Impulse-based dynamic simulation in linear time

Jan Bender
Institut für Betriebs- und Dialogsysteme
Universität Karlsruhe
http://www.impulse-based.de
jbender@ira.uka.de

ABSTRACT

This paper describes an impulse-based dynamic simulation method for articulated bodies which has a linear time complexity. Existing linear-time methods are either based on a reduced-coordinate formulation or on Lagrange multipliers. The impulse-based simulation has advantages over these well-known methods. Unlike reduced-coordinate methods, it handles nonholonomic constraints like velocity-dependent ones and is very easy to implement. In contrast to Lagrange multiplier methods the impulse-based approach has no drift problem and an additional stabilisation is not necessary. The presented method computes a simulation step in $O(n)$ time for acyclic multi-body systems containing equality constraints. Closed kinematic chains can be handled by dividing the model into different acyclic parts. Each of these parts is solved independently from each other. The dependencies between the single parts are solved by an iterative method. In the same way inequality constraints can be integrated in the simulation process in order to handle collisions and permanent contacts with dynamic and static friction.

Keywords

linear-time dynamics, dynamic simulation, physically-based animation, rigid bodies, articulated bodies

1. INTRODUCTION

The dynamic simulation of multi-body systems is an important topic in computer graphics. The task of a simulation method is to compute the motion of multiple rigid bodies in consideration of forces acting on these bodies. In general, there is a distinction between internal and external forces. Internal forces act between two bodies and do not affect the state of motion of the multi-body system. This means that the sum of all internal forces is zero. External forces are forces like gravity that influence the motion state of the system.

Internal forces are needed to satisfy constraints of the multi-body system. Constraints can be used to model joints, collisions and permanent contacts. They can also be used to interact with the dynamic model during the simulation. Dynamic Models often have a sparse system of constraints. In many cases the system of constraints is even acyclic, e.g. the model of a robot usually has no loop. Especially for acyclic models, there exist linear-time methods for dynamic simulation. These methods are either based on a reduced-coordinate formulation or they use Lagrange multipliers.

In this paper, an impulse-based simulation method is presented that has also a linear time complexity. The impulse-based approach has several advantages: it can handle holonomic and nonholonomic constraints, unlike the Lagrange multiplier methods it needs no additional stabilisation, it is easy to implement and it is very fast. Impulses are computed instead of internal forces in order to satisfy the constraints. There exist iterative methods for the computation of the impulses. It is also possible to describe the dependencies between the constraints in a system of linear equations (SLE) which leads to a faster computation, especially if accurate results are demanded. But in the worst case the time complexity of this SLE method is $O(n^3)$. This paper presents a method to determine the needed impulses in $O(n)$ time. It is also shown how models with loops, collisions and contacts are handled. Finally a comparison of the linear-time method using Lagrange multipliers and the presented method is performed.

2. PREVIOUS WORK

The simulation of joint constraints is a well-known task. There exist different approaches to compute the internal forces that act between the articulated bodies. A very simple approach is the penalty method [7]. This method adds a force to a multi-body system, if a constraint is not satisfied. The magnitude and direction of this force depends on the constraint violation. The penalty method has the advantage that it is very easy to implement but it is not suitable for accurate simulations.

A reduced-coordinate formulation provides a more accurate simulation. Holonomic constraints reduce the degrees of freedom of a multi-body system permanently. This property is used by reduced-coordinate methods. For a multi-body system a parametrisation is required to reduce the number of coordinates that describe the system's state to a minimum. For each degree of freedom one coordinate is needed. The equations of motion can be expressed via reduced coor-

dinates by using a Lagrangian formulation of the problem. The algorithms based on this formulation have a time complexity of $O(n^4)$. The first linear-time algorithms were based on a Newton-Euler formulation [8, 9].

The Lagrange multiplier approach has several advantages compared with the reduced-coordinate formulation [2]. The use of Lagrange multipliers permits the simulation of strongly modular systems. A model can be extended during the simulation without the need of a new parametrisation. In contrast to the reduced-coordinate formulation nonholonomic constraints can be handled by using Lagrange multipliers. David Baraff described a linear-time method for acyclic models that is based on Lagrange multipliers. So this method has the same time complexity as the reduced-coordinate method of Featherstone [8]. The disadvantage of the Lagrange multiplier approach is that it has a drift problem due to numerical errors. Since position and velocity constraints are defined in terms of accelerations, the numerical errors that occur during the simulation can not be corrected by this approach. An additional stabilisation method is required to solve this problem. The popular stabilisation method of Baumgarte [3] adds extra terms to the constraints in order to avoid drifting. An alternative solution is to apply additional forces [15]. In [1], Ascher gives a survey about other stabilisation techniques.

