Interactive Simulation of Rigid Body Dynamics in Computer Graphics

Jan Bender[1], Kenny Erleben[2], Jeff Trinkle[3] and Erwin Coumans[4]

[1]Graduate School CE, TU Darmstadt, Germany
[2]Department of Computer Science, University of Copenhagen, Denmark
[3]Department of Computer Science, Rensselaer Polytechnic Institute, USA
[4]Advanced Micro Devices, Inc., USA

Abstract

Interactive rigid body simulation is an important part of many modern computer tools. No authoring tool nor a game engine can do without. The high performance computer tools open up new possibilities for changing how designers, engineers, modelers and animators work with their design problems.
This paper is a self contained state-of-the-art report on the physics, the models, the numerical methods and the algorithms used in interactive rigid body simulation all of which has evolved and matured over the past 20 years. The paper covers applications and the usage of interactive rigid body simulation.
Besides the mathematical and theoretical details that this paper communicates in a pedagogical manner the paper surveys common practice and reflects on applications of interactive rigid body simulation. The grand merger of interactive and off-line simulation methods is imminent, multi-core is everyman's property. These observations pose future challenges for research which we reflect on. In perspective several avenues for possible future work is touched upon such as more descriptive models and contact point generation problems. This paper is not only a stake in the sand on what has been done, it also seeks to give newcomers practical hands on advices and reflections that can give experienced researchers afterthought for the future.

Keywords: Rigid Body Dynamics, Contact Mechanics, Articulated Bodies, Jointed Mechanisms, Contact Point Generation, Iterative Methods.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Physically-based modeling; Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation; Mathematics of Computing [G.1.6]: Numerical Analysis—Nonlinear programming

1. Motivation and Perspective on Interactive Rigid Body Simulation

Rigid body dynamics simulation is an integral and important part of many modern computer tools in a wide range of application areas like computer games, animation software for digital production including special effects in film and animation movies, robotics validation, virtual prototyping, and training simulators just to mention a few.

In this paper we focus on interactive rigid body dynamics simulation a subfield that has evolved rapidly over the past 10 years and moved the frontier of run-time simulation to applications in areas where off-line simulation only recently were possible. As a consequence this changes the computer tools humans use and has great social economical impact on society as a whole.

The term "interactive" implies a loop closed around a human and simulation tool. For applications like games where the feedback is simply animation on a screen, a reasonable goal is that the simulation deliver 60 frames per second (fps). For haptic rendering, the simulation would be part of a feedback loop running at 1000Hz, where this rate is needed to display realistic forces to the user.

In this state-of-the-art paper we will cover the important past 20 years of work on interactive rigid body simulation since the last state-of-the-art report [Bar93b] on the subject. Rigid body dynamics has a long history in computer graph-

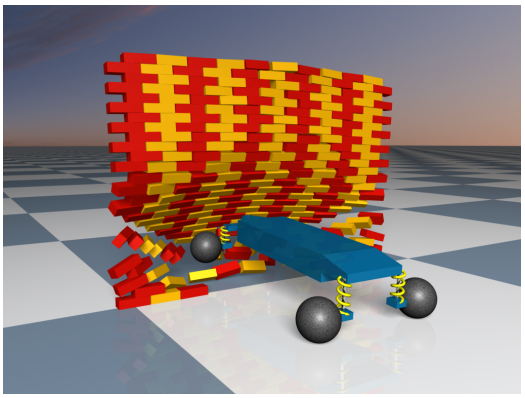


Figure 1: *Interactive rigid body simulations require the efficient simulation of joints, motors, collisions and contacts with friction.*

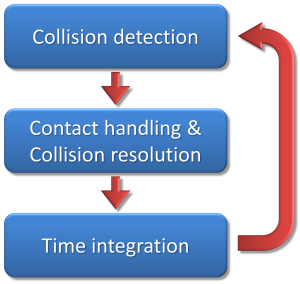


Figure 2: *The simulation loop provides a coarse description of data flow and processes in a rigid body simulator.*

ics for more than 30 years [AG85, MW88, Hah88, Bar89, BBZ91] and a wealth of work exists on the topic. As early as 1993 there were written state-of-the-art reports on the subject [Bar93b]. In his 93 STAR paper Baraff discussed penalty based methods and constraint based methods being an acceleration-level linear complementarity problem formulation. He did not cover many details on solving the linear complementarity problem. Not until 94 where Baraff published his version of a direct method based on pivoting was it feasible to compute solutions for Baraff's complementarity problem formulation. For years the 94 Baraff solution was the de-facto standard method of rigid body dynamics choice in both Maya and Open Dynamics Engine [Smi00]. However, the solution only remained interactive for small sized configurations (below 100 interacting objects or so). When the number of interacting objects increased the computational cost quickly made simulations last for hours and the acceleration-level formulation caused problems too with existence of solutions and uniqueness. Besides, solutions found by his algorithm did not always satisfy the static friction constraints. In the following years after Baraff's 1994 results, the impulse based paradigm was revisited by Mirtich in 96 [Mir96b] and become a strong competitor when concerning interactive simulation. Soon the interactive simulation community moved onto iterative methods and velocity level formulations, eventually evolving into the technology one finds today in engines such as Bullet [Cou05] and Open Dynamics Engine. As of this writing interactive simulation on single core CPUs with several 1000 and up to 10000 interacting objects are feasible. Multi-core and GPU works even go far beyond these limits. Even today much active cross-disciplinary work is ongoing on different contact formulations and iterative solvers taking in people not only from the field of computer graphics, but also from applied math, contact mechanics, robotics and more. Looking beyond contact problems, one also finds that simulation methods for articulated bodies have also undergone rapid development. In computer graphics, the reduced coordinate formulations have won much recognition as being superior for interactive rag-doll simulations.

1.1. The Anatomy and Physiology of a Rigid Body Simulator

A rigid body simulator is a complex and large piece of software. Traditionally it has been broken down into smaller well-defined pieces that each are responsible for solving a simpler single task. All pieces are tied together by a simulation loop shown in Figure 2. The loop begins with a collision detection query to find the contact points between the various bodies. These points are needed to write the physical laws governing the motions of the bodies, which are then solved to determine contact forces that provide proper contact friction effects and prevent bodies from interpenetrating. This phase is termed "contact handling." Newly formed contacts imply collisions, which are accompanied by impulsive forces (i.e., forces with infinite magnitudes over infinitesimal time periods). Impulsive forces cause instantaneous changes in the body velocities and so are often handled separately from pre-existing resting contacts. One refers to this as "collision resolving." After computing all the contact forces, the positions and velocities of the bodies are integrated forward in time before a new iteration of the simulation loop starts. Several iterations of the loop might be performed before a frame is rendered.

In order to derive the correct physical laws for the scene, all contacts between bodies must be found. If there are n bodies, then there are $O(n^2)$ pairs of bodies to test for collisions. To avoid collision detection becoming a computational bottleneck, it is broken into phases. In the first phase, called the "broad phase", bodies are approximated by simple geometric primitives for which distance computations are very fast (see Figure 3). For example, each body is replaced by the smallest sphere that completely contains it. If the spheres covering two bodies do not overlap, then neither do the actual bodies. The broad phase culling happens in global world coordinates. If the individual bodies are complex and

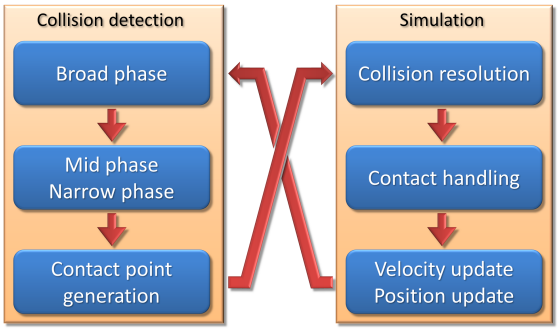


Figure 3: *A modular phase description of the sub tasks of a rigid body simulator helps decomposing a large complex system into simpler components.*

consist of many parts an additional stage called "mid phase" is used to cull parts in local body space. The culling is typically performed using bounding volume hierarchies. In the "narrow phase" the detailed geometries of bodies are used to find the precise body features in contact and the location of the contact points. However, this expensive operation must be done only for the pairs of bodies with overlapping approximations. The narrow phase is often mixed with the mid phase for performance reasons. Note that some narrow phase algorithms do not return all the required contact information, in which case a separate contact point generation algorithm can be applied (see section 6.3)

1.2. The Quest for Robustness, Accuracy, and Performance

The recent trend in interactive rigid body simulation has focused on delivering larger and larger simulations of rigid bodies or creating simulation methods that can deliver results faster. Thus, the old saying *bigger and faster is better* is very descriptive for many past works in the field of interactive rigid body simulation. The need for bigger and faster is motivated by rigid body simulators being used in for instance digital production. It looks more interesting to have a pile of skeleton skulls in a movie than having a hand full of cubes and spheres. Thus, the need in production for creating interesting motion requires more complex simulation scenarios.

The well known tradeoff between accuracy and performance is an inherent property of interactive rigid body simulation. Many applications enforce a performance constraint which leaves too little time for computing accurate solutions. Thus, one must often balance accuracy and stability properties to meet the performance constraint.

Robustness is another desirable numerical trait of a simulator. The motivation for this is often caused by having a human being (or the real world in case of robotics) interacting with a simulator. This may be the cause for much pain and frustrations as humans have a tendency to be unpredictable.

In summary, the holy grail of interactive rigid body simulation is extremely fast and robust simulation methods that can deal gracefully with large scale complex simulation scenarios under hard performance constraints.

1.3. Application Areas of Interactive Rigid Body Simulation

The maturing technology makes it possible to use rigid body simulators as sub-parts in larger systems. For instance in time critical scenarios like tracking humans or maneuvering a robot, a simulator can be used as a prediction tool.

From a digital design viewpoint, one may define a spectrum of technology. At one end of the spectrum one finds off-line simulators that may take hours or days to compute results, but on the other hand they deliver high quality results. For movie production several such computer graphics simulation methods have been presented [Bar94, GBF03, KSJP08]. At the other end of the spectrum one finds the fast run-time simulators capable of delivering plausible results very fast. This kind of simulator often originates from game physics. One example is Bullet. At the middle of the spectrum one finds moderately fast simulators that may deliver high fidelity results. These are very suitable for testing design ideas or training.

In general different application areas have different needs in regards to performance/quality trade-offs and accuracy. With this in mind we will discuss a few application areas in the next four subsections.

1.3.1. Entertainment for Games and Movies

For games and movies rigid body simulation has to be plausible rather than physically realistic. For games, the simulation needs to be real-time. Simulations for movies do not have the real-time constraint, but fast simulation methods are also preferred, since very complex scenarios are simulated for special effects and simulation time costs money. Therefore, the development in the two areas go in the same direction. Iterative constraint solving methods are popular in both areas.

Many games using 2D and 3D graphics rely on a rigid body dynamics engine to deal with collision detection and collision response. In some cases the motion of the objects is fully driven by rigid body dynamics, for example the game Angry Birds,using the Box2D physics engine or a 3D Jenga game. More commonly, object motion is customized in a non-physical way to a certain degree, to favor a satisfying game playing experience over physical realism. This introduces the challenge of interaction between rigid bodies and kinematically animated objects. Kinematically animated objects can be represented as rigid bodies with infinite mass,

so that the interaction is one way. The influence from rigid body to kinematically animated objects is often scripted in a non-physical way.

With increasing CPU budgets, there is growing interest in using more realistic, higher quality simulation. In particular the combination of rag-doll simulation, animation, inverse kinematics and control requires better methods. It requires constraint solvers that can deal with very stiff systems and strong motors that can deal with the large change in velocity. Several game and movie studios are using Featherstone's articulated body method to simulate rag-dolls.

Destruction and fracture of objects can generate a lot of dynamic rigid bodies, and to handle them, games use multicore CPUs or offload the rigid body simulation onto GPUs.

1.3.2. Interactive Digital Prototyping

Interactive virtual prototyping can be an important computer tool for verifying a design idea or as a pre-processing tool to tune parameters for more computational expensive simulation tools. CEA LIST [CEA11], CMLabs, robot simulators Webots, Gazebo, or Microsoft robotics developer studio [KP09, Cyb09, Mic09] are a few examples of many such tools. The main goal is to reduce the time to market and thereby lower overall production costs. A secondary goal is development of better products of higher quality.

Interactive prototyping has been motivated by the computational fast technology that has evolved in the gaming and movie industries. The instant feedback that can be obtained from such simulations is attractive for rapid iterative prototyping. However, although interactivity is attractive, one can not compromise the physical correctness too much. Thus, plausible simulation [BHW96] may not be good enough for trusting a virtual design. A current trend is seen where interactive simulation tools are improved for accuracy and moved into engineering tools [Stu08, TNA08, TNA*10, CA09].

Even the European space agency (ESA) is using PhysX for verification of the Mars sample rover for the ExoMars Programme to investigate the Martian environment [KK11].

1.3.3. Robotics

The main goal of the field of robotics is the development of intelligent man-made physical systems that can safely and efficiently accomplish a wide range of tasks that aid the achievement of human societal goals. Tasks that are particularly difficult, dangerous or boring are good candidates for robotic methods, e.g., assembly in clean-room environments, extra-terrestrial exploration, radioactive materials handling, and laparoscopic surgery. In addition, methods for robotics are increasingly being applied in the development of new generations of active prosthetic devices, including hands, arms and legs. The most challenging of these tasks most directly related to this paper are those that cannot be accomplished without (possibly) intermittent contact, such as walking, grasping and assembly.

Until now, robot manipulation tasks involving contact have been limited to those which could be accomplished by costly design of a workcell or could be conducted via teleoperation. The ultimate goal, however, is to endow robots with an understanding of contact mechanics and task dynamics, so they can reason about contact tasks, automatically plan and execute them, and enhance their manipulation skills through experience. The fundamental missing component has been fast, physical simulation tools that accurately model effects such as stick-slip friction, flexibility, and dynamics. Dynamics is important for robots to perform manipulation tasks quickly or to allow it to run over uneven terrain. Physical simulation today has matured to the point where it can be integrated into algorithms for robot design, task planning, state-estimation and control. As a result, the number of robotic solutions to problems involving contact is poised to experience a major acceleration.

1.3.4. Industrial and Training Simulators

Computer simulators are cheap and risk free ways to train people to handle heavy equipment in critical situations under large stress. Examples include large forest machines as well as bulldozers to cable simulations in tug boats and maneuver belt vehicles [SL08]. Driving simulators are another good example [Uni11, INR11]. The simulators in this field need to be responsive as well as accurate enough to give proper predictions of the virtual equipment being handled. This is similar to interactive virtual prototyping. In fact in our view it is mostly the purpose that distinguishes the two, one is design the other is training.

The driving simulators are very specialized and include many aspects of virtual reality. The largest and most complex simulator, National Advanced Driving Simulator (NADS), costs in the order of $54 million. In contrast, Gazebo is free software and commercial software such as Vortex [CM 11] or Algoryx [Alg11]) are cheap in comparison with NADS.

2. A Quick Primer

Rigid body simulation is analogous to the numerical solution of nonlinear ordinary differential equations for which closed-form solutions do not exist. Assume time t is the independent variable. Given a time period of interest $[t_0, t_N]$, driving inputs, and the initial state of the system, the differential equations (the instantaneous-time model) are discretized in time to yield an approximate discrete-time model, typically in the form of a system of (state-dependent) algebraic equations and inequalities. The discrete-time model is formulated and solved at each time of interest, $(t_0, .., t_N)$. In rigid body simulation, one begins with the Newton-Euler (differential) equations, which describes the dynamic motion

of the bodies without contact. These differential equations are then augmented with three types of conditions: nonpenetration constraints that prevent the bodies from overlapping, a friction model that requires contact forces to remain within their friction cones, and complementarity (or variational inequality) constraints that enforce certain disjunctive relationships among the variables. These relationships enforce critically important physical effects; for example, a contact force must become zero if two bodies separate and if bodies are sliding on one another, the friction force acts in the direction that will most quickly halt the sliding. Putting all these components together yields the instantaneous-time model, as a system of differential algebraic equations and inequalities that can be reformulated as a differential nonlinear complementarity problem (dNCP). The dNCP cannot be solved in closed form or directly, so instead, one discretizes it in time, thereby producing a sequence of NCPs whose solutions approximate the state and contact force trajectories of the system. In the ideal case, the discrete trajectories produced in this process will converge trajectories of the original instantaneous-time model. Computing a discrete-time solution requires one to consider possible reformulations of the NCPs and a choice of solution method. There are many options for instance reformulation as nonsmooth equation using Fischer-Burmeister function or proximal point mappings etc.

2.1. Classical Mechanics

Simulation of the motion of a system of rigid bodies is based on a famous system of differential equations, the *Newton-Euler equations*, which can be derived from Newton's laws and other basic concepts from classical mechanics:

- Newton's 1st law: The velocity of a body remains unchanged unless acted upon by a force.
- Newton's 2nd law: The time rate of change of momentum of a body is equal to the applied force.
- Newton's 3rd law: For every force there is an equal and opposite force.

Two important implications of Newton's laws when applied to rigid body dynamics are: (from the first law) the equations apply only when the bodies are observed from an inertial (non-accelerating) coordinate frame and (from the third law) at a contact point between two touching bodies, the force applied from one body onto the second is equal in magnitude, opposite in direction, and collinear with the force applied by the second onto the first. Applying these two implications to Newton's second law gives rise to differential equations of motion. While the second law actually applies only to particles, Euler was kind enough to extend it to the case of rigid bodies by viewing them as collections of infinite numbers of particles and applying a bit of calculus [GPS02, ESHD05]. This is why the equations of motion are known as the Newton-Euler equations.

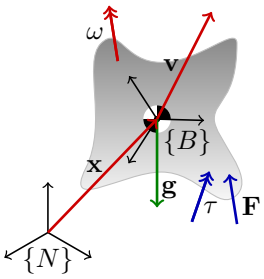


Figure 4: *Illustration of a spatial rigid body showing the body frame $\{B\}$ and inertial frame $\{N\}$ as well as notation for positions, velocities and forces.*

Before presenting the Newton-Euler equations, we need to introduce a number of concepts from classical mechanics. Figure 4 shows a rigid body in space, moving with translational velocity $\mathbf{v}$ and rotational velocity ω, while being acted upon by an applied force $\mathbf{F}$ and moment τ (also known as a torque).

2.1.1. Rigid Bodies

A *rigid body* is an idealized solid object for which the distance between every pair of points on the object will never change, even if huge forces are applied. A rigid body has mass m, which is distributed over its volume. The centroid of this distribution (marked by the circle with two blackened quarters) is called the center of mass. To compute rotational motions, the mass distribution is key. This is captured in a 3×3 matrix known as the mass (or inertia) matrix $\mathbf{I} \in \mathbb{R}^{(3\times3)}$. It is symmetric and positive definite matrix with elements known as moments of inertia and products of inertia, which are integrals of certain functions over the volume of the body [Mei70]. When the integrals are computed in a body-fixed frame, the mass matrix is constant and will be denoted by $\mathbf{I}_{\text{body}}$. The most convenient body-fixed frame for simulation is one with its origin at the center of mass and axes oriented such that $\mathbf{I}_{\text{body}}$ is diagonal. When computed in the inertial frame, the mass matrix is time varying and will be denoted by $\mathbf{I}$.

2.1.2. Rigid Body Kinematics

The body's position in the inertial (or world) frame is given by the vector $\mathbf{x} \in \mathbb{R}^3$, from the origin of the inertial frame $\{N\}$ fixed in the world to the origin of the frame $\{B\}$ fixed in the body. Note that since three independent numbers are needed to specify the location of the center of mass, a rigid body has three translational degrees of freedom.

The orientation of a rigid body is defined as the orientation of the body-fixed frame with respect to the inertial frame. While many representations of orientation exist, here we use rotation matrices $\mathbf{R} \in \mathbb{R}^{3\times3}$ and unit quaternions

$Q \in \mathbb{H}$. Rotation matrices are members of the class of *orthogonal matrices*. Denoting the columns by $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{R}_3$, orthogonal matrices must satisfy: $\| \mathbf{R}_i \| = 1$; $i = 1,2,3$ and $\mathbf{R}_i^T \mathbf{R}_j = 0$; $\forall i \neq j$; $i = 1,2,3$; $j = 1,2,3$. Since the nine numbers in $\mathbf{R}$ must satisfy these six equations, only three numbers can be freely chosen. In other words, a rigid body has three rotational degrees of freedom. A unit quaternion is four numbers $[Q_s,\ Q_x,\ Q_y,\ Q_z]$, constrained so that the sum of their squares is one. The fourth element can be computed in terms of the other three, and this redundancy serves as additional confirmation that orientation has three degrees of freedom. Considering translation and rotation together, a rigid body has six degrees of freedom.

The *rotational velocity* $\omega \in \mathbb{R}^3$ (also known as, *angular velocity*) of a body can be thought of as vector whose direction identifies a line about which all points on the body instantaneously rotate (shown as a red vector with a double arrowhead in Figure 4). The magnitude determines the rate of rotation. While the rate of rotation may be changing over time, at each instant, every point on a rigid body has exactly the same rotational velocity. The three elements of ω correspond to the three rotational degrees of freedom.

Translational velocity $\mathbf{v} \in \mathbb{R}^3$ (also inaccurately referred to as *linear velocity*) is an attribute of a point, not a body, because when a body rotates, not all points have the same velocity (see the red vector with a single arrowhead in Figure 4). However, the velocity of every point can be determined from the velocity of one reference point and the angular velocity of the body. In rigid body dynamics, the center of mass is typically chosen as the reference point.

Next we need velocity *kinematic* relationships. Kinematics is the study of motion without concern for forces, moments, or body masses. By contrast, *dynamics* is the study of how forces produce motions. Since dynamic motions must also be kinematically feasible, kinematics is an essential building block of dynamics. The particular kinematic relationships needed here relate the time derivatives of position and orientation variables to the translational and rotational velocities.

Let us define $\mathbf{q} = (\mathbf{x},\ Q)$ as the tuple containing the position of the center of mass and the orientation parameters. Note that the length of $\mathbf{q}$ is seven if Q is a quaternion (which is the most common choice). The generalized velocity of the body is defined as: $\mathbf{u} = [\mathbf{v}^T\ \omega^T]^T \in \mathbb{R}^6$. The velocity kinematic equations for a rigid body relate $\dot{\mathbf{q}}$ to $\mathbf{u}$, which may have different numbers of elements. The relationship between the translational quantities is simple: $\dot{\mathbf{x}} = \mathbf{v}$. The time rate of change of the rotational parameters Q is a bit more complicated; it is the product of a Jacobian matrix and the rotational velocity of the body: $\dot{Q} = \mathbf{G}(Q)\omega$, where the details of $\mathbf{G}(Q)$ are determined by the orientation representation. In the specific case when Q is a unit quaternion, $\mathbf{G}(Q)$ is defined as follows:

$$\mathbf{G} = \frac{1}{2}\begin{bmatrix} -Q_x & -Q_y & -Q_z \\ Q_s & Q_z & -Q_y \\ -Q_z & Q_s & Q_x \\ Q_y & -Q_x & Q_s \end{bmatrix}.$$

Putting the two velocity kinematic relationships together yields:

$$\dot{\mathbf{q}} = \mathbf{H}\mathbf{u} \tag{1}$$

where $\mathbf{H} = \begin{bmatrix} \mathbf{1}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \mathbf{G} \end{bmatrix}$, where $\mathbf{1}_{3\times 3}$ is the 3-by-3 identity matrix. Note that when the orientation representation uses more than three parameters, $\mathbf{G}$ is not square, although it has the property that $\mathbf{G}^T\mathbf{G} = \mathbf{1}$, where $\mathbf{1}$ is the identity matrix of size 3.

2.1.3. Constraints

Constraints are equations and inequalities that change the way pairs of bodies are allowed to move relative to one another. Since they are kinematic restrictions, they also affect the dynamics. Constraints do *not* provide a direct means to compute the forces that must exist to enforce them. Generally, constraints are functions of generalized position variables, generalized velocities, and their derivatives to any order:

$$C(\mathbf{q}_1, \mathbf{q}_2, \mathbf{u}_1, \mathbf{u}_2, \dot{\mathbf{u}}_1, \dot{\mathbf{u}}_2, ..., t) = 0 \tag{2}$$

or

$$C(\mathbf{q}_1, \mathbf{q}_2, \mathbf{u}_1, \mathbf{u}_2, \dot{\mathbf{u}}_1, \dot{\mathbf{u}}_2, ..., t) \geq 0 \tag{3}$$

where the subscripts indicate the body. Equality and inequality constraints are referred to as *bilateral* and *unilateral* constraints, respectively.

As an example, consider two rigid spheres of radii r_1 and r_2 and with centers located at $\mathbf{x}_1$ and $\mathbf{x}_2$. Consider the constraint function:

$$C(\mathbf{x}_1, \mathbf{x}_2) = ||\mathbf{x}_1 - \mathbf{x}_2|| - (r_1 + r_2),$$

where $||\cdot||$ is the Euclidean two-norm. If $C = 0$, then the surfaces of the spheres touch at a single point. If this bilateral constraint is imposed on the Newton-Euler equations, then regardless of the speeds of the spheres and the sizes of the forces, the surfaces will always remain in single-point contact. Intuitively, for this to happen the constraint force normal to the sphere surfaces can be compressive (the spheres push on each other) or tensile (the spheres pull). By contrast, $C \geq 0$, then the two spheres may move away from each other but never overlap. Correspondingly, the constraint force can only be compressive.

The form of a constraint (see Figure 5) impacts the way in which the Newton-Euler equations should be solved. *Holonomic* constraints are those which can be expressed as an equality in terms of only generalized position variables and time. These are further subdivided into those independent of

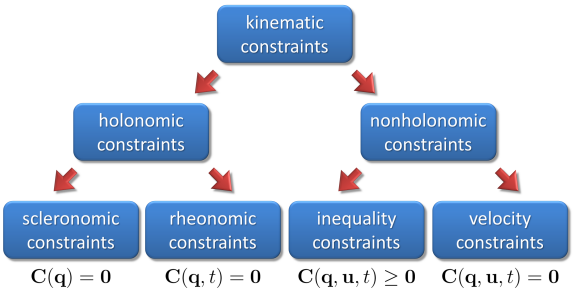


Figure 5: *Constraint classification*

time, known as *scleronomic*, and those dependent on time, *rheonomic*. An example of a scleronomic constraint is the equality constraint of the spheres discussed above). Rheonomic constraints typically arise when one body is kinematically controlled (*i.e.*, it is required to follow a known trajectory regardless of the forces that might be required to make that happen).

Any constraint that is not holonomic is said to be *nonholonomic*. This class includes all unilateral constraints and equality constraints which are not integrable in the sense that generalized velocity variables and derivatives of the generalized position variables (and higher derivatives, if present) cannot be eliminated. The steering constraint for a car on a flat surface whose wheels are not allowed to skid is a nonholonomic equality constraint. If the car is driving along, then its rotational velocity is directly proportional the car's forward speed and the angle of the front wheels. This means the fundamental constraint between two velocities cannot be integrated to yield an equivalent constraint written solely in terms of position variables, hence the constraint is nonholonomic.

Holonomic constraints remove degrees of freedom from the system, *i.e.,* the dimension of the space of possible generalized positions is reduced. For instance two free rigid bodies have a total of 12 degrees of freedom, but as in the previous case of the touching spheres, one degree of freedom is lost. Assume that one sphere can be moved at will through space using all six degrees of freedom. Now view the second sphere from a frame of reference fixed in the first. From this perspective, the second sphere can rotate with all three degrees of freedom while maintaining contact and also translate with the contact point moving across the surface of the first sphere. Since this surface is two-dimensional, the second sphere has only two translational degrees of freedom. Thus a system of two spheres with one contact constraint has 11 degrees of freedom. If instead, two bodies were connected by a hinge joint, the system would have seven degrees of freedom. That is, if you allow one body to move with six degrees of freedom, then the other can only rotate about the hinge joint with respect to the first body. This also implies that a hinge constraint cannot be represented with fewer than five holonomic constraints.

One should note that non-holonomic equality constraints remove only instantaneous, or local, degrees of freedom from the system. In the car example, the car cannot translate instantaneously directly left or right. However, every competent driver can accomplish a lateral move of his car by executing the kind of maneuver used to parallel park in a small space.

