Fast Dynamic Simulation of Multi-Body Systems Using Impulses

Jan Bender and Alfred A. Schmitt

Institut für Betriebs- und Dialogsysteme, Universität Karlsruhe, Germany
{jbender, aschmitt}@ira.uka.de

Abstract

A dynamic simulation method for multi-body systems is presented in this paper. The special feature of this method is that it satisfies all given constraints by computing impulses. In each simulation step the joint states after the step are predicted. In order to obtain valid states after the simulation step, impulses are computed and applied to the connected bodies. Since a valid joint state is targeted exactly, there is no drift as the simulation proceeds in time and so no additional stabilisation is required. In previous approaches the impulses for a multi-body system were computed iteratively. Since dependencies between joints were not taken into account, the simulation of complex models was slow. A novel method is presented that uses a system of linear equations to describe these dependencies. By solving this typically sparse system the required impulses are determined. This method allows a very fast simulation of complex multi-body systems.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

There are many applications for the simulation of realistic mechanical behaviour, for example computer animation, virtual reality or the simulation of robots. It depends on the purpose of the application, if the simulation has to be very accurate or very fast or if a trade-off between accuracy and speed is needed. Usually a computer animation does not need the same degree of accuracy as the simulation of robots but it must run fast, whereas the simulation of a robot may run even several hours but the result must be accurate. For virtual reality applications it is necessary that the simulation always runs in real-time. Otherwise the virtual world appears less realistic. A dynamic simulation method has to support many different joint types to simulate complex models, like e.g. a robot in a virtual environment. For the simulation of such models in real-time the method has to be very fast. In this paper a fast simulation method for multi-body systems containing all kinds of joints is presented. All constraints are satisfied by applying impulses. Collisions and contacts are resolved by using the method presented in [BS06].

Impulse-based methods for the dynamic simulation of multi-body systems are easy to implement and very fast [BFS05, WTF06]. Another advantage of using impulses is that impulses change the velocities of the bodies directly. Hence no integration of continuous constraint forces is needed to determine the new positions of the bodies in a time step. These impulse-based methods work as follows. For every joint in the system a prediction of the joint state after the next simulation step is made. If the predicted joint state is not valid, an impulse is computed and applied in order to correct this. In an iterative process the computation of these correction impulses is continued until all predicted joint states in the system are valid. Then a simulation step is performed. If there exist velocity constraints that are not satisfied after the simulation step, they are solved by iteratively computing impulses. Since the impulse-based methods compute impulses in order to exactly obtain a valid joint state, no numerical drift occurs as the simulation proceeds in time. Rachel Weinstein et al. [WTF06] combined their impulse-based dynamic simulation with the collision and contact handling algorithm of Eran Guendelman et al. [GBF03]. Therefore they changed the typical order of a simulation step. With their method they are able to simulate complex scenes and attain visually plausible results. The method presented by Jan Bender et al. in [BFS05] is fast and provides accurate results. In [SB05] the accuracy of this method is compared to reduced coordinate (generalised coordinate) and Lagrange

multiplier methods. The proof that this impulse-based dynamic simulation method converges towards the exact solution is given in [SBP05]. The main difference between the methods of Weinstein et al. and Bender et al. lies in the way the correction impulses are computed that are needed in order to obtain a valid joint state. In the first method, in each iteration a prediction of the joint state is made first. Then a black box model [SNTH03,WG01] is used to project the predicted state to a desired state. Finally a nonlinear equation is solved with Newton iteration to determine the impulse that is required to reach the desired joint state. The method of Bender et al. uses a simplification instead of solving a nonlinear equation. Dependent on the predicted joint state an approximation of the required velocity change of the bodies is made. An appropriate impulse that causes exactly this velocity change is computed and applied. Since an approximation is used, the computation of impulses is continued in an iterative process which ends, when a valid joint state is reached. Velocity constraints are handled in a similar way in the methods of Weinstein et al. and Bender et al.

Bender et al. describe their method only for a spherical joint. In this paper, their method is extended in order to simulate several kinds of joints in an uniform way. It is shown how six basic joint types are simulated with their method. With these basic joints it is possible to remove all combinations of translational and rotational degrees of freedom between two rigid bodies. After the basic joints are introduced, combinations of these joints are discussed. By using a combination of two basic joints in a simulation all kinds of joints can be simulated. Since the method uses the simplification described above and therefore only linear equations have to be solved, the equations can also be written as a system of linear equations. In the same system dependencies between joints in a multi-body system can be described. Because of the approximations that are used, the system of linear equations must be solved multiple times until all joints get a valid state. But since the dependencies between the joints are taken into account, less impulses must be determined. Especially the accurate simulation of complex models with many joints runs much faster with the method presented in this paper. The goal of the presented method is to perform a fast and accurate simulation.

2. Related work

There exist several further methods for the handling of joint constraints in a dynamic simulation. The most important are the penalty method [dJB94], the Lagrange multiplier method and the simulation with reduced coordinates. The penalty method is easy to implement but it satisfies the constraints not exactly and is slow if a high degree of accuracy is required. The dynamic simulation with Lagrange multipliers is more complicated to implement. The internal forces that act in the joints are computed by solving a system of linear equations. David Baraff presented an algorithm which can solve this system in linear time [Bar96]. After the forces are computed, a system of differential equations must be integrated twice to get the new positions of the rigid bodies. The Lagrange multiplier method has a drifting problem which can be solved by an additional stabilisation, for example by the one proposed by Baumgarte [Bau72] or by post-stabilisation techniques [Asc97]. A survey of stabilisation techniques can be found in [Chi95, ACR95].

Reduced coordinates are a set of unconstrained independent coordinates. Holonomic constraints can be expressed in terms of these coordinates. The number of reduced coordinates is equal to the number of degrees of freedom of the simulated system. The determination of a set of such coordinates for systems without loops is well known [Fea87]. The advantages of the method are that the constraint forces do not need to be computed explicitly and no drift problems occur. The disadvantages are that nonholonomic constraints cannot be expressed in terms of reduced coordinates and it is hard to find these coordinates for systems with loops.

