Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

MODULAR SIMULATION OF 2nd-ORDER ENERGY PRESERVING SYSTEMS

Fernando J. Barros

Department of Informatics Engineering
University of Coimbra
3030 Coimbra, PORTUGAL

ABSTRACT

Co-simulation requires the ability to represent systems in a modular form, while guaranteeing that simulation algorithms rely exclusively on model external interface. These conditions enable models to be composed without exposing their internal state, a requisite needed for the co-simulation of cyber-physical systems. In this paper we provide a modular representation of geometric integrators, a type of integrator essential to the simulation of 2nd-order energy preserving systems. These integrators offer an alternative to the conventional decomposition of systems into 1st-order Ordinary Differential Equations (ODEs). This latter approach, although commonly used in nowadays M&S software, is not acceptable when long simulation runs are needed. Geometric integrators are represented in the Hybrid Systems Specification (HYFLOW), a modeling formalism to represent hybrid modular dynamic topology systems. We show that HYFLOW enables the composition of geometrical solvers, allowing the co-simulation of complex 2nd-order energy preserving systems.

1 INTRODUCTION

Modular models play a fundamental role in the representation of complex systems. Modularity enables the decomposition of systems into small units that can be independently developed and tested, promoting model reuse. Complex models can also be created by the composition of other models available in model libraries, enabling a faster model development. Additionally, there has been a growing interest in co-simulation approaches (Enge-Rosenblatt et al. 2011). Co-simulation enables the interoperability of models that are only known through their interface, requiring the use of both modular models and simulators. A key advantage of co-simulation approaches is the interoperability of binary models that can hide model internal information leveraging, for example, intellectual property protection (Bastian et al. 2011). The concept of modular simulator was introduced in (Zeigler 1984) for discrete event systems. This concept has been extend by the Hybrid Flow System Specification (HYFLOW) formalism (Barros 2003), (Barros 2016b), to enable the co-simulation of hybrid systems.

HYFLOW can provide a direct representation to a large variety of systems while guaranteeing their composition. These systems include digital controllers (Barros 2005b), digital filters (Barros 2005a), 1st-order ordinary differential equations (ODEs) integrators (Barros 2015a) and fluid stochastic Petri nets (Barros 2015b).

In this paper we show HYFLOW ability to co-simulate complex systems described by the composition of 2nd-order (geometric) numerical integrators. These integrators have the remarkable characteristic of energy conservation, making them ideal for modeling some classes of 2nd-order physical systems. The co-simulation of conventional 1st-order integrators has been described in (Zeigler and Lee 1998). However, this type of approach requires mapping all ODEs into 1st-order ODEs. This solution, while very common (Fritzson 2015), does not guarantee energy preservation and it cannot be used for long simulation runs, like those required, for example, in celestial mechanics (Gladman et al. 1991), where simulation times of thousands of years can be required.

In this paper we provide a modular description of geometric integrators based on the HYFLOW formalism. Our approach is demonstrated through several examples that include a series of planar pendulum and a simplified model of the solar system for studying Moon trajectory. To the best of our knowledge, HYFLOW has provided the first modular description of geometric integrators, enabling a new approach for the co-simulation of energy preserving systems (Barros 2016a).

2 THE HYFLOW FORMALISM

The Hybrid Flow System Specification (HYFLOW) is a formalism for representing hybrid systems with a time-variant topology (Barros 2003). HYFLOW achieves the representation of continuous variables using the concept of multi-sampling (Barros 2002), while the representation of discrete events is based on the Discrete Event System Specification (DEVS) (Zeigler et al. 2000). HYFLOW can also represent dense outputs, providing a framework for describing arbitrary continuous function on a digital computer. HYFLOW defines two types of models: basic and network. Basic models provide state representation and state transition functions for describing model dynamic behavior. Network models are a composition of basic models and/or other network models. Given this definition, a network provides an abstraction for representing hierarchical systems.

2.1 HyFlow Basic Model

A HYFLOW basic model associated with name B is defined by

$$M_B = (X, Y, P, P_0, \rho, \omega, \delta, \Lambda_c, \lambda_d)$$

where

$X = X_c \times X_d$ is the set of input flow values
  X_c is the set of continuous input flow values
  X_d is the set of discrete input flow values
$Y = Y_c \times Y_d$ is the set of output flow values
  Y_c is the set of continuous output flow values
  Y_d is the set of discrete output flow values
P is the set of partial states (p-states)
P_0 is the set of (valid) initial p-states
$\rho : P \longrightarrow \mathbb{H}_0^+$ is the time-to-input function
$\omega : P \longrightarrow \mathbb{H}_0^+$ is the time-to-output function
$S = \{(p,e) | p \in P, 0 \leq e \leq \nu(p)\}$ is the state set
  with $\nu(p) = \min\{\rho(p), \omega(p)\}$, the time-to-transition function
$\delta : S \times X^\phi \longrightarrow P$ is the transition function
  where $X^\phi = X_c \times (X_d \cup \{\phi\})$
  and ϕ is the null value (absence of value)
$\Lambda_c : S \longrightarrow Y_c$ is the continuous output function
$\lambda_d : P \longrightarrow Y_d$ is the discrete output function