The impulse-based simulation methods use a different approach than the Lagrange multiplier methods. Position and velocity constraints are handled directly without an acceleration-based formulation [4]. Hence the impulse-based approach has no drift problem and needs no additional stabilisation. Velocity constraints can easily be simulated using impulses, since an impulse causes an instantaneous velocity change. To simulate position constraints a prediction of the actual joint state is made. Weinstein et al. use this prediction to determine the required impulse by solving a nonlinear equation [14]. The dependencies between different joints are solved by an iterative method. Bender et al. use an approximation of the required velocity change in order to linearise the equation [6]. This has several advantages. The resulting linear equation for an impulse can be solved very fast. Dependencies can be resolved by an iterative method [4] which converges to the physical correct solution [12]. The iterative method handles joint constraints, models with loops, collisions and contacts with friction [5]. The iterative process can even be interrupted at any time to get a preliminary result. But the disadvantage of this iterative approach is that if accurate results are demanded, the method needs many iterations, especially for complex models. Because of this, another impulse-based method has been developed for the accurate simulation of complex models [6]. An advantage of the used linearisation is that the dependencies between the joints can be described by a system of linear equations. By solving this SLE the required impulses can be determined simultaneously. Thus even complex models can be simulated in real-time.

3. IMPULSE-BASED DYNAMIC SIMULATION

The state of a rigid body is defined by the position of its centre of mass, its orientation as well as its linear and angular velocity. The orientation of a body is described by a unit quaternion $\mathbf{q}$ [13]. For a simulation step, the actual state of all bodies must be known. The forward simulation of an unconstrained body with mass m is done by integration of the four state parameters. It is assumed that the sum of all external forces $\mathbf{F}_{\text{ext}}$ is constant during the simulation step. In this case the linear velocity $\mathbf{v}$ and the centre of mass $\mathbf{c}$ can be integrated directly:

$$\mathbf{v}(t_0 + h) = \mathbf{v}(t_0) + \frac{\mathbf{F}_{\text{ext}}}{m} h \tag{1}$$

$$\mathbf{c}(t_0 + h) = \mathbf{c}(t_0) + \mathbf{v}(t_0)h + \frac{\mathbf{F}_{\text{ext}}}{2m} h^2 \tag{2}$$

where h is the time step size. It is also assumed that the sum of all external torques $\boldsymbol{\tau}_{\text{ext}}$ is constant during the simulation step. But even if there is no torque acting on a body, in general its angular velocity is not constant. Because of this, numerical integration is required to solve the differential equations of the angular velocity $\boldsymbol{\omega}$ and the quaternion $\mathbf{q}$ of a body with inertia tensor $\mathbf{J}$:

$$\dot{\boldsymbol{\omega}}(t) = \mathbf{J}^{-1}(\boldsymbol{\tau}_{\text{ext}} - (\boldsymbol{\omega}(t) \times (\mathbf{J} \cdot \boldsymbol{\omega}(t)))) \tag{3}$$

$$\dot{\mathbf{q}}(t) = \frac{1}{2}\tilde{\boldsymbol{\omega}}(t) \cdot \mathbf{q}(t) \tag{4}$$

where $\tilde{\boldsymbol{\omega}}(t)$ is the quaternion $[0, \omega_x, \omega_y, \omega_z]$. By directly integrating the translational parameters and numerically integrating the rotational parameters a simulation step for an unconstrained rigid body is done.

An unconstrained body has six degrees of freedom. A joint constraint reduces the degrees of freedom of a multi-body system permanently. If constraints are defined for the bodies in the system, a simulation method must compute the corresponding internal forces that prevent the joints from breaking. The impulse-based simulation method computes impulses instead of internal forces in order to satisfy given joint constraints. There exist six basic joint types that can be combined in order to simulate any existing mechanical joint [6]. Three of them just eliminate translational degrees of freedom and the other three just rotational ones. In the following only these basic joints are regarded.

Each basic joint defines a position constraint and a velocity constraint for two bodies. The position constraint is the one that reduces the degrees of freedom whereas the velocity constraint is only needed to achieve a higher degree of accuracy [6]. A velocity constraint is used to guarantee that the relative velocity of the bodies in direction of the constraint is zero.