The goal of the methods mentioned above is to obtain accurate results. In contrast to that, Ronen Barzel et al. discuss plausible motion in [BHW96], since a high degree of accuracy is not necessarily required for most computer animations. Stephane Redon et al. propose an adaptive algorithm for the simulation of articulated bodies [RGL05]. The algorithm approximates the motion of the bodies by automatically determining a set of active joints whereas the other joints are treated as rigid. By this approximation large-scale simulations of many articulated bodies can be accelerated.

3. Simulation of an unconstrained rigid body

In the dynamic simulation a rigid body is defined by six parameters. The parameters of the translational motion are: the mass m, the position of the centre of mass $C(t)$ and the velocity $v(t)$. The rotational parameters are: the inertia tensor J in body-space coordinates, a unit quaternion $q(t)$ that describes the orientation of the body [Sho85] and the angular velocity $\omega(t)$. If the geometry of a rigid body with uniform mass distribution and its mass are known, the inertia tensor of this body can be computed with the algorithm presented by Brian Mirtich in [Mir96a]. Unit quaternions are used instead of rotation matrices to represent the rotation of all rigid bodies in the simulation because the numerical error that occurs during the simulation is smaller [Bar97].

The motion of an unconstrained rigid body has six degrees of freedom. It depends on the actual state of the body as well as on the external forces (e.g. gravity) and torques acting on the body. The dynamic simulation of a rigid body is done in discrete time steps. The parameters of the body must be known at the beginning of a time step (at time t_0) and a time step size h must be given to compute the parameters at time $t_0 + h$. Because the mass and the inertia tensor in body-space coordinates are constant over time, only four

parameters have to be computed for time $t_0 + h$. In the following let us assume that the sum of all external forces F_{ext} and the sum of all external torques τ_{ext} are constant during the time step.

The position of the centre of mass and the velocity at the end of a time step can be computed by integrating the acceleration due to the external force F_{ext}:

$$\begin{aligned} C(t_0+h) &= C(t_0) + \int_0^h v(t_0) + \frac{F_{ext}}{m} t \; dt \\ &= C(t_0) + v(t_0)h + \frac{1}{2}\frac{F_{ext}}{m}h^2, \qquad (1) \\ v(t_0+h) &= v(t_0) + \int_0^h \frac{F_{ext}}{m} \; dt = v(t_0) + \frac{F_{ext}}{m}h. \qquad (2) \end{aligned}$$

To compute the rotation at time $t_0 + h$ the following differential equation for the unit quaternion $q(t)$ has to be solved:

$$\dot{q}(t) = \frac{1}{2}\tilde{\omega}(t) \cdot q(t) \qquad (3)$$

where $\tilde{\omega}(t)$ is the quaternion $[0, \omega_x, \omega_y, \omega_z]$. The solution of the differential equation can be computed by numerical integration. In this work the fourth order Runge Kutta method is used for numerical integration. The change of the angular velocity during the time step can be computed by numerically integrating

$$\dot{\omega}(t) = J^{-1} \cdot (\tau_{ext} - (\omega(t) \times (J \cdot \omega(t)))) \qquad (4)$$

in body-space coordinates.

A simulation step of an unconstrained rigid body can be done by solving the four equations above. The next section describes how constraints between rigid bodies can be simulated.

4. Simulation of constraints

In this section the handling of joint constraints between two rigid bodies is discussed. The simulation of a joint works as follows (see figure 1). In every simulation step, the joint state is evolved forward in time to obtain the joint state after the simulation step. If this predicted joint state does not satisfy the joint constraint, the error that occurred is determined. In the case of an error, it is approximated how the velocities of the connected bodies must change to eliminate this error. An impulse is computed that causes exactly the approximated velocity change and it is applied at the beginning of the simulation step. The whole procedure is continued in an iterative process until the error vanishes and the constraint is satisfied for the predicted joint state. Then a simulation step can be done as described in section 3. Due to the changed velocities of the bodies, the joint constraint will be satisfied after the simulation step. The iterative computation of joint impulses converges towards the exact solution (the proof can be found in [SBP05]). Velocity constraints are satisfied after the simulation step by applying an impulse. Since the required velocity change is known in the case of such a constraint, the impulse can be determined at once.

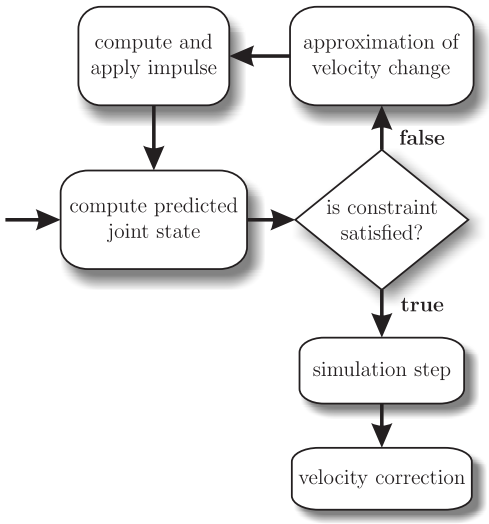

Figure 1: *Simulation step*

In this section first some definitions are introduced that are needed for the computation of constraint impulses. Then three translational and three rotational joint constraints are presented. It is shown that by combining these six constraints every kind of joint can be created. In the beginning only systems with a single constraint are regarded. After that the simulation of systems with multiple constraints is explained. Closed kinematic chains must be treated in a special way and are discussed separately.

