In *Proc. of the 11th. European Simulation Symposium*, Erlangen, 1999, pp. 88-92.

HYBRID MODELLING AND SIMULATION OF TIME-DELAY ELEMENTS

D.A. van Beek, J.E. Rooda, and B.J. Trienekens
Department of Mechanical Engineering
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
E-mail: d.a.v.beek@tue.nl

ASTRACT

Most general purpose simulation languages do not provide support for modelling and simulation of delay elements with variable delay times. In this paper, combined discrete-event / continuous-time (hybrid) models of time delay elements are treated. An introduction is given to discrete-event and continuous-time models of delay elements. The hybrid models are compared to the traditional models using a step and sine input function. The hybrid models outperform the traditional models on accuracy, computation time, and stability. The most important aspect of the hybrid models with a variable delay time is that between two adjacent sample points, the volume entering the element is constant.

INTRODUCTION

Whenever energy or material is physically moved in a process or plant, there is a time delay associated with the movement. Examples can be found in chemical plants where pipes are used to transport liquids between tanks and reactors. If the time delay is not considerably smaller than the time scale of the other relevant phenomena, so that it cannot be ignored, it should be included in the model. The mathematical formulation of the time delay phenomenon is the linear advective equation

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0 \tag{1}$$

The initial profile is $u(x,0) = u_{\mathrm{ip}}(x)$, while the boundary condition $u(0,t) = u_0(t)$ represents the input function for the time delay element. The first order hyperbolic partial differential equation (PDE) (1) is particularly difficult to solve numerically, because it transmits discontinuities without dispersion or dissipation (Carver and Hinds 1978). Furthermore, it is liable to produce numerical oscillations (Carver and Hinds 1978; Hu and Shao 1988).

Time delays are often related to transportation of liquids through pipes in process industry plants. In such cases, only the input-output relations of the pipe are relevant, such as the relation between molar concentrations in the liquid at the input and output of the pipe. The input-output relation can be denoted as

$$u_L(t) = u_0(t - t_{\mathrm{d}}) \tag{2}$$

where $u_L = u(x = L, t)$, L is the length of the pipe, and t_{d} is the time delay. The time delay is constant if the flow rate is constant. If the flow rate is not constant, t_{d} depends on the flow rate.

Morton and Smith (1989) proposed a simulation algorithm for dealing with time delays in a dynamic process simulator. The algorithm can deal with constant and varying time delays, and also with discontinuities. In most simulation languages suited to dynamical modelling of process industry plants, time delays cannot be modelled in the form of Equation (2). The reason for this is that simulation algorithms such as the one described by Morton and Smith (1989) are usually not implemented. Therefore, the time delay element must be modelled in a different way. Several modelling approximations of time-delays are known from literature. The approximation models are either discrete-event (DE) models or continuous-time (CT) models. DE models handle discontinuities well, but are not very accurate. CT models, on the other hand, are accurate, but do not handle discontinuities very well. This paper proposes combined DE/CT (hybrid) models, that combine the advantages of the DE and CT models. The proposed models are not, however, a straightforward combination of the DE and CT models known from literature. New modelling techniques are used that result in short and elegant models. In the sequel, some DE and CT modelling techniques known from literature are discussed. Subsequently, the new hybrid modelling techniques are discussed, preceded by an introduction to the χ language that is used to specify the models. Finally, simulation results and conclusions are presented.

DISCRETE-EVENT MODELS

Discrete-event approximations of delay elements are based on arrays where past samples are stored. Arrays are also used in the steady state simulation language ASCEND II (Kuru 1981), and in the simulation algorithm described by Morton and Smith (1989). They are used in the following way (Coleman 1965; Franks 1972). Consider a delay element with input u_0 and output u_L. Assume the delay time t_{d} to be fixed. The input u_0 is sampled every t_{d}/n time-units. The samples are stored in the array of size n. Figure 1 gives a graphical representation of the array. The start location of the array is connected to the end location, so that a carrousel is obtained. A write and read pointer point to the current array location. At every sample time point, the current location is read and assigned to output u_L; subsequently a new sample is taken of input u_0 and

