Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

TOWARDS A UNIVERSAL FORMALISM FOR MODELING & SIMULATION

Fernando J. Barros

University of Coimbra
Department of Informatics Engineering
3030 Coimbra, PORTUGAL

ABSTRACT

The representation of hybrid systems has shown to be one of the greatest challenges in Modeling & simulation. While discrete event systems can be represented without error, continuous models rely on approximations based in numerical methods. Given the large variety of numerical integrators a unified representation has been elusive due the lack of an universal formalism that can describe all numerical methods and to provide their seamless integration. In this paper, we propose the Hybrid Flow System Specification (HYFLOW) as a unifying representation for different families of numerical integrators for solving ordinary differential equations (ODEs). HYFLOW combines the conventional discrete event approach with a novel representation based on sampling and the support for dense outputs to describe modular and hierarchical hybrid systems. We demonstrate that HYFLOW can describe 1st-order, geometric (2nd-order), and exponential integrators. Additionally, since these integrators share the same underlaying HYFLOW representation they can be seamlessly combined.

1 INTRODUCTION

The representation of hybrid systems has proven to be one of the greatest challenges in Modeling & simulation. While discrete event systems can be represented without error in digital computers, the representation of continuous models has been elusive, since an exact description cannot, in general, be achieved in these computers. In fact, continuous systems can only be exactly represented in analog computers, but these machines are virtually extinct nowadays. Practical representations involve the use of numerical methods, like ODE (ordinary differential equations) integrators but unfortunately no single method can produce an adequate description for all types of continuous systems. In fact, there is a plethora of numerical methods, each one with its strengths and weaknesses, that provide only an approximate representation. A formalism able to provide a unifying representation of all these methods and capable to build a bridge with discrete event systems modeling is currently a big research challenge.

The quest for a unifying formalism requires the identification of the basic constructs that can support the representation of continuous signals in a digital computer. While discrete event systems have long been formalized (Zeigler 1976), the set of constructs able to represent continuous systems has been elusive.

In this paper we propose the concept of generalized sampling and the support for dense outputs as the key constructs for achieving a unifying representation for some of the numerical methods used to describe continuous systems. Sampling and dense outputs constructs are formalized in the Hybrid Flow System Specification (HYFLOW) a formalism for representing hybrid systems. In particular we show that HYFLOW provides a framework for describing several families of numerical integrators, including polynomial, exponential, and geometric (2nd-order) integrators. HYFLOW also guarantees the seamless combination of these numerical methods, enabling the representation of complex systems that require the composition of different integrators for achieving a better description.

Although sampling has long been used in control and signal processing areas, common approaches are mainly synchronous (Benveniste and Berry 1991), making them incompetent to provide a representation of continuous systems based on the more efficient adaptive size integrators.

The representation of continuous systems has traditionally been made through idealized formalisms based on ODEs that abstract the numerical integrators required to solve the ODEs on a digital computer (Henzinger 1996), (Praehofer 1991). These approaches, while hiding simulation details, also prevent the possibility to describe how the different numerical methods can be combined, offering no contribution to model interoperability, in particular when different integrators are needed.

In previous work we have shown that HYFLOW provides a unifying representation for digital controllers (Barros 2005b), digital filters (Barros 2005a), fluid stochastic Petri Nets (Barros 2015b), and their seamless integration with other HYFLOW models. The numerical methods presented in this paper point to the generality of HYFLOW and to its ability to support hybrid modular models that can be simulated as independent units requiring only the information provided in model interface. This modular design is fundamental, for example, in co-simulation (Schierz et al. 2006) and for achieving a better computational performance (Skelboe 2006).

2 RELATED WORK

The use of explicit representations of 1st-order numerical integrators was introduced in (Zeigler and Lee 1998). This work is based on discrete event systems and it relies on the representation of continuous signals through the use of polynomials (Migoni et al. 2013). In this approach, the output of a component is pushed to its peers using the coefficients of a polynomial that encodes a continuous output signal. This work has key differences when compared with HYFLOW sampling-based approach. Since discrete events are instantaneous, polynomial coefficients need to be locally stored for representing a continuous segment. Components need also to keep a list of the incoming polynomials, and when a new input (polynomial) is received it will replace the corresponding outdated element. Besides the additional complexity required for input management, this procedure severely limits hierarchal and modular design. Each component needs to know its influencers making it dependent on the context. In particular this require that a component needs to be adapted when inserted into a new context. For example, if the component is designed to work with a single input, it needs to be redefined to work with two inputs, making it harder to reuse. This contrasts with HYFLOW that through sampling gets a value from all inputs at once. Next these input values become combined by HYFLOW component input function, making the component itself reusable in any context, since it can always receive a single value, irrespective to the network topology. In an extreme case, a HYFLOW component can be used, without modification, in a dynamic topology network, while a polynomial-based approach would make this support very difficult to achieve.