The main idea of the impulse-based simulation method is to use a prediction of the joint state for the computation of the impulses that are needed to satisfy the position constraints [4]. This process of computing impulses for the position constraints is called *position correction*. The state of a joint that eliminates translational degrees of freedom is defined by two joint points $\mathbf{a}$ and $\mathbf{b}$ (one in each body). A prediction of the positions of these points can be computed by solving equation 2 and numerically integrating the differential equation

$$\dot{\mathbf{r}}(t) = \boldsymbol{\omega}(t) \times \mathbf{r}(t)$$

for their corresponding position vectors $\mathbf{r}_a$ and $\mathbf{r}_b$. The new position of point $\mathbf{a}$ after a simulation step of size h is given

by $\mathbf{a}(t_0+h) = \mathbf{r}_a(t_0+h)+\mathbf{c}_a(t_0+h)$. In one step the two joint points diverge by $\mathbf{d}(t_0+h) = \mathbf{b}(t_0+h) - \mathbf{a}(t_0+h)$. Bender et al. describe in [6] how this distance vector is computed for a rotational constraint. For a three-dimensional joint constraint the distance vector exactly describes the drift error $\mathbf{e}$ of one step that has to be prevented by internal forces or impulses. In the case of a lower-dimensional constraint, the vector $\mathbf{d}$ is projected onto the space of the constraint by a projection matrix $\mathbf{P}$ in order to get the corresponding drift vector $\mathbf{e} = \mathbf{P}\mathbf{d}$. The computation of the projection matrices for the basic joints is described in [6]. The problem of computing an impulse that eliminates the occurring drift in one single step can be described by a nonlinear equation [14]. This equation can be solved by Newton iteration. In contrast to that a linearisation is used in this paper. The relative motion of two articulated bodies is almost linear during a small time step. Because of this, the required change of their relative velocity in order to eliminate the drift can be approximated by

$$\boldsymbol{\Delta}\mathbf{v} = \frac{1}{h}\,\mathbf{e}.$$

An impulse (or angular momentum, in the case of a rotational constraint) that causes exactly this velocity change can be computed easily. This impulse must be applied in opposite directions to the bodies to guarantee the conservation of momentum. Since an approximation was used, the impulse computation has to be repeated iteratively until the predicted joint state satisfies its corresponding constraint within a given tolerance. In general, even for small tolerance values only a few iterations are required. After all impulses are determined and applied the new positions and velocities of the bodies can be computed by solving the equations 1–4. The last step is to satisfy the velocity constraints. This is called *velocity correction*. Since an impulse causes an instantaneous velocity change, no prediction is needed for the computation. The required velocity change $\boldsymbol{\Delta}\mathbf{v}$ is given by the relative velocity in direction of the constraint. The impulses are determined in the same way as for the position constraints. The described process converges to the physical correct solution (a proof can be found in [12]).

In the following the determination of the joint impulses will be described in detail. The required velocity change $\boldsymbol{\Delta}\mathbf{v}$ for each joint is already known. The problem is that joints can have common bodies and so the corresponding impulses influence each other. These dependencies must be regarded when computing the impulses. Due to the used linearisation it is possible to describe all dependencies in a system of linear equations:

$$\mathbf{A}\,\mathbf{x} = \boldsymbol{\Delta}\mathbf{v}, \tag{5}$$

where $\mathbf{x}$ is a vector that contains all impulses and angular momenta that should be determined. The vector $\boldsymbol{\Delta}\mathbf{v}$ contains the velocity changes of all joints (either for the position correction or the velocity correction). The dependencies between the joints are mapped on the matrix $\mathbf{A}$. This matrix specifies how the velocities change when the impulses and angular momenta are applied. The SLE of a multi-body system with n joints has the dimension $\sum_{i=1}^{n} \dim(i)$ where $\dim(i)$ is the dimension of the i-th joint constraint. Matrix $\mathbf{A}$ is a block matrix. A block $\mathbf{A}_{i,j} \in \mathbb{R}^{\dim(i)\times\dim(j)}$ describes how the relative velocity of joint i changes, if in joint j an impulse is applied.

The joints of a multi-body system are divided into two disjoint index sets T and R. Let $T = \{1,\ldots,m\}$ be the index set of all translational joint constraints and $R = \{m+1,\ldots,n\}$ be the one of all rotational constraints. The block $\mathbf{A}_{i,j}$ depends on the types of joint i and j. To differentiate between the four possible cases the following matrix is introduced for two joints i and j with a common dynamic body k:

$$\mathbf{N}_{i,j,k} = \begin{cases} \frac{1}{m_k}\mathbf{I}_3 - \mathbf{r}_a^*\mathbf{J}_k^{-1}\mathbf{r}_b^* & \text{if } i,j \in T \\ \mathbf{J}_k^{-1} & \text{if } i,j \in R \\ \mathbf{J}_k^{-1}\mathbf{r}_b^* & \text{if } i \in R \wedge j \in T \\ -\mathbf{r}_a^*\mathbf{J}_k^{-1} & \text{if } i \in T \wedge j \in R. \end{cases}$$

In the case of translational constraints, $\mathbf{a}$ and $\mathbf{b}$ are the corresponding joint points in body k. If the body k is static, the matrix $\mathbf{N}_{i,j,k}$ is zero. This matrix describes how the velocity of joint i changes, if in joint j an impulse or angular momentum is applied.