4.1. Basics

In the following let $\tilde{J}_i$ be the inertia tensor in world space of the rigid body with the index i. The cross product matrix a^* of a vector a is defined as follows:

$$a^* = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}, \; a \times b = a^* \cdot b.$$

The velocity of a point P of the i-th rigid body is given by

$$u_P(t) := v_i(t) + \omega_i(t) \times (P(t) - C_i(t)).$$

Let $P(t)$ and $Q(t)$ be two arbitrary points of the i-th rigid body in world space and let $r_P(t) = P(t) - C_i(t)$ and $r_Q(t) = Q(t) - C_i(t)$ be the vectors from the centre of mass to these points. If an impulse p is applied at $Q(t)$, the change $\Delta u_P(t)$ of the point velocity of $P(t)$ can be computed with the following matrix $K_{P,Q}(t)$:

$$\begin{aligned} K_{P,Q}(t) &:= \begin{cases} \frac{1}{m_i} I_3 - r_P^*(t)\tilde{J}_i^{-1}(t)r_Q^*(t) & \text{if body } i \text{ is dynamic} \\ \mathbf{0} & \text{otherwise} \end{cases} \\ \Delta u_P(t) &= K_{P,Q}(t) \cdot p \end{aligned}$$

where I_3 is the 3x3 identity matrix.

The matrix $L_i(t)$ is used to determine the change of the angular velocity $\Delta\omega_i(t)$, if an angular momentum l is applied to a rigid body i:

$$L_i(t) := \begin{cases} \tilde{J}_i^{-1}(t) & \text{if body } i \text{ is dynamic} \\ \mathbf{0} & \text{otherwise} \end{cases}$$
$$\Delta\omega_i(t) = L_i(t) \cdot l.$$

Two more matrices are needed to describe the dependencies between an impulse and the angular velocity and between an angular momentum and the point velocity. Let $P(t)$ be an arbitrary point of rigid body i in world space and let $r_P(t) = P(t) - C_i(t)$ be the vector from the centre of mass to this point. Then the change of the angular velocity $\Delta\omega_i(t)$ of the body when applying an impulse p at the point $P(t)$ is:

$$W_{i,P}(t) := \begin{cases} \tilde{J}_i^{-1}(t) \cdot r_P^*(t) & \text{if body } i \text{ is dynamic} \\ \mathbf{0} & \text{otherwise} \end{cases}$$
$$\Delta\omega_i(t) = W_{i,P}(t) \cdot p.$$

The velocity of point $P(t)$ changes by $\Delta u_P(t)$ if an angular momentum l is applied to the i-th rigid body:

$$U_{P,i}(t) := \begin{cases} -r_P^*(t) \cdot \tilde{J}_i^{-1}(t) & \text{if body } i \text{ is dynamic} \\ \mathbf{0} & \text{otherwise} \end{cases}$$
$$\Delta u_P(t) = U_{P,i}(t) \cdot l.$$

4.2. Translational joint constraints

In this subsection a simulation step with a single translational joint is described. Systems with multiple joints are discussed in section 4.5. It is assumed that each joint constraint is satisfied at the beginning of a simulation step (at time t_0). Then impulses are computed in order to satisfy the constraints at the end of the step.

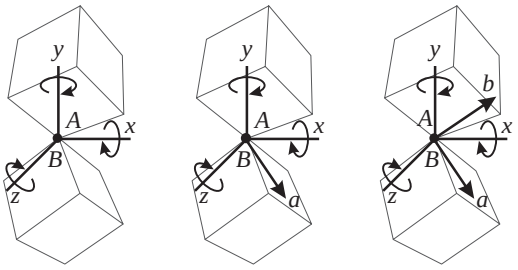

Figure 2: *Degrees of freedom of the translational joints*

A translational constraint removes only translational degrees of freedom. For example a *spherical joint* (see figure 2(a)) connects a point A of a body to a point B of another body. The effect of the joint constraint $|A-B| = 0$ is that the two bodies can only rotate around this position. This results in the removal of three translational degrees of freedom of the system. If both rigid bodies of this joint are simulated without regarding the constraint (see section 3), the joint points will drift apart during the simulation step as shown in figure 3. The goal of the presented dynamic simulation is to find an impulse p_{jc} for the time t_0 that eliminates the distance d between the two points at the end of a simulation step. The impulse has to be applied with a positive sign to the first point and with a negative sign to the second point to avoid a gain of energy in the system. The computation of this impulse is called the *joint correction*. In the following, first the computation of this joint impulse is explained for a spherical joint and then a modification of the introduced equations is presented to simulate joints which remove less translational degrees of freedom.

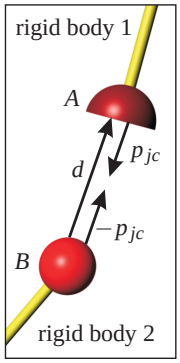

Figure 3: *Impulses for a spherical joint*

Before the joint impulse p_{jc} can be computed, the distance d between the two joint points at the end of the simulation step must be known. If the parameters of a body i and the position of a point A fixed to this body are known at time t_0, the direction of the vector $r(t_0) = A(t_0) - C_i(t_0)$ at time $t_0 + h$ can be determined by solving the following differential equation with the forth order Runge Kutta method:

$$\dot{r}(t) = \omega_i(t) \times r(t). \tag{5}$$

If the position of the centre of mass at the end of the simulation step is determined by using equation 1, the new position of the point A can be computed by $A(t_0 + h) = r(t_0 + h) + C(t_0 + h)$. Since the new positions of the two joint points can be determined, the distance between them at the end of the simulation step is

$$d(t_0 + h) = A(t_0 + h) - B(t_0 + h).$$

Now an impulse p_{jc} must be computed to eliminate the distance between the two points within one time step of size h. Since the points have typically a nonlinear motion, the required impulse can be determined by solving a nonlinear equation iteratively [WTF06]. In this work a simplification was used. If the relative velocity of the two points is changed

by $d(t_0+h)/h$ as if the relative motion of the points is linear, the resulting impulse will reduce the distance $d(t_0+h)$, but in general it will not eliminate it completely. The computation of such impulses is continued iteratively until the distance $d(t_0+h)$ vanishes within a tolerance. In practice the time step size h is normally at most 0.04 s (25 frames per second). In several tests with $h = 0.04$ s the desired impulse was computed. In most cases one or two iteration steps were needed even with a small tolerance of 10^{-6} m. More iterations were only needed, if the connected bodies had very high velocities. The advantage of the introduced simplification is that the equation for the required impulse is linear and can be solved easily.