Discrete events also jeopardize the use of hierarchical models. Since polynomials cannot be combined, coefficients need to be kept in lists. This kind of representation exposes the internal components of a network model, making any component that receives such a list aware and dependent of the internals details of a network, severely limiting model hierarchical design. Another limitation involving polynomial representations is that they provide by no means the best approximation to all kind of continuous functions. Other approximations, like rational fractions, trigonometric functions (Neta and Ford 1984), or exponentials used in Section 5 can, in many cases, provide a more accurate representation.

Another limitation is linked to the discrete event nature of the quantized approaches. While 1st-order polynomial integrators can adjust the step size based on the local error, the geometric integrators described in Section 6 do not admit this error based control of step size in order to keep their structural properties. In fact, geometric integrators are mainly fixed step size methods. Modify the step size requires different assumptions that are not compatible with error-base control principles. Geometric integrators can adjust their step size based on kinetic energy (Hairer 1997), and they cannot accept interruptions imposed by external discrete events bringing, for example, polynomials coefficients.

The most common approaches to continuous modeling rely on idealized representations like ODEs (Henzinger 1996), (Praehofer 1991), (Lynch et al. 2003). However, these approaches do not provide any light on how to describe integrators with asynchronous step sizes, or if, for example, geometric integrators need be used for achieving a correct solution. As in many other cases, the devil is in the details, and idealized solutions are not adequate for developing working M&S environments where numerical methods can be described. In our opinion a sound M&S formalism should be able to provide a rigorous operational semantics so the inner details of the numerical methods can be defined without any ambiguity.

3 THE HYFLOW FORMALISM

The Heterogeneous Flow System Specification (HYFLOW) is a formalism created to represent hybrid modular systems (Barros 2003). HYFLOW defines two types of models: basic and network. Basic models provide state representation and state transition functions. Network models are a composition of basic models and/or other network models. To the best of our knowledge HYFLOW was the first hybrid formalism to use a HyperReal time-base (Barros 2008). Additionally, the HYFLOW formalism merges discrete event system with sampling-based systems that have been traditionally represented by a real time base and a discrete time base, respectively (Zeigler 1976) and (Manna and Pnueli 1993). The HYFLOW formalism proposes a hyperreal time base in order to seamlessly unify these two paradigms.

3.1 HYFLOW Basic Model

We describe HYFLOW basic models and their semantics. Let $\widehat{B}$ be the set of all names corresponding to HYFLOW basic models. A model associated with name $B \in \widehat{B}$ is defined by:

$$M_B = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda),$$

where

$X = \bar{X} \times \check{X}$ is the set of input flow values,
$\bar{X}$ is the set of continuous input flow values,
$\check{X}$ is the set of discrete input flow values,

$Y = \bar{Y} \times \check{Y}$ is the set of output flow values,
$\bar{Y}$ is the set of continuous output flow values,
$\check{Y}$ is the set of discrete output flow values,

P is the set of partial states (p-states),

$\rho : P \longrightarrow \mathbb{H}^0_{+\infty}$ is the time-to-input function,

$\omega : P \longrightarrow \mathbb{H}^0_{+\infty}$ is the time-to-output function,

$S = \{(p,e) | p \in P, 0 \leq e \leq \nu(p)\}$ is the state set,
with $\nu(p) = \min\{\rho(p), \omega(p)\}$, representing the time to transition function,

$\delta : S \times X^{\varnothing} \longrightarrow P$ is the transition function,
where $X^{\varnothing} = \bar{X} \times (\check{X} \cup \{\varnothing\})$,
and $\varnothing$ represents the null value (absence of value),

$\bar{\Lambda} : S \longrightarrow \bar{Y}$ is the continuous output function,

$\lambda : P \longrightarrow \check{Y}$ is the partial discrete output function.

HYFLOW models describe independent entities that can only communicate through input/output interfaces, defined by sets X and Y, respectively. These sets are structured into continuous and discrete parts

since HYFLOW components can accept and produce hybrid signals. The transition function, δ, describes how a component changes from the current p-state to the next one. Transitions can be triggered by different conditions. A component can change its p-state due the presence of a discrete flow. A component can also change the p-state according to its autonomous behavior specified by the time-to-input/output functions ρ and ω. When the time elapsed in the current p-state reaches a value specified by one of these time functions, the component changes its p-state. A change can also be triggered by any combination of the previous conditions. The output function $\bar{\Lambda}$ describes the continuous flow output of a component. $\bar{\Lambda}$ supports dense outputs description making it possible the sampling operation at arbitrarily time instants. Function λ describes the discrete flow. This function can only be non-null when the time elapsed in the current p-state reaches the time-to-output function ω.