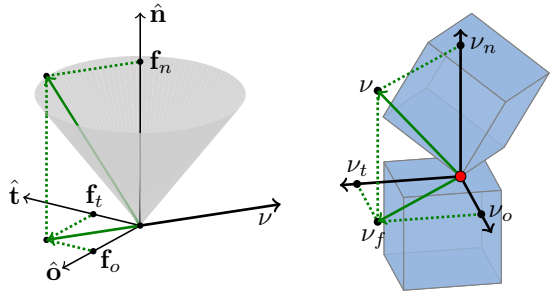
2.1.4. Forces and Moments and Relative Velocity

A *force* $\mathbf{f}$ is a vector with a line of action. A force produces a *moment* τ or *torque* about any point not on the line of action of the force. Let $\mathbf{r}$ and ρ be two distinct points such that $\mathbf{r}$ is on the line of action and ρ is not. Then the moment of $\mathbf{f}$ with respect to $\mathbf{r}$ is defined as $\tau = (\mathbf{r} - \rho) \times \mathbf{f}$. Moments need not be byproducts of forces; they exist in their own right, which is why one is shown applied to the body in Figure 4.

Many sources of forces exist in rigid body dynamics, for example, forces from wind, gravity, and electro-magnetics. However, the forces that are most difficult to deal with, but also critically important in interactive simulation are constraint and friction forces.

Gravity, as we experience it on Earth, acts equally on every particle of mass in a rigid body. Nonetheless, the gravity force is shown in Figure 4 as a single force of magnitude mg with line of action through the center of mass of the body. This is because the affect of gravity acting on an entire body is equivalent to a single force of magnitude mg acting through its center of mass. Friction forces are dissipative. They act in contact interfaces to halt sliding at sliding contacts and to prevent sliding at sticking and rolling contacts. The type of friction force focused on here is *dry friction*, which is assumed to act at contacts between body surfaces, including the inner surfaces of joints. Dry friction, as opposed to viscous friction, allows bodies to stick together and requires a non-zero tangential force to initiate sliding.

For point contacts between body surfaces, we consider the standard isotropic Coulomb friction model. Assume that contact occurs at a single point with a uniquely defined tangent plane. Then place the origin of the contact coordinate frame at the contact point and let the t- and o-axes lie in the tangent plane (see Figure 6(a)). The n-axis is orthogonal to the t- and o-axes and is referred to as the contact normal. A contact force $\mathbf{f}$ is decomposed into a normal componen $\mathbf{f}_n$ and tangential components, $\mathbf{f}_t$ and $\mathbf{f}_o$. Because bodies are able to push against each other, but not pull, the normal force is unilateral, *i.e.,* $\mathbf{f}_n \geq 0$. Similarly, the relative velocity between the touching points on the bodies ν is decomposed into components, ν_n, ν_t, and ν_o (see Figure 6(b)). The contact is sliding if $\nu_n = 0$ and ν_t or ν_o is nonzero, and separating if ν_n is greater than zero. Negative ν_n is not allowed, as it corresponds to interpenetration of the bodies.



(a) A Friction Cone (b) Contact velocities

Figure 6: *The friction cone of a contact and the decomposition of the relative contact velocity.*

The Coulomb model has two conditions: first, the net contact force must lie in a quadratic friction cone (see the gray cone in Figure 6(a)) and second, when the bodies are slipping, the friction force must be the one that directly opposes sliding. The cone is defined as follows:

$$\mathcal{F}(\mathbf{f}_n,\mu) = \{\mu^2\mathbf{f}_n^2 - \mathbf{f}_t^2 - \mathbf{f}_o^2 \geq 0, \mathbf{f}_n \geq 0\} \quad (4)$$

where $\mu \geq 0$ is the friction coefficient. The friction force that maximizes friction dissipation is:

$$\mathbf{f}_t = -\mu\mathbf{f}_n\frac{\mathbf{v}_t}{\beta} \quad (5)$$

$$\mathbf{f}_o = -\mu\mathbf{f}_n\frac{\mathbf{v}_o}{\beta} \quad (6)$$

where $\beta = \sqrt{\mathbf{v}_t^2 + \mathbf{v}_o^2}$ is the sliding speed at the contact (see Figure 6(b)).

Common variations on this model include using two different friction coefficient; one for sticking contact and a lower one for sliding. When friction forces are higher in one direction than another, one can replace the circular cone with an elliptical cone. In some simulation schemes the nonlinearity of the friction cone causes problems, and so it is eliminated by approximating the cone as a symmetric polyhedral cone. Finally, to model the fact that contacts between real bodies are actually small patches, the friction cone can be extended, as done by Contensou, to allow for a friction moment that resists rotation about the contact normal [Con93, TTP01].

A similar model for dry friction acting to resist joint motion will be discussed in section 3.

2.1.5. The Newton-Euler Equations

The Newton-Euler equations are obtained by applying Newton's second law twice; once for translational motion and again for rotational motion. Specifically, the net force $\mathbf{F}$ applied to the body is equal to the time rate of change of translational momentum $m\mathbf{v}$ (*i.e.*, $\frac{d}{dt}(m\mathbf{v}) = \mathbf{F}$) and the net moment τ is equal to the time rate of change of rotational momentum $\mathbf{I}\omega$ (*i.e.*, $\frac{d}{dt}(\mathbf{I}\omega) = \tau$). Specializing these equations to the case of a rigid body (which, by definition, has constant mass) yields:

$$m\dot{\mathbf{v}} = \mathbf{F} \quad (7)$$

$$\mathbf{I}\dot{\omega} + \omega \times \mathbf{I}\omega = \tau. \quad (8)$$

where recall that $\mathbf{I}$ is the 3-by-3 inertia matrix and $\times$ represents the vector cross product.

The second term on the left side of the rotational equation is called the "gyroscopic force" which arises from the proper differentiation of the rotational momentum. The rotational velocity and mass matrix must both be expressed in the same frame, which is usually taken as a body-fixed frame (which is rotating with the body in the inertial frame) or the inertial frame. In a body-fixed frame, $\mathbf{I}_{body}$ is constant, but ω is a vector expressed in a rotating frame, which means that $\mathbf{I}\omega$ is also a vector expressed in a rotating frame. The first term represents the rate of increase of angular velocity along the vector ω.

One might be tempted to try to eliminate the second term by expressing the rotational quantities in the inertial frame and differentiating them there. However this does not work, because the inertia matrix expressed in the inertial frame $\mathbf{I}$ is time-varying, as seen by the following identity $\mathbf{I} = \mathbf{R}\mathbf{I}_{\text{body}}\mathbf{R}^T$. Differentiating inertial frame quantities yields an equivalent expression with equivalent complexity.

The Newton-Euler equations contain the net force $\mathbf{F}$ and moment τ. $\mathbf{F}$ is simply the vector sum of all forces acting on the body. τ is the vector sum of the moments of all the forces and pure moments. One can see from equation (7), that the net force causes the center of gravity to accelerate in the direction of the net force proportional to its magnitude. This is true independent of the location of the line of action in space. Equation (8) implies that the net moment directly affects the rotational velocity of the body, but in a more complicated way. The gyroscopic moments tend to cause the axis of rotation of a rotating rigid body to "precess" about a circular cone.

Simulation of free body motion is done by integrating the Newton-Euler equations (7,8) and the velocity kinematic equation (1) simultaneously. If there are contacts and joints, then these equations must be augmented with the constraint equations (2,3). If in addition, dry friction exists in contacts, then equations (4,5,6) must be included. The complete system of differential and algebraic equations and inequalities is challenging to integrate, but methods to do this robustly have been developed over the past 20 years. To push the boundaries of interactive rigid body dynamics, one must maintain the current level of solution robustness and greatly increase the solution speed.

2.1.6. Impulse

When a pair of bodies collides, those bodies, and any other bodies they are touching, experience very high forces of very

short duration. In the case of ideal rigid bodies, the force magnitudes become infinite and the duration becomes infinitesimal. These forces are referred to as *impulsive forces* or *shocks*. One can see from equation (7), that shocks cause infinite accelerations, which makes direct numerical integration of the Newton-Euler equations impossible. One way to deal with this problem during simulation is to use a standard integration method up to the time of impact, then use an impulse-momentum law to determine the jump discontinuities in the velocities, and finally restart the integrator.

Let $[t, t+\Delta t]$ be a time step during which a collision occurs. Further, define $\mathbf{p} = \int_t^{t+\Delta t} \mathbf{F} dt$ as the impulse of the net force and $m\mathbf{v}$ as translational momentum. Integrating equation (7) from t to $t+\Delta t$ yields $m(\mathbf{v}(t+\Delta t) - \mathbf{v}(t)) = \int_t^{t+\Delta t} \mathbf{F} dt$, which states that impulse of the net applied force equals the change of translational momentum of the body. In rigid body collisions, Δt approaches zero. Taking the limit as Δt goes to zero, one obtains an impulse momentum law that is applied at the instant of impact to compute post collision velocities. Since Δt goes to zero and the velocities remain finite, the generalized position of the bodies are fixed during the impact. After processing the collision, one has the values of the generalized positions and velocities, which are the needed initial conditions to restart the integrator. Note that integration of the rotational equation (8) yields an impulse-momentum law for determining jump discontinuities in the rotational velocities.

Based on impulse-momentum laws, several algebraic collision rules have been proposed. Newton's Hypothesis is stated in terms of the normal component of the relative velocity of the colliding points just before and just after collision: $\mathbf{v}_n^+ = -\varepsilon \mathbf{v}_n^-$, where $\mathbf{v}_n^-$ is relative normal velocity just before impact, $\mathbf{v}_n^+$ is the relative normal velocity just after impact, and $\varepsilon \in [0,1]$ is known as the coefficient of restitution. Setting ε to zero yields a perfectly plastic impact (*i.e.*, an impact with no bounce). Setting this value to 1 yields perfectly elastic impacts (*i.e.*, no energy is lost).

Poisson's Hypothesis is similar, but is a function of collision impulse rather than the rate of approach. The normal impulse is divided into two parts, $\mathbf{p}_n^c$ and $\mathbf{p}_n^r$, which are related as follows $\mathbf{p}_n^r = \varepsilon \mathbf{p}_n^c$, where again $\varepsilon \in [0,1]$. Immediately prior to the collision, v_n^- of the impact points is negative. The compression impulse $\mathbf{p}_n^c$ is defined as the amount of impulse required to cause the relative normal velocity to become zero - just enough to prevent body interpenetration with no bounce. The restitution impulse is applied after the compression impulse to generate bounce (*i.e.*, $\mathrm{v}_n^- > 0$).

The same idea can be applied to frictional collision impulses by replacing the normal components of the impulses and velocities with the tangential components (see for example [Bra91]). The normal and tangential impact hypotheses can be used together to determine the velocity jumps caused by impacts. While simple and intuitive, this approach can unfortunately generate energy during oblique collisions. To prevent such unrealistic outcomes, Stronge developed an energy-based collision law that imposes a condition that prevents energy generation. Chatterjee and Ruina incorporated Stronge's energy constraint and recast the collision law in terms of two parameters that are physically meaningful [CR98].

3. Models for Interactive Simulation

The laws of physics must be combined into what we term an instantaneous-time model, which describes the continuous motions of the rigid bodies. Following this, we discretize this model over the time domain to obtain a discrete-time model, which is a sequence of so-called time-stepping subproblems. The subproblems are formulated and numerically solved at every time step to simulate the system.

In this section, we present generic models for systems with multiple simultaneous frictional contacts in Section 3.1. The particulars of models for dealing with aspects of reduced coordinate formulations are covered in Section 3.2.

3.1. Modeling of Simultaneous Frictional Contacts

Here we take a strict approach trying to keep the physics as correct as possible by only introducing errors of linearization and discretization. The model consists of five parts: the Newton-Euler equation [Lan86], a kinematic map (to relate time derivatives of configuration parameters to translational and angular velocity variables), equality constraints (to model permanent joint connections), normal contact conditions (to model intermittent contact behavior), and a dry friction law satisfying the principle of maximum power dissipation, also known as the principle of maximum work [Goy89]. These five parts will be explained in detail below.

Two types of constraints exist: permanent mechanical joints, each represented by a system of equations (five scalar equations in the case of a one-degree-of-freedom joint), and isolated point contacts with well-defined contact normals, each represented by one scalar inequality constraint. Let $\mathcal{B}$ and $\mathcal{U}$ denote the mutually exclusive sets of bilateral (equality) and unilateral (inequality) contacts:

$$\mathcal{B} = \{i : \text{contact } i \text{ is a joint}\} \tag{9}$$

$$\mathcal{U} = \{i : \text{contact } i \text{ is a point contact}\} \tag{10}$$

where $\mathcal{B} \cup \mathcal{U} = \{1, ..., n_c\}$ and n_c is the number of contacts. Note that distributed contacts can be approximated arbitrarily well by a number of isolated point contacts.

To formulate the equations of motion properly, one needs precise definitions of contact maintenance, sliding, and rolling. It is convenient to partition possible relative motions at each contact into *normal* and *frictional* subspaces. Let ${}^{\kappa}\mathbf{C}_{in}$ and ${}^{\kappa}\mathbf{C}_{if}$, where $\kappa \in \{b, u\}$, denote signed distance

functions (or gap functions) in the normal and friction subspace directions at contact i. If two bodies touch at contact i, then ${}^{\kappa}\mathbf{C}_{in} = 0$. This is always enforced for joints (${}^{b}\mathbf{C}_{in} = 0$), which are permanent contacts, but not for unilateral contacts, which can be broken as bodies separate (${}^{u}\mathbf{C}_{in} > 0$).

The first time derivatives of the distance functions are the relative contact velocities, ${}^{\kappa}\mathrm{v}_{i\sigma} = \frac{d}{dt}\left({}^{\kappa}\mathbf{C}_{i\sigma}\right)$; $\kappa \in \{b,\, u\}, \sigma \in \{n,\, f\}$. Note that ${}^{\kappa}\mathrm{v}_{in}$ and ${}^{\kappa}\mathrm{v}_{if}$ are orthogonal subspaces, where unallowed motions are prevented by body structures and sliding motions are resisted by friction forces, respectively. If a pair of contact points (one on each body at the point of touching) are in rolling contact, instantaneously, the distance between those bodies in the direction of possible sliding is zero (${}^{\kappa}\mathrm{v}_{if} = 0$). If they slip, at least one friction direction displacement will become nonzero. For example, the friction direction of a one-degree-of-freedom joint is in the direction of motion of the joint. For a unilateral contact with isotropic Coulomb friction, the friction subspace will consist of relative translation in the t- and o-directions. The corresponding displacement functions will be denoted by ${}^{u}\mathbf{C}_{it}$ and ${}^{u}\mathbf{C}_{io}$. Relative rotations are not resisted by body structure or friction, so they are not included in either subspace.

We now partition all contacts into sliding and rolling subsets. At the position level, contact i is sustained if the distance function ${}^{\kappa}C_{in}(\mathbf{q},t)$; $\kappa \in \{b,u\}$ is equal to zero for a finite period of time. However, one cannot distinguish sliding from rolling with this position-level condition; one needs time derivatives. The velocity-level set definitions are:

$$\mathcal{S} \;=\; \{i : {}^{\kappa}\mathbf{C}_{in} = 0,\ {}^{\kappa}\mathrm{v}_{in} = 0,\ {}^{\kappa}\mathrm{v}_{if} \neq 0\} \tag{11}$$

$$\mathcal{R} \;=\; \{i : {}^{\kappa}\mathbf{C}_{in} = 0,\ {}^{\kappa}\mathrm{v}_{in} = 0,\ {}^{\kappa}\mathrm{v}_{if} = 0\}, \tag{12}$$

where the sets $\mathcal{S}$ and $\mathcal{R}$ are mutually exclusive.

We are now in a position to develop the system of equations and inequalities defining the instantaneous-time dynamic model of a multi-rigid-body system with bilateral and unilateral contacts. Recall the five parts mentioned above.

Newton-Euler Equations: The Newton-Euler equation can be written as follows:

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} = \mathbf{g}(\mathbf{q},\mathbf{u},t), \tag{13}$$

where $\mathbf{M}(\mathbf{q})$ is the generalized mass matrix containing the body mass properties and $\mathbf{g}(\mathbf{q},\mathbf{u},t)$ is the vector of loads, including the gyroscopic moment (the cross-product term in equation (15)). Specifically for the j^{th} rigid body we have:

$$\mathbf{M}_j = \begin{bmatrix} m_j\mathbf{1}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_j(Q_j) \end{bmatrix}, \tag{14}$$

$$\mathbf{g}_j = \begin{bmatrix} \mathbf{F}_j \\ \tau_j - \omega_j \times \mathbf{I}_j(Q_j)\omega_j \end{bmatrix}, \tag{15}$$

where $\mathbf{1}_{3\times 3}$ is the 3-by-3 identity matrix. Recall that $\mathbf{I}$ is the 3-by-3 inertia matrix, and $\mathbf{F}_j$ and τ_j are the externally applied force and moment. Also note that $\mathbf{M}$ is positive definite and symmetric.

Kinematic Map: The time rate of change of the generalized coordinates of the bodies $\dot{\mathbf{q}}$ is related to the generalized velocities of the bodies $\mathbf{u}$:

$$\dot{\mathbf{q}} = \mathbf{H}(\mathbf{q})\mathbf{u}. \tag{16}$$

where $\mathbf{H}(\mathbf{q})$ is the generalized kinematic map. The diagonal blocks $\mathbf{H}_{jj}$ are given by equation 1 and off diagonal blocks are zero.

Joint Constraints: Since joints are permanent contacts, if contact i is a joint (*i.e.,* $i \in \mathcal{B}$), then the vector function ${}^{b}\mathbf{C}_{in}(\mathbf{q},t) = \mathbf{0}$ for all time. Stacking the ${}^{b}\mathbf{C}_{in}$ functions for all $i \in \mathcal{B}$ into the vector ${}^{b}\mathbf{C}_{n}(\mathbf{q},t)$, yields the position-level constraint for all joints:

$${}^{b}\mathbf{C}_{n}(\mathbf{q},t) = \mathbf{0}. \tag{17}$$

From a physical perspective, these constraints are maintained by reaction forces ${}^{b}\mathbf{f}_{in}$ that are unconstrained. That is, generalized forces normal or anti-normal to the constraint surface in the system's configuration space can be generated. When viewing multibody dynamics from a variational perspective, these forces are Lagrange multipliers [Lan86].

Normal Contact Constraints: For the unilateral contacts, the scalar functions, ${}^{u}\mathbf{C}_{in}(\mathbf{q},t)$ for all $i \in \mathcal{U}$ must be non-negative. Stacking all the gap functions into the vector ${}^{u}\mathbf{C}_{n}(\mathbf{q},t)$ yields the following position-level non-penetration constraint:

$${}^{u}\mathbf{C}_{n}(\mathbf{q},t) \geq \mathbf{0}. \tag{18}$$

From a physical perspective, this constraint is maintained by the normal component of the contact force ${}^{u}\mathbf{f}_{in}$ between the bodies. Again, this force can be viewed as a Lagrange multiplier, but since the constraint is one-sided, so is the multiplier (*i.e.,* ${}^{u}\mathbf{f}_{in} \geq 0$). This means that constraint forces at unilateral contacts must be compressive or zero. Combining all ${}^{u}\mathbf{f}_{in}$ for all $i \in \mathcal{U}$ into the vector ${}^{u}\mathbf{f}_{n}$, we write all normal force constraints as:

$${}^{u}\mathbf{f}_{n} \geq \mathbf{0}. \tag{19}$$

There is one more aspect of unilateral contacts that must be modeled. If contact i is supporting a load (*i.e.,* ${}^{u}\mathbf{f}_{in} > 0$), then the contact must be maintained (*i.e.,* ${}^{u}\mathbf{C}_{in} = 0$). Conversely, if the contact breaks (*i.e.,* ${}^{u}\mathbf{C}_{in} > 0$), then the normal components (and hence the frictional components) of the contact force must be zero (*i.e.,* ${}^{u}\mathbf{f}_{in} = 0$). For each contact, at least one of ${}^{u}\mathbf{f}_{in}$ and ${}^{u}\mathbf{C}_{in}$ must be zero, (*i.e.,* ${}^{u}\mathbf{C}_{in}\ {}^{u}\mathbf{f}_{in} = 0$). These conditions are imposed at every contact simultaneously by an orthogonality constraint:

$${}^{u}\mathbf{C}_{n}(\mathbf{q},t) \;\cdot\; {}^{u}\mathbf{f}_{n} = 0 \tag{20}$$

where $\cdot$ denotes the vector dot product.

Friction Law: At contact i, the generalized friction force ${}^{\kappa}\mathbf{f}_{if}$ can act only in a subset of the unconstrained directions and must lie within a closed convex limit set $\mathcal{F}_i({}^{\kappa}\mathbf{f}_{in}, \mu_i)$. The

limit set must contain the origin, so that a zero friction force is possible. Also, typically, the limit set scales linearly with the normal component of the contact force, thus forming a cone of possible contact forces.

When contact i is rolling, the friction force may take on any value within the limit set. However, when the contact is sliding, the friction force must be the one within $\mathcal{F}_i({}^{\kappa}\mathbf{f}_{in}, \mu_i)$ that maximizes the power dissipation. Such models are said to satisfy the principle of maximum dissipation [Goy89]. At the velocity level, maximum dissipation can be expressed as follows:

$$ {}^{\kappa}\mathbf{f}_{if} \in \arg\max_{\mathbf{f}'_{if}} \left\{ -{}^{\kappa}\mathbf{v}_{if} \cdot \mathbf{f}'_{if} : \mathbf{f}'_{if} \in \mathcal{F}_i({}^{\kappa}\mathbf{f}_{in}, \mu_i) \right\} \qquad (21) $$

where $\mathbf{f}'_{if}$ is an arbitrary vector in the set $\mathcal{F}_i({}^{\kappa}\mathbf{f}_{in}, \mu_i)$. Notice that when this set is strictly convex, then the friction force will be unique. For example, under the assumption of isotropic Coulomb friction at a unilateral contact, the limit set is the disc $\mu_i^2 {}^{u}\mathbf{f}_{in}^2 - {}^{u}\mathbf{f}_{it}^2 - {}^{u}\mathbf{f}_{io}^2 \geq 0$ and the unique friction force is the one directly opposite the relative sliding velocity, $({}^{u}\mathbf{v}_{it},\ {}^{u}\mathbf{v}_{io})$.

Finally, our instantaneous-time dynamic model is the system of differential algebraic inequalities (DAIs) composed of equations (13,16–21), where the sliding and rolling contact sets are defined as in equations (11) and (12) In the current form, the DAI is difficult to solve. However, as will be shown, it is possible to cast the model as a differential complementarity problem [CPS92b,TPSL97], then discretize the result to form simulation subproblems in the form of nonlinear or linear complementarity problems, allowing one to apply well-studied solution algorithms.

Complementarity Problems

The standard nonlinear complementarity problem (NCP) can be stated as follows:

Definition 1 Nonlinear Complementarity Problem: Given an unknown vector $\mathbf{x} \in \mathbb{R}^m$ and a known vector function $\mathbf{y}(\mathbf{x}) : \mathbb{R}^m \to \mathbb{R}^m$, determine $\mathbf{x}$ such that:

$$ \mathbf{0} \leq \mathbf{y}(\mathbf{x}) \perp \mathbf{x} \geq \mathbf{0}, \qquad (22) $$

where $\perp$ implies orthogonality (*i.e.*, $\mathbf{y}(\mathbf{x}) \cdot \mathbf{x} = 0$).

The standard linear complementarity problem (LCP) is a special case in which the function $\mathbf{y}(\mathbf{x})$ is linear in $\mathbf{x}$:

Definition 2 Linear Complementarity Problem: Given an unknown vector $\mathbf{x} \in \mathbb{R}^m$, a known fixed matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$, and a known fixed vector $\mathbf{b} \in \mathbb{R}^m$, determine $\mathbf{x}$ such that:

$$ \mathbf{0} \leq \mathbf{A}\mathbf{x} + \mathbf{b} \perp \mathbf{x} \geq \mathbf{0}. \qquad (23) $$

We adopt the shorthand notation, LCP($\mathbf{A}, \mathbf{b}$).

3.1.1. Complementarity Formulation of the Instantaneous-Time Model

To achieve model formulation as a properly posed complementarity problem, we must write all the conditions (13,16–21) in terms of a common small set of dependent variables. In the current formulation, the dependent variables are positions, velocities, and accelerations. However, by taking the appropriate number of time derivatives, all equations will be written in terms of accelerations, thus generating a model in which all dependent variables are forces and accelerations. This transformation will be carried out below in three steps. First, express the principle of maximum dissipation as a system of equations and inequalities in forces and accelerations, second, reformulate the Newton-Euler equation to expose the forces, and third, differentiate the distance functions twice with respect to time to expose the accelerations.

Reformulation of maximum dissipation The principle of maximum dissipation (21) can be replaced by an equivalent system of equations and inequalities by formulating it as an unconstrained optimization problem, and solving it in closed form. To do this, however, one must choose a specific form of $\mathcal{F}_i$. In this paper, we will demonstrate the solution process for isotropic Coulomb friction at a unilateral contact and apply the result to dry friction of constant maximum magnitude in a one-degree-of-freedom joint. The same procedure can be applied to other friction models, including Contensou [TP97, TTP01].

Closed-form solutions of optimization problems can sometimes be found by obtaining a system of equations corresponding to necessary and sufficient conditions for an optimal solution, then solving them. The most common approach is to augment the objective function with the constraints multiplied by Lagrange multipliers and then obtain the equations, known as the Karush-Kuhn-Tucker (KKT) equations, by partial differentiation. To be valid, the system must satisfy a regularity condition (also known as, "constraint qualification"). In the case of isotropic Coulomb friction, the system does not satisfy any of the possible regularity conditions at the point of the cone (where ${}^{u}\mathbf{f}_{in} = 0$), so the method fails.

Fortunately, the more general Fritz-John conditions [MF67] *do* satisfy a regularity condition everywhere on the cone. In the case of isotropic Coulomb friction, the augmented objective function (recall equation (21) is:

$$ -{}^{u}\beta_{i0}({}^{u}\mathbf{f}_{it}\,{}^{u}\mathbf{v}_{it} + {}^{u}\mathbf{f}_{io}\,{}^{u}\mathbf{v}_{io}) + {}^{u}\beta_i(\mu_i^2 {}^{u}\mathbf{f}_{in}^2 - {}^{u}\mathbf{f}_{it}^2 - {}^{u}\mathbf{f}_{io}^2), \quad (24) $$

where ${}^{u}\beta_i$ and ${}^{u}\beta_{i0}$ are Lagrange multipliers. To obtain a system of equations and inequalities equivalent to the maximum dissipation condition (21), one takes partial derivatives with respect to the unknown friction force components and Lagrange multipliers and then imposes the additional conditions of the Fritz-John method: ${}^{u}\beta_{i0} \geq 0$ and $({}^{u}\beta_{i0}, {}^{u}\beta_i) \neq (0,0)$. Following the derivation on pages 28-30 of [Ber09],

one arrives at the following system of constraints:

$$\left.\begin{aligned} \mu_i\, {}^u\mathbf{f}_{in}\, {}^u\nu_{it} + {}^u\mathbf{f}_{it}\, {}^u\beta_i = 0 \\ \mu_i\, {}^u\mathbf{f}_{in}\, {}^u\nu_{io} + {}^u\mathbf{f}_{io}\, {}^u\beta_i = 0 \\ {}^u\alpha_i = \mu_i^2\, {}^u\mathbf{f}_{in}^2 - {}^u\mathbf{f}_{it}^2 - {}^u\mathbf{f}_{io}^2 \geq 0 \\ 0 \leq {}^u\alpha_i \perp {}^u\beta_i \geq 0 \end{aligned}\right\} \quad \forall\, i \in \{\mathcal{U} \cap \mathcal{S}\}, \quad (25)$$

where ${}^u\alpha_i$ is a slack variable for the friction limit set. Note that ${}^u\beta_i = \| {}^u\nu_{if} \|$ at the optimal solution, and represents the magnitude of the slip velocity at contact i (*i.e.*, ${}^u\beta_i = \| {}^u\nu_{if} \| = \sqrt{{}^u\nu_{it}^2 + {}^u\nu_{io}^2}$). Note that this condition is *not* written in terms of accelerations, because at a sliding contact, the friction force $({}^u\mathbf{f}_{it},\, {}^u\mathbf{f}_{io})$ can be written in terms of the normal force and eliminated. For example, in the case of Coulomb friction, equations (5) and (6) are used. Tangential acceleration at a contact does not enter unless the contact point is rolling.

If contact i is a one-degree-of-freedom joint, we will assume that the maximum magnitude of the dry friction force is independent of the load in the other five component directions. Thus, the friction limit set for a bilateral joint $\mathcal{F}_i(\mu_i)$ will be:

$$\mathcal{F}_i({}^b\mathbf{f}_{if_{\max}}) = \left\{ {}^b\mathbf{f}_{if} : \left| {}^b\mathbf{f}_{if} \right| \leq {}^b\mathbf{f}_{if_{\max}} \right\}, \quad \forall\, i \in \{\mathcal{B} \cap \mathcal{S}\} \quad (26)$$

where $|\cdot|$ denotes the absolute value of a scalar and ${}^b\mathbf{f}_{if_{\max}}$ is the nonnegative maximum magnitude of the generalized friction force in joint i.

