Mathematical and Computer Modelling of Dynamical Systems,
vol. 7, nr. 2, 2001, 173-187.

Index Reduction and Discontinuity Handling using Substitute Equations

G. Fábián[*], D.A. van Beek[†], J.E. Rooda[†]

Abstract

Several techniques exist for index reduction and consistent initialization of higher index DAEs. Many such techniques change the original set of equations by differentiation, substitution, and/or introduction of new variables. This paper introduces substitute equations as a new language element. By means of a substitute equation, the value of a continuous variable or its time derivative is specified by an expression. This expression is evaluated each time that the variable or its time derivative, respectively, is referenced in the model. The advantage of substitute equations is that they enable index reduction and consistent initialization of higher index DAEs without changing the original equations; no existing variables are removed and no new variables are introduced. Substitute equations can also be used to enable the use of general purpose numerical solvers for equations where one or more of the unknowns are discontinuous, and they can be used to prevent functions to be called outside of their domain.

Keywords: differential algebraic equations, index reduction, simulation languages, hybrid systems.

1 Introduction

The majority of continuous-time and hybrid simulation languages are limited to simulation of ODEs, or DAEs with differential index 1. ODEs are relatively easy to solve numerically, and many ODE solvers are available. For ODEs, the initial values of all differential variables can be specified freely, and the calculation of initial conditions is straightforward. For index 1 DAEs, general purpose solvers are also available, but the numerical solution and calculation of initial conditions for these systems is, in general, more difficult than for ODEs.

Mathematical modelling of physical systems, however, may result in higher index DAEs, for which no general purpose solvers are available. Also, specification and calculation of consistent initial conditions for these systems can be rather difficult. Therefore, these DAEs are usually solved by means of index reduction techniques. Many such techniques change the original set of equations by differentiation, substitution, and/or introduction of new variables. This paper discusses some of the modelling problems that are a result of these index reduction techniques, and it introduces a new way to deal with higher index systems in simulation languages. The paper is organized as follows. First, some of the currently known techniques are introduced. Second, substitute equations are introduced as a new language element for index reduction. The syntax, semantics, application, and implementation of this new language element are discussed, and several examples of its application are given. Finally, the use of substitute equations for discontinuity handling in simulators is discussed, and conclusions are presented.

2 The higher index problem

DAEs are differential equations with additional algebraic constraints in the general form

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}, t) = \mathbf{0}, \tag{1}$$

[*]KPN Research, P.O. Box 421, 2260 AK Leidschendam, Netherlands. E-mail: [1]g.fabian@kpn.com

[†]Eindhoven University of Technology, Dept. of Mechanical Engineering, P.O. Box 513, 5600 MB Eindhoven, Netherlands. E-mail: d.a.v.beek@tue.nl

where $\mathbf{x} \in I\!R^n$ is the vector of differential variables, $\mathbf{y} \in I\!R^m$ is the vector of algebraic variables, $t \in I\!R$ is the independent variable and $\mathbf{f} \in I\!R^{2n+m+1} \to I\!R^{n+m}$ is the set of DAEs.

The initial conditions $\dot{\mathbf{x}}(t_0), \mathbf{x}(t_0), \mathbf{y}(t_0)$, denoted by $\dot{\mathbf{x}}_0, \mathbf{x}_0, \mathbf{y}_0$, cannot be specified independently, since they must satisfy (1) at an initial time t_0

$$\mathbf{f}(\dot{\mathbf{x}}_0, \mathbf{x}_0, \mathbf{y}_0, t_0) = \mathbf{0}. \tag{2}$$

DAEs are characterized by their (differential) index [7]. The index of (1) is m, if m is the smallest number such that the system of equations

$$\begin{aligned} \mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}, t) &= \mathbf{0}, \\ \frac{d\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}, t)}{dt} &= \mathbf{0}, \\ &\vdots \\ \frac{d^{(m)}\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}, t)}{dt^{(m)}} &= \mathbf{0}, \end{aligned} \tag{3}$$

can be transformed into an explicit ODE ($[\dot{\mathbf{x}}, \dot{\mathbf{y}}]^{\mathrm{T}} = \mathbf{g}(\mathbf{x}, \mathbf{y}, t)$) by algebraic manipulations. In general, the higher the index, the greater the numerical difficulty one encounters when trying to solve the system numerically.