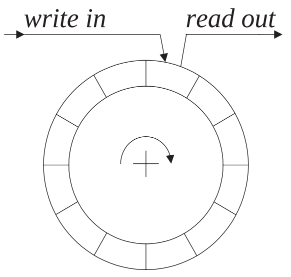

Figure 1: Delay element model using an array.

assigned to the current array location. After this, the carrousel is turned one position, so that the pointers point to the next array location. In this way, every new sample will arrive at the output after $n \cdot t_\mathrm{d}/n = t_\mathrm{d}$ time-units. Initially, the array is filled with values representing the initial state of the delay element.

The algorithm treated above is based on constant time delays. In such as case, the output u_L of the array based models will be exact at the sample time points. Between two sample points, the output can be kept constant or linear interpolation between two subsequent points in the array can be used. In the algorithm of Morton and Smith (1989), multi-point interpolation is used.

Ball (1966) adapted the algorithm described above for variable time delays. The time-delay element considered is a pipe with variable flow rate Q. The sample time t_s remains constant, but besides input u_0, also flow rate Q is sampled. The flow rate is used to approximate the liquid volume that has entered the pipe during the last sample interval: $\Delta V \approx Q_\mathrm{s} \cdot t_\mathrm{s}$. In the array, both the input u_0 and the calculated volume ΔV are stored. When the sum of the volumes stored in the list exceeds the volume of the pipe, an estimation of the output is made. The output of this model is much less accurate than the output of the models with a constant time delay, because of the many approximations that are required.

CONTINUOUS-TIME MODELS

Continuous-time approximations of delay elements are usually based on the method of lines. The principle of this method is to discretize all independent variables except one, such that the PDEs are converted into a set of ordinary differential equations. In the case of the linear advective equation, u is discretized with respect to x. For this space discretization usually the finite-difference or finite-element methods are used (Carver and Hinds 1978; Metzger 1996). Consider a pipe of length 1, and variable x that denotes the position along the pipe: $x \in [0, 1]$. Space discretization implies division of interval $[0, 1]$ into n equal pieces of $1/n$. This leads to $n + 1$ grid points x_j along the x-axis, such that $x_0 = 0, x_1 = \frac{1}{n}, x_2 = \frac{2}{n}, \cdots, x_n = 1$. Dependent variable $u(x, t)$ is now approximated by $n + 1$ variables $u_j(t)$ that are no longer dependent on x. Variables $u_j(t)$ can be regarded as approximations of $u(x, t)$ along the grid points, such that $u_j(t)$ approximates $u(x = x_j, t)$.

Experiments by Metzger (1996) indicate no major performance differences between finite difference and finite element methods. In the sequel the finite difference four-point upwind biased formula (4UB) is used, since it was among the best performers in studies by Trienekens (1998), Metzger (1996), and Metzger (1994):

$$\left(\frac{\partial u}{\partial x}\right)_\mathrm{j} = \frac{2u_{\mathrm{j}+1} + 3u_\mathrm{j} - 6u_{\mathrm{j}-1} + u_{\mathrm{j}-2}}{6\Delta x} \tag{3}$$

At the boundaries of the element the two-point upwind formula (2U) is used:

$$\left(\frac{\partial u}{\partial x}\right)_\mathrm{j} = \frac{u_\mathrm{j} - u_{\mathrm{j}-1}}{\Delta x} \tag{4}$$

Substitution of Equations (3,4) into Equation (1) together with $\Delta x = \frac{1}{n}$ leads to the following set of ODEs (Silebi and Schiesser 1992):

$$\left(\frac{du}{dt}\right)_\mathrm{j} = -nv\left(u_\mathrm{j} - u_{\mathrm{j}-1}\right), \; j = 1, n \tag{5}$$

$$\left(\frac{du}{dt}\right)_\mathrm{j} = \frac{-nv}{6}(2u_{\mathrm{j}+1} + 3u_\mathrm{j} - 6u_{\mathrm{j}-1} + u_{\mathrm{j}-2}), \; j = 2, ..., n-1 \tag{6}$$

