Internal Report 2005-21

Impulse-Based Dynamic Simulation of Higher Order and Numerical Results[1,2]

Alfred A. Schmitt, Jan S. Bender and Hartmut Prautzsch
Institut für Betriebs- und Dialogsysteme
Fakultät für Informatik
Universität Karlsruhe
aschmitt@ira.uka.de, jbender@ira.uka.de, prautzsch@ira.uka.de

Abstract

First, we will provide a short introduction to the impulse-based method for dynamic simulation. Till now, impulses were frequently used to resolve collisions between rigid bodies. In the last years, we have extended these techniques to simulate constraint forces. Important properties of the new impulse method are: (1) Simulation in Cartesian coordinates, (2) complete elimination of the constraint drift known from Lagrange multiplier methods, (3) simple integration of collision and friction and (4) real-time performance even for complex multibody systems like six-legged walking machines. In order to demonstrate the potential of the impulse-based method, we report on numerical experiments. We compare the following dynamic simulation methods: (1) Generalized (or reduced) coordinates, (2) the Lagrange multiplier method with and without several stabilization methods like Baumgarte, the velocity correction and a projection method, (3) impulse-based methods of integration order 2, 4, 6, 8, and 10. We have simulated the mathematical pendulum, the double and the triple pendulum with all of these dynamic simulation methods and report on the attainable accuracy. It turned out that the impulse methods of higher integration order are all of $O(h^3)$ *but have very small factors and are therefore relatively accurate. A Lagrange multiplier method fully stabilized by impulse-based techniques turned out to be the best of the Lagrange multiplier methods tested.*

Keywords: *dynamic simulation, multibody systems, impulse-based dynamic simulation, higher order methods, numerical experiments, Lagrange multiplier methods, generalized coordinates, accuracy*

---

[1] Supported by the Deutsche Forschungsgemeinschaft (German Research Foundation)
[2] This report is an extended version of [Schmitt and Bender 2005b]

1. Introduction and overview

Impulse-based dynamic simulation of multibody systems was introduced in [Schmitt 2003] and [Bender et al. 2003], an English language introduction was given in [Bender et al. 2005]. From the results achieved experimentally it could be demonstrated that the impulse-based dynamic simulation method is competitive with other methods known from literature. The most important advantages are the comparatively simple program structure, the real-time capability even for the simulation of complex models (e.g., six-legged walking machines), and the input specification and internal simulation in Cartesian coordinates as preferably used in computer graphics and also in almost all engineering applications. A further advantage of the impulse-based method is the simple handling of collision and Coulomb friction. Impulse-based dynamic simulation of collision events involving two or more non-linked rigid bodies is nowadays well understood due to the frequently cited work of [Mirtich and Canny 1995] whereas our extended method which also covers linked rigid body systems is more or less unknown to the community of dynamics researchers till now.

In [Schmitt et al. 2005], some open theoretical questions were answered. Now we know that the iterative procedure for velocity correction as well as for the computation of the impulses for the joint correction can generally be replaced by the solution of a linear systems of equations. These sets of equations are always solvable if the respective equations for the Lagrange multiplier method have solutions. In critical situations, the time step size *h* must be chosen sufficiently small. Thus it is unnecessary to set up exact conditions for convergence and the speed of convergence of the iterative procedure because it can be substituted by solutions of linear systems. However, since the iterative method is very attractive because of the extremely simple and easy to implement algorithm, it is advisable to reduce the time step size *h* if convergence problems are encountered. One recognizes convergence problems very easily by the fact that, in the course of iterations, joint distance errors do not decrease monotonously as is usually the case.

Furthermore, it was proven that the impulse-based simulation method converges to the mathematically correct solution of the dynamics problem if $h \to 0$. The error is of order $O(h^3)$ which is sufficiently small for most applications in computer graphics, virtual reality scenarios and computer animation.

In [Schmitt 2003], it was already shown that one can define impulse-based procedures of higher integration order. So far it has been completely unclear whether a gain in accuracy can be expected from these procedures. To answer this open question we will provide a detailed overview of impulse-based procedures of higher integration order and will discuss the method of their derivation in detail.

Since the impulse-based procedures are not directly related to the dynamic simulation methods based on the solution of differential equation systems, one cannot compare the truncation error orders directly. In the sequel, we shall see that it is not possible to compare our integration orders obtained for the impulse method with the orders of the truncation error as discussed in numerics when solving differential equations.

To gain an insight into the numerical performance of the impulse-based procedures of higher order, we perform test simulations. We simultaneously use ten different dynamic simulation methods, which are described in detail in Section 5. Five impulse-based procedures of different integration orders are among them as well as five standard methods on the basis of the solution of differential equations. Four of the procedures work with the Lagrange multiplier method with and without different stabilization techniques and one procedure uses the reduced coordinates approach. To generate comparable results, we use the standard Runge-Kutta method of fourth order for the solution of differential equations. For numeric simulations we define four different relatively simple mechanical models. Among them, there are two mathematical pendulums, a chaotic double pendulum and an even more chaotic triple pendulum. They are described in more detail in Section 4.

We are only interested in experimentally obtained accuracy levels and not in speed and accuracy tradeoffs which play a prominent role in numerics. We believe that the results presented here are not only of interest to the computer graphics community (computer animation, virtual reality) [Bender et al. 2005], but also to mechanical engineering since well-established comparative evaluations of different dynamic simulation methods are hardly found in the literature.

2. Impulse-based dynamic simulation of higher integration orders

The procedures of higher integration order can be deduced without a detailed look at the impulse-based dynamic simulation method. Essentially, these formulae are integration formulae and are not directly related to the impulse-based dynamic simulation method.

Impulses are the result of the integration of forces $F(t) \in \mathbb{R}^3$:

$$I := \int_{t_1}^{t_2} F(t)dt\,.$$

Whereas a force $F(t)$ continuously accelerates a point mass $p(t)$ with mass m according to $\ddot{p}(t) = (1/m) \cdot F(t)$, impulses or impulsive forces do not have a continuous effect on a point mass but at discrete times only. An impulse I then provides the point mass with an instantaneous increment in velocity with the direction and magnitude $(1/m) \cdot I$. The impulse-based dynamic simulation method utilizes impulses to achieve the same effects as when applying continuous forces. The discrete nature of impulses results in significant simplification of the dynamic simulation method because it is possible to largely avoid the solution of differential equations. Discrete application of impulses is certainly comparable with the discretization used when differential equations are solved numerically.

In the following, we develop a methodology for the derivation of formulae of higher order. To calculate an error estimation for the impulse-based simulation method, we solve the very simple differential equation (Newton's second law)

$$\ddot{p}(t) = \frac{1}{m} F(t) \tag{2.1}$$

for a mass point of mass *m* accelerated by a force *F(t)* and for initial values *p(0)* and $\dot{p}(0)$ in the time interval $t \in [0,h]$. We assume that *F(t)* is known and is given as a power series

$$F(t) = \sum_{i=0}^{\infty} a_i t^i = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \ldots \quad (2.2)$$

We first solve (2.1) by an impulse-based method for the time interval $t \in [0,h]$. For the impulse we obtain the integration result

$$I(0,h) = \int_0^h F(t)dt = a_0 h + \frac{a_1}{2} h^2 + \frac{a_2}{3} h^3 + \ldots .$$

An impulse $I(0,h)$ computed for a time interval $t \in [0,h]$ is an average value, which must be applied at time $t = h/2$ - in the center of the time interval - in order to obtain a good approximation $p_1(h)$ at time *h* to the correct solution of (2.1). Using this method, we arrive at

$$p_1(h) = p(0) + h\dot{p}(0) + \frac{h}{2m} I(0,h) =$$

$$= p(0) + h\dot{p}(0) + \frac{1}{m}\left(\frac{a_0}{2} h^2 + \frac{a_1}{4} h^3 + \frac{a_2}{6} h^4 + \ldots\right).$$

If we integrate (2.1) directly, we get the exact solution

$$p_{exact}(h) = p(0) + h\dot{p}(0) + \frac{1}{m}\left(\frac{a_0}{2} h^2 + \frac{a_1}{6} h^3 + \frac{a_2}{12} h^4 + \ldots\right). \quad (2.3)$$

Now, if one compares the coefficients to the powers of *h*, agreement up to terms of order 2 is evident and the difference starts with terms of h^3 ($\frac{a_1}{4m} \cdot h^3$ compared to $\frac{a_1}{6m} \cdot h^3$).