The number of degrees of freedom of a set of DAEs is the number of initial conditions from the total set of $2n + m$ initial conditions $\{\dot{\mathbf{x}}_0, \mathbf{x}_0, \mathbf{y}_0\}$ that can be specified independently; or, in other words, it is the number of variables from the set $\{\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}\}$ that can be given independent initial values. For ODEs ($\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, t)$), which have index 0, the number of degrees of freedom equals the number of differential variables n ($\mathbf{x} \in I\!R^n$). For DAEs (1) of index 1, apart from a small number of exceptions (see Section 7), the number of degrees of freedom also equals the number of differential variables n ($\mathbf{x} \in I\!R^n$); under the assumption that the set of $n + m$ DAEs (1) is regular with respect to the remaining $n + m$ variables. In many cases, the n differential variables of index 1 DAEs are given independent initial values $\mathbf{x}_0$, and values of $\dot{\mathbf{x}}_0$ and $\mathbf{y}_0$ are calculated from (2). Another possibility is steady state initialization, which corresponds to providing initial values $\mathbf{0}$ for $\dot{\mathbf{x}}$, and calculating $\mathbf{x}_0$ and $\mathbf{y}_0$ from (2).

A common feature of higher index DAEs is that there are hidden constraints in the DAEs. Hidden constraints are equations that further restrict the initial conditions given by (2). They can be obtained after differentiation and algebraic manipulations [7, 12]. The presence of hidden constraints means that the number of degrees of freedom of (1) is smaller than n. In such a case, not all differential variables $\mathbf{x}$ may be independently initialized; there are dependencies among them. This can be seen, if equations in the form

$$\mathbf{g}(\mathbf{x}, t) = \mathbf{0} \tag{4}$$

are present in (1), or if they can be obtained after algebraic manipulations. The differential variables in (4) are dependent differential variables. The dependencies among the derivatives become visible after differentiation of (4). In fact, hidden constraints may be present in index 1 systems of DAEs too (see Section 7).

3 Index reduction

It is well known, that mathematical models of physical systems may have a higher (differential) index. The usual technique to solve such higher index DAEs is by index reduction; through differentiation and algebraic manipulations, the index is lowered to 0 or 1. In literature, several algorithms can be found for index reduction. From these, the algorithm of Gear and Petzold [8], the constraint stabilization technique of Gear [7], and the algorithm of Bachmann et al. [1] all differentiate (parts of) the system of equations and use substitution. Two well known general purpose index reduction algorithms that are used in simulation languages are the Pantelides algorithm [12], and the dummy derivatives algorithm [11]. The Pantelides algorithm is used in the hybrid simulation language Modelica [10], whereas the dummy derivatives algorithm is used in the hybrid simulation language ABACUSS [6]. However, since integrating these algorithms in a simulation environment is not a trivial matter, few other simulation languages provide the functionality of numerical solution and consistent initialization of higher index DAE systems.

A problem with many index reduction techniques is that the additional equations obtained by differentiation lead to an overdetermined system, that cannot be solved by widely used general purpose solvers, such as DASSL, or DASRT [13]. Removing some of the additional equations can make the set of equations solvable by general purpose solvers. However, by doing so, the solution will slowly drift away from the intended, original solution. Another problem of index reduction methods is that these methods change the original set of equations by subsequent steps of differentiations and algebraic manipulations (substitution). This may lead to a modelling problem. Since not all differential variables may be freely chosen in such a system (assuming that, by default, the modeller specifies initial values for the differential variables), it must be clear for the modeller which differential variables may be initialized, and which ones are calculated from the equations. Even more, the modeller may want to choose the variables that he or she wants to initialize. In fact, [9] states that many approaches for the consistent initialization problem analyze the set of equations and ask the user for the initial conditions for a particular set of variables, which the user might not have.

