow.

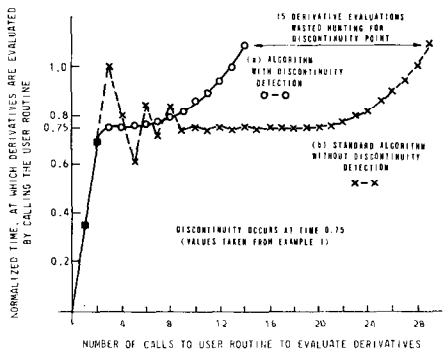


Fig. 2: Algorithm Behaviour at a Discontinuity

6.2 Single ODE, Two Discontinuity Functions, Three States

The equation

$$\frac{dy}{dt} = -A_j y + \sin(wt) \qquad (13)$$

where A = (1.0,0.5,0.2), w=1, $y(\pi/4)=0$, and

j=1 $|y| < 0.5$
j=2 $y \geq 0.5$
j=3 $y \leq -0.5$

can be described by two discontinuity functions

$$\phi_1 = y-.5, \ \phi_2 = -y-.5$$

Results for integrating $\pi/4 \leq t \leq 12.5$, which spans seven state changes, are shown in Table 2. Note that although the equation set to be integrated has been increased in size from one to three, the time required is still reduced.

Table 2: Algorithm Performance for Example 2

	Runge Kutta	Standard Hindmarsh Gear	Modified Hindmarsh Gear
Function Evaluations	11,630	654	387
CP Time	.438	.212	.203

CP Time is that using Control Data FORTRAN 4.6 on NOS/BE Operating System, Cyber 175 Computer

6.3 Two ODE's, Four Discontinuity Conditions, Three States

Consider the rectifier circuit in Figure 3. It contains 2 diodes D, 2 AC sources U, 3 resistors R and 3 inductors L.

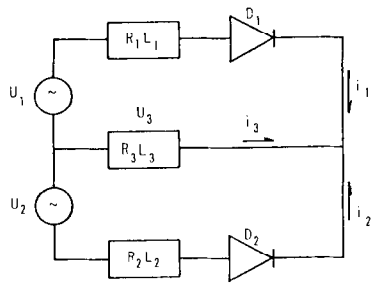


Figure 3: Circuit Example

Direct current and voltage are given by:

$$i_3 = i_1 + i_2 \qquad (14)$$

$$v_3 = R_3(i_1+i_2) + L_3(i_1' + i_2') \qquad (15)$$

i_1 and i_2 depend on whether D_1, D_2 or both are conductive. Thus we have:

State 1: D_1 Only Conductive when

$(i_1 > 0$ or $v_1 > v_3)$ and $(i_2 = 0$ and $v_2 \leq v_3)$

$$i_1' = (v_1 - i_1a_1)/a_2 \qquad (16)$$

$$i_2' = i_2 = 0$$

State 2: D_2 Only Conductive when

$(i_2 > 0$ or $v_2 > v_3)$ and $(i_1 = 0$ and $v_1 \leq v_3)$

$$i_2' = (v_2 - i_2a_3)/a_4 \qquad (17)$$

$$i_1' = i_1 = 0$$

State 3: D_1 and D_2 Conductive when

$(i_1 > 0$ or $v_1 > v_3)$ and $(i_2 > 0$ or $v_2 > v_3)$

$$i_1' = a_5v_1 + a_6v_2 + a_7i_1 + a_8i_2 \qquad (18)$$

$$i_2' = a_9v_1 + a_{10}v_2 + a_{11}i_1 + a_{12}i_2$$

The detailed development of the equations is given by Halin[8]. We merely note here that given $R_1 = R_2 = 2$, $R_3 = 10$, $L_1 = L_2 = 0.04$, $L_3 = 0.2$ and $v_1 = -v_2 = -100 \sin 100\pi t$, the values a_i become (12, 0.24, 12, 0.24, 13.64, -11.64, -50., 0., -11.64, 13.64, 0., -50.0).

To apply the method one must recognize that the discontinuity conditions depend on i_1, i_2, v_1, v_2, v_3 and combine them into four discontinuity functions.

$$\phi_1 = i_1 \qquad \phi_1' = i_1'$$

$$\phi_2 = i_2 \qquad \phi_2' = i_2'$$

$$\phi_3 = v_1-v_3 \qquad \phi_3' = v_1'-v_3'$$

$$\phi_4 = v_2-v_3 \qquad \phi_4' = v_2'-v_3'$$

Results from this case are shown in Table 3.

Table 3: Function Evaluation Required for Example 3

Frequency CPS	25	50	100
Standard Hindmarsh-Gear	585	968	1,575
Modified Hindmarsh-Gear	237	419	804
Improvement	60%	57%	49%

6.4 Severe Short-Lived Discontinuity

Consider integrating a skewed constant amplitude saw tooth wave for n cycles, period τ where

$$\frac{dy}{dt} = C_j,\ C = (+a,-b),\ n = \mathrm{Int}(t/\tau) \qquad (19)$$

$(t-n\tau) \leq \tau-\varepsilon$, $j=1$, $\tau-\varepsilon<(t-n\tau) \leq \tau$, $j=2$

$\varepsilon = \tau a/(a+b)$

For small ε, the descent becomes very rapid and short lived compared to the ascent, so this function is impossible to integrate with standard methods unless the step size is limited to $h < \varepsilon$. For detecting this type of discontinuity it is imperative to check the product s_1 as well as s_2.