We briefly describe the semantics of tauHyFlow components. Figure 1 depicts the typical trajectories of a HYFLOW component. At time t_1 the component in p-state p_0 samples its input since its elapsed time reaches $\rho(p_0) = e$. The component changes its p-state to $p_1 = \delta((p_0, \rho(p_0)), (x_1, \varnothing))$, where x_1 is the sampled value at time t_1, and no discrete flow is present at that time. At time t_2 the discrete flow x_d is received

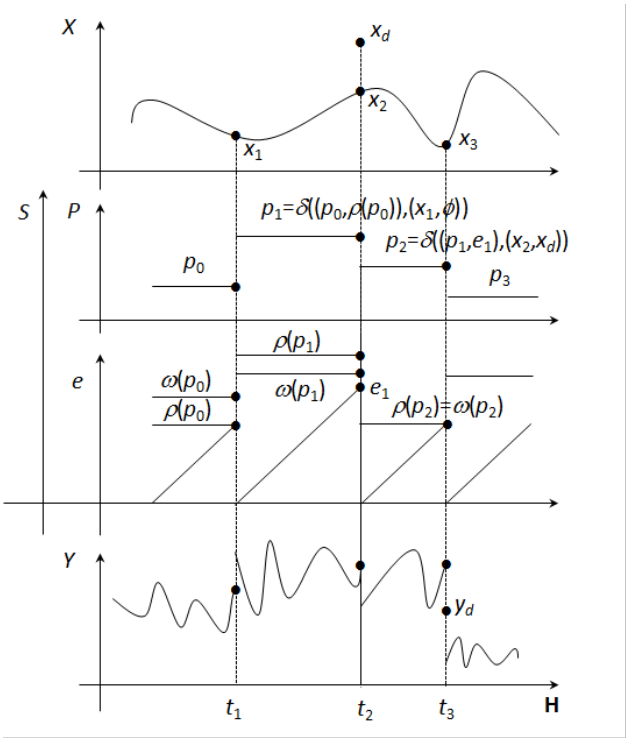

Figure 1: Basic HYFLOW component trajectories.

by the component that changes to p-state $p_2 = \delta((p_1, e_1), (x_2, x_d))$, where x_2 is the continuous flow at t_2. At time t_3 the component reaches the time-to-output limit and it changes to p-state $p_3 = \delta((p_2, \omega(p_2)), (x_3, \varnothing))$. At this time the discrete flow $y_d = \lambda(p_2)$ is produced. Component continuous output flow is always present and given by $\bar{\Lambda}(p_i, e)$, where p_i is the current p-state.

Since HYFLOW sampling is specified by function ρ, sampling time can be modified during simulation. Also, since each model defines its own ρ function, sampling can also be made asynchronously. These features make discrete time formalisms a particular case of an HYFLOW specification were all components are constrained to sample synchronously and at constant time intervals.

3.2 HYFLOW Network Model

HYFLOW network models are compositions of HYFLOW models (basic or other HYFLOW network models). Let $\widehat{N}$ be the set of names corresponding to HyFlow network models, with $\widehat{N} \cap \widehat{B} = \{\}$. Formally, a HYFLOW network model associated with name $N \in \widehat{N}$ is defined by:

$$M_N = (X, Y, \eta),$$

where

N is the network name,

$X = \bar{X} \times \check{X}$ is the set of network input flows,
- $\bar{X}$ is the set of network continuous input flows,
- $\check{X}$ is the set of network discrete input flows,

$Y = \bar{Y} \times \check{Y}$ is the set of network output flows,
- $\bar{Y}$ is the set of network continuous output flows,
- $\check{Y}$ is the set of network discrete output flows,

$\eta \in \widehat{\eta}$ is the name of the dynamic topology network executive,

with
- $\eta \in \widehat{\eta}$ representing the set of all names associated with HYFLOW executive models, constrained to $\widehat{\eta} \cap \widehat{B} = \widehat{\eta} \cap \widehat{N} = \{\}$.

Executives are uniquely assigned to network models, i.e.,

$\forall_{i,j \in \widehat{N}, i \neq j} \eta_i \neq \eta_j$ with $M_k = (X, Y, \eta)_k, \forall_{k \in \widehat{N}}$.

The model of an executive $\eta \in \widehat{\eta}$, is a modified HYFLOW basic model, defined by:

$$M_\eta = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda, \widehat{\Sigma}, \gamma)_\eta,$$

where

$\widehat{\Sigma}_\eta$ is the set of network topologies,

$\gamma_\eta : P_\eta \longrightarrow \widehat{\Sigma}_\eta$ is the topology function.