For simulators that have not implemented index reduction techniques, only lower index systems (index 0 or 1) of equations may be entered. In some cases, higher index systems can be avoided by careful modelling. For chemical systems this is illustrated in [14]. In other cases, the modeller has to perform index reduction, and has to re-formulate the system of equations. In such a case, a new equation set is obtained that is usually less expressive than the original one. Also, variables may be eliminated from the equations due to index reduction. Therefore, each time the values of these variables are needed in the model, they must be re-calculated. This also reduces the readability of models.

4 Substitute equations

To overcome the problems mentioned above, in the χ language [16, 5] substitute equations are used. By means of a substitute equation, the value of a continuous variable or its time derivative is specified by an expression. This expression is evaluated each time that the variable or its time derivative, respectively, is referenced in the model. The advantage of substitute equations is that they enable index reduction and consistent initialization of higher index DAEs without changing the original equations; no existing variables are removed and no new variables are introduced. The modeller has full control over which variables he wants to initialize, and the implementation of substitute equations is straightforward. As an additional advantage, widely used general purpose solvers, such as DASSL, or DASRT [13], can be used to solve the resulting set of equations. Substitute equations are, however, not a general purpose index reduction mechanism. They are applicable to those higher index systems where one (or more) of the differential variables can be expressed as an explicit function of the remaining differential variables. Consider the general implicit form of (1).

If the vector of differential variables $\mathbf{x} \in I\!R^n$ can be partitioned into $\mathbf{z} \in I\!R^{n-1}$ and $v \in I\!R$ such that (1) can be expressed in the form

$$\begin{aligned} \tilde{\mathbf{f}}(\dot{\mathbf{z}}, \mathbf{z}, \dot{v}, v, \mathbf{y}, t) &= \mathbf{0} && \text{(5a)} \\ v &= \phi(\mathbf{z}, t), && \text{(5b)} \end{aligned}$$

i.e., v is expressed as an explicit function of the remaining differential variables and the time, then substitute equations can be used to calculate $\dot{v}$. In order to remove the dependency between the differential variables $[\mathbf{z}, v]^{\mathrm{T}}$ in (5b), this equation is differentiated, leading to the hidden constraint

$$\dot{v} = \phi_t + \phi_{\mathbf{z}}\dot{\mathbf{z}}. \tag{6}$$

By specifying (6) as a substitute equation, the following model is obtained

$$\begin{aligned} \tilde{\mathbf{f}}(\dot{\mathbf{z}}, \mathbf{z}, \dot{v}, v, \mathbf{y}, t) &= \mathbf{0} && \text{(7a)} \\ v &= \phi(\mathbf{z}, t) && \text{(7b)} \\ \dot{v} &\leftarrow \phi_t + \phi_{\mathbf{z}}\dot{\mathbf{z}}. && \text{(7c)} \end{aligned}$$

The left arrow indicates a substitute equation. This model is equivalent to the model given by Equations 5, where every occurrence of $\dot{v}$ in Equations 5 is replaced by $\phi_t + \phi_{\mathbf{z}}\dot{\mathbf{z}}$. In this way, differential variable v is in fact changed to an algebraic variable. This is analogous to the dummy derivatives method [11]; substitution of $\dot{v}$ by a dummy variable d in Equations 5, 6 leads to

$$\begin{aligned}
\tilde{\mathbf{f}}(\dot{\mathbf{z}}, \mathbf{z}, d, v, \mathbf{y}, t) &= \mathbf{0} && \text{(8a)}\\
v &= \phi(\mathbf{z}, t) && \text{(8b)}\\
d &= \phi_t + \phi_{\mathbf{z}}\dot{\mathbf{z}}, && \text{(8c)}
\end{aligned}$$

