DOI: 10.1111/cgf.12272

COMPUTER GRAPHICS *forum*
Volume 00 (2013), number 00 pp. 1–25

Interactive Simulation of Rigid Body Dynamics in Computer Graphics

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

[1]Graduate School CE, TU Darmstadt, Germany
bender@gsc.tu-darmstadt.de
[2]Department of Computer Science, University of Copenhagen, Denmark
kenny@diku.dk
[3]Department of Computer Science, Rensselaer Polytechnic Institute, USA
trinkle@gmail.com

Abstract

Interactive rigid body simulation is an important part of many modern computer tools, which no authoring tool nor game engine can do without. Such 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 have evolved and matured over the past 20 years. Furthermore, the paper communicates the mathematical and theoretical details in a pedagogical manner. This paper is not only a stake in the sand on what has been done, it also seeks to give the reader deeper insights to help guide their future research.

Keywords: rigid body dynamics, contact mechanics, articulated bodies, jointed mechanisms, contact point generation, iterative methods

ACM CCS: Computer Graphics [I.3.5]: Computational Geometry and Object Modelling—Physically-based modelling; 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 such as 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 (as shown in Figure 1), a subfield that has evolved rapidly over the past 10 years and moved the frontier of run-time simulation to applications in areas where, until recently, only off-line simulation was possible. As a consequence, this changes the computer tools humans use and has great economical impact on society as a whole.

The term 'interactive' implies a loop closed around a human and a simulation tool. For applications such as games where the feedback is simply animation on a screen, a reasonable goal is that the simulation delivers 60 frames per second (fps). For haptic rendering, the simulation would be part of a feedback loop running at 1000 Hz, where this rate is needed to display realistic forces to the user [LO08].

Rigid body dynamics has a long history in computer graphics [AG85], [MW88], [Hah88], [Bar89], [BBZ91] and a wealth of work exists on the topic. In this state-of-the-art paper, we will cover the important work over the past 20 years on interactive rigid body simulation since the last state-of-the-art report [Bar93] on the subject. In his STAR paper, Baraff discussed penalty- and constraint-based methods which use an acceleration-level non-linear complementarity problem (NCP) formulation. He did not cover many details on solving the complementarity problem. Not until 1994, when

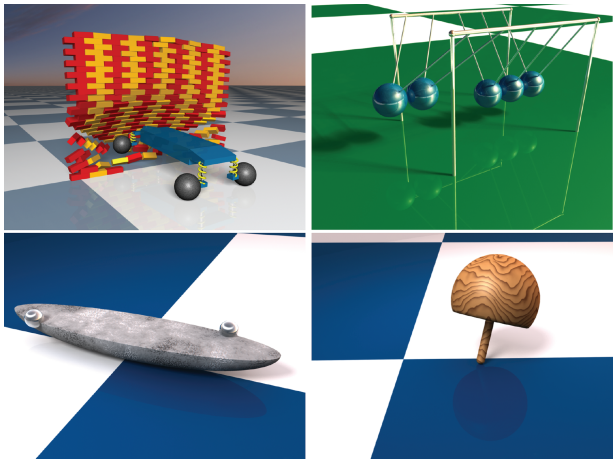


Figure 1: *Interactive rigid body simulations require the efficient simulation of joints, motors, collisions and contacts with friction.*

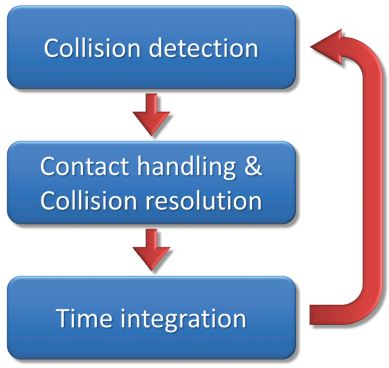


Figure 2: *The simulation loop provides a coarse description of data flow and processes in a rigid body simulator.*

Baraff published his version of a direct method based on pivoting, it was feasible to compute solutions for Baraff's complementarity problem formulation. For years this solution was the *de-facto* standard method of rigid body dynamics and it was used in both Maya and Open Dynamics Engine (ODE) [Smi00]. However, the solution only remained interactive for small-sized configurations (less than 100 interacting objects). When the number of interacting objects increased, the computational cost quickly made simulations take hours to compute. The acceleration-level formulation also caused problems concerning the existence of solutions and their uniqueness. Besides, solutions found by his algorithm did not always satisfy the static friction constraints. In the following years, the impulse-based paradigm was revisited by Mirtich [Mir96] and became 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 [Cou13], [Erl07] and ODE. As of this writing interactive simulations on single-core CPUs with several 1000 and up to 10 000 interacting objects are feasible. Multi-core and GPU works even go far beyond these limits. Today much active cross-disciplinary work is ongoing on different contact formulations and iterative solvers. Looking beyond contact problems, one 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.

The recent trend in interactive rigid body simulation has focused on delivering larger simulations of rigid bodies or creating faster simulation methods. The need for bigger and faster simulations is motivated by rigid body simulators being used (e.g. in digital production). The need in production for creating interesting motion requires more complex simulation scenarios. The well-known trade-off between accuracy and performance is an inherent property of interactive rigid body simulation. Many applications enforce a performance constraint, which leaves insufficient time for computing accurate solutions. Thus, it is a balance between accuracy and stability properties. Robustness is another desirable numerical trait when considering a human (or the real world in case of robotics) interacting with the simulator. 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.

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 manoeuvring 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], [KEP05], [KSJP08], [SKV*12]. 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. At the middle of the spectrum, one finds moderately fast simulators that can deliver high fidelity results. These are suitable for testing design ideas or training. In general, different application areas have different requirements in regards to performance/quality trade-offs and accuracy. We refer the interested reader to [BETC12] for a detailed discussion on this. Bender *et al*. [BETC12] contain a detailed taxonomy of models. We have omitted the taxonomy in this paper and chosen to focus more on the 'key' model and corresponding numerical methods.