Notice that this joint friction model is a special case of the result obtained for Coulomb friction; fix ${}^u\mathbf{f}_{in}\mu_i$ to the value of ${}^b\mathbf{f}_{if_{\max}}$ and remove one of the friction directions, say the t-direction. The result is:

$$\left.\begin{aligned} {}^b\mathbf{f}_{io_{\max}}\, {}^b\nu_{io} + {}^b\mathbf{f}_{io}\, {}^b\beta_i = 0 \\ {}^b\alpha_i = {}^b\mathbf{f}_{io_{\max}}^2 - {}^b\mathbf{f}_{io}^2 \geq 0 \\ 0 \leq {}^b\alpha_i \perp {}^b\beta_i \geq 0 \end{aligned}\right\} \quad \forall\, i \in \mathcal{B}. \quad (27)$$

As before, ${}^b\beta_i = \| {}^b\nu_{if} \|$ at an optimal solution.

Contact Constraints in Terms of Accelerations Contact constraints (unilateral and bilateral) can be written in terms of accelerations through Taylor series expansion of constraint functions, ${}^\kappa C_{i\sigma}(q,t)$; $\kappa \in \{b,\, u\}$; $\sigma \in \{n,\, f\}$ Let $\tilde{q} = q + \Delta q$ and $\tilde{t} = t + \Delta t$ where Δq and Δt are small perturbations.

Then the Taylor expansion of one of these scalar displacement functions truncated to the quadratic terms is:

$$\begin{aligned} \widehat{{}^\kappa C}_{i\sigma}(\tilde{\mathbf{q}}, \tilde{t}) &= {}^\kappa C_{i\sigma}(\mathbf{q}, t) \\ &+ \frac{\partial {}^\kappa C_{i\sigma}}{\partial \mathbf{q}} \Delta\mathbf{q} + \frac{\partial {}^\kappa C_{i\sigma}}{\partial t} \Delta t \\ &+ \frac{1}{2} \left((\Delta\mathbf{q})^T \frac{\partial^2 {}^\kappa C_{i\sigma}}{\partial \mathbf{q}^2} \Delta\mathbf{q} + 2 \frac{\partial^2 {}^\kappa C_{i\sigma}}{\partial \mathbf{q} \partial t} \Delta\mathbf{q} \Delta t + \frac{\partial^2 {}^\kappa C_{i\sigma}}{\partial t^2} \Delta t^2 \right) \end{aligned}$$

Notice that if contact exists at the current values of $\mathbf{q}$ and t, then the first term is zero. Dividing the linear terms by Δt and taking the limit as Δt (and $\Delta\mathbf{q}$) goes to zero, one obtains the relative velocity, ${}^\kappa\nu_{i\sigma}$ at the contact. Dividing the quadratic terms by $(\Delta t)^2$ and taking the limit yields the relative acceleration ${}^\kappa\mathbf{a}_{i\sigma}$:

$${}^\kappa\mathbf{a}_{i\sigma} = {}^\kappa\mathbf{J}_{i\sigma}\dot{\mathbf{u}} + {}^\kappa\mathbf{k}_{i\sigma}(\mathbf{q}, \mathbf{u}, t) \quad (28)$$

where

$${}^\kappa\mathbf{J}_{i\sigma} = \frac{\partial({}^\kappa\mathbf{C}_{i\sigma})}{\partial \mathbf{q}} \mathbf{H}$$

$${}^\kappa\mathbf{k}_{i\sigma}(\mathbf{q}, \mathbf{u}, t) = \frac{\partial({}^\kappa\mathbf{C}_{i\sigma})}{\partial \mathbf{q}} \frac{\partial \mathbf{H}}{\partial t}\mathbf{u} + \frac{\partial^2({}^\kappa\mathbf{C}_{i\sigma})}{\partial \mathbf{q} \partial t}\mathbf{H}\mathbf{u} + \frac{\partial^2({}^\kappa\mathbf{C}_{i\sigma})}{\partial t^2},$$

where recall that κ is either b or u and σ is either n or f.

Stacking all the quantities above for every unilateral and bilateral contact (as defined in equations (11) and (12)), one arrives at the definitions of ${}^\kappa\mathbf{a}_n$, ${}^\kappa\mathbf{J}_n$, and ${}^\kappa\mathbf{k}_n$, which allows us to express equations (17-20) in terms of accelerations as follows:

$${}^b\mathbf{a}_n = \mathbf{0}. \quad (30)$$

$$\mathbf{0} \leq {}^u\mathbf{f}_n \perp {}^u\mathbf{a}_n \geq \mathbf{0}. \quad (31)$$

The principle of maximum dissipation (21) must be considered further. When contact i is sliding, the solutions of conditions (25) and (27) produce the correct results (*i.e.,* the friction force obtains its maximum magnitude and directly opposes the sliding direction) and, we can use these conditions to eliminate ${}^\kappa\mathbf{f}_{if}$. Also as required, when a contact is rolling, these conditions allow the friction force to lie anywhere within the friction limit set. What these conditions do not provide is a mechanism for determining if a rolling contact will change to sliding. However, this problem is easily remedied by replacing the relative velocity variables in equation (21) with the analogous acceleration variables.

$$\left.\begin{aligned} \mu_i\, {}^u\mathbf{f}_{in}\, {}^u\mathbf{a}_{it} + {}^u\mathbf{f}_{it}\, {}^u\beta_i = 0 \\ \mu_i\, {}^u\mathbf{f}_{in}\, {}^u\mathbf{a}_{io} + {}^u\mathbf{f}_{io}\, {}^u\beta_i = 0 \\ {}^u\beta_i = \mu_i^2\, {}^u\mathbf{f}_{in}^2 - {}^u\mathbf{f}_{it}^2 - {}^u\mathbf{f}_{io}^2 \geq 0 \\ 0 \leq {}^u\beta_i \perp {}^u\beta_i \geq 0 \end{aligned}\right\} \quad \forall\, i \in \mathcal{U} \cap \mathcal{R} \quad (32)$$

where ${}^u\beta_i = \| {}^u\mathbf{a}_{if} \|$ at the optimal solution, and

$$\left.\begin{aligned} {}^b\mathbf{f}_{if_{\max}}\, {}^b\mathbf{a}_{if} + {}^b\mathbf{f}_{if}\, {}^b\beta_i = 0 \\ {}^b\beta_i = {}^b\mathbf{f}_{if_{\max}}^2 - {}^b\mathbf{f}_{if}^2 \geq 0 \\ 0 \leq {}^b\beta_i \perp {}^b\beta_i \geq 0 \end{aligned}\right\} \quad \forall\, i \in \mathcal{B} \cap \mathcal{R} \quad (33)$$

where ${}^b\mathbf{f}_{if} = \left| {}^b\mathbf{a}_{if} \right|$ at the optimal solution.

Exposing the Contact Forces in the Newton-Euler Equation Recall that the vector $\mathbf{g}(\mathbf{q}, \mathbf{u}, t)$ represents the resultant generalized forces acting on the bodies, and naturally generated gyroscopic forces. In order to complete the formulation as an NCP, $\mathbf{g}(\mathbf{q}, \mathbf{u}, t)$ is expressed as the sum of the normal

and friction forces at the unilateral and bilateral contacts and all other generalized forces. The Newton-Euler equation becomes:

$$\begin{aligned}\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} = {}^{u}\mathbf{J}_n(\mathbf{q})^T {}^{u}\mathbf{f}_n + {}^{u}\mathbf{J}_f(\mathbf{q})^T {}^{u}\mathbf{f}_f \\ + {}^{b}\mathbf{J}_n(\mathbf{q})^T {}^{b}\mathbf{f}_n + {}^{b}\mathbf{J}_f(\mathbf{q})^T {}^{b}\mathbf{f}_f + \mathbf{g}_{\text{ext}}(\mathbf{q},\mathbf{u},t)\end{aligned} \tag{34}$$

where $\mathbf{g}_{\text{ext}}(\mathbf{q},\mathbf{u},t)$ is the resultant of all non-contact forces and moments applied to the bodies, ${}^{u}\mathbf{f}_f$ and ${}^{b}\mathbf{f}_f$ are formed by stacking the generalized friction vectors at the unilateral and bilateral contacts respectively, and the matrices ${}^{\kappa}\mathbf{J}_\sigma$ map contact forces into a common inertial frame.

3.1.2. A Differential NCP

The instantaneous dynamic model is now complete.

Definition 3 CP1: Equations (16,25,27,30–34) constitute a differential, nonlinear complementarity problem.

It is known that solutions to CP1 do not always exist (see [TPSL97]), however, if the principle of maximum dissipation is relaxed so that friction forces merely need to be dissipative, rather than maximally dissipative, then a solution always exists [PT96]. If one wanted to use this NCP in an integration scheme to simulate the motion of a multibody system, the PATH algorithm by Ferris and Munson is the most robust, general purpose NCP solver available [CPN11]. One should also note that this NCP can be converted into an approximate LCP by linearizing the friction cone constraint (see [TPSL97] for details). The LCP then can be solved by Lemke's algorithm [CPS92b], but the solution non-existence problem persists unless the maximum dissipation requirement is relaxed as just stated above.

3.1.3. A Nonlinear Discrete-Time Model

We recommend against applying a integration method to the instantaneous-time model due the possible non-existence of computable solutions (*i.e.,* a values of the unknown accelerations and constraint forces might not exist *OR* the solution might contain infinite values). However, this problem can be alleviated by applying discrete-time derivative approximations to the model over a time step Δt and then recasting the model as a time-stepping complementarity problem whose unknowns are impulses (time integrals of forces) and velocities (time integrals of accelerations). We demonstrate the process in this section.

Let Δt denote a positive step size and t_ℓ the current time, for which we have estimates of the configuration $\mathbf{q}^{(\ell)} = \mathbf{q}(t_\ell)$ and the generalized velocity $\mathbf{u}^{(\ell)} = \mathbf{u}(t_\ell)$ of the system. Our goal is to compute configurations $\mathbf{q}^{(\ell+1)} = \mathbf{q}(t_\ell + \Delta t)$ and velocities $\mathbf{u}^{(\ell+1)} = \mathbf{u}(t_\ell + \Delta t)$ that lie as close as possible to a solution of the differential NCP, CP1.

In the derivation, we will require estimates of the matrices $\mathbf{M}$, ${}^{\kappa}\mathbf{J}_\sigma$, and the vector $\mathbf{g}_{\text{ext}}$, which all vary with the system configuration $\mathbf{q}$. As will be seen, the simplest choice will be to use their values at $\mathbf{q}^{(\ell)}$, denoted by $\mathbf{M}^{(\ell)}$, ${}^{\kappa}\mathbf{J}_\sigma^{(\ell)}$, and $\mathbf{g}_{\text{ext}}^{(\ell)}$, because that will result in an explicit time-stepping method with each step requiring the solution of an NCP, whose only nonlinearity is the quadratic constraint of the friction cone. Approximating the cone constraint with a polyhedral cone will replace the quadratic constraints with linear ones, and thus will convert the NCP into an LCP, which is typically easier to solve.

In the following section, we discretize the five components of the instantaneous-time model derived above. To simplify our presentation, we choose the simple backward Euler approximation of the state derivatives, *i.e.,* $\dot{\mathbf{u}}(t_{\ell+1}) \approx (\mathbf{u}^{(\ell+1)} - \mathbf{u}^{(\ell)})/\Delta t$ and $\dot{\mathbf{q}}(t_{\ell+1}) \approx (\mathbf{q}^{(\ell+1)} - \mathbf{q}^{(\ell)})/\Delta t$.

Discrete-Time Newton-Euler Equations Applying the backward Euler approximation to the Newton-Euler equation (34) yields the equation below in which all quantities are evaluated at the end of the time step:

$$\begin{aligned}\mathbf{M}^{(\ell+1)}\left(\mathbf{u}^{(\ell+1)} - \mathbf{u}^{(\ell)}\right) = ({}^{u}\mathbf{J}_n^T)^{(\ell+1)} {}^{u}\mathbf{p}_n^{(\ell+1)} \\ + ({}^{u}\mathbf{J}_f^T)^{(\ell+1)} {}^{u}\mathbf{p}_f^{(\ell+1)} + ({}^{b}\mathbf{J}_n^T)^{(\ell+1)} {}^{b}\mathbf{p}_n^{(\ell+1)} \\ + ({}^{b}\mathbf{J}_f^T)^{(\ell+1)} {}^{b}\mathbf{p}_f^{(\ell+1)} + (\mathbf{p}_{\text{ext}})^{(\ell+1)},\end{aligned} \tag{35}$$

where the vectors are unknown generalized contact impulses defined as $\mathbf{p}^{(\ell+1)} = \Delta t \mathbf{f}^{(\ell+1)}$ and $\mathbf{p}_{\text{ext}} = \Delta t \mathbf{g}_{\text{ext}}$ is the impulse of the generalized forces applied to the bodies over the time step.

Discrete-Time Kinematic Map Applying the backward Euler approximation to the kinematic map (16) gives:

$$\mathbf{q}^{(\ell+1)} - \mathbf{q}^{(\ell)} \;=\; \Delta t \mathbf{H}^{(\ell+1)} \mathbf{u}^{(\ell+1)}, \tag{36}$$

which is nonlinear in the unknown system configuration $\mathbf{q}^{(\ell+1)}$. An important issue arises when solving this equation for $\mathbf{q}^{(\ell+1)}$. The "vector" $\mathbf{q}^{(\ell+1)}$, is *NOT* a vector; the orientation part of $\mathbf{q}$ lives in a curved space, not a vector space. For example, when orientation is represented by a unit quaternion, then quaternion elements of $\mathbf{q}^{(\ell)}$ and $\mathbf{q}^{(\ell+1)}$ must have unit length, but adding $\Delta t \mathbf{H}^{(\ell+1)} \mathbf{u}^{(\ell+1)}$ to $\mathbf{q}^{(\ell)}$ slightly increases the length. This problem can be solved simply by normalizing the quaternion elements of $\mathbf{q}^{(\ell+1)}$ after each time step.

Discrete-Time Contact Constraints The contact and joint constraints given in equations (17,18,20) can be incorporated into the discrete-time framework by replacing the configuration variables with their values at time $t_{(\ell+1)}$. An alternative is to expand the functions in a Taylor series and choose the point of truncation, to control the level of accuracy and nonlinearity. Denoting ${}^{\kappa}\mathbf{C}_\sigma\left(\mathbf{q}^{(\ell)}, t^{(\ell)}\right)$ by ${}^{\kappa}\mathbf{C}_\sigma^{(\ell)}$, a

Taylor series expansion is:

$$\widehat{{}^{\kappa}\mathbf{C}_{\sigma}^{(\ell+1)}} = {}^{\kappa}\mathbf{C}_{\sigma}^{(\ell)} + ({}^{\kappa}\mathbf{J}_{\sigma})^{(\ell)}\mathbf{u}^{(\ell+1)}\Delta t + \frac{\partial {}^{\kappa}\mathbf{C}_{\sigma}^{(\ell)}}{\partial t}\Delta t + H.O.T, \tag{37}$$

where the hat over the $\mathbf{C}$ in denotes an approximation and $H.O.T.$ denotes higher-order terms.

Discrete-Time Normal Contact Constraints Given that the discrete time Newton-Euler equations are written in terms of unknown impulses, equation (19) should also be converted to impulses. Since ${}^{u}\mathbf{p}_{n}^{(\ell+1)} = \Delta t\, {}^{u}\mathbf{f}_{n}^{(\ell+1)}$ and Δt is strictly positive, equation (19) becomes:

$${}^{u}\mathbf{p}_{n}^{(\ell+1)} \geq 0. \tag{38}$$

Last, for each unilateral contact, we must enforce complementarity between the gap function at the end of the time step and the normal impulse. Inserting the Taylor series approximation into equation (20) yields:

$$0 \leq {}^{u}\mathbf{p}_{n}^{(\ell+1)} \perp \widehat{{}^{u}\mathbf{C}_{n}^{(\ell+1)}} \geq 0. \tag{39}$$

Note that, in general, the right-hand inequality is nonlinear in the unknown $\mathbf{u}^{(\ell+1)}$. However, it can be made linear by truncating the Taylor series after the linear terms. It is also important to see that this relationship implies that the normal impulse ${}^{u}\mathbf{p}_{n}^{(\ell+1)}$ at the end of the time step can be nonzero only if the approximated distance function at the end of the time step is zero.

Discrete-Time Maximum Dissipation Principle To modify the maximum dissipation condition for use in time stepping, one integrates the force over a short time interval to obtain an impulse. If the direction of sliding changes little over the time step, then the friction law can be well approximated by simply replacing force variables with impulse variables. Thus, equation (21) becomes:

$$\begin{aligned} {}^{\kappa}\mathbf{p}_{if}^{(\ell+1)} \in \arg\max_{\mathbf{p}'_{if}} \Big\{ -\left({}^{\kappa}\mathbf{v}_{if}^{(\ell+1)}\right)^{T} \mathbf{p}'_{if} \;:\; \\ \mathbf{p}'_{if} \in \mathcal{F}_i({}^{\kappa}\mathbf{p}_{in}^{(\ell+1)}, \mu_i) \Big\} \end{aligned} \tag{40}$$

where $\mathbf{p}'_{if}$ is an arbitrary vector in the set $\mathcal{F}_i({}^{\kappa}\mathbf{p}_{in}^{(\ell+1)}, \mu_i)$. As it was the case during the formulation of the instantaneous model, we cannot complete the formulation of the discrete-time model without assuming a particular form of $\mathcal{F}_i$.

3.1.4. The Discrete-Time Model as an LCP

To develop a discrete-time model in the form of an LCP, all equations and inequalities in the model must be linear in the unknown configuration $\mathbf{q}^{(\ell+1)}$, velocity $\mathbf{u}^{(\ell+1)}$, and impulse $(\mathbf{p}_{\text{ext}})^{(\ell+1)}$. The discrete-time Newton Euler equation (35) appears to be linear in the unknown impulses and velocities, but the Jacobian matrices are functions of the unknown configuration and the external impulse $(\mathbf{p}_{\text{ext}})^{(\ell+1)}$ is generally a function of both $\mathbf{q}^{(\ell+1)}$ and $\mathbf{u}^{(\ell+1)}$. The standard way to obtain a linear equation is to evaluate the Jacobians and the external impulse at t_ℓ.

$$\begin{aligned} \mathbf{M}^{(\ell)}\left(\mathbf{u}^{(\ell+1)} - \mathbf{u}^{(\ell)}\right) = ({}^{u}\mathbf{J}_{n}^{T})^{(\ell)}\, {}^{u}\mathbf{p}_{n}^{(\ell+1)} \\ + ({}^{u}\mathbf{J}_{f}^{T})^{(\ell)}\, {}^{u}\mathbf{p}_{f}^{(\ell+1)} + ({}^{b}\mathbf{J}_{n}^{T})^{(\ell)}\, {}^{b}\mathbf{p}_{n}^{(\ell+1)} \\ + ({}^{b}\mathbf{J}_{f}^{T})^{(\ell)}\, {}^{b}\mathbf{p}_{f}^{(\ell+1)} + (\mathbf{p}_{\text{ext}})^{(\ell)}. \end{aligned} \tag{41}$$

Another option is to extrapolate these quantities forward in time using quantities known as time t_ℓ, as done in [PAGT05]. A drawback of this approach is that it requires computation of derivatives of the matrices.

Equation (36) is also nonlinear due to the dependence of $\mathbf{H}$ on $\mathbf{q}^{(\ell+1)}$. As above, evaluating it at t_ℓ yields a linear approximation:

$$\mathbf{q}^{(\ell+1)} - \mathbf{q}^{(\ell)} \;=\; \Delta t \mathbf{H}^{(\ell)}\mathbf{u}^{(\ell+1)}. \tag{42}$$

As mentioned earlier, the distance function constraints can be made linear by truncating the Taylor series after the linear terms. For the bilateral contacts, equation (37) becomes:

$${}^{b}\mathbf{C}_{n}^{(\ell)} + ({}^{b}\mathbf{J}_{n})^{(\ell)}\mathbf{u}^{(\ell+1)}\Delta t + \frac{\partial {}^{b}\mathbf{C}_{n}^{(\ell)}}{\partial t}\Delta t = 0. \tag{43}$$

Similarly, the normal complementarity condition (39) becomes:

$$0 \leq {}^{u}\mathbf{p}_{n}^{(\ell+1)} \perp {}^{u}\mathbf{C}_{n}^{(\ell)} + ({}^{u}\mathbf{J}_{n})^{(\ell)}\mathbf{u}^{(\ell+1)}\Delta t + \frac{\partial {}^{u}\mathbf{C}_{n}^{(\ell)}}{\partial t}\Delta t \geq 0. \tag{44}$$

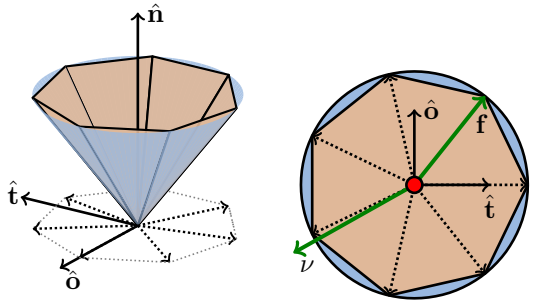
It only remains to linearize the friction model. To do so, however, requires a specific choice of friction limit set. Therefore, at this point, we will choose isotropic Coulomb friction and demonstrate the process of linearization for it, which is illustrated in Figure 7. The circular friction limit set is a circle of radius $\mu_i\, {}^{u}\mathbf{p}_{in}^{(\ell+1)}$ (shown in Figure 7(b)). This circle is approximated by a convex polygon whose vertices are defined by n_d the unit vectors $\hat{\mathbf{d}}_{ij}$ that positively span the friction plane.

To constrain the friction impulse at contact i, ${}^{u}\mathbf{p}_{if}^{(\ell+1)}$, to lie within the polygonal limit set, we employ nonnegative barycentric coordinates, ${}^{u}\alpha_{ij} \geq 0;\ j = \{1,...,n_d\}$. The interior and boundary of the linearized friction impulse limit set can be represented as follows:

$$\left.\begin{aligned} {}^{u}\mathbf{p}_{if}^{(\ell+1)} &= {}^{u}\mathbf{D}_i\, {}^{u}\alpha_i \\ \textstyle\sum_{j=1}^{n_d} {}^{u}\alpha_{ij} &\leq \mu_i\, {}^{u}\mathbf{p}_{in}^{(\ell+1)} \end{aligned}\right\} \quad \forall\, i \in \mathcal{U} \tag{45}$$

where ${}^{u}\mathbf{D}_i$ is the matrix whose j^{th} column is the unit vector $\hat{\mathbf{d}}_{ij}$ and ${}^{u}\alpha_i$ is the vector with j^{th} element given by ${}^{u}\alpha_{ij}$.

For the developments in the next few paragraphs, it is important to see that if ${}^{u}\alpha_{ij} = \mu_i\, {}^{u}\mathbf{p}_{in}^{(\ell+1)}$, then the friction impulse is simply $\mu_i\hat{\mathbf{d}}_{ij}$, which is a vertex of the polygon. If



(a) Friction Cone (b) Limit Set

Figure 7: *Example of friction cone linearization using seven friction direction vectors. Note the linearization side-effect - the friction force that maximizes dissipation is not exactly opposite the relative velocity at the contact point.*

${}^u\alpha_{ij} + {}^u\alpha_{ik} = \mu_i {}^u\mathbf{p}_{in}^{(\ell+1)}$, where $\hat{\mathbf{d}}_{ij}$ and $\hat{\mathbf{d}}_{ik}$ are adjacent direction vectors, then the friction impulse is on an edge of the polygon. Importantly, these are the *only* ways to represent friction impulses on the boundary of the limit set using barycentric coordinates. All other coordinate combinations define a friction impulse on the interior of the polygon.

It remains to enforce that at rolling contacts, the friction impulse must be within the limit set, but while sliding, it must maximize power dissipation, which requires the impulse to be on the boundary of the limit set. Let the non-negative slack variable ${}^u\beta_i$ be a scalar sliding indicator for contact i, where ${}^u\beta_i = 0$ implies rolling and ${}^u\beta_i > 0$ implies sliding. When ${}^u\beta_i = 0$, the friction impulse may be anywhere in the interior of the polygon or on its boundary, but when ${}^u\beta_i > 0$ it must be on the boundary.

These two requirements suggest a complementarity relationship between the representation (second equation in bracketed equations just above) and ${}^u\beta_i$:

$$0 \leq \left(\mu_i \mathbf{p}_{in}^{(\ell+1)} - {}^u\mathbf{e}_i^T {}^u\alpha_i\right) \perp {}^u\beta_i \geq 0 \;\; \forall\, i \in \mathcal{U}. \qquad (46)$$

where $\mathbf{e}_i$ is a vector length n_d with all elements equal to one. This condition ensures that the friction impulse is in the cone, but it does not enforce maximum dissipation. To achieve the latter, one must introduce another condition that allows only one or two consecutive barycentric coordinates to be nonzero. One way to accomplish this is by the introduction of another complementarity constraint that maps the relative velocity of the friction subspace $\mathrm{v}_{if}^{(\ell+1)}$ onto the $\hat{\mathbf{d}}_{ij}$ vectors (this can be accomplished with ${}^u\mathbf{D}_i^T {}^u\mathbf{J}_{if}\mathbf{u}^{(\ell+1)}$) and identifies j such that $\hat{\mathbf{d}}_{ij}$ is most directly opposite to $\mathrm{v}_{if}^{(\ell+1)}$. The following linear complementarity condition, in conjunction with condition (46), identifies the correct $\hat{\mathbf{d}}_{ij}$:

$$\mathbf{0} \leq \left({}^u\mathbf{D}_i^T {}^u\mathbf{J}_{if}\mathbf{u}^{(\ell+1)} + \mathbf{e}_i {}^u\beta_i\right) \perp {}^u\alpha_i \geq \mathbf{0} \;\forall\, i \in \mathcal{U}. \qquad (47)$$

Consider for a moment complementarity condition (47). If the contact is sliding ${}^u\beta_i > 0$, at least one element of ${}^u\alpha_i$ must be positive. The only way to have a positive element of ${}^u\alpha_i$ is to have at least one element of the expression on the left be zero, which can only happen when ${}^u\beta_i$ takes on its minimum value. Note that this minimum value can never be zero as long as the vectors $\hat{\mathbf{d}}_{ij}$; $i = 1,\ldots,n_d$ positively span the friction subspace, and furthermore, it approximates the slip speed, with the approximation converging to the exact slip speed as n_d goes to infinity.

One important side-effect of the above approximation of the principle of maximum dissipation is that a finite cone of relative velocities at contact i leads to exactly the same friction impulse. Even if the direction of sliding changes smoothly, the direction of the friction impulse jumps from one direction vector to the next.

Combining the tangential complementarity conditions for all unilateral contacts yields linear complementarity systems that replace equations (25) and (32) in the instantaneous-time model:

$$\begin{aligned} \mathbf{0} &\leq \left({}^u\mathbf{D}^T {}^u\mathbf{J}_f\mathbf{u}^{(\ell+1)} + {}^u\mathbf{E}{}^u\beta\right) \quad \perp \quad {}^u\alpha \geq \mathbf{0} \\ \mathbf{0} &\leq \left(\mathbf{U}\,{}^u\mathbf{p}_n^{(\ell+1)} - {}^u\mathbf{E}^T {}^u\alpha\right) \quad \perp \quad {}^u\beta \geq \mathbf{0} \end{aligned} \qquad (48)$$

where the column vectors ${}^u\alpha$ and ${}^u\beta$ are formed by stacking the vectors ${}^u\alpha_i$ and scalars ${}^u\beta_i$, ${}^u\mathbf{D}^T$ is formed by stacking the matrices ${}^u\mathbf{D}_i^T$, ${}^u\mathbf{E}$ is a block diagonal matrix with nonzero blocks given by ${}^u\mathbf{e}_i$, and $\mathbf{U}$ is the diagonal matrix with element (i,i) equal to μ_i.

If all joints in the system are one-degree-of-freedom joints, equations (27) and (33) of the instantaneous-time model are replaced with the following:

$$\begin{aligned} \mathbf{0} &\leq \left({}^b\mathbf{D}^T {}^b\mathbf{J}_f\mathbf{u}^{(\ell+1)} + {}^b\mathbf{E}{}^b\beta\right) \quad \perp \quad {}^b\alpha \geq \mathbf{0} \\ \mathbf{0} &\leq \left({}^b\mathbf{p}_{f\max} - {}^b\mathbf{E}^T {}^b\alpha\right) \quad \perp \quad {}^b\beta \geq \mathbf{0} \end{aligned} \qquad (49)$$

where the column vectors ${}^b\alpha$ and ${}^b\beta$ are formed by stacking the vectors ${}^b\alpha_i$ and scalars ${}^b\beta_i$, ${}^b\mathbf{D}^T$ is formed by stacking the matrices ${}^b\mathbf{D}_i^T$, ${}^b\mathbf{E}$ is a block diagonal matrix with nonzero blocks given by ${}^b\mathbf{e}_i$, and ${}^b\mathbf{p}_{f\max}$ is $\Delta t\ {}^b\mathbf{f}_{f\max}$.