HYFLOW time base is the set of hyperreals numbers $\mathbb{H} = \{x + z\varepsilon | x \in \mathbb{R}, z \in \mathbb{Z}\}$, where ε is an infinitesimal (infinitely small) number, such that $\varepsilon > 0$ and $\varepsilon < \frac{1}{n}$ for $n = 1, 2, 3, ...$ (Goldblatt 1998). The set of positive hyperreals is defined by $\mathbb{H}_0^+ = \{h \in \mathbb{H} | h \geq 0\}$. The discrete output of a component described by a HYFLOW basic model is constrained to be null (ϕ) when in states (s,e) with $e \neq \omega(p)$.

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 $p_1 = \delta((p_0, \rho(p_0)), (x_1, \phi))$, where x_1 is the sampled value and no discrete flow is present.

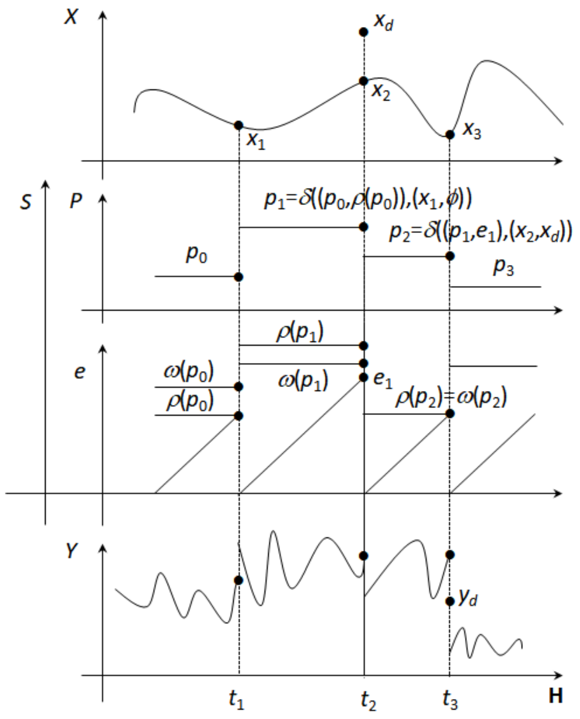

Figure 1: Basic HYFLOW component trajectories.

At time t_2 the discrete flow x_d is received 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 time limit and it changes to p-state $p_3 = \delta((p_2, \omega(p_2)), (x_3, \phi))$. At this time the discrete flow $y_d = \lambda_d(p_2)$ is produced. Additionally, component continuous output flow is always present and given by $\Lambda_c(p, e)$. A detailed description of HYFLOW semantics with the corresponding modular co-simulators is given in (Barros 2008). The use of hyperreals numbers is also shown to be fundamental for obtaining deterministic models (Barros 2008).

2.2 Example: Continuous Flow Generator

In many systems, entities can be described by known functions. For example, the path of a aircraft can be planned to follow some trajectory. The representation of this type of dynamic behavior can be achieved by the continuous function generator $f : \mathbb{R} \longrightarrow \mathbb{R}$ represented by the HYFLOW model:

$$M_f = (X, Y, P, P_0, \rho, \omega, \delta, \Lambda_c, \lambda_d)$$

where

$X = \{\} \times \{\}$, $Y = \mathbb{R} \times \{\phi\}$
$P = P_0 = \{\}$
$\rho(\phi) = \omega(\phi) = \infty$, $\delta((\phi, e), (\phi, \phi)) = \phi$

$\Lambda_c(\phi,e) = f(e_{std})$, where e_{std} is the standard part of the hyperreal e
$\lambda_d(\phi) = \phi$

This model can represent arbitrary continuous functions. For example, a trigonometric signal can be described by $f(t) = A\cos(\varpi t + \varphi)$. The generator is a passive component with no autonomous behavior, where all information can be retrieved through sampling. This model produces an *exact* output since no approximations is used. This contrasts with other modeling approaches that limit continuous signals to be represented by piecewise constant segments (Tripakis et al. 2013).