The network topology $\Sigma_\alpha \in \widehat{\Sigma}_\eta$, corresponding to the p-state $p_\alpha \in P_\eta$, is given by the 3-tuple

$$\Sigma_\alpha = \gamma(p_\alpha) \quad = (C_\alpha, \{I_{i,\alpha}\} \cup \{I_{\eta,\alpha}, I_{N,\alpha}\}, F_{i,\alpha} \cup \{F_{\eta,\alpha}, F_{N,\alpha}\}),$$

where

C_α is the set of names associated with the executive state p_α,

for all $i \in C_\alpha \cup \{\eta\}$,
- $I_{i,\alpha}$ is the sequence of influencers of i,
- $F_{i,\alpha}$ is the input function of i,

$I_{N,\alpha}$ is the sequence of network influencers,

$F_{N,\alpha}$ is the network output function.

For all $i \in C_\alpha$

$M_i = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda)_i$ if $i \in \widehat{B}$,
$M_i = (X, Y, \eta)_i$ if $i \in \widehat{N}$.

The executive is a special component that controls the network topology. Topology depends on the current executive p-state and it is established by the *topology function* γ. Changes in network topology include the ability to modify composition and coupling through add and delete operations. Although HYFLOW relies on a central component to manage the topology of each network, the decision to change this topology can be made by any arbitrary component or in cooperative manner by several components. This decision, however, needs to be communicated to the executive so it can become effective. HYFLOW topology management guarantees well defined and deterministic semantics when changes in topology occur (Barros 2008).

4 1ST-ORDER POLYNOMIAL INTEGRATORS

The solution of a system of ODEs is commonly performed by reducing all ODEs into an equivalent system of 1st-order ODEs. This is achieved by introducing new 1st-order ODEs to help reduce higher order ODEs into a 1st-order system. We describe next 1st-order asynchronous integrators that can use an adaptive step size . These integrators can individually adjust the step size of each ODE for improving simulation performance. Given the ODE $\dot{y} = f(x(t)),$ with $y(0) = y_0$, an asynchronous 1st-order, 1st-degree (Euler) integrator can be described in HYFLOW by:

$$M_B = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda),$$

where

$X = \mathbb{R} \times \mathbb{R}$,
$Y = \mathbb{R} \times \mathbb{R}$,
$P = \{(\alpha, \beta, \beta_p, \kappa, x, y) | \alpha, \beta, \beta_p \in \mathbb{H}_0^+; \kappa, x, y \in \mathbb{R}\}$,
$\rho(\alpha, \beta, \beta_p, \kappa, x, y) = \alpha$,
$\omega(\alpha, \beta, \beta_p, \kappa, x, y) = \beta$,

$$\delta(((\alpha, \beta, \beta_p, \kappa, x, y), e), (x_c, x_d)) = \begin{cases} (\infty, 10^{-5}, 10^{-5}, 0, x_c, y') & \text{if } \alpha = 0, \\ (\infty, \beta', \beta', 0, x_c, y') & \text{if } e = \beta + \varepsilon, \\ (\infty, \min\{\beta_p, |\frac{ERR - \kappa'}{x_c}|\}, \beta_p, \kappa', x_c, y') & \text{otherwise,} \end{cases}$$

where
$\beta' = \min\{K\beta_p, |\frac{ERR}{x_c}|\}$,
$\kappa' = \kappa + |e_{std} \cdot x|$,
$y' = y + e_{std} \cdot x$,
$\lambda(\alpha, \beta, \beta_p, \kappa, x, y) = y$,
$\bar{\Lambda}((\alpha, \beta, \beta_p, \kappa, x, y), e) = y + x \cdot e_{std}$.

The step size $\frac{ERR}{x_c}$ is used for setting a bound to the local error. When the integrator receives an external discrete flow it accumulates the error in variable κ. The accumulated error is used to adjust the step size when the component receives discrete flows. When the discrete flow is released the accumulated error is reset to zero. Increasing step size is limited by the factor K over the previous step β_p, preventing large jumps in consecutive steps. The use of discrete events to stabilize asynchronous ODE integrators was introduced in (Zeigler and Lee 1998) and it can be easily described in HYFLOW. Solver initial state is

given by $s_0 = ((0,\infty,\infty,0,0,y_0),0)$, imposing a first transition at time 0 to sample the derivative x_c. Solver continuous output, defined by $\bar{\Lambda}$, is a 1-degree polynomial. Better accuracy can be obtained, for example, by Adam methods using higher degree polynomial approximations. These integrators can be easily described in HYFLOW requiring the storage of derivative past values (Barros 2015a). A system of two 1st-order, third degree integrators are used in Section 6 (Example II) to solve a 2nd-oder ODE.

