LUND UNIVERSITY

Efficient Predictor for Co-Simulation with Multistep Sub-System Solvers

Andersson, Christian; Führer, Claus; Åkesson, Johan

2016

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Andersson, C., Führer, C., & Åkesson, J. (2016). *Efficient Predictor for Co-Simulation with Multistep Sub-System Solvers*. (Technical Report in Mathematical Sciences; Vol. 2016, No. 1). Centre for Mathematical Sciences, Lund University.

Total number of authors:
3

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Efficient Predictor for Co-Simulation with Multistep Sub-System Solvers

Christian Andersson, Claus Führer, Johan Åkesson

Preprints in Mathematical Sciences
2016:1

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Numerical Analysis

CENTRUM SCIENTIARUM MATHEMATICARUM

Efficient Predictor for Co-Simulation with Multistep Sub-System Solvers

Christian Andersson[a,b], Claus Führer[a], Johan Åkesson[b]

[a] *Centre for Mathematical Sciences, Lund University, Lund, Sweden*
[b] *Modelon AB, Lund, Sweden*

Abstract

An industrial model of a dynamic system is usually not just a set of differential equations. External inputs acting on the system are common, such as an external force acting on a body or wind pressing on a car. Update of these inputs needs to be handled by the numerical solver in an efficient way.

In dynamical simulation, multistep methods are commonly used. A multistep method uses the solution history in order to predict the future solution. When an input is changed, the history is no longer a good approximation for the future solution which may result in order reductions and simulation failure.

In this paper, a modification of the predictor is presented. Modifying the predictor, instead of restarting the method, results in an increased performance of the method. The cost of the modification must be weighed with the cost of restarting the method. Experiments show that the benefit of modifying the predictor outweighs the cost of a restart.

Keywords: IVP, Multistep Methods, Restart, Functional Mock-up Interface; Simulation;

1. Introduction

Simulation of weakly coupled dynamical systems, commonly co-simulation, where each subsystem is bundled with an internal solver, is an important industrial method to support model-based design workflows. For complex systems with different parts modeled in different simulation tools this is the only viable option. In this setting, the dynamics of each system is hidden and information between subsystems is exchanged through sampled inputs and outputs at specified global time points. Consider Figure 1, each of the systems, engine, hydraulics and air-conditioning are separate and include an integrator.

Between the global time points, the internal integrators run independently of each other. At a global time point, inputs and output are exchanged between

Email addresses: chria@maths.lth.se (Christian Andersson), claus@maths.lth.se (Claus Führer), johan.akesson@modelon.com (Johan Åkesson)

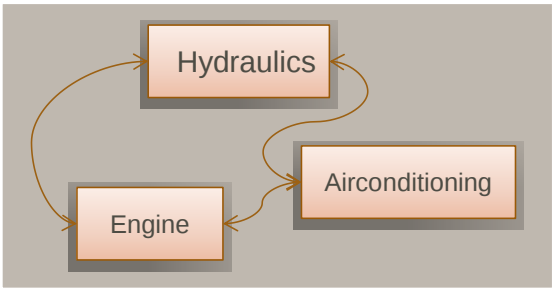

Figure 1: Coupling of three separate systems where each system includes an integrator.

the models and during the next step, the inputs in a subsystem are either kept constant or extrapolated. Furthermore, the updated inputs are typically not consistent with the previous inputs and thus a discontinuity is introduced, cf. Figure 2. This discontinuity introduces difficulties for the internal integrator.

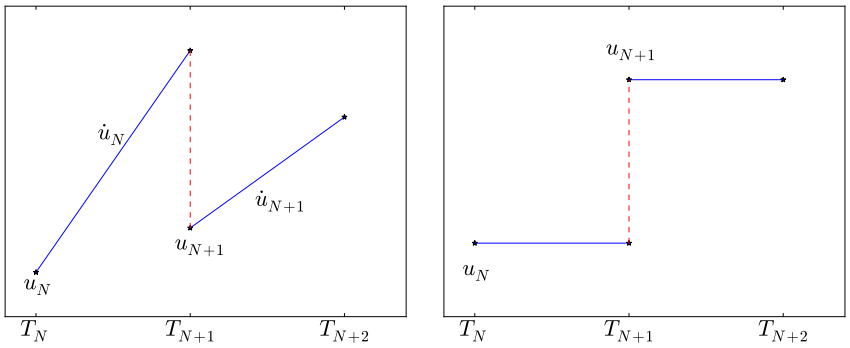

Figure 2: Example inputs. In the left figure, a piecewise linear signal is shown with a discontinuity at a global time point (T_{N+1}) and in the right figure, a constant signal.