3.1.5. A Time-Stepping LCP

Equations (41–44,48,49) constitute a discrete-time model in the form of a mixed LCP, that is "mixed," because it contains equations that cannot be directly put into the form of a linear complementarity condition: the Newton-Euler equation (41), the kinematic map (42), and the normal joint constraint (43). Notice, however, that the only place in the mixed LCP in which the unknown generalized position $\mathbf{q}^{(\ell+1)}$ appears is the kinematic map equation (42). Therefore, the mixed LCP can be decoupled into an LCP with unknown generalized velocities and impulses and the kinematic map

equation with unknown generalized positions. The decoupled systems can be solved in two steps: solve the smaller mixed LCP (LCP1 defined below), then use the solution to LCP1 in the kinematic map to update the generalized positions.

Definition 4 LCP1: A mixed LCP with unknown velocities and impulses is constituted by equations (41,43,44,48,49).

It is known that solutions always exist to LCP1 if the terms ${}^u\mathbf{C}_n^{(\ell)}$, $\frac{\partial^u\mathbf{C}_n^{(\ell)}}{\partial t}\Delta t$, and $\frac{\partial^b\mathbf{C}_n^{(\ell)}}{\partial t}\Delta t$ from equations (43) and (44) are removed (see [AP97a]).

The mixed LCP1 can be solved in its current form by the PATH algorithm [FM99] or it can first be reformulated as a standard LCP and then solved. Since $\mathbf{M}$ is symmetric and positive definite, and under the assumption that the null space of ${}^b\mathbf{J}_n^{(\ell)}$ is trivial (which is usually true if there are no kinematic loops in mechanisms), then one can solve for both $\mathbf{u}^{(\ell+1)}$ and ${}^b\mathbf{p}_n^{(\ell+1)}$ with equations (41) and (43) and substitute the results into equations (39,48,49). Before substitution these equations :

$$0 \le \begin{bmatrix} {}^u\mathbf{J}_n^{(\ell)}\mathbf{u}^{(\ell+1)} + \frac{{}^u\mathbf{C}_n^{(\ell)}}{\Delta t} + \frac{\partial^u\mathbf{C}_n^{(\ell)}}{\partial t} \\ {}^u\mathbf{D}^T\,{}^u\mathbf{J}_f\mathbf{u}^{(\ell+1)} + {}^u\mathbf{E}\,{}^u\beta \\ {}^b\mathbf{D}^T\,{}^b\mathbf{J}_f\mathbf{u}^{(\ell+1)} + {}^b\mathbf{E}\,{}^b\beta \\ \mathbf{U}\,{}^u\mathbf{p}_n^{(\ell+1)} - {}^u\mathbf{E}^T\,{}^u\alpha \\ {}^b\mathbf{p}_{f\max} - {}^b\mathbf{E}^T\,{}^b\alpha \end{bmatrix} \perp \begin{bmatrix} {}^u\mathbf{p}^{(\ell+1)} \\ {}^u\alpha \\ {}^b\alpha \\ {}^u\beta \\ {}^b\beta \end{bmatrix} \ge 0. \quad (50)$$

For the remainder of section 3.1.5, we will drop the superscript $^{(\ell)}$ on all matrices and constraints, since they are evaluated at time t_ℓ. Continuing with the derivation of LCP1, we cast equations (41) and (43) into matrix form:

$$\begin{bmatrix} \mathbf{M} & -{}^b\mathbf{J}_n^T \\ -{}^b\mathbf{J}_n & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{(\ell+1)} \\ {}^b\mathbf{p}_n^{(\ell+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \quad (51)$$

where

$$\mathbf{x}_1 = {}^u\mathbf{J}_n^T\,{}^u\mathbf{p}_n^{(\ell+1)} + {}^b\mathbf{J}_f^T\,{}^b\mathbf{D}\,{}^b\alpha + {}^u\mathbf{J}_f^T\,{}^u\mathbf{D}\,{}^u\alpha + \mathbf{M}\mathbf{u}^{(\ell)} + \mathbf{p}_{\text{ext}}$$

$$\mathbf{x}_2 = \frac{{}^b\mathbf{C}_n}{\Delta t} + \frac{\partial^b\mathbf{C}_n}{\partial t}.$$

Inverting the matrix on the left side of equation (51) yields:

$$\begin{bmatrix} \mathbf{u}^{(\ell+1)} \\ {}^b\mathbf{p}_n^{(\ell+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{12}^T & \mathbf{S}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}. \quad (52)$$

Letting $\mathbf{B} = -{}^b\mathbf{J}_n\mathbf{M}^{-1}\,{}^b\mathbf{J}_n^T$, then $\mathbf{S}_{11}$, $\mathbf{S}_{12}$, and $\mathbf{S}_{22}$ are defined as follows:

$$\mathbf{S}_{11} = \mathbf{M}^{-1} + \mathbf{M}^{-1}\,{}^b\mathbf{J}_n^T\,\mathbf{B}^{-1}\,{}^b\mathbf{J}_n\,\mathbf{M}^{-1} \quad (53a)$$

$$\mathbf{S}_{12} = \mathbf{B}^{-1}\,{}^b\mathbf{J}_n\mathbf{M}^{-1} \quad (53b)$$

$$\mathbf{S}_{22} = \mathbf{B}^{-1}. \quad (53c)$$

Substituting back into inequalities (50) and using the shorthand ${}^\kappa\mathbf{J}_D = {}^\kappa\mathbf{J}_f\,{}^\kappa\mathbf{D}^T$ yields a standard LCP($\mathbf{A}$, $\mathbf{b}$) with $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{x}$ given as follows:

$$\mathbf{b} = \begin{bmatrix} {}^u\mathbf{J}_n\mathbf{r} + \frac{{}^u\mathbf{C}_n}{\Delta t} + \frac{\partial^u\mathbf{C}_n}{\partial t} \\ {}^u\mathbf{J}_D\mathbf{r} \\ {}^b\mathbf{J}_D\mathbf{r} \\ \mathbf{0} \\ {}^b\mathbf{p}_{f\max} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} {}^u\mathbf{p}_n \\ {}^u\alpha \\ {}^b\alpha \\ {}^u\beta \\ {}^b\beta \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} {}^u\mathbf{J}_n\mathbf{S}_{11}\,{}^u\mathbf{J}_n^T & {}^u\mathbf{J}_n\mathbf{S}_{11}\,{}^u\mathbf{J}_D^T & {}^u\mathbf{J}_n\mathbf{S}_{11}\,{}^b\mathbf{J}_D^T & \mathbf{0} & \mathbf{0} \\ {}^u\mathbf{J}_D\mathbf{S}_{11}\,{}^u\mathbf{J}_n^T & {}^u\mathbf{J}_D\mathbf{S}_{11}\,{}^u\mathbf{J}_D^T & {}^u\mathbf{J}_D\mathbf{S}_{11}\,{}^b\mathbf{J}_D^T & {}^u\mathbf{E} & \mathbf{0} \\ {}^b\mathbf{J}_D\mathbf{S}_{11}\,{}^u\mathbf{J}_n^T & {}^b\mathbf{J}_D\mathbf{S}_{11}\,{}^u\mathbf{J}_D^T & {}^b\mathbf{J}_D\mathbf{S}_{11}\,{}^b\mathbf{J}_D^T & \mathbf{0} & {}^b\mathbf{E} \\ \mathbf{U} & -{}^u\mathbf{E}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -{}^b\mathbf{E}^T & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

where

$$\mathbf{r} = \mathbf{S}_{11}\left(\mathbf{M}\mathbf{u}^{(\ell)} + \mathbf{p}_{\text{ext}}\right) + \mathbf{S}_{12}\left(\frac{{}^b\mathbf{C}_n}{\Delta t} + \frac{\partial^b\mathbf{C}_n}{\partial t}\right). \quad (54)$$

Note that it is known that when there are no joints (*i.e.,* rows three and five are removed from $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{x}$ and columns three and five are removed from $\mathbf{A}$) and the sum $\frac{{}^u\mathbf{C}_n}{\Delta t} + \frac{\partial^u\mathbf{C}_n}{\partial t}$ is nonnegative, then a solution always exists.

3.2. Modeling of Articulated Bodies and Jointed Mechanics

An articulated-body is a system of rigid bodies connected by joints (see figure 8). Each joint defines a holonomic constraint which must be resolved during the simulation. A holonomic constraint reduces the number of degrees of freedom of the system permanently (see section 2.1).

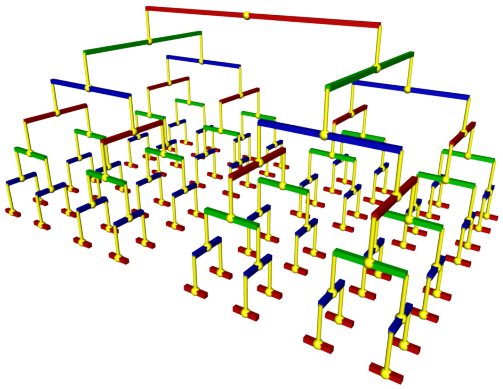


Figure 8: *This articulated body is a tree of rigid bodies which are connected by spherical joints.*

For the simulation of articulated bodies there exist two different formulations: the reduced (or generalized) coordinate formulation and the maximal coordinate formulation. The first one models the holonomic constraints by using a

reduced set of coordinates to describe the state of the system. An articulated-body has as many degrees of freedom as its number of independent coordinates. In contrast to that the maximal coordinate formulation uses the original coordinates of the rigid bodies and introduces additional forces or impulses in order to maintain the constraints.

3.2.1. Maximal coordinate formulation

The maximal coordinate formulation is well-known in the area of computer graphics. One of the first studies about multi-body simulation using this formulation was [BB88] which describes physically-based modeling with constraints. In the following, we will introduce different methods for this formulation.

Penalty force method The penalty force method can handle holonomic and nonholonomic equality constraints. These constraints are given in form of implicit functions in the form $\mathbf{C} = 0$. In a simulation step we can determine the violation of a constraint by evaluating its implicit function for the current state of the system. The result is zero if the constraint is fulfilled. Otherwise we add a force to the system in order to reduce the violation. For a holonomic equality constraint $\mathbf{C}(\mathbf{q},t) = 0$ this force can be computed by the following equation [dJB94]:

$$\mathbf{F}_{\text{penalty}} = -\alpha \mathbf{J}^T \left(\Omega^2 \mathbf{C} + 2\Omega\mu\dot{\mathbf{C}} + \ddot{\mathbf{C}}\right),$$

where $\mathbf{J}$ is the Jacobi matrix of the constraint $\mathbf{C}$. The values α, Ω and μ are constant parameters which are used to control the magnitude of the force. Multiplying the matrix $\mathbf{J}^T$ projects the force into the space of the constraint. The derivatives of the constraint $\dot{\mathbf{C}}$ and $\ddot{\mathbf{C}}$ are used in order to increase the stability. The resulting force is equivalent to the force of a damped spring with the spring constant α, natural frequency Ω and the damping ratio μ.

The force for a nonholonomic constraint is determined by

$$\mathbf{F}_{\text{penalty}} = -\alpha \left(\frac{\partial \mathbf{C}}{\partial \mathbf{u}}\right)^T (\mu\mathbf{C} + \dot{\mathbf{C}}).$$

For the simulation the penalty forces are added as external forces to the equation of motion. The penalty force method is easy to implement and very fast since the computation of penalty forces is very simple. The disadvantage of the method is that constraints can only be fulfilled approximately. An accurate solution is only possible for a very large value of α which leads to stiff differential equations.

Lagrange multipliers In contrast to the penalty force method the Lagrange multiplier method computes forces in order to prevent a violation of constraints. This method can simulate systems with holonomic and nonholonomic equality constraints. These constraints are transformed in the general constraint form

$$\mathbf{J}(\mathbf{q},\mathbf{u},t)\,\dot{\mathbf{u}} + \mathbf{k}(\mathbf{q},\mathbf{u},t) = \mathbf{0}. \tag{55}$$

In an n-dimensional system with an m-dimensional constraint the matrix $\mathbf{J}$ has the dimension $m \times n$ and the vector $\mathbf{k}$ the dimension m. A holonomic constraint is transformed in the general form by differentiating the constraint function $\mathbf{C}$ twice with respect to time. A nonholonomic equality constraint has just to be differentiated once. For a holonomic constraint $\mathbf{C}(\mathbf{q},t) = 0$ we get (cf. Equation 28)

$$\mathbf{J} = \frac{\partial \mathbf{C}}{\partial \mathbf{q}}\mathbf{H}, \quad \mathbf{k} = \frac{\partial \mathbf{C}}{\partial \mathbf{q}}\frac{\partial \mathbf{H}}{\partial t}\mathbf{u} + \frac{\partial^2 \mathbf{C}}{\partial \mathbf{q}\partial t}\mathbf{H}\mathbf{u} + \frac{\partial^2 \mathbf{C}}{\partial^2 t}.$$

A nonholonomic constraint $\mathbf{C}(\mathbf{q},\mathbf{u},t) = 0$ results in

$$\mathbf{J} = \frac{\partial \mathbf{C}}{\partial \mathbf{u}}, \quad \mathbf{k} = \frac{\partial \mathbf{C}}{\partial \mathbf{q}}\mathbf{H}\mathbf{u} + \frac{\partial \mathbf{C}}{\partial t}.$$

Most commonly in computer graphics the Lagrange multipliers λ are computed as follows. For each rigid body the mass matrix is defined by equation 14. The mass matrix for a particle with mass m is just the upper left block $m\mathbf{1}_{3\times 3}$. Constraints are simulated by additional forces $\mathbf{F}_c$ which are added to the equation of motion

$$\dot{\mathbf{u}} = \mathbf{M}^{-1}\left(\mathbf{F}_{\text{ext}} + \mathbf{F}_c\right) \tag{56}$$

where $\mathbf{M}$ is the mass matrix of the system which contains the mass matrices $\mathbf{M}_j$ of all bodies on the diagonal.

Substituting equation (56) into the general constraint (55), we obtain

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{F}_c = -\mathbf{J}\mathbf{M}^{-1}\mathbf{F}_{\text{ext}} - \mathbf{k}.$$

Regarding D'Alembert's principle [GPS02], it follows that

$$\mathbf{F}_c = \mathbf{J}^T\lambda. \tag{57}$$

Hence, the constraint forces always act in the constrained directions of system. Such forces do not influence the motion of the $n-m$ degrees of freedom of an articulated body. Finally, we get a system of linear equations for the Lagrange multipliers

$$\underbrace{\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T}_{\mathbf{A}}\lambda = \underbrace{-\mathbf{J}\mathbf{M}^{-1}\mathbf{F}_{\text{ext}} - \mathbf{k}}_{\mathbf{b}}. \tag{58}$$

The matrix $\mathbf{A}$ is positive definite if there are no conflicting or redundant constraints. Furthermore, the matrix is sparse for the most models since it reflects the structure of the articulated body. After solving for λ the constraint forces are determined by equation 57.

One of the most well-known Lagrange multiplier methods in computer graphics is the one of David Baraff [Bar96]. This method allows the simulation of articulated bodies without closed loops in linear time. Only constraints that act between a pair of bodies are supported.

For a linear time computation the system of linear equations 58 is transformed in the following form

$$\underbrace{\begin{pmatrix}\mathbf{M} & -\mathbf{J}^T\\ -\mathbf{J} & \mathbf{0}\end{pmatrix}}_{\mathbf{K}}\begin{pmatrix}\mathbf{y}\\ \lambda\end{pmatrix} = \begin{pmatrix}\mathbf{0}\\ -\mathbf{b}\end{pmatrix}.$$

Matrix $\mathbf{K}$ is known as the KKT-matrix [NW99]. The matrix $\mathbf{A}$ is smaller than $\mathbf{K}$ and positive definite if $\mathbf{J}$ has full rank while $\mathbf{K}$ is not. $\mathbf{A}$ has a row and column for each constraint while $\mathbf{K}$ has a row and column for each degree of freedom and each constraint. The advantage of the new formulation is that $\mathbf{K}$ is always sparse and symmetric.

The next step is to create an undirected graph for $\mathbf{K}$ with a node for each block of the matrix and an edge between the nodes $i \neq j$ for each $\mathbf{K}_{ij} \neq \mathbf{0}$. This graph is acyclic since the model has no loops. By a depth search in this graph the matrix is reordered so that the row index that corresponds to a node in the graph is greater than the one of its children. Afterwards a $\mathbf{LDL}^T$ decomposition is performed. Due to the reordered matrix structure the decomposition introduces no new nonzero elements, can be stored in linear space and performed in linear time. The system of linear equations for the Lagrange multipliers can also be solved in linear time. Finally, the velocities and positions are determined by numerical integration.

The Lagrange multiplier method computes constraint forces in order to prevent a violation of constraints due to external forces. If the constraints are violated in a different way, e.g. by errors that occur during numerical integration, the method cannot correct this. These errors sum up over the simulation and the method is not able to prevent joints from breaking. The problem is that a holonomic constraint is not regarded directly. It is just demanded that its second derivative is zero. An error term of the form $\mathbf{k}_1 t + \mathbf{k}_2$ where $\mathbf{k}_1$ and $\mathbf{k}_2$ are two arbitrary constant vectors cannot be corrected in this way since

$$\frac{d^2}{d^2t}\mathbf{C}(\mathbf{q},t) = \frac{d^2}{d^2t}\left(\mathbf{C}(\mathbf{q},t) + \mathbf{k}_1 t + \mathbf{k}_2\right).$$

Therefore, joints will break due to numerical errors. An analogous problem exists for nonholonomic equality constraints. Therefore, an additional stabilization is required for the simulation with Lagrange multipliers.

The method of Baumgarte is often used for stabilization [Bau72]. This method replaces the equation $\ddot{\mathbf{C}} = \mathbf{0}$ of a holonomic constraint by

$$\ddot{\mathbf{C}} + 2\alpha\dot{\mathbf{C}} + \beta^2\mathbf{C} = \mathbf{0}$$

where α and β are constant parameters. In this way position and velocity errors are regarded in the simulation. The equation $\dot{\mathbf{C}} = \mathbf{0}$ of a nonholonomic equality constraint is replaced by

$$\dot{\mathbf{C}} + \gamma\mathbf{C} = \mathbf{0}$$

where the parameter γ defines the weight of the velocity error in the multiplier computation.

The stabilization terms can be added to the general form of a constraint (equation (55)). Alternatively, the terms can be taken into account by adding additional forces to the equation of motion [WW90]. The determination of suitable parameters for the stabilization is not easy. Ascher et al. discuss the problems of finding suitable parameters and propose an enhanced stabilization method [ACPR95].

Impulse-based simulation The impulse-based method [BFS05, BS06b, Ben07, WTF06] is similar to the Lagrange multiplier method. The main difference between these methods is that the impulse-based approach determines impulses to perform a simulation with constraints by using a preview while the Lagrange multiplier method computes additional forces just regarding the current state.

Witkin et al. introduced a Lagrange multiplier method based on a constraint formulation with connectors [WGW90]. A connector is e.g. a point or vector in local coordinates of a body which is used to define a constraint. This allows to formulate generic constraints without knowledge about the object itself. For example, if two points are attached together, the corresponding constraint is $C(\mathbf{P}_1, \mathbf{P}_2) = \|\mathbf{P}_1 - \mathbf{P}_2\| = 0$ where the points $\mathbf{P}_1$ and $\mathbf{P}_2$ are connectors. The connector concept also solves the problem that the generalized position $\mathbf{q}$ and velocity $\mathbf{u}$ of a body commonly have not the same length (see Section 2.1).

In the following, constraints are defined by connectors $\mathbf{P}_i(\mathbf{q},t)$ fixed to the bodies:

$$\mathbf{C}(\mathbf{P}_1, \ldots, \mathbf{P}_m, t) = \mathbf{0}.$$

In this way the translational and rotational degrees of freedom can be constrained. By differentiating the constraint function $\mathbf{C}$ with respect to time we get a general constraint form for velocities $\mathbf{J}\mathbf{u} + \mathbf{k} = \mathbf{0}$ which is analogous to the one of equation 55. The matrix $\mathbf{J}$ and the vector $\mathbf{k}$ are determined by

$$\mathbf{J}_{ij} = \sum_k \frac{\partial \mathbf{C}_i}{\partial \mathbf{P}_k}\frac{\partial \mathbf{P}_k}{\partial \mathbf{q}_j}\mathbf{H}_j, \qquad \mathbf{k}_i = \frac{\partial \mathbf{C}_i}{\partial t} + \sum_k \frac{\partial \mathbf{P}_k}{\partial t}.$$

Now a system of linear equations for the impulses could be created which is analogous to the one of equation 58. However, the impulse-based method uses a different right hand side $\mathbf{b}$ for the system in order to solve the stabilization problem of the Lagrange multiplier method. The vector $\mathbf{b}$ is determined by a prediction of the joint state. This idea was first introduced by Bender et al. [BFS05] and later also used by Weinstein et al. [WTF06].

Figure 9 shows the preview of a ball joint with the holonomic constraint $\mathbf{C} = \mathbf{a} - \mathbf{b} = \mathbf{0}$. For the preview we assume that both rigid bodies are unconstrained. Then, we can easily compute the positions of the connector points $\mathbf{a}(t+h)$ and $\mathbf{b}(t+h)$ after a time step of size h by integration. The preview of a vector $\mathbf{r}$ fixed to a body is obtained by solving the differential equation

$$\dot{\mathbf{r}} = \boldsymbol{\omega} \times \mathbf{r}. \tag{59}$$

The predicted position of a point $\mathbf{a}$ is obtained by first solving equation 59 for the vector $\mathbf{r}(t) = \mathbf{a}(t) - \mathbf{x}(t)$ from the

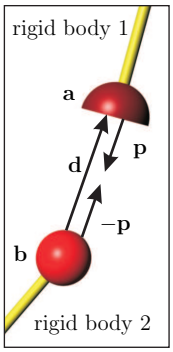


Figure 9: *Preview of a ball joint. The points* **a** *and* **b** *have different positions which must be corrected by a pair of impulses* **p** *and* $-\mathbf{p}$.

center of mass of the body to the point to get $\mathbf{r}(t+h)$. Then, we solve the equation of motion for the center of mass and determine the new position as $\mathbf{a}(t+h) = \mathbf{r}(t+h) + \mathbf{x}(t+h)$. For the predicted state we evaluate the constraint function and get the distance vector $\mathbf{d}(t+h) = \mathbf{a}(t+h) - \mathbf{b}(t+h)$. This vector shows us the violation which would occur without additional impulses in the system. Now we want to compute a pair of impulses $\mathbf{p}$ and $-\mathbf{p}$ for time t to prevent the violation. These impulses must cause a velocity change of the connectors so that the constraint $\mathbf{d}(t+h) = \mathbf{0}$ will be fulfilled.

The required impulses for a constraint can be determined by solving a nonlinear equation. Weinstein et al. use Newton iteration for the solution [WTF06]. In a system with multiple constraints there exist dependencies between the constraints if they have a common body. These dependencies are handled in an iterative way by Weinstein et al. In contrast to that Bender et al. linearize the equation by an approximation of the required velocity change. If the connectors have a linear relative motion, the required velocity change would be $\Delta\mathbf{v} = \mathbf{d}(t+h)/h$. Bender et al. use this value as an approximation for the nonlinear case which leads commonly to small errors [BS06b]. These errors are eliminated by solving the following system for the impulses $\mathbf{p}$ iteratively

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{p} = \Delta\mathbf{v}$$

where $\Delta\mathbf{v}$ is the vector containing the velocity changes for all constraints.

Numerical comparisons and of the impulse-based approach with other simulation methods can be found in [SB05,SBP05]. Bender showed in [Ben07] that the system of linear equations can be solved in linear time for articulated bodies without loops. In [BS06a] the impulse-based method is extended by inequality constraints in order to simulate collisions and resting contacts. Bayer et al. [BDB09] presented different optimizations for the impulse-based approach which increase the performance.

3.2.2. Reduced coordinate formulation

If we have a system of rigid bodies with m degrees of freedom and remove c of them by constraints, only the remaining $n = m - c$ degrees of freedom have to be simulated. The reduced coordinate formulation uses a parameterization for the m maximal coordinates in terms of n independent coordinates q_i which are called reduced or generalized coordinates. The m maximal coordinates of the system can be written as function of the reduced coordinates

$$x_i = x_i(q_1, \ldots, q_n), \quad i \in [1, \ldots, m].$$

Figure 10 shows an example for a reduced coordinate formulation. A particle with position $\mathbf{x} \in \mathbb{R}^2$ rotates around the

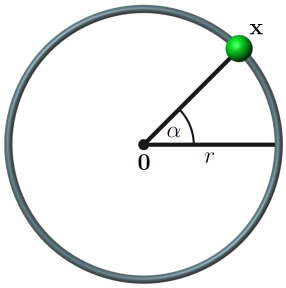


Figure 10: *The only degree of freedom of a particle moving on a circular path can be described by the angle* α.

origin on a circular path with radius r. This particle has only one degree of freedom which can be described by the angle α. The position in maximal coordinates is determined by

$$\mathbf{x}(\alpha) = r\begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix}. \tag{60}$$

Equation 60 defines all valid positions $\mathbf{x}$ for the particle.

In order to perform a simulation with reduced coordinates we must first find a parameterization for our model. Then, we form the equations of motion with respect to the reduced coordinates. The resulting system of differential equations can be solved by numerical methods.

The equations of motion can be obtained by using the Lagrange formulation [GPS02, Fea07]. We require the Lagrangian function $L = T - V$ where T and V are the total kinetic and potential energy respectively. This function describes the difference of the total kinetic and potential energy of the system which should be conserved. By the Euler-Lagrange equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = 0, \quad i = 1, \ldots, n$$

we get a system of differential equations for the motion of the bodies which can be solved numerically. Methods based on Lagrange formulation have a complexity of $O(n^4)$.

An overview over more efficient reduced coordinate

methods can be found in [FO00]. One of them is the well-known articulated-body algorithm (ABA) of Featherstone with a complexity of $O(n)$ for articulated bodies with tree-structure [Fea87]. This algorithm works in two phases. In the first phase the kinematic parameters are determined considering external forces. The parameters of a body in the tree only depend on the parameters of its parent. Therefore, the kinematic parameters are computed in one traversal of the tree. In the second phase the tree is traversed in reverse. During this traversal the internal forces are determined for each body which just depend on the children of the body. A detailed description of the ABA of Featherstone can be found in [Mir96b] and [Fea07].

The method of Featherstone is used in different areas of computer graphics. One application area is the simulation of rag-dolls which have a tree-structure. These are used for example for improved motion synthesis techniques which combine motion capture data with physical simulation [MZS09]. There are also other application areas like the simulation of strands [Had06] or in games [Kok04]. Redon et al. [RGL05] presented an adaptive variant of Featherstone's method in order to improve the performance. This approach allows to reduce the numbers of degrees of freedom (at the cost of accuracy) while it automatically determines the best set of active joints.

The method of Featherstone works for models without kinematic loops. Models with loops cause problems which are discussed in detail by Wittenburg [Wit77]. An example is shown in Figure 11. The model consists of one static rigid body and five dynamic bodies where each has six degrees of freedom. Therefore, the free system has 30 degrees of freedom. If we create hinge joints between the bodies as shown in the figure, these degrees of freedom are reduced. Each of the six hinge joints eliminates five degrees of freedom in a loop-free model. Hence, we could expect that the model has no degrees of freedom left. But in fact it has still one.

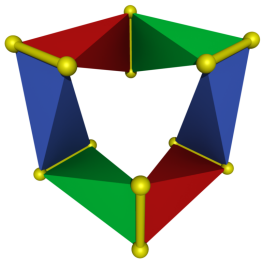


Figure 11: *Closed kinematic loop with one degree of freedom.*

The degrees of freedom of a closed-loop model can vary and forces in such a model can be indeterminate when the system is overconstrained. Therefore, these models needs some special treatment. A common approach to handle closed loops is to remove joints from the articulated body until we have a tree structure [FO00]. This is done by extracting a spanning tree from the connectivity graph. Now a simulation step is performed for the spanning tree and additional forces are added to mimic the effects of the kinematic loops. Loop handling is explained in detail in [Fea07].

4. The Numerical Solution Methods

Once discrete models have been obtained we must apply numerical methods to compute solutions. We start with how to integrate the motion of free moving rigid bodies such as bodies in ballistic motion without any collisions or contact. Subsequently in Sections 4.2- 4.4 we cover numerical methods for computing solutions of the discrete LCP contact model from Section 3.1.