5 EXPONENTIAL INTEGRATORS

The use of non-polynomial integrators is relatively recent (Brock and Murray 1952), (Pope 1963), (Certaine 1959), being currently an active area of research (Hochbruck 2010). These integrators approximate ODE solutions by exponential functions, and they were created to enable larger time steps and to fix the stiffness problems introduced by polynomial integrators. We describe here the 1st-order exponential method developed in (Pope 1963). Given the ODE

$$\dot{y} = f(t,y),$$

the second derivative can be computed as:

$$\ddot{y} = f_t + f f_y$$

. A 1st-order Exponential integrator is described by (Pope 1963)

$$y_{n+1} = y_n + hf + \ddot{y} f_y^{-2}(\mathrm{e}^{hf_y} - hf_y - 1).$$

For small values of $|hf_y|$ the recurrence needs to be approximated by the series (Pope 1963):

$$y_{n+1} = y_n + hf + \ddot{y}\sum_{k=2}^{+\infty} \frac{h^k f_y^{k-2}}{k!}.$$

The local truncation error is given by (Pope 1963):

$$ERR = \frac{1}{6}h^3[f_{tt} + 2ff_{ty} + f^2 f_{yy}] + O(h^4).$$

Given the HYFLOW ability to represent arbitrary functions, computing the dense output by an exponential or a sum of values poses no problem. Trigonometric and hyperbolic functions, as required for 2nd-order methods (Pope 1963), can also be easily represented in HYFLOW. For simplicity we describe here a integrator with no explicit time dependency, i.e, $\dot{y} = f(y)$. The exponential integrator is given by the HYFLOW model:

$$M_X = (X,Y,P,\rho,\omega,\delta,\bar{\Lambda},\lambda),$$

where

$$X = \emptyset \times \emptyset,$$
$$Y = \mathbb{R} \times \mathbb{R},$$
$$P = \{(\alpha,\beta,\beta_p,y) | \alpha,\beta \in \mathbb{H}_0^+; y \in \mathbb{R}\},$$
$$\rho(\alpha,\beta,\beta_p,y) = \alpha,$$
$$\omega(\alpha,\beta,\beta_p,y) = \beta,$$
$$\delta((\alpha,\beta,\beta_p,y),e) = \begin{cases} (\alpha,10^{-5},10^{-5},y) & \text{if } \alpha = 0, \\ (\infty,\min\{K\cdot\beta_p,\sqrt[3]{\frac{6\cdot,ERR}{f^2(y)|f_{yy}(y)|}}\},\beta,y+\mathrm{ITG}(e)) & \text{otherwise,} \end{cases}$$

$$\text{where ITG}(h) = \begin{cases} hf(y) + \ddot{y}\sum_{k=2}^{+\infty} \frac{h^k f_y^{k-2}(y)}{k!} & \text{if } |hf_y(y)| \leq 0.1, \\ hf(y) + \ddot{y} f_y^{-2}(y)(e^{hf_y(y)} - hf_y(y) - 1) & \text{otherwise,} \end{cases}$$

and $\ddot{y} = f(y)f_y(y)$,

$\bar{\Lambda}((\alpha, \beta, \beta_p, y), e) = y + \text{ITG}(e)$,

$\lambda(\alpha, \beta, \beta_p, y) = y$.

Likewise the polynomial integrator in the previous section, we use the factor K to limit the amplitude of consecutive step sizes.

Example I: Flame Model

Exponential integrators are know to handle stiff ODEs. There seems to be case that stiffness is a consequence of the polynomial approximation rather than the eigenvalue considerations usually taken as the cause for ODE stiffness (Hairer et al. 2000). A well known stiff ODE is the flame propagation model (Abelman and Patidar 2008), represented by the equation:

$$\dot{y} = y^2(1-y).$$

Results are presented in Figure 2 for a simulation time $T = 2 \cdot 10^4$, $y(0) = 10^{-4}$, $ERR = 10^{-6}$, and $K = 1.1$. The numerical integration was also efficient taking only 483 transitions. The step size is represented in Figure 3. As shown in Figure 2, the exponential integrator exhibits no stiffness, i.e, the typical oscillating

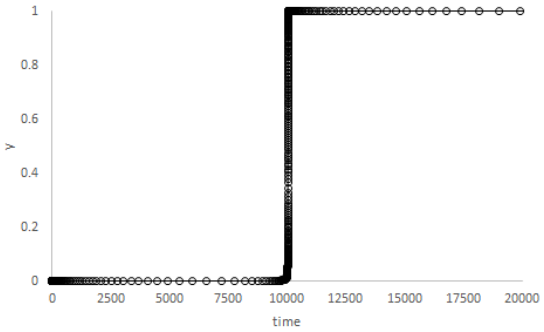

Figure 2: Flame size.

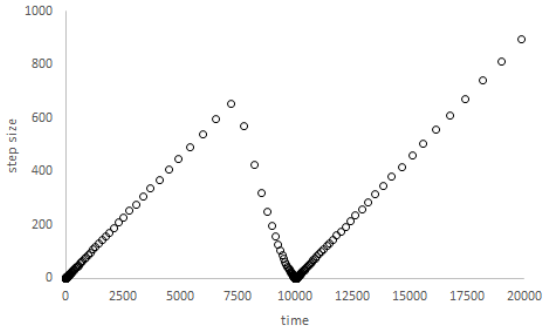

Figure 3: Step size.

behavior characteristic of the explicit polynomial integrators that impose a small step size. As it can be observed in Figure 3, the step size is very large in the region $t > 10^4$, whereas a conventional polynomial integrator would require small values for the integration step due to stiffness (Abelman and Patidar 2008).