The impulse that changes the relative velocity of the two points by $d(t_0+h)/h$ must be applied with a positive sign to point A and with a negative sign to point B. This impulse can be computed by solving the following equation:

$$K_{A,A}(t_0)\cdot p_{jc} - K_{B,B}(t_0)\cdot(-p_{jc}) = \frac{1}{h}d(t_0+h).$$

The matrix $K(t_0) := K_{A,A}(t_0)+K_{B,B}(t_0)$ is constant at time t_0, nonsingular, symmetric and positive definite (the proof can be found in [Mir96b]). Hence the equation can be solved by inverting the matrix $K(t_0)$:

$$p_{jc} = \frac{1}{h}K(t_0)^{-1}d(t_0+h). \tag{6}$$

Impulses are computed iteratively with equation 6 until the distance vanishes. In each iteration the distance $d(t_0+h)$ must be updated. The impulses are applied at the beginning of the time step (at time t_0). When the iteration ends, a simulation step for unconstrained motion (see section 3) can be done and the joint constraint will be satisfied at time t_0+h due to the changed velocities.

After the simulation step it is not guaranteed that the velocities of the two joint points are equal. A velocity difference $\Delta u(t_0+h) = u_B(t_0+h) - u_A(t_0+h)$ between the two joint points can be corrected by applying the following impulse at time t_0+h:

$$p_{vc} = K^{-1}(t_0+h)\cdot\Delta u(t_0+h). \tag{7}$$

The velocity change caused by this impulse eliminates the difference $\Delta u(t_0+h)$ immediately. This is called the *velocity correction*. The computation of an impulse in order to correct the velocities is not absolutely necessary for the dynamic simulation because the joint impulse of the next simulation step will solve the problem as well, but a higher degree of accuracy can be achieved. All translational degrees of freedom are removed between the connected bodies by the impulses computed above and in this way a spherical joint can be simulated.

The next kind of joint removes only two translational degrees of freedom (see figure 2(b)). This joint is defined by a line $A+\lambda a$ which is fixed to the first body and a point B which is fixed to the second body. The point of the second body can move freely on the line of the first body. Each time before an impulse is computed the point A is moved on the line to the position where it has the smallest distance to the point B. This is necessary to obtain a physical correct result. The impulses for this joint can be computed by projecting equation 6 and 7 in the two-dimensional space. Let b and c be two linearly independent vectors perpendicular to the given vector a. These vectors correspond to the two degrees of freedom that should be removed. The two-dimensional joint impulse is then computed by solving the following equation in an iterative process until the distance $Pd(t_0+h)$ vanishes:

$$PK(t_0)P^T\cdot p'_{jc} = \frac{1}{h}\cdot Pd(t_0+h) \tag{8}$$

where $P = \begin{pmatrix} b^T \\ c^T \end{pmatrix} \in \mathbb{R}^{2\times 3}$ is the projection matrix. The three-dimensional joint impulse p_{jc} is given by

$$p_{jc} = P^T\cdot p'_{jc}.$$

Equation 7 is also projected onto the plane spanned by b and c to compute the two-dimensional impulse for the velocity correction at once:

$$PK(t_0+h)P^T\cdot p'_{vc} = P\cdot\Delta u(t_0+h). \tag{9}$$

This impulse must be transformed in world space before applying it to the bodies.

The last translational joint removes one degree of freedom (see figure 2(c)). The joint is defined by a plane $A+\lambda a+\mu b$ that is fixed to the first rigid body and a point B that is fixed to the second body. The joint allows B to move freely in the plane of the first body. In order to obtain a physical correct result the point A is moved in the plane to the position where it has the smallest distance to the point B before an impulse is computed. The normal vector of the plane is given by $c = a\times b$. This joint can be realised by using the projection matrix $P = \begin{pmatrix} c^T \end{pmatrix} \in \mathbb{R}^{1\times 3}$ in equations 8 and 9 and solving them as described for the last joint.

The impulses for a spherical joint can also be computed by solving equations 8 and 9 if the projection matrix P is the identity matrix. In conclusion all translational joint constraints can be satisfied by computing impulses with these two equations.

4.3. Rotational joint constraints

In this subsection three rotational joints are discussed which remove one, two or three rotational degrees of freedom. In every simulation step an angular momentum l_{jc} is determined to satisfy the joint constraint at the end of the time step and another angular momentum l_{vc} is computed to correct the difference between the angular velocities of the connected bodies at the end of the simulation step.

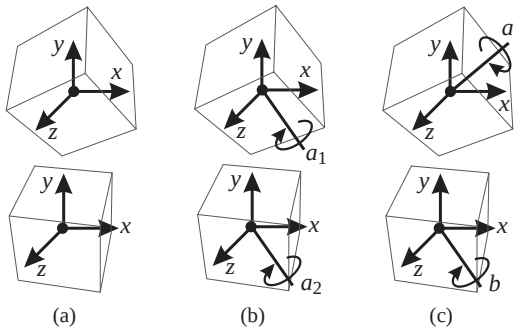

Figure 4: *Degrees of freedom of the rotational joints*

The first rotational joint that is discussed removes all rotational degrees of freedom between two rigid bodies (see figure 4(a)). This means that the bodies can move freely in all directions but are not allowed to rotate relative to each other. Before an angular momentum l_{jc} for the joint correction can be computed, the error that occurs, if both bodies are simulated without any constraint, must be determined. If $q_1(0)$ and $q_2(0)$ are the rotation quaternions of two rigid bodies at the beginning of the simulation, then the change of the relative rotation at time t_0+h is described by the following quaternion:

$$\Delta q(t_0+h) = (q_2(0)^{-1} \cdot q_2(t_0+h))^{-1} \cdot (q_1(0)^{-1} \cdot q_1(t_0+h)).$$