4.1. Time Integration of Free Motion

For the simulation of constrained rigid bodies we extend the equation of motion by additional forces or impulses. Alternatively, we can formulate these equations in terms of reduced coordinates. In both cases we have to perform an integration step to obtain the dynamic state of a body for the next time step. Therefore, we want to introduce the most important numerical integration methods: semi-implicit Euler (also called symplectic Euler), Runge-Kutta methods and adaptive methods like the embedded Runge-Kutta.

In contrast to the well-known explicit Euler, the semi-implicit Euler uses the velocity at time t_0+h instead of time t_0 for the integration of the position vector:

$$\begin{aligned}\mathbf{u}(t_0+h) &= \mathbf{u}(t_0)+h\mathbf{M}^{-1}\mathbf{g}(\mathbf{q},\mathbf{u},t_0)\\ \mathbf{q}(t_0+h) &= \mathbf{q}(t_0)+h\mathbf{H}\mathbf{u}(t_0+h)\end{aligned}$$

where $\mathbf{M}$, $\mathbf{g}(\mathbf{q},\mathbf{u},t)$ and $\mathbf{H}$ are defined by equations 1, 14 and 15. The semi-implicit Euler is a first-order symplectic integrator. The advantage of integrating the velocities first is that the new velocities can be adapted before the position integration in order to resolve collisions or to simulate damping [GBF03, MHHR07].

Runge-Kutta methods are also very popular in the field of rigid body dynamics [BWAK03, RGL05, BS06b, Ben07] for solving the initial value problem given by the equation of motion. An initial value problem is an ordinary differential equation

$$\dot{\mathbf{z}} = \mathbf{h}(t,\mathbf{z}(t))$$

together with an initial value $(t_0,\mathbf{z}_0)$. One of the most important methods in this field is the fourth-order Runge-Kutta:

$$\begin{aligned}\mathbf{k}_1 &= h\mathbf{h}(t_i,\ \mathbf{z}_i)\\ \mathbf{k}_2 &= h\mathbf{h}(t_i+\frac{1}{2}h,\ \mathbf{z}_i+\frac{1}{2}\mathbf{k}_1)\\ \mathbf{k}_3 &= h\mathbf{h}(t_i+\frac{1}{2}h,\ \mathbf{z}_i+\frac{1}{2}\mathbf{k}_2)\\ \mathbf{k}_4 &= h\mathbf{h}(t_i+h,\ \mathbf{z}_i+\mathbf{k}_3)\end{aligned}$$

$$\mathbf{z}_{i+1} \;=\; \mathbf{z}_i + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4).$$

This method can also be combined with adaptive time-stepping [PFTV92, Mir96b]. The goal of using an adaptive time step size is to achieve a predefined accuracy ε with a minimal computational effort. For the determination of the current step size, the truncation error of a step must be estimated. One way to do this is using the embedded Runge-Kutta formulas introduced by Fehlberg. Cash and Karp [CK90] use the general form of a fifth-order Runge-Kutta formula

$$\begin{aligned}
\mathbf{k}_1 &= h\mathbf{h}(t_i,\, \mathbf{z}_i) \\
\mathbf{k}_2 &= h\mathbf{h}(t_i + a_2 h,\, \mathbf{z}_i + b_{21}\mathbf{k}_1) \\
&\vdots \\
\mathbf{k}_6 &= h\mathbf{h}(t_i + a_6 h,\, \mathbf{z}_i + b_{61}\mathbf{k}_1 + \ldots + b_{65}\mathbf{k}_5)
\end{aligned}$$

in combination with two different sets of parameters for the solution

$$\begin{aligned}
\mathbf{z}_{i+1} &= \mathbf{z}_i + \sum_{j=1}^{6} c_j \mathbf{k}_j + O(h^6) \\
\mathbf{z}^*_{i+1} &= \mathbf{z}_i + \sum_{j=1}^{6} c^*_j \mathbf{k}_j + O(h^5).
\end{aligned}$$

The first set gives us a fifth-order Runge-Kutta. From the second set we obtain an embedded fourth-order formula. The values for the parameters a_i, b_{ij}, c_i and c_i^* can be found in [CK90]. Now the error can be estimated as

$$\Delta\mathbf{z} = \mathbf{z}_{i+1} - \mathbf{z}^*_{i+1} = \sum_{i=1}^{6} (c_i - c_i^*) \cdot \mathbf{k}_i.$$

Since this error is of order five, a new step size can be determined by

$$h_{new} = h \left(\frac{\varepsilon}{|\Delta\mathbf{z}|} \right)^{\frac{1}{5}}.$$

If the estimated error is smaller than the desired accuracy, the step size is increased. Otherwise if the desired accuracy could not be reached, the last step has to be simulated again with the new decreased step size.

4.2. Direct Methods

Direct methods are known to be computational heavy to use. Therefore, they are often not preferred for interactive simulation. However, their ability to deliver accurate solutions make them ideal to handle problems such as large mass ratios. Thus, for some applications direct methods are the only option. Among direct methods for linear complementarity problems (LCPs) based on pivoting are the Lemke method and the Keller method [CPS92a, Lac03]. We will present an incremental pivoting method in the spirit of [Bar94]. Before doing so we will first present a guessing approach that exploits that a LCP is a combinatorial problem. The LCP can be written as

$$\mathbf{y} \geq \mathbf{0},\; \mathbf{x} \geq \mathbf{0},\; \text{and } \mathbf{y}^T\mathbf{x} = 0,$$

where $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$. By algebraic manipulation we have,

$$\begin{bmatrix} \mathbf{1} & -\mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} = \mathbf{b}.$$

Next we define the whole index set $\mathcal{I} = \{1, \ldots, n\}$ and introduce one index set of free variables $\mathbf{y}_i > 0$ and one of active variables $\mathbf{y}_i = 0$,

$$\mathcal{F} \equiv \{i \mid \mathbf{y}_i > 0\} \text{ and } \mathcal{A} \equiv \{i \mid \mathbf{x}_i > 0\}.$$

We assume strict complementarity holds meaning we never simultaneously have $\mathbf{y}_i = 0$ and $\mathbf{x}_i = 0$. Thus, $\mathcal{F} \cap \mathcal{A} = \emptyset$ and $\mathcal{F} \cup \mathcal{A} = \{1, \ldots, n\}$. The idea is to create a method that can verify if any guess of $\mathcal{F}$ and $\mathcal{A}$ is a solution for the given LCP formulation. Using the index sets we make the partitioning

$$\underbrace{\begin{bmatrix} \mathbf{1}_{\cdot\mathcal{F}} & -\mathbf{A}_{\cdot\mathcal{A}} \end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix} \mathbf{y}_{\mathcal{F}} \\ \mathbf{x}_{\mathcal{A}} \end{bmatrix}}_{\mathbf{x}} = \mathbf{b}.$$

where $\mathbf{1}_{\cdot\mathcal{F}}$ and $\mathbf{A}_{\cdot\mathcal{A}}$ are the sub matrices given by the column indices $\mathcal{F}$ and $\mathcal{A}$. Our problem is simplified to verifying if the linear programming (LP) problem

$$\mathbf{C}\mathbf{x} = \mathbf{b} \quad \text{subject to} \quad \mathbf{x} \geq \mathbf{0}$$

has a solution (same as $\mathbf{b}$ in positive cone of $\mathbf{C}$). This can be done by first computing $\mathbf{x}_{\mathcal{A}} = -\mathbf{A}^{-1}_{\mathcal{A}\mathcal{A}}\mathbf{b}_{\mathcal{A}}$, and verify if $\mathbf{x}_{\mathcal{A}} \geq 0$. Next one uses the feasible $\mathbf{x}_{\mathcal{A}}$ to compute $\mathbf{y}_{\mathcal{F}} = \mathbf{A}_{\mathcal{F}\mathcal{A}}\mathbf{x}_{\mathcal{A}} + \mathbf{b}_{\mathcal{F}}$ and finally verify if $\mathbf{y}_{\mathcal{F}} \geq 0$. If that last verification succeeds then a solution has been found. Observe that during the verification processes we only need to compute $\mathbf{A}^{-1}_{\mathcal{A}\mathcal{A}}$. If $\| \mathcal{A} \| \ll n$ then verification will be fast.

In worst case the time complexity of guessing would be $\mathcal{O}(n^3 2^n)$ which is not computational very efficient. Another strategy is to be clever in making new guesses. For instance by applying a pivoting strategy or some strategy that builds up the index sets incrementally. Here we will focus on the latter idea. In the k^{th} iteration a new index will be selected from the current set of unprocessed indices, $\mathcal{U} \equiv \mathcal{I} \setminus \{\mathcal{F} \cup \mathcal{A}\}$. The index sets $\mathcal{F}$ and $\mathcal{A}$ are initially both empty. Throughout, the complementarity conditions are kept as invariants. We will use superscript k to denote the values at a given iteration number. For any unprocessed index $j \in \mathcal{U}$ we implicitly assume $\mathbf{x}^k_j = 0$. Initially in the k^{th} iteration we use the partitioning

$$\begin{bmatrix} \mathbf{y}^k_{\mathcal{A}} \\ \mathbf{y}^k_{\mathcal{F}} \\ \mathbf{y}^k_{\mathcal{U}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\mathcal{A},\mathcal{A}} & \mathbf{A}_{\mathcal{A},\mathcal{F}} & \mathbf{A}_{\mathcal{A},j} \\ \mathbf{A}_{\mathcal{F},\mathcal{A}} & \mathbf{A}_{\mathcal{F},\mathcal{F}} & \mathbf{A}_{\mathcal{F},j} \\ \mathbf{A}_{\mathcal{U},\mathcal{A}} & \mathbf{A}_{\mathcal{U},\mathcal{F}} & \mathbf{A}_{\mathcal{U},\mathcal{U}} \end{bmatrix} \begin{bmatrix} \mathbf{x}^k_{\mathcal{A}} \\ \mathbf{x}^k_{\mathcal{F}} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_{\mathcal{A}} \\ \mathbf{b}_{\mathcal{F}} \\ \mathbf{b}_{\mathcal{U}} \end{bmatrix}.$$

The next candidate index to be processed in the method is selected as the index $j \in \mathcal{U}$ that minimize $\mathbf{y}^k_j$ as this corresponds to an index in $\mathcal{U}$ with a most violated complementarity constraint. If for the minimum value $\mathbf{y}^k_j \geq 0$ is fulfilled,

the method terminates as this would indicate that all the remaining unprocessed indices trivially fulfill the complementarity conditions. If no unique feasible minimum exists then one may pick a minimizing index at random. In the k^{th} iteration we use the partitioning and keep the complementarity conditions as invariants implying $\mathbf{y}_{\mathcal{A}}^{k+1} = \mathbf{0}$ and $\mathbf{x}_{\mathcal{F}}^{k+1} = \mathbf{0}$, so

$$\begin{bmatrix} \mathbf{0} \\ \mathbf{y}_{\mathcal{F}}^{k+1} \\ \mathbf{y}_{j}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\mathcal{A},\mathcal{A}} & \mathbf{A}_{\mathcal{A},\mathcal{F}} & \mathbf{A}_{\mathcal{A},j} \\ \mathbf{A}_{\mathcal{F},\mathcal{A}} & \mathbf{A}_{\mathcal{F},\mathcal{F}} & \mathbf{A}_{\mathcal{F},j} \\ \mathbf{A}_{j,\mathcal{A}} & \mathbf{A}_{j,\mathcal{F}} & \mathbf{A}_{j,j} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{A}}^{k+1} \\ \mathbf{0} \\ \mathbf{x}_{j}^{k+1} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_{\mathcal{A}} \\ \mathbf{b}_{\mathcal{F}} \\ \mathbf{b}_{j} \end{bmatrix}.$$

The changes in $\mathbf{y}_{\mathcal{F}}$ and $\mathbf{x}_{\mathcal{A}}$ with respect to $\mathbf{x}_j^{k+1} > 0$ are given by

$$\begin{aligned} \mathbf{x}_{\mathcal{A}}^{k+1} &= \mathbf{x}_{\mathcal{A}}^{k} + \Delta\mathbf{x}_{\mathcal{A}}\mathbf{x}_{j}^{k+1}, \\ \mathbf{y}_{\mathcal{F}}^{k+1} &= \mathbf{y}_{\mathcal{F}}^{k} + \Delta\mathbf{y}_{\mathcal{F}}\mathbf{x}_{j}^{k+1}, \\ \mathbf{y}_{j}^{k+1} &= \mathbf{y}_{j}^{k} + \Delta\mathbf{y}_{j}\mathbf{x}_{j}^{k+1} \end{aligned}$$

where

$$\begin{aligned} \Delta\mathbf{x}_{\mathcal{A}} &= -\mathbf{A}_{\mathcal{A},\mathcal{A}}^{-1}\mathbf{A}_{\mathcal{A},j}, \\ \Delta\mathbf{x}_{\mathcal{A}} &= -\mathbf{A}_{\mathcal{A},\mathcal{A}}^{-1}\mathbf{A}_{\mathcal{A},j}, \\ \Delta\mathbf{y}_{j} &= -\mathbf{A}_{j,j} - \mathbf{A}_{j,\mathcal{A}}\mathbf{A}_{\mathcal{A},\mathcal{A}}^{-1}\mathbf{A}_{\mathcal{A},j}. \end{aligned}$$

The idea is to increase $\mathbf{x}_j^{k+1}$ as mush as possible without breaking any of the complementarity constraints. Thus, $\mathbf{x}_j^{k+1}$ is limited by the blocking constraint set

$$\mathcal{B}_{\mathcal{A}} \equiv \left\{ \frac{-\mathbf{x}_q^k}{\Delta\mathbf{x}_q} \;\middle|\; q \in \mathcal{A} \wedge \Delta\mathbf{x}_q < 0 \right\}, \tag{63a}$$

$$\mathcal{B}_{\mathcal{F}} \equiv \left\{ \frac{-\mathbf{y}_r^k}{\Delta\mathbf{y}_r} \;\middle|\; r \in \mathcal{F} \wedge \Delta\mathbf{y}_r < 0 \right\}. \tag{63b}$$

If no blocking constraints exist then $\mathbf{x}_j^{k+1}$ is unbounded by $\mathcal{A}$ and $\mathcal{F}$. Thus, each partition results in the bounds

$$\mathbf{x}_j^{\mathcal{A}} = \begin{cases} \infty & : \mathcal{B}_{\mathcal{A}} = \emptyset \\ \min \mathcal{B}_{\mathcal{A}} & ; \end{cases}, \tag{64a}$$

$$\mathbf{x}_j^{\mathcal{F}} = \begin{cases} \infty & : \mathcal{B}_{\mathcal{F}} = \emptyset \\ \min \mathcal{B}_{\mathcal{F}} & ; \end{cases}, \tag{64b}$$

$$\mathbf{x}_j^{j} = \begin{cases} \frac{-\mathbf{y}_j^k}{\Delta\mathbf{y}_j} & ; \Delta\mathbf{y}_j < 0 \\ 0 & ; \end{cases}. \tag{64c}$$

The solution for the value of $\mathbf{x}_j^{k+1}$ will be the minimum bound. If a blocking constraint is found from $\mathcal{B}_{\mathcal{A}}$ then a pivot operation is initiated moving the blocking index from $\mathcal{A}$ to $\mathcal{F}$ and vice versa if a blocking constraint is found in $\mathcal{B}_{\mathcal{F}}$.

The blocking constraint sets are changed as the active and free index sets $\mathcal{A}$ and $\mathcal{F}$ are changed by a pivoting operation. This implies that one could increase $\mathbf{x}_j^{k+1}$ further after a pivoting step. Thus, we will continue to look for blocking constraints and perform pivoting on them until no more blocking constraints exist. Depending on the final value of $\mathbf{x}_j^{k+1}$ index j is assigned to either $\mathcal{F}$ or $\mathcal{A}$.

Noticing that the pivot step only swaps one index and therefore only changes the size of $\mathcal{A}$ by one an incremental factorization method can be used for computing $\mathbf{A}_{\mathcal{A},\mathcal{A}}^{-1}$. There exist incremental factorization running in $\mathcal{O}(n^2)$ time complexity. Baraff [Bar94] proved that the outer loop runs at most $\mathcal{O}(n)$. Thus, a positive overall time complexity for the pivoting method is $\mathcal{O}(n^3)$.

The pivoting method is capable of finding an accurate solution for the LCP whereas the iterative methods we cover in Section 4.3 and 4.4 only find approximate solutions. However, the accuracy is at the expense of having to form the **A**-matrix in the first place whereas the iterative methods often can exploit a factorization of the **A**-matrix given by the constraint Jacobians and the mass matrix, $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$. These matrices are extremely sparse and one can evaluate matrix-vector products more efficiently using the factorization than by first assembling the **A**-matrix which can be very dense even if it consists of products of sparse matrices.

If we let b be the number of rigid bodies then storage complexity of the matrix product factorization $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ is $\mathcal{O}(b+n)$ compared by the $\mathcal{O}(n^2)$ storage complexity for the full **A**-matrix. The assembly of the **A**-matrix takes at worse $\mathcal{O}(nb^2 + bn^2)$. The iterative methods often only need to compute matrix-vector products using the matrix-product factorization since we often have $b \ll n$ this takes $\mathcal{O}(n)$ time compared against a full matrix which takes $\mathcal{O}(n^2)$ time.

4.3. Iterative Fixed Point Schemes

Most open source software for interactive real-time rigid body simulation uses the Projected Gauss–Seidel (PGS) method for computing contact forces. This includes the two most popular open source simulators Bullet and Open Dynamics Engine. PGS is computational very efficient with an iteration cost of $\mathcal{O}(n)$, careful memory layout of sparse matrices allows for a memory footprint of $\mathcal{O}(n)$. In addition to being computational and memory-wise efficient PGS is very robust and can deal gracefully with even bad or erroneous problems. For these reasons PGS is well suited for interactive applications like computer games.

4.3.1. Matrix Splitting Methods

We introduce the matrix splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$. Next we let $\mathbf{c}^k = \mathbf{b} - \mathbf{N}\mathbf{x}^k$ then the LCP

$$\mathbf{A}\mathbf{x} + \mathbf{b} \geq \mathbf{0}, \tag{65a}$$

$$\mathbf{x} \geq \mathbf{0}, \tag{65b}$$

$$(\mathbf{x})^T(\mathbf{A}\mathbf{x} + \mathbf{b}) = 0. \tag{65c}$$

becomes

$$\mathbf{M}\mathbf{x}^{k+1} + \mathbf{c}^k \geq \mathbf{0}, \tag{66a}$$

$$\mathbf{x}^{k+1} \geq \mathbf{0}, \tag{66b}$$

$$(\mathbf{x}^{k+1})^T(\mathbf{M}\mathbf{x}^{k+1} + \mathbf{c}^k) = 0. \tag{66c}$$

This results in a fixed-point formulation where we hope that for a suitable choice of $\mathbf{M}$ and $\mathbf{N}$ the complementarity subproblem might be easier to solve than the original problem. The splitting method can be summarized as

Step 0 Initialization, set $k = 0$ and choose an arbitrary non-negative $\mathbf{x}^0 \geq \mathbf{0}$.
Step 1 Given $\mathbf{x}^k \geq \mathbf{0}$ solve the LCP (66).
Step 2 If $\mathbf{x}^{k+1}$ satisfy some stopping criteria then stop otherwise set $k \leftarrow k+1$ and go to step 1.

The splitting is often chosen such that $\mathbf{M}$ is a Q-matrix. This means that $\mathbf{M}$ belongs to the matrix class of matrices where the corresponding LCP has a solution for all vectors $\mathbf{c}^k$. Clearly if $\mathbf{x}^{k+1}$ is a solution for (66) and we have $\mathbf{x}^{k+1} = \mathbf{x}^k$ then by substitution into the subproblem given by (66) we see that $\mathbf{x}^{k+1}$ is a solution of the original problem (65).

Next we will use the minimum map reformulation on the complementarity subproblem, this is equivalent to

$$\min(\mathbf{x}^{k+1}, \mathbf{M}\mathbf{x}^{k+1} + \mathbf{c}^k) = \mathbf{0}. \tag{67}$$

Subtract $\mathbf{x}^{k+1}$ and multiply by minus one,

$$\max(\mathbf{0}, -\mathbf{M}\mathbf{x}^{k+1} - \mathbf{c}^k + \mathbf{x}^{k+1}) = \mathbf{x}^{k+1}. \tag{68}$$

Again we re-discover a fixed-point formulation. Let us perform a case-by-case analysis of the i^{th} component. If

$$\left(\mathbf{x}^{k+1} - \mathbf{M}\mathbf{x}^{k+1} - \mathbf{c}^k\right)_i < 0 \tag{69}$$

then $\mathbf{x}_i^{k+1} = 0$. Otherwise

$$\left(\mathbf{x}^{k+1} - \mathbf{M}\mathbf{x}^{k+1} - \mathbf{c}^k\right)_i = \mathbf{x}_i^{k+1}. \tag{70}$$

That is

$$(\mathbf{M}\mathbf{x}^{k+1})_i = \mathbf{c}_i^k. \tag{71}$$

For a suitable choice of $\mathbf{M}$ and back-substitution of $\mathbf{c}^k = \mathbf{b} - \mathbf{N}\mathbf{x}^k$ we have

$$\left(\mathbf{M}^{-1}\left(\mathbf{N}\mathbf{x}^k - \mathbf{b}\right)\right)_i = \mathbf{x}_i^{k+1}. \tag{72}$$

Combining it all we have derived the closed form solution for the complementarity subproblem,

$$\max\left(\mathbf{0}, \left(\mathbf{M}^{-1}\left(\mathbf{N}\mathbf{x}^k - \mathbf{b}\right)\right)\right) = \mathbf{x}^{k+1}. \tag{73}$$

Iterative schemes like these are often termed projection methods. The reason for this is that if we introduce the vector $\mathbf{z}^k = \mathbf{M}^{-1}\left(\mathbf{N}\mathbf{x}^k - \mathbf{b}\right)$ then

$$\mathbf{x}^{k+1} = \max\left(\mathbf{0}, \mathbf{z}^k\right). \tag{74}$$

That is the $k+1$ iteration is obtained by projecting the vector $\mathbf{z}^k$ onto the positive octant. In a practical implementation one would rewrite the matrix equation (74) into a for loop that sweeps over the vector components and updates the $\mathbf{x}$-vector in place. The result is the same pseudo code as given in Section 4.3.2.

One would want to use a clever splitting such that the inversion of $\mathbf{M}$ is computationally cheap. Letting $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{U}$ be the strict lower, diagonal and strict upper parts of $\mathbf{A}$, then three popular choices are: the projected Jacobi method $\mathbf{M} = \mathbf{D}$ and $\mathbf{N} = \mathbf{L} + \mathbf{U}$, the projected Gauss–Seidel (PGS) method $\mathbf{M} = (\mathbf{L} + \mathbf{D})$ and $\mathbf{N} = \mathbf{U}$, and the projected Successive Over Relaxation (PSOR) method $\mathbf{M} = (\mathbf{D} + \gamma\mathbf{L})$ and $\mathbf{N} = ((1-\gamma)\mathbf{D} - \gamma\mathbf{U})$ where $0 \leq \gamma \leq 2$ is the relaxation parameter. More about this parameter in Section 4.3.2.

It is worthwhile to note that $\mathbf{A}$ must at least have nonzero diagonal for these splittings to work. In general for non-symmetric matrices one may experience divergence. This means we can not apply these methods directly to the LCP model. Thus, In computer graphics an alternative model has been used which drops the principle of maximum dissipation. This alternative allows for a matrix splitting method to be derived [PNE10]. One may improve the accuracy and convergence rate of the resulting numerical method by using sub-space minimization [NSE10] or a nonsmooth nonlinear conjugate gradient method [SNE10b].

It seems that all hope of using matrix splitting for the LCP model is lost. However, as we show in Section 4.3.3 and 4.3.4 a blocked version of the matrix splittings can be used for the LCP model.

4.3.2. Using A Quadratic Programming Problem

In our second approach for deriving the iterative methods PGS and PSOR we will make use of the quadratic programming (QP) problem reformulation. Our derivation follows in the footsteps of [Man84]. The reformulation allows us to prove convergence properties of the PGS and PSOR methods. We assume that $\mathbf{A}$ is symmetric and positive semi-definite then the LCP can be restated as a minimization problem of a constrained convex QP problem

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\geq\mathbf{0}} f(\mathbf{x}) \tag{75}$$

where $f(\mathbf{x}) \equiv \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{b}$. The first order optimality (Karush-Kuhn-Tucker) conditions [NW99] is equivalent to the LCP (65).

Given the i^{th} unit axis vector $\hat{e}^i$ where $\hat{e}^i_j = 0$ for all $j \neq i$ and $\hat{e}^i_i = 1$ then the i^{th} relaxation step consists in solving the one dimensional problem

$$\tau^* = \arg\min_{\mathbf{x}\geq\mathbf{0}} f(\mathbf{x} + \tau\hat{e}^i) \tag{76}$$

and then setting $\mathbf{x} \leftarrow \mathbf{x} + \tau\hat{e}^i$. One relaxation cycle consists of one sequential sweep over all i^{th} components.

The one dimensional objective function is rewritten as

$$f(\mathbf{x}+\tau\hat{e}^i) = \frac{1}{2}\tau^2\mathbf{A}_{ii} + \tau(\underbrace{\mathbf{A}\mathbf{x}+\mathbf{b}}_{\equiv\mathbf{r}})_i + f(\mathbf{x}).$$

From which we find the unconstrained minimizer as $\tau_u = -\frac{\mathbf{r}_i}{\mathbf{A}_{ii}}$. Considering the constraint $\mathbf{x}_i + \tau \geq 0$ we find the constrained minimizer to be $\tau_c = \max(\tau_u, -\mathbf{x}_i)$ which yields the final update rule for the relaxation step

$$\mathbf{x}_i \leftarrow \max\left(0, \mathbf{x}_i - \frac{\mathbf{r}_i}{\mathbf{A}_{ii}}\right). \tag{78}$$

This is algebraic equivalent to the i^{th} component in the PGS update (74). Consider the polynomial $g(\tau) \equiv \frac{1}{2}\tau^2\mathbf{A}_{ii} + \tau\mathbf{r}_i$. We know $\mathbf{A}_{ii} > 0$ so the legs of the polynomial are pointing upwards. The polynomial has one trivial root $\tau = 0$ and a minimum at $\tau = -\frac{\mathbf{r}_i}{\mathbf{A}_{ii}}$ where $g\left(-\frac{\mathbf{r}_i}{\mathbf{A}_{ii}}\right) = -\frac{\mathbf{r}_i^2}{\mathbf{A}_{ii}} < 0$. The other root is found at $\tau = -2\frac{\mathbf{r}_i}{\mathbf{A}_{ii}}$. Thus, any τ value in the interval between the two roots has the property

$$\tau_\gamma = -\gamma\frac{\mathbf{r}_i}{\mathbf{A}_{ii}} \Rightarrow g(\tau_\gamma) < 0, \quad \forall\gamma \in [0..2]. \tag{79}$$

From this it follows that

$$f(\mathbf{x}+\tau_\gamma\hat{e}^i) = g(\tau_\gamma) + f(\mathbf{x}) \leq f(\mathbf{x}), \quad \forall\gamma \in [0..2] \tag{80}$$

with equality if $\tau_\gamma = 0$. This results in the over relaxed version

$$\mathbf{x}_i \leftarrow \max\left(0, \mathbf{x}_i - \gamma\frac{\mathbf{r}_i}{\mathbf{A}_{ii}}\right). \tag{81}$$

This is in fact algebraic equivalent to the i^{th} component of the PSOR update and contains the PGS method as a special case of $\gamma = 1$. Observe that by (80) we are guaranteed a non increasing sequence of iterates by our relaxation method. The complete iterative method can be listed as

```
1 : method PSOR(N, γ, x, A, b)
2 :   for k = 1 to N
3 :     for all i
4 :       r_i ← A_{i*} x + b_i
5 :       x_i ← max(0, x_i − γ r_i / A_ii)
6 :     next i
7 :   next k
8 : end method
```

where N is the maximum number of allowed iterations and γ is the relaxation parameter.