THE HYBRID χ LANGUAGE

The hybrid models are specified in the hybrid χ language. The χ language has been designed from the start as a hybrid language that can be used for specification, verification (Kleijn et al. 1998), simulation and real-time control of discrete-event systems (van de Mortel-Fronczak et al. 1995), continuous-time systems and combined discrete-event / continuous-time systems (van Beek and Rooda 1998; van Beek et al. 1997). The language is based on mathematical concepts with well defined semantics (Bos and Kleijn 1999). The discrete-event part of χ is based on Communicating Sequential Processes, the continuous-time part on differential algebraic equations (DAEs). Processes are parametrized and can be grouped into systems; discrete and continuous channels are used for inter-process communication and synchronisation. High level data types are available such as arrays, lists and sets along with many associated operators.

The χ simulator is described by Naumoski and Alberts (1998), and Fábián (1999). It has been successfully applied to a large number of industrial cases, such as an integrated circuit manufacturing plant (Rulkens et al. 1998) and a beer brewery. In the sequel, only a minimal subset of the χ language is used. The syntax and operational semantics of the language constructs are explained in an informal way. Some language constructs are explained in this section, other constructs are explained when they are first used in the models. The models are kept as short as possible, showing only the essentials. This implies that the models are specified as stand-alone processes. In reality, models of delay elements have incoming and outgoing channels so that they can be used in systems. Such models, and an example case of a biochemical plant are described by Trienekens (1998).

A χ process may consist of a continuous-time part only (DAEs: differential algebraic equations), a discrete-event part only, or a combination of both.

```
proc name(parameter declarations) =
|[ variable declarations ; initialization
| DAEs | discrete-event statements
]|
```

All data types and variables are declared as either continuous using a double colon (e.g. $V :: [\mathrm{m}^3]$), or discrete using a single colon (e.g. d : real). The value of a discrete variable is determined by assignments (e.g. $d := 1$). Between two subsequent assignments the variable retains its value. The value of a continuous variable, on the other hand, is determined by equations. An assignment to a continuous variable (e.g. initialization $V ::= 0$) determines its value for the current point of time only. Some discrete data types are predefined like bool (boolean), int (integer) and real. Variables may be declared with units (e.g. $v :: [\mathrm{m/s}]$). All variables with units are of type real.

The continuous-time part consists of a set of DAEs that are separated by commas: $DAE_1, DAE_2, \ldots, DAE_n$. A time derivative is denoted by a prime character (e.g. x').

The discrete-event part consists of a sequence of statements. *Assignment* statements have been treated in the previous paragraphs. *Time passing* is denoted by Δt, where t is an expression of type real. A process executing this statement is blocked until the time is increased by t time-units. *Repetition* of statement $[S]$ is denoted by $*[S]$. By means of *state event* statement ∇r, the discrete-event part of a process can synchronize with the continuous part of a process. Execution of ∇r, where r is a relation involving at least one continuous variable, causes the process to be blocked until the relation becomes true.

HYBRID MODELS

Constant Time Delay

First a specification is given of a model with the same functionality as the carrousel described in Section 'Discrete-Event Models'. Between samples the output remains constant. Parameters of the model are the total delay time t_d and the number of samples n. The input and output variables of the delay element are represented by u_i and u_o, respectively. Declaration $u_\mathrm{i}, u_\mathrm{o} :: [-]$ means that both variables are continuous, but their units are not specified. Compare this with declaration $V :: [\mathrm{m}^3]$, where the units of V are m^3. Declaration *us* : real* declares *us* as a list of reals. The list is used to store the samples.

