# The problem with time
# in mixed continuous/discrete time modelling

Kenneth C. Rovers
K.C.Rovers@utwente.nl

Jan Kuper
J.Kuper@utwente.nl

Gerard J.M. Smit
G.J.M.Smit@utwente.nl

Computer Architecture for Embedded Systems group*
CTIT, Department of EEMCS, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

## ABSTRACT

The design of cyber-physical systems requires the use of mixed continuous time and discrete time models. Current modelling tools have problems with time transformations (such as a time delay) or multi-rate systems.

We will present a novel approach that implements signals as functions of time, directly corresponding to their mathematical representation. This enables an exact implementation of time transformations and as an additional advantage enables local control over time. A representation of components and signals in both domains is provided, together with composition operators to allow the specification of signal flow diagrams.

## 1. INTRODUCTION

In designing cyber-physical systems, i.e. embedded systems which include peripherals interfacing with the physical world, a modelling tool for simulation is essential. Such a tool must support both the *continuous time* (CT) and *discrete time* (DT) domain, because of the tight integration of the environment and the systems's computational and physical facets [7].

Considering time is such an import aspect, it might be surprising that current modelling tools have a problem with time. This problem occurs in systems with time delays or multi-rate systems for example. The problem is that a single (global) time step is chosen by the solver for simulation, at which the whole system under design is evaluated. This time step must meet the requirements of *all* components in the system. Several components, such as integration, may require a very small time step to achieve enough accuracy, a resolution not needed for the rest of the system. Further, CT signals are implemented as a sequence of discrete values at simulation time steps. A time delay component therefore buffers values and interpolates between available values introducing inaccuracies caused by the modelling tool.

In this paper we propose a novel approach to mixed CT/DT modelling. By implementing CT signals as *functions of time*, time delays or multi-rate systems can be implemented *exactly* without loosing efficiency. Of course, considering a signal as a function of time is not new, but *implementing* them as such is. We will set the scope in section 2 and analyse the problem in more detail in section 3. Next, we will identify the many faces of time (section 4) and present our solution (section 5) followed by some results (section 6).

## 2. ABOUT TIME

The scope of this paper involves the modelling of mixed CT and DT systems. In such systems signals represents measurable quantities over an independent value, in our case time. So it concerns time.

We differentiate between continuous time and discrete time. Continuous time is unbroken or whole, i.e. defined for all time. Discrete time quantises time to distinct separate moments in time. Thus there are two kinds of signals:

- Signals in the CT domain are functions of time, i.e. they represent the value of the signal over all time.

- Signals in the DT domain are a list of values at discrete moments, also called samples. However, computationally an operation is defined on an individual value (possibly depending on previous values using state).

A function that takes a signal and transform it to a new signal is called a *signal transformation*. In case of DT signals, the implementation of a signal transformation corresponds to an operation or function on a value. In case of CT, a signal transformation corresponds to a function on a function, i.e. it is a so called *higher order function*.

A signal transformation can be with respect to the independent value, i.e. a time-shift corresponds to a delay of the signal. Other transformations are reflection or scaling.

Signal flow diagrams or block diagrams are popular in system design tools because of their intuitive use and ease of understanding [2]. Components in such diagrams denote signal transformations and arrows denote signals. Composition of signal transformations is analogous to connection blocks in a signal flow diagram.

To compose components from different domains, the signal representation must be changed. To go from the CT domain to the DT time domain, the signal is sampled at specific sample times by an analogue-to-digital converter (ADC). To go from the DT domain to the CT domain, the sample is hold until the next value by a digital-to-analogue converter (DAC).

