Int. J. Appl. Math. Comput. Sci., 2009, Vol. 19, No. 3, 455–467
DOI: 10.2478/v10006-009-0037-z

USES OF NEW SENSITIVITY AND DAE SOLVING METHODS IN SMARTMOBILE FOR VERIFIED ANALYSIS OF MECHANICAL SYSTEMS

EKATERINA AUER, WOLFRAM LUTHER

Faculty of Engineering, INKO
University of Duisburg-Essen, D-47048 Duisburg, Germany
e-mail: {auer,luther}@inf.uni-due.de

Software for modeling 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, we outline the main features and functionalities of SMARTMOBILE. In particular, we focus on its use of newly developed methods for sensitivity analysis and DAE solving for several practically relevant mechanical systems.

Keywords: multibody systems, result verification, sensitivity, DAE, uncertainty.

1. Introduction

Three major phases during modeling and simulation process according to (Schlesinger, 1979) are analysis (qualification), implementation (verification) and simulation (validation), shown in Fig. 1. The first step is to *analyze* the real world problem and to design a formal model of the system under consideration. The application domain is described, relevant parameters and their ranges defined and types of uncertainty in the model and its parameters identified. The second step is to *implement* this model, which includes code verification—that is, finding logical and programming errors in the code—followed by numerical verification of results. The third and final step is *validation*, during which model fidelity is established, for example, through comparing the outcomes of simulations and real world experiments.

Modern numerical modeling and simulation software (MSS), for example, MOBILE (Kecskeméthy and Hiller, 1994), automatizes parts of this cycle and in this way accelerates the development process for a product reducing production costs. However, some of the goals presented in the cycle remain uncovered. First, the stage of analysis cannot be fully automatized. Second, 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. Next, developers or users are themselves responsible for

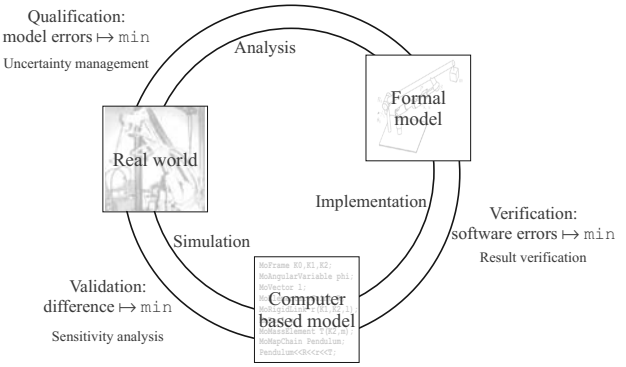

Fig. 1. Modeling and simulation cycle.

the task of code verification, although the existing technique of literate programming (Knuth and Levy, 1993) would allow them to generate theory, code, and documentation from the same source files minimizing implementation and maintenance errors. Finally, the validation stage seems to be least automated and needs some external software to cope, for example, with sensitivity analysis of a system model to its parameters.

The 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 (Auer, 2007) interfaces libraries for result verification with MOBILE to be able to cover more tasks from the modeling and simulation cycle. At the stage of analysis, it offers techniques helping to take into account model errors (Auer and Luther, 2007) and to perform uncertainty analysis in general. At the implementation stage, it provides result verification for the kinematics and dynamics of various mechanical systems. Finally, the use of algorithmic differentiation and newly developed methods for sensitivity analysis (Rauh *et al.*, 2009) identifies critical parameters and in this way makes the validation stage easier.

In this paper, we focus on the use of SMARTMOBILE for sensitivity analysis and reliable (or verified) simulation of the kinematics and dynamics of closed loop systems. We begin by describing the main features of SMARTMOBILE, which include a free choice of the underlying arithmetic. The example of a double pendulum demonstrates the usage and comparative advantages of integrated initial value problem (IVP) solvers and arithmetic in SMARTMOBILE. Next, we show in Section 3 how to compute sensitivities of models to their parameters. Two models of practically relevant subsystems of a human body are considered here. In Section 4, we demonstrate the options for result verification of kinematic or dynamic simulations for models of closed loop systems. Such simulations are especially difficult since they have differential-algebraic equations (DAEs) as their basis. Finally, we overview our current work toward the integration of a verified DAE solver (Rauh *et al.*, 2007b) and summarize the verifying and validating options SMARTMOBILE currently offers.

2. Main features of SMARTMOBILE

The main topic of this section is a verified modeling and simulation tool SMARTMOBILE based on floating point MSS MOBILE. SMARTMOBILE is one of the first integrated environments providing result verification for kinematic and dynamic simulations of mechanical systems. Besides, it offers model validation options described in detail in Section 3. At the stage of analysis, it helps to minimize model errors by providing sloppy joints (Auer and Luther, 2007), a direct method of DAE solving or formalized computation of Jacobians (cf. Section 4). An advantage of this environment is its flexibility owing to the template structure: users can choose the kind of (non-)verified arithmetic according to their task.

In this section, we explain how to verify an already existing modeling and simulation tool first. Then, we overview the main functionalities of SMARTMOBILE and compare several model design options. Finally, a double pendulum with an uncertain initial angle exemplifies the usage of SMARTMOBILE for dynamic simulations and compares the available IVP solvers.

2.1. Result verification of existing MSS.

Initially, the main goal of SMARTMOBILE was to simulate dynamics of mechanical systems in a guaranteed way based on a supplied formal model from MOBILE. The concept behind this latter tool, however, presupposes that kinematics is also modeled and so it is easy to simulate it afterwards. MOBILE belongs to the numerical type of modeling software, that is, it does not produce symbolic expressions for the resulting mathematical model. Only the values of output parameters for the user-defined values of input parameters and the source code of the program itself are available. In this case, it is necessary to integrate verified techniques into the core of the software itself, as opposed to the tools of the symbolic type, where the task is basically reduced to the application of the verified methods to the obtained system of equations.

To simulate dynamics, we have to solve an IVP for the differential equations of motion for the system model (usually, in the state space form):

$$\dot{x}(t) = f(x(t), t), \quad x(t_0) \in [x_0], \tag{1}$$

where $t \in [t_0, t_m] \subset \mathbb{R}$ for some $t_m > t_0$, $f \in C^{\alpha-1}(\mathcal{D})$ for some $\alpha > 1$, $\mathcal{D} \subseteq \mathbb{R}^n$ is open, $f : \mathcal{D} \times \mathbb{R}^1 \mapsto \mathbb{R}^n$, and $[x_0] \subset \mathcal{D}$. Usually, the problem is discretized on a grid $t_0 < t_1 < \cdots < t_m$. Denote the solution with the initial condition $x(t_{k-1}) = x_{k-1}$ by $x(t; t_{k-1}, x_{k-1})$ and the set of solutions $\{x(t; t_{k-1}, x_{k-1}) \mid x_{k-1} \in [x_{k-1}]\}$ by $x(t; t_{k-1}, [x_{k-1}])$. Then the goal is to find interval vectors $[x_k]$ for which the relation $x(t_k; t_0, [x_0]) \subseteq [x_k]$, $k = 1, \ldots, m$ holds.