It is clear from these equations, that variable v has become algebraic instead of differential, so that (8b) no longer specifies a direct dependency between the differential variables $\mathbf{z}$. A difference between the dummy derivatives algorithm and the substitute equations technique, is that the latter technique leaves the original equations unchanged, retaining the expressivity of the original model. A further difference is that the substitute equation technique does not provide an algorithm for automatic index reduction, such as the dummy derivatives algorithm, or the Pantelides algorithm. Rather, it provides the modeller with substitute equations as a new language element; but the modeller should perform the necessary algebraic manipulations and differentiations to obtain equations of the form (7). Higher index systems that are candidate for index reduction by means of substitute equations, generally contain equations of the form

$$\mathbf{g}(\mathbf{x}, t) = \mathbf{0}, \tag{9}$$

or

$$\begin{aligned}
\mathbf{g}(\mathbf{x}, \mathbf{u}, t) &= \mathbf{0} && \text{(10a)}\\
\mathbf{u} &= \mathbf{h}(t), && \text{(10b)}
\end{aligned}$$

where $\mathbf{x}$ are differential variables, and $\mathbf{u}$ are algebraic variables. The conditions on the form of $\tilde{\mathbf{f}}$ in (5a), that guarantee that the index of Equations 7 is 1 lower than the index of Equations 5 have not yet been determined.

5 Substitute equations in the χ language

In the χ language, substitution can be specified explicitly by means of substitute equations in two forms. The simple form, expressed in Backus-Naur Form (BNF), is

$$\begin{aligned}
S &::= v \leftarrow E \mid v' \leftarrow E\\
E &::= e
\end{aligned}$$

where S and E are nonterminals, v is a continuous variable, v' is the time derivative of a continuous variable and e is an expression of type real. The variable that is defined on the left hand side of a substitute equation (in this case, v and v') is termed a substituted variable. There may only be one substitute equation for each substituted variable. The meaning of a substitute equation $v \leftarrow e$ (or $v' \leftarrow e$) is that all occurrences of v (or v') in the model are replaced by e. The guarded form of E is

$$E \;::=\; [\; b_1 \longrightarrow e_1 \;[]\; \ldots \;[]\; b_n \longrightarrow e_n \;]$$

where b_i is a boolean guard and e_i is an expression of type real ($i = 1 \ldots n$). In this case, variable v (or v') is substituted dynamically, depending on the values of the guards. If b_i is true, variable v (or v') is substituted by e_i. If more guards are true at the same time, one alternative is chosen nondeterministically and all occurrences of v (or v') in the model are calculated from this alternative.

All variables occurring in the right-hand-side of substitute equations (in expressions e, e_i and b_i, $i = 1 \ldots n$) must be well-defined; they can either be defined as a constant, as a discrete variable, or as a continuous variable that is either defined by another substitute equation or by normal, non-substitute, equations. Substitute equations are evaluated recursively; if a substituted variable occurs on the right-hand-side of a substitute equation, first, its value is calculated by substitution. Therefore, substitute equations can be specified in arbitrary order; the only requirement is that they may not contain circular dependencies. The implementation of substitute equations is straightforward. Whenever the value of a substituted variable is required, the simulator evaluates the right hand side of the substitute equation that defines the value of the substituted variable. In the next sections, some examples of the use of substitute equations are given.