## 3. PROBLEM ANALYSIS

There are many mixed continuous/discrete time modelling tools [2]. Probably the most well-known is Simulink. Existing tools perform a simulation by extracting a set of ordinary differential equations (ODEs) from the model. This set is (in the general case) solved numerically. Typical solvers are the Euler methods or the Runge-Kutta methods. Such solvers operate iteratively with a fixed or variable step size.
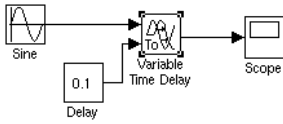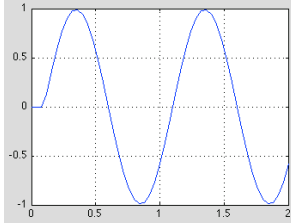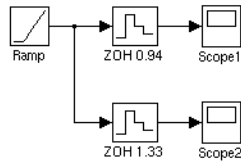
Figure 1: Simulink example



Figure 2: Model with multiple ADCs
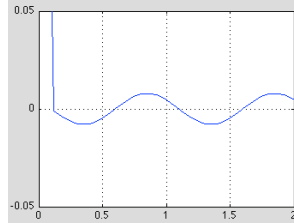


Figure 3: Delayed sine



Figure 4: Delay error

Thus, simulation for existing tools consists of quantising time in time steps to iteratively solve the set ODEs. This means time is a global property of the model. For a mixed CT/DT model, the DT is represented as piecewise-continuous for the solver. At each simulation step the complete model is evaluated and a resulting value is calculated. So, the inputs and outputs of the model components are the values at a certain time step.

The solver must determine the time instants to solve the set of equations. This is problematic in case of time transformations or incompatible components.

For the first problem, consider the very simple Simulink system in figure 1 consisting of a sine wave source, a (variable) time delay and a scope. A 1Hz sine with a 0° initial phase is used. The (maximum) time step size for simulation is 0.04s, so 25 simulation results per period. The step size is not adjusted by Simulink, because the model contains no differential equations (only algebraic). The time delay is 0.1s, which is deliberately chosen not to fit the step size.

A plot of the output is shown in figure 3, a shifted sine wave as expected. The ideal result is a sine wave with an initial phase of $-2 \cdot \pi \cdot 0.1$, but when compared to the output of the delayed sine wave of figure 1 the Simulink output has an error (shown in figure 4). This error is caused by interpolation. The time delay block buffers values each time step and retrieves a value for the delayed time. When a value at the delayed time is not available, the result is interpolated from the surrounding values. The error is directly related to the frequency of the signal and the step size. Higher frequencies need smaller steps or will give larger errors.

One might expect that a solution is to make the time step a *multiple of the delay* as the delay block can then retrieve the *exact* value. There are at least three problems with this:

a) if the delay is small the step size needs to be small, resulting in many (fixed) steps and thus inefficiency,

b) if multiple time delays are used, without a common factor, separate time steps for each time delay are needed for accurate results,

c) if the delay is variable, the current time step depends on a unknown delay in the future.

In general the time delay *can* be variable. Simulink completely ignores the delays and only uses the solver to determine the time step to use because of these problems.

A second example of a problematic model is shown in figure 2. An ADC, implemented as a zero-order-hold (ZOH), *is* assumed to have a fixed sample rate (otherwise a sample-and-hold should be used). The time steps are indeed determined by the sample rate for both fixed and variable step sizes in Simulink. For a multi-rate system with more ADCs the time steps match the sample times of *all* ADCs with a variable step. However, a fixed step size results in a very small time step, matching the common denominator. Already for more than two ADCs the time step becomes too small and generates an error in Simulink.

A third problem is that the time step determined by the solver for numerical approximation of a differential equation is applied to all equations. This global time step causes the whole system to be evaluated, while it is very well possible that most of the system does not need to be evaluated at this fine-granularity, reducing efficiency; for example, the DT domain with a sample period much larger than the approximation step.

## 4. ALL THE TIME

We have observed that using a global time step set by the solver results in problems with efficiency or with accuracy. In fact, we have seen different notions of time:

- the time step when observing signals during simulation
- the period when sampling a CT signal by an ADC,
- the time steps when approximating a differential equation, of an integral block for example,
- the delay when shifting a signal by a time delay block.