6 GEOMETRIC INTEGRATORS

The prevalent rule in most common modeling and simulation tools is to map the set of arbitrary order ODEs into a system of 1st-order ODEs (Fritzson 2003). However, these methods do not preserve system energy being not acceptable for simulating long time periods. Geometric integrators are based on a strategy involving the direct integration of 2nd-order ODEs to obtain solutions that preserve some system properties, like notably, the total energy (Hairer et al. 2005). We introduce here a HYFLOW representation for geometric ODEs. Given the 2nd-order ODE

$$\ddot{y} = f(x(t), y) \quad \text{with } y(0) = y_0, \dot{y}(0) = \dot{y}_0,$$

and using the variable $v = \dot{y}$, a fixed step size h, 2nd-order ODE, 2nd-degree polynomial approximation, geometric integrator is described by the equations (Swope et al. 1982):

$$y_{n+1} = y_n + hv_n + \frac{1}{2}h^2 f_n,$$

$$v_{n+1} = v_n + \frac{h}{2}(f_n + f_{n+1}).$$

The HYFLOW corresponding model is given by:

$$M_\Gamma = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda),$$

where

$X = \mathbb{R} \times \varnothing,$

$Y = \mathbb{R} \times \varnothing,$

$P = \{(y_n, v_n, f_n) | y_n, v_n, f_n \in \mathbb{R}\},$

$\rho(y_n, v_n, f_n) = h,$

$\omega(y_n, v_n, f_n) = \infty,$

$\delta((y_n, v_n, f_n), e), (x_c, x_d)) = (y_{n+1}, v_n + \frac{1}{2}e_{std}(f_n + f_{n+1}), f_{n+1})$, with
$y_{n+1} = y_n + e_{std}v_n + \frac{1}{2}e_{std}^2 f_n$ and
$f_{n+1} = f(x_c, y_{n+1}),$

$\bar{\Lambda}((y_n, v_n, f_n), e) = y_n + e_{std}v_n + \frac{1}{2}e_{std}^2 f_n,$

$\lambda(y_n, v_n, f_n) = \varnothing.$

The recurrences are computed by the transition function δ and the fixed sampling period h is specified by function ρ. The output flow is given by the 2nd-degree polynomial provided by $\bar{\Lambda}$ that defines dense values enabling sampling by external components as shown in Example III.

Example II: Oscillator

Let us consider an oscillator composed by a body M with mass m and a spring K with elasticity constant k, described by the equation:

$$\ddot{y} = -\frac{k}{m}y \qquad \text{with } y(0) = 0, \dot{y}(0) = 2.0.$$

This energy conservation system can be solved by a geometric integrator. The plot of y vs. velocity $\dot{y}$ is depicted in Figure 4. Simulation parameters are $m = 1$ kg, $k = 8$ N/m, $y(0) = 0$, and $\dot{y}(0) = 2$ m/s. Results were obtained for a constant step size of 0.1 s and a run time of 100 s. Although the oscillator has been simulated for a large number of cycles it can be observed that the plot remains stable (a closed ellipse), showing energy preservation. Simulation results were obtained using G-HYFLOW, an implementation of the HYFLOW formalism in the Groovy language (König et al. 2015). Geometric integrators (GIs) outstanding features can be contrasted with the poor performance of 1st-order polynomial integrators (PIs). Figure 5 plots $y \times \dot{y}$ when the oscillator is represented by a system of two 1st-order ODEs and integrated by a 3rd-degree version of the polynomial integrator described in Section 4. The graph is no longer an ellipse becoming a thick line revealing that the solution is not periodic showing also energy variation. Additionally, the GI is more efficient than the two PIs. The GIs used a fixed sampling interval of 0.1 s requiring 1000 transitions for the 100 s interval. The PIs have required a total of 7308 transitions, a larger number for a worse accuracy.

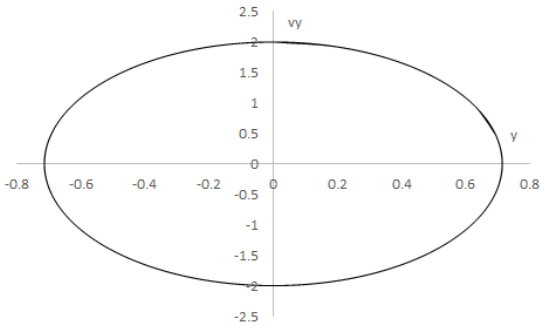

Figure 4: $y \times \dot{y}$ computed by a geometric integrator.