6 Examples

6.1 PID control example

As an example of a higher index system, take the following PID (proportional integral differential) controller. A horizontal force F is applied to a body of mass 1 on a flat surface, without friction. The position of the body is denoted by x. The control objective is to keep the body at a given position x_{set}. The unknowns are x, v, i, e, u. Variable F is an input variable (depending only on time), x_{set}, k_P, k_D and k_I are constants.

$$\begin{aligned} \dot{x} &= v && \text{(11a)} \\ \dot{v} &= F - u && \text{(11b)} \\ \dot{i} &= e && \text{(11c)} \\ e &= x - x_{\text{set}} && \text{(11d)} \\ u &= k_P e + k_D \dot{e} + k_I i && \text{(11e)} \end{aligned}$$

This is an index 2 system of DAEs. Differentiation of (11d) yields

$$\dot{e} = \dot{x}. \tag{12}$$

After differentiating (11a), (11e), and differentiating (12) a second time, the ODE form can be obtained by performing the following substitutions in the differentiated version of (11e): v for $\dot{e}$, $F - u$ for $\ddot{e}$ ($\ddot{e} = \ddot{x} = \dot{v} = F - u$), and e for $\dot{i}$. Typically to higher index systems, (11d) contains a hidden constraint that appears after differentiation: (12) must also hold when initial conditions are calculated.

For automatic calculation of the initial conditions, in general, the differential variables (x, v, i, and e) are considered as known (their initial values are supplied by the modeller), and the derivatives and the algebraic variables are calculated from the equations. This approach generally works well for index 1 systems, but it clearly fails in this case. Although the number of unknowns (5: $\dot{x}$, $\dot{v}$, $\dot{i}$, $\dot{e}$, u) is equal to the number of equations, the unknowns occur in only 4 out of the 5 equations, making it impossible to calculate them in this way.

A substitute equation can be used to reduce the index to 1, and to allow automatic calculation of the initial conditions. Equation 11d is already in the form of (5b). After differentiating this equation and selecting $\dot{e}$ for substitution, the system is specified in χ as follows

$$\begin{aligned} &\quad x' = v \\ &, v' = F - u \\ &, e \ = x - x_{\text{set}} \\ &, i' \ = e \\ &, u \ = k_P e + k_D e' + k_I i \\ &, e' \leftarrow x' \end{aligned}$$

Whenever the value of expression e' is required, the value of x' is evaluated instead. In this model, x can be freely initialized, but the value of e depends on x. The actual set of equations solved by numerical solvers is

$$\begin{aligned} \dot{x} &= v && \text{(13a)} \\ \dot{v} &= F - u && \text{(13b)} \\ e &= x - x_{\text{set}} && \text{(13c)} \\ \dot{i} &= e && \text{(13d)} \\ u &= k_P e + k_D \dot{x} + k_I i. && \text{(13e)} \end{aligned}$$

Note that for the solvers, $\dot{e}$ is not present in the equations, e has thus become an algebraic variable. After initialization of the differential variables (x, v, i), the derivatives and algebraic variables ($\dot{x}$, $\dot{v}$, $\dot{i}$, e, u) can now be automatically calculated from Equations 13. By means of the substitute equation, the index of the systems has been reduced from 2 to 1.

If the modeller would like to specify the initial value of e, instead of x, the substitute equation would become

$$x' \leftarrow e'. \tag{14}$$

If x_{set} is not a constant, but a prescribed function of time, say $\sin(\omega t)$, instead of one equation $e = x - x_{\mathrm{set}}$, we get two equations

$$\begin{aligned} e &= x - x_{\mathrm{set}} && (15a)\\ x_{\mathrm{set}} &= \sin(\omega t). && (15b) \end{aligned}$$

The substitute equation for e then becomes

$$e' \leftarrow x' - \cos(\omega t). \tag{16}$$

The advantage of using substitute equations is that the process of substitution is transparent; the original form of the equations is preserved and the additional information used ($e' = x'$) is made explicit. Also, references to the substituted variable in the discrete-event part of the model need not be altered; in the example, e' can be referenced in any discrete statement.

