Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational and Applied Mathematics 199 (2007) 251–256

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

Interval modeling of dynamics for multibody systems

Ekaterina Auer

University of Duisburg-Essen, 47048 Duisburg, Germany

Received 15 December 2004

Abstract

Modeling of multibody systems is an important though demanding field of application for interval arithmetic. Interval modeling of dynamics is particularly challenging, not least because of the differential equations which have to be solved in the process. Most modeling tools transform these equations into a (non-autonomous) initial value problem, interval algorithms for solving of which are known. The challenge then consists in finding interfaces between these algorithms and the modeling tools. This includes choosing between "symbolic" and "numerical" modeling environments, transforming the usually non-autonomous resulting system into an autonomous one, ensuring conformity of the new interval version to the old numerical, etc. In this paper, we focus on modeling multibody systems' dynamics with the interval extension of the "numerical" environment MOBILE, discuss the techniques which make the uniform treatment of interval and non-interval modeling easier, comment on the wrapping effect, and give reasons for our choice of MOBILE by comparing the results achieved with its help with those obtained by analogous symbolic tools.

Keywords: Multibody modeling; Validated methods; Verified dynamics

1. Introduction

Multibody system modeling is an important part of applied physics, largely because this type of modeling is employed in many areas of our everyday life, from car driving to computer gaming. Before certain products can appear on the market, their characteristics have to be studied by means of their models, those being mostly systems of differential or algebraic equations (or, in complicated cases, both). The corresponding model is generated by modeling programs from the manually formalized description of a multibody system. This provides the characteristics of interest such as the system's kinematic behavior (*transmission of properties* from the source into the goal coordinate system is studied here) or its dynamic behavior (in this case the corresponding *equations of motion* have to be built and solved).

Although each of these characteristics can be obtained with the help of modern modeling tools, the results are often unreliable owing to either errors that are unavoidably generated by numerical solvers of differential (or algebraic) systems, or inaccuracy induced by the idealization of the model itself. This fact forces modeling software developers to look for more reliable options, one of which is interval arithmetic (IA). The main reasons for using IA are, first, the guaranteed correctness of results and, second, the ability to allow for uncertainty in parameters, which helps to provide more realistic models or study the system's behavior for different parameters.

E-mail address: auer@informatik.uni-duisburg.de.

0377-0427/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2005.08.045

Consequently, an interface between multibody modeling tools and IA is necessary. Although it is possible to develop an external IA interface for some modeling programs (SYMKIN [7]), there are cases (MOBILE [5]) for which the alteration of the multibody software itself is unavoidable. This presupposes a fusion of data types and algorithms from IA and mechanics. Such a fusion means, however, that IA developers need access to the code for the goal modeling software as well as some help from its programmers. A good environment for such cooperation formed itself at our university since a number of modeling tools (e.g. MOBILE or SYMKIN, as mentioned above) had been developed there.

This paper discusses our experience in interfacing multibody modeling software and IA. We will consider two major modeling software types and strategies for verifying their results (Section 2), then focus on one possible design of a validating tool based on MOBILE (Section 3), and, finally, give reasons for choosing this strategy by comparing it to the approach based on the tools of the SYMKIN type (Section 4). A summary is given in Section 5.

2. Types of modeling software and strategies for their verification

In most cases simulating dynamics means solving a certain initial value problem. Methods based on floating point arithmetic, which are usually derivative-free, can be employed at this stage. However, if guaranteed results are required, validated (for example, IA) methods are necessary. They need derivatives [8,9], which can be obtained by considering some mechanical characteristics of a multibody system. However, this option is not supplied by modeling software developers. Another method of finding derivatives is to use automatic or algorithmic differentiation.

The difference between these two techniques can be illustrated by the difference in the modeling software to which they can be applied. From the point of view of validation, all modeling tools can be nominally divided into "*symbolic*" and "*numerical*" according to the representation of models they produce [2]. A "symbolic" program (like SYMKIN or the MOMAPLE package for MOBILE) provides closed expressions for the equations of motion for the goal system, whereas a "numerical" one (like MOBILE) generates the values of the right side of the equations' state-space form at each requested point. In the latter case, only the corresponding program (or the *algorithmic representation* of the expression on the right side) is available for acquiring the necessary derivatives. With algorithmic differentiation it is possible to "record" this program as a directed acyclic graph (DAG) and then apply the usual computer algebra differentiation methods to it. Therefore, the "numerical" type is bound to use algorithmic differentiation, whereas it is possible to utilize both techniques while validating "symbolic" modeling tools, because given expressions can always be represented as pieces of code.