We consider non-autonomous systems because, in our context, it is more advantageous to work with them directly rather then transform them into an autonomous system. For IVP solvers that do not support direct solving of non-autonomous systems, suitable equations can be generated automatically inside SMARTMOBILE. In some cases, f can also depend on uncertain parameters p, for which the upper and lower bounds on uncertainty are known. Moreover, we consider IVPs for differential-algebraic systems of equations of the form

$$g(x(t), \dot{x}(t), t) = 0 \tag{2}$$

if the mechanical system under examination has loops.

As opposed to usual numerical IVP solvers, programs with result verification need derivatives of the right side of these equations. Divided differences cannot be of use here since they only provide approximations to the true derivatives. A better option is to employ algorithmic differentiation (Griewank, 2000), the method that is practicable but might consume a lot of CPU time in the case of such a large program as MOBILE. An alternative is to exploit the system's mechanics for this purpose. This option is not provided by MOBILE developers yet and seems to be rather difficult to algorithmize for (arbitrary) higher

orders of derivatives. Besides, as shown in Section 4.1, the overestimation for intervals might be too large. That is why it was decided to employ the first possibility in SMARTMOBILE.

Basically, there are two methods to implement algorithmic differentiation: code transformation and overloading. The latter was chosen to obtain code derivatives from MOBILE. The overloading technique in this case means that a new data type is developed that is capable of computing the derivative along with the function value. This new data type is used instead of the simple one in the code piece. The drawback of this method is the lack of automatic optimization during derivative computation. The technique of code transformation might evaluate derivatives more efficiently. However, this approach seems to be difficult to implement for large code pieces which are self-contained programs themselves.

Consequently, the following components are necessary to verify results of an existing MSS: a basic verified arithmetic, a tool to compute derivatives (if not contained in the basic arithmetic), and an IVP solver. During the last decades, a variety of libraries have appeared in each area. We use the library PROFIL/BIAS (Knüppel, 1994) for intervals, COSY INFINITY (Berz and Makino, 2006) and RiOT (Eble, 2007) for Taylor models, FADBAD++ (Bendsten and Stauning, 1996) as well as CppAD (Bell, 2006) for algorithmic differentiation, and, finally, VNODE (Nedialkov, 2002), ValEncIA-IVP (Rauh *et al.*, 2007a), RiOT again, and VSPODE (Lin and Stadtherr, 2006) for solving initial value problems.

2.2. Working with SMARTMOBILE. Since we chose overloading as a technique to obtain verified results, it is now clear why a floating point based code itself has to be changed in the case of numerical MSS. We have to substitute a new enhanced data type for the `double` data type of the original version, which results in a new program. In SMARTMOBILE, all relevant occurrences of `MoReal` (an alias of `double` in MOBILE) are replaced with an appropriate new data type. Almost each verified solver needs a different basic data type (cf. Table 1). Therefore, the whole structure of MOBILE was turned into templates in SMARTMOBILE for the user to be able to chose a solver according to the task at hand. Basically, all transmission elements in SMARTMOBILE contain placeholders instead of occurrences of `MoReal`, for which the actual data type is substituted at the end of the modeling stage.

The strategy in SMARTMOBILE is to use pairs type/solver. For example, to provide interval verification with the help of VNODE-based solver `TMoAWAIntegrator`, the basic data type `TMoInterval` including data types necessary for algorithmic differentiation should be used. The data type `TMoFInterval` enables the use of `TMoValenciaIntegrator`, an adjustment of the basic version of VALENCIA-IVP. `TMoRiotIntegrator` is based on the IVP solver from the library RIOT, an independent C++ version of COSY and COSY VI, and requires the class `TMoTaylorModel`, a SMARTMOBILE-compatible wrapper of the library's own data type `TaylorModel`. Analogously, to be able to use COSY, the wrapper `RDAInterval` is necessary. A current addition to the available solvers in SMARTMOBILE is the one based on VSPODE. More advanced users can incorporate their own solvers into SMARTMOBILE following the guidelines from (Auer, 2007).

In general, kinematics can be simulated with the help of all of the above mentioned basic data types. However, other basic data types might become necessary for more specific tasks such as finding equilibrium states of a system since they require specific solvers. SMARTMOBILE provides an option of modeling equilibrium states in a verified way with the help of the interval-based data type `MoFInterval` and the class `MoIGradientStaticEquilibriumFinder` using either a version of the Newton-Gauss-Seidel iteration from the C-XSC TOOLBOX (Hammer *et al.*, 1995) or the Krawczyk algorithm (Krawczyk, 1969).

A model in MOBILE and, consequently, SMARTMOBILE, is an executable program in C++ which uses predefined classes as transmission elements. Owing to the structure of SMARTMOBILE, it is easy for a MOBILE user to switch to the verified version. As shown in Fig. 3, the differences are, first, in the names of transmission elements (they have a preceding letter `T` and are templates so that the actual data type has to be specified for them) and, second, in the solvers that are used (e.g., `TMoAWAIntegrator` instead of `TMoAdamsIntegrator`). Besides, two converters were developed to reduce the amount of manual work for the user. The first one transforms MOBILE models into the syntax prescribed by SMARTMOBILE for the two major design options mentioned in Section 2.3. The second one helps to convert newly developed MOBILE elements into SMARTMOBILE templates via a series of automatically generated LINUX scripts. Elements generated by both converter types might require a heuristic improvement by the user, if they contain code fragments the transformation of which cannot be automated, for example, non-verified equation solvers.

2.3. Design vs. efficiency. Having decided to use overloading as a strategy to verify the results of kinematic or dynamic simulations as well as to obtain necessary derivatives, we are confronted with the following situation. For one and the same goal function f, we need to compute at least two sets of values: the function values themselves and the values of their derivatives. This means that, for example, in the case of the

Table 1. Available data types and solvers in SMARTMOBILE.

Type	Integrator	Purpose
MoReal	MoAdamsIntegrator,...	nonverified dynamics
TMoInterval TMoFInterval TMoTaylorModel --	TMoAWAIntegrator TMoValenciaIntegrator TMoRiOTIntegrator TMoVSPODEIntegrator	verified dynamics of ODE based systems
RDAInterval	—	Taylor model based kinematics
MoFInterval	MoIGradientEquilibriumFinder TMoImplicitConstraintSolver	verified equilibria kinematics with constraints
MoSInterval	TMoValenciaSIntegrator	verified sensitivity

VNODE-based solver TMoAWAIntegrator, the right hand side of motion equations has to be computed with three data types: INTERVAL for the simple function values, T<INTERVAL> for the Taylor coefficients, and T<F<INTERVAL>> for the Jacobians of the Taylor coefficients. There are several ways to handle this situation.

The first one is to make a template out of the function f. This is a common solution in all verified IVP solvers but it cannot be implemented in SMARTMOBILE as is. The reason is the optimization of the basic MOBILE models with respect to their runtime: all transmission elements there do not work with copy constructors but use references to the actual instances of objects in the main() function.