Table 4 shows results for τ=10, ε.5, a=1, b=19, t_{max}=25.

Table 4: Algorithm Performance for Example 4

	Function Evaluation	CP Time
Standard Hindmarsh-Gear	463	.16
Modified Hindmarsh-Gear	185	.12

The improvement in the new method is due to the efficient isolation of the discontinuities as in the other examples, but here the fact that step size need only be limited to $h<\tau$ also speeds integration.

6.5 Friction Example

The equation of motion of a sliding object, mass m subject to an applied force F_a and frictional resistance F_f is

$$m\ddot{x} = F_a - F_f \qquad (20)$$

Friction, F_f, is a discontinuous function for static and dynamic conditions such that

if $\dot{x}=0$ and $|F_a| \leq F_s$, $F_f = F_a$

if $\dot{x}=0$ and $F_a > F_s$

or $\dot{x}>0$, $F_f = F_1+F_2\dot{x}$

if $\dot{x}=0$ and $F_a < -F_s$

or $\dot{x}<0$, $F_f=-F_1+F_2\dot{x}$

The system was run for the parameters $x=\dot{x}(0)=0$, $m=0.64$, $F=0.83$, $F_1=0.75$, $F_2=0.28$,

$0 < t < 0.1$ $\quad F_a=0$

$0.1 \leq t < 0.5$ $\quad F_a=5t$

$0.5 \leq t < 1.0$ $\quad F_a=-t$

$1.0 \leq t < 1.5$ $\quad F_a=0$

Apart from the externally applied switches, the problem has internal sources of discontinuity when motion changes direction and when the applied force exceeds the static friction limit. Applying discontinuity detection to this problem gives the results shown in Table 5.

Table 5: Results for Friction Example

	Number of Function Evaluations
Standard Method	425
New Method	298

7. CONCLUSIONS

While the Runge-Kutta-Romberg algorithm is usually reliable, it is inefficient for handling discontinuities. A Merson step control might perform better, but the problem of attempting to interpolate discontinuities remains.

In comparison, the standard Hindmarsh-Gear algorithm does quite a good job of transcending a discontinuity, but in cases such as Example 1, short-lived discontinuities can be missed unless the user is sufficiently alert to impose a step size limit. In other cases, such as the study of the equations describing high-temperature creep reported in [12], the algorithm often actually fails at a discontinuity. Although it was this particular study that led to various attempts to handle discontinuities which culminated in the current method, the equations are unfortunately too complex to present here. The basic problem here was to handle extremely severe but short-lived discontinuities comparable to Example 4.

It is apparent that the mechanism for detecting discontinuities can pay off in accuracy, reliability and efficiency at the cost of minimal changes to the logic structure of the integration algorithm.

8. ACKNOWLEDGEMENTS

The author wishes to thank F.E. Cellier, H.J. Halin, and A.C. Hindmarsh for their constructive comments.

9. REFERENCES

(1) A.P. Baudoin, and M.B. Carver, Solution of Reactor Kinetics Problems Using Sparse Matrix Techniques in an ODE Integrator for Stiff Equations, Atomic Energy of Canada Limited report AECL-5177, also Advances in Computer Methods for Partial Differential Equations, R. Vichnevetsky, ed. pp. 377-381, AICA Press, Rutgers University, New Jersey, 1975.

(2) M.B. Carver, The FORSIM System for Automated Solution of Sets of Implicitly Coupled Partial Differential Equations, AAICA, No. 3, July 1975, pp. 195-202.

(3) M.B. Carver, D.G. Stewart, J.M. Blair, and W.N. Selander, the FORSIM VI Simulation Package for the Automated Solution of Arbitrarily Defined Partial and/or Ordinary Differential Equation Systems, Atomic Energy of Canada Report, AECL-5821, February 1978.

(4) F.E. Cellier, and D.F. Rufer, An Algorithm Suited for the Solution of Initial Value Problems in Engineering Applications, Proceedings, Simulation '75, ETH, Zurich, M. Hamza, Ed., Acta Press, Calgary, 1975, pp. 160-165.

(5) R.E. Crosbie, and J.L. Hay, Digital Techniques for the Simulation of Discontinuities, Proceedings, 1974 Summer Computer Simulation Conference, pp. 87-91, SCI Press, La Jolla, California.

(6) C.W. Gear, The Automatic Integration of Ordinary Differential Equations, Comm. ACM 14, No. 3, pp. 176-190, 1970.

(7) J.L. Hay, R.E. Crosbie, and R.I. Chaplin, Integration Routines for Systems with Discontinuities, The Computer Journal, 17, No. 3, pp. 275-278, 1974.

(8) H.J. Halin, Integration of Ordinary Differential Equations containing Discontinuities, Proceedings, 1976 Summer Computer Simulation Conference, SCI Press, La Jolla, California.

(9) A.C. Hindmarsh, GEAR: Ordinary Differential Equations Solver, Lawrence Livermore Laboratory report UCID-30001, Rev. 2, August 1972, Rev. 3, December 1974.

(10) A. Nordsieck, On the Numerical Integration of Ordinary Differential Equations, Math-Comp, 16, pp. 22-49, 1962.

(11) P.G. O'Regan, Step Size Adjustment at Discontinuities for Fourth-Order Runge-Kutta Methods, The Computer Journal, 13, No. 4, pp. 401-404, 1970.

(12) S.R. MacEwan, and V. Fidleris, Verification of a Model for In Reactor Creep Transients in Zirconium, Phil. Mag. 31, pp. 1149-1157, 1975.