1.1. The anatomy and physiology of a rigid body simulator

A rigid body simulator is a complex and large piece of software, which traditionally implements a simulation loop similar to the one 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 inter-penetrating. This phase is termed as *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

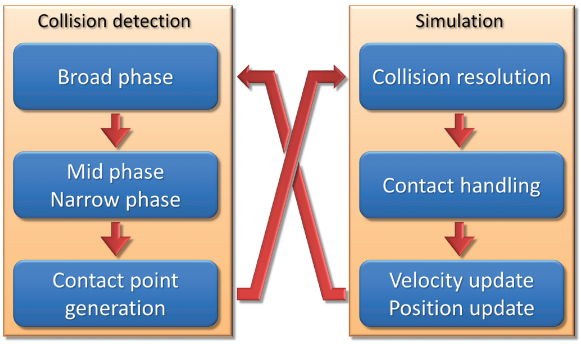


Figure 3: *A modular phase description of the sub-tasks of a rigid body simulator.*

body velocities, so are often handled separately from pre-existing contacts. This is termed 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 (see Figure 3). In the first phase, called the *broad phase*, bodies are approximated by simple geometric primitives, for which distance computations are very fast. For example, each body is replaced by the smallest sphere that completely contains it. If the spheres of 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 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 remaining bodies are used to find the precise body features in contact and the location of the contact points. The narrow phase is often combined with the mid-phase for performance reasons. Note that some narrow phase algorithms do not return all the contact information needed to formulate the dynamic model, in which case a separate contact point generation algorithm must be applied. This paper will not treat the subject of collision detection instead we refer to [BETC12] for further discussion. For more details in general about collision detection, we recommend [LG98], [Eri04].

1.2. Outline

We first give a short introduction to rigid body dynamics in Section 2. For a better understanding, we separated the explanation of fundamental models for systems with joints and contacts in Section 3 from the discussion of numerical methods to compute solutions in Section 4. The parallelization and optimization of these methods is covered in Section 5. Finally, we conclude in Section 6 pointing out several avenues for future work.

2. A Quick Primer

Rigid body simulation requires the numerical solution of non-linear ordinary differential equations constrained by algebraic inequalities for which closed-form solutions do not exist. Assume that time t is the independent variable. Given a time period $[t_0,\ t_N]$, 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, in the form of a system of (state-dependent) algebraic equations and inequalities. The discrete-time model is formulated and solved at each time step $(t_0, .., t_N)$.

In rigid body simulation, one begins with the Newton–Euler (differential) equations, which describe the dynamic motion of the bodies without contact. These differential equations are then augmented with three types of conditions: non-penetration 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 NCP (dNCP) that cannot be solved in closed form. Instead, it is discretized in time, 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 to trajectories of the original instantaneous-time model. Computing a discrete-time solution requires a consideration of possible reformulations of the NCPs and the choice of a solution method. Two good options are a reformulation as non-smooth equation using the Fischer–Burmeister function (Section 4.3) or proximal point mappings [Stu08].

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 first law: The velocity of a body remains unchanged unless acted upon by a force.
- Newton's second law: The time rate of change of momentum of a body is equal to the applied force.
- Newton's third 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

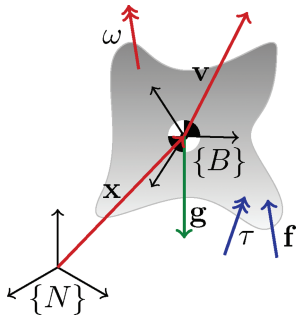


Figure 4: *Illustration of a spatial rigid body showing the body frame $\{B\}$ and inertial frame $\{N\}$ as well as notation for positions, velocities and forces.*

to differential equations of motion. While the second law actually applies only to particles, Euler extended 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.

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 infinitely large 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 centre of mass. To compute rotational motions, the mass distribution is important. This is captured in a 3×3 matrix known as the mass (or inertia) matrix $\mathbf{I} \in \mathbb{R}^{(3\times3)}$. It is a 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 centre 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 centre of mass, a rigid body has three translational degrees of freedom (DOFs).

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. Thus, a rigid body has three rotational DOFs. 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 DOFs. Considering translation and rotation together, a rigid body has six DOFs.

The *rotational velocity* $\omega \in \mathbb{R}^3$ (also known as, *angular velocity*) of a body can be thought of as vector where the 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 DOFs.

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 centre 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 centre of mass and the orientation parameters. Note that the number of elements 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 parameter Q 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:

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

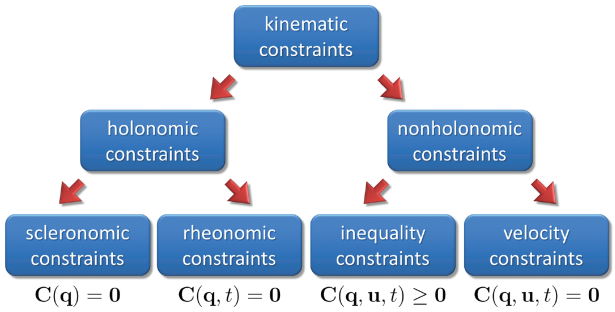


Figure 5: *Constraint classification.*

where $\mathbf{H} = \begin{bmatrix} \mathbf{1}_{3\times3} & \mathbf{0} \\ \mathbf{0} & \mathbf{G} \end{bmatrix}$, where $\mathbf{1}_{3\times3}$ is the 3×3 identity matrix. Note that when the orientation representation uses more than three parameters, $\mathbf{G}$ is not square, although it has the property that $\mathbf{G}^T\mathbf{G} = \mathbf{1}_{3\times3}$.

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. The constraints alone 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:

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

or

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

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 centres located at $\mathbf{x}_1$ and $\mathbf{x}_2$. Consider the constraint function:

$$C(x_1, x_2) = ||x_1 - x_2|| - (r_1 + r_2), \tag{4}$$

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, if C is non-negative, 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 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. Rhenomic 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 *non-holonomic*. 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 non-holonomic equality constraint. If the car is driving, 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 non-holonomic.

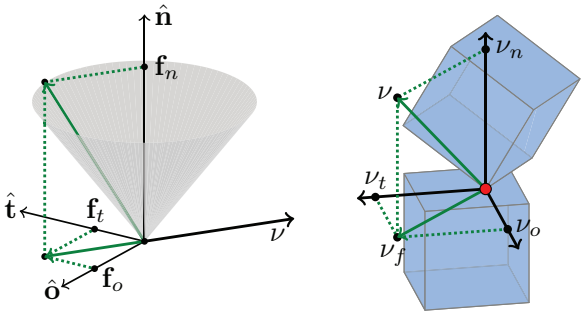
Holonomic constraints remove DOFs from the system, i.e. the dimension of the space of possible generalized positions is reduced. For instance, two free rigid bodies have a total of 12 DOFs, but as in the previous case of the touching spheres, one DOF is lost. Assume that one sphere can be moved at will through space using all six DOFs. 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 DOFs 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 DOFs. Thus, a system of two spheres with one contact constraint has 11 DOFs. If instead, two bodies were connected by a hinge joint, the system would have seven DOFs. That is, if you allow one body to move with six DOFs, 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, DOFs from the system. In the car example, the car cannot translate instantaneously left or right. However, every competent driver can accomplish a lateral move of his car by executing the kind of manoeuvre 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 by-products 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.