With the matrix $\mathbf{N}_{i,j,k}$ a block matrix $\mathbf{B}$ for three-dimensional constraints can be defined. For the computation of a block $\mathbf{B}_{i,j}$ it must be considered, if the common body is the first or the second body of the joints. Furthermore, two joints can have more than one common body. So a single block is defined as follows:

$$\mathbf{B}_{i,j} = \begin{cases} \mathbf{N}_{i,j,k_{i_1}} & \text{if } k_{i_1} = k_{j_1} \wedge k_{i_2} \neq k_{j_2} \\ \mathbf{N}_{i,j,k_{i_2}} & \text{if } k_{i_2} = k_{j_2} \wedge k_{i_1} \neq k_{j_1} \\ \mathbf{N}_{i,j,k_{i_1}} + \mathbf{N}_{i,j,k_{i_2}} & \text{if } k_{i_1} = k_{j_1} \wedge k_{i_2} = k_{j_2} \\ -\mathbf{N}_{i,j,k_{i_1}} & \text{if } k_{i_1} = k_{j_2} \wedge k_{i_2} \neq k_{j_1} \\ -\mathbf{N}_{i,j,k_{i_2}} & \text{if } k_{i_2} = k_{j_1} \wedge k_{i_1} \neq k_{j_2} \\ -(\mathbf{N}_{i,j,k_{i_1}} + \mathbf{N}_{i,j,k_{i_2}}) & \text{if } k_{i_1} = k_{j_2} \wedge k_{i_2} = k_{j_1} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

where an index k_{i_1} belongs to the first body of joint i.

The block matrix $\mathbf{A}$ for all basic joint constraints must consider lower-dimensional constraints as well. For a lower-dimensional constraint the corresponding equations must be projected onto the space of the joint. This can be done by using the projection matrix $\mathbf{P}$ that has already been used for the computation of the drift vector. A block of the matrix $\mathbf{A}$ has the following general form:

$$\mathbf{A}_{i,j} = \mathbf{P}_i\,\mathbf{B}_{i,j}\,\mathbf{P}_j.$$

The impulses and angular momenta of all joints are determined by solving the described system of linear equations. But before they can be applied to the bodies they must be transformed from the corresponding joint space to world space using the projection matrix:

$$\mathbf{p}_{\text{world}} = \mathbf{P}^T\,\mathbf{p}_{\text{joint}}.$$

The factorisation of the matrix is the most time-consuming part of the simulation step. The matrix of the SLE is constant at a time t. For this reason the velocity correction and the position correction of the following simulation step have the same matrix. Hence the system of linear equations for the impulses has to be factorised only once per simulation step.

4. LINEAR-TIME DYNAMICS

The matrix of the used SLE is often sparse. In the case of a sparse system special solvers like PARDISO [10, 11] can be used in order to accelerate the dynamic simulation. But in the general case for example a LU factorisation of the matrix has a time complexity of $O(n^3)$.

In this section a dynamic simulation method for acyclic models is presented that has a time complexity of $O(n)$. The main idea of this method is to bring the SLE in the following form:

$$\mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T\,\mathbf{x} = \boldsymbol{\Delta}\mathbf{v}, \tag{6}$$

where $\mathbf{M}$ is the mass matrix of all bodies and $\mathbf{C}$ is the constraint matrix. The mass matrix of a single rigid body k is defined by the mass and the inertia tensor:

$$\mathbf{M}_k = \begin{pmatrix} m_k\,\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_k \end{pmatrix},$$

where $\mathbf{I}_3 \in \mathbb{R}^3$ is the identity matrix. The matrix $\mathbf{M}$ is a block matrix with the blocks $\mathbf{M}_k$ of all bodies on the diagonal. The constraint matrix $\mathbf{C}$ is also a block matrix. A block $\mathbf{C}_{i,k}$ is non-zero if and only if the dynamic body k is connected by joint i. Hence the constraint matrix of the impulse-based method has exactly the same structure as the constraint matrix of the Lagrange multiplier method in [2]. David Baraff showed in his paper that a SLE for Lagrange multipliers in form of equation 6 can be solved in linear time, if the simulated model has no cycles. In the same way, the impulse-based method can determine all impulses in linear time, since the corresponding SLE has the same structure as the one of Baraff.