A common underlying integrator in a subsystem is a multistep method [1]. A multistep method uses the solution from previous steps to predict future solutions. It is usually implemented using a variable time grid, which means that the step size of the method adapts to the problem during the simulation. Additionally this is complemented with adapting the order of the method.

Restarting a multistep method used for solving initial value problem is expensive. A restart resets the current integrator order to one and discards the information from previous steps. In certain situations this is necessary, such as when a discontinuity in the states are detected. Here, we intend to use the knowledge of what has changed in the problem and modify the method so that a restart becomes unnecessary.

Instead of considering a fully coupled system, we consider an initial value

problem (IVP) together with an external input $u(t)$,

$$\dot{x}(t) = f(x(t), u(t)), \quad x(T_0) = x_0, \quad t \in [T_0, T_M]. \tag{1}$$

The input is assumed to be a given piecewise constant signal, on a global grid, defined as,

$$u(t) = u_i, \quad t \in [T_i, T_{i+1}) \tag{2}$$

with $i = 0, \ldots, M-1$. Alternatively, we consider a piecewise linear input signal, possibly discontinues,

$$u(t) = u_i + (t - T_i)\dot{u}_i, \quad t \in [T_i, T_{i+1}]. \tag{3}$$

The IVP (1), is solved using a variable-step, variable-order multistep method. The assumption is that during a global step, i.e., $t \in [T_i, T_{i+1}]$, the method requires many local steps to satisfy the tolerance requirements.

Simulation of coupled systems is highly relevant in the context of the Functional Mock-up Interface [2]. Another situation where the above is relevant is the case when the IVP is coupled to an external process which is responsible for providing the inputs. Consider a hardware-in-the-loop simulation [3], here the inputs are not known for the complete simulation horizon at initial time (due to that they do not exist), but rather at known time-points where an update occur.

There are two approaches that first come to mind when simulating the IVP using the discussed inputs. Either the method is restarted at each segment or the method proceeds using values computed from the previous segment. In Figure 3, a typical plot of the step size history and order history is shown for the two cases. As can be seen, neither are efficient. Both approaches experience order and step size reductions at the global input changes.

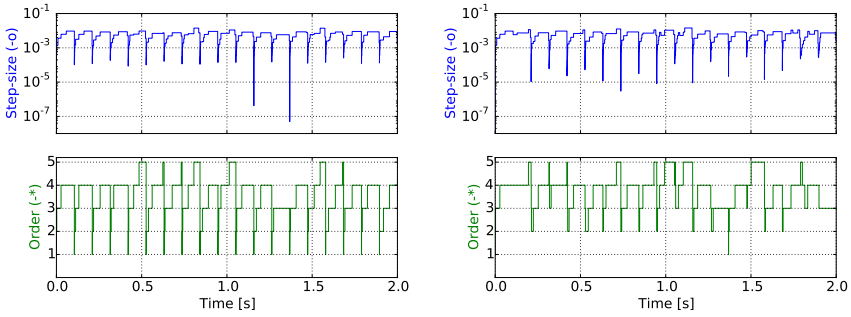

Figure 3: Step size and order for a simulation of an example model using linear input segments. In the left figure, the method is restarted at every global step while in the right figure, it proceeds from stored values.

In this article, an approach is presented that modifies the predictor in a multistep method at the start of each global step. The modification is shown to significantly improve the simulation performance.

2. Multistep methods

A general multistep method for solving an IVP is defined as,

$$\sum_{i=0}^{q} \alpha_{n,i} x_{n-i} + h_n \sum_{i=0}^{q} \beta_{n,i} f(x_{n-i}) = 0 \tag{4}$$

where q determines the number of steps, i.e. the number of previous solution points used, in the formula. The coefficients α and β determine the method. They are dependent on the step size history and order.