2.3 HyFlow Network Model

HyFlow network models are compositions of HyFlow models (basic or other HyFlow network models). A HyFlow network model associated with name N is defined by

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

where

N is the network name
$X = X_c \times X_d$ is the set of network input flows
  X_c is the set of network continuous input flows
  X_d is the set of network discrete input flows
$Y = Y_c \times Y_d$ is the set of network output flows
  Y_c is the set of network continuous output flows
  Y_d is the set of network discrete output flows
η is the name of the dynamic topology network executive

The model of the executive is a modified HyFlow basic model, defined by

$$M_\eta = (X_\eta, Y_\eta, P, P_0, \rho, \omega, \delta, \Lambda_c, \lambda_d, \widehat{\Sigma}, \gamma)$$

where

$\widehat{\Sigma}$ is the set of network topologies
$\gamma : P \longrightarrow \widehat{\Sigma}$ is the topology function

The network topology $\Sigma_\alpha \in \widehat{\Sigma}$, corresponding to the p-state $p_\alpha \in P$, is given by

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

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 $i \in C_\alpha$

$M_i = (X, Y, P, P_0, \rho, \omega, \delta, \Lambda_c, \lambda_d)_i$ if i is a basic model
$M_i = (X, Y, \eta)_i$ if i is a network model

The topology of a network is defined by its executive through the topology function γ, that maps executive p-state into network composition and coupling. Thus, topology adaption can be achieved by changing executive p-state.

Contrarily to other modeling formalisms (Zeigler et al. 2000), HyFlow uses the set of influencers and not the set of influencees to describe model interaction. This choice enables the interoperability and reuse of models having different interfaces, since model input function, defined at composition time, enables the adaptation between arbitrary influencers interfaces and the model. Thus, influencers enable a model to be be (re)used without modification in any context. In this paper we exploit the ability to combine several output values into a single value required by integrators, making them reusable since they are independent of the set of influencers. HyFlow networks are used in the next section to describe examples of model interoperability.

3 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 2015). This is not acceptable for simulating long periods in systems, like many Hamiltonian systems, where solutions have properties that can only be observable after long intervals. For example, in systems involving celestial mechanics, properties like the precession of planets, may require the simulation of hundreds of years to be characterized. The traditional methods based on decomposition are not acceptable in these cases, and a direct representation of higher order ODEs have been proposed (Hairer et al. 2005). We introduce now the HyFlow representation for geometric ODEs. Given the 2nd-order ODE

$$y'' = f(x(t), y) \quad \text{with } y(0) = y_0, y'(0) = v_0$$

and using the variable $v = y'$, a fixed stepsize T, 2nd-order ODE, 2nd-degree polynomial approximation, geometric integrator is described by the equations (Swope et al. 1982):

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

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

Considering $y'' = f(x(t))$, the HyFlow model of the geometric integrator is given by:

$$M_\Gamma = (X, Y, P, P_0, \rho, \omega, \delta, \Lambda_c, \lambda_d)$$

where

$X = \mathbb{R} \times \phi$
$Y = \mathbb{R} \times \phi$
$P = \{(\alpha, y_n, v_n, f_n) | \alpha, y_n, v_n, f_n \in \mathbb{R}\}$
$P_0 = \{(0, y_n, v_n, f_n) \in P\}$
$\rho(\alpha, y_n, v_n, f_n) = \alpha$
$\omega(\alpha, y_n, v_n, f_n) = \infty$

$$\delta((\alpha,y_n,v_n,f_n),e),(x_c,x_d)) = (T, y_{n+1}, v_n + \tfrac{1}{2}e_{std}(f_n + f_{n+1}), f_{n+1})$$
$$\text{with } y_{n+1} = y_n + e_{std}v_n + \tfrac{1}{2}e_{std}^2 f_n \text{ and } f_{n+1} = f(x_c)$$
$$\Lambda_c((\alpha,y_n,v_n,f_n),e) = y_n + e_{std}v_n + \tfrac{1}{2}e_{std}^2 f_n$$
$$\lambda_d(\alpha,y_n,v_n,f_n) = \phi$$

The recurrences are computed by the transition function δ and the sampling period T is specified by function ρ. The initial sampling period is set to 0 so the associated component can read the input and compute the value of f_n. The output flow is given by the 2nd-degree polynomial provided by function Λ_c. Given that sampling rate can be adjusted by the ρ function, HYFLOW can also represent variable sampling geometric integrators (Hairer and Soderlind 2005).

3.1 Example: Simple Harmonic Oscillator

As a first example showing the application of a geometric integrator we simulate the simple oscillator composed by a mass m, and a spring with elasticity K and unstretched length L depicted in Figure 2. The acceleration obeys equation:

$$x'' = -\frac{K}{m}(x - L) \tag{1}$$

The HYFLOW network describing the oscillator is depicted in Figure 3 and defined by:

$$C = \{M\}, I_M = \{M\}, I_\eta = \{\}$$
$$F_M(x_c,x_d) = (-\frac{K}{m}(x_c - L), \phi)$$

where M is a geometric integrator that computes mass position and velocity. A network description is required for the oscillator since we have chosen the 2nd derivative to be given as the input to the geometric integrator. This choice simplifies the composition of oscillators as shown in the next example.

