Computers chem. Engng, Vol. 15, No. 10, pp. 701–713, 1991
Printed in Great Britain.

0098-1354/91 $3.00 + 0.00

ALGORITHMS FOR THE LOCATION OF DISCONTINUITIES IN DYNAMIC SIMULATION PROBLEMS

A. J. PRESTON and M. BERZINS
School of Computer Studies, University of Leeds, Leeds LS2 9JT, U.K.

(Received 21 May 1990; final revision received 17 June 1991; received for publication 11 July 1991)

Abstract—Dynamic simulation problems in the chemical engineering industry often involve computing the solution to large sparse stiff systems of index-one or -two differential-algebraic equations (Gear and Petzold, *Siam J. Numer. Anal.* **21,** 716–728, 1984) with frequent discontinuities in the solution. This paper is concerned with new algorithms that are needed to handle the location of these discontinuities accurately and efficiently. The algorithms are implemented using the SPRINT (Berzins *et al. Proc. 3rd Euro. Conf. for Maths in Industry*, 1988a) software.

1. INTRODUCTION

The chemical and production processes employed in the petroleum and chemical industries involve increasingly specialized machinery and complex control systems. The efficient and robust control of such processes requires the accurate prediction of the dynamic response to changing operation conditions. The mathematical modelling of these systems requires the solution to a large set of sparse, stiff differential-algebraic equations (DAEs) representing gas compositions, temperatures, pressures, flowrates, control signals and valve positions. Discontinuities arise from the opening and closing of valves in the networks and manifest themselves in either the system variables representing the valve positions, or in their derivatives.

A survey of alternative approaches for the handling of discontinuities in the simulation of physical systems is provided in Capstick (1987) and Chua (1982). The approaches are mostly interpolation based, or regard the discontinuity condition(s) as additional equations to be solved (Carver and MacEwan, 1978; Smith, 1985).

Discontinuities first have to be detected and then located. (Gear, 1980) has shown that if no information is provided as to the conditions in which discontinuities occur in a system of ODEs gross inefficiencies can result when using a linear multistep method. In a recent paper (Preston *et al.*, 1989), a table of results was presented illustrating the need for special handling of discontinuities for a typical dynamic simulation problem (DSP). The location of these discontinuities for DSPs is difficult to determine, since their position is unknown before the integration takes place.

Once the discontinuity has been located, integration may have to be restarted. This is equivalent to the initialization problem for DAEs, which a number of authors have recently addressed, e.g. Gupta *et al.* (1985). Pantelides (1988a) and Leimkuhler *et al.* (1991). For the DSPs considered in this paper, the integration is restarted after a discontinuity only when absolutely necessary. That is whenever parts of the network suddenly become (in)active, referred to in Stubbe *et al.* (1989) as a "type a" discontinuity. This strategy reflects the cost of the restarting process for multistep methods, since a first-order formula is used at the restart. Discontinuities in elements of the Jacobian matrix require no restarting phase, and this type of discontinuity has been classified as a "type c" discontinuity (Stubbe *et al.*, 1989). In "type b" discontinuities, a subsystem of equations representing the process giving rise to the disturbance are isolated, and integrated separately over the interval straddling the discontinuity. This localization occurs only in very large DSPs with valves appearing in each of the network modules. In this paper only types a and c will be considered.

On restarting, it may be permissible to suspend error control in the solution, allowing reasonably large timesteps to develop quickly. This is the approach used in Chua and Dew (1984) for the simulation of a gas transmission network. This strategy is used whenever the integration error in successive steps has decreased in magnitude, and may be a viable proposition to increase the efficiency of the integration. Implicit Runge–Kutta methods (Ralston and Rabinowitz, 1978) have an inherent advantage over multistep methods since they can restart at high order and can also be used to generate accurate starting values for high-order BDF methods (Brenan, 1983). The potential benefits of these methods for DSPs require further investigation.

Finally, the integration should move away from the discontinuity as quickly as possible, by selecting an

appropriate stepsize and order. The efficient selection of the order of the BDF method to be used when integrating through smooth regions of the solution components has already been examined (Berzins *et al.*, 1988).

The idea of this paper is to use a case-study approach to investigate how to improve the numerical solution of a particular class of DSP problems arising in industry. The paper will thus describe how to accurately and efficiently determine the times at which discontinuities in these DSPs occur. This will be achieved by approximating a continuous valve signal before (or across) the discontinuity with a polynomial, and extrapolating forward (or interpolating) to find the time at which the discontinuity is expected to occur. Integration will be taken in controlled steps until the discontinuity is located sufficiently accurately.

Although the measures adopted here are specific to a particular class of industrial problems, they can also be applied to other problems. The main restrictions on the methods described in this paper are that it must be possible to identify the discontinuity by a switch function and that it is possible to use polynomial interpolation or extrapolation possibly coupled with either a phase plane approach or analytic information about a particular component equation.

2. A MODEL DYNAMIC SIMULATION PROBLEM

To highlight the main features of the location problem, consider the model problem shown in Fig. 1. This model consists of a compressor with a *suction-throttle* valve to control discharge pressure. A simplified mathematical model of this system as supplied by Shell Research Ltd is given by the following set of seven nondimensional equations. In these equations y_1 is the suction-throttle valve position, y_2 is the controller output signal (C_{out}), y_3 is the compressor outlet pressure (P_{out}), y_4 is the compressor inlet pressure (P_D), y_5 is the outlet mass flow (m_{out}), y_6 is the mass in drum (M_D) and y_7 is the inlet mass flow (m_{in}). The driving function $y_5(t)$ models a typical sharp rise and fall in customer demand:

$$\dot{y}_1 = \begin{cases} \min\{(y_2 - y_1)/2, 0\} & \text{if } y_1 \geqslant 1, \\ \max\{(y_2 - y_1)/2, 0 & \text{if } y_1 \leqslant 0, \\ (y_2 - y_1)/2 & \text{if } 0 < y_1 < 1, \end{cases} \tag{1a}$$

$$\dot{y}_2 = -[\dot{y}_3 + (y_3 - PSET)/5]/15, \tag{1b}$$

$$y_3/y_4 = 3.35 - 0.075\, y_5 + 0.001\, y_5^2, \tag{1c}$$

$$y_4^2 = 49.58^2 - (y_7/1.2\, y_1)^2, \tag{1d}$$

$$y_6 = 20\, y_4, \tag{1e}$$

$$\dot{y}_6 = y_7 - y_5, \tag{1f}$$

$$y_5 = 15 + 5\tanh(t - 10) - 5[1 + \tanh(t - 15)], \quad t \in [0, 150]. \tag{1g}$$

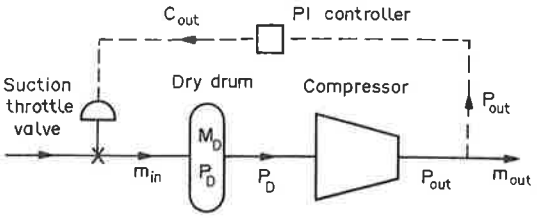

Fig. 1. A model dynamic simulation program.

The initial conditions are:

$$y_1(0) = 0.75, \quad y_2(0) = 0.75, \quad y_3(0) = PSET',$$
$$y_4(0) = 36.7, \quad y_5(0) = 10,$$
$$y_6(0) = 734, \quad y_7(0) = 10.$$

Note that the model (1a) caters for incorrect numerical values of the valve position y_1, outside the range of its physical interval [0, 1]. This is because the valve position is fixed (by setting $\dot{y}_1 = 0$) when the valve opens at (or "beyond") fully open (i.e. $\dot{y}_1 > 0$ and $y_1 = 1 + \epsilon$ for small $\epsilon > 0$), or when the valve closes at (or "beyond") fully closed (i.e. $\dot{y}_1 < 0$ and $y_1 = -\epsilon$ for $\epsilon > 0$ and small).