Therefore, a possible option is to generate a "collective" data type such as the already mentioned TMoInterval containing all three necessary data types:

```
class TMoInterval{
    INTERVAL         enc;
    T<INTERVAL>      tenc;
    T<F<INTERVAL>>   tfenc;
}
```

The SMARTMOBILE model for this case is shown in Fig. 3. The advantage of this approach is that the function f has to be called only once to obtain all necessary values. The disadvantage is the additional amount of work which is performed if only the function values without derivatives are of interest at a given algorithm stage.

This disadvantage can be overcome by several strategies. The simplest one is to implement f multiple times: in our example, this means instantiating all the templates in the model from Fig. 3 with INTERVAL, T<INTERVAL>, and T<F<INTERVAL>> data types successively. The references to instances of each of MoMechanicalSystem objects containing the equations of motion are then passed to the corresponding IVP solver. In this way, the existing IVP solvers can be integrated into SMARTMOBILE more easily. On the other hand, the computing time is also improved.

The disadvantage of the strategy above is code inflation. Two more approaches can be used to overcome it. (Note that no considerable improvement in computing time is to be expected here.) The first one is to encapsulate the whole model in a separate template object, for example, MoModel. Then the SMARTMOBILE model from our example will consist of three template instantiations and the call to the IVP solver:

```
int main(){
    MoModel<INTERVAL>        iModel;
    MoModel<T<INTERVAL>>     tiModel;
    MoModel<T<F<INTERVAL>>>  tfiModel;

    TMoAWAIntegrator integrator(
        iModel.system,timodel.system,
        tfiModel.system,... );
    integrator.doMotion();
}
```

Here, we do not have to write the same model twice. However, this representation seems to be less intuitive. A converter for this case is under development at the moment.

The last of the options we can think of is the one proposed by John D. Pryce. For each transmission element in SMARTMOBILE, a new one with the same name should be automatically generated to contain template instantiations of this element for each necessary data type. In the same example, it would mean generating for each template element from TMoFrame to TMoMechanicalSystem a non-template container of the following type:

```
class TMoFrame{
    TMoFrame<INTERVAL>        iFrame;
    TMoFrame<T<INTERVAL>>     tiFrame;
    TMoFrame<T<F<INTERVAL>>>  tfiFrame;
}
```

The model itself would not differ from the one in MOBILE but for the employed verified solver. However, regenerating the whole of the SMARTMOBILE structure to be able to integrate every new solver seems to be too much of an effort without a gain in computing time.

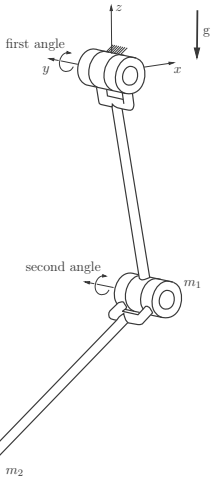

Fig. 2. Iconic model of the double pendulum.

2.4. Verified simulation of dynamics of a double pendulum with an uncertain initial angle. Let us consider the example of a double pendulum shown in Fig. 2 described in more detail in (Rauh, Auer and Hofer, 2007a). The initial conditions for the angles (in rad) and angular velocities (in rad/s) are given below:

$$\left[\frac{3\pi}{4} \pm 0.01 \cdot \frac{3\pi}{4} \quad -\frac{11\pi}{20} \quad 0.43 \quad 0.67\right]^{\mathrm{T}}.$$

The first value has an uncertainty of $\pm 1\%$ of its nominal value.

The corresponding SMARTMOBILE model is shown in Fig. 3 and consists of five frames, two revolute joints, two rigid links, and two mass elements, all connected into one chain by the instance `Pend` of the class `TMoMapChain`. The angles and angular velocities associated with the joints are the independent variables of the equations of motion obtained in their state space form by an instance of `TMoMechanicalSystem`. We use the basic data type `TMoInterval` and the IVP solver `TMoAWAIntegrator` to solve the equations and simulate dynamics in this example.

To give an overview of the comparative advantages of the currently available IVP solvers in SMARTMOBILE, we simulate the dynamics of this example using each of them with its optimal settings from out point of view. All simulations were performed on a four processor dual core computer (Intel Xeon CPU 2.00GHz) under Linux 2.6.25.14-69.fc8. The results are summarized in Table 2. The first row defines the verified strategy used for simulation. `TMoAWAIntegrator` is used with a variable step size h and QR-factorization method with a Taylor series of order 20. The order of Taylor models in the case of `TMoRiOTIntegrator` is equal to 5. This solver is used with a variable step size ranging from 0.0002 to 0.2. `TMoValenciaIntegrator`, which does not implement a step size control strategy yet, is employed with

```
# define TMoInterval t;
TMoFrame<t> K0, K1, K2, K3, K4;
TMoAngularVariable<t> psi1, psi2;
// transmission elements
TMoVector<t> l1(0,0,-1), l2(0,0,-1);
TMoElementaryJoint<t> R1(K0,K1,psi1,xAxis);
TMoElementaryJoint<t> R2(K2,K3,psi2,xAxis);
TMoRigidLink<t> rod1(K1,K2,l1),rod2(K3,K4,l2);
t m1(1),m2(1);
TMoMassElement<t> Tip1(K2,m1),Tip2(K4,m2);
// the complete system
TMoMapChain<t> Pend;
Pend << R1<<rod1<<Tip1<<R2<<rod2<<Tip2;
// dynamics
TMoVariableList<t> q; q << psi1<<psi2;
TMoMechanicalSystem<t> S(q,Pend,K0,zAxis);
TMoAWAIntegrator I(S,0.0001,ITS_QR,15);
I.doMotion();
```

Fig. 3. Model of the double pendulum in SMARTMOBILE.

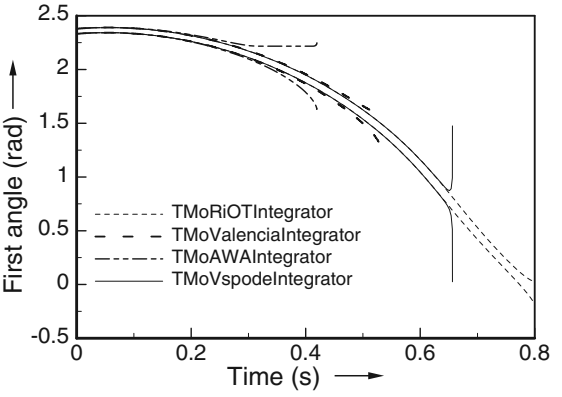

Fig. 4. Simulation results for the double pendulum in SMARTMOBILE.

the constant step size 0.0001. Finally, the recently developed `TMoVspodeIntegrator` is set to use a variable step size, a Taylor series of order 20 for expansion in time, and Taylor models of order 17 for expansion in initial conditions.

The second row of Table 2 contains the times in seconds after which the verification of results is no longer possible. This happens here due to the chaotic character of the system considered and the resulting overestimation.