6.2 Composition of subsystems example

Higher index systems can be a result of connecting two index 1 or index 0 systems. Consider two capacitors with capacitance $C_1$and C_2, respectively, that are described by the ODEs

$$\begin{aligned} \dot{v}_1 &= \frac{i_1}{C_1} && (17a)\\ \dot{v}_2 &= \frac{i_2}{C_2}. && (17b) \end{aligned}$$

If the two systems are interconnected, so that the capacitors are in parallel, the equations become

$$\begin{aligned} \dot{v}_1 &= \frac{i_1}{C_1} && (18a)\\ \dot{v}_2 &= \frac{i_2}{C_2} && (18b)\\ i_1 + i_2 &= 0 && (18c)\\ v_1 &= v_2. && (18d) \end{aligned}$$

This is an index 2 system, because in order to derive an expression for $\dot{i}_1$ and $\dot{i}_2$, the first three equations need to be differentiated. This results in equations for $\ddot{v}_1$ and $\ddot{v}_2$. By differentiating (18d) twice, $\ddot{v}_1$ and $\ddot{v}_2$ can be eliminated, so that $\dot{i}_1$ and $\dot{i}_2$ can be solved. Together with the already available equations for $\dot{v}_1$ and $\dot{v}_2$, the ODE is then obtained. In the parallel system, the two voltages v_1 and v_2 can no longer be initialized independently; even if the two voltages are initialized to the same value, automatic calculation of the initial conditions by taking v_1 and v_2 as known, and $\dot{v}_1$, $\dot{v}_2$, i_1, and i_2 as unknown fails, because the four equations are singular with respect to these four variables.

A corresponding index 1 model with substitute equations would be

$$\begin{aligned} \dot{v}_1 &= \frac{i_1}{C_1} && (19a)\\ \dot{v}_2 &= \frac{i_2}{C_2} && (19b)\\ i_1 + i_2 &= 0 && (19c)\\ v_1 &= v_2 && (19d)\\ \dot{v}_2 &\leftarrow \dot{v}_1. && (19e) \end{aligned}$$

The meaning of the additional substitute equation is that all occurrences of $\dot{v}_2$ are substituted by $\dot{v}_1$. Therefore, for the solvers, $\dot{v}_2$ is no longer present in the model, so that variable v_2 becomes algebraic. The initial value of v_1 may be specified by the modeller. Automatic calculation of initial conditions is now possible, because v_1 is taken as known, and $\dot{v}_1$, v_2, i_1, and i_2 are taken as unknown. Mathematically, Equations 19 are equivalent to

$$\dot{v}_1 = \frac{i_1}{C_1} \tag{20a}$$

$$\frac{i_1}{C_1} = \frac{i_2}{C_2} \tag{20b}$$

$$i_1 + i_2 = 0 \tag{20c}$$

$$v_1 = v_2. \tag{20d}$$

7 Consistent initialization of index 1 systems

In general, higher index DAEs coincide with hidden constraints in the equations. Hidden constraints can, however, also be present in index 1 DAEs. This is the case when there are dependent differential variables in the index 1 DAE system. Again, the solution is the use of a substitute equation. The PID control system is used as an example.

The index of the PID control example can also be reduced by replacing variable u by the derivative of a dummy variable z. The set of equations then becomes

$$\dot{x} = v \tag{21a}$$

$$\dot{v} = \frac{F - \dot{z}}{m} \tag{21b}$$

$$e = x - x_{set} \tag{21c}$$

$$\dot{i} = e \tag{21d}$$

$$\dot{z} = k_P e + k_D \dot{e} + k_I i. \tag{21e}$$

This is an index 1 problem, because after differentiating (21c), the equations can be re-arranged into an ODE. Yet, e and x remain dependent differential variables, so that they cannot be initialized independently. As a consequence, there is a hidden constraint in (21c), which appears after differentiation of the equation. The initialization problem can easily be solved, as before, by adding a substitute equation for $\dot{e}$.

8 Modelling and simulation of discontinuities

8.1 Handling of discontinuous variables by means of substitute equations

Another application area for substitute equations is the modelling of discontinuous functions. General purpose DAE and ODE solvers cannot usually integrate discontinuous functions [3]. The usual approach is that discontinuities are specified by so called *switching functions*. When the sign of the switching function changes, a discontinuity occurs. Integration stops, and is re-started again after the discontinuity. For more on numerical methods with respect to discontinuities we refer to [4].