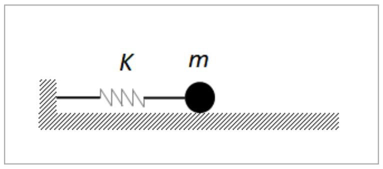

Figure 2: Horizontal oscillator.

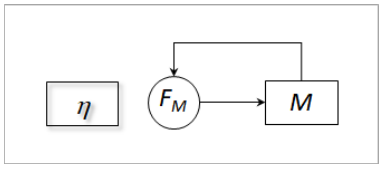

Figure 3: HYFLOW network model for the oscillator.

Mass position and velocity are represented in Figure 4. Results were obtained with a fixed sampling interval of 0.01s, $m = 4$ kg, $L = 1.0$ m and $K = 500$ N/m, $x_0 = 0.5$ m, $v_0 = 0$, and a simulation time of 10 s. The plot of x vs. $v = x'$ is depicted in Figure 5.

For executing the simulation we have used HYFLOW++, an implementation of HYFLOW in the C++ language. HYFLOW++ definition of the oscillator network is described in Figure 6. Class `Executive` is the base class for all network executives. Mass M is added in Line 8, as an instance of the `Geometric<double>` class. The input function is set in Line 9 according to Equation (1). Integrator sampling time is set to 10^{-2} s (Line 9). The `Geometric` class can be parameterized with 3D vectors so it can be reused to describe 3D models, like planets, in Section 3.3.

A distinctive feature of the geometrical integrator is energy conservation.Conventional integrators, on the other hand, do not exhibit the energy conservation property, making them unsuitable for long simulation runs (Hairer et al. 2005).

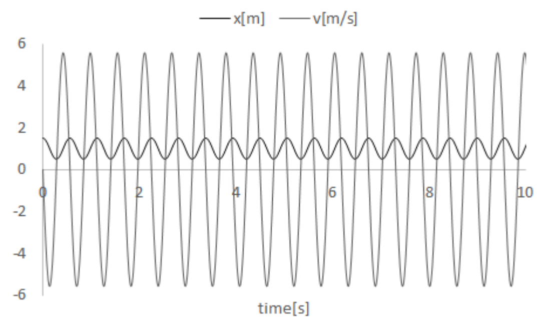

Figure 4: Pendulum position x and velocity v.

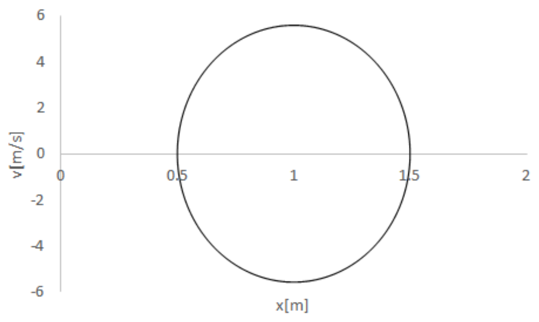

Figure 5: $x \times v$ plot.

```
class Oscillator: public Executive {
private:
   const double K = 500.0;//N m-1
   const double L = 1.0;//m
   const double M = 4.0;//kg
public:
   Oscillator(const std::string& name): Executive(name) {
      add(new Geometric<double>("M", 1.5, 0.0, 1.0e-2));
      influencers("M", {"M"}, [&](const std::vector<double>& x) {return -K/M*(x[0] -L);});
   }
};
```

Figure 6: HYFLOW++ oscillator network topology.

3.2 Example: Composition of Vertical Pendula

Geometric integrators can be combined to model more complex systems. We consider now the 3-pendulum system represented in Figure 7. Each pendulum is described by the equation:

$$\theta_i'' = \frac{T_i|z}{mL^2}, \quad i = 1,2,3$$

where torque is given by

$$T_i = R_i \times (F_{i,i-1} + F_{i,i+1} + (0, -mg, 0)),$$

with R_i the position of pendulum (mass) i, and $T|z$ the z-component of the torque T. Forces $F_{i,i-1}$ and $F_{i,i+1}$ are excerpted by the left and right neighbor springs, respectively. Spring forces between positions p_i and p_j are given by:

$$F_{i,j} = -K((p_i - p_j) - Lu_{i,j}) \quad (2)$$

where $u_{i,j}$ is the unit vector of $p_i - p_j$ and L represents the unstretchable spring length. We consider that pendulum are equally spaced distance $2L$. Positions $p_0 = (0, L, 0)$ and $p_4 = (8L, 0, L)$ are the left and right ground positions, respectively. Each pendulum can be modeled by the geometric integrator described before. HYFLOW enables the seamless composition of models and the system can be represented the HYFLOW network given in Figure 8.