This can be stated as $p_{exact}(h) = p_1(h) + O(h^3)$. We have observed this error behavior in numerous test simulations. The error term of $O(h^3)$ was also proven in [Schmitt et al. 2005] using a completely different method.

A more accurate result can be obtained by first computing an impulse $I(0,h/2)$ for the subinterval [0, *h*/2], afterwards the impulse $I(h/2,h)$ for the subinterval [*h*/2, *h*], and application of these impulses at time *h*/4 (for I_1) and 3*h*/4 (for I_2). With this two-step method, we obtain

$$I(0,h/2) = \int_0^{h/2} F(t)dt = \frac{a_0}{2} h + \frac{a_1}{8} h^2 + \frac{a_2}{24} h^3 + \ldots$$

as well as $I(h/2,h) = \int_{h/2}^{h} F(t)dt = \frac{a_0}{2}h + \frac{3a_1}{8}h^2 + \frac{7a_2}{24}h^3 + \ldots$. Thus

$$p_2(h) = p(0) + h\dot{p}(0) + \frac{1}{m}(\frac{3h}{4}I(0,h/2) + \frac{h}{4}I(h/2,h))$$

$$= p(0) + h\dot{p}(0) + \frac{1}{m}(\frac{a_0}{2}h^2 + \frac{3a_1}{16}h^3 + \frac{5a_2}{48}h^4 + \ldots).$$

A comparison with $p_{exact}(h)$ still shows an error term of order $O(h^3)$. A substantially better approximation for $p_{exact}(h)$ can be computed by

$$p_{1+2}(h) = (-1/3)\,p_1(h) + (4/3)\,p_2(h)$$

$$= p(0) + h\dot{p}(0) + \frac{1}{m}(\frac{a_0}{2}h^2 + \frac{a_1}{6}h^3 + \frac{a_2}{12}h^4 + O(h^5)).$$

A comparison with (2.3) confirms equality with the exact solution up to terms of $O(h^4)$, i.e., $p_{exact}(h) = p_{1+2}(h) + O(h^5)$. The weights (- 1/3) and (4/3) used above can be determined quite simply by solving a small system of linear equations.

The method just used can be continued for higher orders. With $I(\alpha,\beta) = \int_{\alpha}^{\beta} F(t)dt$, we can define the three-step solution:

$$p_3(h) = p(0) + h\dot{p}(0) + \frac{1}{m}(\frac{5h}{6}I(0,h/3) + \frac{3h}{6}I(h/3,2h/3) + \frac{h}{6}I(2h/3,h)) \ .$$

With suitably determined weights, we define

$p_{1+2+3}(h) = (5/120)\,p_1(h) - (128/120)\,p_2(h) + (243/120)\,p_3(h)$.

Comparison with the exact solution delivers $p_{exact}(h) = p_{1+2+3}(h) + O(h^7)$. With

$$p_4(h) = p(0) + h\dot{p}(0) + \frac{1}{m}(\frac{7h}{8}I(0,h/4) + \frac{5h}{8}I(h/4,h/2)$$

$$+ \frac{3h}{8}I(h/2,3h/4) + \frac{h}{8}I(3h/4,h))$$

and

$$p_5(h) = p(0) + h\dot{p}(0) + \frac{1}{m}(\frac{9h}{10}I(0,h/5) + \frac{7h}{10}I(h/5,2h/5)$$

$$+ \frac{5h}{10}I(2h/5,3h/5) + \frac{3h}{10}I(3h/5,4h/5) + \frac{h}{10}I(4h/5,h)),$$

we can define

$$p_{1+2+3+4}(h) = -(1/360)\,p_1(h) + (16/45)\,p_2(h) - (729/280)\,p_3(h) + (1024/315)\,p_4(h)$$

where $p_{exact}(h) = p_{1+2+3+4}(h) + O(h^9)$, and

$$p_{1+2+3+4+5}(h) = (1/8640)\,p_1(h) - (64/945)\,p_2(h) + (6561/4480)\,p_3(h) - (16384/2835)\,p_4(h) + (390625/72576)\,p_5(h)$$

where $p_{exact}(h) = p_{1+2+3+4+5}(h) + O(h^{11})$ respectively.

The weight coefficients occurring in the formulae are easily derived by simple linear systems. We assume that the method can be continued further on. For practical applications, formulae with truncation errors less than $O(h^{11})$ are not of interest.

These results are only valid for mass points accelerated by known forces. This is different from the situation we have with the impulse-based simulation because impulses are determined in such a way that the joint conditions are fulfilled again after a time step. Nevertheless, one can combine the integration formulae with the impulse-based dynamic simulation of mass point systems and analyze the gain in accuracy. In addition one has to switch to the middle step procedure. It is defined as follows: With the middle step procedure, the mass point system is in a consistent condition at the beginning of a time step. Then a ballistic motion takes place for a time interval of $h/2$. At this point, impulses are calculated and applied in order to satisfy all constraints at time *h* with the help of a look ahead procedure. After sufficient correction of constraint errors, a further ballistic motion takes place for a time interval of $h/2$ and is then finished by a velocity correction. This brings the mechanical system back into a consistent condition finally.

The mass point positions p_1, p_2, p_3, p_4 as well as p_5 introduced above are then exchanged by:

$\breve{p}_1(h)$ with one middle step of step size h,

$\breve{p}_2(h)$ with two consecutive middle steps of size $h/2$,

$\breve{p}_3(h)$ with three middle steps of $h/3$ etc.

After inserting these mass point positions into the formulae derived above, we arrive at formulae like

$$\breve{p}_{1+2+3}(h) = (5/120)\,\breve{p}_1(h) - (128/120)\,\breve{p}_2(h) + (243/120)\,\breve{p}_3(h),$$

where $\breve{p}_{1+2+3}(h)$ should be more accurate than $\breve{p}_1(h)$ since we can expect a truncation error of $O(h^7)$, while the truncation error of $\breve{p}_1(h)$ is only of order 3.

Our numerical simulation results presented in Section 6 do not confirm this. One cannot compare our formulae of higher integration order with the truncation orders given for numerical procedures when solving differential equations.

It is obvious that the impulse-based methods of higher order can be implemented in a straightforward manner. If the function $Integrate(p(t),h) \to p(t+h)$ defines a middle step of the dynamic simulation, then

$$\breve{p}_1(h) := Integrate(p(0),h),$$
$$\breve{p}_2(h) := Integrate(Integrate(p(0),h/2),h/2),$$
$$\breve{p}_3(h) := Integrate(Integrate(Integrate(p(0),h/3),h/3),h/3)$$
etc.

and this considerably simplifies an implementation of the higher order codes.