(a) A Friction Cone (b) Contact velocities

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

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 centre of mass of the body. This is because the effect of gravity acting on an entire body is equivalent to a single force of magnitude mg acting through its centre 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 6a). 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 component $\mathbf{f}_n$ 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 v is decomposed into components, v_n, v_t and v_o (see Figure 6b). The contact is sliding if $v_n = 0$ and v_t or v_o is non-zero, and separating if v_n is greater than zero. Negative v_n is not allowed, as it corresponds to interpenetration of the bodies.

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

$$\mathcal{F}(\mathbf{f}_n, \mu) = \{\mu^2 \mathbf{f}_n^2 - \mathbf{f}_t^2 - \mathbf{f}_o^2 \geq 0, \mathbf{f}_n \geq 0\}, \tag{5}$$

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

$$\mathbf{f}_t = -\mu \mathbf{f}_n \frac{v_t}{\beta}, \quad \mathbf{f}_o = -\mu \mathbf{f}_n \frac{v_o}{\beta}, \tag{6}$$

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

Common variations on this model include using two different friction coefficients; one for sticking contact and a lower one for sliding. When friction forces are larger in one direction than another, one can replace the circular cone with an elliptical cone. In some simulation schemes, the non-linearity of the friction cone causes problems, and so it is eliminated by approximating the cone as a 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 [Con62], [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 and mass distribution) yields:

$$m\dot{\mathbf{v}} = \mathbf{f}, \tag{7}$$

$$\mathbf{I}\dot{\omega} + \omega \times \mathbf{I}\omega = \tau, \tag{8}$$

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

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. This is usually a body-fixed frame (which is rotating with the body in the inertial frame) or the inertial frame. In a body-fixed frame, $\mathbf{I}_{\text{body}}$ 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 centre 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 rotation axis of a rotating rigid body to 'precess' about a circular cone.

The Newton–Euler equations can also be written as

$$\mathbf{M}\dot{\mathbf{u}} = \mathbf{g}, \tag{9}$$

where $\mathbf{M}$ is the generalized mass matrix containing the body mass properties and $\mathbf{g}$ is the vector of loads, including the gyroscopic moment. The generalized mass matrix $\mathbf{M}$ is a block diagonal matrix defined as

$$\mathbf{M} = \begin{bmatrix} m\mathbf{1}_{3\times3} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{6\times6}, \tag{10}$$

where $\mathbf{1}_{3\times3}$ is the 3×3 identity matrix. Since each block is positive definite and symmetric, so too is $\mathbf{M}$. The load vector of a rigid body is defined by

$$\mathbf{g} = \begin{bmatrix} \mathbf{f} \\ \tau - \omega \times \mathbf{I}\omega \end{bmatrix} \in \mathbb{R}^6. \tag{11}$$

Simulation of free body motion is done by integrating the Newton–Euler (Equation 9) 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 (5) and (6) must be included. The complete system of differential and algebraic equations and inequalities is challenging to integrate and the development of robust methods has been a research topic for more than 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 two bodies collide, 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 $\mathbf{p} = \int_t^{t+\Delta t} \mathbf{f}\mathrm{d}t$ as the impulse of the net force and $m\mathbf{v}$ as translational momentum. Integrating Equation (7) from t to $t+\Delta t$ yields $m(\mathbf{v}(t+\Delta t) - \mathbf{v}(t)) = \int_t^{t+\Delta t} \mathbf{f}\mathrm{d}t$, 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 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 the 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. $v_n^+ > 0$).

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 Systems with Frictional Joints and Contacts

This section introduces the most important models for systems with joints and contacts. In this paper, we focus on simultaneous models using a constraint-based approach. Numerical methods for the computation of solutions are presented in Section 4.

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

3.1. Model components

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 behaviour) 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 (see Section 2.1.3): permanent mechanical joints, each represented by a system of equations (five scalar equations in the case of a one-DOF joint), and isolated point contacts with well-defined contact normals, each represented by one scalar inequality. Let $\mathcal{B}$ and $\mathcal{U}$ denote the mutually exclusive sets of bilateral (equality) and unilateral (inequality) contacts:

$$\mathcal{B} = \{i \ : \ \text{contact } i \text{ is a joint}\}, \tag{12}$$

$$\mathcal{U} = \{i \ : \ \text{contact } i \text{ is a point contact}\}, \tag{13}$$

where $\mathcal{B} \cup \mathcal{U} = \{1, \ldots, n_c\}$ 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 sticking. 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 are broken as bodies separate (${}^{u}\mathbf{C}_{in} > 0$). The constraint function (4) for two rigid spheres is a simple example of a gap function that can be written in closed form. Simple gap functions also arise in one-DOF joints with limits; if q_i is the displacement of joint i, and $q_{i,\min}$ and $q_{i,\max}$ are the minimum and maximum displacements, then joint i has two gap functions: $q_i - q_{i,\min} \geq 0$ and $q_{i,\max} - q_i \geq 0$. Generally, gap functions are not available in closed form, but fortunately, they are not needed by most simulation algorithms. In explicit time-stepping methods, such as considered in this paper, one simply needs gap values at the start of the current time step, and these can be obtained from collision detection algorithms. In the case of geometrically implicit algorithms, such as the one developed by Chakraborty *et al.* [CBAT13], one needs closed-form expressions of the body geometries, but not closed-form gap functions.

The first time derivatives of the distance functions are the relative contact velocities, ${}^{\kappa}v_{i\sigma} = \frac{\mathrm{d}}{\mathrm{d}t}\left({}^{\kappa}\mathbf{C}_{i\sigma}\right);\ \kappa \in \{b,\ u\}, \sigma \in \{n,\ f\}$. Note that ${}^{\kappa}v_{in}$ and ${}^{\kappa}v_{if}$ 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 sticking contact, instantaneously, the relative velocity of the contact points projected into the frictional subspace is zero (${}^{\kappa}v_{if} = 0$). If they slip, at least one friction direction displacement will become non-zero. For example, the friction direction of a one-DOF 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, o-plane. 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 sticking 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 sticking with this position-level condition; one needs time derivatives. The velocity-level set definitions are

$$\mathcal{S} = \{i \ : \ {}^{\kappa}\mathbf{C}_{in} = 0,\ {}^{\kappa}v_{in} = 0,\ {}^{\kappa}v_{if} \neq 0\}, \tag{14}$$

$$\mathcal{R} = \{i \ : \ {}^{\kappa}\mathbf{C}_{in} = 0,\ {}^{\kappa}v_{in} = 0,\ {}^{\kappa}v_{if} = 0\}, \tag{15}$$

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.

Newton–Euler equations: The Newton–Euler equation for a system of rigid bodies is defined by

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} = \mathbf{g}(\mathbf{q}, \mathbf{u}, t), \tag{16}$$

where $\mathbf{M}(\mathbf{q})$ is the generalized mass matrix and $\mathbf{g}(\mathbf{q}, \mathbf{u}, t)$ is the load vector (cf. Equation 9). $\mathbf{M}$ is a positive definite and symmetric block diagonal matrix with the jth block defined by the mass matrix of body j (see Equation 10). Hence, its dimension is $(6n_b \times 6n_b)$, where n_b is the number of rigid bodies in the system. The load vector $\mathbf{g} \in \mathbb{R}^{6n_b}$ is formed by stacking the load vectors of the individual bodies (see Equation 11).