A list is an ordered collection of elements of the same type. For example [0, 1, 1, 2] is a list containing the numbers 0, 1, 1, and 2, in that order. Operator $+\!\!+$ concatenates two lists. For example, $[0] +\!\!+ [1, 1] = [0, 1, 1]$. Function hr (head right) returns the right most element of the list. For example, $\mathrm{hr}([0, 1, 1, 2]) = 2$. Function tr returns the list without the right most element. For example, $\mathrm{tr}([0, 1, 1, 2]) = [0, 1, 1]$. Using a list instead of an array leads to a much more elegant specification of the delay element.

```
proc $ConstDelay(t_\mathrm{d} : \mathrm{real},\ n : \mathrm{nat}) =$
|[ $u_\mathrm{i}, u_\mathrm{o} :: [-]$
, $us : \mathrm{real}^*,\ t_\mathrm{s} : \mathrm{real}$
; $t_\mathrm{s} := t_\mathrm{d}/n$
; $us := [\,];\ *[\ \mathrm{len}(us) < n \longrightarrow us := [0] +\!\!+ us\ ]$
| $u_\mathrm{i} = f(\tau)$
, $u_\mathrm{o} = \mathrm{hr}(us)$
| $*[\ us := [u_\mathrm{i}] +\!\!+ us;\ \Delta\, t_\mathrm{s};\ us := \mathrm{tr}(us)\ ]$
]|
```

First the sample time is initialized ($t_\mathrm{s} := t_\mathrm{d}/n$). Subsequently *us* is initialized as an empty list ($us := [\,]$), which is then filled with n zeros by $*[\ \mathrm{len}(us) < n \longrightarrow us := [0] +\!\!+ us\ ]$. This repetition is executed for as long as the condition $\mathrm{len}(us) < n$ is true. After the first bar (|) the equations are specified. The input function is given by f. Predefined variable τ represents the current simulation time. In models that are used as a part of a bigger system, input variable u_i would be defined in another process. Its value would then be made available in the model of the delay element by means of a continuous channel. Continuous output variable u_o is always equal to the last (right most) element of the list *us* ($u_\mathrm{o} = \mathrm{hr}(us)$). The discrete-event part of the specification consists of an endless loop. First, a new sample is added to the list ($us := [u_\mathrm{i}] +\!\!+ us$). Next, the model time is incremented to the next sample time point (Δt_s). During this sample interval, output variable u_o is equal to the right most element of the list. At the new sample time point, first the last element of the list is removed ($us := \mathrm{tr}(us)$), so that the before last sample is moved to the end of the list. Subsequently, the loop statements are repeated.

Constant Time Delay with Interpolation

The accuracy of the model can be improved considerably by using linear interpolation of the output between two samples. This has been done in the following model.

```
proc $ContDelay_2(t_\mathrm{d} : \mathrm{real},\ n : \mathrm{nat}) =$
|[ $u_\mathrm{i}, u_\mathrm{o} :: [-]$
, $us : \mathrm{real}^*,\ t_\mathrm{s}, d : \mathrm{real}$
; $t_\mathrm{s} := t_\mathrm{d}/n$
; $us := [\,];\ *[\ \mathrm{len}(us) < n \longrightarrow us := [0] +\!\!+ us\ ]$
; $u_\mathrm{o} ::= \mathrm{hr}(us);\ us := \mathrm{tr}(us);\ d := 0$
| $u_\mathrm{i} = f(\tau)$
, $u_\mathrm{o}' = d$
| $*[\ d := \frac{\mathrm{hr}(us) - u_\mathrm{o}}{t_\mathrm{s}};\ us := [u_\mathrm{i}] +\!\!+ us$
    ; $\Delta\, t_\mathrm{s};\ us := \mathrm{tr}(us)$
    ]
]|
```

Output equation $u_\mathrm{o} = \mathrm{hr}(us)$ has been replaced by equation $u_\mathrm{o}' = d$, where d is a discrete variable that is equal to the derivative of the output between samples. The output variable is initialized to the last element of the list ($u_\mathrm{o} := \mathrm{hr}(us)$). Immediately after that this last element is removed from the list ($us := \mathrm{tr}(us)$). The value of d is set to $\frac{\mathrm{hr}(us) - u_\mathrm{o}}{t_\mathrm{s}}$ at the beginning of the repetition. At that time point, the value of the output variable equals u_o. At the next sample time point, the output variable should be reach the value of the last sample in the list, which is equal to $\text{hr}(us)$. This change takes t_s time units. Therefore, the derivative of the output variable should equal $\frac{\text{hr}(us)-u_o}{t_s}$.