The SLE for the impulses is not solved directly. First it is transformed into the following form:

$$\underbrace{\begin{pmatrix} \mathbf{M} & -\mathbf{C}^T \\ -\mathbf{C} & \mathbf{0} \end{pmatrix}}_{\mathbf{H}} \begin{pmatrix} \mathbf{y} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -\boldsymbol{\Delta}\mathbf{v} \end{pmatrix}. \tag{7}$$

In this form the SLE is larger but it has the advantage that $\mathbf{H}$ is always sparse. By a depth-first search on the corresponding joint graph the matrix $\mathbf{H}$ is ordered so that the index of each node is greater than the indices of its children. For the factorisation of the matrix a decomposition $\mathbf{H} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ is used. Due to the reordered matrix this decomposition does not introduce new nonzero elements. This property can be used to factorise the matrix and solve the SLE in linear time. This is described in detail in [2].

In the following is presented how the decomposition $\mathbf{A} = \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T$ of matrix $\mathbf{A}$ is done in order to bring the SLE in form of equation 6. Since the mass matrix $\mathbf{M}$ is already known, just the constraint matrix $\mathbf{C}$ must be determined. First, only single joints are regarded and the constraint matrix is defined for translational and rotational constraints. Then it is shown how the dependencies in a multi-body system are taken into account.

4.1 Translational constraints

For a single translational joint constraint connecting two dynamic bodies k_1 and k_2 with the joint points $\mathbf{a}$ and $\mathbf{b}$ the matrix $\mathbf{A}$ is defined as follows:

$$\mathbf{K}_{a,k} := \begin{cases} \frac{1}{m_k}\mathbf{I}_3 - \mathbf{r}_a^*\mathbf{J}_k^{-1}\mathbf{r}_a^* & \text{if } k \text{ is dynamic} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\mathbf{A} = \mathbf{P}(\mathbf{K}_{a,k_1} + \mathbf{K}_{b,k_2})\mathbf{P}^T.$$

The projection matrix of a three-dimensional constraint is the identity matrix. In this special case the constraint matrix $\tilde{\mathbf{C}}$ must satisfy $\mathbf{K}_{a,k_1} + \mathbf{K}_{b,k_2} = \tilde{\mathbf{C}}\mathbf{M}^{-1}\,\tilde{\mathbf{C}}^T$. This is true for

$$\tilde{\mathbf{C}} = \begin{pmatrix} \mathbf{I}_3 & \mathbf{r}_a^* & \mathbf{I}_3 & \mathbf{r}_b^* \end{pmatrix}.$$

A lower-dimensional constraint is decomposed in the following way:

$$\begin{aligned} \mathbf{A} &= \mathbf{P}\left(\tilde{\mathbf{C}}\mathbf{M}^{-1}\tilde{\mathbf{C}}^T\right)\mathbf{P}^T \\ &= (\mathbf{P}\,\tilde{\mathbf{C}})\,\mathbf{M}^{-1}\,(\mathbf{P}\,\tilde{\mathbf{C}})^T = \mathbf{C}\,\mathbf{M}^{-1}\,\mathbf{C}^T. \end{aligned}$$

Hence the constraint matrix for translational joint constraints is defined by

$$\mathbf{C} = \mathbf{P}\,\tilde{\mathbf{C}}. \tag{8}$$

4.2 Rotational constraints

The definition of the constraint matrix for a rotational constraint is analogue. The matrix $\mathbf{A}$ of the SLE for a single rotational joint constraint which connects the dynamic bodies k_1 and k_2 is:

$$\mathbf{L}_k := \begin{cases} \mathbf{J}_k^{-1} & \text{if } k \text{ is dynamic} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\mathbf{A} = \mathbf{P}(\mathbf{L}_{k_1} + \mathbf{L}_{k_2})\mathbf{P}^T.$$

The corresponding three-dimensional constraint matrix is defined as:

$$\tilde{\mathbf{C}} = \begin{pmatrix} \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{I}_3 \end{pmatrix}.$$

The matrix for a lower-dimensional rotational constraint is determined by equation 8.

4.3 A system of constraints

The constraint matrix for a system of constraints must consider the dependencies between different joints with a common body. For each joint i there exist $\dim(i)$ rows in the constraint matrix $\mathbf{C}$. The matrix has $\sum_{j=1}^{n_k}\dim(k_j)$ columns, where n_k is the number of articulated bodies and $\dim(k_j)$ is the number of degrees of freedom of body k_j. Matrix $\mathbf{C}$ is a block matrix and consists of blocks $\mathbf{C}_{i,k}$ which have a dimension of $\dim(i) \times \dim(k)$ [1]. A block is not zero if and only if k is a body of joint i.