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 $\mathbf{u}$ (cf. Equation 1):

$$\dot{\mathbf{q}} = \mathbf{H}(\mathbf{q})\mathbf{u}, \tag{17}$$

where $\mathbf{H}(\mathbf{q})$ is the generalized kinematic map, which is block diagonal, with non-zero blocks $\mathbf{H}_{jj}$ given by Equation (1). If unit quaternions are used to represent body configurations, then the size of $\mathbf{H}$ is $(7n_b \times 6n_b)$.

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:

$${}^{b}\mathbf{C}_{n}(\mathbf{q}, t) = \mathbf{0}, \tag{18}$$

which is a holonomic constraint (see Section 2.1.3). From a physical perspective, these constraints are maintained by reaction forces ${}^{b}\mathbf{f}_{in}$ that are un-constrained. That is, generalized forces normal or anti-normal to the constraint surface in the system's configuration space can be generated. When viewing multi-body 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:

$${}^{u}\mathbf{C}_{n}(\mathbf{q}, t) \geq \mathbf{0}, \tag{19}$$

which is a non-holonomic inequality constraint (see Section 2.1.3). 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 ${}^{u}\mathbf{f}_{n}$, all normal force constraints can be written as ${}^{u}\mathbf{f}_{n} \geq \mathbf{0}$.

There is one more aspect of unilateral contacts that must be modelled. 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 sticking, 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 $\mathcal{F}_i({}^{\kappa}\mathbf{f}_{in}, \mu_i)$ (see Equation 5) 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:

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

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, ${}^{\kappa}v_{if} = [{}^{u}v_{it}\ {}^{u}v_{io}]^T$.

Finally, our instantaneous-time dynamic model is the system of differential algebraic inequalities (DAIs) composed of Equations (9) and (17)–(21), where the sliding and sticking contact sets are defined by Equations (14) and (15).

In its current form, the DAI is difficult to solve, partly because its has more unknowns (all of the positions, velocities, acceleration and forces) than equations. However, as will be shown, it is possible to cast the model as a differential complementarity problem [CPS92], [TPSL97] (a square system) in terms of only accelerations and forces, and then discretize the result to form time-stepping subproblems in the form of NCP or linear complementarity problem (LCP), which allows one to apply well-studied solution algorithms.

3.2. Complementarity problems

Several standard complementarity problems that will be used later in this section are the non-linear, linear, mixed non-linear and mixed LCPs.

Definition 1. *(NCP):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*

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

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

If the function $\mathbf{y}(\mathbf{x})$ is linear, then we have

Definition 2. *(LCP): 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*

$$ \mathbf{0} \leq \mathbf{A}\mathbf{x} + \mathbf{b} \perp \mathbf{x} \geq \mathbf{0}. \tag{23} $$

For LCPs, we adopt the shorthand notation, LCP($\mathbf{A}$, $\mathbf{b}$*).*

Mixed complementarity problems include equalities and unrestricted variables:

Definition 3. *Mixed Nonlinear Complementarity Problem (mNCP): Given unknown vectors* $\mathbf{x} \in \mathbb{R}^m$ *and* $\mathbf{w} \in \mathbb{R}^n$*, and known vector functions* $\mathbf{y}(\mathbf{x},\ \mathbf{w}) : \mathbb{R}^{m+n} \to \mathbb{R}^m$ *and* $\mathbf{z}(\mathbf{x},\ \mathbf{w}) : \mathbb{R}^{m+n} \to \mathbb{R}^n$*, find* $\mathbf{x}$ *and* $\mathbf{w}$ *such that*

$$ \mathbf{z}(\mathbf{x},\ \mathbf{w}) = \mathbf{0}. \tag{24} $$

$$ \mathbf{0} \leq \mathbf{y}(\mathbf{x},\ \mathbf{w}) \perp \mathbf{x} \geq \mathbf{0}. \tag{25} $$

Definition 4. *Mixed Linear Complementarity Problem (mLCP): Given unknown vectors* $\mathbf{x} \in \mathbb{R}^m$ *and* $\mathbf{w} \in \mathbb{R}^n$*, known fixed square matrices* $\mathbf{F} \in \mathbb{R}^{m \times m}$ *and* $\mathbf{D} \in \mathbb{R}^{n \times n}$*, known fixed rectangular matrices* $\mathbf{B} \in \mathbb{R}^{m \times n}$ *and* $\mathbf{C} \in \mathbb{R}^{n \times m}$*, and known fixed vectors* $\mathbf{a} \in \mathbb{R}^m$ *and* $\mathbf{r} \in \mathbb{R}^n$*, determine* $\mathbf{x}$ *and* $\mathbf{w}$ *such tha:*

$$ \mathbf{0} \leq \mathbf{F}\mathbf{x} + \mathbf{B}\mathbf{w} + \mathbf{a} \perp \mathbf{x} \geq \mathbf{0}, \tag{26} $$

$$ \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{w} + \mathbf{r} = \mathbf{0}. \tag{27} $$

3.3. Complementarity formulation of the instantaneous-time model

To achieve model formulation as a properly posed complementarity problem, we must rewrite all the conditions (9) and (17)–(21) in terms of a common set of dependent variables. By taking the appropriate number of time derivatives, all equations will be written in terms of accelerations, thus generating a model in which all unknowns are forces and accelerations. In addition, this model has the same number of equations and unknowns. This transformation will be carried out below in three steps. First, differentiate the distance functions twice with respect to time to expose the acceleration variables, second, express the principle of maximum dissipation as a system of equations and inequalities in forces and accelerations and third, reformulate the Newton–Euler equation to expose the forces.