Accordingly, there is a difference between the strategies employed to verify results obtained with "symbolic" or "numerical" tools, although both approaches need several common components. These are a library for IA (in our case PROFIL/BIAS [6]), a package for algorithmic (or automatic) differentiation (FADBAD and TADIFF [3,4]), and an interval initial value problem solver, abbreviated as IIVPS below (VNODE [9]). In the "numerical" case, the source modeling software is enhanced with IA and algorithmic differentiation. The chosen IIVPS is then adjusted and integrated into it. This leads to a new product (the verifying extension of the original software), which provides the guaranteed results.

In case of the "symbolic" approach, the original software produces—often with the intermediate help of some known computer algebra tools such as MAPLE or MATHEMATICA—the exact expressions for the mathematical model of the system. The equations are then solved by the IIVPS, which provides the verified data. The function of the new product to be developed here is to automatize the information transfer from the source tool to the IIVPS.

3. Design of a validated modeling tool of the "numerical" type

We have already reported on the implementation of an enhanced version of MOBILE capable of producing guaranteed results [2]. Only significant or new aspects of such an implementation will be mentioned here.

To be able to model dynamics with interval methods, we need to know the Taylor coefficients of the solution of a certain (as a rule non-autonomous) initial value problem along with their Jacobians. To obtain them we used the libraries FADBAD and TADIFF, which implement algorithmic differentiation through overloading. In this case three data types are required: `INTERVAL` from PROFIL for ordinary interval values, `TINTERVAL` from TADIFF for Taylor coefficients, and `TFINTERVAL` from FADBAD/TADIFF for Jacobians of these Taylor coefficients.

This technique leads to a *design problem*: Given a function to compute the right side of an initial value problem, we need two further functions that differ from the first in data types only to acquire the necessary DAGs.

```
class TMoInterval{
    INTERVAL   Enclosure;
    TINTERVAL  TEnclosure;
    TFINTERVAL TFEnclosure;
    ...
}
```

Fig. 1. The basic class of the extended MOBILE version.

Fig. 2. `MoFrame` vs. `TMoFrame` $\langle\cdots\rangle$ usage.

There are many ways to solve the task, our preference being for a class that unites these three data types (see Fig. 1). The advantage lies in obtaining all necessary DAGs through a single function call; the drawback is the memory overuse. The class in Fig. 1 forms the basis of all other extended MOBILE classes up to the class `TMoAWAIntegrator`, which incorporates the adjusted VNODE and solves the actual equations of motion (cf. [2]).

The next aim is the *universality* of the extended tool. After having decided on `TMoInterval` as the basis, we could reimplement the whole MOBILE using this class instead of `MoReal` (MOBILE's variant of `double`). That would produce a MOBILE version operating only with intervals. But there are situations in which the usage of the old floating point version is preferable. Generally, the option of using MOBILE with different basic data types depending on the modeling task should be ensured so that there is no need to reimplement MOBILE each time it has to be used with data types other than `TMoInterval` or `MoReal`. Therefore, a *universal* version of MOBILE, which would unite all the above-mentioned variants or at least facilitate the process of integration of new basic types, is required.

Our solution in this case is to use templates. Each of the MOBILE classes is replaced with its template equivalent. Type-specific classes such as integrators have to be implemented separately for each data type. The example in Fig. 2 shows how to obtain the floating point and interval versions of the original MOBILE class `MoFrame`. The template-based usage is similar to the original one.

This template-based version of MOBILE can model kinematics and dynamics as well as find equilibrium for various classes of systems including those with explicitly solvable closed loops and non-autonomy. Compared to the results presented in [2], the computing time is reduced by approximately 80% (includes a more powerful test PC), but the wrapping effect is still substantial (Fig. 3). In defense of the extended MOBILE, it should be said that VNODE's solution to $\ddot{y} + 9.81 \sin y = 0$ (the mathematical model for the simple pendulum) is also unsatisfactory although VNODE has to deal with a much "shorter" right-hand side.

4. A comparison between "symbolic" and "numerical" validation strategies for dynamics

As shown in the previous section, modeling with the "numerical" tool was verified although there was room for improvements in computing time and reducing the wrapping effect. In this section we will compare both strategies by means of some characteristic examples to try to answer the question whether the "symbolic" approach is more effective.