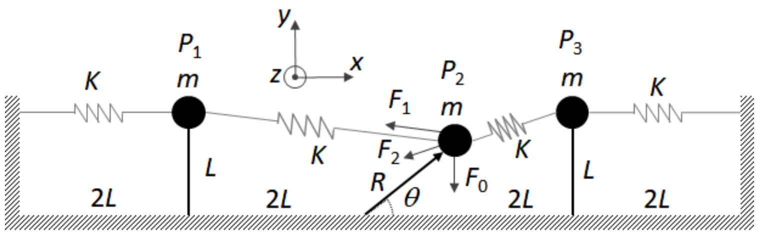

Figure 7: System of 3 planar vertical pendula.

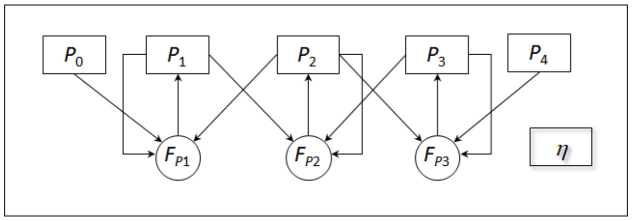

Figure 8: HyFlow representation of the pendula system.

The network has a topology described by:

$$C = \{P_0, P_1, P_2, P_3, P_4\}, I_\eta = I_{P_0} = I_{P_4} = \{\}$$
$$I_{P_1} = \{P_0, P_1, P_2\}, I_{P_2} = \{P_1, P_2, P_3\}, I_{P_3} = \{P_2, P_3, P_4\}$$
$$F_{P_1}((p_0,\phi),(p_1,\phi),(p_2,\phi)) = (\frac{p_1 \times (F_{1,0} + F_{1,2} + (0,-mg,0))|_z}{mL^2}, \phi)$$
$$F_{P_2}((p_1,\phi),(p_2,\phi),(p_{3c},\phi)) = (\frac{p_2 \times (F_{2,1} + F_{2,3} + (0,-mg,0))|_z}{mL^2}, \phi)$$
$$F_{P_3}((p_2,\phi),(p_3,\phi),(p_4,\phi)) = (\frac{p_3 \times (F_{3,2} + F_{3,4} + (0,-mg,0))|_z}{mL^2}, \phi)$$

where forces $F_{1,0}, F_{1,2}, F_{2,1}, F_{2,3}, F_{3,2}$ and $F_{3,4}$ are computed by Equation (2).

Components P_0 and P_4 provide the two ground locations for the first an last spring, respectively. They are modeled by the HyFlow generator described of Section 2.1. For example, P_0 continuous output function is given by $\Lambda_c(\phi, e) = (0, L, 0)$, representing the point where the first spring is anchored to. The system was simulated during a period of 10 s using a fixed stepsize of 10^{-3} s, with parameters: $L = 0.5$m, $K = 100$ N/m, $m = 1$ kg, $\theta_{0,1} = \pi/2, \theta_{0,2} = \pi/4, \theta_{0,3} = \pi/2$, $\theta'_{0,1} = 2, \theta'_{0,2} = 0, \theta'_{0,3} = -2$. The plots $\theta \times \theta'$ for pendula P_1 and P_2 are depicted in Figures 9 and 10, respectively. Although the total system energy of should remain constant, since there are no dissipative elements, the energy is transfered between pendula, making the phase plot departing from the ellipse obtained for the simple oscillator of the previous example.

The $x \times y$ trajectories of the 3 pendula is plotted in Figure 11. HyFlow enables the composition of modular geometric integrators to model complex systems. To the best of our knowledge, HyFlow is the only modeling formalism to allow the co-simulation of geometric integrators. Given its modular definition, HyFlow also enables the composition of these integrators with other models already described in the formalism (Barros 2003).

3.3 Example: Earth/Moon System

We describe next an abridged model of the solar system intended to study the Moon motion around the Earth. In this model only the Sun, Earth, Moon and Jupiter are represented. The acceleration of a body i

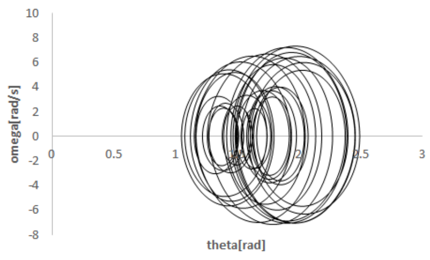

Figure 9: $\theta \times \omega$ plot for pendulum P_1.