2.1. *The suction-throttle valve*

The suction-throttle valve acts as an input device to the system and controls the mass flow entering the network. The position of the suction-throttle valve shown in Fig. 1 will be either open ($y_1 = 1$) or closed ($y_1 = 0$), or else somewhere in between the two ($0 < y_1 < 1$), and discontinuities will be formed in $\dot{y}_1$ when the valve changes its state. This type of (derivative) discontinuity is commonly referred to as an *order* 2 discontinuity (Gear and Osterby, 1981). If the suction-throttle valve is closed, then the flow is internal within the network. The network is therefore always active and no restarting phase is necessary when the suction-throttle valve changes its state.

For later use it is convenient to define a new variable (y_8 = error signal), in the model problem (1a–g), defined as the difference between the outlet pressure coming from the controller and some constant predefined setpoint *PSET*:

$$y_8 = y_3 - PSET. \tag{2}$$

This gives an equation of the form:

$$\dot{y}_1 + a_1 y_1 = a_1 y_2(t), \quad \text{s.t. } 0 < y_1(t) < 1, \quad t \geqslant 0. \tag{3a}$$

where

$$y_2(t) = a_2[y_8(t) + a_3 y_8(A) + a_3 \int_A^t y_8(x)\, dx]. \tag{3b}$$

Equation (3a) will be referred to as *the discontinuity equation* for the *suction-throttle* valve. Here, A is a constant of integration, a_1 is a fixed constant representing the response speed of the controller, a_2 and a_3 relate to the controller gain and integral time constants respectively, all of which arise in engineering

control theory (Zeigler and Nichols, 1942). In particular, for problem (1a–g), $a_1 = 1/2$, $a_2 = -1/15$ and $a_3 = 1/5$ and in solving the problem equation (1a) is replaced by equation (3a) and the new equation (2) is used to define y_8 in the new system.

2.2. The anti-surge valve

In larger and more realistic networks, the compressor would be protected by an *anti-surge* valve, operating in a similar manner to the suction-throttle valve in that it can remain *partly* open. (Throughout the course of this paper, the anti-surge and suction-throttle valves will together be referred to as *control* valves.)

The operating characteristics of the anti-surge valve differs from the suction-throttle valve in that the recycle mass flow, error, control signal and valve position are all set to zero when the anti-surge valve closes. This effectively switches off part of the network, and necessitates an integration restart.

Unlike the suction-throttle case, the error signal for the anti-surge valve is defined as a function of the system variables representing the compressor inlet and outlet pressures resp., and *QIN*, representing the volume flowrate of the gas leaving the dry drum. *QIN* is defined as the ratio of the outlet mass flow from the dry drum, to the change in density of the gas leaving the drum; whereas in the suction-throttle case, *QIN* becomes a constant (*PSET*). For the DSPs under investigation, the anti-surge valves never become fully open. The error signal causing the anti-surge valve to open from fully-closed is defined by:

$$y_8 = \begin{cases} \min\{QIN - (0.28\, y_3/y_4)^{1/2}, 0\} & \text{if } y_1 \leqslant 0, \\ QIN - (0.28\, y_3/y_4)^{1/2} & \text{if } 0 < y_1 < 1. \end{cases} \tag{4a}$$

This is the *discontinuity equation* for the *anti-surge* valve, with the controller output signal y_2 defined in the differentiated form of equation (3b) as:

$$\dot{y}_2 = a_2(\dot{y}_8 + a_3 y_8). \tag{4b}$$

Note that y_2 is discontinuous since y_8 is discontinuous, but that the function:

$$QIN - (0.28\, y_3/y_4)^{1/2}, \tag{4c}$$

is continuous and zero at the point of discontinuity.

2.3. Check valves

Larger networks also contain *check valves* which act as a diode for the direction of mass flow. Check valves are operated by a spring-loaded mechanism, and are either *fully*-open or *fully*-closed. This means that jump-type [*order* 1 (Gear and Osterby 1981)] discontinuities are formed in the variables dependent on the check valve position whenever these valves switch between fully-open and fully-closed. In Gear and Osterby (1981) an algorithm is presented for the determination of the order of a discontinuity in an ODE system. For the DSPs considered in this paper though, the order of the discontinuity is known immediately upon detection, from the type of valve concerned, and its position before the discontinuity.

The *discontinuity equation* is defined only in terms of a linear inequality of a single variable in the network, the compressor outlet pressure (y_3), and a constant downstream pressure, *PDOWN* with constant resistance force *R*. There is no system variable explicitly representing the position of the check valve, but its state y_5 is adequately described from the following equation, which replaces equation (1g):

$$y_5 = \begin{cases} 0 & \text{if } y_3 \leqslant PDOWN \quad \text{(check valve closed)}, \\ y_3 - PDOWN)/R & \text{if } y_3 > PDOWN \quad \text{(check valve open)}. \end{cases} \tag{5}$$

This explains how a change in position of the check valve can lead to other parts of the network becoming (in)active. The *index* (Gear and Petzold, 1984) of the DAE system can therefore change, so special care needs to be taken with the choice of integrator and application of a sparse matrix package. Indeed, practical experiments have shown that one particular network containing a check valve exhibits index 2 behaviour when the valve is open, and index 1 behaviour when the valve is closed. This was confirmed by an analysis of the equations. The system of equations defined in the model problem (1a–g) has index 1 throughout, however.

3. DISCONTINUITY DEFINITIONS

In standard simulation terminology all the discontinuities considered in this paper are state events in that they are triggered by changes in solution values at times unknown beforehand and are thus implicit in some sense. Authors commonly refer to **explicit** discontinuities, when the times at which the discontinuities occur are known *a priori* (Capstick, 1987; Pantelides, 1988b; Ellison, 1981). Of the state implicit discontinuities there are two important subtypes of interest. These are defined as *partly*-**implicit** and *fully*-**implicit**, respectively. The contrast between these two types is that in the fully-implicit case the solution value after the discontinuity is not known beforehand whereas in the partially-implicit case the value after the discontinuity is known beforehand.

3.1. Partly-implicit discontinuities

The *partly*-implicit case appears once a valve changes its state *from partly-open* to fully-open (or fully-closed, respectively). The derivative of the variable representing the valve position is known to be zero after the discontinuity. These are order 2 discontinuities (Gear and Osterby, 1981) because

the variable representing the valve position is a C^1 continuous function.

3.2. *Full-implicit discontinuities*

The *fully*-implicit case appears when a valve changes its state from *fully-open* (or equivalently from *fully-closed*). Check valves belong only to this class since they are operated by a spring-loaded mechanism, and are assumed to switch directly between fully-open and fully-closed in the mathematical model. The control valves can indeed change from *partly-open* or *fully-open* and so belong to either class, depending on their current state. This case includes both order 1 and 2 discontinuities (Gear and Osterby, 1981) since the variable representing the valve position can have either C^0 or C^1 continuity.

4. DISCONTINUITY DETECTION: THE SPRINT SOFTWARE (BERZINS *et al.*, 1989)

SPRINT (Berzins *et al.*, 1989) is a general-purpose software package for the numerical solution of time-dependent differential equations. The kernel of the software is a DAE integrator package based on backward differentiation formulae (BDFs). Although SPRINT (Berzins *et al.*, 1989) is widely used by mathematical modellers within Shell Research Ltd (Frost, 1987), it was not designed to cope with the particular difficulties associated with discontinuities arising from DSPs. The main components of the present software are:

(i) a "Step" routine to advance the solution over one timestep;
(ii) a nonlinear equations solver called at each timestep;
(iii) a problem definition routine, RESID, to evaluate the residual equations of the DAEs being solved; and
(iv) a "Monitor" routine to interrogate the solution and take evasive action if necessary.