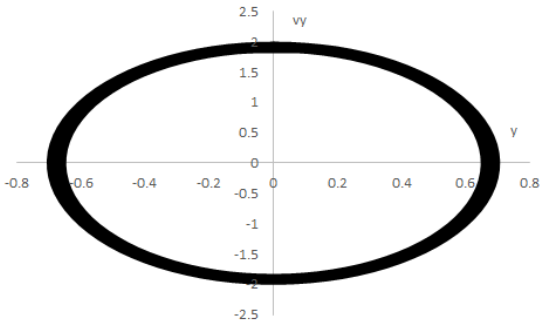

Figure 5: $y \times \dot{y}$ computed by two 1st-order polynomial integrators.

Example III: Double-Mass Oscillator

As mentioned before HYFLOW enables the seamless combination of any models including, obviously, the numerical integrators that are represented as HYFLOW models. To show the combination of integrators we consider a system composed by two bodies M_1 and M_2 with masses m_1 and m_2 connected by a spring K with unstretched length L and elasticity constant k. This system is represented in Figure 6. Positions x_1 and x_2 of the two masses are given by the following 2nd-order ODEs:

$$\ddot{x}_1 = -\frac{k}{m_1}(x_1 - x_2 + L), \tag{1}$$

$$\ddot{x}_2 = -\frac{k}{m_2}(x_2 - x_1 - L). \tag{2}$$

The system is thus represented by two 2nd-order ODEs. A natural choice for solving the equations is to use two geometric integrators, one for representing each body. The HYFLOW network for modeling the oscillator is depicted in Figure 7. Simulation results for a 5 s interval are depicted in Figure 8 with parameters

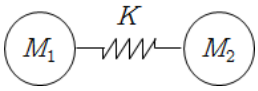

Figure 6: Double mass oscillator.

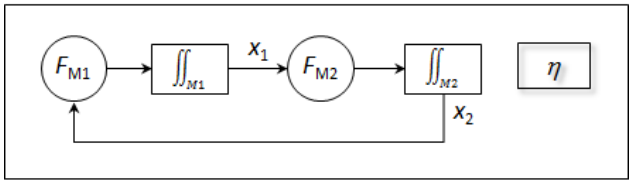

Figure 7: HYFLOW network for representing equations (1) and (2).

$m_1 = 5$ kg, $m_2 = 1$ kg, $k = 50$ N/m, $L = 0.2$ m, $x_1(0) = 0$, $\dot{x}_1 = 0$, $x_2(0) = L$ and $\dot{x}_2 = 0.5$ m/s. Representing the oscillator by a HYFLOW network enables solution to be computed by two geometric integrators with independent sampling rates. For M_1 a step size of 0.05 s was used, while M_2 was integrated with a step size of 0.01 s. The ability to use different rates enables faster simulations. Efficiency can be further improved through the use of asynchronous variable step geometric integrators (Stoffer 1995). Geometric integrators provide an accurate alternative to the representation of 2nd-order ODEs into 1st-order ODEs. HYFLOW enables the seamless composition of geometric integrators with polynomial and exponential integrators described in Sections 4 and 5, since they all share the same underlaying representation.

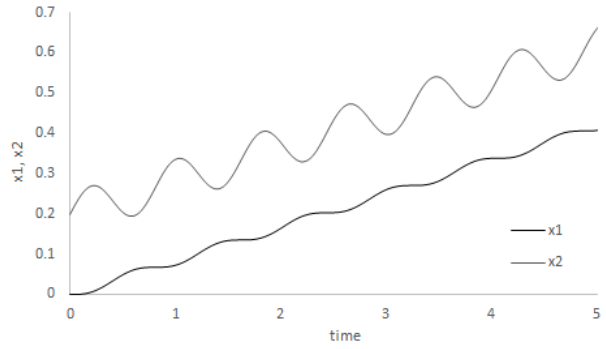

Figure 8: Positions x_1 and x_2 of bodies M_1 and M_2.

7 CONCLUSION

HYFLOW generalized sampling and dense output support enable a novel representation of continuous systems. HYFLOW can exactly represent continuous functions in a digital computer, supporting both time and component varying sampling to provide access to (dense) output values. We have shown that sampling is a basic operator to represent numerical ODE integrators. In particular, we have demonstrated that HYFLOW provides a unifying representation for polynomial, geometric and exponential integrators. Since integrators are represented as particular models in HYFLOW they can be seamlessly combined to describe complex systems based on independent components. As future work we plan to represent other ODE numerical integrators, like BDF methods (Ascher and Petzold 1988).

REFERENCES

Abelman, S., and K. Patidar. 2008. "Comparison of some recent numerical methods for initial-value problems for stiff ordinary differential equations". *Computers and Mathematics with Applications* 55:733–744.