The rotation quaternions of the bodies at time t_0+h can be determined by solving the differential equation 3. The quaternion $\Delta q(t_0+h)$ is converted to a rotation axis $a(t_0+h)$ and an angle $\alpha(t_0+h)$. An angular momentum l_{jc} must be computed that eliminates the rotation $d(t_0+h) = \alpha(t_0+h)a(t_0+h)$ within one time step of size h. The same simplification is used as the one introduced for the translational joints. An angular momentum is determined that changes the relative angular velocity of the connected bodies by $d(t_0+h)/h$ as if the relative rotatory motion of the bodies is linear. The required angular momentum l_{jc} is obtained by computing such angular momenta in an iterative process until $d(t_0+h)$ vanishes within a tolerance. The following equation must be solved to determine the angular momentum that changes the relative angular velocity of the bodies by $d(t_0+h)/h$ within one time step of size h:

$$(L_1(t_0)+L_2(t_0)) \cdot l_{jc} = \frac{1}{h} d(t_0+h). \tag{10}$$

Since the inertia tensor of a rigid body and its inverse are symmetric and positive definite [Mir96b], $L(t) = L_1(t) + L_2(t)$ is also symmetric and positive definite, if at least one of the two bodies is dynamic. This implies that $L(t)$ is nonsingular and the equation can be solved by inverting the matrix $L(t_0)$.

At the end of the time step the difference between the angular velocities $\Delta\omega(t_0+h) = \omega_2(t_0+h) - \omega_1(t_0+h)$ must be eliminated by applying an angular momentum l_{vc}. This is determined at once by solving the equation:

$$(L_1(t_0+h)+L_2(t_0+h)) \cdot l_{vc} = \Delta\omega(t_0+h). \tag{11}$$

The next joint removes two rotational degrees of freedom of the two connected bodies. This means that both bodies are allowed to rotate around one common rotation axis and to move freely in all directions. Let a_1 and a_2 be this axis with unit length fixed to the first and the second body respectively (see figure 4(b)). The joint constraint forces both axes to have the same orientation. Because of this, the joint is called *orientation joint*. At time t_0 the constraint is satisfied, so $a_1(t_0) = a_2(t_0)$. Let b and c be two linearly independent vectors perpendicular to the axis $a_1(t_0)$. The error that occurs during the simulation step can be described by the cross product of the two rotation axes at time t_0+h:

$$d(t_0+h) = a_1(t_0+h) \times a_2(t_0+h).$$

The two-dimensional angular momentum l'_{jc} maintaining the joint constraint for a time step of size h is computed by solving the following equation in an iterative process until $Pd(t_0+h)$ is zero:

$$P(L_1(t_0)+L_2(t_0))P^T l'_{jc} = \frac{1}{h} Pd(t_0+h) \tag{12}$$

where $P = \begin{pmatrix} b^T \\ c^T \end{pmatrix} \in \mathbb{R}^{2\times 3}$ is the projection matrix. The three-dimensional angular momentum for the joint correction is $l_{jc} = P^T \cdot l'_{jc}$. The angular momentum l'_{vc} needed for the velocity correction is determined by projecting equation 11 onto the plane spanned by b and c:

$$P(L_1(t_0+h)+L_2(t_0+h))P^T l'_{vc} = P\Delta\omega(t_0+h). \tag{13}$$

The last rotational joint allows the connected bodies to move in all directions and to rotate around two linearly independent axes a and b. Axis a is fixed to the first body and axis b to the second body (see figure 4(c)). In the following it is assumed that both axes are normalised. The joint constraint prevents the bodies from rotating around the axis $c = a \times b$. This means that the angle $\varphi(t) = \arccos(a(t) \cdot b(t))$ must be constant during the simulation. The three-dimensional error that occurs during one simulation step is given by

$$d(t_0+h) = (\varphi(t_0+h) - \varphi(0)) \cdot c.$$

The angular momentum to correct this error can be determined by computing angular momenta with equation 12 in an iterative process where $P = \begin{pmatrix} c^T \end{pmatrix} \in \mathbb{R}^{1\times 3}$ is the projection matrix. The same projection matrix is used in order to correct the angular velocities with equation 13.

All rotational joints can be simulated by computing angular momenta with equation 12 for the joint correction and with equation 13 for the velocity correction, if the identity matrix is used as projection matrix P for the first joint.

4.4. Combinations of joint constraints

In the preceding subsections six different joint constraints have been introduced. By combining two of these constraints new joint types can be created. A hinge joint can be simulated by combining a spherical and an orientation joint. The spherical joint eliminates the translational degrees of freedom between the bodies and the orientation joint allows the connected bodies to rotate around a common axis. If a torque is applied to the bodies in direction of the rotation axis, a motor can be simulated. A PID controller can be used to control this motor. Different kinds of sliders can be simulated by combining a joint which removes two translational degrees of freedom with different rotational joints. It is possible to remove every combination of translational and rotational degrees of freedom using the introduced constraints. Combined joints that have been implemented are: hinge joints, fixed joints, universal joints and all kinds of sliders.

4.5. Systems of joint constraints

If a system of rigid bodies connected with multiple joints is given, it is not possible to satisfy all constraints by simply computing an impulse for every joint. In figure 5 a double pendulum is shown. The constraint of the left joint is satisfied but not the right one. If an impulse is computed and applied to correct the right joint, the left joint will break up. This problem can be solved by computing impulses for each

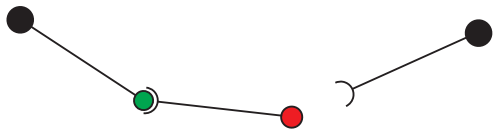

Figure 5: *System with two spherical joints*

joint in an iterative loop as Weinstein et al. and Bender et al. do [WTF06, BFS05]. The iteration stops if all joint constraints are satisfied. This iterative method is very robust and even closed kinematic chains can be simulated without any additional effort. Another advantage of the method is that it does not have drift problems like the Lagrange multiplier method because the computed impulses correct every drift that occurs. The only problem is that the iterative method does not regard the dependencies between the joints in a multi-body system. Because of this, the method needs many iterations to simulate systems which have a complex joint structure, especially if a high degree of accuracy is required. Therefore the simulation of such systems is slow.

