Toward Verified Modelling and Simulation of Closed Loop Systems in SmartMOBILE

E. Auer

Faculty of Engineering, NKO, University of Duisburg-Essen, D-47048 Duisburg, Germany, auer@inf.uni-due.de

Summary. Software for modelling and simulation (MSS) of mechanical systems helps to reduce production costs for industry. Usually, such software relies on (possibly erroneous) finite precision arithmetic and does not take into account uncertainty in the input data. The program SmartMOBILE enhances the existing MSS MOBILE with verified techniques to provide a guarantee that the obtained results are correct and measure the influence of data uncertainty. In this paper, particular attention is paid to the current strivings toward verified modelling and simulation of closed loop systems.

1 Introduction

Three major phases during modelling and simulation process are analysis, implementation (verification) and simulation (validation) [5]. The first step is to *analyze* the real world problem and to design a formal model of the system under consideration. The second is to *implement* this model. Usual tasks at this stage include code verification (finding logical and programming errors in the code) and numerical verification (minimizing numerical errors in results). The final step is *validation* during which model fidelity is established.

Modern numerical modelling and simulation software (MSS) such as MOBILE [2] automatizes parts of this cycle and in this way accelerates the development process for a product reducing production costs. However, some of the tasks are not covered. For example, MSS usually relies on floating point arithmetic, which either shifts the task of result verification into the validation stage of the cycle or leaves that question unanswered.

Development of methods for numerical result verification is the research area of the whole branch of numerics also known as "interval", "validated" or "verified" arithmetic. Such methods not only provide a guarantee that the obtained results are correct but also propagate initial data uncertainty almost as their by-product. A recently developed tool SmartMOBILE [1] interfaces libraries for result verification with MOBILE to be able to cover more tasks

A.D. Fitt et al. (eds.), *Progress in Industrial Mathematics at ECMI 2008*,
Mathematics in Industry 15, DOI 10.1007/978-3-642-12110-4_91,

from the modelling and simulation cycle. At the stage of analysis, it offers techniques helping to take into account model errors and to perform uncertainty analysis in general. At the implementation stage, it provides result verification for kinematics and dynamics of various mechanical systems. Finally, the use of algorithmic differentiation and newly developed methods for sensitivity analysis identifies critical parameters and in this way makes the stage of validation easier.

In this paper, we focus on the uses of SmartMOBILE for accurate (and, in some cases, verified) simulation of kinematics and dynamics of closed loop systems. We begin by describing main features of SmartMOBILE which include free choice of underlying arithmetic. Further, we demonstrate options for result verification for models of closed loop systems. Such simulations are especially difficult since they have differential-algebraic equations (DAEs) as their basis. Finally, we recapitulate the main results.

2 Main Features of SmartMOBILE

SmartMOBILE is an object-oriented software for verification of mechanical systems based on MOBILE which employs floating point numerics. Models in both tools are executable C++ programs built of the supplied classes for transmission elements such as rigid links, for scalar or spatial objects such as reference frames and for solvers such as those for differential equations.

SmartMOBILE is one of the first integrated environments providing result verification for kinematical and dynamic simulations of mechanical systems. The advantage of this environment is its flexibility due to its template structure: the user can choose the kind of (non)verified arithmetics according to his task. Intervals and Taylor model arithmetics are currently available in SmartMOBILE. However, advanced users are not limited to them and are free to plug in their own implementations.

Although switching from MOBILE to SmartMOBILE is easy, users are assisted by converters in this process. However, automatically generated elements might require a heuristic improvement by the user, if they contain code fragments transformation of which cannot be algorithmized, for example, non-verified equation solvers.

For most kinematical problems, it is sufficient to use the supplied basic data types (e.g. intervals) as parameters to all the template classes for a particular model. The main idea for dynamic and special kinematical tasks such as finding of system equilibria is to use pairs basic data type/corresponding solver. Here, the basic data type should be constructed in such as way as to allow us to apply the given solver. That is, it should automatically deliver, for example, all the derivatives the solver requires.

3 Options for Simulation of Closed Loop Systems in SmartMOBILE

There are several options in MOBILE to simulate kinematics and dynamics of closed loop systems. We single out an aspect of this complicated process which is important for the following considerations.

Mathematical models behind this type of problems are systems of DAEs. Since the index of such systems is usually equal to three, common IVP solvers for DAEs such as Dassl cannot handle them as they are. This fact is one of the reasons why the original system of DAEs is automatically transformed into an equivalent system of ODEs in MOBILE. Two transmission elements are developed for this purpose. `MoExplicitConstraintSolver` handles systems with one or two constraints of a certain form where explicit solution of the corresponding algebraic system is possible. `MoImplicitConstraintSolver` uses Newton's method to obtain solutions to arbitrary (nonlinear) systems of algebraic equations.