A multistep method needs during the integration access to the previous solutions points, i.e. the solution history. The representation of the history varies between implementations of a multistep method. Most commonly used history representation is via a Nordsieck array or an array of modified divided differences. For a detailed description cf. [4].

In a Nordsieck history array, the history is represented as,

$$z_{n-1} = \Big[x_{n-1}, h_n \dot{x}_{n-1}, \ldots, \frac{h_n^q x_{n-1}^{(q)}}{q!}\Big]. \tag{5}$$

From the solution history z_{n-1}, a prediction to z_n, denoted $z_{n(0)}$, and also x_n ($x_{n(0)}$), is computed. This prediction is used as an initial guess to the nonlinear equation system, resulting from Equation 4,

$$G(x_n) = 0. \tag{6}$$

In order to accept the computed solution x_n, an error test must be passed,

$$||e_n|| = ||x_n - x_{n(0)}|| \leq \text{TOL}. \tag{7}$$

If the error test succeeds for a given tolerance TOL the step is accepted. The above steps are independent of how the solution history is represented.

3. Problem formulation

Here we consider the IVP, Equation 1, with either a piecewise constant input or a piecewise linear input resulting from a co-simulation. At the start of a global segment, $[T_{i+1}, T_{i+2}]$, the solution history of the multistep method is based on,

$$\dot{x} = f(x, u_i), \quad t \in [T_i, T_{i+1}]. \tag{8}$$

Once the inputs are updated, at T_{i+1}, the equation,

$$\dot{x} = f(x, u_{i+1}) \tag{9}$$

are no longer consistent,

$$\dot{x}_{i+1}^{-} := f(x_{i+1}, u_i,) \quad \neq \quad f(x_{i+1}, u_{i+1}) =: \dot{x}_{i+1}^{+}. \tag{10}$$

This means that the solution history is no longer valid and that the the error test (Equation 7) is likely to fail due to a poor prediction. Additionally, there is also the problem that the predicted step delivers a bad initial guess for the nonlinear system (Equation 6), resulting in convergence failures.

4. Modifying the predictor

As previously mentioned, most of the order reductions and step size reductions seen are due to error test failures caused by a poor prediction.

The predicted next step is computed by extrapolating the polynomial defined with the solution history array. By modifying the history array, a better prediction is achieved resulting in a reduced risk for error test failures and convergence failures.

Considering specifically the Nordsieck representation, Equation 5, the second term in the array is,

$$z_{i+1,1} = h\dot{x}_{i+1}. \tag{11}$$

Here, a correction is computed as,

$$\Delta\dot{x}_{i+1} = \dot{x}^{-}_{i+1} - \dot{x}^{+}_{i+1} \Rightarrow z_{i+1,1} = z_{i+1,1} - h\Delta\dot{x}_{i+1} \Rightarrow z_{i+1,1} = \dot{x}^{+}_{i+1}. \tag{12}$$

A single additional function evaluation is required in order to correct the first derivative. For the second derivative,

$$\Delta\ddot{x}_{i+1} = \ddot{x}^{-}_{i+1} - \ddot{x}^{+}_{i+1} = \ddot{x}^{-}_{i+1} - \Big[\frac{\partial f(x_{i+1}, u_{i+1})}{\partial x}\dot{x}^{+}_{i+1} + \frac{\partial f(x_{i+1}, u_{i+1})}{\partial u}\dot{u}_{i+1}\Big] \tag{13}$$

we correct by,

$$z_{i+1,2} = z_{i+1,2} - \frac{h^2}{2}\Delta\ddot{x}_{i+1} \Rightarrow \tag{14}$$

$$z_{i+1,2} = \Big[\frac{\partial f(x_{i+1}, u_{n+1})}{\partial x}\dot{x}^{+}_{i+1} + \frac{\partial f(x_{i+1}, u_{i+1})}{\partial u}\dot{u}_{i+1}\Big]. \tag{15}$$

Correcting the second derivative requires a new Jacobian evaluation. Additionally, if the inputs are linear segments, an evaluation of the partial derivatives with respect to the inputs is necessary.

Higher order corrections are considered to costly and not considered here.

