Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 48-1 (2015) 262–267

Efficient Computation of State Derivatives for Multi-Rate Integration of Object-Oriented Models

Francesco Casella [*]

[*] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy (e-mail: francesco.casella@polimi.it).*

Abstract: Multi-rate integration algorithms offer huge advantages in terms of simulation time savings when large, loosely coupled systems or systems with mixed fast and slow dynamics are considered. Such algorithms require the computation of different sub-sets of the state derivative vector at each time step. This paper presents a method to efficiently compute these sub-sets, starting from a continuous-time, object-oriented dynamic model. The method is demonstrated on a simple test case.

© 2015, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Object-oriented modelling, multi-rate integration algorithms, simulation tools

1. INTRODUCTION

Equation-based, object-oriented modelling languages and tools have now established themselves as the best option for the system-level modelling of physical and cyber-physical systems, in particular when focusing on control system studies and when multiple physical domains are involved. In this field, the Modelica language (Mattsson et al. (1998)) has emerged as a de-facto standard and is supported by an increasing number of commercial and open-source simulation tools.

Limiting our analysis to purely continuous-time models without events for simplicity, object-oriented models, once reduced to flat form, are equivalent to systems of differential-algebraic equations (DAE). The typical strategy followed by Object-Oriented (OO) tools to simulate such models, in particular by Modelica tools such as Dymola or OpenModelica, is to first symbolically remove trivial equations such as $a \pm b = 0$ and $a = \pm b$ from the system, and possibly to apply more sophisticated symbolic simplifications. If necessary, they also symbolically reduce its structural index to 1 by means of Pantelides' algorithm (Pantelides (1988)) and of the Dummy Derivatives algorithm (Mattsson and Söderlind (1993)), in case of high-index systems. Assuming a fixed set of states can be chosen, the system model is thus brought to a simpler, index-1 DAE form

$$F(x, \dot{x}, v, t) = 0,$$

where $F(\cdot)$ is a vector function, x is the vector of state variables, v the vector of algebraic (i.e., non-state) variables and t is time. A procedure is then established to solve the DAE system for $(\dot{x}, v)$ assuming (x, t) known. In order to do so, equations and variables are re-ordered so that their incidence matrix is in block-lower-triangular (BLT) form. Each block on the diagonal corresponds to a smaller system of equations to be solved (either symbolically or numerically), following an ordered sequence and exploiting the previously computed values of variables appearing earlier in the sequence. Thus, the system is conceptually brought into state-space form with explicit ordinary differential equations (ODEs) and output equations

$$\dot{x} = f(x, t)$$
$$v = g(x, t).$$

The computational procedure that corresponds to the computation of $f(x,t)$ and $g(x,t)$ is then linked to an ordinary differential equation solver in order to numerically compute the system transients.

State-of-the-art algorithms employed for this purpose are currently of the single-rate type: based on the known value of the state vector at a certain time $x(t)$, and possibly also on the values of previous time steps in case of multi-step algorithms, the value at the next time step $x(t + h)$ is computed, where the step length h can be either fixed or selected in order to comply with some pre-defined error tolerance bound. The key feature of all single-rate ODE integration algorithms is that the *entire* vector of derivatives $f(x,t)$ is evaluated at once at each time step t, and possibly at other intermediate time values in the case of high-order algorithms.

As the size of the system grows, this approach can become increasingly inefficient for two classes of models. The first is given by loosely coupled interconnections of sub-systems with asynchronous localized activity, such as smart grids, thermal or electrical power distribution networks, etc. The second is given by the interconnection of slow, but computationally demanding, sub-systems with fast, but easy to compute, subsystems. One example are models of steam boiler-turbine units coupled to the electrical generator and transmission line models to form comprehensive power system descriptions.

In both cases, assuming adaptive error control is used, when the system shows fast changes in one of its local or fast subsystems, short time steps are taken to limit the error below the given tolerance, and this requires recomputing the *entire* derivative vector $f(x,t)$ at very closely spaced time steps. This can be extremely inefficient,

2405-8963 © 2015, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2015.05.055

as the derivative of all other slow and/or loosely coupled sub-systems will hardly change over such a short time span, so that most of those computations will not in fact bring any new useful information to the solver. The problem is particularly severe if the computation of $f(x,t)$ is very CPU-intensive, as in the case of thermo-fluid systems with sophisticated fluid models, and becomes worse as the size of the model increases. Furthermore, in the case of implicit algorithm, the computation and inversion of the *entire* Jacobian $\partial f/\partial x$ is also performed. The end result is that the simulation of larger models becomes infeasible even if modern CPUs and state-of-the-art tools are employed.