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 truncated after the quadratic terms is

$$\widehat{{}^{\kappa}C_{i\sigma}} \qquad (\tilde{\mathbf{q}}, \tilde{t}) = {}^{\kappa}C_{i\sigma}(\mathbf{q}, t) + \tfrac{\partial^{\kappa} C_{i\sigma}}{\partial \mathbf{q}} \Delta\mathbf{q} + \tfrac{\partial^{\kappa} C_{i\sigma}}{\partial t} \Delta t + \tfrac{1}{2}\left((\Delta\mathbf{q})^T \tfrac{\partial^2 {}^{\kappa}C_{i\sigma}}{\partial \mathbf{q}^2} \Delta\mathbf{q} + 2 \tfrac{\partial^2 {}^{\kappa}C_{i\sigma}}{\partial \mathbf{q} \partial t} \Delta\mathbf{q}\Delta t + \tfrac{\partial^2 {}^{\kappa}C_{i\sigma}}{\partial t^2} \Delta t^2 \right),$$

where $\widehat{{}^{\kappa}C_{i\sigma}}$ is an approximation of ${}^{\kappa}C_{i\sigma}$. 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, ${}^{\kappa}v_{i\sigma}$, at the contact. Dividing the quadratic terms by $(\Delta t)^2$ and taking the limit yields the relative acceleration ${}^{\kappa}\mathbf{a}_{i\sigma}$:

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

where

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

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

Stacking all the quantities above for every unilateral and bilateral contact (as defined in Equations 14 and 15), one arrives at the definitions of ${}^{\kappa}\mathbf{a}_n$, ${}^{\kappa}\mathbf{J}_n$ and ${}^{\kappa}\mathbf{k}_n$. Under the assumption that normal distance functions and the normal components of the relative velocity are zero, Equations (18)–(20) can be expressed at the acceleration level as follows:

$${}^{b}\mathbf{a}_n = \mathbf{0}. \tag{29}$$

$$\mathbf{0} \le {}^{u}\mathbf{f}_n \perp {}^{u}\mathbf{a}_n \ge \mathbf{0}. \tag{30}$$

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-DOF 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 yield a valid solution, the system must satisfy a regularity condition (also known as, 'constraint qualification') on the boundary of the feasible set. 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

$$-{}^{u}\beta_{i0}({}^{u}\mathbf{f}_{it}\,{}^{u}v_{it} + {}^{u}\mathbf{f}_{io}\,{}^{u}v_{io}) + {}^{u}\beta_i(\mu_i^2\,{}^{u}\mathbf{f}_{in}^2 - {}^{u}\mathbf{f}_{it}^2 - {}^{u}\mathbf{f}_{io}^2), \tag{31}$$

where ${}^{u}\beta_{i0}$ and ${}^{u}\beta_i$ 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 constraint qualification conditions of the Fritz–John method: ${}^{u}\beta_{i0} \ge 0$ and $({}^{u}\beta_{i0}, {}^{u}\beta_i) \ne (0,0)$. Following the derivation on pages 28–30 of [Ber09], one arrives at the following system of constraints:

$$\left.\begin{aligned} \mu_i\,{}^{u}\mathbf{f}_{in}\,{}^{u}v_{it} + {}^{u}\mathbf{f}_{it}\,{}^{u}\beta_i &= 0 \\ \mu_i\,{}^{u}\mathbf{f}_{in}\,{}^{u}v_{io} + {}^{u}\mathbf{f}_{io}\,{}^{u}\beta_i &= 0 \\ {}^{u}\gamma_i = \mu_i^2\,{}^{u}\mathbf{f}_{in}^2 - {}^{u}\mathbf{f}_{it}^2 - {}^{u}\mathbf{f}_{io}^2 &\ge 0 \\ 0 \le {}^{u}\gamma_i \perp {}^{u}\beta_i &\ge 0 \end{aligned}\right\} \quad \forall\, i \in \{\mathcal{U} \cap \mathcal{S}\}, \tag{32}$$

where ${}^{u}\gamma_i$ is a slack variable for the friction limit set. Note that ${}^{u}\beta_i = \| {}^{u}v_{if} \|$ at the optimal solution, and represents the magnitude of the slip velocity at contact i (i.e. ${}^{u}\beta_i = \| {}^{u}v_{if} \| = \sqrt{{}^{u}v_{it}^2 + {}^{u}v_{io}^2}$). Also, 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, Equation (6) is used.

If contact i is a one-DOF 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

$$\mathcal{F}_i({}^{b}\mathbf{f}_{if_{\max}}) = \left\{ {}^{b}\mathbf{f}_{if} \ :\ \left|{}^{b}\mathbf{f}_{if}\right| \le {}^{b}\mathbf{f}_{if_{\max}} \right\}, \quad \forall\, i \in \{\mathcal{B} \cap \mathcal{S}\}, \tag{33}$$

where $|\cdot|$ denotes the absolute value of a scalar and ${}^{b}\mathbf{f}_{if_{\max}}$ is the non-negative 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

$$\left.\begin{aligned} {}^{b}\mathbf{f}_{io_{\max}}\,{}^{b}v_{io} + {}^{b}\mathbf{f}_{io}\,{}^{b}\beta_i &= 0 \\ {}^{b}\gamma_i = {}^{b}\mathbf{f}_{io_{\max}}^2 - {}^{b}\mathbf{f}_{io}^2 &\ge 0 \\ 0 \le {}^{b}\gamma_i \perp {}^{b}\beta_i &\ge 0 \end{aligned}\right\} \quad \forall\, i \in \{\mathcal{B} \cap \mathcal{S}\}. \tag{34}$$

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

The principle of maximum dissipation (21) must be considered further. When contact i is sliding, the solutions of conditions (32) and (34) 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 sticking, 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 sticking contact will change to sliding. However, this problem is easily remedied by considering Equations (32) and (34), the relative velocity is zero. In this case, the relative velocity at the onset of slipping is proportional to the relative acceleration. Therefore, they can be replaced with the analogous acceleration variables yielding:

$$\left.\begin{aligned} \mu_i\, {}^u\mathbf{f}_{in}\, {}^u\mathbf{a}_{it} + {}^u\mathbf{f}_{it}\, {}^u\beta_i &= 0 \\ \mu_i\, {}^u\mathbf{f}_{in}\, {}^u\mathbf{a}_{io} + {}^u\mathbf{f}_{io}\, {}^u\beta_i &= 0 \\ {}^u\gamma_i = \mu_i^2\, {}^u\mathbf{f}_{in}^2 - {}^u\mathbf{f}_{it}^2 - {}^u\mathbf{f}_{io}^2 &\geq 0 \\ 0 \leq {}^u\gamma_i \perp {}^u\beta_i &\geq 0 \end{aligned}\right\} \quad \forall\, i \in \{\mathcal{U} \cap \mathcal{R}\}, \tag{35}$$

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

$$\left.\begin{aligned} {}^b\mathbf{f}_{if_{\max}}\, {}^b\mathbf{a}_{if} + {}^b\mathbf{f}_{if}\, {}^b\beta_i &= 0 \\ {}^b\gamma_i = {}^b\mathbf{f}_{if_{\max}}^2 - {}^b\mathbf{f}_{if}^2 &\geq 0 \\ 0 \leq {}^b\gamma_i \perp {}^b\beta_i &\geq 0 \end{aligned}\right\} \quad \forall\, i \in \{\mathcal{B} \cap \mathcal{R}\}, \tag{36}$$

where ${}^b\beta_{if} = \left|{}^b\mathbf{a}_{if}\right|$ 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:

$$\begin{aligned} \mathbf{M}(\boldsymbol{q})\dot{\mathbf{u}} = {}^u\mathbf{J}_n(\mathbf{q})^T\, {}^u\mathbf{f}_n + {}^u\mathbf{J}_f(\mathbf{q})^T\, {}^u\mathbf{f}_f \\ + {}^b\mathbf{J}_n(\mathbf{q})^T\, {}^b\mathbf{f}_n + {}^b\mathbf{J}_f(\mathbf{q})^T\, {}^b\mathbf{f}_f + \mathbf{g}_{\text{ext}}(\mathbf{q}, \mathbf{u}, t), \end{aligned} \tag{37}$$

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.