These different notions of time (steps) are in principle unrelated, however they are coalesced into a single time step by the solver. The notions, including the approximation time step of the solver are *actually* a local property in the model. In this paper we propose a solution that enables local control over the time, i.e. by locally applying time transformations or time steps the problems described in section 3 can be resolved while retaining efficiency.

## 5. FORMALISM

We will show the problems mentioned above are resolved if a local time reference is used by applying operations on functions of time instead of values.

### 5.1 Continuous time
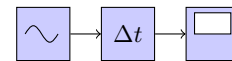
Consider the same system as in figure 1:



Figure 5: Simple system block diagram

As explained in section 2, signals are functions of time and components in this model represent signal transformations. Thus, the mathematical definition of the relevant types is:

$$Time = \mathbb{R}$$
$$SigCT = Time \to \mathbb{R}$$
$$ComponentCT = SigCT \to SigCT$$

where $A \to B$ denotes the class of functions from $A$ to $B$.

The first component is a sine *source* and as such does not really "transform" a signal. That is, it transforms a vacuous input:

$$source() = t \mapsto a \cdot sin(\omega t)$$

where $a$ is the amplitude, $\omega$ the frequency and $t$ is time. The notation $t \mapsto ..t..$ denotes the function which maps $t$ to $..t..$.

The next component is a time delay. The delay can have any value and even be variable. Existing modelling tools have problems with a time delay because also the CT signals are discretised for simulation. Mathematically, however, a time delay component is simply defined as:

$$delay_\delta(f) = t \mapsto f(t - \delta) \tag{1}$$

where $f$ is a signal, i.e. a function of time, and *delay* adjusts the time of $f$ with $\delta$. Thus, to find the function value on time $t$ *after* a 0.1 time delay, one has to know the function value at time $t - 0.1$.

The final component is a scope *sink*. The scope plots the signal, hence, as a transformation it may be rather meaningless. That is, the input signal is transformed to a vacuous output, with a plot of the signal as a side-effect:

$$sink(f) = t \mapsto ()$$

Now suppose a 1Hz sine with amplitude 1 and a 0.1 time delay. The signal that is plotted is then:

$$delay_{0.1}(sin(2\pi t)) = t \mapsto sin(2\pi(t - 0.1))$$

The time delay is accurately included in the final function, independent from the time used for simulation. In Simulink and other existing tools it is exactly at this point that inaccuracies are introduced.

## 5.2 Discrete time

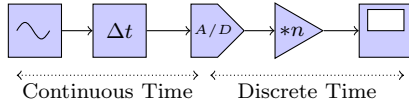We extend the system with an ADC and a signal gain to include the DT domain:

Continuous Time        Discrete Time

**Figure 6: Mixed CT/DT system block diagram**

As mentioned, signals in the DT domain represent values on discrete moments, but from a computational view a component operates on a single value:

$$SigDT = \mathbb{R}$$
$$ComponentDT = SigDT \to SigDT$$

The ADC component transforms the signal $f$ into a discretized signal with time interval $d$. This is achieved by flooring the time of the CT input to the latest sample time:

$$adc_d(f) = t \mapsto f(\lfloor t/d \rfloor \cdot d) \tag{2}$$

Applying this result to a time $t$ gives the latest value that was sampled before $t$.

The $gain_n$ transformation multiplies a signal value by $n$:

$$gain_n(x) = x * n \tag{3}$$

However, the output of the ADC gives the latest sample for time $t$. The $gain_n$ component must therefore be changed

so it operates on the ADC samples independent of time, i.e. the input signal is applied to time $t$, but not $gain_n$ itself:

$$\widehat{gain}_n(f) = t \mapsto f(t) * n$$

The operation $\widehat{\phantom{n}}$ is is called "lifting" from a function on values to a function on functions, and is automatically performed when a DT component is connected to a CT signal. The result for plotting is then a piecewise horizontal-function. This is implemented efficiently by re-using the results from the latest sample in between sample times.