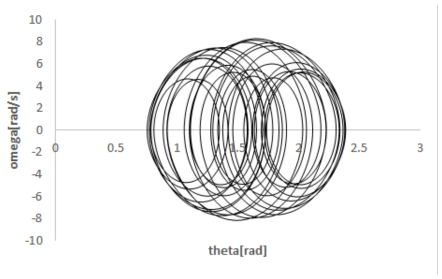

Figure 10: $\theta \times \omega$ plot for pendulum P_2.

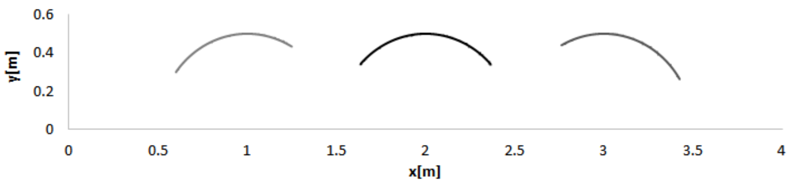

Figure 11: Position of pendula P_1, P_2 and P_3.

can be computed under Newton classical gravity by:

$$x_i'' = -G \sum_{j \in r_i} \frac{m_j d_{ji}}{\|d_{ji}\|^3}$$

where r_i is the set of bodies that have a *relevant* influence over i, and d_{ji} is the distance between bodies j and i. In reality, all bodies affect each other, but in this simplified model we have discarded, for example, the influence of Moon in Sun or Jupiter motion. For defining the HYFLOW network of the solar system the following set of influencers have been considered:

I_{Sun} = {Sun, Jupiter}, I_{Jupiter} = {Sun, Jupiter}
I_{Earth} = {Sun, Earth, Moon, Jupiter}, I_{Moon} = {Sun, Earth, Moon, Jupiter}

Likewise in the previews examples, we have considered that body i belongs to its set of influencers so we can computer its relative distance to the other bodies. Moon trajectory, taken Earth as the center, is depicted in Figure 12 for a 1-year simulation time.

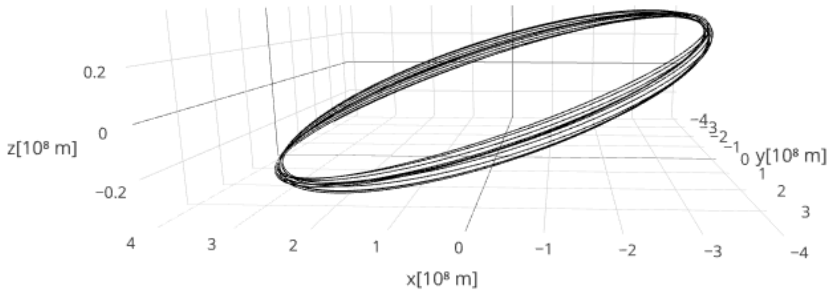

Figure 12: Moon trajectory viewed from Earth.

HyFlow modularity has enabled the representation of each planet/star as an independent unit that is co-simulated individually. Given the HyFlow definition of geometric integrators, only composition is required to simulate the (abridged) solar system. HyFlow++ reflects integrator modular definition by enabling the same implementation of the integrators to be used in all the examples described in this paper.

4 RELATED WORK

Most of continuous simulation tools impose the representation of ODEs into a set of 1^{st}-order differential equations. This is the case of state-of-the art M&S environments like Modelica (Fritzson 2015), where the language supports operators for expressing first derivatives, being user responsibility to map higher order ODEs into 1^{st}-order ODEs. Other tools like PowerDEVS (Bergero and Kofman 2011), based on the discrete event paradigm, provide only support for 1^{st}-order ODEs. Modeling formalisms like DEV&DESS (Präehofer 1991), provide also only support for 1^{st}-order ODEs. Formalisms like Hybrid Automata (Henzinger 1996), DEV&DESS (Präehofer 1991), and Fluid Stochastic Petri Nets (Ciardo et al. 1999), for example, are closed and do not support higher order ODEs, requiring major changes in their representation to support new models. We consider that, in some cases, the user should have more control over the integration algorithms. For example, although mapping a 2^{nd}-order ODE into a system of two 2^{nd}-order ODEs can be automatically generated by the simulation tool, this mapping can lead to erroneous results and the user should be allowed to choose the most appropriate numerical methods.