Variable Time Delay

The key to an accurate, stable and elegant model of a time delay element is not to keep the *sample time* constant, but instead to keep the *transported volume* constant in every sample interval. In this way, the output value of the delay element model at the sample times is almost exact. The error in the output values at these sample times depends only on the accuracy of the DAE or ODE solver used. The model follows below.

```
proc $VarDelay(V_{tot} : \text{real},\ n : \text{nat}) =$
$[\![\ u_i, u_o, Q, V :: [-]$
$,\ us : \text{real}^*,\ V_s : \text{real}$
$;\ V_s := V_{tot}/n$
$;\ us := [\,];\ *[\ \text{len}(us) < n \longrightarrow us := [0] +\!\!+\ us\ ]$
$;\ u_o ::= \text{hr}(us);\ us := \text{tr}(us);\ V ::= 0$
$|\ \ u_i = f(\tau)$
$,\ \ Q = g(\tau)$
$,\ \ V' = Q$
$,\ \ u'_o = \frac{\text{hr}(us)-u_o}{(V_s-V)/Q}$
$|\ *[\ us := [u_i] +\!\!+\ us$
    $;\ V ::= 0;\ \nabla\ V \ge V_s;\ us := \text{tr}(us)$
    $]$
$]\!]$
```

Two additional continuous variables have been introduced: Q and V. The delay element is considered as a pipe with a constant area. Variable Q represents the flow in the pipe; variable V represents the volume that enters the pipe during a sample interval ($V' = Q$). The sample volume V_s equals the total volume of the pipe V_{tot} divided by the number of samples in the list n. Flow Q changes according to function g ($Q = g(\tau)$). The discrete-event specification is analogous to that of the model *ContDelay*$_2$. Discrete variable d is no longer needed, and time passing during the sample interval is now modelled by $V ::= 0;\ \nabla V \ge V_s$ instead of Δt_s. First the volume is initialized to 0, then time passes until the volume V, that has entered into the pipe, equals the sample volume V_s. This means that after n samples, the amount of liquid that has entered into the pipe equals $V_s \cdot n$, which in turn equals the total volume of the pipe V_{tot}. Therefore, the liquid particles that are at the beginning of the pipe arrive at the end of the pipe after exactly n samples. The output is interpolated in a way analogous to model *ContDelay*$_2$. The derivative u'_o of the output is approximated as $\frac{\Delta u_o}{\Delta t}$. At any time point during the sample interval, the output equals u_o. The value at the end of the interval should be $\text{hr}(us)$. Therefore $\Delta u_o = \text{hr}(us) - u_o$. Likewise, the transported volume at any time point during the sample interval equals V. Therefore, the remaining time Δt required to transport the sample volume V_s equals $(V_s - V)/Q$, so that $\frac{\Delta u_o}{\Delta t} = \frac{\text{hr}(us)-u_o}{(V_s-V)/Q}$. In this equation, $\text{hr}(us)$ and V_s remain constant during the sample interval, but u_o, V and Q are functions of time.

SIMULATION RESULTS

The models *ContDelay*$_2$, *VarDelay*, and *FD4UB* have been compared with a step and sine input. The *FD4UB* model is a χ model that implements Equations (5,6). The average absolute error has been determined by integration

$$\bar{e} = \int_0^{t_0+t_d+t_m} \frac{|u_o(t) - f(t-t_d)|}{t_m}\,dt$$

where $u_o(t)$ is the actual output; $f(t)$ is the input ($f(t) = 0$ for $t < 0$); $t_d = 10$ is the delay time, which is the same for all models; and $t_m = t_d = 10$ for the step input, and $t_m = 24$ equals the period of the sine for the sine input. The value of t_0 is 1 for the step input and 0 for the sine input. The errors for models *ContDelay*$_2$ and *VarDelay* depend on the exact value of t_0. The error is maximal when t_0 coincides with a sample point, and minimal when it is exactly in the middle of two adjacent sample points. Results presented for model *VarDelay* have been limited to constant delay times, because of the difficulty to produce reliable analytical reference results for the output with varying delay times. The solver used is the differential algebraic equation solver DASSL (Petzold 1983). The current version of the χ simulator does not yet provide any ODE solvers.