At each step of the integration, a system of nonlinear equations has to be solved, with program control frequently passing to the RESID routine. Discontinuities are detected inside the RESID routine through a FLAG which terminates the current step whenever the valve position changes its state. (Values detected by the RESID routine lying beyond the discontinuity will be termed as *predicted* values throughout the course of this paper.) Program control then passes to the Monitor routine to select the appropriate stepsize needed to march up to the discontinuity.

It is possible to include regular checks for these discontinuities through the use of a *switch function* which changes sign when a valve opens or closes. Discontinuities are detected once a sign change occurs. It is then necessary to interpolate backwards on the switching function (or extrapolate forwards using corrected values from previous timesteps), to find an approximation for the value of t at which the discontinuity occurred. The switch function would intuitively be based on the valve position. Providing a switch function in this way is the classical approach taken to solving systems of ODEs with discontinuities, and is generally attributable to Carver (1978). This "reverse interpolation" approach is principally the same as that used in Smith (1985) and others.

5. EXISTING DISCONTINUITY ALGORITHMS

A description follows of two current discontinuity handling algorithms than can be used within the SPRINT (Berzins *et al.*, 1989) software. Firstly, the robust practical algorithm, MONITR (Frost, 1987), which has been used successfully on DSPs at Shell Research; and secondly, a general-purpose discontinuity algorithm, MONITD (Berzins and Furzeland, 1984), which has been applied to a wide variety of problems with discontinuities in time.

5.1. *The MONITR algorithm (Frost, 1987) for partly-implicit discontinuities*

In the approach used inside SPRINT (Berzins and Furzeland, 1984), partly-implicit discontinuities are located by performing a linear interpolation backwards on the valve position making use of the valve position before the discontinuity, the known state of the valve after the discontinuity, and the predicted valve position detected in the RESID routine. This is illustrated in Fig. 2.

Using a switching function $f(t, y_1) = y_1 - 1$, where y_1 represents the valve position, the change in position to fully-open can easily be detected by comparing the sign of $f[t_n, y_1(t_n)]$ with that of $f[t^p_{n+1}, y^p_1(t^p_{n+1})]$. From Fig. 2, the fact that triangles ABC and ADE are similar is used to refine the interval containing the discontinuity. The diagram shows that although the position of the valve is known to take value 1 after the discontinuity, a predicted value (according to the definition given earlier) close to 1 would mean slow convergence to the root of the discontinuity, since the difference in area between similar triangles ABC and ADE is small. In the existing approach, the interval containing the discontinuity is refined to a sufficient accuracy by repeating the similar triangles approach a fixed number of times ($ITMAX$), usually between 5 and 7

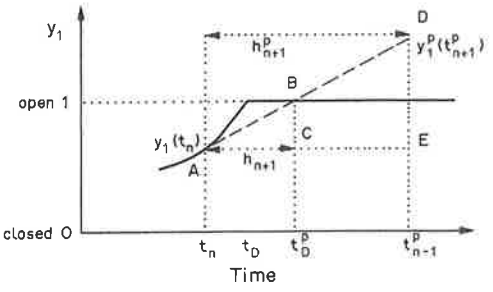

Fig. 2. The MONITR approach (Frost, 1987) for the location of a partly-implicit discontinuity.

times depending on the problem. In order to compare this algorithm with the new algorithm described later, it is necessary to change this termination criterion.

If a discontinuity location algorithm is applied a fixed number of times, this is no guarantee that the discontinuity has been located accurately enough. At fine tolerances, the stepsizes taken by the integrator are likely to be small in the region of the discontinuity, and applying MONITR (Frost, 1987) will probably result in a highly accurate location, but may be unnecessarily expensive with respect to integration over the whole time period of simulation. At coarse tolerances, the same number of root-finding iterations is likely to produce a poor approximation to the point of discontinuity, since the stepsizes taken by the integrator will be large. This may result in the integrator taking many small stepsizes after the discontinuity since the discontinuity was not located accurately enough, with the error test restricting the growth of stepsize.

An efficient algorithm will control the error bound in the discontinuity estimate. Bearing this in mind, an adaptive approach can be taken in which the termination criterion for convergence depends on the size of the interval spanning the discontinuity being less than some prescribed constant multiplied by the local error tolerance. Since the integrator controls the stepsize to a large extent at tighter tolerances, the discontinuity location algorithm is now terminated whenever the difference between the endpoints of the time interval straddling the discontinuity is less than some small positive number ϵ. the complete algorithm is listed below. The notation has been simplified for brevity: i.e. $y_n = y_1(t_n)$, $y^p_{n+1} = y^p_1(t^p_{n+1})$:

(i) given an interval known to contain the discontinuity: $[t_n, t^p_{n+1}]$, $y_n, y^p_{n+1}, h^p_{n+1}$, calculate an improved stepsize $h_{n+1} = h^p_{n+1}(1 - y^p_{n+1} - 1/y^p_{n+1} - y_n)$ and evaluate y_{n+1};
(ii) if $f(t_n, y_n) \cdot f(t_{n+1}, y_{n+1}) < 0$, then refine interval to $[t_n, t_{n+1}]$, reject the step and replace y^p_{n+1} by y_{n+1}, otherwise refine the interval to $[t_{n+1}, t^p_{n+1}]$, replace t_n by $t_n + h_n$ and calculate the predicted value of y_1 for the next step;
(iii) repeat steps (i) and (ii) above until $|t_n - t^p_{n+1}| < \epsilon$.

5.2. The MONITR algorithm (Frost, 1987) for fully-implicit discontinuities

The approach taken for fully-implicit discontinuities is interval bisection and is equivalent to the method used in Gear and Osterby (1981) for locating the root of an implicit discontinuity in a system of ODEs. The detailed algorithm is given below with the same simplified notation and modified termination criterion:

(i) given an interval known to contain the discontinuity: $[t_n, t^p_{n+1}]$, $y_n, y^p_{n+1}, h^p_{n+1}$, calculate an improved stepsize $h_{n+1} = h^p_{n+1}/2$ and evaluate y_{n+1} using the predictor in the DAE code;
(ii) if $0 < y_{n+1} < 1$ then refine the interval to $[t_n, t_{n+1}]$, reject the step, and replace y^p_{n+1} by y_{n+1}, otherwise refine the interval to $[t_{n+1}, t^p_{n+1}]$, replace t_n by $t_n + h_n$ and calculate the predicted value of y_1 for the next step;
(iii) repeat steps (i) and (ii) above until $|t_n - t^p_{n+1}| < \epsilon$.

Once the algorithm has terminated, the predicted location of the discontinuity t^n_D is such that $|t_D - t^p_D| < \epsilon$ where t_D is the actual point of discontinuity. Note that the location of the discontinuity depends on values of the solution components which may be in error, so the estimate of the discontinuity location error above is not exact.

In Shampine *et al* (1987) the difficulties in solving a system of initial value ODEs are emphasized, so too is determining the time(s) at which some algebraic function of the solution has a root. A Sturm sequence (Ralston and Rabinowitz, 1978) algorithm is used to determine the number of discontinuities existing in the interval of integration. Each root is then located accurately by bisecting a polynomial interpolant, or using a secant approach. This root finding approach at order one is very similar to the two MONITR (Frost, 1987) approaches described above.