A block $\mathbf{C}_{i,k}^t$ for the translational joint constraint i and one of its bodies k is defined analogously to the constraint matrix in section "Translational constraints":

$$\mathbf{C}_{i,k}^t = \begin{pmatrix} \mathbf{P}_i & \mathbf{P}_i\,\mathbf{r}_a^* \end{pmatrix},$$

where $\mathbf{a}$ is the joint point of the corresponding body k and $\mathbf{P}_i$ is the projection matrix of the joint i. The blocks of

[1] In the case of rigid bodies $\dim(k)$ is 6. If particles are simulated, the dimension is 3 since the mass matrix for a particle is defined by $m_k\mathbf{I}_3 \in \mathbb{R}^3$.

rotational joint constraints are defined analogously to the matrix of section "Rotational constraints":

$$\mathbf{C}^r_{i,k} = \begin{pmatrix} \mathbf{0} & \mathbf{P}_i \end{pmatrix}.$$

If there exist multiple joints with a common dynamic body, the corresponding impulses influence each other. These dependencies are described by the off-diagonal blocks of the matrix $\mathbf{A}$ (see section "Impulse-based dynamic simulation"). If the constraint matrix $\mathbf{C}$ is generated by using the blocks $\mathbf{C}^t_{i,k}$ and $\mathbf{C}^r_{i,k}$, the diagonal blocks of the matrices $\mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T$ and $\mathbf{A}$ are equal but the off-diagonal blocks can have different signs. To obtain the correct signs, the following matrix blocks are used to generate the constraint matrix $\mathbf{C}$:

$$\mathbf{C}_{i,k} = \begin{cases} \mathbf{C}^t_{i,k} & \text{if } i \in T \wedge k = k_{i_1} \\ -\mathbf{C}^t_{i,k} & \text{if } i \in T \wedge k = k_{i_2} \\ -\mathbf{C}^r_{i,k} & \text{if } i \in R \wedge k = k_{i_1} \\ \mathbf{C}^r_{i,k} & \text{if } i \in R \wedge k = k_{i_2} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

where k_{i_1} and k_{i_2} are the first and second body of joint i respectively.

Since the decomposition $\mathbf{A} = \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T$ is known, the SLE for the impulses (see equation 5) can be transformed into a sparse SLE in form of equation 7. This sparse system has the same structure as the one of the Lagrange multiplier method described in [2]. Hence it can be factorised and solved in linear time by reordering the matrix and using a $\mathbf{L}\mathbf{D}\mathbf{L}^T$ decomposition.

5. CLOSED KINEMATIC CHAINS AND COLLISIONS

The linear-time algorithm can not simulate models with loops directly. It is necessary to break up loops by removing constraints and to handle the removed constraints separately. The method can also not handle collisions and permanent contacts because collisions and contacts define inequality constraints for the bodies. These constraints can not be described in a SLE. Hence collisions and contacts must be resolved independently of the joint constraints.

In the following is described how systems of constraints can be split up in different parts and be simulated separately while regarding the dependencies between the parts. The constraints of a model with loops and collisions should be split up in at least three parts: one part for collisions and contacts and two acyclic parts containing the joint constraints. The division in more parts makes sense, if the constraints should be handled in parallel, e.g. on a multi-core system. The collision and contact constraints can be handled with the method described in [5]. If a model contains cycles, it has to be split up in at least two parts. Cycles are determined in a preprocessing step before the simulation starts. The connection structure of the model can be described by an undirected graph. This graph is used to determine all cycles. Then the model is split up in multiple acyclic parts of similar size. These parts are simulated with the linear-time method presented in this paper.

If there exist two joints of different parts with a common dynamic body, the parts depend on each other. Dependencies are resolved by an iterative method. The impulse-based method which uses an iterative approach is able to simulate loops without special treatment. The presented linear-time method can also benefit from this property. Let us assume that a model has been divided into multiple acyclic parts which depend on each other. The constraints of each part are satisfied by the linear-time method. The dependencies between the single parts are resolved, if the parts are handled in an iterative loop until all constraints are satisfied within a given tolerance.

An example for the presented approach for models with loops is the simulation of cloth. A piece of cloth can be represented by a grid of particles that are connected by distance joints. Figure 1 shows such a grid where the particles are represented by dots and the distance joints by lines. The

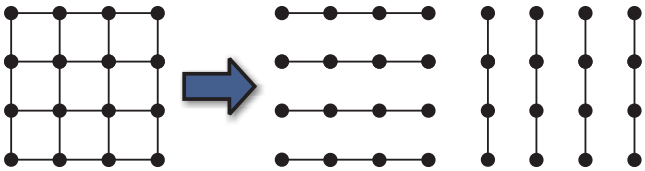

Figure 1: Division of a grid in 8 acyclic chains

grid is split up in eight acyclic chains. These chains can be simulated in parallel with the presented linear-time method. The dependencies between the chains are resolved by computing impulses for the chains in an iterative loop. In this way, the cloth in figure 2 which consists of a 21×21 grid can be simulated.

