`vle.extension.differential-equation` : simulation of ordinary differential equations into VLE.

September 9, 2014

Contents

1 Introduction

The package `vle.extension.differential-equation` provides DEVS models into VLE that allow the simulation of ordinary differential equation.

2 Background theory

Three methods are provided for numerical integration: Euler, Runge Kutta 4 and QSS2 [6]. Euler and Runge Kutta 4 are well known explicit and forward methods based on the discretization of the time. They are said to be time-slicing methods (TSM). QSS2 is also a forward and explicit method but it is based on the discretization of variables, and developped especially for DEVS models. For QSS2 implementation, the most general case is considered; ie.

gradient expressions are non linear functions depending on all state variables and thus :

- gradient derivatives are computed numerically and not analytically as proposed for linear gradients.
- quantization of variables involves an update of all variable gradients.

Here, we focus on the strategies for handling discontinuities into the systems and propose one strategy for VLE. These discontinuities are the result of coupling discrete systems whith continuous ones.

2.1 State of the art: time events, state events

Hybrid systems are defined as systems exhibiting both continuous and discrete behaviors [1]. For numerical integration methods developper, these systems present the difficulty of handling discontinuities.

For Cellier and Kofman [3], discontinuities are discrete events and there exists two type of discontinuities :

- state events. E.g. a ball that bounces the soil has a continuous behavior except at the time of contact with the soil, at which the direction is changed. Equations of the ball can be written whith *switching function* like in Dymola modeling environment, such as
  $y' =$ if $y < 0$ then -1 else $+1$
- time events. E.g. a control agent that takes decisions on the conduct of a continuous system.

One of the principal concern for handling state events is to detect them. This can be achieved by using for example step-size control [4] and zero-crossing methods [5]. The family of QSSn methods [2] are well suited for handling state events since they rely on a discretization of state variable rather than discretization of time.

HFSS [1] and DEV&DESS [8] are formalisms that allow the simulation of hybrid systems. DEV&DESS rely on quantization methods and HFSS do not make assumptions on the integration method. It is not clear nevertheless how discontinuities are handled into these frameworks.

For time events handling, no dedicated integration methods have been proposed, only advises are given, as in [3]. If time events are scheduled (known in advance), step-size control can be achieved in order to integrate over these events. In any case, the consequence of a time event should be the beginning of a completely new integration, that necesseraly starts with a re-initialisation of the system of differential equations.

2.2 Proposed strategy for time events management

The strategy proposed for handling perturbations consists in reinitialising the system of differential equation when a perturbation occurs as suggested in [3].

On perturbation, a propagation of the discontinuity is performed into the system.

Figure 1 is an example of the consequence of a discontinuities into a system of differential equations. S is a state variable which gradient expression depends on variable E. At t_p, a perturbation on state variable S occurs and a discontinuity on S is provoked. At t_d, a discontinuity on variables E occurs and provokes a discontinuity on the gradient value of S.

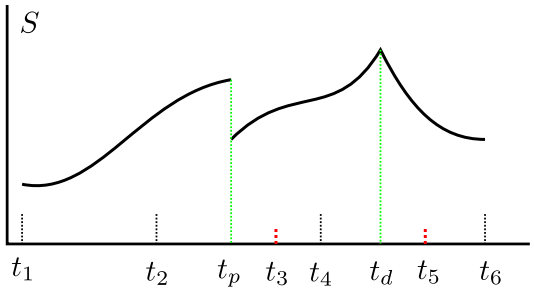

Figure 1: Perturbations management strategy

On this example, the pertubation handling strategy is the following. At time t_2, there is no perturbation yet and the numerical integration processes as usual. Gradients computed at t_1 are used to compute the new value of S and a time step $\Delta_t = t_3 - t_2$ is provided for the duration of integration step. Into DEVS, this consists in calculating the time advance function as $\sigma = \Delta_t$.