A discontinuity in a variable that is used by the solver can be avoided in cases where the discontinuous variable can be expressed in a closed form. This variable can then be calculated by substitution, and thereby, it is removed from the equation set that is actually solved by numerical solvers. As an example, consider a tank described in [15], where overflow occurs if the level h of its contents reaches a maximum height h_{max}. The incoming and outgoing flows are denoted by Q_i and Q_o, respectively; the area of the tank by A, and the overflow by Q_x. The system described by a conditional equation is

```
[ $h < h_{max} \vee Q_i < Q_o \longrightarrow Ah' = Q_i - Q_o,\ Q_x = 0$
[] $h \geq h_{max} \wedge Q_i \geq Q_o \longrightarrow Ah' = 0,\ Q_x = Q_i - Q_o$
]
```

The general form of a conditional equation is: $[\, b_1 \longrightarrow \mathit{DAEs}_1 \ [] \ \ldots \ [] \ b_n \longrightarrow \mathit{DAEs}_n \,]$, where DAEs_i $(1 \le i \le n)$ represents one or more equations separated by commas. Boolean expression b_i denotes a guard. At any time, (at least) one of these guards must be open (true), so that the DAE(s) associated with the open guard (after the arrow of the open guard) is (are) activated. The discontinuous variable Q_x can be removed from the equations by substitution

$$\begin{array}{lll} & Ah' = & Q_i - Q_o - Q_x \\ , & Q_x \leftarrow & [\ h < h_{max} \vee Q_i < Q_o \longrightarrow 0 \\ & & [] \ h \ge h_{max} \wedge Q_i \ge Q_o \longrightarrow Q_i - Q_o \\ & &] \end{array}$$

In this case, only the first equation, $Ah' = Q_i - Q_o - Q_x$ is solved by integration. The fact that variable h has a discontinuous first derivative is usually not a problem for numerical integrators.

8.2 Substitute equations and root finding

A well known mechanism to handle discontinuities in simulators is based on the root-finding algorithm present in certain general purpose solvers, such as DASRT [2]. Using such a mechanism, relations in the guards of conditional equations, such as $h < h_{max}$, and $Q_i < Q_o$, are converted into so called root functions $h - h_{max}$, and $Q_i - Q_o$, respectively. During integration, no discontinuities take place, and the equation set is not changed. In the example of the overflowing tank from the previous section, the equation set would be either $\{Ah' = Q_i - Q_o,\ Q_x = 0\}$, or $\{Ah' = 0,\ Q_x = Q_i - Q_o\}$, depending on the values of the guards (e.g. $h < h_{max} \vee Q_i < Q_o$) at the start of the integration interval. Integration continues until a root function changes its sign (has crossed zero), and causes the value of a guard to change. The exact location of the zero crossing is then determined (root finding), and integration stops. Integration is restarted *after* the 'discontinuity', using a new set of equations, and after first calculating the new initial conditions. Please note that the new initial conditions do not necessarily imply any variables to be discontinuous. It is also possible that there is only a discontinuous derivative, or even that there is no discontinuity at all. This approach, based on root finding, is not always possible. Consider, for example, the conditional equation that calculates the square root of an argument, and returns zero for negative arguments:

$$\begin{array}{l} [\ x \ge 0 \longrightarrow y = \sqrt{x} \\ [] \ x < 0 \longrightarrow y = 0 \\ ] \end{array}$$

Here, the root finding mechanism may not be used, because it may cause the square root function to be called outside of its domain. This happens when x crosses zero coming from above zero, so that equation $y = \sqrt{x}$ is active. In such as case, a root of a negative number would be taken. In order to prevent this from happening, a substitute equation can be used:

$$\begin{array}{ll} y \leftarrow & [\ x \ge 0 \longrightarrow \sqrt{x} \\ & [] \ x < 0 \longrightarrow 0 \\ &] \end{array}$$

In substitute equations, no root finding takes place. The advantage of using a substitute equation for this purpose, is that the language is kept small, because no additional language elements are required. In the Modelica [10] language, the *noevent* directive is used to indicate that the root finding mechanism may not be used.