The explicit representation of 1^{st}-order ODE numerical solvers in a modeling formalism was introduced by quantized DEVS (Zeigler and Lee 1998) and it uses in QSS (Migoni et al. 2013), the polynomial representation introduced in (Giambiasi et al. 2001), to achieve a better accuracy through higher degree approximating polynomials. This work provides a discrete event description of 1^{st}-order numerical solvers opening new perspectives in model interoperability and in the representation of hybrid systems. QSS relies on the local estimation of the error and on the transmission of polynomial coefficients to achieve the representation of continuous signals. QSS provides the conventional decomposition of the 2^{nd}-order ODE: $y'' = f(x(t), y)$, into the 1^{st}-order system: $y_2' = f(x(t), y_1)$ and $y_1' = y_2$. QSS integrators are asynchronous DEVS models producing discrete event outputs based on their internal quantization-based behavior. Unfortunately, conventional error control procedures cannot be used for setting the time step in geometric integrators (Calvo and Sanz-Serna 1993), preventing discrete event schemes, like QSS, to be used for supporting these integrators.

Multi-paradigm modeling and model transformation have been proposed as solutions for representing cyber-physical systems (Vangheluwe 2000). However, as shown before, the interoperability of geometric integrators is a complex problem by itself, although it is intended to couple models within the same (ODE) paradigm. We consider multi-paradigm modeling to be orthogonal to co-simulation, being the challenge the quest for a unifying formalism that can guarantee the interoperability of the large variety of numerical methods used for representing hybrid systems. As shown here, model transformation is also orthogonal to co-simulation, since no transformations were required to enable HyFlow-based numerical integrators to interoperate. The mapping of 2^{nd}-order ODEs into a modeling formalism could be achieved through model transformation. However, this is only meaningful/useful if that formalism can support the basic constructs that guarantee component interoperability.

Conceptual approaches for modeling and simulation has been proposed (Tolk et al. 2018), (Saiku et al. 2013). However, these solutions lack an explicit representation for sampling and dense output that were used here to represent numerical integrators.To the best of our knowledge we have developed the first modular representation of geometric integrators that can be seamlessly combined with other families of hybrid models (Barros 2016a).

5 CONCLUSION

Geometric integrators play a key role in simulating 2^{nd}-order energy preserving systems. These integrators enable long simulation times while keeping accurate results. In this paper we have shown that the HyFlow formalism enables a modular representation of geometric integrators being able to represent complex systems by composing these solvers. HyFlow departs from conventional approaches that require all ODEs to be mapped into a system of 1st-order ODEs. Given that geometric integrators are specific models in the HyFlow formalism, other numerical methods can be described in HyFlow without requiring the modification of the formalism, while guaranteeing model interoperability.

REFERENCES

Barros, F. 2002. “Towards a Theory of Continuous Flow Models”. *International Journal of General Systems* 31(1):29–39.

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*, edited by J. Rozenblit, T. ONeill, and J. Peng, 373–377. April 4th-7th, Greenbelt, MD, USA.

Barros, F. 2005b. “A System Theory Approach to the Representation of Mobile Digital Controllers Agents”. In *International Workshop on Radical Agent Concepts*, edited by M. Hinchey, P. Rago, J. Rash, C. Rouff, R. Sterritt, and W. Truszkowski, 321–333. September 20th-22nd, Greenbelt, MD, USA.

Barros, F. 2008. “Semantics of Discrete Event Systems”. In *Distributed Event-Based Systems*, edited by R. Baldoni, A. Buchmann, and S. Piergiovanni, 252–258. July 1st-4th, Rome, Italy.

Barros, F. 2015a. “Asynchronous, Polynomial ODE Solvers based on Error Estimation”. In *Symposium on Theory of Modeling and Simulation*, edited by F. Barros, M. Hwang, H. Prähofer, and X. Hu, 115–121. April 12th-15th, Alexandria, VA, USA.

Barros, F. 2015b. “A Modular Representation of Fluid Stochastic Petri Nets”. In *Symposium on Theory of Modeling and Simulation*, edited by F. Barros, M. Hwang, H. Prähofer, and X. Hu, 122–128. April 12th-15th, Alexandria, VA, USA.

Barros, F. 2016a. “A Modular Representation of Asynchronous, Geometric Solvers”. In *Symposium on Theory of Modeling and Simulation*, edited by F. Barros, H. Prähofer, X. Hu, and J. Denil. April 3rd-5th, Pasadena, CA, USA.

Barros, F. 2016b. “On the Representation of Time in Modeling & Simulation”. In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. Roeder, P. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. Chick, 1571–1582: Piscatawayy, New Jersey: IEEE.

Bastian, J., C. Clauß, S. Wolf, and P. Schneider. 2011. “Master for Co-Simulation Using FMI”. In *8th Modelica Conference*, edited by C. Clauß, 115–120. March 20th-22nd, Dresden, Germany.

Bergero, F., and E. Kofman. 2011. “PowerDEVS: A Tool for Hybrid System Modeling and Real Time Simulation”. *Simulation: Transactions of the Society for Modeling and Simulation International* 87(1-2):113–132.