Definition 5. (Model-dNCP)**.** *Collecting equations (17), (29), (30), (32) and (34)–(37) a dynamic model in the form of a differential NCP (dNCP) is defined.*

If one wanted to use this model in an integration scheme, the PATH algorithm by Ferris and Munson is the most robust, general-purpose NCP solver available [CPN11]. However, we recommend against this, because a solution does not always exist [PT96], and when one does, it could have infinite values. Fortunately, the existence problem can be eliminated by discretizing Model-dNCP over time.

3.4. A non-linear discrete-time model as an mNCP

Now, we discretize the five components of the instantaneous-time model derived above. Let t_ℓ be 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. Given a positive time step $\Delta t = t_{\ell+1} - t_\ell$, 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 dNCP. 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 (37) yields the equation below in which all quantities are evaluated at the end of the time step:

$$\begin{aligned} \mathbf{M}^{(\ell+1)}\left(\mathbf{u}^{(\ell+1)} - \mathbf{u}^{(\ell)}\right) = ({}^u\mathbf{J}_n^T)^{(\ell+1)}\, {}^u\mathbf{p}_n^{(\ell+1)} \\ + ({}^u\mathbf{J}_f^T)^{(\ell+1)}\, {}^u\mathbf{p}_f^{(\ell+1)} + ({}^b\mathbf{J}_n^T)^{(\ell+1)}\, {}^b\mathbf{p}_n^{(\ell+1)} \\ + ({}^b\mathbf{J}_f^T)^{(\ell+1)}\, {}^b\mathbf{p}_f^{(\ell+1)} + (\mathbf{p}_{\text{ext}})^{(\ell+1)}, \end{aligned} \tag{38}$$

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. Note that, in general, the inertia matrix, Jacobians and external impulse all depend non-linearly on the state, $(\mathbf{q}, \mathbf{u})$.

Discrete-time kinematic map: Applying the backward Euler approximation to the kinematic map (17) gives

$$\mathbf{q}^{(\ell+1)} - \mathbf{q}^{(\ell)} = \Delta t \mathbf{H}^{(\ell+1)} \mathbf{u}^{(\ell+1)}, \tag{39}$$

which is non-linear in the unknown system configuration $\mathbf{q}^{(\ell+1)}$ due to the dependence of $\mathbf{H}$ on $\mathbf{q}$. 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 joint and normal contact constraints: At the end of each time step, the constraints should be satisfied. Therefore, the discrete-time joint constraints, Equation (18), are simply the joint constraints enforced at time $t_{\ell+1}$:

$${}^b\mathbf{C}_n^{(\ell+1)} = \mathbf{0}, \tag{40}$$

where ${}^b\mathbf{C}_n^{(\ell+1)}$ denotes ${}^b\mathbf{C}_n(\mathbf{q}^{\ell+1},\ t_{\ell+1})$.

The analogous approach applies to the normal contact constraints, but with one additional consideration; the contact normal forces must be replaced with normal impulses. Thus, Equations (19)–(20) become:

$$0 \leq {}^u\mathbf{p}_n^{(\ell+1)} \perp {}^u\mathbf{C}_n^{(\ell+1)} \geq 0. \tag{41}$$

Note that, in general, the right-hand inequality is non-linear in the unknown $\mathbf{u}^{(\ell+1)}$. However, it can be made linear by a Taylor series expansion. It is also important to note that this relationship implies that the normal impulse ${}^u\mathbf{p}_n^{(\ell+1)}$ at the end of the time step can be non-zero only if the distance function at the end of the time step is also zero. It says nothing about what can happen during the time step.

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:

$$ {}^{\kappa}\mathbf{p}_{if}^{(\ell+1)} \in \arg\max_{\mathbf{p}'_{if}} \left\{ -\left({}^{\kappa}\nu_{if}^{(\ell+1)}\right)^{T} \mathbf{p}'_{if} : \mathbf{p}'_{if} \in \mathcal{F}_i({}^{\kappa}\mathbf{p}_{in}^{(\ell+1)}, \mu_i) \right\}, $$

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 $\mathcal{F}_i$.

The discrete-time forms of the friction models presented earlier are formed simply by replacing forces with impulses and enforcing the conditions at the end of the time step. The contact friction model becomes:

$$ \begin{aligned} \mu_i \, {}^{u}\mathbf{p}_{in}^{\ell+1} {}^{u}\nu_{it}^{\ell+1} + {}^{u}\mathbf{p}_{it}^{\ell+1} {}^{u}\beta_i^{\ell+1} &= 0, \\ \mu_i \, {}^{u}\mathbf{p}_{in}^{\ell+1} {}^{u}\nu_{io}^{\ell+1} + {}^{u}\mathbf{p}_{io}^{\ell+1} {}^{u}\beta_i^{\ell+1} &= 0, \\ {}^{u}\gamma_i^{\ell+1} = \mu_i^2 \, ({}^{u}\mathbf{p}_{in}^{\ell+1})^2 - ({}^{u}\mathbf{p}_{it}^{\ell+1})^2 - ({}^{u}\mathbf{p}_{io}^{\ell+1})^2 &\geq 0, \\ 0 \leq {}^{u}\gamma_i^{\ell+1} \perp {}^{u}\beta_i^{\ell+1} &\geq 0, \end{aligned} \tag{42} $$

where $\forall\, i \in \mathcal{A}(\mathbf{q}, \mathbf{u})$ and $\mathcal{A}$ is the set of active constraints, typically defined by $\mathcal{A} = \{i : {}^{u}\mathbf{C}_{in} \leq \epsilon(\mathbf{u})\}$, where ϵ is an estimate of the size of the gap between bodies that could close during the next time step. Notice that because the velocities in the discrete-time model are unknown, there is no way to distinguish sliding and sticking contacts during problem formulation. The contact interactions are determined as a by-product of solving the time-stepping subproblem. The discrete-time form of dry friction in the joints is given similarly:

$$ \left.\begin{aligned} {}^{b}\mathbf{p}_{io_{\max}} {}^{b}\nu_{io}^{\ell+1} + {}^{b}\mathbf{p}_{io}^{\ell+1} {}^{b}\beta_i^{\ell+1} &= 0 \\ {}^{b}\gamma_i^{\ell+1} = {}^{b}\mathbf{p}_{io_{\max}}^{2} - ({}^{b}\mathbf{p}_{io}^{2})^{\ell+1} &\geq 0 \\ 0 \leq {}^{b}\gamma_i^{\ell+1} \perp {}^{b}\beta_i^{\ell+1} &\geq 0 \end{aligned}\right\} \quad \forall\, i \in \mathcal{B}. \tag{43} $$