## 5.3 Composition

A signal flow diagram is a *composition* of signal transformations. We will define operators for sequential, parallel and feedback composition.

Sequential composition is expressed as:

$$\varphi \triangleright \psi = f \mapsto \psi(\varphi(f)) \tag{4}$$

where $\varphi$ and $\psi$ are transformations, i.e., components, and $\triangleright$ is the operation to compose transformations. That is, $\varphi \triangleright \psi$ is the transformation that takes a signal function $f$ as an argument and determines its result by first applying $\varphi$ to $f$ and then $\psi$ to the resulting signal function.

Assume the same source and delay as before, an ADC sample period of 0.3 and a gain of 2. The diagram in figure 6 can then be expressed as:

$$source \triangleright delay_{0.1} \triangleright adc_{0.3} \triangleright gain_2 \triangleright sink$$
$$= gain_2 \, (adc_{0.3} \, (delay_{0.1} \, (sin(2\pi t))))$$
$$= t \mapsto 2 * sin(\lfloor (t - 0.1)/0.3 \rfloor * 0.3)$$

Likewise, parallel composition (figure 7) is defined as:

$$\varphi \parallel \psi = (f, g) \mapsto (\varphi(f), \psi(g)) \tag{5}$$

i.e. multiple inputs are represented as tuples and the composition connects $\varphi$ to the first and $\psi$ to the second.

Feedback composition (figure 8) is defined as:

$$\circlearrowright \varphi = f \mapsto g \text{ where } (g, h) = \varphi(f, h) \tag{6}$$

i.e. the component $\varphi$ takes two inputs $f$ and $h$, of which $h$ is the second output signal of itself.

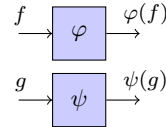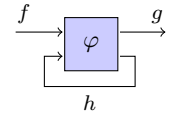**Figure 7: Parallel**

**Figure 8: Feedback**

## 5.4 Implementation

A key requirement for implementation is support for higher-order functions to express signal transformations. This is supported by functional languages, which have the additional advantage of directly expressing the mathematics. We choose for the functional programming language Haskell, because it also provides a *type class* feature to conveniently overload algebraic and composition operators.

The *delay*, *adc* and *gain* components are straightforward reformulations of equation (1), (2) and (3) as the reader may check immediately:

```
delay delta f  =  \t -> f (t-delta)
adc   d     f  =  \t -> f (floor (t/d) * d)
gain  n     x  =  x * n
```

The composition operators ▷, ‖ and ↻ of equation (4), (5) and (6) are written in Haskell as `>>>`, `||` and `loop` and are also in direct correspondence to their mathematical definition:

```
phi >>> psi = \f       -> psi (phi f)
phi ||  psi = \(f, g) -> (phi f, psi g)
loop phi    = \f       -> let (g, h) = phi (f, h) in g
```

Note that we require lazy evaluation to allow `phi` to break the recursive dependence on `h` in `loop`.

The signal flow diagram of figure 6 is then expressed as:

```
source >>> delay 0.1 >>> adc 0.3 >>> gain 2 >>> sink
```

## 6. TIME WILL TELL

To verify our solution and illustrate its usefulness, we have implemented a phased array system (used in e.g. radar and radio astronomy) and compared it with an implementation in Simulink. A phased array receiver consists of an array of antennas used to electronically steer the reception. The path length from a transmitter to each antenna element is different, resulting in a different time delay for each antenna signal. To validate an adaptive steering algorithm [1] we have developed, the modelled time delays must be *exact*.

Our implementation is used to steer the receiver into the direction of a signal-of-interest, while a second signal is interfering. The second source is rejected and the resulting signal is exactly as expected (mathematically). The same system in Simulink has an error in the range of the step size. The simulation takes about 0.93s for both tools on a 2GHz Core2 Duo with 4GB RAM when interpreted. For the compiled versions, Haskell is about 10 times faster.