5.3. The MONITD algorithm (Berzins and Furzeland, 1984)

The second existing discontinuity handling routine in SPRINT, MONITD (Berzins and Furzeland, 1984), is based on an algorithm similar to that in Carver and MacEwan (1987), and was designed only for the efficient and accurate handling of discontinuities arising in stiff ODEs and not for the discontinuities in the DAE systems arising from DSPs.

Discontinuities are located by applying a Newton method to a polynomial interpolating past values of the switch function. This method as found to be unsuitable when partly-implicit discontinuities were first detected, since the initial estimate was often too far away for the Newton method to converge. Moreover, in the fully-implicit case, when the interval of time embracing the discontinuity was small, the polynomial interpolating the valve position was found to oscillate before the discontinuity, contrary to the nature of the valve position. This is because the polynomial interpolant is prescribed to pass through a number of fixed solution values before the discontinuity, and a single value based on the estimated solution beyond the discontinuity without any guarantee that any monotonicity in the solution values is preserved.

6. CHOOSING AN APPROPRIATE SWITCH FUNCTION

In either of the two existing approaches mentioned above, rapid convergence to the root of the discontinuity depends on the switch function being sufficiently smooth across the discontinuity, since

interpolation is usually performed on this switch function to find its root (Pantelides, 1988b; Gear, 1980; Curtis, 1986) and hence the discontinuity.

Suppose a fully-implicit discontinuity in $\dot{y}_1$ has been observed to occur at some time t_D^p, bounded between integration times t_n and t_{n+1}. In the locality of the discontinuity, the valve position is constant prior to $t = t_D^p$ (e.g. $y_1 = 1$ for $t \leqslant t_D^p$) and linear immediately after $t = t_D^p$ (i.e. $\dot{y}_1^+ = \alpha \neq 0$). In this case the valve is fully open prior to the discontinuity, so attempting to estimate its position using linear interpolation gives the estimate of t_D^p as t_n.

Interpolation on two values beyond the discontinuity would involve using possibly unphysical estimates of the valve position. This could result in all kinds of errors (e.g. negative composition etc.), and does not seem to be a sensible approach, since the solution components can take unphysical values at the step beyond the discontinuity.

In an attempt to find a suitable switch function for a control valve closing from fully-open, Preston (1991) considers a variety of C^0 continuous switch functions. The conclusion of his investigations is that the best switch function is based on the valve position, its time derivative and using linear interpolation. This is equivalent to the bisection method in Gear and Osterby (1981) as described above.

7. IMPROVEMENTS TO DISCONTINUITY HANDLING

In an attempt to improve the performance of the algorithms mentioned above in terms of both efficiency and accuracy, consider again the discontinuity equations (3–5) that govern the position of these valves. The objective is either to interpolate on the left- or right-hand side of (3b), and solve the resulting equation (3a) for t. Rather than use the complicated right-hand side of equation (3b) as the function for interpolation, it is easier to use the variable y_2, which is directly available, and also has considerable practical significance, since y_2 is the signal from the output controller that causes the suction-throttle and anti-surge valves to change their state.

The solution of the linear first-order ODE (3a) for y_1 consists of a complementary function (Y_c) and a particular integral (Y_p):

$$Y_c = k \exp[-a_1(t - t_0)] \quad \text{for } 0 \leqslant t_0, \tag{6}$$

where t_0 is any fixed time between the initial time and the time of the discontinuity, and the constant k is determined through applying the initial conditions to the contribution from both the complementary function and the particular integral. Provided the discontinuity does not occur within two steps of the left-hand end of the interval of integration, the value of y_2 is available at the step just before the discontinuity and at the step previous to this. The predicted value of y_2 shortly after the discontinuity is certainly available. In principle, assuming that $y_2(t)$ and $y_8(t)$ can be described as polynomials in t, a quadratic (or linear) interpolant for y_2 can therefore be constructed from information obtained within the two (or single) accepted step(s) prior to the discontinuity, and the predicted step afterwards. If there is insufficient information available, the method of similar triangles can be used (Frost, 1987), as outlined earlier in Section 5.1. In the event of the discontinuity occuring within the first timestep, the bisection method (Gear and Osterby, 1981) can always be used (Section 5.2).

In the locality of the discontinuity, both the solution and the computed derivatives at the beginning and end of a step are available from SPRINT and can be used to construct an interpolation polynomial. Although a cubic or higher degree polynomial could be constructed, at course tolerances the order of the integration method has been observed to rarely go above two in the location of these discontinuities. For this reason, a quadratic interpolation polynomial has been used as this provides a more accurate approximation than a first-order polynomial. The derivative information can be used to construct a piecewise quadratic Bernstein polynomial (McAllister and Roulier, 1981) with continuous first derivative, that preserves monotonicity, convexity or concavity of the data between two integration steps. This form of polynomial has been used because the standard interpolants associated with stiff DAE integrators do not always preserve monotonicity, convexity of concavity.

The drawbacks to this approach are that the predicted values of the solution and derivative supplied by SPRINT (Berzins *et al.*, 1989) after the discontinuity are less accurate than the values prior to the discontinuity. On the other hand, the quadratic spline is not guaranteed to preserve monotonicity if *extrapolation* is performed *outside* the two integration steps immediately prior to the discontinuity. It is clearly desirable, however, that the interpolant preserves any local monotonicity, convexity or concavity.

7.1. *Partly-implicit discontinuities*

In an attempt to analyze the equations causing the suction-throttle valve to change its state, consider using the Bernstein quadratic spline approximation to $y_2(t)$. This transforms equation (3a) into:

$$\dot{y}_1 + a_1 y_1 = b_1 t_*^2 + b_2 t_* + b_3 + o(t_*^2), \tag{7}$$

where $t_* = t - t_0$ measures the time elapsed since t_0, the last successful step before the discontinuity was detected. b_1, b_2 and b_3 are constants computed over each of the two spline intervals (McAllister and Roulier, 1981), whose union defines the solution across an integration step. The particular integral is seen to be:

$$Y_p = b_1 t_*^2/a_1 + (b_2 a_1 - 2b_1)t_*/a_1^2 b_2 a_1 + 2b_1)/a_1^3. \tag{8}$$

The "initial condition" is given by the value of y_1 at some chosen timestep shortly before the discontinuity, t_0 say, and this can then be used to determine the contribution from the complementary function in equation (6). For the case of the suction-throttle valve changing from partly-open to fully-open, the constant k in equation (6) is determined from:

$$k = y_1(t_0) - b_3/a_1 + (b_2 a_1 - 2b_1)/a_1^3. \qquad (9)$$

In order to find the time at which the valve is fully-open, the fully-open condition $y_1(t) = 1$ is imposed. This reduces equation (7) to the following equation:

$$h(t) = -c_4 \exp(c_3 t_*) + 1 - c_2 t_*^2 - c_1 t_* - c_0 + o(t_*^3), \qquad (10)$$

where c_0, c_1, c_2, c_3 and c_4 are all available constants. [Using a cubic Hermite interpolant for $y_2(t)$ would result in an additional term in t_*^3 in equation (10).] Further, a unique solution is assumed to exist (at t_D^p) where $t_D^p \in (t_n, t_{n+1}^p)$. The end-points of this interval can then be used as starting estimates in the secant method, in order to estimate the root ($t = t_D$) of equation (10):

$$t_D^p = t_n - h(t_n)\frac{t_n - t_{n+1}^p}{h(t_n) - h(t_{n+1}^p)}. \qquad (11)$$

The implied constraint on equation (3a) becomes invalidated whenever values are used for interpolation that lie beyond the discontinuity. From a root-finding point of view, it therefore seems reasonable to either extrapolate on accepted values, or to interpolate with an "unphysical" value. Indeed, linear interpolation of y_1 using one accepted value and one "unphysical" value is precisely the original method (Frost, 1987), and has been shown to work reasonably effectively under most conditions. However, numerical experiments have shown better results for extrapolation using the two values prior to the discontinuity than the two points straddling the discontinuity. It should also be noted that the index two version of the model problem (1a–g), in which y_7 (rather than y_5) is specfied as a function of t, leads to order 1 discontinuities in the controller output signal y_2. This will, however, not affect the continuity of the interpolant, since the predicted value of the controller output signal beyond the discontinuity will be based on a smooth projection of the accepted values before the discontinuity.