It was decided to use MoMaple as the original "symbolic" software. This is a package for MOBILE, which alters some of MOBILE's transmission elements in such a way as to produce the *expressions* for the equations of motion. The input is a C++ program, which differs only slightly from MOBILE's: it uses the identifier `MoMaple` instead of `Mo` at the appropriate places. The program produces a MAPLE worksheet, which helps to obtain the necessary symbolic description. At present, it is not possible to model mechanisms with springs with MoMaple, because the corresponding

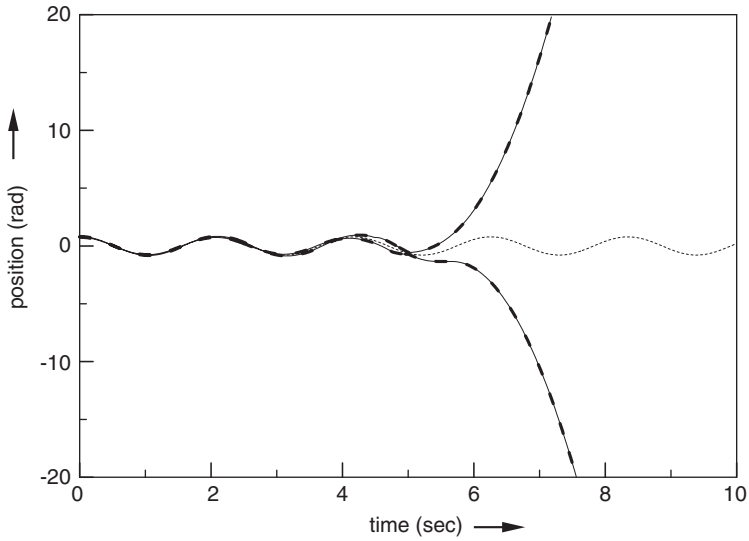

Fig. 3. Position of the simple pendulum for the initial condition [0.76969, 0.801106]: extended MOBILE and VNODE (QR factorization), MOBILE (Adams-Moulton).

Table 1
Comparison data for the "numerical" (N) and "symbolic" (S) approaches

Example	Time (s)		Global error		Break-down	
	N	S	N	S	N	S
Conical pendulum	235	21.7	2.28e − 09	1.67e − 09	105.06	110.78
Triple pendulum	646	928	1.61e − 07	1.96e − 07	33.6	35.6
One-axis manipulator	1758	117	1.1e − 12	1.35e − 08	2.139	0.582

transmission elements are still under development. Furthermore, closed-loop mechanisms cannot be modeled there. In these cases it is necessary to refer to SYMKIN.

To find the overall computing time for validating with MOMAPLE, it is necessary to sum up the following: the time needed by MOMAPLE to generate a MAPLE worksheet, the time needed by MAPLE to generate the expressions for equations of motion, VNODE's time for solving them, and a certain time for the process of starting MAPLE, translating the expressions from MAPLE's syntax into VNODE's, etc. This process has not yet been automatized, which is why it was decided to use only VNODE's time as a reference here.

We integrated the models for the following multibody systems over the time interval [0; 10] with the QR factorization method (Taylor expansion order 15, constant stepsize 0.01, where possible). The tests were performed on a PC (Pentium 4, 2.80 GHz) under CYGWIN. The goal was to compare both strategies by measuring their respective computing times (only a reference is available here), global errors, and break-down times.

The following systems, ordered by their increasing complexity, were considered: a conical pendulum (four significant modeling elements, the resulting initial value problem has four variables), a triple pendulum [2] (12 significant elements, six variables) and a one-axis manipulator [1] (a closed-loop system, 18 elements, two variables). The comparison data is summarized in Table 1.

We have chosen the first example because it is analytically solvable [10] and therefore provides a good starting point for our comparison. A conical pendulum is a simple pendulum the tip of which is forced to move along a circumference. This can be achieved through the special initial condition during simulation. The corresponding initial value problem actually has two variables, but to make the simulation more complex we chose a different parametrization of the rotational movement of the pendulum's tip (along the x and y axes instead of z axis), which led to a system of the