At time $t_p \in [t_2, t_3]$, a perturbation occurs on the model and the following steps are performed:

1 Update S values at t_p using a time step of value $t_p - t_2$.

2 Apply perturbation: reset the value of S and his gradient.

3 Propagate a new discontinuity to variables that depends on S in order to reinitialise them at t_p.

4 Compute the new σ (in the figure, $\sigma = t_4 - t_p$)

At time $t_d \in [t_4, t_5]$, similar steps are performed, unless that the discontinuity is propagated and not built:

5 Update S values at t_d using a time step of value $t_d - t_4$.

6 Apply discontinuity: reset the value of S and his gradient.

7 Propagate a discontinuity to variables that depends on S in order to reinitialise them at t_d.

8 Compute the new σ (in the figure, $\sigma = t_6 - t_d$)

Intuitively, a couple P-DEVS model that represents these systems are formed by a set of atomic models, each representing one variable. In the case of a cycle of atomic models, e.g. when S depends on E and E depends on S, using the strategy proposed will lead to discontinuities for S that are simultaneous ($t_d = t_p$) and an illegitimate P-DEVS model [7] will be formed (ie. a infinite succession of bags with $ta = 0$) due to the propagation of discontinuity at t_d (item 7).

To prevent this to happen, we build a discontinuity structure whith a message $< iP, M_d >$ where iP is an identifier of the perturbation and M_d is the set of identifiers of models the perturbation passes through. At time t_d, model S propagates the discontinuity $< iP, M_d \cup \{S\} >$ only if new models identifiers are present into M_d for perturbation iP. On a system of differential equations of n_v variables which is perturbated by n_p events at time t_p, the number of bags with $ta = 0$ dedicated to reinitialisation will be in the worst case $n_v * n_p * (n_v - 1)$ and is finite. Indeed, the worst case is when each variable depends on all others variables of the system. Using this strategy, the P-DEVS model is legitimate.

3 User Documentation

3.1 Atomic model interface

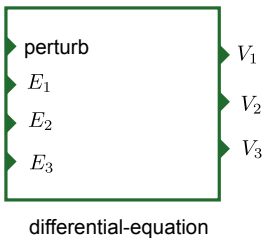

Figure 2: The user interface of an atomic model.

Evolution of state variables V_i is described by differential equation. These expressions can rely on the value of external variables E_i. For time sclicing methods, external variables E_i are expected to be piecewise constant functions computed from continuous and derivable functions. For QSS2, external variables are expected to be piecewise linear functions.

Atomic model ports E_i and V_i can carry data at time t that contain:

- 'name': the name of the external variable (or state variable)
- 'value': this is the value at t of the variable 'name'.
- 'gradient' (optionnal): this is the value of the gradient of the variable 'name' at t.

- 'discontinuities' (the structure proposed in section 2.2): this should not be handled by the user.

No other assumption, than continuity and derivability of the producing function, is made regarding external variables updates. Particularly, updates of E_i variables can occur at any time.

Port 'perturb' can carry instant data (with no assumption on the producing function) that is used to reset some state variables values. Reinitialisation of the state variables and their gradients is achieved by a function *reinit* that the user can override. By default, the pertubation is interpreted as a reset of one state variable, and data must contain:

- 'name': the name of the state variable targeted by the perturbation.
- 'value': the new value at of the state variable 'name'.

Perturbations and discontinuities are handled by the formalism and do not require special care of the user (see section 2.2) unless perturbation is required to behave differently, in which case the function *reinit* should be proposed by the user.

3.2 Writing differential equations atomic models

Below is given an example of dynamic for an atomic model that relies on the `vle.extension.differential-equation`.

```
class MyModel : public DifferentialEquation
{
public:
 MyModel(const DynamicsInit& model,
             const InitEventList& events) :
    DifferentialEquation(model,events)
 {
        //Initialisation of variables is done
        //into the class constructor:
        v = createVar("v");
        e = createExt("e");
 }

 //gradients of state variables are expressed
 //into the 'compute' function
 void compute(const Time& time)
 {
        grad(v) = v() - e();
 }
 //Reinitialisation of state variables after a perturbation or
 //a discontinuity is performed into the 'reinit' function
 //(optionnal)
```

```
 void reinit(bool isPerturb, const Value& message)
 {
 }
 //states and external variables are attributes of the class
 Var v;
 Ext e;
};
```

3.3 Configuring atomics models into vpz conditions

The common structure of the conditions for configuring atomic models is the following:

- 'variables': a map that gives initialisation values of the state variables.
- 'method': the name of the numerical integration method to use for simulation ('rk4','euler' or 'qss2').
- 'method-parameters': a map that contains parameters of the specified method (see above).