4.3.3. The Blocked Gauss–Seidel Method

The matrix splitting and QP reformulation approaches imply that Gauss–Seidel methods can not be used for the LCP contact model due to its zero diagonal values and non symmetry of $\mathbf{A}$. However, the splitting idea can be applied in a blocked version. This results in a numerical method that is very easy to implement and still preserves the good numerical properties of the PGS method. A block is defined as all variables from one contact point. In the case of a four sided friction pyramid the i^{th} block will consist of the normal impulse $\mathbf{x}_{n,i}$, four friction impulses $\mathbf{x}_{t_1,i}$, $\mathbf{x}_{t_2,i}$, $\mathbf{x}_{t_3,i}$, $\mathbf{x}_{t_4,i}$ and one slack variable β_i. We introduce the block notation $[\mathbf{x}]_i = \begin{bmatrix}\mathbf{x}_{n,i} & \mathbf{x}_{t_1,i} & \cdots & \beta_i\end{bmatrix}^T$. Similar $[\mathbf{A}]_{ij}$ is the sub block of $\mathbf{A}$ corresponding to the i^{th} and j^{th} contact point variables. Thus, the blocked LCP can be written

$$[\mathbf{y}]_i = \sum_j [\mathbf{A}]_{ij}[\mathbf{x}]_j + [\mathbf{b}]_i \geq \mathbf{0} \quad \forall i, \tag{82a}$$

$$[\mathbf{x}]_i \geq \mathbf{0} \quad \forall i, \tag{82b}$$

$$[\mathbf{y}]_i^T [\mathbf{x}]_i = 0 \quad \forall i. \tag{82c}$$

Now we may apply the Gauss–Seidel splitting to the blocked LCP. The result is a blocked Gauss–Seidel (BGS) method,

```
1 : method BGS(N, x, A, b)
2 :   for k = 1 to N
3 :     for all i
4 :       [b]'_i ← [b]_i − Σ_{j≠i} [A]_ij [x]_j
5 :       solve-sub-lcp([x]_i, [A]_ii, [b]'_i)
6 :     next i
7 :   next k
8 : end method
```

The intuition behind the numerical method is that all contact point variables other than the i^{th} block are momentarily frozen while solving for the variables of the i^{th} block. The BGS approach is also known as a "sweeping process" or as the non-smooth contact dynamics (NSCD) method [Mor99, Jea99].

The sub block LCP in line 5 can be solved using any LCP solver one wants. Usually one would apply yet a splitting dividing the sub block LCP into a normal impulse sub block and a frictional sub block. The normal part is a 1D problem and can be solved by a projection. The frictional part would in our case be a 5D problem. It is a bit unpleasant as we have zero diagonal terms and non-symmetry of the frictional sub block part of $\mathbf{A}$. However, the low dimensionality would allow for an efficient direct enumeration approach or one may drop the principle of maximum dissipation – changing the contact model – but allowing us to reduce the number of variables to a 2D problem with a symmetric positive semi-definite frictional sub block matrix.

From a computer science viewpoint an implementation of this method is indistinguishable from an implementation of the propagation model. The main difference is that this is a numerical method for solving a simultaneous contact model whereas the other is a model in itself. Besides the former solves for force impulses whereas the latter solves for collision impulses. The similarity with the propagation model also give intuition to some of the traits of the numerical method. One may see propagation effects even though one is using a simultaneous model.

The blocked Gauss–Seidel method offers many possibil-

ities. In Section 4.3.4 we divide a LCP into two sub blocks one with normal variables only and the other containing the rests. In fact one may use any kind of partitionings to create the sub blocks. For instance If the LCP includes joints one may create a sub block for all the joint variables. This joint sub block of the LCP is known to be equivalent to a symmetric positive semi-definite linear system. Thus, one may use a preconditioned conjugate gradient (PCG) solver to solve for joint impulses rather than a PGS method. As PCG has the same per-iteration cost as PGS but better convergence rate the result is much less joint drifting errors at the same cost as PGS. If the number of joints is sufficiently small one may even use an incomplete Cholesky factorization to solve for joint impulses resulting in very accurate solutions. One may even take the BGS idea one step further and solve the joint sub block with a completely different approach like the reduced coordinate formulation in Section 3.2. In the extreme case BGS can be used to partitioning a configuration into sub blocks where one can apply specialized solvers for each sub block. This has been termed hierarchical solvers by the graphics and gaming community.

4.3.4. A Staggered Approach

One may combine the ideas of splitting the LCP and use QP reformulations. The idea is referred to as staggering [Lot84, KSJP08]. We partition the LCP variables into three index sets, one corresponding to normal impulses $\mathcal{N}$, and one to friction impulses $\mathcal{F}$ and the last one is simply the slack variables β. Applying our partition would require us to solve the two coupled LCPs,

$$\mathbf{A}_{\mathcal{N}\mathcal{N}}\mathbf{x}_{\mathcal{N}} + (\mathbf{b}_{\mathcal{N}} + \mathbf{A}_{\mathcal{N}\mathcal{F}}\mathbf{x}_{\mathcal{F}}) \geq \mathbf{0} \quad \perp \quad \mathbf{x}_{\mathcal{N}} \geq \mathbf{0}$$

and

$$\begin{bmatrix} \mathbf{A}_{\mathcal{F}\mathcal{F}} & \mathbf{e} \\ -\mathbf{e}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{F}} \\ \beta \end{bmatrix} + \begin{bmatrix} \mathbf{b}_{\mathcal{F}} + \mathbf{A}_{\mathcal{F}\mathcal{N}}\mathbf{x}_{\mathcal{N}} \\ \mu\mathbf{x}_N \end{bmatrix} \geq \mathbf{0} \quad \perp \quad \begin{bmatrix} \mathbf{x}_{\mathcal{F}} \\ \beta \end{bmatrix} \geq \mathbf{0}.$$

Taking a staggered approach one solves the top-most LCP first (normal force problem) and then the bottom-most LCP second (the friction force problem) and continues iteratively until a fixed-point is reached. This is in fact a blocked Gauss–Seidel splitting method.

Observe that the normal force problem has a symmetric positive semi-definite coefficient matrix $\mathbf{A}_{\mathcal{N}\mathcal{N}}$ making QP reformulations possible whereas the frictional problem has an non-symmetric matrix. One may exploit a QP reformulation anyway. Because the friction LCP is the first order optimality conditions of the QP problem

$$\mathbf{x}^*_{\mathcal{F}} = \arg\min \frac{1}{2}\mathbf{x}^T_{\mathcal{F}}\mathbf{A}_{\mathcal{F}\mathcal{F}}\mathbf{x}_{\mathcal{F}} + \mathbf{c}^T_{\mathcal{F}}\mathbf{x}_{\mathcal{F}} \tag{83}$$

subject to

$$\mathbf{x}_{\mathcal{F}} \geq \mathbf{0} \quad \text{and} \quad c_{\mathcal{N}} - \mathbf{e}^T\mathbf{x}_{\mathcal{F}} \geq 0, \tag{84}$$

where $c_{\mathcal{N}} = \mu\mathbf{x}_{\mathcal{N}}$ and $\mathbf{c}_{\mathcal{F}} = \mathbf{b}_{\mathcal{F}} + \mathbf{B}_{\mathcal{F}\mathcal{N}}\mathbf{x}_{\mathcal{N}}$. Thus, any convex QP method can be used to solve for the normal and friction forces and one is guaranteed to find a solution for each subproblem. Whether the sequence of sub QP problems converge to a fixed point is not obvious.

There exist many variations over this staggering scheme [LL11]. For instance one variation is to use a blocked Gauss–Seidel method for the frictional problem rather than a QP reformulation. This is mostly due to performance. Keeping a QP solver for the normal problem helps getting accurate normal forces which are needed to deal with large mass ratios whereas accurate friction can be given up to some degree in interactive applications which means a Gauss–Seidel method is suitable for the friction problem.

4.4. Newton Methods

The PGS methods from Section 4.3 may suffer from viscous artifacts due to linear convergence rate. One remedy is to use Newton methods. These can provide quadratic convergence rates and thus offers more accurate solutions at a slightly higher per iteration computational cost than PGS methods. PATH [Pat05] is a well known Newton type solver for LCPs and used by many researchers in graphics and robotics. One drawback of PATH is that computing time scales quadratically in the number of contacts $\mathcal{O}(n^2)$. Here we will present a specialized Newton type solver and an open source implementation can be found in [Erl11].

The Fischer function is defined as

$$\phi(a,b) = \sqrt{a^2+b^2} - (a+b) \quad \text{for } a,b \in \mathbb{R}. \tag{85}$$

If one has the complementarity problem $a \geq 0 \perp b \geq 0$, a solution (a^*,b^*) is only a solution if and only if $\phi(a^*,b^*) = 0$. This may be proven by a case-by-case analysis of the signs of a and b. Now consider the LCP

$$\mathbf{x} \geq \mathbf{0} \quad \perp \quad \mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b} \geq \mathbf{0} \tag{86}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$ are given constants. Using the Fischer function the LCP may be reformulated as the nonsmooth root search problem

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x},\mathbf{y}) = \begin{bmatrix} \phi(\mathbf{x}_1,\mathbf{y}_1) \\ \vdots \\ \phi(\mathbf{x}_n,\mathbf{y}_n) \end{bmatrix} = \mathbf{0}. \tag{87}$$

Thus, our problem is changed to that of finding the root of a nonlinear nonsmooth equation. This problem may be solved using a generalized Newton method which is an iterative method. In the k^{th} iteration the Newton method solves the generalized Newton system

$$\mathbf{J}\Delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^k) \tag{88}$$

for the Newton direction $\Delta\mathbf{x}^k$. Here $\mathbf{J} \in \partial\mathbf{F}(\mathbf{x}^k)$ is any member from the generalized Jacobian $\partial\mathbf{F}(\mathbf{x})$. After having computed the Newton direction one performs a Newton update

to obtain the next iterate,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \tau^k \Delta \mathbf{x}^k. \tag{89}$$

Here τ^k is the step length of the k^{th} Newton direction. A line search method will be used to determine the value τ^k.

We will briefly introduce some definitions and theorems from nonsmooth analysis. Let $\mathbf{F} : \mathbb{R}^n \mapsto \mathbb{R}^n$ and let $\mathcal{D} \subset \mathbb{R}^n$ denote the set of all $\mathbf{x} \in \mathbb{R}^n$ where $\mathbf{F}$ is continuously differentiable. Assume $\mathbf{F}$ is Lipschitz continuous at $\mathbf{x}$ then the B-subdifferential of $\mathbf{F}$ at $\mathbf{x}$ is defined as

$$\partial_B \mathbf{F}(\mathbf{x}) \equiv \{\mathbf{H} \in \mathbb{R}^{n\times n} \quad | \quad \exists (\mathbf{x}^j) \subset \mathcal{D} \quad \text{and} \quad \lim_{\mathbf{x}^j \to \mathbf{x}} \nabla \mathbf{F}(\mathbf{x}^j) = \mathbf{H}\}.$$

Clarke's generalized Jacobian of $\mathbf{F}$ at $\mathbf{x}$ is defined as the convex hull of the B–subdifferential [Cla90],

$$\partial \mathbf{F}(\mathbf{x}) \equiv \mathbf{co}\left(\partial_B \mathbf{F}(\mathbf{x})\right). \tag{90}$$

As an example consider the Euclidean norm $e : \mathbb{R}^2 \mapsto \mathbb{R}$ then for $\mathbf{z} \in \mathbb{R}^2 \setminus \{\mathbf{0}\}$ we have

$$\partial e(\mathbf{z}) = \partial_B e(\mathbf{z}) = \nabla e(\mathbf{z}) = \frac{\mathbf{z}^T}{\| \mathbf{z} \|}; \ \forall \mathbf{z} \neq \mathbf{0}. \tag{91}$$

For $\mathbf{z} = \mathbf{0}$ we have

$$\partial_B e(\mathbf{0}) = \{\mathbf{y}^T \quad | \quad \mathbf{y} \in \mathbb{R}^2 \quad \text{and} \quad \| \mathbf{y} \| = 1\} \tag{92a}$$

$$\partial e(\mathbf{0}) = \{\mathbf{y}^T \quad | \quad \mathbf{y} \in \mathbb{R}^2 \quad \text{and} \quad \| \mathbf{y} \| \leq 1\}. \tag{92b}$$

For $\mathbf{z} = \begin{bmatrix} a & b \end{bmatrix}^T \in \mathbb{R}^2$ we write the Fischer function as $\phi(a,b) = \phi(\mathbf{z}) = e(\mathbf{z}) - f(\mathbf{z})$ where $f(\mathbf{z}) = \left(\begin{bmatrix} 1 & 1 \end{bmatrix}^T \mathbf{z}\right)$ is a everywhere continuous differentiable function. From this we find

$$\partial_B \phi(\mathbf{z}) = \partial_B e(\mathbf{z}) - \nabla f(\mathbf{z}) \tag{93a}$$

$$\partial \phi(\mathbf{z}) = \partial e(\mathbf{z}) - \nabla f(\mathbf{z}). \tag{93b}$$

Hence for $\mathbf{z} \neq 0$,

$$\partial \phi(\mathbf{z}) = \partial_B \phi(\mathbf{z}) = \left\{ \frac{\mathbf{z}^T}{\| \mathbf{z} \|} - \begin{bmatrix} 1 & 1 \end{bmatrix}^T \right\} \tag{94}$$

and

$$\partial_B \phi(\mathbf{0}) = \{\mathbf{y}^T - \begin{bmatrix} 1 & 1 \end{bmatrix}^T \quad | \quad \mathbf{y} \in \mathbb{R}^2 \text{ and } \| \mathbf{y} \| = 1\}$$

$$\partial \phi(\mathbf{0}) = \{\mathbf{y}^T - \begin{bmatrix} 1 & 1 \end{bmatrix}^T \quad | \quad \mathbf{y} \in \mathbb{R}^2 \quad \text{and} \quad \| \mathbf{y} \| \leq 1\}.$$

The Clarke generalized Jacobian of the Fischer reformulation (87) can be written as

$$\partial F(\mathbf{x}) \equiv \mathbf{D}_a(\mathbf{x}) + \mathbf{D}_b(\mathbf{x})\mathbf{A} \tag{96}$$

where $\mathbf{D}_a(\mathbf{x}) = \mathbf{diag}(a_1(\mathbf{x}), \ldots, a_n(\mathbf{x}))$, $\mathbf{D}_b(\mathbf{x}) = \mathbf{diag}(b_1(\mathbf{x}), \ldots, b_n(\mathbf{x})) \in \mathbb{R}^{n\times n}$ are diagonal matrices. If $\mathbf{y}_i \neq 0$ or $\mathbf{x}_i \neq 0$ then

$$a_i(\mathbf{x}) = \frac{\mathbf{x}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1, \tag{97a}$$

$$b_i(\mathbf{x}) = \frac{\mathbf{y}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1 \tag{97b}$$

else if $\mathbf{y}_i = \mathbf{x}_i = 0$ then

$$a_i(\mathbf{x}) = \alpha_i - 1, \tag{98a}$$

$$b_i(\mathbf{x}) = \beta_i - 1 \tag{98b}$$

for any $\alpha, \beta \in \mathbb{R}$ such that $\| \begin{bmatrix} \alpha_i & \beta_i \end{bmatrix}^T \| \leq 1$.

Proof: Here we will only show the case for $\mathbf{y}_i \neq 0$ or $\mathbf{x}_i \neq 0$. The differential of the i^{th} component is given by

$$d\mathbf{F}_i(\mathbf{x}, \mathbf{y}) = d\left(\mathbf{x}_i^2 + \mathbf{y}_i^2\right)^{\frac{1}{2}} - d\left(\mathbf{x}_i + \mathbf{y}_i\right). \tag{99}$$

Using the chain rule we have

$$\begin{aligned} d\mathbf{F}_i(\mathbf{x}, \mathbf{y}) &= \frac{1}{2}\left(\mathbf{x}_i^2 + \mathbf{y}_i^2\right)^{-\frac{1}{2}} d\left(\mathbf{x}_i^2 + \mathbf{y}_i^2\right) - d\mathbf{x}_i - d\mathbf{y}_i \\ &= \frac{\mathbf{x}_i d\mathbf{x}_i + \mathbf{y}_i d\mathbf{y}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - d\mathbf{x}_i - d\mathbf{y}_i \\ &= \left[\underbrace{\left(\frac{\mathbf{x}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1\right)}_{a_i(\mathbf{x})} \quad \underbrace{\left(\frac{\mathbf{y}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1\right)}_{b_i(\mathbf{x})}\right] \begin{bmatrix} d\mathbf{x}_i \\ d\mathbf{y}_i \end{bmatrix}. \end{aligned}$$

Finally, let $\mathbf{A}_{i\cdot}$ be the i^{th} row of $\mathbf{A}$ then we have $d\mathbf{y} = \mathbf{A} d\mathbf{x}$, so $d\mathbf{y}_i = \mathbf{A}_{i\cdot} d\mathbf{x}$ substitution of this results in

$$d\mathbf{F}_i(\mathbf{x}, \mathbf{y}) = \underbrace{\left(a_i(\mathbf{x})\hat{e}_i^T + b_i(\mathbf{x})\mathbf{A}_{i\cdot}\right)}_{\equiv \nabla F_i(\mathbf{x})} d\mathbf{x}. \tag{101}$$

The case $\mathbf{x}_i = \mathbf{y}_i = 0$ follows from the previous examples.

We can choose any element in the generalized Jacobian. If $\mathbf{x}_i = \mathbf{y}_i = 0$ we could choose $\beta_i = 1$ and $\alpha_i = 0$. Thus, resulting in using the negative i^{th} unit axis vector as the i^{th} row of $\mathbf{J}$. A more practical implementation approach would simply consist in whenever $\mathbf{x}_i = \mathbf{y}_i = 0$ one would use $\mathbf{x}_i' = \mathbf{x}_i + \varepsilon$ inplace of $\mathbf{x}_i$ when evaluating the generalized Jacobian where ε is a sufficiently small value.

A line search method is often used to achieve global convergence of the Newton method. We propose a backtracking line search with an Armijo condition to ensure sufficient decrease and that the chosen step length is not too small [NW99]. The line search uses the natural merit function of $\mathbf{F}(\mathbf{x})$ as a measure of convergence. The natural merit function is defined as $\Psi(\mathbf{x}) = \frac{1}{2} \| \mathbf{F}(\mathbf{x}) \|^2$. The Armijo condition is given by

$$\Psi(\mathbf{x}^k + \Delta \mathbf{x}^k) \leq \Psi(\mathbf{x}^k) + c\tau^k \nabla \Psi(\mathbf{x}^k)^T \Delta \mathbf{x}^k \tag{102}$$

where the sufficient decrease parameter is $c \in (0,1)$ and

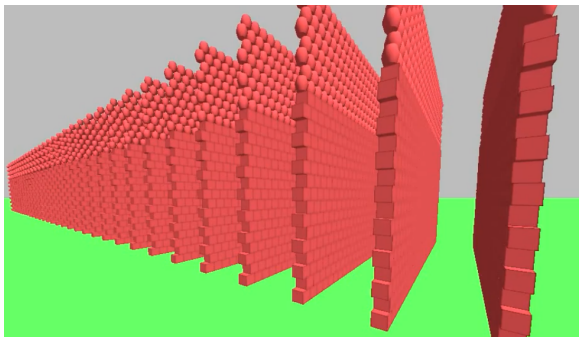


Figure 12: *A simulation of 100k rigid bodies running real-time on a Radeon 7970 GPU using OpenCL.*

the gradient of the merit function is given by $\nabla\Psi(\mathbf{x}^k) = \mathbf{J}^T\mathbf{F}(\mathbf{x}^k)$.

The objective of the line search method is to find a step length τ^k such that (102) is satisfied. The back tracking approach starts with the guess of $\tau^k = 1$ and then test if (102) holds. If not τ^k is reduced by a step reduction fraction and the test is repeated. This continues until the test passes and one will have obtained the final value τ^k.

5. Parallel Processing and Optimizations

Parallelization is an important topic since multi-core systems and massively parallel GPUs are very common today.

OpenMP (Open Multi-Processing) or MPI (Message Passing Interface) are often used for developing parallel applications for multi-core systems. OpenMP is designed for shared memory computers and provides a very simple and flexible interface for programmers. A programmer can use simple compiler directives in order to parallelize his code. MPI runs also on distributed memory architectures and can be used on a wider range of problems than OpenMP but it is harder to program. In general a MPI program consists of multiple processes that communicate by messages in order to solve a problem in parallel.

The parallel programming of GPUs is a far more complex task than programming a multi-core CPU. GPUs have a SIMD architecture since they were designed for rendering. The first parallel simulation methods on GPUs were implemented as shader programs which were executed in the render pipeline for each pixel or each vertex of a special scene. Such a scene had exactly the same number of pixels or vertices as required program executions. The data of the simulation had to be encoded as textures. The introduction of high-level languages for programming GPUs like OpenCL and NVIDIA's Compute Unified Device Architecture [Khr11, NVI11] made General Purpose Computation on Graphics Processing Unit (GPGPU) more interesting for the community. A GPU program (also called *kernel*) can access different kinds of memories with different sizes and different performance characteristics. Therefore, the memory access and memory layouts play an important role for getting a high performance. OpenCL and CUDA provide access to global, local and shared memory. The interaction between CPU and GPU is also important since memory transfers between both are costly. The number of kernel calls also influence the performance significantly. Therefore, it is desirable to reduce the number of calls to a minimum. Since the parallelization on a GPU is not straightforward, efficient data structures and algorithms are required that are optimized for parallel rigid body simulations.

For the simulation of bilateral constraints one has to solve a system of linear equations (see Section 3.2). This can be done in parallel by using a solver like PARDISO [SG04] which is optimized for multi-core processors. Alternatively, there exist multiple methods for solving such a system on the GPU. Bolz et al. [BFGS03] as well as Krüger and Westermann [KW03] used shader programs and special textures to implement different parallel solvers on the GPU. Optimized data structures for sparse matrix operations on the GPU have been developed in [BG09] and [BCL09]. For achieving a high performance a good memory layout of these structures is very important. The optimized matrix operations allow the efficient solution of sparse systems which generally occur in multibody simulations with bilateral constraints. Another approach was presented by Bayer et al. [BBD09]. They create groups of independent constraints in a precomputation step. Then, all constraints in a group can be solved independently from each other. This is done in parallel using different pixel shader programs. The dependencies between the groups are resolved by a Gauss-Seidel iteration approach. A similar approach was used in [BB08].

For the computation of contact forces we have unilateral constraints in the simulated multibody system. Since we have inequalities in this case, the unilateral constraints cannot be solved by a linear solver. Therefore, the parallel computation of contact forces was also a research topic of interest in the last years.

Harada [Har08] used rigid bodies that are represented by sets of particles. This representation makes a parallelization of the collision detection and response very simple. For the collision detection each particle is represented by a sphere which results in an efficient detection due to a very simple collision test. The accuracy and the performance of the collision detection directly depend on the resolution of the particle representation. For the collision response Harada used a discrete element method (DEM) where a repulsive force, a damping force and a shear force are computed for each colliding particle.

Tasora et al. [TNA08, TNA*10] used a Cone Complementarity Problem (CCP) formulation instead of a classical LCP solver with a polyhedral approximations of the friction cone in order to parallelize the contact problem. One

of the challenges when working on the GPU is to avoid concurrent updates of shared data. Due to the high latency on GPU memory access *global atomic operations* can be computational costly. Tasora et al. argued that the probability for a concurrent velocity update of contacts associated with the same body is very small for large scenarios with hundreds of thousands of contacts. Harada showed how to efficiently solve this problem by partitioning, synchronizing and scheduling the operations using *local atomics* within each *compute unit* [Har11]. An open source implementation as shown in Figure 12 is available as part of a rigid body simulation pipeline running entirely on the GPU for the Bullet physics engine [Cou12].

Courtecuisse and Allard [CA09] introduced a parallel Gauss-Seidel iteration method for dense matrices. Their method works on multi-core processors and GPUs. It maintains the invariant that in each block row, the diagonal element is the last to be updated. This is used to schedule the block computations, eliminating the need for global synchronization.

The research in the area of parallelization shows us that the performance of simulations can be increased significantly taking advantage of multi-core processors and GPUs. But this performance gain is not achieved straightforward, it demands a computational rethinking of the used algorithms.

6. Collision Detection for Rigid Body Dynamics

Collision detection provides important information used by rigid body dynamics. We briefly discuss the most relevant shape representations, collision detection queries and contact generation methods. A more complete overview of the field is available in the collision detection surveys [LG98, JTT00] and books [Eri04, Ber04]

6.1. Shape Representations

The geometry type of simulated rigid bodies is important for collision detection and contact point generation. It significantly influences the performance and the complexity of the simulation system. This section will discuss the role of geometry in simulations.

Structured polygonal models are very popular in the graphics community. There exist many tools and efficient algorithms for this type of geometry. These models are also popular in the field of interactive rigid body simulations. The mass properties of a polyhedral body can be determined fast and accurately [Mir96a]. Furthermore, there exist different very fast collision detection methods which only work for closed convex polygonal meshes.

Since fast collision detection methods are essential for an interactive simulation, the usage of *convex polygonal models* is often required in this area. Therefore, non-convex shapes have to be decomposed in convex parts in a precomputation

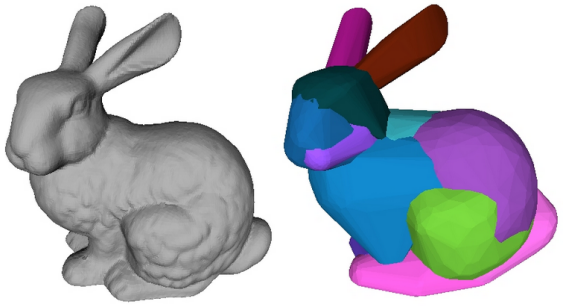


Figure 13: *Illustration of a convex decomposition of the surface of a polygonal model. The decomposition is used in place of the real geometry in interactive simulation.*

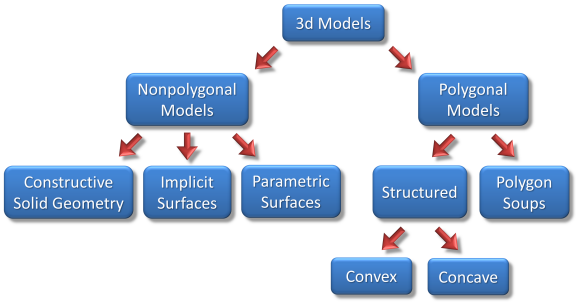


Figure 14: *Collision Shape Taxonomy*

step (see Figure 13) There are several ways to decompose a closed non-convex polygonal mesh into a *convex decomposition*. The decomposition can be generated either manually or automatically [MG09].

Non-moving concave world geometry is often represented as a concave triangle mesh, and collision queries are performed on individual triangles. As mentioned before, *mid phase* acceleration structures can be used to cull most triangles.

6.2. Collision Queries

The exact collision queries between two objects are known as *narrow phase* collision detection. The choice of algorithm and complexity of the query depends on the collision shape representation of the objects involved (see Figure 14). Aside from the shape type, we can classify queries into *discrete* and *continuous queries*. *Discrete methods* perform the collision check at a specific time instant, while *continuous collision detection* (CCD) methods take the motion of the objects into account over a time interval.

Discrete Collision Detection Various discrete collision queries exist, ranging from a simple intersection test to full

contact information generation. A *discrete intersection test* produces a boolean result that determines whether collision shapes overlap or not. When using a simulation loop with an adaptive timestep, the intersection test can be used to search for the time of impact using a technique called bisection.

When objects are separated by a positive distance, we can compute this *closest distance* and the *closest points*, also known as witnesses. The GJK algorithm [GJK88, Ber04] is versatile and it has been used for different queries between convex shapes. GJK can be used to perform an intersection test, and when objects are separated by a positive distance, it can compute the distance and the corresponding closest points (one on each body).

If objects are overlapping, we can compute the *penetration depth*. A common way to define penetration depth is the shortest relative translation of the objects to eliminate the overlap. In addition to the penetration depth vector, we can compute *witness points* on both objects where the object will touch. The penetration depth between general convex shapes can be computed using the *expanding polytope algorithm* [Ber04], while the *separating axis test* (SAT) can be used between convex polyhedra. Concave shapes can be represented as a union of convex shapes (see Section 6.1). Alternatively, a collision check is performed for each triangle in a concave triangle mesh (or triangle soup).

A single contact point pair is often not sufficient for stable resting contact in rigid body dynamics. Section 6.3 will provide more information about contact point generation.

Continuous Collision Detection Discrete collision checking algorithms can fail to detect a collision due to *temporal aliasing*, which commonly occurs with fast moving or small objects that can pass completely through an object in one time step. To avoid this, we can take the motion into account for a certain time interval and compute the time of impact. This information can be used to subdivide the simulation timestep, or it can be used to formulate contact and distance constraints to prevent penetration (see Section 3.1).

Mirtich [Mir96b] determined a lower bound for the time of collision for each pair of bodies. These times are stored in a heap which has to be updated after each collision since the bodies then change their motion. The minimum collision time of a pair of bodies describes how long a simulation can run at least without a penetration occurring. The detection is accelerated by bounding volumes which take the ballistic motion of the bodies into account.

Continuous collision detection methods approximate the motion of the bodies during a time step. The collision detection is performed on the resulting trajectories. Different methods were introduced to perform the motion approximation for unconstrained bodies, which are based on interpolation [RKC00, RKC02, KR03]. In contrast, Redon presented a continuous collision detection which is designed for articulated bodies [RKLM04, ZRLK07]. Continuous collision detection has the advantage that no collision is missed but at the price of a higher computational complexity.