Assuming the nonlinear equation can be solved for t, a step is taken to this value and predicted values y_1^p are calculated. If $y_1^p \leqslant 1$, the step is accepted and the above process is used to refine the coefficients in equation (10) which is then solved again. If $y_1^p > 1$, the step is rejected, since the discontinuity has been passed. The coefficients of the interpolation polynomial in (7) are updated and the upper bound for the interval containing the discontinuity is reduced. As before, the termination criterion depends on the predicted stepsize being less than some user-defined constant ϵ. The value of $\epsilon = 10^{-2}$ was used in the numerical experiments below.

Althought this method is specific to equation (10) it could be applied to any problem for which it is possible to integrate a subset of the full problem analytically once the polynomial approximation has been inserted.

7.2. Fully-implicit discontinuities

7.2.1. Control valves. The principles of the method of polynomial interpolation can be applied to both partly- and fully-implicit discontinuities occurring in the control valves. In the fully-implicit case though, equation (3a) holds only after the point of discontinuity. To the left of the discontinuity, the value of y_1 is constant (either 0 or 1) and thus $\dot{y}_1 = 0$. Hence, equation (3a) is given by either of the two cases:

$$\text{Control valve opening:} \quad y_2(t) = 0; \qquad (12a)$$

$$\text{Control valve closing:} \quad y_2(t) = 1; \qquad (12b)$$

In the case of the control valves, the discontinuity equation (3a) cannot be used unless predicted information for y_1 is considered, as the approach of Section 7.1 relies on the predicted value y_1^p being computed accurately, and is therefore rejected in favour of solving equation (12a, b).

The solution of the resulting polynomial equation can then be used either to advance the solution closer to the discontinuity and thereby improve the lower bound on t, or to improve the upper bound on t if the value of t turns out to lie beyond the discontinuity. In either case, at each solution of the above equation, the coefficients of the interpolant are updated. As described in the previous section, the process can be terminated when the stepsize to be taken, and hence the interval bounding the discontinuity is less than ϵ.

7.2.2. The phase-plane approach applied to check valves. The approach of Section 7.2.1. does not deal with the case of the check valves. In order to accommodate these a phase-plane approach is considered. It can be seen that equation (1a) is equvalent to:

$$\dot{y}_1 = \begin{cases} 0 & \text{if } (y_2 > y_1 \text{ and } y_1 \geqslant 1) \text{ or } (y_2 < y_1 \text{ and } y_1 \leqslant 0), \\ (y_2 - y_1)/2 & \text{otherwise.} \end{cases} \qquad (13)$$

Consider taking a (y_1, y_2) phase-plane approach, and calculating the time at which the solution trajectory crosses the lines $y_1 = y_2$, $y_1 = 1$ and $y_1 = 0$, respectively. Since these lines are known *a priori* (from the known state of the valve before the discontinuity), interpolants $y_1 = Q_1(t)$, $y_2 = Q_2(t)$ can be computed,

and the appropriate equation required can be selected from those listed below:

Fully-implicit discontinuity:

$$Q_1(t) = Q_2(t); \tag{14a}$$

Partly-implicit discontinuity (valve becoming fully-open):

$$Q_1(t) = 1; \tag{14b}$$

Partly-implicit discontinuity (valve becoming fully-closed):

$$Q_1(t) = 0. \tag{14c}$$

Of course, applying this method to partly-implicit discontinuities uses no information from the discontinuity equation (3a), and seems to gain little advantage over the MONITR algorithm of similar triangles (Frost, 1987). This approach applied to partly-implicit discontinuities has therefore been rejected.

If the phase-plane method is applied to the full-implicit case, practical experiments have shown that it is better to fix $X_1(t)$ and use the interpolant $Q_2(t) = 1$ if a control valve is about to close, or fix $Q_2(t) = 0$ if a control valve is about to open, rather than to apply equation (14a) directly. This is because the position of the valve is constant before the discontinuity, but may not take the exact value of either 0 or 1, owing to numerical error. Note that equation (14a) is then reduced to the equivalent expression (12a, b), so that this phase-plane method becomes identical to that described in Section 7.2 for fully-implicit discontinuities.

However, unlike the polynomial method, this phase-plane approach can be applied to the case of the check valve. The suggestion is to interpolate on y_3 and compute the time at which this component takes the value *PDOWN*. Numerical experiments have shown that the interpolant passing through the two steps before the discontinuity usually fails to preserve monotonicity up to the discontinuity. For this reason, the interpolant used is that passing through the two points straddling the discontinuity. However, in the opening of the anti-surge valve, caution must be taken when interpolating on $y_2(t)$, since convergence to the root may be poor. Unfortunately, there is no variable in the system explicitly describing the behaviour of the function given in equation (4c). There is therefore no derivative information available, which is needed for the Bernstein quadratic spline. Hence, in this case, the interpolant used is the unique quadratic polynomial passing through the two accepted values of (4c) prior to the discontinuity, and the predicted value of (4c) after the discontinuity.

7.2.2.1. Modification of the phase-plane approach for fully-implicit discontinuities. In the fully-implicit case, practical knowledge of the network can be used to improve the phase-plane method estimate of the root of the discontinuity resulting from a change in state of the control valves. It is known that a small amount of time elapses for the controller output signal to pass from the controller and actuate the valve position. This also becomes apparent from the following analysis.

The particular integral for y_1 given by equation (8) can be rewritten as:

$$Y_p = y_2(t) - 2b_1 t/a_1^2 - b_2/a_1^2 + 2b_1/a_1^3 + o(t^3). \tag{15}$$

Recall that the interpolant for $a_1 y_2(t)$ is given by:

$$a_1 y_2(t) = b_1 t^2 + b_2 t + b_3 + o(t^3). \tag{16}$$

Replacing t by $t - \tau$ gives:

$$a_1 y_2(t-\tau) = b_1 t^2 + b_2 t + b_3 + b_1\tau^2 - 2tb_1\tau - b_2\tau + o(t^3) + o(\tau^3), \tag{17}$$

and therefore

$$a_1 y_2(t) = a_1 y_2(t-\tau) - b_1\tau^2 + (2tb_1 + b_2)\tau + o(\tau^3) + o(t^3). \tag{18}$$

Substituting for $y_2(t)$ from equation (18) in equation (15) gives at time $t = t_D^p$ (the time at which the valve has first been detected to change its state):

$$Y_p = y_2(t_D^p - \tau) - b_1\tau^2/a_1 + (2t_D^p b_1 + b_2)\tau/a_1 - (2t_D^p b_1 + b_2 - 2b_1/a_1)/a_1^2, \tag{19}$$

but taking the initial condition for the first-order ODE to lie at $t = t_D^p$ enables the contribution from the complementary function to be calculated [see equations (6) and (8)]. At $t = t_D^p$ we have:

$$k = y_1(t_D^p) - b_3/a_1 - b_1(t_D^p)^2/a_1 - (b_2a_1 - 2b_1) \times t_D^p/a_1^2 + (b_2a_1 - 2b_1)/a_1^3 + o(t^3). \tag{20}$$

This assumes of course that the valve is unconstrained for $t < t_D^p$. The above two equations lead to:

$$0 = -b_1\tau^2/a_1 + (2t_D^p b_1 + b_2)\tau/a_1, \tag{21}$$

where Y_c can be calculated from equation (20). Equation (21) can be interpreted as a quadratic equation for τ, measuring the amount of time for the control valve to respond to the controller output signal. In essence a subsystem of *delay-differential equations* (Wille and Baker, 1988) has been solved.