As reported in [1], it is possible to verify the kinematics and dynamics of closed loop systems in SmartMOBILE by using `TMoExplicitConstraintSolver`. As for the second element, there exists a version of it called `MoIImplicitConstraintSolver` using Newton-Gauss-Seidel or Krawczyk methods to verify kinematics of systems modelled with it. The implementation of the latter element for dynamics seems impracticable since all iterations of a verified zero-finding method would have to be taken into the algorithmic differentiation graph for computing derivatives, which still cannot be handled satisfactorily by the software.

An alternative is to solve the DAE system directly. Unfortunately, verified solution of IVPs to DAEs is a very new research area. One tool available to us is an extension of ValEncIA-IVP which is still under development. However, the first results [4] are promising. Since this solver requires an approximation of the DAE solution in its first stage, an accurate solver for this purpose should be integrated into SmartMOBILE.

4 Equations of Motion for a Spatial Four Bar Mechanism with Result Verification

Four bar mechanisms are simplest closed loop systems relevant for real life applications. In this section, we consider the one shown in Fig. 1, left side. This closed system consists of two revolute joints `R1` and `R2`, a double-revolute joint modelled by two joints `R3` and `R4`, a spherical joint `S1`, and four rigid links `base`, `link_1`, `link_2`, and `coupler` between them. To model this task, the loop is dissected at the body `coupler` (cf. Fig. 1, right side). The closure condition `core` is the equality of the corresponding displacements and rotations for the reference frames `K7` and `K10`. Usually, `core` is an instance of the measurement object `MoChord3DPose`.

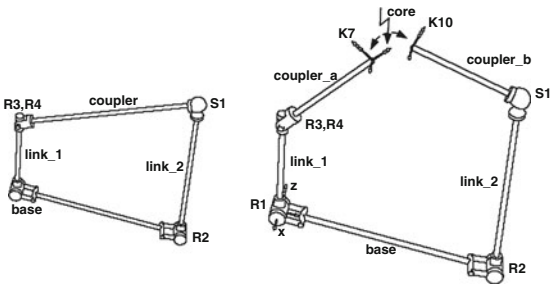

Fig. 1. The iconic model of a spatial four bar mechanism

For this type of closure conditions, we employ `TMoIImplicitConstraintSolver` in SmartMOBILE. The task is to find the mass matrix and the force for this system with result verification. However, this example shows more than just the possibility of verification. Using it, we can compare the method of obtaining derivatives of a function by algorithmic differentiation to the one based on physical considerations.

If we compute the Jacobian of the goal function in the interval version by using the force-based method supplied by MOBILE, the enclosure is equal to

$$\begin{pmatrix} [\pm 10^{-2}] & [-1.1;-0.9] & [\pm 10^{-4}] & [\pm 10^{-3}] & -[1.1] & [0;0] \\ [-0.8;1.8] & [\pm 10^{-3}] & [-0.7;-0.2] & [-1.3;-0.5] & [\pm 10^{-3}] & [0;0] \\ [0.5;3.1] & [0;0] & [0.9;1.1] & [0.2;1.0] & [\pm 10^{-3}] & [0;0] \\ [\pm 10^{-3}] & [0.2;0.6] & [\pm 10^{-3}] & [\pm 10^{-3}] & [0.0;1.1] & [0.3;1.4] \\ [\pm 10^{-3}] & [0.8;1.0] & [\pm 10^{-3}] & [\pm 10^{-3}] & [-1.4;-0.3] & [0.0;1.1] \\ [0.3;1.4] & [\pm 10^{-3}] & [-1.4;-0.3] & [0.3;1.4] & [\pm 10^{-3}] & [\pm 10^{-3}] \end{pmatrix}.$$

Here, the numbers are rounded up to the first digit after the decimal point. For the same parameter values, the enclosure of the Jacobian obtained with algorithmic differentiation is much tighter:

$$\begin{pmatrix} [0.0] & -[1.0] & [0.0] & [0.0] & -[1.1] & [0.0] \\ [1.5] & [0.0] & -[0.5] & -[0.5] & [0.0] & [0.0] \\ [1.0] & [0.0] & [1.0] & [1.0] & [0.0] & [0.0] \\ [0.0] & [0.5] & [0.0] & [0.0] & [0.0] & [1.0] \\ [0.0] & [0.9] & [0.0] & [0.0] & -[1.0] & [0.0] \\ [1.0] & [0.0] & -[1.0] & [1.0] & [0.0] & [0.0] \end{pmatrix}.$$

The notation [*number*] means that an enclosure of a *number* with a diameter of at most 10^{-12} is obtained. Since this Jacobian is important not only for the zero-finding method but also for correct computation of velocities and accelerations inside the implicit solver, it is crucial to obtain its tight enclosure.