The CPU times (in seconds) each program needs to solve the system over the time interval $[0; 0.4]$ are recorded in the third row. Note that these times are given only as a reference. They cannot be taken absolutely for various reasons, one of which is the lack of guidelines to the optimal settings for each solver. We recorded times provided by the programs themselves. The values of break-down times in the table are rounded up to the third digit after the decimal point. In Fig. 4, trajectories for the first angle are shown. The curves of the same pattern represent the upper and lower bound of the solution obtained with the corresponding solver under the given uncertainty in the first initial angle.

Table 2. Performance of verified IVP solvers for the double pendulum.

Strategy	TMoAWA (variable h)	TMoRiOT ($0.0002 \leq h \leq 0.2$)	TMoValencia ($h = 10^{-4}$)	TMoVSPODE (variable h)
Break–down	0.420	0.801	0.531	0.656
CPU Time	5	285	22	10

The use of `TMoValenciaIntegrator` improves the tightness of the resulting enclosures in comparison to `TMoAWAIntegrator` for this example. Although `TMoRiOTIntegrator` breaks down much later than both former solvers, it needs more CPU time. The use of new `TMoVspodeIntegrator` seems promising because it provides tighter enclosures than `TMoValenciaIntegrator` or `TMoAWAIntegrator` in a comparable CPU time even though some of its important characteristics have not been integrated into SMARTMOBILE yet.

3. Sensitivity analysis in SMARTMOBILE

In this section, we consider options for obtaining the sensitivity of system models in SMARTMOBILE with respect to their parameters. In the dynamic case, we define sensitivity as

$$[s] = \frac{\partial [x]}{\partial [p]} \ ,$$

where p is a(n) (uncertain) parameter, the bounds for which are given by the interval $[p]$, and $[x]$ is the verified solution of the corresponding IVP. To find $[s]$, SMARTMOBILE provides the class `TMoValenciaSIntegrator` based on the corresponding algorithm from VALENCIA-IVP (Rauh *et al.*, 2009). On the other hand, the same definition can be used for kinematics, where the solution $[x]$ is replaced by the quantity of interest as an interval evaluation of a function $f([p])$ to obtain it. In SMARTMOBILE, the user can employ algorithmic differentiation data types to compute this sensitivity. Defined this way, $[s]$ provides a linear measure of uncertainty influence on the system model. Unfortunately, verified sensitivity cannot be always computed due to possible overestimation in interval values. In this case, an approximation to the influence of uncertainty on the system can be obtained as

$$[u] = \sum_{i=1}^{n} \frac{\partial f(p_1, \ldots, p_n)}{\partial p_i} \times [p_i]$$

with intervals, where the sensitivity $s = [s_1, \ldots, s_n]$ is a `double`-based value of the partial derivative of $f(p)$, $p = [p_1, \ldots, p_n]$. The computation of floating point sensitivities for dynamics, although possible in principle, is not currently implemented in SMARTMOBILE.

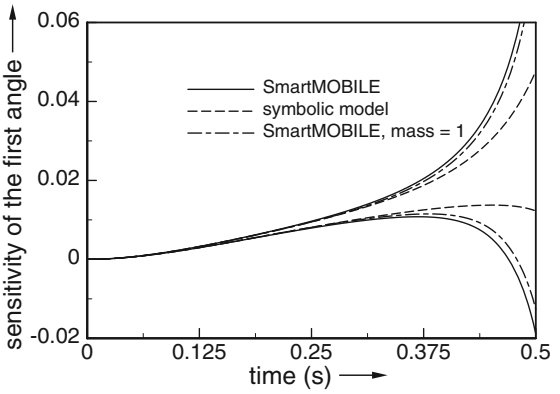

Fig. 5. Sensitivity of the double pendulum with respect to the first mass.

This section is structured as follows: First, we show how `TMoValenciaSIntegrator` works using the example of the double pendulum from Section 2.4. Then sensitivities are computed in kinematic and dynamic cases for two practically relevant subsystems of the human body.

3.1. Sensitivity of the double pendulum to mass. Under the same conditions as in Section 2.4, we compute the sensitivity of the first angle of the double pendulum to the first mass m_1 with the help of the class `TMoValenciaSIntegrator`. In Fig. 5, the dashed-dotted curves show results obtained with SMARTMOBILE for $m_1 = 1$ kg, the solid ones for $m_1 \in [0.99; 1.01]$ kg. As a comparison, results for the symbolic equations of the double pendulum with $m_1 \in [0.99; 1.01]$ kg from VALENCIA-IVP are represented by the dashed curves.

The enclosures obtained in VALENCIA-IVP are tighter, which is not very remarkable since the symbolic model contains less numerical operations and is therefore less prone to overestimation. The enclosures for certain and uncertain masses do not differ much over the time interval considered. This fact leads to the conclusion that mass uncertainty does not contribute as much as uncertainty in the initial conditions to the overall overestimation that shows itself in the continuous widening of enclosure widths over time.

This conclusion is also confirmed by Figs. 6 and 7. The first one shows sensitivities of the double pendulum with respect to the first uncertain initial condition. Again, the dashed-dotted curves represent results obtained

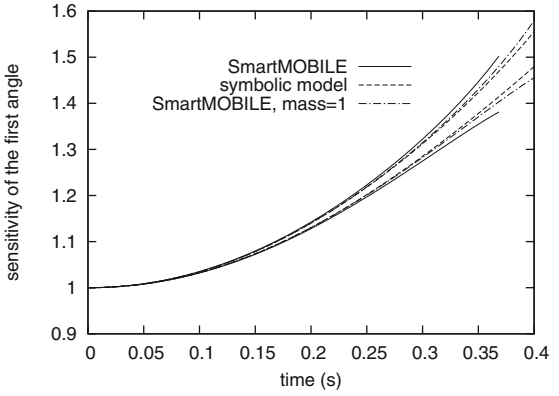

Fig. 6. Sensitivity of the double pendulum with respect to the first initial condition.

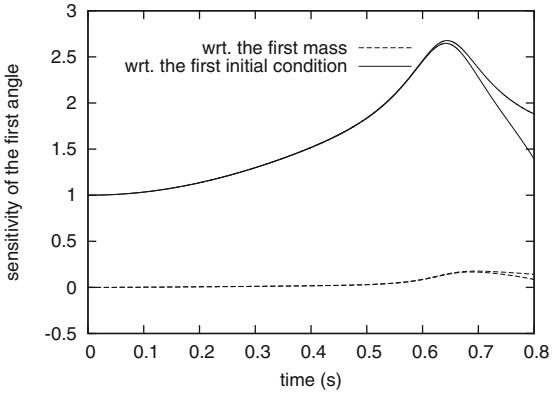

Fig. 7. Sensitivity of the double pendulum with respect to the first mass and the first initial condition without uncertainty in parameters.

with SMARTMOBILE for $m_1 = 1$ kg, the solid ones for $m_1 \in [0.99; 1.01]$ kg, and the dashed curves for the symbolic equations of the double pendulum with $m_1 \in [0.99; 1.01]$ kg from VALENCIA-IVP. The curves from SMARTMOBILE do differ slightly more from each other than the corresponding curves in Fig. 5. However, this difference can be ascribed largely to overestimation since the problem at hand is chaotic.