If a rigid body is connected with multiple joints to other bodies, the impulses needed to maintain the constraints depend on each other. These dependencies can be described in a system of linear equations:

$$\mathbf{M} \cdot \mathbf{p}' = \Delta\mathbf{u}. \tag{14}$$

The dimension of this system is equal to the amount of degrees of freedom that are removed in the multi-body system. The matrix of the system of linear equations $\mathbf{M}$ is a block matrix. The diagonal block $M_{i,i}$ is the matrix that is necessary to compute the joint impulse of the i-th joint. This means that $M_{i,i}$ is the matrix of equation 8 if i is a translational joint and the one of equation 12 if i is a rotational joint. The off-diagonal block $M_{i,j}$ describes the dependency between joint i and joint j. The two joints depend on each other, if they are connected to the same rigid body. Otherwise $M_{i,j}$ is a zero matrix. If there is a dependency, the matrix $M_{i,j}$ must describe how the velocities at joint i change, if at joint j an impulse or an angular momentum is applied. This can be done by using the matrices defined in subsection 4.1.

In the following we assume that we have a system with n joints and that there are no loops in this system. Two disjoint index sets T and R are used. Let $T = \{1,\dots,m\}$ be the indices of all translational joints and $R = \{m+1,\dots,n\}$ be the indices of all rotational joints. The block matrix $M_{i,j}$ depends on the joint types of i and j. If the two joints have a common body k, then the following matrix is used to differentiate between the four possible cases:

$$N_{i,j}(k,X,Y) = \begin{cases} K_{X,Y} & \text{if } i,j \in T \\ L_k & \text{if } i,j \in R \\ W_{k,Y} & \text{if } i \in R \text{ and } j \in T \\ U_{X,k} & \text{if } i \in T \text{ and } j \in R \end{cases}$$

where X and Y are the joint points fixed to body k of joint i and joint j respectively. A joint point is only used in the matrix, if the corresponding joint is a translational one.

With the matrix $N_{i,j}(k,X,Y)$ the block matrices $\tilde{M}_{i,j}$ in three-dimensional space can be computed. The matrices for constraints with a lower dimension can be determined by a projection of matrix $\tilde{M}_{i,j}$ which is explained later. The matrix $\tilde{M}_{i,j}$ describes the number of common bodies the two joints have and how they are connected to the bodies. Only the translational joints have joint points but in the following we will assume that each rotational joint also has two joint points. The position of these points is arbitrary because they are only needed for a simpler notation. Let A_i and B_i be the joint points of the i-th joint and k_{i_1} and k_{i_2} be the connected rigid bodies. The matrix $\tilde{M}_{i,j}$ is defined as

$$\tilde{M}_{i,j} = \begin{cases} N_{i,j}(k_{i_1},A_i,A_j) & \text{if } k_{i_1} = k_{j_1} \wedge k_{i_2} \neq k_{j_2} \\ N_{i,j}(k_{i_2},B_i,B_j) & \text{if } k_{i_2} = k_{j_2} \wedge k_{i_1} \neq k_{j_1} \\ (N_{i,j}(k_{i_1},A_i,A_j) + & \\ \quad N_{i,j}(k_{i_2},B_i,B_j)) & \text{if } k_{i_1} = k_{j_1} \wedge k_{i_2} = k_{j_2} \\ -N_{i,j}(k_{i_1},A_i,B_j) & \text{if } k_{i_1} = k_{j_2} \wedge k_{i_2} \neq k_{j_1} \\ -N_{i,j}(k_{i_2},B_i,A_j) & \text{if } k_{i_2} = k_{j_1} \wedge k_{i_1} \neq k_{j_2} \\ -(N_{i,j}(k_{i_1},A_i,B_j) + & \\ \quad N_{i,j}(k_{i_2},B_i,A_j)) & \text{if } k_{i_1} = k_{j_2} \wedge k_{i_2} = k_{j_1} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

The last step to build the matrix $\mathbf{M}$ is to project the matrices $\tilde{M}_{i,j}$ using the projection matrices of the corresponding joints. The resulting system of linear equations is

$$\begin{pmatrix} P_1\tilde{M}_{1,1}P_1^T & \dots & P_1\tilde{M}_{1,m}P_m^T \\ \vdots & \ddots & \vdots \\ P_m\tilde{M}_{m,1}P_1^T & \dots & P_m\tilde{M}_{m,m}P_m^T \end{pmatrix} \begin{pmatrix} p_1' \\ \vdots \\ p_m' \end{pmatrix} = \begin{pmatrix} P_1\Delta u_1 \\ \vdots \\ P_m\Delta u_m \end{pmatrix}$$

where p_i' is the projected impulse or angular momentum of the i-th joint and Δu_i is the velocity difference that has to be corrected. In the case of joint correction, the value $\frac{1}{h}d(t_0+h)$ is used as velocity difference where $d(t_0+h)$ is the distance of the corresponding joint as it was defined in sections 4.2 and 4.3. By solving the system of linear equations all impulses and angular momenta are determined at once. It has to be kept in mind that the velocity difference $\frac{1}{h}d(t_0+h)$ is just an approximation. Hence, after the impulses and angular momenta computed with the system of linear equations are applied to the rigid bodies, for every joint a new prediction of the distance $d(t_0+h)$ has to be made in order to verify, if all joint constraints will be satisfied after a simulation step. As long as there exist joints whose constraints are not satisfied, the system of linear equations has to be solved for the actual prediction of the distances $d(t_0+h)$ in an iterative process. Since all dependencies of the joints are taken into account during the computation of the impulses and angular momenta, only a few iterations are needed even for complex multi-body systems. No approximation is made for the velocity correction and so the exact solution can be determined in one step.