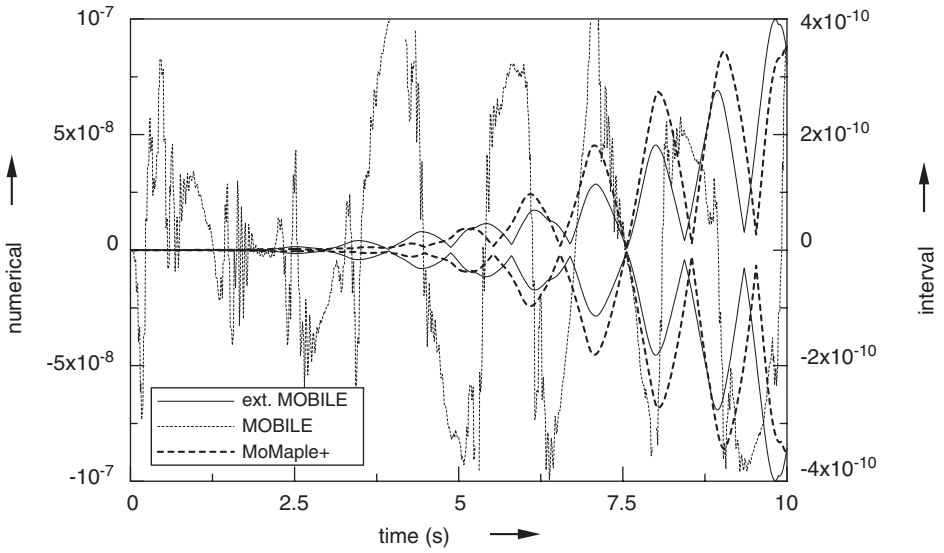

Fig. 4. The relative position of the conical pendulum.

dimension four. Fig. 4 shows the solutions obtained with MOBILE, its extended version, and MoMAPLE plus VNODE in relation to the exact solution. In the scale of 10^{-10} it is possible to discern that both validated solutions contain the exact one. The standard MOBILE solution, however, differs considerably (in the scale of 10^{-7}) due to inaccuracies in initial approximations. Overall, the computing time needed by the "symbolic" approach is much less than that of the "numerical" one, the global error is somewhat smaller, and the algorithm breaks down later (see Table 1).

The data for both the latter, more complex examples shows, however, that the "numerical" approach either needs less time and has almost the same global error (the triple pendulum) or proves to be more robust (the one-axis manipulator). Note that both approaches behave unsatisfactorily because they break down very quickly. Nonetheless, although the standard MOBILE integrator (backward differentiation formulae) provides a solution over a longer period of time, it lies considerably (10^{-6}) outside the validated boundaries right from the start and is therefore unreliable.

5. Summary

Our goal was to verify dynamics in MOBILE. We solved this task through enhancing MOBILE with algorithmic differentiation and the IIVPS VNODE, allowing for the universality of application through templates. The extended version of MOBILE verifies the dynamics for various classes of multibody systems including non-autonomous and explicitly solvable closed-loop ones.

However, the computing time and wrapping effect in our version are considerable. Therefore, a comparison of our approach to the "symbolic" was necessary. This comparison shows that if the "symbolic" strategy is better for small multibody systems, the "numerical" one proves to be more effective for (at least several) complicated systems in regard to either computing time or global error (unfortunately, it could not be shown that this approach was both faster and more robust). The major drawback of the "symbolic" way is its lack of automatization, which appears to be complicated and might either claim a lot of computing time or not be possible at all (only manual adjustment of "multibody" syntax to IA).

Therefore, rather than the choice of a different approach, a further reduction of computing time and wrapping effect in the extended version of MOBILE seems to be the more promising option.

References

[1] E. Auer, E. Dyllong, W. Luther, D. Stankovic, H. Traczinski, Integration of accurate distance algorithms into a modeling tool for multibody systems, in: 17th IMACS World Congress, Paris, 2005(Paper T2-I-73-0397).

[2] E. Auer, A. Kecskeméthy, M. Tändl, H. Traczinski, Interval algorithms in modeling of multibody systems, in: R. Alt, A. Frommer, R. Kearfott, W. Luther (Eds.), Numerical Software with Result Verification, Lecture Notes in Computer Science, vol. 2991, Springer, Berlin, Heidelberg,

New York, 2004, pp. 132–159.
[3] C. Bendsten, O. Stauning, FADBAD, a flexible C++ package for automatic differentiation using the forward and backward methods, Technical Report 1996-x5-94, Technical University of Denmark, Lyngby, 1996.
[4] C. Bendsten, O. Stauning, TADIFF, a flexible C++ package for automatic differentiation using Taylor series, Technical Report 1997-x5-94, Technical University of Denmark, Lyngby, 1997.
[5] A. Kecskeméthy, MOBILE Version 1.3. User's guide, 1999.
[6] O. Knüppel, PROFIL/BIAS—a fast interval library, Computing 53 (1994) 277–287.
[7] T. Krupp, Symbolische Gleichungen für Mehrkörpersysteme mit kinematischen Schleifen, Berichte aus der Softwaretechnik, Shaker Verlag, Aachen, 1999.
[8] R. Lohner, Einschließung der Lösung gewönlicher Anfangs- und Randwert-aufgaben und Anwendungen, Ph.D. Thesis, University of Karlsruhe, 1988.
[9] N. Nedialkov, Computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation, Ph.D. Thesis, University of Toronto, 1999.
[10] K. Wohlhart, Dynamics, Vieweg, Braunschweig, 1998, pp. 66–67.