3. Higher order formulae for angular velocities ω

The impulse-based procedures of higher integration order deduced above are applicable for mass point systems. Their implementation is rather simple. However, we also want to use the formulae of higher order for the impulse-based simulation of multi-body systems. Since we are working with the Newton-Euler formalism, we can split the motion of a rigid body into the motion of its center of mass and the change of the angular velocity ω. The formulae of higher order are clearly applicable to the centers of mass. The same applies to the orientation matrix R, whose column vectors are the axes of the body's coordinate system.

Yet, the question "How to proceed with the $\omega's$?" remains. We call it the "omega problem". We studied and compared numerical simulations of multi-body systems with equivalent point mass systems. It turned out that spin vectors ω are transformed to higher order by exactly the same formulae and weights as derived for point masses in Section 2. In order to confirm this, we must solve Euler's differential equation

$$(3.1) \quad \dot{\omega}(t) = J^{-1} \cdot ((J \cdot \omega(t)) \times \omega(t)) + J^{-1} \cdot D(t)$$

by the power series method. Here $D(t)$ is the total torque acting on the rigid body. It replaces $F(t)$ appearing in (2.1). It is sufficient to carry out the calculation in body space coordinates with a constant and diagonalized inertia tensor J. For the *n*th derivative $\omega^{(n)}(0)$ of $\omega(t)$ at t=0, we obtain the following recursive formula:

$$\omega^{(n)}(0) = J^{-1} \cdot \left(\sum_{i=0}^{n-1} \left(\frac{n-1}{i} \right) (J \cdot \omega^{(i)}(0)) \times \omega^{(n-i-1)}(0) \right) + J^{-1} \cdot D^{(n-1)}(0) \ .$$

Without torque we obtain

$$\bar{\omega}^{(n)}(0) = J^{-1} \cdot \left(\sum_{i=0}^{n-1} \left(\frac{n-1}{i} \right) (J \cdot \bar{\omega}^{(i)}(0)) \times \bar{\omega}^{(n-i-1)}(0) \right).$$

The exact solution includes torque and is given by the power series

$$\omega_{exact}(h) = \sum_{i=0}^{n-1} \frac{\omega^{(i)}(0)}{i!} h^i + O(h^n) \cdot$$

The torque-free ballistic motion of the rigid body in a time interval h leads to

$$Ball(\omega(t_0), h) = \omega(t_0 + h) = \sum_{i=0}^{n-1} \frac{\bar{\omega}^{(i)}(t_0)}{i!} h^i + O(h^n) \ .$$

Note the $\bar{\omega}'s$ inside the formula. The torque impulse is calculated by

$$I_D(\alpha, \beta) := \int_{\alpha}^{\beta} J^{-1} \cdot D(t) dt \, .$$

The new ω_1 after a time step *h* in the impulse-based dynamic simulation is computed by the middle step procedure as follows:

$$\omega_{1a} := Ball(\omega(0), h/2);$$
$$\omega_{1b} := \omega_{1a} + I_D(0, h);$$
$$\omega_1 := Ball(\omega_{1b}, h/2).$$

Two consecutive steps of size h/2 lead to:

$$\omega_{2a} = Ball(\omega(0), h/4); \quad \omega_{2b} = \omega_{2a} + I_D(0, h/2);$$
$$\omega_{2c} = Ball(\omega_{2b}, h/2); \quad \omega_{2d} = \omega_{2c} + I_D(h/2, h);$$
$$\omega_2 = Ball(\omega_{2d}, h/4).$$

With efficient formula manipulators like Mathematica™, we can prove that

$$\omega_{exact}(h) = \omega_{1+2}(h) + O(h^5),$$

where we treat the starting parameters $\omega(0)$, J and $D(t)$ as variables and compare the terms $\omega_{1+2}(h) = (-1/3)\omega_1(h) + (4/3)\omega_2(h)$ and $\omega_{exact}(h)$.

Thus we have the same integration formulae for the angular velocity vectors ω as for point masses, which we have shown up to order $O(h^9)$ with Mathematica™.

It is very amazing to learn that the formulae derived in Section 2 are also applicable to spin vectors. A possible reason for this may be that the angular momentum and the linear momentum have exactly the same structure. For the impulse $p(t) = m \cdot v(t)$ of a mass point with mass *m* and velocity $v(t)$, we have $\dot{p}(t) = K(t)$, where $K(t)$ is the force applied. For the angular momentum $L(t) = J \cdot \omega(t)$, we have $\dot{L}(t) = D(t)$,where *D*(*t*) is the torque affecting the rigid body. We conjecture that the complete analogy of these formulae for both motion quantities is the reason for the validity of the integration formulae of higher order deduced above.

4. The dynamically simulated mechanical models and accuracy measurements

Methodically, our course of action is that we simulate relatively simple mechanical models, e.g., the mathematical pendulum, the double pendulum and the triple pendulum, see Figures 1a, 1b and 1c. The last two are chaotic systems. Dynamic simulations for these chaotic models with high accuracy over a longer period of time are practically impossible (lack of a Lipschitz condition). All our models are point mass systems, but it should be noted that, for dynamic simulations, point mass systems are equivalent to rigid body systems, see [Schmitt et al. 2005a]. The parameter values of the models are:

Pendulum10: Mathematical pendulum with a length of 1 m, a mass of 1 kg and a maximum amplitude of $10°$, planar motion,

Pendulum90: Like Pendulum10, but maximum amplitude of 90 degrees, planar motion,

Double Pendulum: Distance of the masses 1 m, masses 1 kg, starting configuration horizontally stretched to the right, leading to a planar motion,

Triple Pendulum: Distance of the 3 masses 1 m each, masses 1 kg, starting configuration horizontally stretched to the right, planar motion.