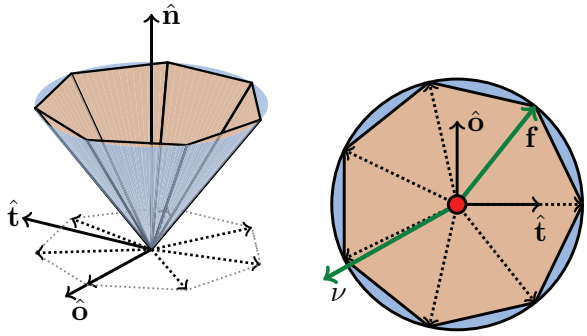
Again, one need not distinguish between sliding and sticking contacts at the time of formulation.

Definition 6. (Model-mNCP)**.** *Equations (38)–(41), (42) and (43) define the model as a sequence of time-stepping subproblems.*

3.5. The discrete-time model as an mLCP

Theoretical support and solution algorithms are better developed for LCPs than for NCPs [CPS92]. Therefore, it is sometimes preferable to use a linearized version of Model-mNCP. The discrete-time Newton–Euler equation (38) is linear in the unknown impulses and velocities, but the inertia and 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 assume that all of these quantities are equal to their values at t_ℓ and constant over the next step.

$$ \begin{aligned} \mathbf{M}^{(\ell)} \left(\mathbf{u}^{(\ell+1)} - \mathbf{u}^{(\ell)}\right) &= ({}^{u}\mathbf{J}_n^T)^{(\ell)} {}^{u}\mathbf{p}_n^{(\ell+1)} \\ &+ ({}^{u}\mathbf{J}_f^T)^{(\ell)} {}^{u}\mathbf{p}_f^{(\ell+1)} + ({}^{b}\mathbf{J}_n^T)^{(\ell)} {}^{b}\mathbf{p}_n^{(\ell+1)} \\ &+ ({}^{b}\mathbf{J}_f^T)^{(\ell)} {}^{b}\mathbf{p}_f^{(\ell+1)} + (\mathbf{p}_{\text{ext}})^{(\ell)}. \end{aligned} \tag{44} $$



(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.*

Since the values of these quantities vary over the time step, fixing their values at t_ℓ introduces error, which reduces with the size of the time step and the velocities of the bodies.

Equation (39) is also non-linear due to the dependence of $\mathbf{H}$ on $\mathbf{q}^{(\ell+1)}$. As above, evaluating it at t_ℓ yields a linear approximation:

$$ \mathbf{q}^{(\ell+1)} - \mathbf{q}^{(\ell)} = \Delta t \mathbf{H}^{(\ell)} \mathbf{u}^{(\ell+1)}. \tag{45} $$

Linearized joint and contact constraints: The contact and joint displacement functions given in Equations (18) and (19) can be linearized by Taylor series expansion about t_ℓ:

$$ \widehat{{}^{\kappa}\mathbf{C}_{\sigma}^{(\ell+1)}} = {}^{\kappa}\mathbf{C}_{\sigma}^{(\ell)} + ({}^{\kappa}\mathbf{J}_{\sigma})^{(\ell)} \mathbf{u}^{(\ell+1)} \Delta t + \frac{\partial {}^{\kappa}\mathbf{C}_{\sigma}^{(\ell)}}{\partial t} \Delta t, \tag{46} $$

where the hat over the displacement function connotes linear approximation. For joints, the approximation of Equation (46) becomes:

$$ {}^{b}\mathbf{C}_{n}^{(\ell)} + ({}^{b}\mathbf{J}_n)^{(\ell)} \mathbf{u}^{(\ell+1)} \Delta t + \frac{\partial {}^{b}\mathbf{C}_{n}^{(\ell)}}{\partial t} \Delta t = 0. \tag{47} $$

Similarly, the normal complementarity condition (41) becomes:

$$ 0 \leq {}^{u}\mathbf{p}_{n}^{(\ell+1)} \perp {}^{u}\mathbf{C}_{n}^{(\ell)} + ({}^{u}\mathbf{J}_n)^{(\ell)} \mathbf{u}^{(\ell+1)} \Delta t + \frac{\partial {}^{u}\mathbf{C}_{n}^{(\ell)}}{\partial t} \Delta t \geq 0. \tag{48} $$

It only remains to linearize the friction model. However, this 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 friction limit set is a circle of radius $\mu_i \, {}^{u}\mathbf{p}_{in}^{(\ell+1)}$ (as shown in Figure 7b). This circle is approximated by a convex polygon whose vertices are defined by n_d 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 non-negative barycentric coordinates, ${}^{u}\alpha_{ij} \geq 0;\ j = \{1, \ldots, n_{\text{d}}\}$. The interior and boundary of the

linearized friction impulse limit set can be represented as follows:

$$\left.\begin{array}{l} {}^u\mathbf{p}_{if}^{(\ell+1)} = {}^u\mathbf{D}_i\, {}^u\alpha_i \\ \sum_{j=1}^{n_d} {}^u\alpha_{ij} \le \mu_i\, {}^u\mathbf{p}_{in}^{(\ell+1)} \end{array}\right\} \quad \forall\ i \in U, \tag{49}$$

where ${}^u\mathbf{D}_i$ is the matrix whose jth column is the unit vector $\hat{\mathbf{d}}_{ij}$ and ${}^u\alpha_i$ is the vector with jth 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 ${}^u\alpha_{ij} + {}^u\alpha_{ik} = \mu_i\, {}^u\mathbf{p}_{in}^{(\ell+1)}$, 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 linearized limit set using barycentric coordinates. All other coordinate combinations define a friction impulse on the interior of the polygon.