If the antenna elements are moving with respect to the source, they experience the Doppler effect. For antenna elements with different speeds, for example on a moving ship, this effect can not be represented by a simple frequency shift. Instead time-scaling per channel must be applied. This is implemented analogous to the time delay.

## 7. RELATED WORK

An extensive survey of mixed domain modelling tools can be found in [2]. Simulink is the standard for mixed CT/DT system modelling. Also widely known is Ptolemy [3], which supports many more domains with the goal of researching their interaction. Ptolemy also support higher order components, but not higher order signals. SystemC-AMS extends SystemC for mixed signal modelling [8], adding support for signal flow models and conservative-law models. Modelica [5] specifies a object-oriented, declarative, multi-domain modelling language of which there are a number of implementations. Surprising as it may seem, all these tools solve a set of differential equations using a global time step and pass values. Although Ptolemy for example has a notion of super-dense time, this is still a time (plus index) and value pair. Some tools can locally influence the step size, Ptolemy components can reject a step size until all agree and SystemC-AMS modules propagate the time step for consistency, but it remains a global parameter. None of the tools researched [2, 5, 8, 9] separate the notions of time and allow local transformations using the higher abstraction of signals as function of time, as proposed in the present paper.

Functional Reactive Programming (FRP) [4, 6] also uses functions of time (originally behaviours) in Haskell, but for different domains. FRP also does not identify and use different notions of time nor apply it to signal flow diagrams.

## 8. CONCLUSION

The problem with time is tackled by analysing the meaning of signals and time in the CT and DT domains and realising that CT signals are functions of time and should be implemented as such. A further contribution is the realisation that time is a local property, which can be exploited *because* CT signals are functions of time. Therefore, the value of a time delayed or sampled signal can be determined *exactly* without reducing efficiency.

We have developed a mathematical representation and implementation of signals and components in the CT and DT domain, as well as their composition. The results indicate at least a comparable performance to Simulink while providing an exact simulation.

## 9. IN TIME

Future work will be to extend into the direction of domains such as dataflow. Furthermore, we were not able to adequately discuss stateful computations and feedback in this paper.

## 10. REFERENCES

[1] K. C. H. Blom, M. D. van de Burgwal, K. C. Rovers, A. B. J. Kokkeler, and G. J. M. Smit. DVB-S signal tracking techniques for mobile phased arrays. In *IEEE 72nd Vehicular Technology Conference (VTC 2010-Fall)*, page 5, Ottawa, ON, Canada, Sep 2010.

[2] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Angiovanni-Vincentelli. Languages and Tools for Hybrid Systems Design. *Foundations and Trends in Electronic Design Automation*, 1(1):1–193, 2006.

[3] J. Eker, J. Janneck, E. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity - the Ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, Jan 2003.

[4] C. Elliott and P. Hudak. Functional Reactive Animation. In *ICFP '97*, pages 263–273. ACM, 1997.

[5] P. Fritzson and V. Engelson. Modelica - A Unified Object-Oriented Language for System Modeling and Simulation. In E. Jul, editor, *ECOOP'98 — Object-Oriented Programming*, volume 1445 of *Lecture Notes in Computer Science*. 1998.

[6] P. Hudak, A. Courtney, H. Nilsson, and J. Peterson. Arrows, robots, and functional reactive programming. In *Advanced Functional Programming*, Lecture Notes in Computer Science. 2003.

[7] E. A. Lee. Cyber physical systems: Design challenges. Technical Report UCB/EECS-2008-8, EECS Department, University of California, Berkeley.

[8] A. Vachoux, C. Grimm, and K. Einwich. SystemC-AMS Requirements, Design Objectives and Rationale. In *Proceedings of the conference on Design, Automation and Test in Europe (DATE '03)*. IEEE, 2003.

[9] H. Zheng. *Operational semantics of hybrid systems*. PhD thesis, Berkeley, CA, USA, 2007.