Ascher, U., and L. Petzold. 1988. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM.

Barros, F. 2003. "Dynamic Structure Multiparadigm Modeling and Simulation". *ACM Transactions on Modeling and Computer Simulation* 13 (3): 259–275.

Barros, F. 2005a. "Simulating the Data Generated by a Network of Track-While-Scan Radars". In *12th Annual IEEE International Conference on Engineering Computer- Based Systems*, 373–377.

Barros, F. 2005b. "A System Theory Approach to the Representation of Mobile Digital Controllers Agents". In *International Workshop on Radical Agent Concepts*.

Barros, F. 2008. "Semantics of Discrete Event Systems". In *Distributed Event-Based Systems*, 252–258.

Barros, F. 2015a. "Asynchronous, Polynomial ODE Solvers based on Error Estimation". In *Symposium on Theory of Modeling and Simulation*.

Barros, F. 2015b. "A modular representation of Fluid Stochastic Petri Nets". In *Symposium on Theory of Modeling and Simulation*, 122–128.

Benveniste, A., and G. Berry. 1991. "The Synchronous Approach to Reactive and Real-Time Systems". *Proceedings of the IEEE* 79 (9): 1270–1282.

Brock, P., and F. Murray. 1952. "The use of Exponential Sums in Step by Step Integration". *Mathematics of Computation*:63–78.

Certaine, J. 1959. *Mathematical Methods for Digital Computers*, Volume 1, Chapter The Solution of Ordinary Differential Equations with Large Time Constants, 128–132. John Wiley.

Fritzson, P. 2003. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley.

Hairer, E. 1997. "Variable time step integration with Symplectic methods". *SIAM Journal of Scientific Computation* 26 (6): 1838–1851.

Hairer, E., C. Lubich, and G. Wanner. 2005. *Geometrical Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer.

Hairer, E., S. Nörsett, and G. Wanner. 2000. *Solving Ordinary Differential Equations II: Stiff Problems*. Springer.

Henzinger, T. 1996. "The Theory of Hybrid Automata". In *11th Annual IEEE Symposium on Logic in Computer Science*, 278–292.

Hochbruck, M. 2010. "Exponential Integrators". *Acta Numerica*:209–286.

König, D., P. King, G. Laforge, H. D'Arcy, C. Champeau, E. Pragt, and J. Skeet. 2015. *Groovy in Action*. Manning.

Lynch, N., R. Segala, and F. Vaandrager. 2003. "Hybrid I/O Automata". *Information and Computation* 185:105–157.

Manna, Z., and A. Pnueli. 1993. "Verifying Hybrid Systems". In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, 4–35.

Migoni, G., M. Bortolotto, E. Kofman, and F. Cellier. 2013. "Linearly implicit quantization-based integration methods for stiff ordinary differential equations". *Simulation Modelling Practice and Theory* 35:118–136.

Neta, B., and C. Ford. 1984. "Families of Methods for Ordinary Differential Equations based on Trigonometric Polynomials". *Journal of Computational and Applied Mathematics* 22:33–38.

Pope, D. A. 1963. "An Exponential Method of Numerical Integration of Ordinary Differential Equations". *Communications of the ACM* 6 (8): 491–493.

Praehofer, H. 1991. *System Theoretic Foundations for Combined Discrete-Continuous System Simulation*. Ph.d. diss., University of Linz, Austria.

Schierz, T., M. Arnold, and C. Clauss. 2006. "Co-simulation with communication step size control in an FMI compatible master algorithm". In *Proceedings of the 9th International Modelica Conference*, 205–214.

Skelboe, S. 2006. "Adaptive Partitioning Techniques for Ordinary Differential Equations". *BIT Numerical Mathematics* 46:617–629.

Stoffer, D. 1995. "Variable Steps for Reversible Integration Methods". *Computing* 55:1–22.

Swope, W. C., H. C. Andersen, P. H. Berens, and K. R. Wilson. 1982. "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters". *Journal of Chemical Physics* 76 (1): 637–649.

Zeigler, B. 1976. *Theory of Modelling and Simulation*. Wiley.

Zeigler, B., and J. Lee. 1998. "Theory of Quantized Systems: Formal Basis for DEVS/HLA Distributed Simulation Environment". In *Proceedings of the SPIE: Enabling Technology for Simulation Science II*, Volume 3369, 49–58.

AUTHOR BIOGRAPHY

FERNANDO J. BARROS is a professor at the University of Coimbra, Portugal. His research interests include theory of modeling & simulation and hybrid dynamic topology systems. He published more than 70 papers in journals, book chapters and conference proceedings, and he has organized several conferences and workshops in the area of simulation. Fernando Barros is a member of the editorial board of the *Int. J. Simulation and Process Modelling* and associate editor of the *Int. J. Agent Technologies and Systems*. His email address is barros@dei.uc.pt.