Table 1: Step input comparison.

	n	$\bar{e}$	t_{comp}[s]
ContDelay$_2$	50	0.005 - 0.01	0.9
	100	0.0025 - 0.005	1.9
VarDelay	50	0.005 - 0.01	2.6
	100	0.0025 - 0.005	4.8
FD4UB	50	0.036	3.7
	100	0.021	11.5

Table 2: Sine input comparison.

	n	$\bar{e}$	t_{comp}
ContDelay$_2$	50	0.00014	1.9
	100	0.00003	3.7
VarDelay	50	0.00014	2.9
	100	0.00003	5.6
FD4UB	50	0.0025	2.8
	100	0.00070	7.8

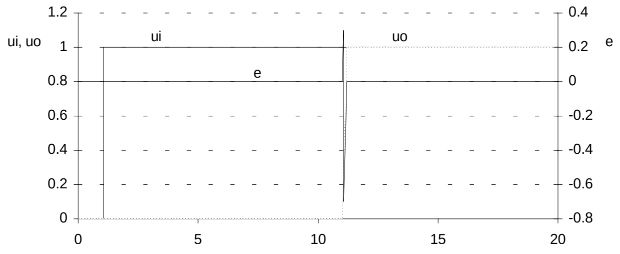

Figure 2: Model *ContDelay*$_2$, step input, $n = 50$

CONCLUSIONS

Hybrid models of time delay elements have been shown to outperform traditional models, both on accuracy, com-

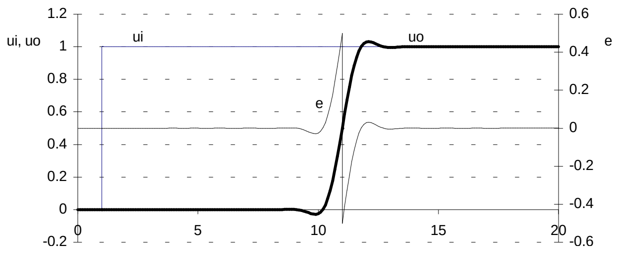

Figure 3: Model *FD4UB*, step input, $n = 50$.

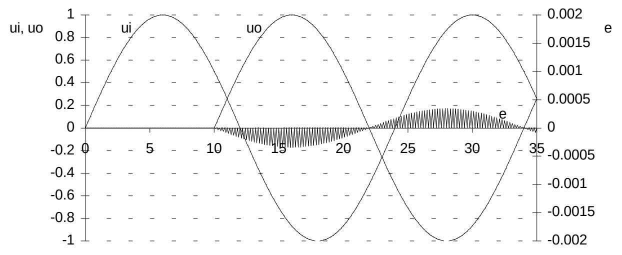

Figure 4: Model *ContDelay*$_2$, sine input, $n = 50$.

putation time, and stability. At the same time, the hybrid models are short and elegant. Simulations have been done using the χ simulator with the DASSL differential algebraic equation solver. The key to an accurate, stable and elegant model of a time-delay element with a variable delay time is not to keep the *sample time* constant, but instead to keep the *transported volume* constant in every sample interval.

REFERENCES

Ball, S.J. 1966. A Variable Time-Delay Subroutine for Digital Simulation Programs. *Simulation 22*, 23–24.

Bos, V. and J.J.T. Kleijn. 1999. Structured Operational Semantics of Chi. Computing Science Reports 99/01, Eindhoven University of Technology, Eindhoven.