In Fig. 7, the sensitivity enclosures of the first angle of the double pendulum to the first mass and the first initial condition are juxtaposed. For better representation, we considered $m_1 = 1$ and the first initial condition without uncertainty (i.e., $3\pi/4$) in the symbolic model. The system is much more sensitive to changes in the first initial condition because the absolute value of the respective sensitivity in each point is considerably larger than the one with respect to the mass. The sensitivity to the mass is increasing in the examined interval in Fig. 5. However, Fig. 7 gives an indication of how this value would behave over a larger time interval if we had no overestimation. It stays small and even decreases against the end of the simulation. This holds for all angles and their velocities while the difference for the first angle is the most prominent since we chose the first initial value as the parameter. Therefore, the first mass does not influence the system model as much as the initial conditions.

3.2. Identification of body segment motion using marker trajectories. Now we consider the problem of the reconstruction of the hip joint position from positions of markers fastened to specified places on a patient's leg, a task described in greater detail in (Auer and Luther, 2009). The corresponding model is purely kinematic. At first, the segment frame motion is obtained by orthogonalizing the bone and joint axes sequentially. In the second step, the model parameters and the motion of the model segments are adjusted to the marker trajectories using nonlinear optimization. For the purposes of verification, the fitting task of this second stage was simplified in such a way as to be explicitly solvable. For a detailed description of the non-verified method, see (Tändl *et al*., 2009).

The data on marker positions contains measurement errors which appear, for example, due to skin movement during motion. These uncertainties are empirically set to ± 10 mm for each marker displacement tangential to skin and the one due to soft tissue movement. The marker displacement normal to skin can be up to ± 5 mm. Besides, both knee and ankle widths with nominal values of 120 mm and 80 mm, respectively, are also measured with an error of ± 10 mm.

In (Auer and Luther, 2009), it was shown that the length of the femur bone under these uncertainties was as stated in Table 3 (rounded up to the first digit after the decimal point). Here, the second row identifies enclosures resulting from taking into consideration uncertainties in knee and ankle widths and the third row for all three above mentioned uncertainties due to marker displacements. The simulations with intervals did not produce a meaningful result because of the large overestimation (cf. the third column of the table).

By using Taylor models, we reduced overestimation and obtained acceptable enclosures shown in the second column. Here, Taylor models were bounded by the LDB algorithm from COSY (the linear dominated bounder) to obtain their interval-like enclosures. For uncertainties in knee and ankle widths, the overall uncertainty in the length of the femur bone amounted to 20 mm. Marker displacements caused an enclosure of almost 622 mm in diameter, which is less meaningful in real life cases. This result indicates the need to perform all corresponding measurements with great care if the proposed algorithm is to be used. On the other hand, it might be worth while to devise an algorithm that would be less sensitive to marker displacements.

An interesting measure on overestimation in this case can be supplied by the reference uncertainty $[u]$ defined at the beginning of this section. In Table 4, we show the sensitivity s of the length of the femur bone with respect

Table 3. Verified length of femur bone (in mm).

	RDAInterval	INTERVAL
Knee, ankle	[377.6; 396.7]	$[0; \infty]$
Marker displacements	[0.000; 621.4]	no answer

Table 4. Sensitivity of the femur length.

Knee	Ankle	Tangential	Normal	Soft
0.4	-0.3	-2	0.7	1.4

to the knee width, ankle width and marker displacements tangential to skin, normal to skin, and those due to soft tissue movement. Here, s is computed in SMARTMOBILE using algorithmic differentiation and the floating point data type `F<double>`. The numbers in the table are rounded up to the first digit after the decimal point. The approximation $[u]$ is equal to ± 7 mm if uncertainties in knee and ankle widths are considered and ± 37.5 mm if the influence of marker displacements is of interest. This indicates that the large diameter of the enclosure shown in the last line in the second column of Table 3 is not entirely due to overestimation but results from the high sensitivity of the model to this kind of parameters.

One more interesting conclusion which the sensitivities in Table 4 allow originates from the comparison with a result from (Auer and Luther, 2009), shown again in Table 5. The femur length was measured under ± 5, ± 10 and ± 20 mm uncertainty individually for each kind of displacement. According to the enclosures obtained there, the displacement due to soft tissue movement has the biggest influence on the resulting uncertainty of the model. However, Table 4 shows that the model is most sensitive to the marker displacement tangential to skin because this partial derivative has the largest absolute value. That means that the interval implementation of the model might have room for improvements where displacements due to soft tissue movements are concerned so that overestimation larger than for other parameters is not generated.

3.3. Simplified muscle activation model. The model under investigation (Fig. 8) represents a simplified subsystem of the human leg described in greater detail in (Strobach *et al.*, 2005; Auer *et al.*, 2007). It consists of pelvis, thigh and shank. To drive the model in forward dynamics simulations, the muscle *biceps femoris short head* is included, which is responsible for knee flexion. For the purposes of the first verified study, the overall model is simplified so that it is everywhere continuously differentiable. For example, the force law of the involved muscle model is not HILL-type anymore but corresponds to the simple rule

$$F(q) = \frac{P}{1 + Tq},$$

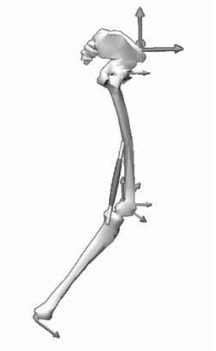

Fig. 8. Examined subsystem of the human leg.

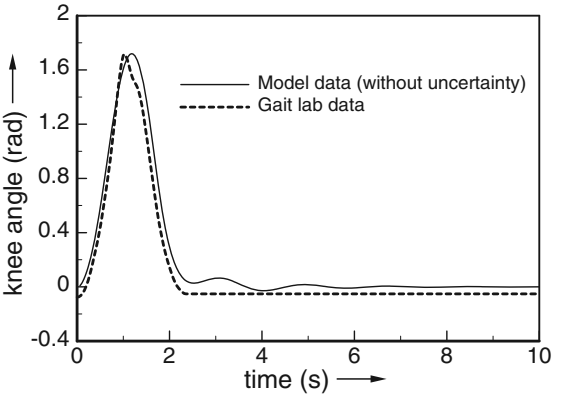

Fig. 9. Comparison of simulations without uncertainties with the data from the gait lab.

where P and T are constants. Besides, the activation function is allowed to be negative, which does not take into account mechanical constraints. Under these restrictions, the simulated results do not quite fit the results obtained in the gait lab (cf. Fig. 9 for the knee angle).

Since most of the model parameters cannot be measured exactly, the task consists in investigating how uncertainty in parameters influences the outcome of simulations. The parameters of interest are the thigh length, the shank length, and the point of the muscle insertion at the hip (its z and y coordinates).

In (Auer *et al.*, 2007), thigh and shank lengths were identified empirically as the most influential parameters considered. The obtained enclosures from Table 6 show that the model reacts most sensitively to the changes in these parameters. Here, an uncertainty of $\pm 0.1\%$ in the nominal value of each parameter is considered. The dynamic simulations are then performed to obtain the point after which verification is not possible.