Figure 2: Objects falling onto a piece of cloth

6. RESULTS

In this section the presented linear-time method is compared to well-known impulse-based simulation methods and to Lagrange multiplier methods. The simulations have been performed on a PC with a 3.4 GHz Intel Pentium 4 processor. All implemented simulation methods use a fourth-order Runge-Kutta method for numerical integration. A tree model was simulated for the comparison like the one shown in figure 3. The shown model consists of 128 rigid

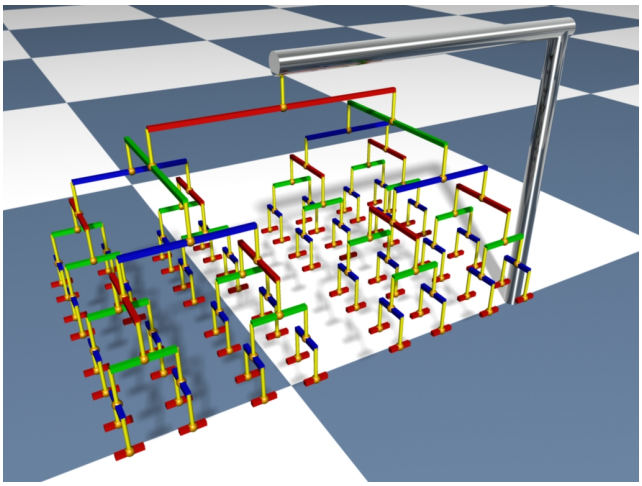

Figure 3: A tree model consisting of 128 rigid bodies and 127 ball joints

bodies and 127 ball joints which have three-dimensional constraints. The width and height of each body in the tree is 4 cm and its length depends on its depth in the tree in the following way:

$$l = 1.5^{(\text{max. depth} - \text{actual depth})} \cdot 0.1\,\text{m}.$$

All bodies have the same density of $\rho = 600\,\frac{\text{kg}}{\text{m}^3}$. The time step size used for the dynamic simulation was $h = \frac{1}{30}\,\text{s}$ in order to generate 30 frames per second. At the beginning of each simulation the top body of the tree was accelerated by an external torque of 10 Nm.

Tree models of different sizes have been simulated in order to show the scalability of the methods. The smallest model had 31 and the largest 255 joints. Figure 4 shows the average computation times of a single simulation step. Each

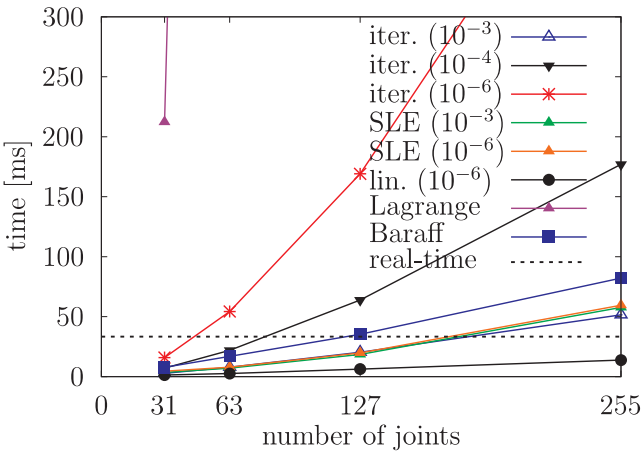

Figure 4: Computation time per simulation step

model has been simulated using the iterative method (iter.) [4], the SLE method (SLE) [6], the linear-time method presented in this paper (lin.), the standard Lagrange multiplier approach (Lagrange) and the linear-time method using Lagrange multipliers of David Baraff (Baraff) [2]. The method of Baumgarte [3] was used to stabilise the Lagrange multiplier methods.

The Lagrange multiplier method using the standard approach was the slowest method. The SLE for the Lagrange multipliers was solved with a LU factorisation that has a complexity of $O(n^3)$. The algorithm of David Baraff benefits from the tree structure of the model. It determines the Lagrange multipliers in linear time by using an optimised $\mathbf{LDL}^T$ decomposition [2]. This method simulated a tree with 63 joints faster than real-time. For the largest tree with 255 joints it needed an average computation time of 82.05 ms per simulation step.

The simulation with the iterative method has been performed with three different tolerance values. The number of required iterations is heavily dependent on these values. The most accurate simulation used a tolerance of 10^{-6} m for the position correction and a value of $10^{-6}\,\frac{\text{m}}{\text{s}}$ for the velocity correction. Using these values the tree with 31 joints was simulated two times faster than real-time. The simulation of the largest tree was much slower and needed 494.15 ms per step. The iterative method using tolerance values of 10^{-4} m and $10^{-4}\,\frac{\text{m}}{\text{s}}$ was almost three times faster for this model. For the last simulation with the iterative method the values 10^{-3} m and $10^{-3}\,\frac{\text{m}}{\text{s}}$ were used. All models were simulated faster than using the method of David Baraff. The tree with 127 joints could be simulated about 1.6 times faster than real-time, whereas the largest tree needed a computation time of 51.47 ms.