9 Conclusions

Substitute equations make the mechanism of index reduction transparent to users. The original equation set is unchanged, so that substituted variables do not disappear from the model; they can still be used in discrete-event statements. In this way, expressiveness of the models is preserved. Furthermore, the use of substitute equations makes it clear which variables can be initialized independently of one another by the modeller, and which ones are calculated. Substitute equations can also be used to reveal hidden constraints in index 1 DAEs. Finally, substitute equations enable the use of general purpose numerical solvers for equations where an unknown variable is discontinuous, and they can be used to prevent functions to be called outside of their domain.

References

[1] Bachmann, R., Brüll, L., Mrziglod, T., and Pallaske, U.: On methods for reducing the index of differential algebraic equations. *Computers and Chemical Engineering*, 14: 1271–1273, 1990.

[2] Brenan, K.E., Campbell, S.L., and Petzold, L.R.: *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM's Classics in Applied Mathematics. Siam, Philadelphia, 1996.

[3] Cellier, F.E., Elmqvist, H., Otter, M., and Taylor, J.H.: Guidelines for modeling and simulation of hybrid systems. In *IFAC 12th Triennial World Congress*, pp. 1219–1225, Sydney, 1993.

[4] Eich-Soellner, E. and Fuehrer, C.: *Numerical methods in multibody dynamics*. Teubner, Stuttgart, 1998.

[5] Fábián, G.: *A Language and Simulator for Hybrid Systems*. Ph.D. thesis, Eindhoven University of Technology, 1999.

[6] Feehery, W.F. and Barton, P.I.: A differentiation-based approach to dynamic simulation and optimization with high-index differential-algebraic equations. In *Proceedings of 2nd. International Workshop on Computational Differentiation*, pp. 239–52, Santa Fe, 1996.

[7] Gear, C.W.: Differential-algebraic equation index transformations. *SIAM. J. Sci. Stat. Comp.*, 9: 39–47, 1988.

[8] Gear, C.W. and Petzold, L.R.: ODE methods for the solution of differential/algebraic systems. *SIAM Journal on Numerical Analysis*, 21: 716–728, 1984.

[9] Gopal, V. and Biegler, L.: A successive linear programming approach for initialization and reinitialization after discontinuities of differential-algebraic equations. *SIAM J. Sci. Comput.*, 20(2): 447–467, 1998.

[10] Mattsson, S.E., Elmqvist, H., and Otter, M.: Physical system modeling with Modelica. *Control Engineering Practice*, 6: 501–510, 1998.

[11] Mattsson, S.E. and Söderlind, G.: Index reduction in differential-algebraic equations using dummy derivatives. *SIAM J. Sci. Comput.*, 14(3): 677–692, 1993.

[12] Pantelides, C.C.: The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comput.*, 9(2): 213–231, 1988.

[13] Petzold, L.R.: A description of DASSL: A differential/algebraic system solver. *Scientific Computing*, 65–68, 1983.

[14] Ponton, J.W. and Gawthrop, P.J.: Systematic construction of dynamic models for phase equilibrium processes. *Computers & Chemical Engineering*, 15(12): 803–808, 1991.

[15] Van Beek, D.A. and Rooda, J.E.: Specification of discontinuities in hybrid models. In J. Zaytoon, editor, *Hybrid Dynamical Systems—Proc. of 3rd International Conference on Automation of Mixed Processes*, pp. 415–420, Reims, 1998.

[16] Van Beek, D.A. and Rooda, J.E.: Languages and applications in hybrid modelling and simulation: Positioning of Chi. *Control Engineering Practice*, 8(1): 81–91, 2000.