To prove this, we compute the verified sensitivities of the solution with respect to all parameters of interest using the class `TMoValenciaSIntegrator` in SMARTMOBILE. The results for the sensitivity of the knee angle are shown in Fig. 10. We notice again that the curves for the sensitivity with respect to thigh and shank lengths

Table 5. Sensitivity of the model with respect to marker displacements (m) due to skin movements: femur length (m).

Marker displacement	$[-0.005; 0.005]$	$[-0.010; 0.010]$	$[-0.020; 0.020]$
tangential to skin	$[0.3492; 0.4203]$	$[0.3008; 0.4576]$	$[0.1146; 0.5468]$
normal to skin	$[0.3742; 0.3985]$	$[0.3279; 0.4413]$	$[0.3335; 0.4427]$
soft tissue	$[0.3593; 0.4125]$	$[0.3279; 0.4413]$	$[0.0000; 0.6330]$

Table 6. Influence of uncertainties on the knee angle.

Uncertainty ($\pm 0.1\%$)	Break down
Thigh length $= 0.45$ m	0.118 s
Shank length $= 0.49$ m	0.125 s
$z = -0.2281$ m	0.135 s
$y = -0.0253$ m	0.389 s

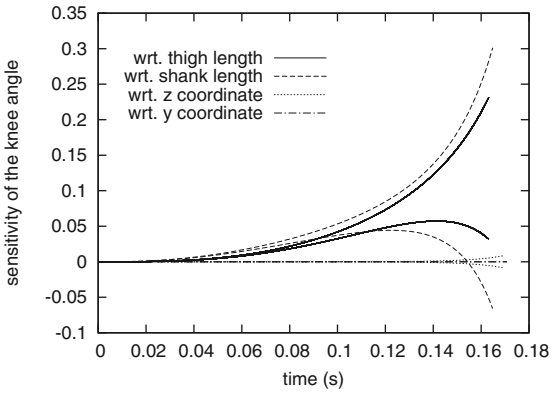

Fig. 10. Verified sensitivity of the knee angle to four uncertain parameters.

have a significantly bigger absolute value in each point than those for muscle insertion and are also more prone to overestimation. This confirms the empirical conclusion made earlier.

4. Options for the simulation of closed loop systems in SMARTMOBILE

There are several options in MOBILE to simulate the kinematics and dynamics of closed loop systems. Without going into much detail, to be found elsewhere (Kecskeméthy and Hiller, 1994), we single out an aspect of this complicated process which is important from our point of view. Mathematical models behind this type of problems are systems of differential-algebraic equations. 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 (non-linear) systems of algebraic equations.

As already reported (Auer, 2007), it is possible to verify the kinematics and dynamics of closed loop systems in SMARTMOBILE by using a verifying version of the first element called `TMoExplicitConstraintSolver`. As for the second element, there exists a version of it called `MoIImplicitConstraintSolver` using the Newton-Gauss-Seidel or Krawczyk methods to verify the kinematics of systems modeled 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 as reported in (Rauh *et al.*, 2007b) are promising. Since this solver requires an approximation of the DAE solution in its first stage, a reliable solver for this purpose should be integrated into SMARTMOBILE.

Therefore, we structure this section as follows: First, an example showing how `MoIImplicitConstraintSolver` can be employed in SMARTMOBILE is described. Note that SMARTMOBILE is used here not only for simulation but also for enhancing the modeling of the example. In the next subsection, an accurate floating point based solver `TMoDEATSIntegrator` is presented. We conclude with a short description of our current work toward the integration of a verified DAE solver based on the corresponding version of VALENCIA-IVP into SMARTMOBILE.

4.1. Equations of motion for a spatial four bar mechanism with result verification.

Four bar mechanisms are the simplest closed loop systems relevant for real life applications. In this subsection, we consider the one shown in Fig. 11, left side. This closed system consists of two revolute joints `R1` and `R2`, a double-revolute joint modeled by two joints `R3` and `R4`, a spherical joint `S1`, and four massless rigid links `base`, `link_1`, `link_2`, and

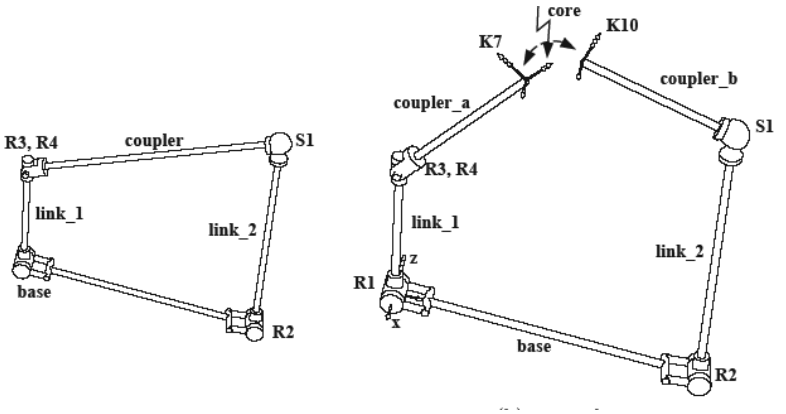

Fig. 11. Iconic model of a spatial four bar mechanism.

coupler between them. To model this task, the loop is dissected at the body coupler (cf. Fig. 11, 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.

For this type of closure conditions, the element MoImplicitConstraintSolver should be used. In SMARTMOBILE, we employ TMoIIImplicit-ConstraintSolver based on the Newton-Gauss-Seidel or Krawczyk zero-finding algorithms. 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

$$\begin{pmatrix} [-10^{-2};10^{-2}] & [-1.1;-0.9] & [-10^{-4};10^{-4}] \\ [-0.8;1.8] & [-10^{-3};10^{-3}] & [-0.7;-0.2] \\ [0.5;3.1] & [0;0] & [0.9;1.1] \end{pmatrix}.$$

Here, the numbers are rounded up to the first digit after the decimal point. We show only 3×3 left upper selection out of this 6×6 matrix for space reasons. For the same parameter values, the enclosure of the Jacobian obtained with algorithmic differentiation is much tighter:

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

Here, 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.

However, automatic computation of the Jacobian calls for different constraint modeling. Instead of the measurement transmission element MoChord3DPose between the frames K7 and K10, the following group of measurements should be used:

```
TMoChord3DPosition<type> p(K7,K10);
TMoChordPlanePlane<type> xy(K10, K7);
TMoChordPlanePlane<type> xz(K10, K7);
TMoChordPlanePlane<type> yz(K10, K7);
TMoChordList<type> core;
core<<p<<xy<<xz<<yz.
```

They are later used by MoIImplicit-ConstraintSolver in the same way as the old core element

```
MoIImplicitConstraintSolver
    (core, dependents, dChain);
```

where dependents are the dependent variables theta2 to theta7 (angles associated with S1 and R2 to R4). The element dChain is the dependent chain consisting of R2, link_2, the double-revolute joint R3-R4, the dissected body (coupler_a, coupler_b), and the spherical joint S1.

The measurement element MoChord3DPose should not be used in this case because it is implemented in such a way that the formal derivation of its code does not produce the same Jacobian as provided by the element itself by the function doJacobian() owing to numerical stability reasons.