5. Experiments

In the experiments below, the models were modeled in the modeling language Modelica [5] and compiled into Functional Mock-Up Units (FMUs) [6] using the open-source tool JModelica.org [7]. The multistep method used was CVode [8] to which the correction, discussed in Section 4, has been implemented. No changes were made inside CVode. Further, the correction to the predictor does not disable any of the features in CVode for step size reductions or order reductions for cases when the error test still fails or when the nonlinear solver fail to converge.

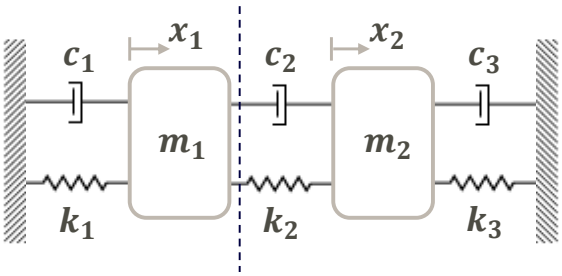

Figure 4: A triple spring, triple damper example.

Table 1: Parameters used in Section 5.1.

$k_1 = 10\ \mathrm{Nm}^{-1}$	$k_2 = 25\ \mathrm{Nm}^{-1}$	$k_3 = 50\ \mathrm{Nm}^{-1}$	$m_1 = 1\ \mathrm{kg}$
$c_1 = 1\ \mathrm{Nsm}^{-1}$	$c_2 = 0.1\ \mathrm{Nsm}^{-1}$	$c_3 = 2\ \mathrm{Nsm}^{-1}$	$m_2 = 1\ \mathrm{kg}$

5.1. Triple spring, triple damper

Consider a linear spring-damper problem with three coupled springs and three coupled dampers which are connected between two fixed points and two masses, cf. Figure 4. The problem is governed by the equations,

$$m_1\ddot{x}_1 = -(k_1 + k_2)x_1 - (c_1 + c_2)\dot{x}_1 + k_2x_2 + c_2\dot{x}_2 \tag{16a}$$
$$m_2\ddot{x}_2 = -(k_3 + k_2)x_2 - (c_3 + c_2)\dot{x}_2 + k_2x_1 + c_2\dot{x}_1. \tag{16b}$$

The parameters are listed in Table 1.

Dividing the system into two sub-systems along the dotted line in Figure 4 results in that the dynamics for the right sub-system is governed by,

$$m_2\ddot{x}_2 = -(k_3 + k_2)x_2 - (c_3 + c_2)\dot{x}_2 + k_2u_{21} + c_2u_{22} \tag{17a}$$

where u_{21} and u_{22} are inputs.

In the following experiments, Equation 16, was solved with high accuracy and used as the reference solution. Furthermore, the inputs to the right sub-system were computed from the reference solution. In Figure 5 the inputs are shown when using piecewise linear segments.

The right sub-system, Equation 17, was simulated for two seconds using twenty piecewise linear segments. The tolerances were set to 10^{-6} for both the relative and absolute tolerance. In Figure 6 and Figure 7, the step size history and order history is shown when using different approaches for handling the crossing of segments. In Table 2 the simulation statistics is shown.

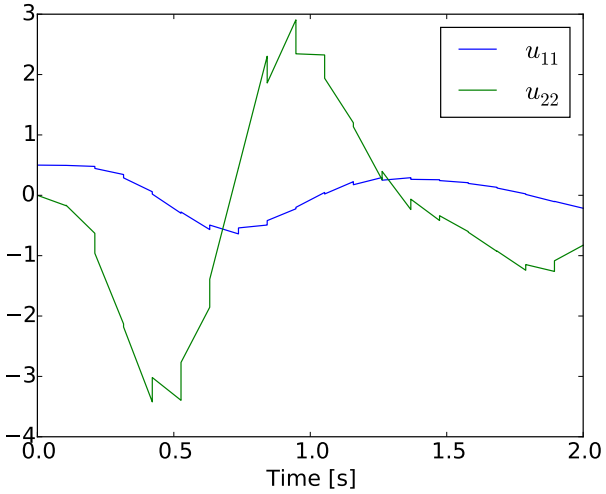

Figure 5: The piecewise linear inputs to Equation 17.