It was possible to enclose the mass matrix and the force in the spatial four bar mechanism, used later to obtain equations of motion, in tight intervals

[1.03043; 1.03043] and [−0.05104; −0.05104], respectively (rounded, the diameter is at most 10^{-12}). However, these are results for the case in which all parameters are chosen to be point intervals. In general, this model is prone to overestimation. This is confirmed by the relatively narrow search intervals which both Krawczyk and Newton-Gauss-Seidel methods require to be able to compute zeros.

5 An Accurate Direct DAE Solving Method in SmartMOBILE

In this Section, we show how the DAE system underlying closed loop models can be solved directly in SmartMOBILE. For this purpose, the integrator `TMoDAETSIntegrator` has been implemented recently. It is based on the solver DAETS which computes accurate floating point solutions to IVPs for DAEs [3]. One advantage of `DAETS` is that it solves the problem as it is without the user having to transform it into an ODE problem or eliminating higher order derivatives, regardless of the problem's index.

A new element `TMoMechanicalSystemDAE` (Martin Tändl) was developed to provide equations of motion in the form $g(q, q', t) = 0$ required by DAETS. Note that the mass matrix does not have to be inverted for this representation.

The four bar mechanism we consider as an example is simpler than that from Sect. 4. Now the system consists of two simple pendulums modelled with two revolute joints and two rigid links with masses. They are connected by the third rigid link. The instance of the measurement class `TMoChordPointPointQuadratic` helps to formulate closure conditions for the loop.

Note that the DAE-based system cannot be solved in MOBILE because it uses Dassl for DAE solving. Besides, the solver DAETS can be employed only in SmartMOBILE because this MSS version, as opposed to the usual one, supplies the necessary derivatives.

We simulated the above system in SmartMOBILE with the help of the usual ODE-based floating point method using the explicit solver and the Adams integrator, the accurate DAE-based method with `TMoDAETSIntegrator`, and the verified ODE-based method using the explicit solver and `TMoValencia-Integrator`. The solutions for the consistent initial conditions supplied by `TMoDEATSIntegrator` are shown in Fig. 2. The trajectories coincide as expected. Both of the non-verified solutions lie inside the obtained verified bounds. For this simple example, it is not possible to decide if the DAE-based method is more accurate than the ODE-based one, although the solver `DAETS` is reported to be so in more complicated cases [3].

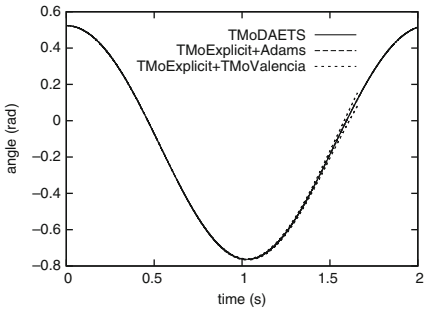

Fig. 2. The first angle of a four bar mechanism

6 Conclusions

In this paper, we presented the tool SmartMOBILE for guaranteed modelling and simulation of kinematics and dynamic of mechanical systems. With its help, the behavior of different classes of systems can be obtained with the guarantee of correctness, the option which is not given by tools based on floating point arithmetics. SmartMOBILE is flexible and allows the user to choose the kind of underlying arithmetics according to the task at hand.

A recent development concerned modelling and simulation of closed loop systems. New types of them were verified and a different kind of modelling was made possible.

Our short term task is the verification of the DAE-based approach to modelling and simulation of closed loop systems in SmartMOBILE using Valencia-IVP for DAEs as the basis. Almost all auxiliary components are already prepared for this purpose. First, `TMoMechanicalSystemDAE` generates the equations of motion in the required form. Next, a solver to compute the approximate solution is given by `TMoDAETSIntegrator`. Further, a verified zero-finding routine as required by ValEncIA-IVP for DAEs is already present in SmartMOBILE. Finally, a program for testing and computing consistent initial values is at the final stage of implementation. With these preparations, only the main DAE solving algorithm will have to be transferred.

References

1. Auer, E., Luther, W.: Proceedings of ICINCO **RA-1**, (2007)
2. Kecskeméthy, A., Hiller, M.: Comp. Meth. App. Mech. Eng. **115**, 287–314 (1994)
3. Nedialkov, N.S., Pryce, J.D.: J. Num. Anal. Ind. App. Math. **1**(1), 1–30 (2007)
4. Rauh, A., Auer, E., Minisini, J., Hofer, E.P.: Proc. App. Math. and Mech. **7**(1):1023001–1023002 (2007)
5. Schlesinger, S.: Simulation **32**:3, 103–104 (1979)