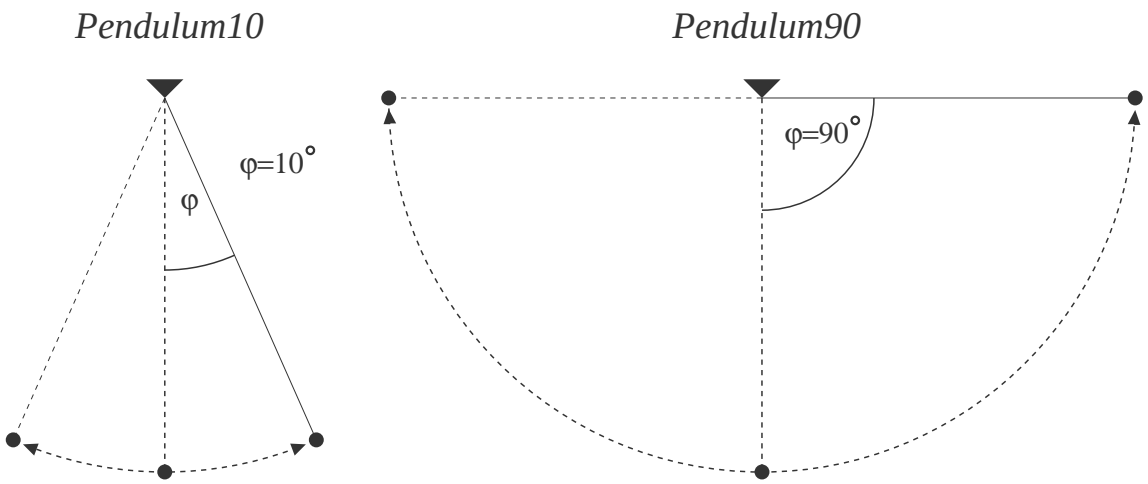

Figure 1a

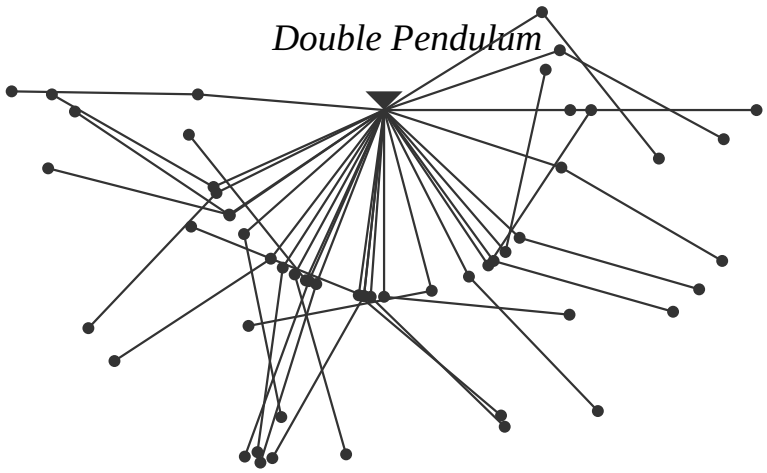

Figure 1b

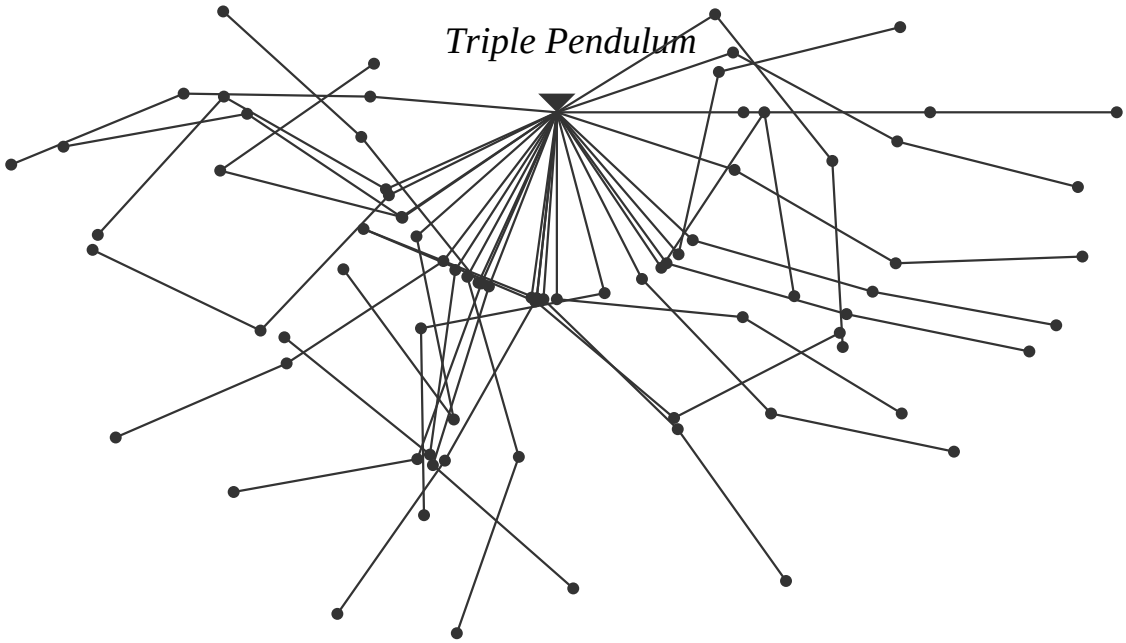

Figure 1c

If these models are simulated dynamically without friction or other external forces except gravity, the total energy consisting of kinetic and potential energy should remain constant. Thus, if the total energy is E_0 at the start of the simulation and the total energy of the simulated model after the *i*th time step is E_i, then we define the quantity "energy drift" as

$$\textit{Energy Drift} := ED := (1/n) \sum_{i=1...n} \left| E_i - E_0 \right| ,$$

where *n* is the total number of time steps recorded. To test the stability of the numerical simulation methods, we always simulate over a time interval of 60 seconds. A completely accurate dynamic simulation must result in *ED=0.* With the Pendulum10 and Pendulum90 models, *ED* is a good indicator for the accuracy of the simulation. Nevertheless, it is well known that an energy drift of *ED=0* does not guarantee a perfectly accurate simulation.

For the Pendulum10 and Pendulum90 models, we can also determine the deviations from the correct oscillation time. With formulae from theoretical physics, the oscillation time for Pendulum10 is given as $T_{10} = 2.00989262729860$ s and for Pendulum90 as $T_{90} = 2.36784194757623$ s, where we used g=9.81, which is also used during the numerical computations. By the choice of time steps $h = T_\varphi / k$ for integers *k* and $\varphi = 10$ and 90 respectively, one can measure very small deviations from the theoretical oscillation time by observing the perpendicular crossover of the simulated pendulum:

$$Oscillation\ Time\ Drift = OTD := (1/m) \sum_{i=1...m} \left| x(t = i \cdot T_\varphi) \right|$$

Here, *x(t)* is the *x* coordinate of the respective pendulum at time *t* and we start the numerical simulation in a perpendicular position with *x*(0)=0. A numerical simulation with the theoretically computed oscillation period thus results in *OTD*=0 and deviations result in *OTD*>0.

With the chaotic models Double Pendulum and Triple Pendulum, the measure ED is not really useful because it is often observed with these models that energy errors compensate each other later on due to chaotic influences. For these models, we therefore use the measure

$$Energy\ Increment\ Drift := EID := (1/n) \sum_{i=1...n} \left| E_i - E_{i-1} \right|,$$

where we sum up the absolute values of energy changes at time steps $i = 1...n$.

5. The dynamic simulation methods used

The models described above are simulated with a total of ten different simulation methods or codes:

GC4: Generalized coordinates, which is the method of reduced or minimal coordinates simulated with the standard Runge-Kutta method of order 4. We did not use adaptive time steps, but always used constant step sizes *h*. Whenever it is possible to write down the differential equations of the dynamic motion of a multibody system in a minimal set of generalized coordinates, i.e., in such a way that for each degree of freedom these equations contain only one parameter, then this should lead to the most accurate simulation results because the computation of constraint forces is completely eliminated.

For a mathematical pendulum of length 1 m, a respective equation reads $\ddot{\varphi} = -g \cdot \sin(\varphi)$, where the generalized coordinate φ is the angle between the pendulum and the perpendicular axis. For more complex models, e.g., the triple pendulum, the formulae are already so complex that they should be generated with a system like Mathematica™, e.g., using Lagrange's equation of the second kind. Our results for the double pendulum are

$$\ddot{\varphi}_1 = \frac{(g \cdot (3 \cdot \mathrm{Sin}(\varphi_1) + \mathrm{Sin}(\varphi_1 - 2 \cdot \varphi_2)) + \mathrm{Sin}(2 \cdot (\varphi_1 - \varphi_2)) \cdot \dot{\varphi}_1^2 + 2 \cdot \mathrm{Sin}(\varphi_1 - \varphi_2) \cdot \dot{\varphi}_2^2)}{(-3 + \mathrm{Cos}(2 \cdot (\varphi_1 - \varphi_2)))},$$

$$\ddot{\varphi}_2 = \frac{(-2 \cdot \mathrm{Sin}(\varphi_1 - \varphi_2) \cdot (2 \cdot g \cdot \mathrm{Cos}(\varphi_1) + 2 \cdot \dot{\varphi}_1^2 + \mathrm{Cos}(\varphi_1 - \varphi_2) \cdot \dot{\varphi}_2^2))}{(-3 + \mathrm{Cos}(2 \cdot (\varphi_1 - \varphi_2)))}.$$