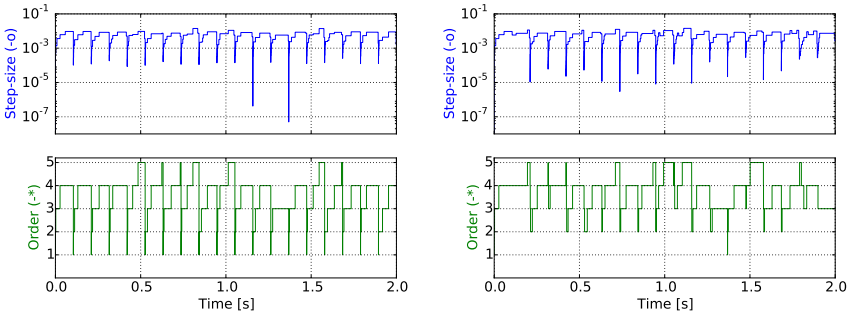

Figure 6: Step size history and order history when simulating Equation 17 using linear input segments. The figure to the left show a simulation when CVode is restarted at each global step while the right show a simulation when CVode proceeds without any modifications.

From the statistics we draw the conclusion that a correction in both the first and second derivative is beneficial as it reduced the number of function evaluations and number of steps taken as compared with the restart case. Additionally, the worst choice is to proceed with old values. Considering the figures, we note that CVode, in the corrected case, persistently remain at high order. The global error is on the same magnitude and are nearly equal in all four cases.

5.2. Coupled pendula

Consider two pendula coupled via a spring, cf. Figure 8. For a detailed description, cf. [9]. The pendula are described in polar coordinates as,

$$\dot{x}_1^{[i]} = x_2^{[i]} \tag{18a}$$

$$\dot{x}_2^{[i]} = (-\mathrm{g} + u_3^{[i]})\sin(x_1^{[i]}) + (u_1^{[i]} + u_2^{[i]})\cos(x_1^{[i]}) \tag{18b}$$

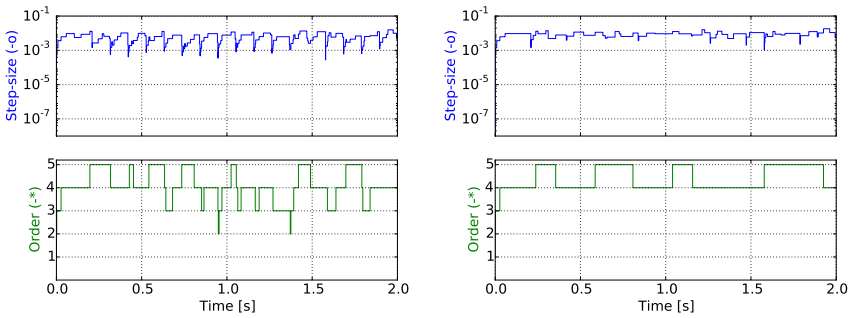

Figure 7: Step size history and order history when simulating Equation 17 using linear input segments. The figure to the left show a simulation when the predictor is corrected in the first derivative while the right figure show when the predictor is corrected in the first and second derivative.

Table 2: Statistics from simulating Equation 17 with different options for crossing the global segments. The elapsed time is normalized with respect to the restart option.

	Restart	Proceed	Pred. Corr. in $\dot{x}$	Pred. Corr. in $\dot{x}, \ddot{x}$
# steps	598	876	584	343
# fevals	847	1257	833	466
# jacs	19	17	11	19
# errfails	22	87	47	18
time	1.00	1.22	0.85	0.59

for $i = 1, 2$. The inputs to the pendula are external excitation forces acting on the pivot $u_1^{[i]}$ and the inputs $u_2^{[i]}$ and $u_3^{[i]}$ are computed through the spring coupling with the coupled pendula. As in the previous example, we consider only part of the full system, the left pendulum ($i = 1$), and regard the inputs as known. The known inputs are computed from a simulation of the fully coupled system.

The left pendulum was simulated for two seconds with forty constant segments for the three inputs. The tolerances were set to 10^{-6} for both the relative and absolute tolerance. In Table 3 the simulation statistics is shown and in Figure 9 and Figure 10 the step size histories and order histories are shown. In Figure 11, the predictor polynomial for a time step is shown to illustrate the impact of the correction.

Table 3: Statistics from simulating Equation 18 with different options for crossing the global segments. The elapsed time is normalized with respect to the restart option.