A recent paper by Ranade and Casella (2014) proposes to introduce multi-rate algorithms in these scenarios, where state-of-the-art object-oriented tools coupled to state-of-the-art single-rate integration algorithms fail to provide adequate performance. Multi-rate algorithms have been studies since the early sixties, see, e.g., Rice (1960), Gear and Wells (1984), Engstler and Lubich (1997), Savcenco et al. (2007). Rigorous proofs exist regarding their stability, convergence and accuracy properties, which are outside the scope of this paper. However, thes algorithms have not found application in the context of general-purpose object-oriented system modelling so far.

The basic idea of these algorithms is that only for those state variables whose integration error estimation exceeds the tolerance, the computation is refined on a finer time grid, while the values of states whose error is below the threshold can be estimated by means of interpolation. In this way, the expensive and useless re-computation of a large portion of the vector $f(x,t)$ (and, possibly, of its Jacobian) is avoided.

A representative case study presented in Ranade and Casella (2014) shows that the number of evaluations of individual components of $f(x,t)$ during the simulation of a transient scales up with the square of the system size, compared to the cubic increase obtained by standard single-rate algorithms. It is then apparent that these methods can provide huge advantages in simulation speed as the size and degree of detail of the model increases.

A basic requirement for the use of multi-rate algorithms is that at each time step, only a sub-set of the state derivative vector $f(x,t)$ needs to be computed. Furthermore, the elements of this set can change at each time step in an unpredictable way. This is trivial to obtain if the system is originally described by explicit ODEs, but not in the case of generic object-oriented models, which are given by implicit DAEs. On the other hand, the well-established procedure mentioned above to get the ODEs out of the DAEs (see, e.g., Cellier and Kofman (2006) for further details) is meant to efficiently compute the *entire* vector $\dot{x}$ at once.

The goal of this paper is thus to show how to efficiently compute *any sub-set* of that vector, starting from an object-oriented model, thus enabling the use of multi-rate integration algorithms for the simulation of object-oriented models. The ideas shown in this paper could be used as a basis for the implementation within any OO simulation tool.

The paper is structured as follows: Section 2 reviews the basic concepts of multi-rate integration algorithms; Section 3 describes the procedure to efficiently compute any subset of state derivatives, while Section 4 illustrates the application of the proposed procedure to an example case. In Section 5, concluding remarks and future research direction are given.

2. MULTI-RATE INTEGRATION

The basic idea of a multi-rate integration algorithm is to split the system

$$\dot{x} = f(x,t) \tag{1}$$

into a partitioned system characterized by *active* states x_a and of the *latent* states x_l

$$\begin{aligned}\dot{x_a} &= f_a(x_a, x_l, t)\\ \dot{x_l} &= f_l(x_a, x_l, t).\end{aligned}$$

The partitioned system is then integrated with a smaller time step h_a for the active part and with a larger time step h_l for the latent part. The coupling between the two parts is handled by interpolations with high enough order to guarantee the required precision. This idea can be applied to a wide range of different integration algorithms, both single-step (e.g., Runge-Kutta) and multi-step (e.g. BDF).

If the dynamics of the system is well known, one can think of partitioning the set of states into two sub-sets which are defined a priori. However, this requires the end user to provide information to the object-oriented simulation tool that is not trivial to obtain in general. The most suitable multi-rate algorithms for the integration of OO models are then those equipped with automatic error control. The basic ideas behind those algorithms are sketched here for the reader's convenience, for further details see Ranade and Casella (2014) and Savcenco et al. (2007).

A tentative solution of the entire system of ODEs is first computed from t_{n-1} to t_n with a global time step h. A single-step method with embedded lower-order method can be used to estimate the local integration error *for each component of the state vector*. Those components whose error is below the threshold are put in the latent partition, while the other ones are refined by re-solving the active part using two steps of lenght $h/2$, and using interpolation to obtain the required values of x_l at the intermediate time steps. The procedure can then be recursively iterated: after refining the grid on the active component, errors are evaluated once more for each state variable; those for which the error still exceed the threshold are further refined, while the others end up in the latent partition. The procedure is repeated until no further refinements are necessary, then it can resume with another global time step.

As already noted in the Introduction, each refinement step for the active part will require one or more (depending on the order of the algorithm) evaluations of a subset of the state derivatives, using interpolated values for the latent states. If the model being simulated is an object-oriented one, i.e., it is defined by DAEs, it is in general not a good idea to do this with procedures that correspond to the *explicit* computation of each such derivatives, because this might lead to a lot of duplicate computations, and to a very inefficient procedure, and might not always be possible for

Download English Version:

https://daneshyari.com/en/article/713593

Download Persian Version:

https://daneshyari.com/article/713593

Daneshyari.com