For the triple pendulum, we obtained

$\ddot{\varphi}_1 =$

$(-(g \cdot (-5+\text{Cos}(2 \cdot (\varphi_2 - \varphi_3))) \cdot \text{Sin}(\varphi_1))+2 \cdot g \cdot \text{Sin}(\varphi_1 - 2 \cdot \varphi_2)+2 \cdot \text{Sin}(2 \cdot (\varphi_1 - \varphi_2)) \cdot \dot{\varphi}_1^2 +$

$2 \cdot \text{Sin}(\varphi_1 - \varphi_2) \cdot (2 \cdot \dot{\varphi}_2^2 + \text{Cos}(\varphi_2 - \varphi_3) \cdot \dot{\varphi}_3^2))/$

$(-5+2 \cdot \text{Cos}(2 \cdot (\varphi_1 - \varphi_2))+\text{Cos}(2 \cdot (\varphi_2 - \varphi_3))),$

$\ddot{\varphi}_2 =$

$(g \cdot \text{Cos}(\varphi_1) \cdot (-7 \cdot \text{Sin}(\varphi_1 - \varphi_2)+\text{Sin}(\varphi_1 + \varphi_2 - 2 \cdot \varphi_3))+(-7 \cdot \text{Sin}(\varphi_1 - \varphi_2)+\text{Sin}(\varphi_1 + \varphi_2 - 2 \cdot \varphi_3)) \cdot \dot{\varphi}_1^2 +$

$(-2 \cdot \text{Sin}(2 \cdot (\varphi_1 - \varphi_2))+\text{Sin}(2 \cdot (\varphi_2 - \varphi_3))) \cdot \dot{\varphi}_2^2 + (-\text{Sin}(2 \cdot \varphi_1 - \varphi_2 - \varphi_3)+3 \cdot \text{Sin}(\varphi_2 - \varphi_3)) \cdot \dot{\varphi}_3^2)/$

$(-5+2 \cdot \text{Cos}(2 \cdot (\varphi_1 - \varphi_2))+\text{Cos}(2 \cdot (\varphi_2 - \varphi_3))),$

$\ddot{\varphi}_3 =$

$(-2 \cdot \text{Sin}(\varphi_2 - \varphi_3) \cdot (2 \cdot (\text{Cos}(\varphi_1 - \varphi_2) \cdot (g \cdot \text{Cos}(\varphi_1)+\dot{\varphi}_1^2)+\dot{\varphi}_2^2)+\text{Cos}(\varphi_2 - \varphi_3) \cdot \dot{\varphi}_3^2))/$

$(-5+2 \cdot \text{Cos}(2 \cdot (\varphi_1 - \varphi_2))+\text{Cos}(2 \cdot (\varphi_2 - \varphi_3))),$

where the angles $\varphi_1, \varphi_2, \varphi_3$ of the rods form the minimal set of generalized coordinates.

In cases of larger systems to be analyzed dynamically, the method of minimal coordinates generates very large expressions for the differential equations, and simulations on this basis can be laborious.

LM4: Lagrange multiplier method numerically solved with the standard Runge-Kutta method of order 4. This well-known and widely used simulation method is scalable and can be automated. That means one only has to describe the inertia tensors and the joints of mechanical models in Cartesian coordinates, and there is no limit to the complexity of the mechanical models. All further steps, e.g., the generation of the differential equations and their numerical solution, are easily implemented. These properties permit the integration of this dynamic simulation method as a sub-module in computer animation and virtual reality systems, where larger models, e.g., walking machines, have to be simulated frequently. Our calculation of the Lagrange multipliers for point mass systems is based on the Taylor expansion method as described in [Schmitt et al. 2005a].

The only serious disadvantage of this method is the constraint drift problem. During the simulation, small numerical inaccuracies with respect to constraint conditions grow steadily and cannot be corrected by the basic LM algorithm. This is due to the fact that neither the consistency of the mechanical system with respect to distance constraints nor the velocity constraints are maintained in the course of the simulation. More details are given in papers and books on the numeric solution of differential algebraic equations related to dynamic simulations and also in [Schmitt et al. 2005a].

LMBS4: Similar to LM4, but with the well known and often cited stabilization method of [Baumgarte 1972]. We exactly used the stabilization terms given in Baumgarte's "Example 1" and also his parameter settings $\alpha = 10, \beta = 10$.

LMV4: Similar to LM4, but the velocity errors across constant distance joints are corrected at the end of each integration step. To do this, we exactly use the impulse-based algorithm which we also use in the implementations of our impulse-based dynamic simulation methods, for details see [Schmitt et al. 2005a].

LMVD4: Similar to LMV4, but not only a correction of velocity errors is done at the end of each integration step, but also an impulse-based projection method correcting distance errors. In order to cope with the drift phenomenon, projection methods and other regularization methods like Baumgarte's stabilization are frequently discussed in the literature, see e.g. [Ascher et al. 1994], [Eich-Soellner and Führer 1998],[Schwerin 1999] and many others. Here we use a new and very promising technique that is totally based on impulse methods. An integration step is composed of the following phases:

(a) Standard integration step with the LM4 method resulting in mass point positions and velocities $P(t+h), \dot{P}(t+h)$.

(b) Save $\dot{P}_{save}(t+h) := \dot{P}(t+h)$ and set the velocities temporarily to zero: $\dot{P}(t+h) := (0,0,0)$.

(c) Switch off gravity and carry out an impulse-based integration step as described in [Schmitt et al. 2005b]. After this, all constraint errors are eliminated. All constraint distances are free of drift. The time parameter is reset to $t+h$.

(d) Reconstitute the old velocities $\dot{P}(t+h) := \dot{P}_{save}(t+h)$ and finally apply the impulse-based velocity correction. The dynamic system is now in a consistent state.

The procedure LMVD4 derived from LM4 has the best stability and accuracy behavior in the family of our LM methods. It is relatively efficient because only one system of linear equations needs to be solved for each step (c) and (d). On the other hand, LM4 needs to solve four linear systems to compute the Lagrange multipliers with the standard Runge-Kutta method.

Imp2, Imp4, Imp6, Imp8, Imp10: Impulse-based dynamic simulation methods of integration orders $O(h^2), O(h^4), O(h^6), O(h^8), O(h^{10})$. A detailed derivation of these methods is given in Section 2. These higher integration orders cannot be compared with the respective orders of the truncation error as is known from numerical analysis. It will turn out that the higher integration orders only have a minor influence on the order of the asymptotic truncation error, but they have a significant influence on the magnitude of the error functions.

6. Simulation results and discussion

The numerical simulations were carried out with double precision reals (64 bit) for a total time of 60 seconds and with 12 different time step sizes in the range from about h=0.00125 s to h=0.08 s. A procedure is marked by "fail" in a diagram if the maximal constraint distance error has grown in the course of the 60 seconds to values greater than 1 m which is by far too large for a realistic dynamic simulation.

We approximate the error curves simply by a function $f(h) := c \cdot h^a$. If f is given by the two distinct values *h1=0.005, h2=0.04* and *w1=f(h1), w2=f(h2),* then the order a is given

by

$$a = \big(\log(w1) - \log(w2)\big) / \big(\log(h1) - \log(h2)\big)$$

and the factor c by $c = w2 / \big(h2\big)^a$. In the sequel, we will simply use the terms "order" and "factor" when the growth characteristics of functions are discussed.