	Restart	Proceed	Pred. Corr. in $\dot{x}$	Pred. Corr. in $\dot{x}, \ddot{x}$
# steps	1377	1962	1275	850
# fevals	1989	2858	1792	1144
# jacs	55	38	24	39
# errfails	69	210	102	42
time	1.00	1.33	0.88	0.67

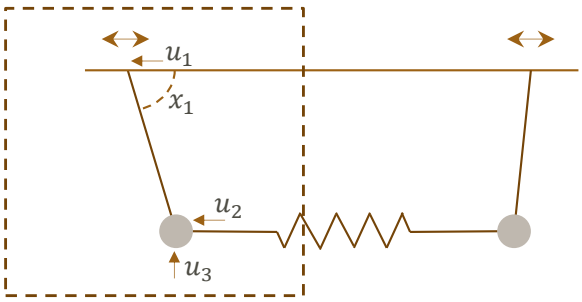

Figure 8: Two coupled pendula with a spring. The dashed square is the model considered in Equation 18.

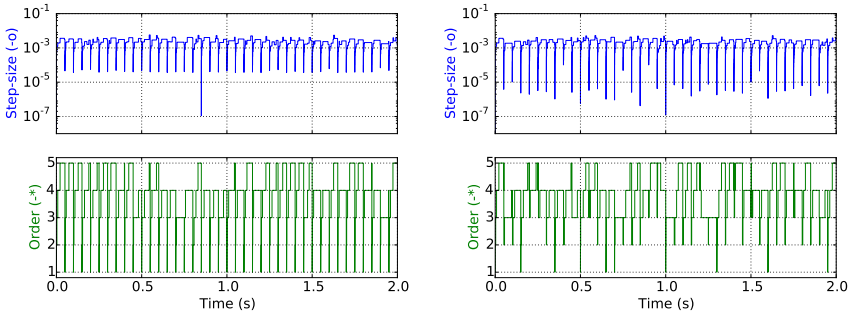

Figure 9: Step size history and order history when simulating Equation 18 using constant input segments. The figure to the left show a simulation when CVode is restarted at each global step while the right show a simulation when CVode proceeds without any modifications.

Again, as in the previous example, we draw the conclusions, that a correction in both the first and second derivative is beneficial as it reduced the number of function evaluations and number of steps taken as compared with the restart case. Additionally, the worst choice is to proceed with old values. Considering the figures, we note that CVode, in the corrected case, persistently remain at high order. The global errors are on the same magnitude and are nearly equal in all four cases.

5.3. Race car

For racing applications, finding the maximal performance of the car is crucial. One method to quickly estimate the impact on performance of a change to the vehicle setup is to solve for the steady state limits under different driving conditions. Identifying a set of critical points along a race track and calculating the maximum achievable speed for each point can give a good indication on how the change will affect the lap time. To investigate the dynamic response, simulations can be carried out with predefined input or by a feedback loop using either a simulator or a virtual driver model. Here, a car is driven by a virtual driver that tries to stay onto an eight shaped course with increasing velocity in

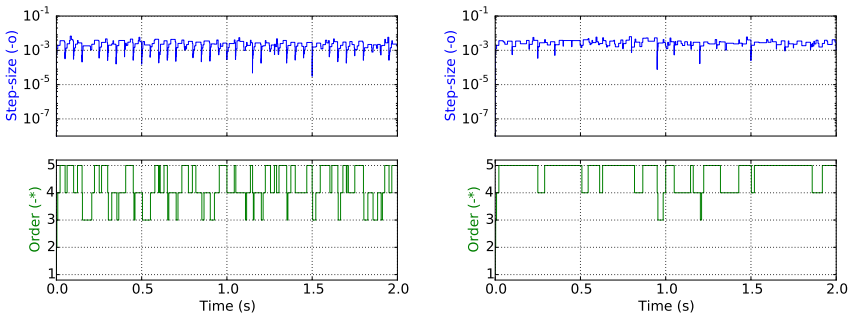

Figure 10: Step size history and order history when simulating Equation 18 using constant input segments. The figure to the left show a simulation when the predictor is corrected in the first derivative while the right figure show when the predictor is corrected in the first and second derivative.