The system of linear equations can be solved, for example by using a LU decomposition of the matrix $\mathbf{M}$. Because the matrix is constant at a time t, the decomposition can be used for the velocity correction and the joint correction of the next step (since both are computed for the same simulation time). This means that the decomposition must be computed only once per simulation step. Hence the method does not slow down much even if several iterations are necessary. Since the matrix is typically sparse, a sparse solver is a better choice than using a LU decomposition. Therefore in this work the solver PARDISO was used [SG02, SG04].

4.6. Closed kinematic chains

The system of linear equations of a model which contains closed kinematic chains can have a higher dimension than the amount of degrees of freedom that are removed [Wit77]. In this case solving the system of linear equations can lead to unstable results. Multi-body systems with closed kinematic chains can be simulated by breaking the loops in the model. An undirected graph is used to find these loops (see figure 6). Every rigid body in the model is represented by a node in this graph. Two nodes are connected by an edge, if there exists a joint between the corresponding rigid bodies. Loops in the model can be detected by finding cycles in the graph. If a cycle is found the corresponding joint is marked and the edge is removed. This is continued until there are no more cycles in the graph. The impulses for the multi-body system without the marked joints and the impulses for the marked joints are computed by using two separate systems of linear equations. Both systems are solved in a common iterative loop. The loop ends, if the constraints of both parts are satisfied.

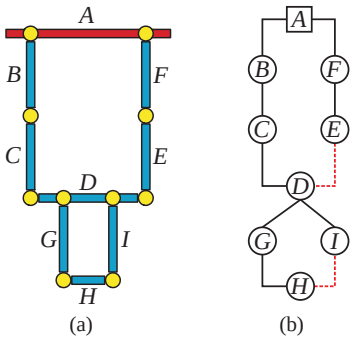

Figure 6: *Multi-body system with loops*

5. Results

All simulations in this section have been performed on a PC with a 3.4 GHz Intel Pentium 4 processor. All differential equations have been solved with the fourth-order Runge-Kutta method. At first a tree with 127 rigid bodies that are

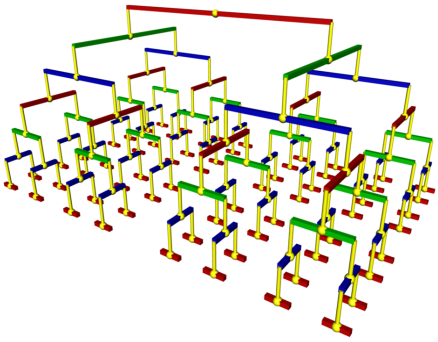

Figure 7: *A tree with 127 spherical joints*

connected by 127 spherical joints (see figure 7) was simulated without collisions and contacts. This means that 381 degrees of freedom were removed by the constraints. The bodies were between 1 m and 11.4 m long and had equal densities. In order to obtain accurate results, the tolerances $\varepsilon_{jc} = 10^{-6}\,\text{m}$ and $\varepsilon_{vc} = 10^{-6}\,\frac{\text{m}}{\text{s}}$ were used for the joint and the velocity correction respectively. Since the presented method with systems of linear equations (SLE) computes the exact impulses for the velocity correction in one step, the tolerance ε_{vc} was only required for the iterative method

without using systems of linear equations which was mentioned in the beginning of section 4.5. A time step size of $h = \frac{1}{30}$ s was used in order to produce 30 frames per second. At the start of the simulation a torque acts on the top body of the model causing a rotation of the tree. The computation times of the simulation steps with both methods were measured and the results are shown in figure 8. Table 1 shows

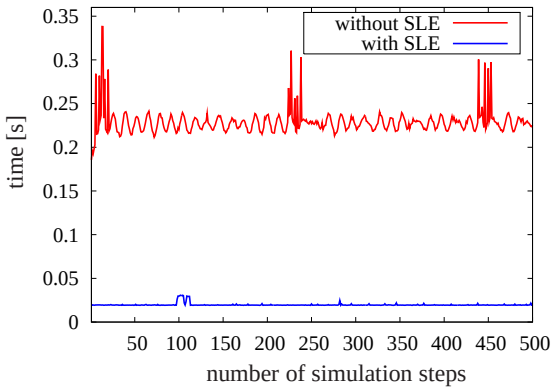

Figure 8: *Computation times*

the average values of both methods. The simulation with the second method runs faster than real-time and is more than eleven times faster than the first method. This speed up is even higher, if a model with more dependencies between the joints is simulated as shown later. The first method needed

	without SLE	with SLE
average time per step	228.57 ms	20.09 ms
average iteration steps (jc)	518.15	2
average iteration steps (vc)	745.54	1

Table 1: *Average values*

more than 518 iterations for the joint correction and 745 for the velocity correction. The second method needed only 2 iterations for the joint correction but the computation of the impulses in one iteration step needed more time because a system of linear equations had to be solved.

Trees of different sizes have been simulated to show the scalability of the methods. Furthermore different values for the tolerances ε_{jc} and ε_{vc} have been used. The average time needed for one simulation step has been measured for the method without using systems of linear equations and for the new method. The results of the first method are shown in table 2. Equal tolerance values have been used for ε_{jc} and ε_{vc}. The times needed by the new method are shown in table 3. In the simulation with large tolerance values the use of systems of equations has no advantage, since the computation of single impulses runs very fast and not many iterations are needed. If smaller tolerances were used, the use of a SLE

	10^{-2}	10^{-4}	10^{-6}
Tree 31	0.97 ms	6.78 ms	14.31 ms
Tree 63	2.48 ms	28.46 ms	67.50 ms
Tree 127	6.25 ms	87.05 ms	228.57 ms
Tree 255	15.86 ms	246.71 ms	703.48 ms

Table 2: *Simulation times without SLE*

	10^{-2}	10^{-4}	10^{-6}
Tree 31	3.23 ms	3.54 ms	3.79 ms
Tree 63	7.12 ms	7.14 ms	7.69 ms
Tree 127	18.78 ms	18.95 ms	20.09 ms
Tree 255	62.06 ms	63.21 ms	63.61 ms

Table 3: *Simulation times with SLE*

accelerated the simulation. The average simulation times of the first method strongly depend on the tolerance values that are used. The times of the second method seem to be independent from the tolerances. The reason for this is that the degree of accuracy of the results is always high, even if this is not demanded.