In Table 7, the enclosures for the mass matrix and the force in the spatial four bar mechanism, used later to obtain equations of motion, are shown. The results are rounded up to the fifth digit after the decimal point. The second column shows results for the case in which all parameters are chosen to be point intervals. Here, we also get point intervals as the answer.

The third column records enclosures obtained for $\pm 1\%$ uncertainty in the nominal value of the angular ve-

Table 7. Enclosures for the mass matrix and force of the spatial four bar mechanism.

	points	$\beta' \pm 1\%$
$M(q)$	[1.03043]	[1.03043]
$F(q, q')$	$[-0.05104]$	$[-0.26551, 0.16208]$

locity associated with the revolute joint `R1`. We know theoretically that it does not have any influence on the mass matrix so that the result in this case is the same. But the enclosure of force has changed, and although its diameter is not very large, we cannot deduce the sign of the force from it anymore. This is an indication that the interval based model contains a lot of overestimation and special strategies to overcome it or another basic data type should be used in this case. This conclusion is confirmed by the relatively narrow search intervals which both Krawczyk and Newton-Gauss-Seidel methods require to be able to compute zeros.

The positive side is that this problem can be verified and transmission elements themselves can be enhanced by using algorithmic differentiation in SMARTMOBILE.

4.2. Non-verified accurate direct DAE solving method in SMARTMOBILE. In this subsection, we consider the example of a simple four bar mechanism and show how the underlying DAE system can be solved directly in SMARTMOBILE. Presently, this can only be done in floating point arithmetic there. 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 (Nedialkov and Pryce, 2007). 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 has to be developed to provide equations of motion in the form required by DAETS. This was supplied by Martin Tändl and is called `TMoMechanicalSystemDAE`. It constructs equations of motion in the form $g(x, \dot{x}, t) = 0$. One of the advantages of this representation is that the mass matrix does not have to be inverted.

The four bar mechanism we consider in this example is somewhat simpler than that from Subsection 4.1. Now the system consists of two simple pendulums modeled with two revolute joints `R1` and `R2` and two rigid links `rod1` and `rod2` (cf. Fig. 12). They are connected by the rigid link `offsetToSecondPendulum`. The instance `chord` of the measurement class `MoChordPointPointQuadratic` helps to formulate closure conditions for the loop.

The difference in the usage of `TMoMechanicalSystem` and `TMoMechanicalSystemDAE` is

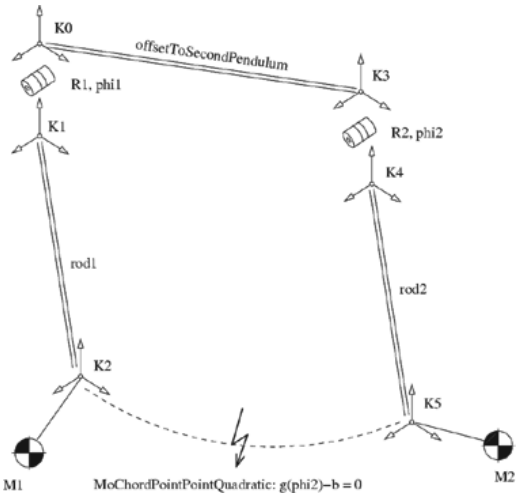

Fig. 12. Iconic model of a four bar mechanism.

```
MoMapChain dependentChain;
dependentChain << R2 << rod2;

MoLinearVariable chordOffset;
MoChordPointPointQuadratic chord(K2,K5,chordOffset,
                                        dependentChain) ;
MoExplicitConstraintSolver solver(chord,phi2,"solver") ;
MoMassElement Tip1 ( K2, 1 ) ;
MoMassElement Tip2 ( K5, 1 ) ;
MoMapChain PendulumSolved ; MoMapChain Pendulum ;
PendulumSolved << R1 << rod1 << Tip1
          << offsetToSecondPendulum << solver << Tip2 ;
Pendulum << R1 << rod1 << Tip1
       << offsetToSecondPendulum << R2 << rod2 << Tip2 ;
MoVariableList  varsODE, varsDAE ;
varsODE << phi1 ;   varsDAE << phi1  << phi2 ;
MoChordList constraintEquations ;
constraintEquations << chord ;
MoMechanicalSystemDAE  mechSysDAE( varsDAE , Pendulum,
                  constraintEquations  , K0 , zAxis ) ;
MoMechanicalSystem mechSysODE(varsODE, PendulumSolved,
                                          K0, zAxis) ;
```

Fig. 13. MOBILE model of a simple four bar mechanism (abridged).

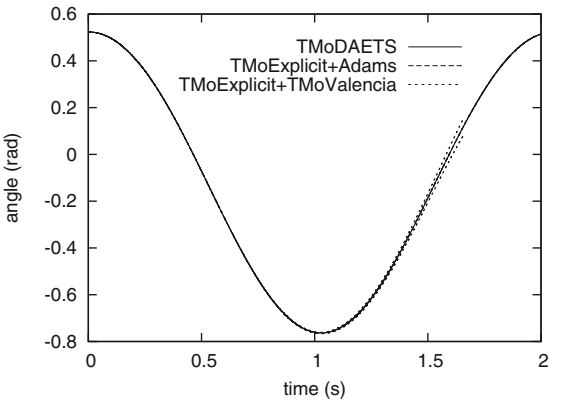

Fig. 14. First angle of a four bar mechanism.

demonstrated in Fig. 13 (for space reasons, only a part of original model from MOBILE is shown there). To obtain the solution with the help of the usual ODE based method, an instance solver of MoExplicitConstraintSolver has to be initialized with constraint equations chord. Then, solver is added to the overall chain called PendulumSolved, which is in turn passed to the MoMechanicalSystem object to produce equations of motion. In the case of the DAE based method, chord and the *original* system Pendulum are passed directly to the instance of MoMechanicalSystemDAE to obtain the equations.

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 supplies the necessary derivatives.

We simulated the above system in SMARTMOBILE with the help of the following strategies:

- the usual ODE based floating point method using the explicit solver and the Adams integrator;
- the accurate DAE based method with TMoDAETSIntegrator;
- the verified ODE based method using the explicit solver and TMoValenciaIntegrator.

The solutions for the initial conditions $\varphi_1(0) = 0.523686971267079$, $\varphi_2(0) = 0.763504826177631$ are shown in Fig. 14. (The numbers are supplied by TMoDEATSIntegrator as the consistent initial conditions.) The trajectories coincide, which is not surprising in this case since we simulate the same system which is modeled differently (verified, ODE and DAE based in floating point arithmetic). 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 (Nedialkov and Pryce, 2007). If we subtract the midpoints of the verified solution from each of the other obtained solutions at each point of time, the deviation is less than 10^{-13}.