Three numerical integration methods are provided Runge Kutta 4 (rk4) Euler (euler) and QSS2 (qss2). These are the parameters specific for each method.

- For rk4 and euler:
  - 'timestep' is a double value that gives the time step for integration.
  - 'synchronisation' (optionnal) is a boolean. If true, integration steps are synchronized with external variables updates.
  - output_period (integer, default=1). It gives the period at which output of variable values are performed. They are output every $output\_period * timestep$ time unit.
- For qss2:
  - 'DeltaQ' is a map that gives for each state variable the quantum for integration.
  - 'expect-gradients' (optionnal) is a boolean. If true, gradients are required for initialisation of state variables and external variables updates.

4 Technical details

Some technical details are given below:

- Description of the dynamic is based on DEVS state graph transition and `confluentTransition` implementation is used (see e.g. DEVS state graph for Time Slicing Methods implementation)

- The base class `DifferentialEquation` implementation is based, as far as possible, on the PIMPL idiom. Thus dynamics can be distinguished (eg. between QSS2 and Time Slicing Methods) and most of bugs corrections would be stable regarding ABI.
- In an optimized use (no external variable and no perturbation), the integration step is ensured to be performed in only one bag.

4.1 Global architecture

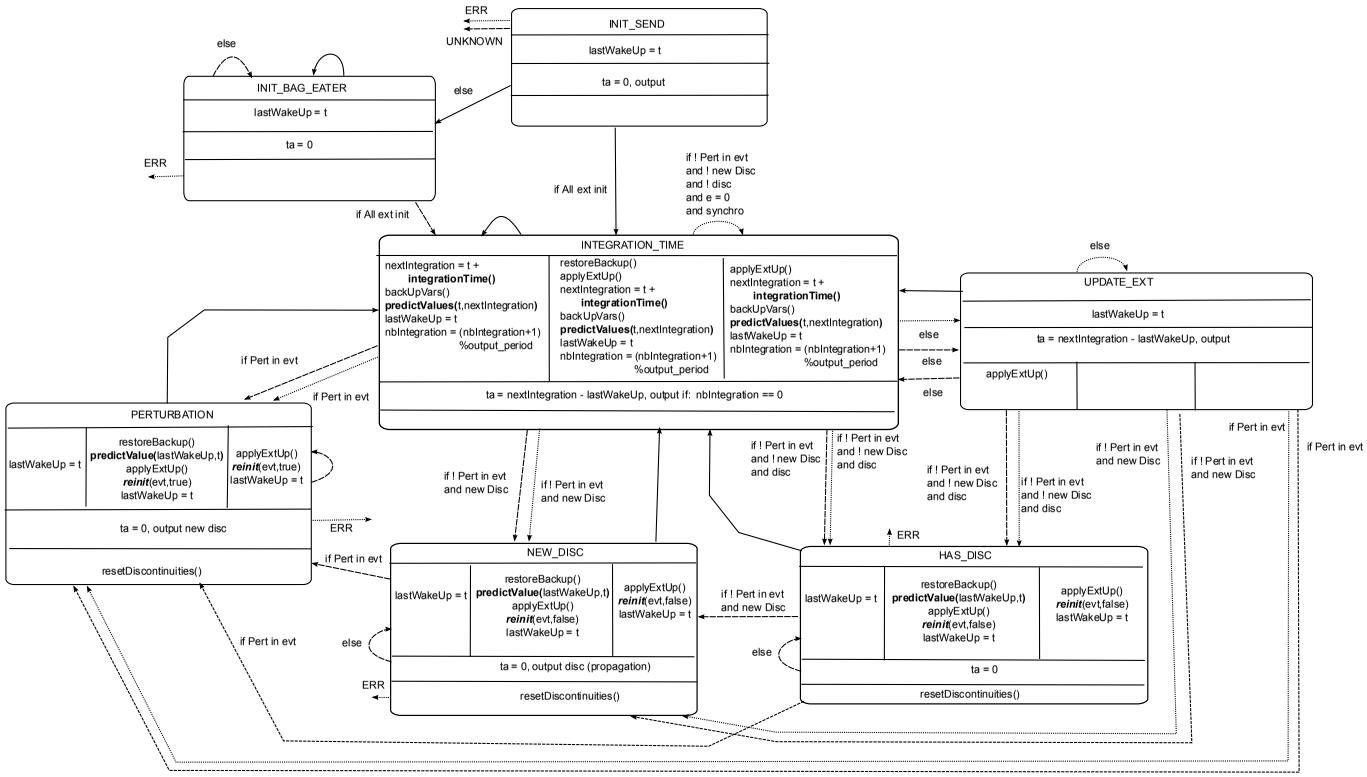

Figure 3: The DEVS state transition graph for Time Slicing methods (TSM).

soft	observation	RK4	QSS2
powerdevs	gnuplot	-	3.86
	null	-	0.76**
deSolve	timed, mem. storage	14.776*	-
VLE	timed, mem. storage	6.468*	10.536
	timed, file storage	3.468	7.812
	null	1.072	4.824**

Figure 4: Comparison of time executions given in seconds.

4.2 Simulation time profiling

Simulation time comparisons are based on the simulation of the Lotka Volterra model and concern the three softwares VLE, powerdevs and R package deSolve. End time of simulation is set to 1500. Two integration schemes are tested:

- RK4 scheme with a time step of 0.01 (for deSolve and VLE)
- QSS2 scheme with a quantum value of 0.0001 (for powerdevs and VLE). Note that a lower quantum value results in a divergence process (and a $ta \to 0$).

Different observations schemes are tested:

- For VLE and powerdevs. A null observation which provides no output data.
- For VLE. A timed observation with a time-step of 0.01 and a storage into a file.
- For VLE. A timed observation with a time-step of 0.01 and a storage into memory. At the beginning, the matrix contains 10000 rows and is updated with 10000 more rows each time it is required to enlarge the matrix (these are the maximal values for VLE).
- For deSolve. A timed observation with a time-step of 0.01 and a storage into a R matrix.
- For powerdevs. A plot using gnuplot of quantized values.

Values quoted ($*$) can be used to compare deSolve and VLE. This first experiment shows better results for VLE. Moreover, VLE results should be improved if initialization size of the matrix is directly set to the final size which is 150000 (rather than 10000). Values quoted ($**$) can be used to compare powerdevs and VLE and show clearly that powerdevs is faster.

soft	observation	RK4	QSS2
powerdevs	gnuplot	-	3.86
	null	-	0.76**
deSolve	timed, mem. storage	14.776*	-
VLE	timed, mem. storage	0.74*	1.07
	timed, file storage	0.94	1.28
	null	0.1	0.37**

Figure 5: Comparison of time executions given in seconds. New TO BE confirmed?!!.

References

[1] Fernando J. Barros. Dynamic structure multiparadigm modeling and simulation. *ACM Transactions on Modeling and Computer Simulation*, 13:259–275, July 2003.

[2] F. Cellier, E. Ernesto Kofman, G. Migoni, and M. Bortolotto. Quantized state system simulation. In *Proceedings of Summer Simulation Multiconference 08*, 2008.

[3] F.E. Cellier and E. Kofman. *Continuous system simulation.* Springer, 2006.

[4] Joel M. Esposito and Vijay Kumar. A state event detection algorithm for numerically simulating hybrid systems with model singularities. *ACM Trans. Model. Comput. Simul.*, 17, January 2007.

[5] Mao G. and Petzold L.R. Efficient integration over discontinuities for differential-algebraic systems. *Computers and Mathematics with Applications*, 43(1):65–79, 2002.

[6] E. Kofman. A second order approximation for devs simulation of continuous systems. *Simulation (Journal of The Society for Computer Simulation International)*, 78(2):76–89, 2002.

[7] B. P. Zeigler. *Theory Of Modeling and Simulation.* Krieger Publishing Compagny, 1984. 2nd Edition.

[8] Bernard P. Zeigler. Embedding dev&dess in devs. In *DEVS Integrative M&S Symposium (DEVS'06)*, April 2006.