The use of the new method has advantages if a rigid body is connected with many joints. To show this, the tree with 127 joints has been simulated again. This time a body in the tree was not connected to its direct parent but to the root of the tree. So the model had 127 joints and all joints were connected to the root body. The tolerances $\varepsilon_{jc} = 10^{-4}$ m and $\varepsilon_{vc} = 10^{-4} \frac{\text{m}}{\text{s}}$ have been used for this simulation. The simulation with the new method was almost 80 times faster than without using a SLE.

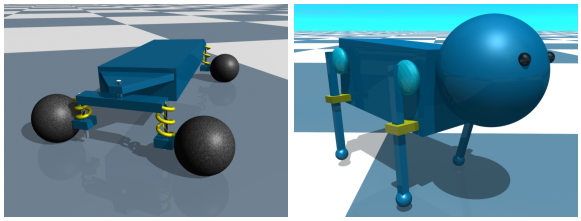

Figure 9: *A car and a walking machine*

At last the practical use of different joint types is shown. Therefore a car and a walking machine were build (see figure 9). The car has a servo motor for each wheel and one servo motor for the steering. A slider joint has been combined with a spring to simulate the dampers for the wheels. The simulation of the joints of this car ran nearly six times faster than real-time. Each leg of the walking machine is simulated by one servo motor, two hinge joints and one slider. This model has one closed kinematic chain per leg.

6. Conclusion

An extension of the method of Bender et al. has been presented that allows to simulate several different kinds of joints in an uniform way. For the fast simulation of complex models a new method has been introduced that uses a system of linear equations to describe the dependencies in a multibody system. The advantages of the method are that it is easy to implement, it has no drift problem, accurate results can be achieved and it is fast.

References

[ACR95] ASCHER U. M., CHIN H. S., REICH S.: Stabilization of constrained mechanical systems with daes and invariant manifolds. *J. Mech. Struct. Machines 23* (1995), 135–158.

[Asc97] ASCHER U. M.: Stabilization of invariants of discretized differential systems. *Numerical Algorithms 14*, 1–3 (1997), 1–24.

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 137–146.

[Bar97] BARAFF D.: An introduction to physically based modeling: Rigid body simulation 1 - unconstrained rigid body dynamics. SIGGRAPH Course Notes, 1997.

[Bau72] BAUMGARTE J.: Stabilization of constraints and integrals of motion in dynamical systems. In *Computer Methods in Applied Mechanics* (1972), pp. 1–16.

[BFS05] BENDER J., FINKENZELLER D., SCHMITT A.: An impulse-based dynamic simulation system for VR applications. In *Proceedings of Virtual Concept 2005* (Biarritz, France, 2005), Springer.

[BHW96] BARZEL R., HUGHES J. F., WOOD D. N.: Plausible motion simulation for computer graphics animation. In *Computer Animation and Simulation '96* (1996), pp. 183–197.

[BS06] BENDER J., SCHMITT A.: Constraint-based collision and contact handling using impulses. In *Proceedings of the 19th international conference on computer animation and social agents* (Geneva (Switzerland), July 2006), pp. 3–11.

[Chi95] CHIN H. S.: *Stabilization methods for simulations of constrained multibody dynamics*. PhD thesis, Inst. of App. Math., University of British Columbia, 1995.

[dJB94] DE JALON J. G., BAYO E.: *Kinematic and Dynamic Simulation of Multibody Systems : the Real Time Challenge*. Springer-Verlag, New York, 1994.

[Fea87] FEATHERSTONE R.: *Robot dynamics algorithms*. Kluwer, 1987.

[GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Nonconvex rigid bodies with stacking. *ACM Transactions on Graphics 22*, 3 (July 2003), 871–878.

[Mir96a] MIRTICH B.: Fast and accurate computation of polyhedral mass properties. *J. Graph. Tools 1*, 2 (1996), 31–50.

[Mir96b] MIRTICH B. V.: *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California, Berkeley, 1996.

[RGL05] REDON S., GALOPPO N., LIN M. C.: Adaptive dynamics of articulated bodies. *ACM Trans. Graph. 24*, 3 (2005), 936–945.

[SB05] SCHMITT A., BENDER J.: Impulse-based dynamic simulation of multibody systems: Numerical comparison with standard methods. In *Proc. Automation of Discrete Production Engineering* (2005), pp. 324–329.

[SBP05] SCHMITT A., BENDER J., PRAUTZSCH H.: *On the Convergence and Correctness of Impulse-Based Dynamic Simulation*. Internal Report 17, Institut für Betriebs- und Dialogsysteme, 2005.

[SG02] SCHENK O., GÄRTNER K.: Two-level dynamic scheduling in pardiso: improved scalability on shared memory multiprocessing systems. *Parallel Comput. 28*, 2 (2002), 187–197.

[SG04] SCHENK O., GÄRTNER K.: Solving unsymmetric sparse systems of linear equations with pardiso. *Future Gener. Comput. Syst. 20*, 3 (2004), 475–487.

[Sho85] SHOEMAKE K.: Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM Press, pp. 245–254.

[SNTH03] SHAO W., NG-THOW-HING V.: A general joint component framework for realistic articulation in human characters. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics* (New York, NY, USA, 2003), ACM Press, pp. 11–18.

[WG01] WILHELMS J., GELDER A. V.: Fast and easy reach-cone joint limits. *J. Graph. Tools 6*, 2 (2001), 27–41.

[Wit77] WITTENBURG J.: *Dynamics of systems of rigid bodies*. Teubner, 1977.

[WTF06] WEINSTEIN R., TERAN J., FEDKIW R.: Dynamic simulation of articulated rigid bodies with contact and collision. In *IEEE Transactions on Visualization and Computer Graphics* (2006), vol. 12, pp. 365–374.