Carver, B. and H.W. Hinds. 1978. The Method of Lines and the Advective Equation. *Simulation 31*, 59–69.

Coleman, T.G. 1965. A Time-Delay ‘Special Element’ for PACTOLUS. *Simulation 5*(11), 308.

Fábián, G. 1999. *A Language and Simulator for Hybrid Systems*. Ph. D. thesis, Eindhoven University of Technology, The Netherlands.

Franks, R.G.E. 1972. *Modelling and Simulation in Chemical Engineering*. London: Wiley-Interscience.

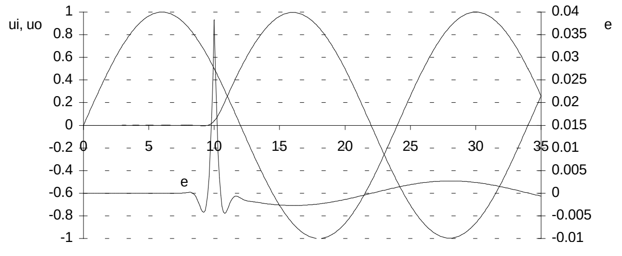

Figure 5: Model *FD4UB*, sine input, $n = 50$.

Hu, S.S. and X.M. Shao. 1988. Adaptive Hybridized Spline Differentiators for Numerical Solution of the Advective Equation. *Computers & Mathematics with Applications 15*, 525–534.

Kleijn, J.J.T.; M.A. Reniers; and J.E. Rooda. 1998. A Process Algebra Based Verification of a Production System. In *Proc. of the 2nd IEEE International Conference on Formal Engineering Methods (ICFEM’98)*, Brisbane, 90–99.

Kuru, S. 1981. Ph. D. thesis, Carnegie-Mellon University.

Metzger, M. 1994. Modelling and Simulation of Time-Delay Element via Advective Equation. In *Proc. Conference on Modelling and Simulation*, 756–759.

Metzger, M. 1996. Comparison of Finite-Difference and Orthogonal Collocation Methods in Simulations of Sampled-Data-Controlled Continuous Plants with Time-Delay. In *Proc. 1996 European Simulation Multiconference*, 118–121.

Morton, W. and G.J. Smith. 1989. Time Delays in Dynamic Simulation. *Computers & Chemical Engineering 13*(6), 631–640.

Naumoski, G. and W. Alberts. 1998. *A Discrete-Event Simulator for Systems Engineering*. Ph. D. thesis, Eindhoven University of Technology, The Netherlands.

Petzold, L.R. 1983. A Description of DASSL: A Differential/Algebraic System Solver. *Scientific Computing*, 65–68.

Rulkens, H.J.A.; E.J.J. van Campen; J. van Herk; and J.E. Rooda. 1998. Batch size optimization of a furnace and pre-clean area by using dynamic simulations. In *Proc. SEMI/IEEE Advanced Semiconductor Manufacturing Conference*, Boston, 439–444.

Silebi, C.A. and W.E. Schiesser. 1992. *Dynamic Modelling of Transport Process Systems*. San Diego: Academic Press.

Trienekens, B.J. 1998. Hybrid Modelling of Continuous Delay Elements. Master’s thesis, Eindhoven University of Technology, Department of Mechanical Engineering, The Netherlands.

van Beek, D.A.; S.H.F. Gordijn; and J.E. Rooda. 1997. Integrating Continuous-Time and Discrete-Event Concepts in Modelling and Simulation of Manufacturing Machines. *Simulation Practice and Theory 5*, 653–669.

van Beek, D.A. and J.E. Rooda. 1998. Languages and Applications in Hybrid Modelling: Positioning of Chi. In *Proc. 9th Symposium on Information Control in Manufacturing*, Nancy, 77–82.

van de Mortel-Fronczak, J.M.; J.E. Rooda; and N.J.M. van den Nieuwelaar. 1995. Specification of a Flexible Manufacturing System Using Concurrent Programming. *Concurrent Engineering: Research and Applications 3*(3), 187–194.