6.3. Contact Point Generation

The quality of the contact point information greatly influences the overall robustness and stability of an interactive simulator. Since all subsequent contact force and stabilization computations are affected by the quality of the contact point information, the requirements of the method for contact point generation are: high performance, robustness and consistency.

Collision detection methods like the GJK algorithm often return only one pair of points representing either the minimum distance or the maximal overlap distance. For accurate collision resolution and stable resting contact handling between rigid bodies, we need more than a single contact point in general. In practice, post processing is often done to generate the complete contact region between objects. However, having too many contact points between two rigid bodies can cause performance and stability issues.

Feature based contact point generation was among the first approaches [Bar90], it has since been extended to cover continuous collision detection [SMT08]. Basically, contacts are represented by either edge-edge or vertex-face feature pairs. Since edge-face feature pairs correspond to penetrations, these are used when objects are overlapping. Feature based contacts are local definitions and might show inconsistencies on a global scale. Further, some methods have a tendency to generate redundant feature pairs [CTM08]. Collision envelopes are used to ward off numerical imprecision, round-off and truncation errors in floating point arithmetic. When using the feature based contact point generation, ones method needs access to the actual features of the mesh representation.

Another approach for generating a contact region is to track contact points over time. In each time-step, new contact points are added to a region while filtering out old ones that no longer agree with the current contact plane. Contacts can be tracked based on features between convex polyhedra [Mir98]. Feature information is not always available, so to track contacts between general convex objects a heuristic based on the distance between closest points can be used [Cou05].

For convex polyhedra an approximation of the entire contact region can be computed at once. In the case of two colliding bodies, first, a separating plane is determined. The collision geometry can then be clipped and projected onto the separating plane. The contact region is determined by intersecting the resulting convex polygon in the plane [BG10].

Signed distance fields have been popular for deformable models [MAC04], cloth [BMF03] and rigid bodies [GBF03]. According to Erleben [Erl05], signed distance fields add a

certain smoothness to the contact point generation, which avoids many of the difficulties in choosing contact normals and computing penetration depths. On the other hand, the smoothness is related to the resolution of the distance field and can be troublesome for stacking configurations. The memory footprint can make signed distance fields intractable for interactive simulations. Other approaches using discrete Voronoi diagrams also exist [SGG*06], these tend to be similar to signed distance fields methods, apart from using a Voronoi diagram as the basic representation.

A general issue in contact point generation, is choosing between global and local solutions [KOLM02]. In theory, contact point generation is a global issue, however, in practice the local solution is often used to satisfy performance considerations. There is also some discussion on how a good penetration depth measure is defined. For convex polytopes the generalized penetration depth is same as the translational penetration depth [ZKVM06].

7. Rigid body dynamics in practice

The literature on rigid body dynamics, robotics and contact mechanics are vast, cross-disciplinary and have a long history. Thus, several attempts have been made in the past to classify previous work in order to make differences more clear to the communities. In this section, we shortly review some of the terminology that has been used in the past.

Furthermore, we want to give an overview over existing commercial and open source simulation software as well as a classification for this software. We also want to give a survey of benchmark papers in computer graphics and discuss the common practice regarding benchmarking and validation of simulators.

7.1. The Simulation Paradigms and Contact Models

In the terminology of Moreau [Mor99] one may classify simulation methods as being event-driven, smoothing, or contact dynamics approaches. *Event-driven approaches* are classified by models where motion in between events are assumed to be sufficiently smooth and not changing too much. This can be understood in the sense that the contact regions between objects are non-changing and reaction forces do not change direction. Changes then only occur at specific single events in time and must be dealt with specifically at these events. An example of such a method could be one that assumes that interactions only consist of instantaneous collisions. Like the impulse-based model of Hahn [Hah88], and Mirtich and Canny [MC95, Mir96b]. Here it is assumed that objects are in free ballistic flight in between collisions. Another example is the acceleration-level based formulations like the ones in Baraff's work [Bar89, Bar90, Bar93a, Bar93b, Bar94, Bar95, Bar96]. Here, the equation of motion is formed and treated as second order ordinary differential equation – assuming the motion is continuous differentiable. Discontinuous instantaneous changes in velocities and accelerations must then be treated at specific events. *Smoothing approaches* essentially apply some kind of regularization, like replacing a nonsmooth non-penetration law by a stiff repulsion law. The smoothing can be applied both in time and space. For instance in the work by Moore and Wilhelms [MW88] springs are cleverly used to model both sustained contact as well as instantaneous collisions. *Contact dynamics approaches* are described by Moreau as time stepping algorithms that determine the evolution of the velocity function by applying the principles of dynamics and assumed force laws. This means that no concept of acceleration is needed and the detailed dynamics over a single time-step is treated and resolved in a one step manner. The later work by Stewart and Trinkle, Moreu, and Jean [ST96, Mor99, Jea99] are examples of velocity based formulations that apply a fixed time-stepping procedure to advance the simulation state. Recently, this type of methods is simply referred to as time-stepping schemes [Stu08].

Event-driven approaches are often not the preferred choice for interactive simulation. The reason being that it can be highly unpredictable how many events need to be processed before reaching the next frame in ones simulation. Thus, sometimes ones simulator appears to be fast and at other times it may even stagnate. For configurations with a lot of dynamic and fast moving objects that bounce around an event-driven approach can be very efficient. However, for large piles or stacks of objects undergoing some transient salient motion the rate of events can explode and stagnate the simulation. This is one of the reasons why fixed time-stepping schemes are preferred as they always take one step ahead in time no matter what the interaction is. The computational cost of a time-stepping scheme often scales in the number of constraints. However, with iterative methods this scaling can be as fast as linear and often the number of iterations can be bounded yielding a fast simulator with a highly reliable predictable performance. The smoothing approach bare some similarity to the penalty-based paradigms which we cover later and suffers from the same difficulties.

Baraff [Bar93b] applies the terms *Continuous methods* (originally Baraff termed this continuum methods but community seems to have converged on the term continuous collision detection [vdB05, ZRLK07]) and *discrete methods* for dealing with the numerical time aspect of collision detection and contact point generation methods. The discrete setting can be thought of as taking a photograph and compute all geometric and physical information from that time instant. In such an approach one really does not know what occurs between two consecutive discrete points in time. The continuous methods on the other hand resolves what occurs over time intervals. Baraff describes two groups of simulation methods for dealing with constraints. One is termed *constraint-based methods* and the other is termed the *penalty methods*. In the first group constraint forces are solved for

analytically such that they exactly fulfill the constraints of the system. In the second method constraints are rephrased as penalty functions in an optimization sense [NW99]. Later Baraff [Bar94] adopted the term *analytical methods* to describe methods that compute contact/constraint forces in an analytical setting (like solving a linear system of equations or an linear complementarity problem) that fulfills the imposed constraints. These terms are essentially a classification of which type of numerical method that is used to find a solution for a system of constraints whether that is expressed as a linear system of equations or a more complex mathematical formulation like a complementarity problem formulation. More recently people make the distinction between *direct methods* and *iterative methods* [Stu08, KSJP08, BDCDA11]. Again, this is a classification of the numerical method applied.

Early work tended to use direct methods based on pivoting for solving complementarity problem formulations [Bar94, ST96, AP97b]. For interactive simulations it was quickly recognized that these type of numerical methods scaled too poorly although they were accurate. To deliver a fast performance that scales well iterative methods have been employed. In particular Gauss–Seidel like methods [Mor99, Jea99, Erl07, Stu08, CA09] have been investigated. Proper exploitation of matrix factorizations allow these type of iterative methods to scale linearly in the number of constraints. The poor convergence of Gauss–Seidel type solvers have been countered by Newton-type algorithms [AC91, Ort07, EO08, SNE09, BDCDA11] that offer a theoretical second order convergence rate over the linear rate of Gauss–Seidel type solvers. Linear scaling can be obtained for Newton-type methods resulting in Quasi-Newton methods this sacrifices the convergence rate though. A well-known Newton type solver is the PATH solver [FM99, Pat05]. One downside of PATH is that it needs a global coefficient matrix and for that reason it scales quadratic in the number of constraints.

Other authors refer to models that are based on the dynamics and assumed force laws formulated as constraints as *constraint-based paradigms* and make the distinction of whether they are formulated on a position, velocity or acceleration based level [ST96, AP97b, CR98, MS01, KEP05, Erl05, TNA08, KSJP08, TNA*10, BDCDA11]. This type of paradigm shares some similarity traits with the contact dynamics approaches of Moreau and the constraint-based methods of Baraff. Constraint-based paradigms are often further subdivided into being *maximal or reduced coordinate formulations*. This refers to whether knowledge of joint constraints are used to remove unneeded degrees of freedom from ones system of equations. If such action is taken, a reduced formulation is created containing a smaller number of variables, hence the term "reduced". Maximal coordinate formulations on the other hand do not reduce the number of variables but rather keep joint constraints as an extra set of equations that must be fulfilled. The work of Armstrong and Green, and Featherstone [AG85, Fea87] are examples of numerical methods that in a recursive manner very efficiently finds solutions to a reduced coordinate formulation whereas Baraff [Bar96] is an example that works with a maximal coordinate formulation where sparsity pattern of the first-order optimality conditions (known as the KKT-matrix [NW99]) is exploited to find solutions for Lagrange multipliers in linear time. The sparsity pattern arises from tree-like jointed mechanism. The term *penalty-based paradigms* seems to have converged on the meaning that some type of repulsive force is used to penalize violations of constraints or penetrations. Recent work tend to compute penalty forces based on volume violation, i.e. the actual overlapping volume [AFC*10]. In the same spirit *impulse-based paradigms* refers to models that approximate continuous contact with a series of instantaneous contacts [Mir96b, Mir00, GBF03]. Thus, the above simulation paradigms each classify the underlying model of the physical interaction as being based on penalty forces, collision impulses (i.e. instantaneous impacts) or some simultaneous mathematical formulation of the whole system.

Penalty-based paradigms are notoriously hard to work with, since it requires extensive parameter tweaking to perform optimally. Physical plausibility is hard to achieve with this paradigm, in part because collisions are never solved exactly, making stable stacking nearly impossible to simulate. The impulse-based paradigms is simple to implement, however, stable stacking is often difficult to achieve. This has been improved upon in later work with a technique of shock-propagation [GBF03]. The constraint-based paradigm has become the paradigm of choice in many interactive rigid body simulators [Smi00, Cou05] as it offers both great control and stability.

Maximal coordinate formulations are in computer animation dominated by complementarity formulations. There exist alternatives on kinetic energy [MS01] and motion space [RKC03]. However, the former solves a more general problem but is not attractive for performance reasons, and the latter is of limited use for realistic animation since it does not include friction. Recently Kaufman et al. [KEP05] presented a velocity-based method using projections onto convex subspaces of feasible velocities. The authors used a contact model, which is based on limit surfaces and principle of maximum dissipation [GRP89] , together with an ad-hoc model for bounciness and an approximation of momentum conservation. Kaufman et al. [KSJP08] also explored an iterative staggered approach for solving a velocity-level linear complementarity problem for contact problems by splitting the solver iterating into a normal force only solve followed by a friction only solve phase. Neither the 2005 nor the 2008 work was for interactive simulation. Recently, Newton type methods have been explored [BDCDA11] which apply a blocking strategy for solving the Newton system. The model is very similar to original work by Alart and Curnier [AC91]. The focus in this work is contact problem for hair and not rigid bodies, in fact the solver has problems dealing with the often overdetermancy

and large mass ratio properties encountered in rigid body dynamics. Complementarity formulations come in two flavors: acceleration-based formulations [Bar94, Bar95] and velocity-based formulations [ST96]. Acceleration-based formulations cannot handle collisions, and one must stop at the point of collision and switch to a impulse-momentum law [BWAK03,PW96,AP97b,Cha99]. Further, acceleration-based formulations suffer from indeterminacy and inconsistency [Ste00]. Although mostly overlooked in the computer graphics literature, the velocity-based formulation suffers from none of these drawbacks.

One other way to classify methods is by examining the underlying assumptions applied in their models of contact. For instance many impulse-based simulators apply a *sequential (or propagating) contact model*. Here, a local contact model of what happens during an instantaneous collision at a single point of contact between two rigid bodies is applied in a one-by-one sequential manner. Whereas many complementarity-based formulations take a more global view and use a *simultaneous contact model* that describe how the dynamics is coupled through multiple contacts between multiple objects [PG96,CR98,Mos07]. Impulse-based methods are examples of sequential models whereas complementarity problem formulations are simultaneous models of contact. One can further distinguish a contact model as being *a hard (also called nonsmooth) or smooth contact model*. The complementarity constraints used for non-penetration constraints are examples where a hard contact model is applied whereas the penalty force formulation is an example of an application of a smooth model. Of course one may add compliance or regularization to a hard contact model making it more smooth. It is often easier to model propagation/wave effects using local contact models or soft contact models whereas nonsmooth constraints often rely on simultaneous contact models and disregard any kind of propagation/wave effect.

A real-life working rigid body simulator is often not limited solely to one type of paradigm or contact model. Often things are combined in an ingenious and careful manner. Each piece of a simulator combats different artifacts and helps ensuring robustness and stability of the simulator. For instance position-level constraint-based formulation may be used to compute projections of rigid bodies in order to remove penetration errors or a penalty-based paradigm may be used to stabilize discretization errors of the ordinary differential equations as they evolve the system state in time. For example joint drifting in maximal coordinate formulations are countered through stabilization terms acting much like penalty forces. Compliance is added to nonsmooth contact models by adding a penalty force term making pure rigid bodies behave as quasi-rigid bodies.

For interactive simulators some common trends appear to be velocity-based constraint-based paradigms using fixed time-stepping methods. Reduced coordinate formulations are also widely popular for character animation as these by "design" do not visually appear to suffer from discretization errors and usually can deal with high speed moving limbs of a character.

7.2. Commercial and Open Source Software Solutions

There exist many open source alternatives like Bullet, Open Dynamics Engine (ODE), Newton Game Dynamics, daVinci code (dVC3d), Dynamo, dynamY, LMGC90, Jingine, Box2D, OpenTissue, IBDS as well as more commercial alternatives such as Vortex from CMLabs, PhysX from NVIDIA, Havok or Algoryx.

Most of these are multi-purpose physics engines and usually implement several methods: several different constraint solvers, friction models, collision detection method etc. ODE for example has both a pivoting/direct method and iterative methods such as blocked projected Gauss–Seidel (PGS). Bullet have similar iterative methods and is exploring nonsmooth nonlinear conjugate gradient methods as well. Recently many has focused on using GPUs in their simulators such as ChronoEngine, SOFA, PhysX and Bullet. The algorithmic choices are still based on iterative methods in these works.

Most game physics engines use a constraint solver based on iterative methods like the ones known from ODE and Bullet. The methods are confusingly named "sequential impulses" by the computer gaming community but the models are based on the constraint based paradigm and are solved using an iterative method such as PGS or similar. Although the constraint based paradigm has rooted itself as a very dominant method there are examples of other types of rigid body simulators. SIMPACK is such an example using a penalty based paradigm based on computing contact forces from volume overlaps. One difficulty of the penalty based approaches is often that they have many parameters and stability can be hard to come by at times this makes one abandon using physical meaningful parameter values. Examples of the impulse based paradigm and reduced coordinate method may be found in DynaMech.

The robotics community have made good use of rigid body simulators and large open source simulation frameworks such as Gazebo or Weebots are build on top of the ODE simulator. Commercial alternatives also exist like Microsofts Robot Simulator that utilities the PhysX engine. The frameworks offer a large library of existing joint and motor models as well as different kinds of controllers. Thus, allowing robot designers to test ideas and do off line programming of their control algorithms. Very detailed contact simulations can be done with simulators such as Adams from MSC software. These are often based on finite element methods and penalty based paradigms and are often not even near to interactive simulation.

Many authoring tools like Blender, Maya, Cinema 4D, LightWave, Houdini, 3ds Max, Autocad etc. offer rigid body

simulation as their features. Most of such tools even offer a variety of plug ins allowing users to pick and choose between which rigid body simulator they wish to use. Render engines like Ogre3D provides plug-ins for most of the popular rigid body simulators. The need for being able to switch one simulator for another has motivated initiatives such as COLLADA which is an XML scheme for describing rigid body physics in digital content and open PAL (Physics Abstraction Layer) which offers a common application programming interface for many rigid body simulators.

7.3. Benchmarking and Validation of Simulators

Few examples of benchmark tests and simulator comparison tests exist in the computer graphics literature [LH00a, LH00b, AS06, BB07, WM09, WSM*10]. These limit themselves to case-by-case studies comparing simulated results against analytical results or by feature list comparisons. No benchmark databases exist with both simulation setups and ground truth data. Looking at experiments and results from research papers several tendencies about common practice can be extracted. In summary, computer graphics rarely perform temporal convergence studies, rather focus is on creating stable and robust simulators that work with large time steps. Thus, there is little point in examining what happens if the time-step size goes to zero. In a similar fashion constraint errors are rarely examined in detail. From animation viewpoint this can be justified as a simulation may appear plausible as long as any constraint error is less than the size of a pixel and thus can not be seen by the naked eye. Computer graphics tends to care more about generality and robustness of a simulator than whether it can accurately reproduce a contact force to some given numerical precision. This is often exemplified by showing several simulation results from production environment like scenarios taking robustness and generality to the extreme. These are of course over-simplifying statements that would not be valid in all application areas of interactive rigid body simulation but holds for many cases of entertainment where focus is more on creating interesting motions than correct motions.

Statistics Many works settle with reporting the number of bodies, constraints, contact points and the frame time for selected test cases [Bar94, GBF03, WTF06, WGF08, SSF09]. Some work report more detailed wall-clock times as a function of frames [KEP05]. Others investigate convergence rates of iterative solvers [Lac03, Erl07]. Often single test cases are constructed showing how a simulator can deal robustly with different types of simulation [Mir96b, AS06, BB07]. In our view for interactive simulation it is often of interest not only to know average numbers, but also minimum, maximum and variance are important to figure out the range of applications a simulator can be used for.

Accuracy and Error Correction Piles of objects are very popular, and stable stacking like towers or card houses [GBF03, KEP05, WTF06, Erl07, KSJP08, AFC*10]. Examples are shown in Figure 15. Some works address scaling by increasing the number of objects or joints and reporting how computation time is affected by this [MS04, Ben07]. Some papers introduce test cases like a block sliding on a inclined plane which allows one to validate the friction law or visually identify creeping problems [Mir96b, SSF09]. Dense structured stacks are very good for verifying the accuracy of contact force computations and to determine if error correction is working properly [Erl07]. Dense stacks usually severely suffer from constraint overdetermancy (i.e. redundancy in contact point information) and therefore stress the numerical methods. Iterative methods often deal very well with overdetermancy whereas other methods suffer from the singularity that appears in the coefficient matrix due to the overdetermancy. Static structures like the card house or dry stone masonry structures like a stone arch are excellent for testing a simulator's ability to accurately reproduce static friction. In real-life it is basically static friction that holds these kind of structures in place [Erl07, KSJP08]. Dominoes falling down is also often used to test if friction makes the dominoes come to rest [Mir96b].

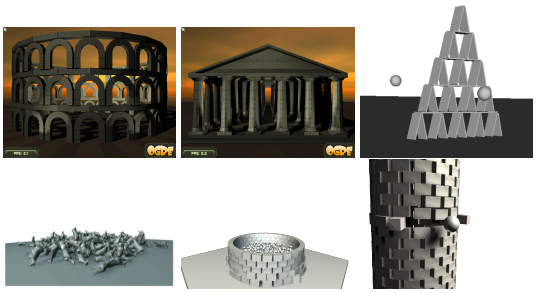


Figure 15: *Test-cases often used to test accuracy and error correction in interactive rigid body simulation.*

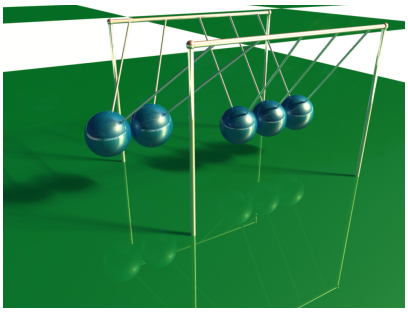


Figure 16: *The model of Newton's cradle is a test-case for a correct impulse propagation.*

Proper Event Handling Dynamic examples include the see-saw which is great for stressing event-driven approaches and showing impulse propagation [Mir96b]. The Newton

Figure 17: *Test-cases that can reveal problems with freezing (sleeping policies). If too aggressive bricks will hang unnaturally in the air.*

cradle (see Figure 16) can be used to determine whether a simulator's underlying model is a simultaneous contact model or a sequential contact model. Falling dominoes are often used as a test case for event-driven schemes [Mir96b]. Destruction of buildings as shown in Figure 17 can be helpful to test if sleeping policies (freezing) is too aggressive or even if it is present [Erl05]. Configurations with an in-build designed jamm (like billard balls in a racket) can be used to test whether simulators terminate in case they use a sequential contact model or if they can deal gracefully with errors and symmetries.

Numerical Methods Examples of objects being wedged have a tendency to stress the numerics. In particular if the setup is created in an illegal state. These types of tests mimic the effect of a real-life user doing "bad" things. The tests are thus helpful in determining if an interactive simulator is robust and stable. Large mass ratio tests are extremely good at stressing the numerical properties of simulators like a heavy box placed on top of a light box [SNE09]. Structured stacks exhibit some of these mass-ratio problems as the bottom-most objects feel the weight of all objects lying on top of them. Most iterative methods have difficulties dealing accurately with large mass ratios within a limited time budget or they converge so badly that even infinite amount of iterations won't help. Thus, large mass ratios are well suited as worst-case scenarios for the numerical methods.

Joints Vehicles or spinning wheels during a turning motion are stressful for most maximal coordinate formulations as turning the high speed wheel axis can make wheels numerically fly off. This is caused by discretization errors that are enhanced by applying large-time steps and using infinite orientations. Most engines have special fixes for this well-known problem. To stress joint drifting errors in maximal coordinate formulations large mass objects can be connected through joints to an articulated figure that is pinned down. Much like a prisoner in a jail with a ball and chain attached to the legs.

Gyroscopic Forces The tippe top and rattleback (see Figure 18) are famous test cases for testing if gyroscopic forces are working as expected in a simulator. Many game engines have a tendency to simplify the gyroscopic forces as they are a major cause to numerical stability problems prohibiting large time-steps. Thus, a case such as the tippe top can be used for identifying these problems.

Figure 18: *Rattleback (left) and tippe top (right) are good test cases for correct handling of gyroscopic forces.*

Interactivity Performance is critical to achieve interactive interaction. Thus, scalability testing is very interesting to see how large problems one can deal with at interactive rates. Robustness in interactive applications is more difficult to verify. Goal oriented tasks like stacking objects and tipping the stack without making it fall down have been used for simulators running under extreme conditions to test robustness towards user interaction as shown in Figure 19 and Figure 20.

Generality and Robustness Often simulations of complex scenes or interactions like interlocked gears or moving belt tracks are used to demonstrate generality and robustness of a simulation method [WGF08, SNE10a]. Complex geometry is often of interest in particular from a production viewpoint where concave surfaces and/or sharp features of the geometries may be encountered. However, from a contact mechanics viewpoint most contact laws being used are planar thus computing any contact forces at any given discrete instant in time is completely independent of the curvature of the surface. It is only when time-stepping comes in play that things may go wrong as surfaces may have higher "order" than the contact laws and time-integration methods being used. For fixed-time-stepping methods it is often not meaningful to use higher order integration schemes due to the fact that the contact laws are planar. Letting a box slide down a curved slope can be a good test case for illustrating these kind of problems. Some illustrations of complex scenes are shown in Figure 21.

Figure 19: *Test-cases for interactive testing. Bricks are pulled rapidly out of a wall without making the wall collapse.*

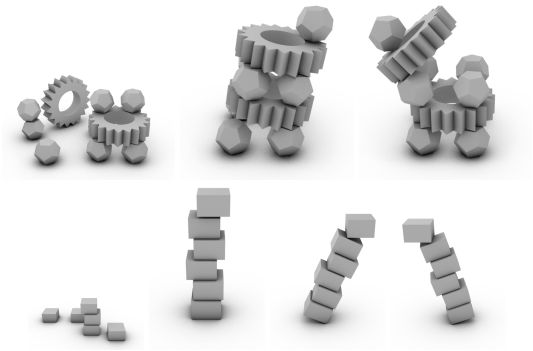


Figure 20: *Goal oriented task based testing for interactivity. Stacks of objects are to be created and tipped over without falling down.*

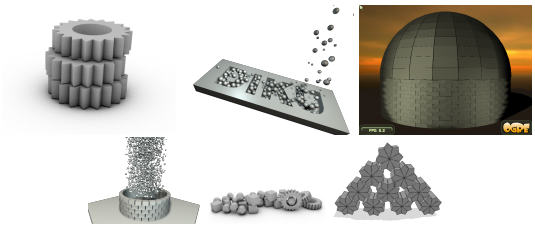


Figure 21: *More complex interacting geometry showing robustness.*

Physical Correctness Time plotting mechanical energy as well as kinetic and potential energy can help analyze physical correctness of ones simulator. Time plots of linear and angular momentum can be useful in this manner [AS06]. This allows one to validate if the physical conservation laws are fulfilled. One may test physical laws from simple physical test systems. For instance for impact laws one may test the proper effect of setting the coefficient of restitution. Sliding friction can be validated using for instance a box sliding down a plane. Inertia can be tested by spinning objects and verify if the proper axis of rotation is reached. In many cases one can from sufficient simple tests derive analytical results to compare against simulated results. For interactive simulators it can be quite useful to compare against a non-interactive but very high-fidelity simulator to confirm that similar simulation results are obtained.

Performance and Numerical Properties Many works list performance measurements as average frame times or frames per second achieved for a small portfolio of test cases [Bar94, GBF03, WTF06, WGF08, SSF09]. There are examples of papers doing complexity analysis and comparing against scalability studies, i.e. plotting computing time as a function of problem size [RGL05, PNE10, SNE10b]. For multi-core approaches speedup factors and floating point operations per second are plotted [CA09]. It is however not always clear what is the base reference that is compared against. In our view best practice should be an optimal tuned state-of-the-art solution. Detailed time-measurements of each sub-part of a simulator can be very helpful in analyzing performance bottlenecks too [KEP05]. Convergence plots of iterative methods are very helpful for determining a solvers ability to converge fast or slow [Lac03, Erl07]. Investigating how convergence error changes as a function of parameter choices (parameter studies) is a very methodical approach to find values suitable for robust interactive applications [BDCDA11]. Others compare against state-of-the-art competing methods [KSJP08, SNE10b].

Perception The saying that in computer graphics if it looks good then it is good is often stated by simulation people. The question is how one can quantify and measure if this is true. Although the literature is sparse, perception and user sensitivity studies have been performed [OD01, ODGK03, RP03, NLB*07, RO09]. Among other things these studies have shown that users find it hard to detect abnormalities in spinning objects and more difficult to detect abnormalities in collisions between complex objects than between simpler objects such as spheres. From these studies perception error metrics are developed. In principle such metrics could be used to validate if ones simulation results looks "good". However, simulation papers rarely do so.

8. Conclusion and Future Work

Interactive rigid body simulations have become an important part in different application areas. Such a simulation requires efficient and accurate methods for handling joint and contact constraints as well as a fast collision detection.

The simulation of more complex scenes and improvements of accuracy are current goals in this field. To reach these goals massively parallel GPUs and multi-core processors are taken into account. This parallelization trend requires a computational rethinking and provides the possibility to develop new efficient algorithms.

In the last years much research has been done on coupling of rigid body simulations with other animation and simulation techniques. One topic in this area is the combination of techniques like inverse kinematics with rigid bodies. Another important one is the coupling of rigid bodies with fluids [CMT04, RMSG*08, RMEF09], cloth and deformable bodies [SSIF07, SSF08]. Coupling allows the usage of different kinds of bodies in the same simulation environment by simulating the interaction between these bodies.

Simulation is a good way to generate realistic looking animations. But compared to keyframe techniques there is one big drawback. The results of a simulation can only be controlled indirectly by manipulating simulation parameters or adding forces to the system. Many physical parameters have

to be defined for a simulation. It is hard to reach certain predefined goals just by tweaking these parameters. Therefore, more control over the simulation is required. In order to solve this problem different methods have been developed which give a high-level control to the user. Some works propose inverse dynamics methods [PSE03, TJ08], others perform multiple simulations and discard unfitting ones [TJ07]. These methods let the user sketch a desired motion or define specific goals which must be reached by the simulation. But controlling the simulation is still a problem where much work has to be done.