Pendulum10: This model has a calm dynamic behavior. In contrast to Pendulum90, there are no fail events, not even with LM4. Here and in all the other diagrams, the impulse-based methods perform in accordance with their integration orders. Note that, theoretically, Imp2 is of order $O(h^2)$ whereas GC4 and all the LM methods are of order $O(h^4)$.

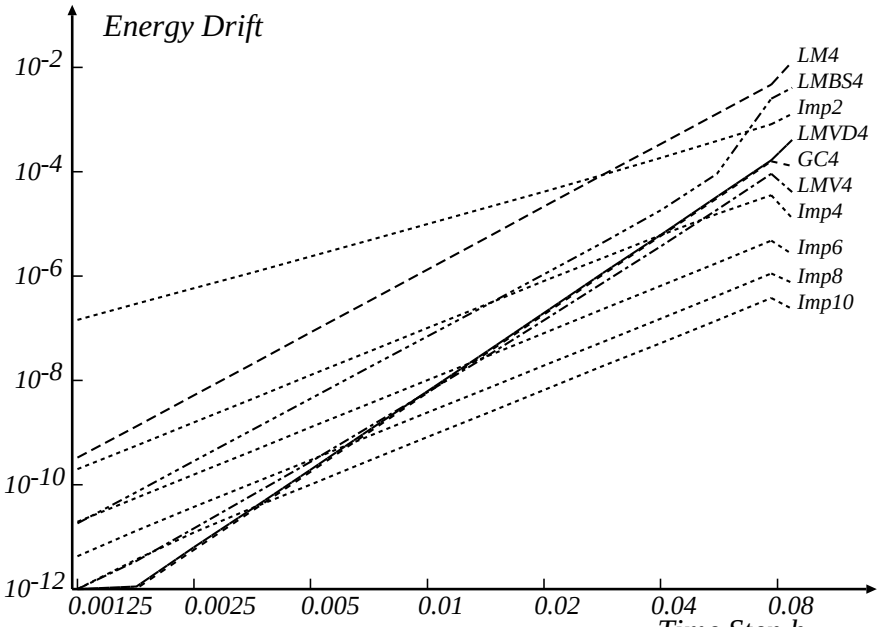

Figure 2 : Model Pendulum10

Energy drift, Fig. 2: The methods GC4, LMV4 and LMVD4 have about the same characteristics whereas LM4 and LMBS4 have a larger energy drift. Since the impulse-based methods Imp4, Imp6, Imp8 and Imp10 have the same inclination, they should have nearly the same order, but different factors. This is confirmed by the following table:

Model: Pendulum10	***Function: Energy Drift***	
GC4	Order: 5.00	Factor: 57.489726
LM4	Order: 4.00	Factor: 133.908765
LMV4	Order: 4.57	Factor: 9.375275
LMVD4	Order: 4.99	Factor: 59.295039
LMBS4	Order: 4.00	Factor: 7.266698
Imp2	Order: 2.10	Factor: 0.160045
Imp4	Order: 2.97	Factor: 0.086561
Imp6	Order: 3.00	Factor: 0.010339
Imp8	Order: 3.00	Factor: 0.002436
Imp10	Order: 3.00	Factor: 0.000831

Since the energy drift is not directly proportional to the truncation error, the orders differ slightly from the theoretical numbers. All the methods Imp4 up to Imp10 have the same order $O(h^3)$, but their factors shrink substantially for the higher integration orders. Note also the differences between the LM methods, where LMVD4 is finally equivalent to GC4.

Oscillation Time Drift, Fig. 3: The impulse-based methods show a very good performance even for small values of *h*. The oscillation time drift should be proportional

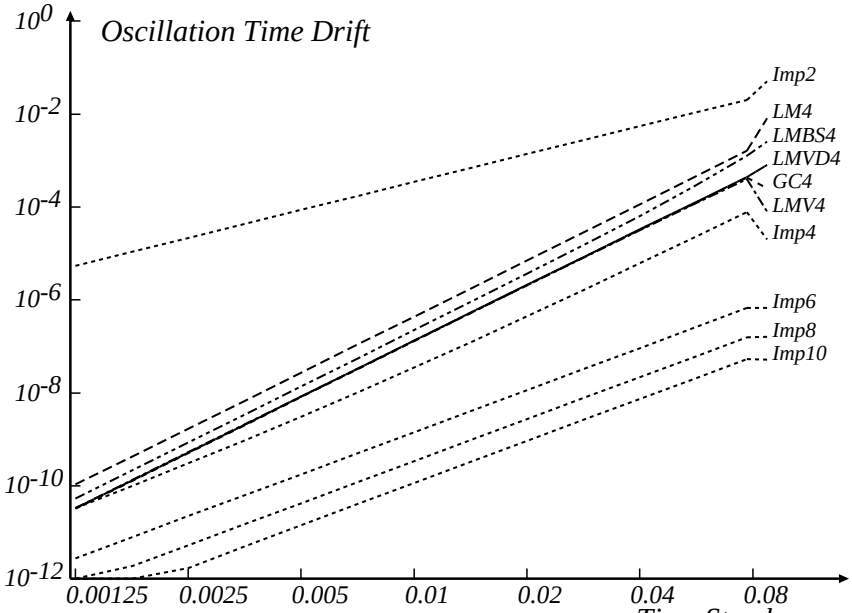

Figure 3 : Model Pendulum10

to the truncation error. This is confirmed by the following table, where all methods using Runge-Kutta are of $O(h^4)$. We do not know why the order of Imp4 is higher than those of the methods with higher integration order.

Model: Pendulum10 Function: Location Drift

GC4	Order:	3.99	Factor:	12.105792
LM4	Order:	4.02	Factor:	47.245186
LMV4	Order:	3.96	Factor:	10.994672
LMVD4	Order:	3.99	Factor:	12.362825
LMBS4	Order:	4.07	Factor:	31.294747
Imp2	Order:	2.00	Factor:	3.423206
Imp4	Order:	3.65	Factor:	0.790714
Imp6	Order:	3.00	Factor:	0.001450
Imp8	Order:	3.00	Factor:	0.000341
Imp10	Order:	3.00	Factor:	0.000116

Pendulum90: This model has greater velocities and accelerations than Pendulum10. Therefore, the errors are generally larger than with Pendulum10, see Figures 4 and 5. Here GC4 is the most accurate dynamic simulation method since all other procedures