If the impulses are computed using a system of linear equations, all dependencies between the joints in the tree are regarded. Because of this, the tolerance values have not much influence on the computation times. To demonstrate this property, the SLE method has been simulated using two different tolerance values. Both simulations had nearly the same computation times as the iterative method with the largest tolerance value. The computation time of the SLE method for the tree with 255 joints was 57.72 ms using a tolerance of 10^{-3} (for the position and the velocity correction) and 59.48 ms using a tolerance value of 10^{-6} respectively. The factorisation of the matrix is the most time-consuming part of this method. The factorisation of a tree with 127 joints took 15.81 ms whereas the determination of the impulses in one iteration required 1.01 ms. Even if high accuracy is demanded, the SLE method needs only very few iterations. That is the reason why the computation times are very similar for different tolerance values.

The linear-time method presented in this paper used a tolerance value of 10^{-6} in order to achieve very accurate results. The simulation of the smallest model with 31 joints took 1.26 ms per simulation step. So it was more than 26 times faster than real-time. Even the largest tree with 255 joints has been simulated almost three times faster than real-time. A simulation step with this model required 13.70 ms. The dynamic simulation using the presented method was nearly six times faster than using the Lagrange multiplier method of David Baraff. The main reason why the impulse-based method is so much faster is the use of impulses instead of internal forces. The impulses have to be determined just once per simulation step. Then the positions and velocities

for the next step are determined by integration. Since the constraints are satisfied by the applied impulses, no internal forces have to be considered while integrating. In contrast to that the Lagrange multiplier method satisfies the constraints by internal forces. These forces can be computed in linear time by solving the corresponding SLE with the method of David Baraff. For one integration step with the fourth-order Runge-Kutta method the internal forces have to be determined four times. This explains the speed-up when using impulses.

7. CONCLUSION

An impulse-based dynamic simulation method for articulated bodies was presented that has a linear time complexity. This method can handle all kind of translational and rotational constraints for the positions and velocities of bodies. The presented method can handle the same models as Lagrange multiplier methods but it has no drift problem and needs no additional stabilisation. It has been shown that the new method is almost six times faster than the linear-time method using Lagrange multipliers when simulating a tree model. Closed loops, collisions and permanent contacts with friction can be handled by the integration of the linear-time method in an iterative process. In the same way it is possible to split up the model and simulate the different parts in parallel.

8. REFERENCES

[1] U. M. Ascher, H. Chin, L. R. Petzold, and S. Reich. Stabilization of constrained mechanical systems with daes and invariant manifolds. *Journal of Mechanics of Structures and Machines*, 23:135–158, 1995.

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

[3] J. W. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 1:1–16, 1972.

[4] J. Bender, D. Finkenzeller, and A. Schmitt. An impulse-based dynamic simulation system for VR applications. In *Proceedings of Virtual Concept 2005*, Biarritz, France, 2005. Springer.

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

[6] J. Bender and A. Schmitt. Fast dynamic simulation of multi-body systems using impulses. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)*, pages 81–90, Madrid (Spain), Nov. 2006.

[7] J. G. de Jalon and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems: the Real Time Challenge*. Springer-Verlag, New York, 1994.

[8] R. Featherstone. *Robot Dynamics Algorithm*. Kluwer Academic Publishers, Norwell, MA, USA, 1987. Manufactured By-Kluwer Academic Publishers.

[9] R. Featherstone and D. Orin. Robot dynamics: Equations and algorithms. *International Conference on Robotics and Automation*, pages 826–834, 2000.

[10] O. Schenk and K. Gärtner. On fast factorization pivoting methods for sparse symmetric indefinite systems. Technical Report, Department of Computer Science, University of Basel, 2004.

[11] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems*, 20(3):475–487, 2004.

[12] A. Schmitt, J. Bender, and H. Prautzsch. On the convergence and correctness of impulse-based dynamic simulation. Internal Report 17, Institut für Betriebs- und Dialogsysteme, 2005.

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

[14] R. L. Weinstein, J. Teran, and R. Fedkiw. Dynamic simulation of articulated rigid bodies with contact and collision. In *IEEE Transactions on Visualization and Computer Graphics*, volume 12, pages 365–374, 2006.

[15] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 243–252, New York, NY, USA, 1990. ACM Press.