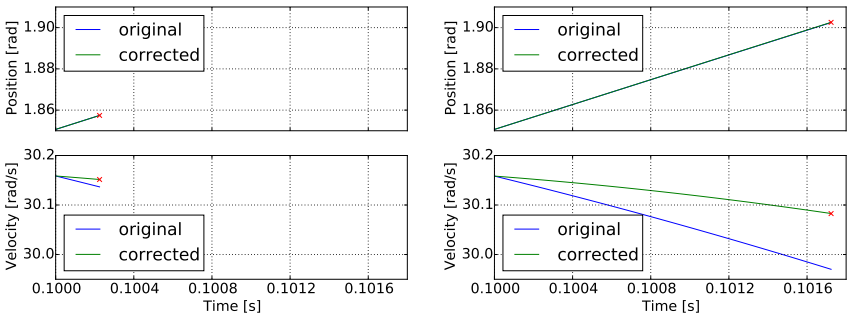

Figure 11: Predictor trajectories at $t = 0.1s$ extrapolated to the next accepted solution when simulating Equation 18 using constant input segments. The figures show the original (non-corrected) predictor polynomial together with the corrected predictor polynomial. The figure to the left uses first order correction while the second uses first and second order correction.

order to investigate the dynamic response of the car, especially when changing the turning direction.

In this example, a race car is modeled in Modelica and exported as an FMU using Dymola [10]. Additionally, a separate Modelica model of the wheels used in the race car is exported from JModelica.org. The interest is in the right front wheel, cf. Figure 13, and the impact of the corrections to the predictor on the simulation. In each wheel, there are 37 inputs. As before, CVode is used, in the wheel, with a relative and absolute tolerance set to 10^{-6} and 200 global segments. The reference trajectories was computed using Assimulo [11] with the solver Radau5 [12] together with a relative and absolute tolerance set to 10^{-10}. The input trajectories for the wheel were computed with a simulation of the full race car.

The wheel is simulated used the different approaches for when crossing a global segments. In Figure 13 the state trajectories are shown together with the error in the states for when using the different approaches. The figure show that the error is on the same magnitude for all approaches, i.e. at the requested

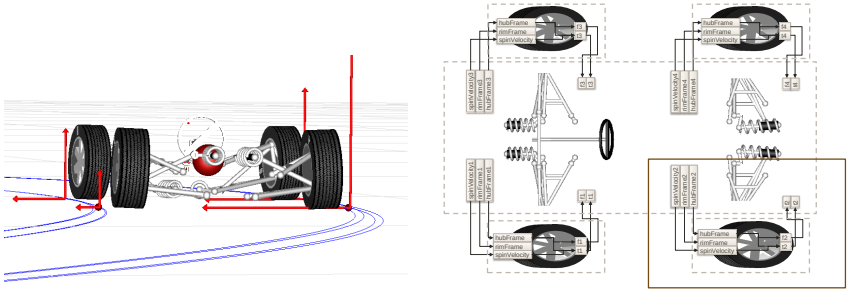

Figure 12: Illustration of the couplings in the race car model from Section 5.3. © Modelon.

accuracy, and thus the error is not impacted by the correction. In Table 4

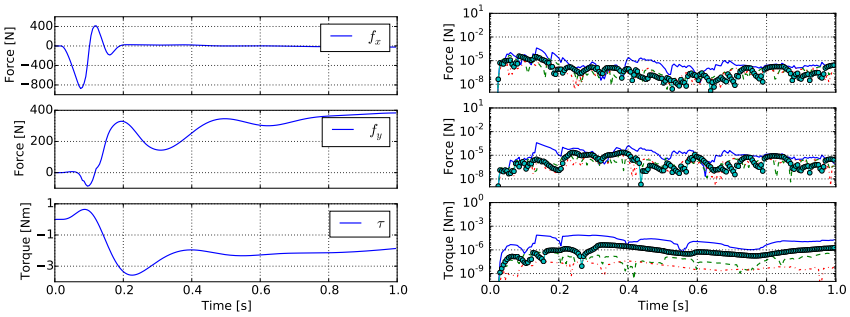

Figure 13: In the left figure, the reference state trajectories when simulating the right front wheel in the race car example, Section 5.3. In the right figure, the state errors when using the different approaches for crossing the global segments. The solid line represents a proceeded simulation, the dashed line represents a correction in the first derivative, the dashed dot in the second while the circled represents a restarted simulation.

the simulation statistics are shown. The statistics show a clear decrease in the number of steps taken when using the corrections for the predictor. However, for the second order correction, the cost of computing a new Jacobian has to be weighed against the reduction in the number of steps. For this case, a second order correction is beneficial.