Calvo, M., and J. Sanz-Serna. 1993. “The Development of Variable-step Symplectic Integrators with Application to the Two-body Problem”. *SIAM Journal Science Computing* 14(4):936–952.

Ciardo, G., D. Nicol, and K. Trivedi. 1999. “Discrete-Event Simulation of Fluid Stochastic Petri Nets”. *IEEE Transactions on Software Engineering* 25(2):207–217.

Enge-Rosenblatt, O., C. Clausß, A. Schneider, and P. Schneider. 2011. “Functional Digital Mock-up and the Functional Mock-up Interface-Two Complementary Approaches for a Comprehensive Investigation of Heterogeneous Systems”. In *8th International Modelica Conference*, edited by C. Clauß, 748–755. March 20th-22nd, Dresden, Germany.

Fritzson, P. 2015. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach.* 2nd ed. Piscataway: IEEE Press, Wiley-Interscience.

Giambiasi, N., B. Escude, and S. Ghosh. 2001. “Generalized Discrete Event Simulation of Dynamic Systems”. *Transactions of the Society for Computer Simulation International* 18(4):216–229.

Gladman, B., M. Duncan, and J. Candy. 1991. “Symplectic Integrators for Long-term Integrations in Celestial Mechanics”. *Celestial Mechanics and Dynamical Astronomy* 52(3):221–240.

Goldblatt, R. 1998. *Lectures on the Hyperreals: An Introduction to Nonstandard Analysis*. Number 188 in Graduate Texts in Mathematics. New York: Springer-Verlag.

Hairer, E., C. Lubich, and G. Wanner. 2005. *Geometrical Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. 2nd ed. Number 31 in Springer series in Computational Mathematics. Berlin: Springer-Verlag.

Hairer, E., and G. Soderlind. 2005. “Explicit, Time Reversible, Adaptive Step Size Control”. *SIAM Journal of Scientific Computation* 26(6):1838–1851.

Henzinger, T. 1996. “The Theory of Hybrid Automata”. In *11th Annual IEEE Symposium on Logic in Computer Science*, 278–292. July 27th-30th, New Brunswick, NJ, USA.

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.

Präehofer, H. 1991. *System Theoretic Foundations for Combined Discrete-Continuous System Simulation*. Ph. D. thesis, University of Linz.

Saiku, D., J. Padilla, R. Gore, H. Herencia-Zapana, and A. Tolk. 2013. “Toward a Formalism of Modeling and Simulation using Model Theory”. *Complexity* 19(3):56–63.

Swope, W., H. Andersen, P. Berens, and K. 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.

Tolk, A., F. Barros, A. D’Ambrogio, A. Rajhans, P. Mosterman, S. Shetty, M. T. H. Vangheluwe, and L. Yilmaz. 2018. “Hybrid Simulation for Cyber Physical Systems - A Panel on where we Are Going Regarding Complexity, Intelligence, and Adaptability of CPS using Simulation”. In *Symposium on Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems*, edited by S. Mittal, J. Risco-Martín, and M. Lützenberger. April 15th-18th, Baltimore, MD, USA.

Tripakis, S., C. Stergiou, C. Shaver, and E. Lee. 2013. “A Modular Formal Semantics for Ptolemy”. *Mathematical Structures in Computer Science* 23(4):834–881.

Vangheluwe, H. 2000. “DEVS as a Common Denominator for Muti-formalism Hybrid Systems Modelling”. In *IEEE Proceedings of the International Symposium on Computer-Aided Control Systems Design*, edited by P. Misra and A. Varga, 129–134. September 25th-27th, Anchorage, AK, USA.

Zeigler, B. 1984. *Multifaceted Modelling and Discrete Event Simulation*. San Diego: Academic Press.

Zeigler, B., and J. Lee. 1998. “Theory of Quantized Systems: Formal Basis for DEVS/HLA Distributed Simulation Environment”. In *Enabling Technology for Simulation Science II*, edited by A. Sisti, Volume 3369 of *SPIE*, 49–58. April 13th-17th, Orlando, FL, USA.

Zeigler, B., H. Praehofer, and T. Kim. 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. 2nd ed. Amsterdam: Academic Press.

AUTHOR BIOGRAPHY

FERNANDO J. BARROS is a professor at the University of Coimbra. His research interests include theory of modeling and simulation, hybrid systems and dynamic topology models. He has authored more than 60 scientific publications and he has been responsible for the organization of several M&S conferences. Fernando Barros is a member of the editorial board *International Journal of Simulation and Process Modelling*, and associate editor of the *International Journal of Agent Technologies and Systems*. His e-mail address is barros@dei.uc.pt.