The discontinuity is then expected to occur at τ seconds beyond the point at which the signal indicated the valve should change its position. Integration can then be taken to time $t = t_D^p + \tau$, and the coefficients updated if necessary. Unfortunately though, the method depends on predicted information for the valve position which could be unreliable. Furthermore, at fine tolerances the time at which $t = t_D^p$ is observed, may be such that the valve has still not yet changed its state. Using this value of the valve position is likely to lead to inaccurate modelling, though taking the last observed value beyond the discontinuity can lead to implementation

difficulties and relies on the accuracy of old information. In order to create a robust integrator, the algorithm only solves the quadratic equation (21) for τ once. Since the signal can be linear, with the coefficient of τ^2 small relative to the other coefficients, τ is computed by selecting the smallest positive root of the following expression:

$$\tau = \frac{-2C}{B \pm (B^2 - 4AC)^{1/2}}, \tag{22}$$

where A, B, C are the coefficients of the quadratic equation in (21). In the event of the two roots being complex or negative, the estimate of τ is discarded, and the restarting process commences.

8. IMPLEMENTATION OF THE DISCONTINUITY DETECTION ALGORITHMS

In this section,the implementation issues for the new algorithms are described. A number of problem areas are also addressed.

8.1. Detecting discontinuities when the timestep is large

Consider the case when an explicit discontinuity is first detected (at t_{n+1}^p), much earlier than is really the case (at t_D). This is likely to happen if the code takes a large stepsize, assuming the solution to change little over the next timestep. Suppose the interpolation process consistently takes the value of y_1 before the discontinuity and the predicted value of y_1 when the discontinuity was first detected. In this case, it is important that the algorithm allows frequent updating of the predicted time of the discontinuity, and the predicted solution values, in order that it reflects the current solution, rather than the solution from a previous timestep.

Once a discontinuity has been detected, the algorithms described in Section 3 are able to approximate the time at which the discontinuity occurred and set the stepsize to be taken by the integrator on the next step, in order to march up to the discontinuity. The problem here is that the stepsize set by the MONITR routine can be larger than the code allows in the next step as it is based on integrator step size prior to the discontinuity. This can result in error test failures, leading to the unnecessary expense of multiple Jacobian evaluations. The reason for this is that before the discontinuity is first detected, the solution may only vary smoothly, but may vary much more rapidly in the next few steps after the discontinuity.

The above problems have been alleviated by applying the discontinuity location algorithm whenever the predicted time of the discontinuity has just been updated. If the integration step to be attempted is beyond the latest recorded time of the discontinuity, then the stepsize is restricted to be a fraction of the interval straddling the discontinuity (currently 0.77), but on the third successive predicted time beyond the discontinuity, integration is allowed to attempt the step anyway. Numerical experiments have shown that this procedure is robust and enables the discontinuity to be bounded between the latest updated estimate of the discontinuity, and the last integration point. This is useful in that if the discontinuity is detected too early than the upper bound is only restrictive for a short time. If the discontinuity algorithm results in three successive failures, then a switch to the original bisection or similar triangles methods is made. In this case, information from the most recent predicted location of the discontinuity is used.

In the application of an interpolant, it should be pointed out that if the interpolating polynomial passes through two points that are close together, but distant from the discontinuity, then small changes in the interpolated values are likely to produce large changes in the predicted location of the discontinuity.

Another problem can occur using the partly-implicit method if the maximum stepsize set by the user is quite large, since then the contributin from the exponential function can be so large to cause overflow error. Taking an initial condition at a large timestep before the discontinuity is obviously going to seriously affect the accuracy of the algorithm.

8.2. The steady state case

In the steady state $y_1 = y_2$, and the control valve is fixed partly open for some time. The opening or closing of the valve is also sufficient for the condition $y_1 = y_2$ to occur. The Fig. 3 shows the behaviour of y_1 and y_2 against t for the system of equations (1a–g). The crosses mark the timesteps using a local error tolerance of 0.001 and using the MONITR algorithm (Frost, 1987). The steady state is observed at the beginning and end of the time integration.

Clearly a discontinuity occurs only when the solution trajectory for y_1 crosses either 0 or 1. This is why equation (16) has to be adjusted if this approach is used as in Section 7.2.2. The switch function supplied by the user needs to take account of this fact.

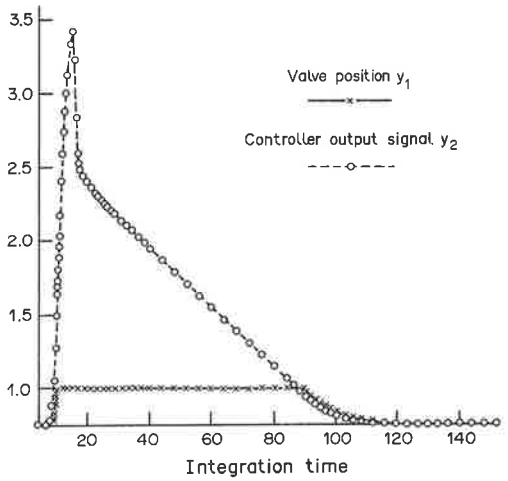

Fig. 3. Graph of y_1 and y_2 against t using the MONITR algorithm (Frost, 1987).

Table 1. Algorithms used for the location of discontinuities

Name	Type of disc.	Method	*PTSB*	*PTSA*
SIMTRi	Partly-implicit	Similar triangles (Frost, 1987)	1	1
BISECT	Fully-implicit on disc. on first step	Bisection (Gear and Osterby, 1981)	1	1
BPOLY*	Partly-implicit	Solution of (18)/Bernstein polynomial	2	0
PHASE*	Fully-implicit	Solution of (22a, 27) fixing $Q_1(t)$	1	1
INTER	Anti-surge valve opening from zero	Interpolation on (4c)	2	1

8.3. Detecting spurious discontinuities caused by the integrator

It is possible for the time integrator to generate spurious discontinuities by the way in which the Jacobian matrix is updated using predicted solution values. The Jacobian matrix is updated whenever convergence failures occur in the Newton method, or the error test fails. The formation of the Jacobian requires a large number of calls to the RESID routine with perturbed solution values. The problem here is that the perturbation could be large enough for the FLAG in the RESID routine to be activated, though this has not been observed for the problems investigated here. This perhaps is due to the fact that the SPRINT software requires rapid convergence of the nonlinear iteration. One way to avoid this possible problem would be to evaluate the Jacobian using the solution values at the end of the previous step. This has, however, been observed to be less efficient in general.

9. COMPARISON OF DISCONTINUITY DETECTION TECHNIQUES

An experimental discontinuity module has been constructed for use with SPRINT (Berzins *et al.*, 1989) which consists of the most suitable discontinuity location algorithms for DSPs discussed earlier in the paper. These are listed in Table 1.

In the above table, *PTSB* denotes the number of interpolation points that are used by the algorithm before the discontinuity, and *PTSA* denotes the number of interpolation points that are used by the algorithm after the discontinuity. The asterisk denotes that derivative information has been supplied to construct a Bernstein quadratic spline.

In the results that follow the method developed for partly-implicit discontinuities is compared with the similar triangles approach (Frost, 1987). The phase-plane approach and modification method developed for fully-implicit discontinuities is also compared with bisection (Gear and Osterby, 1981).