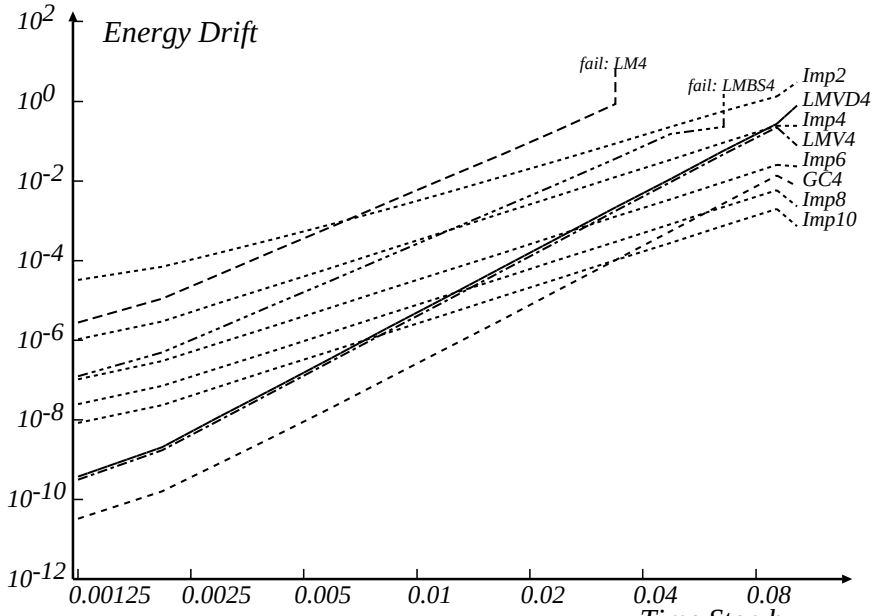

Figure 4 : Model Pendulum90

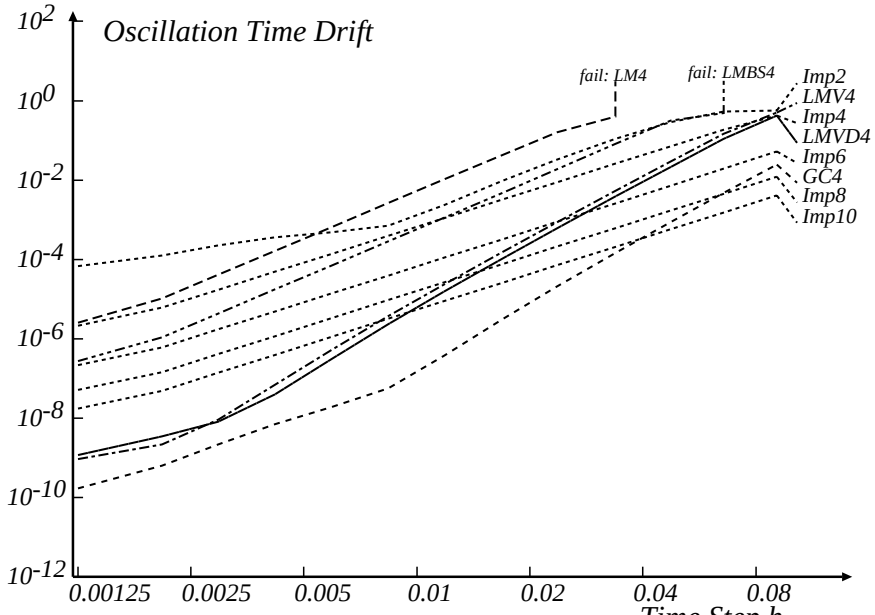

Figure 5 : Model Pendulum90

have to deal with greater constraint forces. LM4 and LMBS4 fail for larger time steps, whereas the LM methods stabilized with impulse techniques (LMV4, LMVD4) have, for small time steps, a better performance than even the best impulse method. However, note that these Lagrange multiplier methods are stabilized by impulse methods.

```
Model: Pendulum90   Function: Location Drift
  GC4    Order: 5.16  Factor:   5960.114141
  LMV4   Order: 5.25  Factor:  268724.937027
  LMVD4  Order: 5.36  Factor:  270707.968280
  LMBS4  Order: 4.04  Factor:  69444.354991
  Imp2   Order: 3.08  Factor:   3480.824040
  Imp4   Order: 3.00  Factor:    678.967591
  Imp6   Order: 3.00  Factor:     68.549409
  Imp8   Order: 3.00  Factor:     16.013713
  Imp10  Order: 3.00  Factor:      5.433129
```

The table with the order numbers shows some differences with model Pendulum10. Whereas the impulse methods show exactly the same orders with the exception of Imp2, GC4 as well as the stabilized LM methods show orders higher than 5. We do not have an explanation for this.

Constraint drift Fig. 6, 7: Only the three procedures LM4, LMBS4 and LMV4 are burdened with a constraint drift. The numbers given in the figures are mean values for the whole 60 seconds of simulation time. For this reason, they are relatively small. A fail event is detected if the constraint error is greater than 1 m. Note that LMVD4 is free of drift due to the impulse-based projection method.

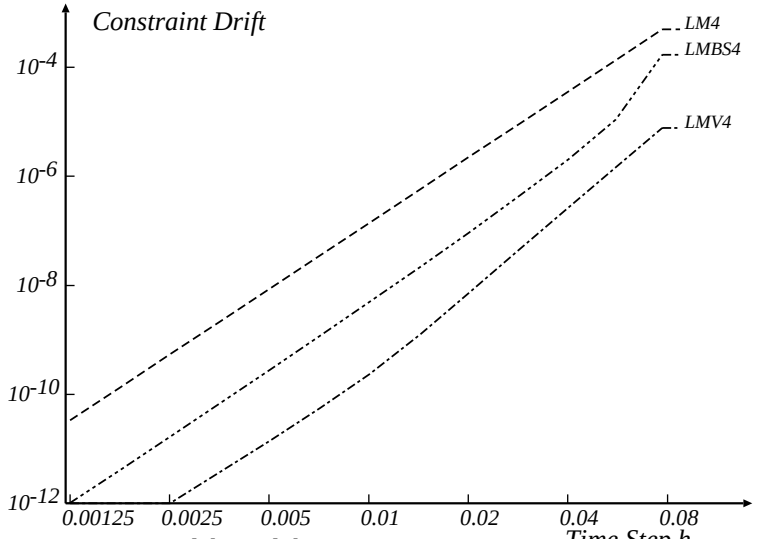

Figure 6 : Model Pendulum10

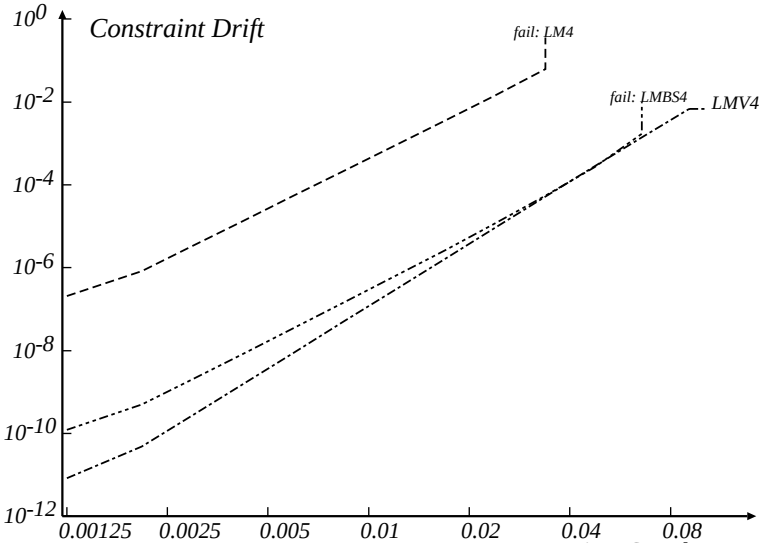

Figure 7 : Model Pendulum90

Double and Triple Pendulum: The chaotic nature of these models leads to curves (Figures 8, 9) which are a little bit wavy and not as smooth as for the mathematical pendulum although the measure EID=Energy Increment Drift is a sort of smoothing of ED. The fail events clearly document the ranking LM4, LMBS4, LMV4 with respect to instability. It is also of interest that the other methods of order 4 have very similar curves.