6. Conclusions

In this article, we presented efficient restart of the multistep method CVode in the context of the Functional Mock-up Interface and co-simulation FMUs. Modifications to the predictor is computed when inputs are set instead of restarting the multistep method. The approach show a significant reduction of work necessary for computing the solution trajectories. The method has been implemented in the open-source tool JModelica.org.

Table 4: Statistics from simulating the right front wheel in the race car example, Section 5.3, with different options for crossing the global segments. The elapsed time is normalized with respect to the restart option.

	Restart	Proceed	Pred. Corr. in $\dot{x}$	Pred. Corr. in $\dot{x}, \ddot{x}$
# steps	3500	5278	3043	1598
# fevals	6670	8843	5048	2970
# jacs	199	105	59	195
# errfails	323	844	384	70
time	1.00	1.29	0.77	0.47

If the structure of a given problem is available, further improvements to the correction can be made, for instance if the problem is linear in the states. In the tool JModelica.org, this information can be made available and should be considered for future improvements. Secondly, if the dependency information between state derivatives and inputs are given, then only a partial update of the partial derivatives might be necessary. Finally, further investigations into wheter or not an update of the Jacobian is required at a given global segment is needed.

Acknowledgements

This work was supported in part by the Lund Center for Control of Complex Engineering Systems (LCCC), funded by the Swedish Research Council, and is gratefully acknowledged.

References

[1] E. Hairer, S. P. Nørsett, G. Wanner, Solving Ordinary Differential Equations. I. Nonstiff Problems, Springer Ser. Comput. Math., Springer-Verlag, Berlin, 1991.

[2] T. Blochwitz, et al., The functional mockup interface for tool independent exchange of simulation models, in: C. Clauss (Ed.), Proc. 8th Int. Modelica Conf, LiU E-Press, 2011, pp. 105–114. doi:10.3384/ecp11063105.

[3] F. E. Cellier, E. Kofman, Continuous System Simulation, Springer US, 2006. doi:10.1007/0-387-30260-3.

[4] A. Sjö, Analysis of computational algorithms for linear multistep methods, Ph.D. thesis, Lund University (1999).

[5] S. E. Mattsson, H. Elmqvist, J. F. Broenink, Modelica: An international effort to design the next generation modelling language, in: Special issue on Computer Aided Control System Design, CACSD, 1997, pp. 16–19.

[6] MODELISAR, Functional Mock-up Interface, Version 2.0, Interface specification, MODELISAR (July 2014).

[7] J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, H. Tummescheit, Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problem, Comput. Chem. Eng. 34 (11) (2010) 1737–1749.

[8] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, C. S. Woodward, Sundials: Suite of nonlinear and differential/algebraic equation solvers, ACM Trans. Math. Softw. 31 (3) (2005) 363–396. doi:10.1145/1089014.1089020.

[9] P. Pannu, C. Andersson, C. Führer, J. Åkesson, Coupling model exchange fmus for aggregated simulation by open source tools, in: Proc. 11th Int. Modelica Conf., LiU E-Press, 2015, pp. 903–909. doi:10.3384/ecp15118903.

[10] Dassault Systèmes, Dymola - multi-engineering modeling and simulation - version 2016, http://www.dymola.com/, [accessed: 2016-02-03] (2015).

[11] C. Andersson, C. Führer, J. Åkesson, Assimulo: A unified framework for ODE solvers, Math. Comput. Simul. 116 (2015) 26 – 43. doi:10.1016/j.matcom.2015.04.007.

[12] E. Hairer, G. Wanner, Solving Ordinary Differential Equations. II. Stiff and Differential-Algebraic Problems, 2nd Edition, Springer Ser. Comput. Math., Springer-Verlag, Berlin, 1996.

Preprints in Mathematical Sciences 2016:1
ISSN 1403-9338

LUTFNA-5007-2016

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/