9.1. Results for typical dynamic simulation problems

The performance of the discontinuity module was assessed on three dynamic simulation problems of industrial interest. The statistics are given for intervals of integration which contain a single discontinuity.

The starting point for the observation is marked by integration passing a specified time. The end point of the observation occurs when the algorithm has *detected* the location of the discontinuity to within ϵ. The number of accepted and failed steps, and Jacobian evaluations are then recorded over this interval.

The experimental work has been carried out on the Amdahl 5860 machine at Leeds University. In the tables that follow, STP records the number of steps taken by the integrator using a local error tolerance TOL. The number of accepted steps with the number of failed steps in parenthesis. JAC is the number of Jacobian evaluations, Loc denotes the computed location of the discontinuity and CPU denotes the amount of CPU time taken to locate the discontinuity.

9.1.1. Case 1a. Partly-implicit discontinuity: closing the anti-surge valve. This problem has 76 equations defining the simulation. The interval of observation was taken to be $[130, t_D]$, from a total interval of integration [0, 500]. The maximum stepsize set for the problem was $h = 10$ s. The discontinuity has been "accurately" located at $t = 163.425$. This was obtained using a tolerance of $0.1D - 5$, setting the maximum stepsize to $h = 0.02$ with the original MONITR algorithm (Frost, 1987).

9.1.2. Case 1b. Partly-implicit discontinuity: suction-throttle valve fully-open. This problem is the same as that mentioned above. The interval of observation was taken to be $[240, t_D]$. The discontinuity has been "accurately" located at $t = 255.427$, using a tolerance of $0.1D - 5$ and a maximum stepsize $h = 0.02$ with the original MONITR algorithm (Frost, 1987).

9.1.3. Case 2a. Fully-implicit discontinuity: check valve switching to fully-open. This problem has 77 equations defining the simulation. The interval of

Table 2. Results for Case 1a

	SIMTRI				BPOLY			
TOL	STP	JAC	Loc	CPU	STP	JAC	Loc	CPU
$0.1D-2$	9(18)	24	163.531	19.63	8(8)	14	162.984	22.20
$0.1D-3$	13(15)	22	163.367	23.01	14(4)	11	163.400	25.52
$0.1D-4$	18(15)	21	163.416	20.41	19(6)	11	163.409	19.94
$0.1D-5$	16(12)	15	163.415	10.97	18(2)	5	163.424	10.66

Table 3. Results for Case 1b

	SIMTRI				BPOLY			
TOL	STP	JAC	Loc	CPU	STP	JAC	Loc	CPU
$0.1D-2$	6(14)	18	255.285	10.59	6(5)	8	255.258	7.78
$0.1D-3$	10(16)	21	255.412	13.36	7(9)	13	255.422	10.44
$0.1D-4$	11(15)	21	255.424	15.89	8(3)	6	255.424	7.89
$0.1D-5$	12(12)	15	255.397	8.56	8(5)	7	255.424	5.65

observation was taken to be $[85, t_D]$, from a total interval of integration [0, 150]. The maximum stepsize to be taken was $h = 10$ s. This problem can be shown to be index 1 before the discontinuity, and an index 2 problem after the discontinuity. Note that this is the phase plane method with no modification. (The modified phase plane approach does not apply to the check valve.) For practical reasons, the finest tolerance the program will run at is $0.1D - 3$. Using this tolerance with the original MONITR algorithm (Frost, 1987), and maximum stepsize $h = 0.02$, the discontinuity was "accurately" located at $t = 92.209$.

9.1.4. Case 2b. Fully-implicit discontinuity: suction throttle valve switching from fully-open. For this small problem, described by equations (1a–g) and (2), the interval of observation was taken to be $[70, t_D]$. The maximum stepsize taken by the integrator was $h = 4$ s. The discontinuity has been "accurately" located at $t = 86.478$, using a tolerance of $0.1D - 5$ and maximum stepsize $h = 0.15$ with the original MONITR algorithm (Frost, 1987).

9.1.5. Case 3. Fully-implicit discontinuity: opening the anti-surge valve. The problem taken is equivalent to that described in Case 1a above. The interval of integration was taken to be $[h_1, t_D]$, where h_1 denotes the first step of the integration. The discontinuity has been "accurately" located at $t = 9.940$, using a tolerance $0.1D - 5$ and maximum stepsize $h = 0.02$ with the original MONITR algorithm (Frost, 1987).

Overall, the CPU time is often better than that of the MONITR algorithm (Frost, 1987). The work performed by the new code in terms of the number of successful steps taken is similar to that before, except that the modified phase plane method has resulted in a significant improvement over the number of successful steps taken by MONITR (Frost, 1987). Apart from Case 3, the new algorithm has resulted in significant savings in the number of failed steps. This is due partly to the fact that the MONITR algorithm (Frost, 1987) has no memory of the predicted location of the discontinuity once a successful integration step is taken. Successive predicted steps can therefore frequently overshoot this value.

Further numerical experiments with MONITR (Frost, 1987), using the predicted location of the discontinuity, and the same stepsize strategy as in the new discontinuity algorithm, resulted in a reduction of failed steps and Jacobian evaluations. There was no significant decrease in the number of successful steps taken although the MONITR algorithm (Frost, 1987) with the new strategy was able to locate the discontinuities more accurately than before. As in the new algorithm, the discontinuity location algorithms used by MONITR (Frost, 1987) are applied only after a failed step.

Apart from Case 3, there is also a large reduction in the number of Jacobian evaluations over the MONITR algorithm (Frost, 1987). This is due to the stepsize to be taken for the next step being computed less frequently than in MONITR (Frost, 1987), helped by a reduction in the number of failed steps. Notice also that the CPU time is influenced much more by Jacobian evaluations, rather than integration steps. The reason for this is that the RESID routine contains a large number of calls to complicated thermodynamic routines, and has to be called many times when evaluating the Jacobian.

In Case 3, the efficiency gain is less marked than in Cases 1 and 2. The reason for this is that rapid changes in the solution components in the given

Table 4. Results for Case 2a

	BISECT				PHASE			
TOL	STP	JAC	Loc	CPU	STP	JAC	Loc	CPU
$0.1D-0$	2(12)	14	91.572	8.98	4(10)	14	92.946	12.54
$0.1D-1$	18(13)	24	92.013	21.40	17(3)	14	92.618	23.33
$0.1D-2$	21(10)	18	92.159	14.97	24(3)	15	91.721	25.48
$0.1D-3$	29(7)	12	92.204	9.99	38(2)	7	92.211	10.25

Table 5. Results for Case 2b

	BISECT				PHASE MOD			
TOL	STP	JAC	Loc	CPU	STP	JAC	Loc	CPU
$0.1D-2$	34(25)	31	87.115	0.076	6(2)	4	85.405	0.022
$0.1D-3$	32(23)	29	86.542	0.072	20(4)	13	86.457	0.059
$0.1D-4$	42(31)	38	86.466	0.094	8(6)	10	86.473	0.049
$0.1D-5$	24(16)	21	86.463	0.053	5(2)	4	86.480	0.022

Table 6. Results for Case 3

	BISECT				INTER			
TOL	STP	JAC	Loc	CPU	STP	JAC	Loc	CPU
$0.1D-2$	12(12)	26	9.943	34.17	8(4)	14	9.947	25.80
$0.1D-3$	12(11)	20	9.934	24.14	9(13)	20	9.925	19.15
$0.1D-4$	25(8)	23	9.936	37.57	25(9)	25	9.936	37.23
$0.1D-5$	45(6)	21	9.932	36.66	45(8)	24	9.939	38.80

interval of time resulted in SPRINT (Berzins *et al.*, 1989) severely restricting the stepsize. In short, the stepsizes specified by the location algorithm were largely ignored by the time integrator, except when the tolerance was coarse. At these coarse tolerances, the new algorithm shows inprovement over the old code.