References

[AC91] ALART P., CURNIER A.: A mixed formulation for frictional contact problems prone to newton like solution methods. *Comput. Methods Appl. Mech. Eng. 92* (November 1991), 353–375. 31

[ACPR95] ASCHER U. M., CHIN H., PETZOLD L. R., REICH S.: Stabilization of constrained mechanical systems with daes and invariant manifolds. *Journal of Mechanics of Structures and Machines 23* (1995), 135–158. 18

[AFC*10] ALLARD J., FAURE F., COURTECUISSE H., FALIPOU F., DURIEZ C., KRY P. G.: Volume contact constraints at arbitrary resolution. *ACM Trans. Graph. 29* (July 2010), 82:1–82:10. 31, 33

[AG85] ARMSTRONG W. W., GREEN M. W.: The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer 1*, 4 (1985), 231–240. 2, 31

[Alg11] Algoryx. `http://www.algoryx.se`, 2011. 4

[AP97a] ANITESCU M., POTRA F.: Formulating multi-rigid-body contact problems with friction as solvable linear complementarity problems. *ASME Journal of Nonlinear Dynamics 14* (1997), 231–247. 16

[AP97b] ANITESCU M., POTRA F. A.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics. An International Journal of Nonlinear Dynamics and Chaos in Engineering Systems* (1997). 31, 32

[AS06] AXEL SEUGLING M. R.: *Evaluation of Physics Engines and Implementation of a Physics Module in a 3d-Authoring Tool*. Master's thesis, Department of Computing Science, Umeå University, Sweden., March 2006. 33, 35

[Bar89] BARAFF D.: Analytical methods for dynamic simulation of non-penetrating rigid bodies. *SIGGRAPH Comput. Graph. 23*, 3 (1989), 223–232. 2, 30

[Bar90] BARAFF D.: Curved surfaces and coherence for non-penetrating rigid body simulation. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM, pp. 19–28. 29, 30

[Bar93a] BARAFF D.: Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica. An International Journal in Computer Science 10*, 2-4 (1993), 292–352. Computational robotics: the geometric theory of manipulation, planning, and control. 30

[Bar93b] BARAFF D.: Non-penetrating rigid body simulation. In *in State of the Art Reports, Eurographics '93*. Eurographics Association, Barcelona, Spain, September 1993. 1, 2, 30

[Bar94] BARAFF D.: Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994). 3, 21, 22, 30, 31, 32, 33, 35

[Bar95] BARAFF D.: Interactive simulation of solid rigid bodies. *IEEE Comput. Graph. Appl. 15*, 3 (1995), 63–75. 30, 32

[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. 17, 30, 31

[Bau72] BAUMGARTE J. W.: Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering 1* (1972), 1–16. 18

[BB88] BARZEL R., BARR A. H.: A modeling system based on dynamic constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM, pp. 179–188. 17

[BB07] BOEING A., BRÄUNL T.: Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia* (New York, NY, USA, 2007), GRAPHITE '07, ACM, pp. 281–288. 33

[BB08] BENDER J., BAYER D.: Parallel simulation of inextensible cloth. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Grenoble (France), Nov. 2008), pp. 47–56. 27

[BBD09] BAYER D., BENDER J., DIZIOL R.: Impulse-based dynamic simulation on the GPU. In *Computer Graphics and Visualization (CGV 2009) - IADIS Multi Conference on Computer Science and Information Systems* (Algarve (Portugal), 2009). 27

[BBZ91] BADLER N., BARSKY B., ZELTZER D.: *Making them move: mechanics, control, and animation of articulated figures*. Morgan Kaufmann series in computer graphics and geometric modeling. Morgan Kaufmann Publishers, 1991. 2

[BCL09] BUATOIS L., CAUMON G., LEVY B.: Concurrent number cruncher: a GPU implementation of a general sparse linear solver. *Int. J. Parallel Emerg. Distrib. Syst. 24* (June 2009), 205–223. 27

[BDB09] BAYER D., DIZIOL R., BENDER J.: Optimized impulse-based dynamic simulation. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Karlsruhe (Germany), Nov. 2009), pp. 125–133. 19

[BDCDA11] BERTAILS-DESCOUBES F., CADOUX F., DAVIET G., ACARY V.: A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies. *ACM Trans. Graph. 30* (February 2011), 6:1–6:14. 31, 35

[Ben07] BENDER J.: Impulse-based dynamic simulation in linear time. *Computer Animation and Virtual Worlds 18*, 4-5 (2007), 225–233. 18, 19, 20, 33

[Ber04] BERGEN G.: *Collision detection in interactive 3D environments*. The Morgan Kaufmann series in interactive 3D technology. Morgan Kaufman Publishers, 2004. 28, 29

[Ber09] BERARD S.: *Using Simulation for Planning and Design of Robotic Systems with Intermittent Contact*. PhD thesis, Rensselaer Polytechnic Institute, Department of Computer Science, 2009. 11

[BFGS03] BOLZ J., FARMER I., GRINSPUN E., SCHRÖDER P.: Sparse matrix solvers on the GPU: conjugate gradients and multigrid. *ACM Trans. Graph 22* (2003), 917–924. 27

[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. 18

[BG09] BELL N., GARLAND M.: Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (New York, NY, USA, 2009), SC '09, ACM, pp. 18:1–18:11. 27

[BG10] BERGEN G., GREGORIUS D.: *Game Physics Pearls*. A.K. Peters, 2010. 29

[BHW96] BARZEL R., HUGHES J. F., WOOD D. N.: Plausible motion simulation for computer graphics animation. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96* (1996), Springer-Verlag New York, Inc., pp. 183–197. 4

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 28–36. 29

[Bra91] BRACH R. M.: *Mechanical Impact Dynamics: Rigid Body Collisions.* John Wiley and Sons, New York, 1991. 9

[BS06a] 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. 19

[BS06b] BENDER J., SCHMITT A.: Fast dynamic simulation of multi-body systems using impulses. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Madrid (Spain), Nov. 2006), pp. 81–90. 18, 19, 20

[BWAK03] BARAFF D., WITKIN A., ANDERSON J., KASS M.: Physically based modeling. Siggraph Course Notes, 2003. 20, 32

[CA09] COURTECUISSE H., ALLARD J.: Parallel Dense Gauss-Seidel Algorithm on Many-Core Processors. In *High Performance Computation Conference (HPCC)* (jun 2009), IEEE CS Press. 4, 28, 31, 35

[CEA11] CEA LIST. `http://www-list.cea.fr`, 2011. 4

[Cha99] CHATTERJEE A.: On the realism of complementarity conditions in rigid body collisions. *Nonlinear Dynamics 20*, 2 (October 1999), 159–168. 32

[CK90] CASH J. R., KARP A. H.: A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. *ACM Transactions on Mathematical Software 16*, 3 (1990), 201–222. 21

[Cla90] CLARKE F.: *Optimization and Nonsmooth Analysis*. Society for Industrial Mathematics, 1990. 26

[CM 11] CM LABS: Vortex - behaviour in motion. `http://www.vxsim.com`, 2011. 4

[CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. 23* (August 2004), 377–384. 35

[Con93] CONTENSOU P.: *Kreiselprobleme und Gyrodynamics, IUTAM Symposium Celerina, 1962*. Springer-Verlag, Berlin, 1993, ch. Couplage entre frottement de glissement et frottement de pivotement dans la thÃl'orie de la toupie, pp. 201–216. 8

[Cou05] COUMANS E.: The bullet physics library. `http://www.bulletphysics.org`, 2005. 2, 29, 31

[Cou12] COUMANS E.: Coumans experiments. Published online at https://github.com/erwincoumans/experiments, January 2012. Open source experiments and research for the Bullet physics engine. 28

[CPN11] CPNET: Complementarity problem net. `http://www.cs.wisc.edu/cpnet`, 2011. 13

[CPS92a] COTTLE R., PANG J.-S., STONE R. E.: *The Linear Complementarity Problem*. Academic Press, 1992. 21

[CPS92b] COTTLE R. W., PANG J., STONE R. E.: *The Linear Complementarity Problem*. Academic Press, 1992. 11, 13

[CR98] CHATTERJEE A., RUINA A.: A new algebraic rigid body collision law based on impulse space considerations. *Journal of Applied Mechanics 65*, 4 (1998), 939–951. 9, 31, 32

[CTM08] CURTIS S., TAMSTORF R., MANOCHA D.: Fast collision detection for deformable models using representative-triangles. In *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2008), ACM, pp. 61–69. 29

[Cyb09] CYBERBOTICS: Webots 6. `http://www.cyberbotics.com/products/webots`, 2009. 4

[dJB94] DE JALON J. G., BAYO E.: *Kinematic and Dynamic Simulation of Multibody Systems: the Real Time Challenge*. Springer-Verlag, New York, 1994. 17

[EO08] ERLEBEN K., ORTIZ R.: A non-smooth newton method for multibody dynamics. In *ICNAAM 2008. International conference on numerical analysis and applied mathematics 2008* (2008). 31

[Eri04] ERICSON C.: *Real-Time Collision Detection*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. 28

[Erl05] ERLEBEN K.: *Stable, Robust, and Versatile Multibody Dynamics Animation*. PhD thesis, Department of Computer Science, University of Copenhagen (DIKU), 2005. 29, 31, 34

[Erl07] ERLEBEN K.: Velocity-based shock propagation for multibody dynamics animation. *ACM Transactions on Graphics (TOG) 26*, 2 (2007), 12. 31, 33, 35

[Erl11] ERLEBEN K.: num4lcp. Published online at code.google.com/p/num4lcp/, October 2011. Open source project for numerical methods for linear complementarity problems in physics-based animation. 25

[ESHD05] ERLEBEN K., SPORRING J., HENRIKSEN K., DOHLMANN H.: *Physics-based Animation*. Charles River Media, Aug. 2005. 5

[Fea87] FEATHERSTONE R.: *Robot dynamics algorithms*. Kluwer international series in engineering and computer science: Robotics. Kluwer Academic Publishers, 1987. 20, 31

[Fea07] FEATHERSTONE R.: *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. 19, 20

[FM99] FERRIS M. C., MUNSON T. S.: Interfaces to path 3.0: Design, implementation and usage. *Comput. Optim. Appl. 12* (January 1999), 207–227. 16, 31

[FO00] FEATHERSTONE R., ORIN D.: Robot dynamics: Equations and algorithms. *International Conference on Robotics and Automation* (2000), 826–834. 20

[GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Nonconvex rigid bodies with stacking. *ACM Trans. Graph.* (2003). 3, 20, 29, 31, 33, 35

[GJK88] GILBERT E. G., JOHNSON D. W., KEERTHI S. S.: A fast procedure for computing the distance between complex objects in three-dimensional space. *Robotics and Automation, IEEE Journal of 4*, 2 (1988), 193–203. 29

[Goy89] GOYAL S.: *Planar Sliding of a Rigid Body with Dry Friction: Limit Surfaces and Dynamics of Motion*. PhD thesis, Department of Mechanical Engineering, Cornell University, January 1989. 9, 11

[GPS02] GOLDSTEIN H., POOLE C., SAFKO J.: *Classical mechanics*. Addison Wesley, 2002. 5, 17, 19

[GRP89] GOYAL S., RUINA A., PAPADOPOULOS J.: Limit surface and moment funktion descriptions of planar sliding. In *Proc. of the 1989 IEEE International Conference on Robotics and Automation (Vol. 2)* (Scottsdale, AZ, 1989), pp. 794–799. 31

[Had06] HADAP S.: Oriented strands: dynamics of stiff multibody system. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), SCA '06, Eurographics Association, pp. 91–100. 20

[Hah88] HAHN J. K.: Realistic animation of rigid bodies. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (1988). 2, 30

[Har08] HARADA T.: Real-time rigid body simulation on GPUs. In *GPU Gems 3*, Nguyen H., (Ed.). Addison-Wesley, 2008, pp. 611–632. 27

[Har11] HARADA T.: A parallel constraint solver for a rigid body simulation. In *SIGGRAPH Asia 2011 Sketches* (New York, NY, USA, 2011), SA '11, ACM, pp. 22:1–22:2. 28

[INR11] INRETS: Driving simulator links. `http://www.inrets.fr/ur/sara/Pg_simus_e.html`, 2011. 4

[Jea99] JEAN M.: The non-smooth contact dynamics method. *Computer Methods in Applied Mechanics and Engineering 177*, 3–4 (July 1999), 235–257. 24, 30, 31

[JTT00] JIMÉNEZ P., THOMAS F., TORRAS C.: 3D collision detection: A survey. *Computers and Graphics 25* (2000), 269–285. 28

[KEP05] KAUFMAN D. M., EDMUNDS T., PAI D. K.: Fast frictional dynamics for rigid bodies. *ACM Trans. Graph. 24*, 3 (2005), 946–956. 31, 33, 35

[Khr11] KHRONOS: *The OpenCL Specification*, 2011. Version 1.2, http://www.khronos.org/opencl. 27

[KK11] K. KAPELLOS L. J.: Planetary exploration missions simulation using 3drov. Euromech colloquium on "Nonsmooth contact and impact laws in mechanics", Grenoble, France, July 6th-8th 2011. 4

[Kok04] KOKKEVIS E.: Practical physics for articulated characters. In *Proc. of Game Developers Conference (GDC)* (2004). 20

[KOLM02] KIM Y. J., OTADUY M. A., LIN M. C., MANOCHA D.: Fast penetration depth computation for physically-based animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), ACM, pp. 23–31. 30

[KP09] KOENG N., POLO J.: Gazebo, 3d multiple robot simulator with dynamics. `http://playerstage.sourceforge.net/index.php?src=gazebo`, 2009. 4

[KR03] KIM B., ROSSIGNAC J.: Collision prediction for polyhedra under screw motions. In *Proceedings of the eighth ACM symposium on Solid modeling and applications* (New York, NY, USA, 2003), SM '03, ACM, pp. 4–10. 29

[KSJP08] KAUFMAN D. M., SUEDA S., JAMES D. L., PAI D. K.: Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph. 27*, 5 (2008). 3, 25, 31, 33, 35

[KW03] KRÜGER J., WESTERMANN R.: Linear algebra operators for GPU implementation of numerical algorithms. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 908–916. 27

[Lac03] LACOURSIERE C.: Splitting methods for dry frictional contact problems in rigid multibody systems: Preliminary performance results. In *The Annual SIGRAD Conference* (November 2003), Ollila M., (Ed.), no. 10 in Linkøping Electronic Conference Proceedings. 21, 33, 35

[Lan86] LANCZOS C.: *The Variational Principles of Mechanics*. University of Toronto Press, 1986. 9, 10

[LG98] LIN M. C., GOTTSCHALK S.: Collision detection between geometric models: A survey. In *In Proc. of IMA Conference on Mathematics of Surfaces* (1998), pp. 37–56. 28

[LH00a] LANDER J., HECKER C.: Product review of physics engines, part one: The stress tests. *Gamasutra* (September 2000). `http://www.gamasutra.com/features/20000913/lander_01.htm`. 33

[LH00b] LANDER J., HECKER C.: Product review of physics engines, part two: The rest of the story. *Gamasutra* (September 2000). `http://www.gamasutra.com/features/20000920/lander_01.htm`. 33

[LL11] LACOURSIERE C., LINDE M.: *Spook: a variational time-stepping scheme for rigid multibody systems subject to dry frictional contact*. Tech. Rep. UMINF 11.09, Department of Computer Science, Umeå University, 2011. 25

[Lot84] LOTSTEDT P.: Numerical simulation of time-dependent contact and friction problems in rigid body mechanics. *SIAM Journal on Scientific and Statistical Computing 5*, 2 (1984), 370–393. 25

[MAC04] MARCHAL D., AUBERT F., CHAILLOU C.: Collision between deformable objects using fast-marching on tetrahedral models. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 121–129. 29

[Man84] MANDEL J.: A multilevel iterative method for symmetric, positive definite linear complementarity problems. *Applied Mathematics and Optimization 11*, 1 (February 1984), 77–95. 23

[MC95] MIRTICH B., CANNY J.: Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics* (New York, NY, USA, 1995), I3D '95, ACM, pp. 181–ff. 30

[Mei70] MEIROVITCH L.: *Methods of Analytical Dynamics*. McGraw-Hill, 1970. 5

[MF67] MANGASARIAN O., FROMOVITZ S.: The fritz-john necessary optimality conditions in the presence of equality and inequality constraints. *Journal of Mathematical Analysis and Applications 17* (1967), 37–47. 11

[MG09] MAMOU K., GHORBEL F.: A simple and efficient approach for 3D mesh approximate convex decomposition. In *Proceedings of the 16th IEEE international conference on Image processing* (2009), ICIP'09, pp. 3465–3468. 28

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (2007), 109–118. 20

[Mic09] MICROSOFT: Microsoft robotics. `http://www.microsoft.com/robotics`, 2009. 4

[Mir96a] MIRTICH B.: Fast and accurate computation of polyhedral mass properties. *J. Graph. Tools 1* (February 1996), 31–50. 28

[Mir96b] MIRTICH B. V.: *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California, Berkeley, 1996. 2, 20, 21, 29, 30, 31, 33, 34

[Mir98] MIRTICH B.: *Rigid Body Contact: Collision Detection to Force Computation*. Tech. Rep. TR98-01, MITSUBISHI ELECTRIC RESEARCH LABORATORIES, December 1998. 29

[Mir00] MIRTICH B.: Timewarp rigid body simulation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 193–200. 31

[Mor99] MOREAU J. J.: Numerical aspects of the sweeping process. *Computer Methods in Applied Mechanics and Engineering 177*, 3–4 (July 1999), 329–349. 24, 30, 31

[Mos07] MOSTERMAN P. J.: On the normal component of centralized frictionless collision sequences. *Journal of Applied Mechanics 74*, 5 (2007), 908–915. 32

[MS01] MILENKOVIC V. J., SCHMIDL H.: Optimization-based animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001). 31

[MS04] MILENKOVIC V. J., SCHMIDL H.: A fast impulsive contact suite for rigid body simulation. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (2004). 33

[MW88] MOORE M., WILHELMS J.: Collision detection and response for computer animation. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (1988). 2, 30

[MZS09] MACCHIETTO A., ZORDAN V., SHELTON C. R.: Momentum control for balance. *ACM Trans. Graph. 28* (July 2009), 80:1–80:8. 20

[NLB*07] NUSSECK M., LAGARDE J., BARDY B., FLEMING R., BÜLTHOFF H. H.: Perception and prediction of simple object interactions. In *Proceedings of the 4th symposium on Applied perception in graphics and visualization* (New York, NY, USA, 2007), APGV '07, ACM, pp. 27–34. 35

[NSE10] NIEBE S., SILCOWITZ M., ERLEBEN K.: Projected gauss-seidel subspace minimization method for interactive rigid body dynamics. In *Proceedings of the Fifth International Conference on Computer Graphics Theory and Applications* (Angers, France, May 2010), INSTICC Press, pp. X–Y. 23

[NVI11] NVIDIA: *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide*, 2011. Version 4.0, http://nvidia.com/cuda. 27

[NW99] NOCEDAL J., WRIGHT S. J.: *Numerical optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999. 18, 23, 26, 31

[OD01] O'SULLIVAN C., DINGLIANA J.: Collisions and perception. *ACM Trans. Graph. 20* (July 2001), 151–168. 35

[ODGK03] O'SULLIVAN C., DINGLIANA J., GIANG T., KAISER M. K.: Evaluating the visual fidelity of physically based animations. *ACM Trans. Graph. 22*, 3 (2003). 35

[Ort07] ORTIZ R.: *Newton/AMG algorithm for solving complementarity problems arising in rigid body dynamics with frictional impacts*. PhD thesis, University of Iowa, July 2007. 31

[PAGT05] POTRA F., ANITESCU M., GAVREA B., TRINKLE J.: Linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints, and friction. *International Journal for Numerical Methods in Engineering 66*, 7 (Dec. 2005), 1079–1124. 14

[Pat05] PATH: Path cpnet software, 2005. `www.cs.wisc.edu/cpnet/cpnetsoftware/`. 25, 31

[PFTV92] PRESS W. H., FLANNERY B. P., TEUKOLSKY S. A., VETTERLING W. T.: *Numerical Recipes: The Art of Scientific Computing*, 2. ed. Cambridge University Press, Cambridge (UK) and New York, 1992. 21

[PG96] PFEIFFER F., GLOCKER C.: *Multibody dynamics with unilateral contacts*. Wiley series in nonlinear science. John Wiley & Sons, inc., 1996. 32

[PNE10] POULSEN M., NIEBE S., ERLEBEN K.: Heuristic convergence rate improvements of the projected gauss-seidel method for frictional contact problems. In *Proceedings of WSCG* (2010). 23, 35

[PSE03] POPOVIĆ J., SEITZ S. M., ERDMANN M.: Motion sketching for control of rigid-body simulations. *ACM Trans. Graph. 22* (October 2003), 1034–1054. 36

[PT96] PANG J., TRINKLE J.: Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with coulomb friction. *Mathematical Programming 73* (1996), 199–226. 13

[PW96] PFEIFFER F., WÖSLE M.: Dynamics of multibody systems containing dependent unilateral constraints with friction. *Journal of Vibration and Control 2*, 2 (1996), 161–192. 32

[RGL05] REDON S., GALOPPO N., LIN M. C.: Adaptive dynamics of articulated bodies. *ACM Trans. Graph. 24* (July 2005), 936–945. 20, 35

[RKC00] REDON S., KHEDDAR A., COQUILLART S.: An algebraic solution to the problem of collision detection for rigid polyhedral objects. In *Proc. of IEEE Conference on Robotics and Automation* (2000). 29

[RKC02] REDON S., KHEDDAR A., COQUILLART S.: Fast continuous collision detection between rigid bodies. In *Proc. of Eurographics (Computer Graphics Forum)* (2002). 29

[RKC03] REDON S., KHEDDAR A., COQUILLART S.: Gauss least constraints principle and rigid body simulations. In *In proceedings of IEEE International Conference on Robotics and Automation* (2003). 31

[RKLM04] REDON S., KIM Y. J., LIN M. C., MANOCHA D.: Fast continuous collision detection for articulated models. In *Proceedings of ACM Symposium on Solid Modeling and Applications* (2004). 29

[RMEF09] ROBINSON-MOSHER A., ENGLISH R. E., FEDKIW R.: Accurate tangential velocities for solid fluid coupling. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 227–236. 35

[RMSG*08] ROBINSON-MOSHER A., SHINAR T., GRETARSSON J., SU J., FEDKIW R.: Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph. 27* (August 2008), 46:1–46:9. 35

[RO09] REITSMA P. S. A., O'SULLIVAN C.: Effect of scenario on perceptual sensitivity to errors in animation. *ACM Trans. Appl. Percept. 6* (September 2009), 15:1–15:16. 35

[RP03] REITSMA P. S. A., POLLARD N. S.: Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *ACM Trans. Graph. 22* (July 2003), 537–542. 35

[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. 19

[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. 19

[SG04] SCHENK O., GÄRTNER K.: Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems 20*, 3 (2004), 475–487. 27

[SGG*06] SUD A., GOVINDARAJU N., GAYLE R., KABUL I., MANOCHA D.: Fast proximity computation among deformable models using discrete voronoi diagrams. *ACM Trans. Graph. 25*, 3 (2006), 1144–1153. 30

[SL08] SERVIN M., LACOURSIÈRE C.: Rigid body cable for virtual environments. *IEEE Transactions on Visualization and Computer Graphics 14* (July 2008), 783–796. 4

[Smi00] SMITH R.: Open dynamics engine. `http://www.ode.org`, 2000. 2, 31

[SMT08] SIFAKIS E., MARINO S., TERAN J.: Globally coupled collision handling using volume preserving impulses. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), Eurographics Association, pp. 147–153. 29

[SNE09] SILCOWITZ M., NIEBE S., ERLEBEN K.: Nonsmooth newton method for fischer function reformulation of contact force problems for interactive rigid body simulation. In *Proceedings of Virtual Reality Interaction and Physical Simulation (VRIPHYS)* (November 2009). 31, 34

[SNE10a] SILCOWITZ M., NIEBE S., ERLEBEN K.: Contact point generation for convex polytopes in interactive rigid body dynamics. Poster at SCA 10', 2010. 34

[SNE10b] SILCOWITZ M., NIEBE S., ERLEBEN K.: A nonsmooth nonlinear conjugate gradient method for interactive contact force problems. *The Visual Computer* (2010). 23, 35

[SSF08] SHINAR T., SCHROEDER C., FEDKIW R.: Two-way coupling of rigid and deformable bodies. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), Eurographics Association, pp. 95–103. 35

[SSF09] SU J., SCHROEDER C., FEDKIW R.: Energy stability and fracture for frame rate rigid body simulations. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 155–164. 33, 35

[SSIF07] SIFAKIS E., SHINAR T., IRVING G., FEDKIW R.: Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 81–90. 35

[ST96] STEWART D. E., TRINKLE J. C.: An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering* (1996). 30, 31, 32

[Ste00] STEWART D. E.: Rigid-body dynamics with friction and impact. *SIAM Review* (2000). 32

[Stu08] STUDER C. W.: *Augmented time-stepping integration of non-smooth dynamical systems*. PhD thesis, ETH Zürich, 2008. Diss., Technische Wissenschaften, Eidgenössische Technische Hochschule ETH Zürich, Nr. 17597. 4, 30, 31

[TJ07] TWIGG C. D., JAMES D. L.: Many-worlds browsing for control of multibody dynamics. *ACM Trans. Graph. 26* (July 2007). 36

[TJ08] TWIGG C. D., JAMES D. L.: Backward steps in rigid body simulation. *ACM Trans. Graph. 27* (August 2008), 25:1–25:10. 36

[TNA08] TASORA A., NEGRUT D., ANITESCU M.: Large–scale Parallel Multi-body Dynamics with Frictional Contact on the Graphical Processing Unit. In *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics* (2008), Professional Engineering Publishing, pp. 315–326. 4, 27, 31

[TNA*10] TASORA A., NEGRUT D., ANITESCU M., MAZHAR H., HEYN T. D.: Simulation of Massive Multibody Systems using GPU Parallel Computation. In *18th International Conference on Computer Graphics, Visualization and Computer Vision, WSCG 2010* (2010), University of West Bohemia, Czech Republic, pp. 1–1. 4, 27, 31

[TP97] TRINKLE J., PANG J.: Dynamic multi-rigid-body systems with concurrent distributed contacts. In *Proceedings, IEEE International Conference on Robots and Automation* (April 1997), pp. 2276–2281. 11

[TPSL97] TRINKLE J., PANG J., SUDARSKY S., LO G.: On dynamic multi-rigid-body contact problems with coulomb friction. *Zeitschrift für Angewandte Mathematik und Mechanik 77*, 4 (1997), 267–279. 11, 13

[TTP01] TRINKLE J., TZITZOURIS J., PANG J.: Dynamic multi-rigid-body systems with concurrent distributed contacts: Theory and examples. *Philosophical Transactions: Mathematical, Physical, and Engineering Sciences 359*, 1789 (December 2001), 2575–2593. 8, 11

[Uni11] UNIVERSITY OF IOWA: The national advanced driving simulator. `http://www.nads-sc.uiowa.edu`, 2011. 4

[vdB05] VAN DEN BERGEN G.: Ray casting against general convex objects with application to continuous collision detection. Slides from Game Developer Conference, Accessed online 2011 `http://www.dtecta.com/`, 2005. 30

[WGF08] WEINSTEIN R., GUENDELMAN E., FEDKIW R.: Impulse-based control of joints and muscles. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 37–46. 33, 34, 35

[WGW90] WITKIN A., GLEICHER M., WELCH W.: Interactive dynamics. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics* (New York, NY, USA, 1990), ACM Press, pp. 11–21. 18

[Wit77] WITTENBURG J.: *Dynamics of systems of rigid bodies*, 1. ed. Teubner, 1977. 20

[WM09] WOULFE M., MANZKE M.: A framework for benchmarking interactive collision detection. In *Proceedings of the 2009 Spring Conference on Computer Graphics* (New York, NY, USA, 2009), SCCG '09, ACM, pp. 205–212. 33

[WSM*10] WELLER R., SAGARDIA M., MAINZER D., HULIN T., ZACHMANN G., PREUSCHE C.: A benchmarking suite for 6-dof real time collision response algorithms. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2010), VRST '10, ACM, pp. 63–70. 33

[WTF06] WEINSTEIN R., TERAN J., FEDKIW R.: Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Transactions on Visualization and Computer Graphics 12*, 3 (2006), 365–374. 18, 19, 33, 35

[WW90] WITKIN A., WELCH W.: Fast animation and control of nonrigid structures. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 243–252. 18

[ZKVM06] ZHANG L., KIM Y. J., VARADHAN G., MANOCHA D.: Generalized penetration depth computation. In *SPM '06: Proceedings of the 2006 ACM symposium on Solid and physical modeling* (New York, NY, USA, 2006), ACM, pp. 173–184. 30

[ZRLK07] ZHANG X., REDON S., LEE M., KIM Y. J.: Continuous collision detection for articulated models using taylor models and temporal culling. *ACM Trans. Graph. 26* (July 2007). 29, 30