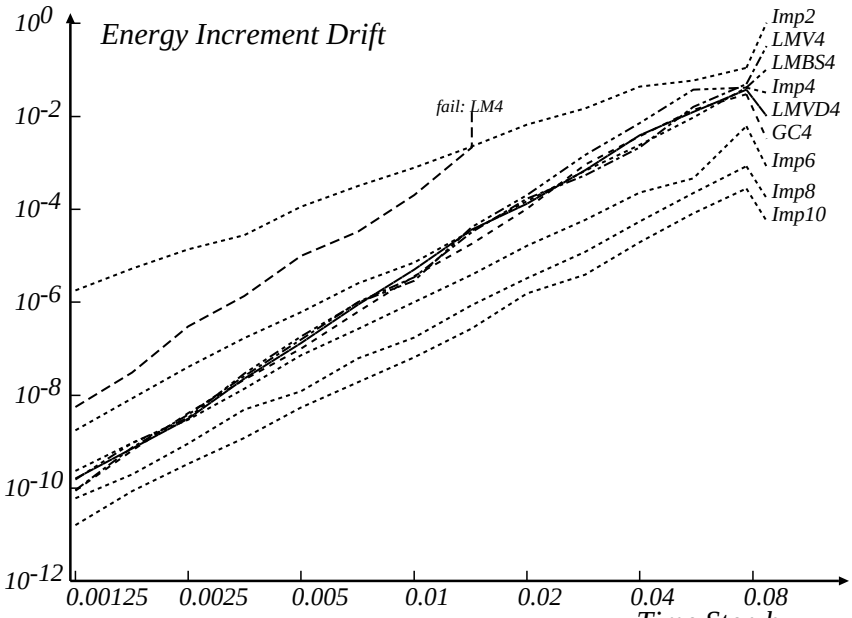

Figure 8 : Model Double Pendulum

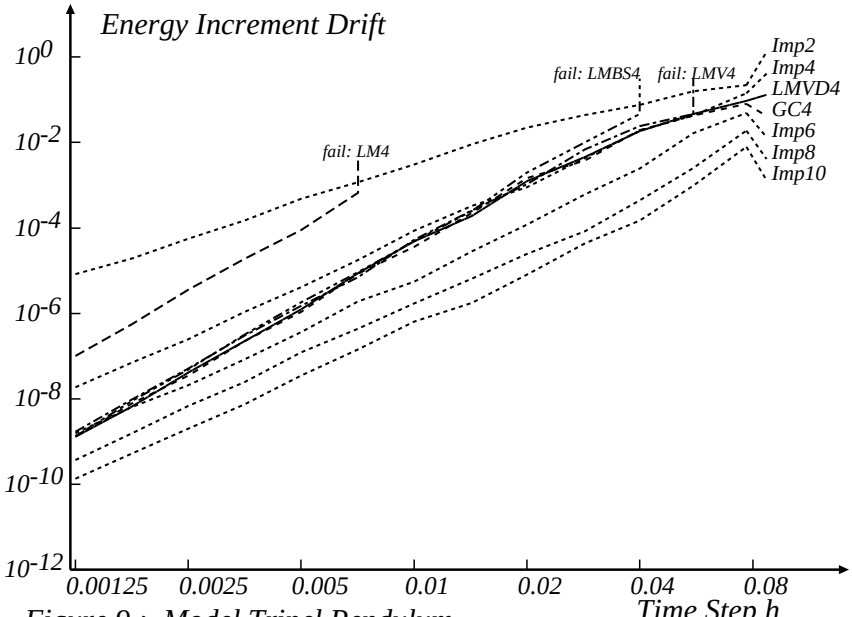

Figure 9 : Model Tripel Pendulum

7. Conclusion

The most important results of the comparative test simulations concern the impulse-based procedures of higher integration order. We derived these formulae of higher order by computing impulses through integration of a known force function. In the genuine impulse-based simulation, the force functions are unknown and impulses are determined by a look-ahead function and by constraint checking. The simulation results show that the procedures Imp4,…, Imp10 are all of order 3, whereas the standard Runge-Kutta code is of order 4. Nevertheless, a substantial increase in accuracy results due to the very small constant factors of the asymptotic truncation error functions. This shows up clearly in

Fig. 3, where Imp6, Imp8 and Imp10 have a very high accuracy. A further characteristic of the impulse-based procedures is their high stability, which is documented by the linear shape of the error functions. A further advantage is the simple and straightforward implementation of the impulse methods of higher integration order, which was demonstrated briefly at the end of Section 2.

The non-scalable procedure GC4 with minimal coordinates is certainly not a candidate for a general dynamic simulation module in computer graphics applications. This method is not really applicable in scenarios where collision and friction have to be modeled, and it cannot be automated in a straightforward manner as is the case for the impulse and LM methods. When compared to the other methods, the gain in accuracy is not very significant. Even with model Pendulum90, where GC4 performs very well, it is possible to obtain a comparable accuracy if one uses LMVD4 with a step size of about *h/2*.

A comparison between the procedures that are scalable and can be automated, i.e., the LM methods and the impulse methods, shows clear advantages with respect to the impulse methods, since the competitive methods LMV4 and LMVD4 are also stabilized using impulse methods. This statement is based on accuracy and energy drift statistics and not on a comparison of computing time.

Application of the standard Lagrange multiplier method (LM4) cannot be recommended because the simulations are unstable and can be highly inaccurate. The same is true to a certain extent for the LM method with Baumgarte stabilization. Only the fully stabilized method LMVD4 is a serious candidate for stable and accurate dynamic simulations. However, it should be noted that in order to implement LMVD4, one also has to implement the impulse method as part of the stabilization. From an implementer's point of view, it turned out to be much easier to first implement the impulse method because the LM method uses the same matrices for the linear systems as the impulse method.

References

Ascher, U. M., H. Chin, S. Reich (1994): Stabilization of DAEs and invariant manifolds, Numerische Mathematik 67(1994), 131-149.

Baumgarte J. (1972): Stabilization of constraints and integrals of Motion in dynamical systems. Comp. Meth. Appl. Mech. Eng. 1(1972), 1-16.

Bender, J., M. Baas, A. Schmitt (2003): Ein neues Verfahren für die mechanische Simulation in VR-Systemen und in der Robotik, 17. Symposium Simulationstechnik (ASIM 2003), 16.-19. September 2003, pp. 111-116, Magdeburg 2003.

Bender, J., D. Finkenzeller, A. Schmitt (2005): An impulse-based dynamic simulation system for VR applications, Proceedings of Virtual Concept 2005, 8.-10. November 2005, Biarritz.

Eich-Soellner, E., C. Führer (1998): *Numerical Methods in Multibody Dynamics*, B. G. Teubner, Stuttgart 1998.

Mirtich, B., und Canny, J. (1995): Impulse-based simulation of rigid bodies. Symposium on Interactive 3D Graphics, 181–188, 1995.

Schmitt, A. (2003): Dynamische Simulation von gelenkgekoppelten Starrkörpersystemen mit der Impulstechnik. Interner Bericht 2003-19, Fakultät für Informatik, Universität Karlsruhe.

Schmitt, A., J. Bender and H. Prautzsch (2005a): On the Convergence and Correctness of Impulse-Based Dynamic Simulation, Internal Report 2005/17, Fakultät für Informatik, Universität Karlsruhe.

Schmitt, A.. and J. Bender (2005b): Impulse Based Dynamic Simulation of Multibody Systems: Numerical Comparison with Standard Methods. Proc. of the Conference ADP-2005 (Automation of Discrete Production Engineering), Sozopol (Bulgaria), 9.-12. Sept. 2005.

Schwerin, R. von (1999): Multibody System Simulation: Numerical Methods, Algorithms, and Software. Lecture Notes in Computational Science and Engineering, Springer Verlag, Berlin 1999.

-.-.-.-