For a considerable reduction in work, the new discontinuity algorithm has provided an estimate to the location of the discontinuity, accurate to a user-specified accuracy ϵ at a prescribed tolerance. The user has control over the error in this location by selecting a suitable value of ϵ before the start of the integration. The modified phase plane method worked well in improving the location of the discontinuity at little extra cost. It is very important that an algorithm locates the discontinuity accurately, particularly in abnormal operating events. If the simulator gives a good representation of the actual system response to control, this can be very beneficial to cost effectiveness.

10. CONCLUSIONS

The efficient handling of discontinuities in a system of DAEs resulting from DSPs has been considered, with the emphasis firmly placed on location. Current discontinuity algorithms, MONITD (Berzins and Furzeland, 1984) and MONITR (Frost, 1987) used within the SPRINT software (Berzins *et al.*, 1989) have been examined. The Newton method used in MONITD was found to be successful only when the switch function supplied by the user was continuous, and when the interval containing the discontinuity was small.

A distinction has been made within the implicit class of discontinuities, resulting in the definition of partly- and fully-implicit discontinuities. A study of the equations causing the valve to change its state led to an alternative approach for the location of partly-implicit discontinuities. This amounted to interpolation on the controller output signal. The interpolant used was a piecewise Bernstein quadratic polynomial which preserved monotonicity and convexity (concavity) in the data between two integration steps. A secant iteration has been applied to solve the resulting nonlinear equation.

The problems of obtaining a suitable switch function for fully-implicit discontinuities appearing in control valves have been discussed. A phase-plane approach has been presented for the fully-implicit case and a considerable gain in efficiency has been achieved over the MONITR approach (Frost, 1987). The exact location of the discontinuity was improved upon by considering the delay parameter from a delay-differential equation.

The research has resulted in a new discontinuity module for SPRINT (Berzins *et al.*, 1989), which has been successfully tested on realistic problems and found to work satisfactorily. This module can handle index 1 and 2 DAEs with discontinuities in solution components or their derivatives for the DSPs under investigation. The need for accurate location has been stressed. Within the prescribed accuracy bounds, the new algorithm was found to do considerably less work than the original method based on similar triangles (Frost, 1987), and bisection (Gear and Osterby, 1981). This improvement was found to be due partly to the change in location algorithm, and also partly due to the stepsize strategy employed. The modified phase plane resulted in further improvements to the location of fully-implicit discontinuities at little extra cost.

Acknowledgements—The help received by Simon Frost and Laurence Scales of Shell Research Ltd for providing the models and insight as to their behaviour, is gratefully acknowledged. The first-named author also fully acknowledges the financial support of SERC and Shell Research Ltd through a Case Studentship.

REFERENCES

Berzins M. and R. M. Furzeland, *A User's Manual for SPRINT: Part 1—Algebraic and Ordinary Differential Equations*. Report TNER.85.058, Thornton Research Centre, Shell Research Ltd (1984).

Berzins M., P. M. Dew and R. M. Furzeland, Developing P.D.E. software using the method of lines and differential algebraic integrators. *Appl. Numer. Math.* **5**, 375–397 (1989).

Berzins M., P. M. Dew and A. J. Preston, Integration algorithms for the dynamic simulation of production processes. Report 88.20, School of Computer Studies, Univers. A condensed version of this report to appear in *Proc. 3rd Euro. Conf. for Maths in Industry*, held at University of Strathclyde (1988).

Brenan K. E., Stability and convergence of difference approximations for higher index differential-algebraic systems with applications in trajectory control. Ph.D. Dissertation, University of California at Los Angeles (1983).

Capstick M. A., On improving the performance of dynamic process simulators. Ph.D. Dissertation, University of Leeds, Dept Chem. Engng/Comput. Studies (1987).

Carver M. B., Efficient integration over discontinuities in ordinary differential simulators. *Math. Comput. Simul.* **20**, 190–196 (1978).

Carver M. B. and S. R. MacEwan, Numerical analysis of a system described by implicitly-defined ODEs containing

numerous discontinuities. *Appl. Math. Modelling* **2,** 281–286 (1978).
Chua T. S., Mathematical software for gas transmission networks, Ph.D. Dissertation, University of Leeds, Dept Comput. Studies (1982).
Chua T. S. and P. M. Dew, The design of a variable-step integrator for the simulation of gas transmission networks. *Inst. J. Numer. Math. Engng* **20,** 1797–1813 (1984).
Curtis A. R., Stiff ODE initial value problems and their solution, the state of the art in numerical analysis. *Proc. of IMA/SIAM Conf.*, pp. 433–450 (1986).
Ellison D., Efficient automatic integration of ODEs with discontinuities. *Math. Comput. Simul.* **23,** 12–20 (1981).
Frost S. R., Private Communication (1987).
Gear C. W., Initial value problems: practical theoretical developments. *Proc. Conf. on Comput. Techniques of ODEs*, University of Manchester (1978). *Computational Techniques of ODEs* (G. Sayers, Ed.), pp. 143–162. Academic Press. New York (1980).
Gear C. W. and O. Osterby, Solving ordinary differential equations with discontinuities. Report UIUCDCS-R-81-1064, Dept Comput. Sci., University of Illinois (1981).
Gear C. W. and L. R. Petzold, O.D.E. methods for the solution of differential algebraic systems. *SIAM J. Numer. Anal.*, **21,** 716–728 (1984).
Gupta G. K., C. W. Gear and B. Leimkuhler, Implementing linear multistep formulas for solving D.A.E.s. Report UIUCDCS-R-85-1205, Dept Comput. Sci., University of Illinois (1985).
Leimkuhler B. J., L. R. Petzold and C. W. Gear, On the consistent initialization of differential algebraic equations. *SIAM. J. Numer. Anal.* **28,** 205-226 (1991).
McAllister D. F. and J. A. Roulier, An algorithm for computing a shape-preserving osculatory quadratic spline. *ACM Trans. Math. Software* **7,** 331–347 (1981).
Pantelides C. C., The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comput.* **9,** 213–232 (1988a).
Pantelides C. C., SPEEDUP—recent advances in process simulation. *Computers chem. Engng* **12,** 745–755 (1988b).
Preston A. J., Integration of dynamic simulation problems. Ph.D. Thesis, School of Computer Studies, Leeds University (1991).
Preston A. J., M. Berzins, P. M. Dew and L. E. Scales, Towards efficient D.A.E. solvers for the solution of dynamic simulation problems. *Proc. IMA Conf. on Numer. Analysis*, Imperial College of Science, Technology and Medicine, London (1989).
Ralston A. and P. Rabinowitz, *A First Course in Numerical Analysis* (2nd Edn). McGraw Hill, New York (1978).
Shampine L. F., I. Gladwell and R. W. Brankin, Reliable solution of special event location problems for ODEs. Numerical Analysis Report 138, Department of Maths, University of Manchester (1987).
Smith G. J., Dynamic simulation of chemical engineering. Ph.D. Dissertation, University of Cambridge (1985).
Stubbe M., A. Bihain, J. Deuse and J. C. Baader, STAG—a new unified software program for the study of the dynamic behaviour of electrical power systems. *IEEE Trans. Power Systems* **4,** 129–136 (1989).
Wille D. R. and C. T. H. Baker, The propagation of derivative discontinuities in systems of delay-differential equations. Numerical Analysis Report No. 160 Dept of Mathematics, University of Manchester (1988).
Ziegler J. G. and N. B. Nichols, Optimum settings for automatic controllers, *ASME Trans.* **64,** (1942).