4.3. Current work toward a verified IVP solver for DAEs in SMARTMOBILE. The new extension of VALENCIA-IVP for DAEs works with (roughly) the following formula:

$$0 = g(x_{\mathrm{app}}([t_0; t_m]) + [R(t_0)] + [t_0; t_m][\dot{R}([t_0; t_m])]) \ ,$$

where g is the goal function depending on the solution x, its first derivative $\dot{x}$, and time t. In the formula above, the solution x is replaced with the directive for its computation, where x_{app} is an approximate solution and R is the enclosure of its error. This points out the main components we need in SMARTMOBILE for incorporating this solver into its core so that the DAE based method for simulation of closed loop systems can be verified.

At first, an element returning the function g is required. Such an element, called TMoMechanicalSystemDAE, is already present in SMARTMOBILE and was described in the previous subsection. Next, a solver to compute the approximate solution is necessary. The solver TMoDAETSIntegrator provides such a possibility. Further, a verified zero-finding routine is required by VALENCIA-IVP for DAEs, which already exists in SMARTMOBILE and was tested for various kinematic problems. Finally, a program for testing and computing consistent initial values required by any DAE solver is in the final stage of implementation in SMARTMOBILE. Therefore, almost all auxiliary routines are prepared and so only the algorithm itself has to be transferred. This is our short-term task.

5. Conclusions

In this paper, we presented the tool SMARTMOBILE for guaranteed modeling and simulation of the 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 arithmetic according to the task at hand.

Special attention was paid to recently developed means of computing sensitivities in SMARTMOBILE. The new methods help the developer to identify critical parameters of the model and thus make the process of validation easier.

Another recent development concerned the modeling and simulation of closed loop systems. The kinematics of systems with arbitrary constraints (defined by MoIImplicitConstraintSolver) could be verified and the direct accurate DAE based kind of modeling was made possible.

Our future work will consist in verifying the DAE based approach to the modeling and simulation of closed loop systems as well as devising further methods for the reduction of overestimation in SMARTMOBILE.

References

Auer, E. (2007). *SmartMOBILE: A framework for reliable modeling and simulation of kinematics and dynamcis of mechanical systems*, Ph.D. thesis, Universität Duisburg-Essen, Duisburg.

Auer, E. and Luther, W. (2007). SMARTMOBILE—An environment for guaranteed multibody modeling and simulation, *Proceedings of the 4th International Conference on*

Informatics in Control, Automation and Robotics ICINCO, *Angers, France*, pp. 109–116.

Auer, E. and Luther, W. (2009). Numerical verification assessment in computational biomechanics, *Proceedings of the Dagstuhl Seminar 08021: Numerical Validation in Current Hardware Architectures, Dagstuhl, Germany,* Lecture Notes in Computer Science, Vol. 5492, Springer-Verlag, Berlin/Heidelberg, pp. 145–160.

Auer, E., Tändl, M., Strobach, D. and Kecskeméthy, A. (2007). Toward validating a simplified muscle activation model in SMARTMOBILE, *Proceedings of 12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006)*, *Duisburg, Germany*, p. 7.

Bell, B. M. (2006). Automatic differentiation software CppAD. `http://www.coin-or.org/CppAD/`.

Bendsten, C. and Stauning, O. (1996). FADBAD, a flexible C++ package for automatic differentiation using the forward and backward methods, *Technical Report 1996-x5-94*, Technical University of Denmark, Lyngby.

Berz, M. and Makino, K. (2006). COSY INFINITY 9.0. Programmer's manual, *Technical Report MSUHEP 060803*, Michigan State University, East Lansing, MI.

Eble, I. (2007). *Über Taylor-Modelle*, Ph.D. thesis, Universität Karlsruhe, Karlsruhe.

Griewank, A. (2000). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, PA.

Hammer, R., Hocks, M., Kulisch, U. and Ratz, D. (1995). *C++ Toolbox for Verified Computing I—Basic Numerical Problems*, Springer-Verlag, Heidelberg/New York, NY.

Kecskeméthy, A. and Hiller, M. (1994). An object-oriented approach for an effective formulation of multibody dynamics, *Computer Methods in Applied Mechanics and Engineering* **115**(3–4): 287–314.

Knüppel, O. (1994). PROFIL/BIAS—A fast interval library, *Computing* **53**(3–4): 277–287.

Knuth, D. E. and Levy, S. (1993). *The CWEB System of Structured Documentation*, Addison-Wesley, Reading, MA.

Krawczyk, R. (1969). Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken, *Computing* **4**(3): 187–201.

Lin, Y. and Stadtherr, M. A. (2006). Validated solution of initial value problems for ODEs with interval parameters, *Proceeding of the NSF Workshop on Reliable Engineering Computing, Savannah, GA, USA*.

Nedialkov, N. and Pryce, J. (2007). Solving differential-agebraic equations by Taylor series (III): The DAETS code, *Journal of Numerical Analysis, Industrial and Applied Mathematics* **1**(1): 1–30.

Nedialkov, N. S. (2002). The design and implementation of an object-oriented validated ODE solver, *Technical report*, University of Toronto, Toronto.

Rauh, A., Auer, E. and Hofer, E. P. (2007a). VALENCIA-IVP: A comparison with other initial value problem solvers, *Proceedings of the 12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006), Duisburg, Germany,* p. 36.

Rauh, A., Auer, E., Minisini, J. and Hofer, E. P. (2007b). Extensions of VALENCIA-IVP for reduction of overestimation, for simulation of differential algebraic systems, and for dynamical optimization, *PAMM* **7**(1): 1023001–1023002.

Rauh, A., Minisini, J. and Hofer, E. P. (2009). Towards the development of an interval arithmetic environment for validated computer-aided design and verification of systems in control engineering, *Proceedings of the Dagstuhl Seminar 08021: Numerical Validation in Current Hardware Architectures, Dagstuhl, Germany,* Lecture Notes in Computer Science, Vol. 5492, Springer–Verlag, Berlin/Heidelberg, pp. 175–188.

Schlesinger, S. (1979). Terminology for model credibility, *Simulation* **32**(3): 103–104.

Strobach, D., Kecskeméthy, A., Steinwender, G. and Zwick, B. (2005). A Simplified Approach for Rough Identification of Muscle Activation Profiles via Optimization and Smooth Profile Patches, *CD Proceedings of the International ECCOMAS Thematic Conference on Advances in Computational Multibody Dynamics, ECCOMAS, Madrid, Spain.*

Tändl, M., Stark, T., Erol, N.E., Löer, F. and Kecskeméthy, A. (2009). An object-oriented approach to simulating human gait motion based on motion tracking, *International Journal of Applied Mathematics and Computer Science* **19**(3): 469–483.

Ekaterina Auer received her diplomas in mathematics and computer science from Ulyanovsk State University, Russia, in 2001 and from the University of Duisburg-Essen, Germany, in 2002. Since 2002, she has been working at the Chair for Computer Graphics and Scientific Computing at the University of Duisburg-Essen as a research assistant, receiving her Ph.D. in 2006. Her main interests are scientific computing and the development of software for applications to problems in mechanics and engineering.

Wolfram Luther leads a research group of a dozen persons in scientific computing, computer graphics, image and text processing. The team is specialized in the development of software and algorithms with result verification and of interactive teaching and learning systems in several contexts.

Received: 22 September 2008
Revised: 9 February 2009