We must also enforce that at sticking contacts, the friction impulse lies 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 sticking 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 ${}^u\beta_i$:

$$0 \le \left(\mu_i\, \mathbf{p}_{in}^{(\ell+1)} - {}^u\mathbf{e}_i^T\, {}^u\alpha_i\right) \ \perp\ {}^u\beta_i \ge 0 \quad \forall\ i \in \mathcal{U}, \tag{50}$$

where $\mathbf{e}_i$ is a vector of 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 non-zero. One way to accomplish this is by the introduction of another complementarity constraint that maps the relative velocity of the friction subspace $v_{if}^{(\ell+1)}$ 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 $v_{if}^{(\ell+1)}$. The following linear complementarity condition, in conjunction with condition (50), identifies the correct $\hat{\mathbf{d}}_{ij}$:

$$\mathbf{0} \le \left({}^u\mathbf{D}_i^T\, {}^u\mathbf{J}_{if}\mathbf{u}^{(\ell+1)} + \mathbf{e}_i\, {}^u\beta_i\right) \ \perp\ {}^u\alpha_i \ge \mathbf{0}\ \forall\ \ i \in \mathcal{U}. \tag{51}$$

Consider for a moment complementarity condition (51) which enforces maximal dissipation. 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 $\hat{\mathbf{d}}_{ij};\ i = 1, ..., n_d$ positively span the friction subspace, and furthermore, ${}^u\beta_i$ approximates the slip speed, with the approximation converging to the exact slip speed as n_d goes to infinity. If ${}^u\beta_i = 0$, all elements of ${}^u\alpha_i$ may be positive, which corresponds to a friction force within the friction cone.

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 (32) and (35) in the instantaneous-time model:

$$\begin{array}{r} \mathbf{0} \le \left({}^u\mathbf{D}^T\, {}^u\mathbf{J}_f\mathbf{u}^{(\ell+1)} + {}^u\mathbf{E}\, {}^u\beta\right) \perp {}^u\alpha \ge \mathbf{0}, \\ \mathbf{0} \le \left(\mathbf{U}\, {}^u\mathbf{p}_n^{(\ell+1)} - {}^u\mathbf{E}^T\, {}^u\alpha\right) \ \perp {}^u\beta \ge \mathbf{0}, \end{array} \tag{52}$$

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 non-zero 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-DOF joints, Equations (34) and (36) of the instantaneous-time model are replaced with the following:

$$\begin{array}{r} \mathbf{0} \le \left({}^b\mathbf{D}^T\, {}^b\mathbf{J}_f\mathbf{u}^{(\ell+1)} + {}^b\mathbf{E}\, {}^b\beta\right) \perp {}^b\alpha \ge \mathbf{0}, \\ \mathbf{0} \le \left({}^b\mathbf{p}_{f\max} - {}^b\mathbf{E}^T\, {}^b\alpha\right) \ \perp {}^b\beta \ge \mathbf{0}, \end{array} \tag{53}$$

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 non-zero blocks given by ${}^b\mathbf{e}_i$, and ${}^b\mathbf{p}_{f\max}$ is $\Delta t\ {}^b\mathbf{f}_{f\max}$.

The above-linearized conditions, taken together, define a time-stepping subproblem as an mLCP (see Definition 4). However, notice that our approach to linearization has decoupled the kinematic map, Equation (45). Therefore, a smaller mLCP can be solved for unknown generalized velocity, impulses and slack variables first, and then the kinematic map can be used to update the system configuration.

Definition 7. (Model-mLCP)**.** *Equations (44), (47), (48), (52) and (53) constitute an mLCP.*

Using the notation of the definition of the standard mLCP given above, the variables of the Model-mLCP are:

$$\mathbf{x}^{\ell+1} = \begin{bmatrix} {}^u\mathbf{p}_n^{\ell+1} \\ {}^b\alpha^{\ell+1} \\ {}^u\alpha^{\ell+1} \\ {}^b\beta^{\ell+1} \\ {}^u\beta^{\ell+1} \end{bmatrix} \qquad \mathbf{w}^{\ell+1} = \begin{bmatrix} \mathbf{u}^{\ell+1} \\ {}^b\mathbf{p}_n^{\ell+1} \end{bmatrix}.$$

The constants are

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & {}^b\mathbf{E}^\ell & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & {}^u\mathbf{E}^\ell \\ \mathbf{0} & -({}^b\mathbf{E}^\ell)^T & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{U} & \mathbf{0} & -({}^u\mathbf{E}^\ell)^T & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} ({}^u\mathbf{J}_n^\ell)^T & ({}^b\mathbf{J}_D^\ell)^T & ({}^u\mathbf{J}_D^\ell)^T & \mathbf{0}\ \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\ \mathbf{0} \end{bmatrix},$$

$$\mathbf{B} = \mathbf{C}^T \qquad \mathbf{D} = \begin{bmatrix} -\mathbf{M}^\ell & ({}^b\mathbf{J}_n^\ell)^T \\ {}^b\mathbf{J}_n^\ell & \mathbf{0} \end{bmatrix},$$

$$\mathbf{a} = \begin{bmatrix} \frac{{}^{u}\mathbf{C}_n^\ell}{\Delta t} + \frac{\partial {}^{u}\mathbf{C}_n^\ell}{\partial t} \\ \frac{\partial {}^{b}\mathbf{C}_f^\ell}{\partial t} \\ \frac{\partial {}^{u}\mathbf{C}_f^\ell}{\partial t} \\ \mathbf{0} \\ {}^{b}\mathbf{p}_{f\max}^\ell \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} \mathbf{M}^\ell \mathbf{u}^\ell + \mathbf{p}_{\text{ext}}^\ell \\ \frac{{}^{b}\mathbf{C}_n^\ell}{\Delta t} + \frac{\partial {}^{b}\mathbf{C}_n^\ell}{\partial t} \end{bmatrix}.$$

It is known that solutions always exist to Model-mLCP if the inequalities are feasible (see [AP97]). Practically speaking, they are not feasible only when bodies are forced into situations in which overlap cannot be avoided. This can be caused by specifying body or joint trajectories as given functions of time or infeasible initial conditions.

3.6. The discrete-time model as an LCP

Model-mLCP can be solved in its current form or it can be reformulated as a standard LCP and then solved. When the null space of ${}^{b}\mathbf{J}_n^{(\ell)}$ is trivial (which is usually true if there are no kinematic loops in mechanisms), then, because $\mathbf{M}$ is symmetric and positive definite, one can solve Equations (44) and (47) for $\mathbf{u}^{(\ell+1)}$ and ${}^{b}\mathbf{p}_n^{(\ell+1)}$.

Definition 8. (Model-LCP)**.** *This model is the LCP(*$\mathbf{A}$, $\mathbf{b}$*), where* $\mathbf{A} = (\mathbf{F} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})$ *and* $\mathbf{b} = \mathbf{a} - \mathbf{B}\mathbf{D}^{-1}\mathbf{r}$.

The size of Model-LCP is the same as that of Model-mLCP even though we have solved for the generalized velocities and normal constraint impulses in the joints in advance. It is also important to note that since Model-LCP was derived from Model-mLCP, their solutions are identical. However, Model-LCP can be solved by pivoting methods such as Lemke's algorithm [CPS92].

Model sizes: The size of a model is the number of unknowns. In the case of models with differential equations, the unknowns are the dependent variables, which are unknown functions of the independent variable. For the models developed above, the independent variable is time and the unknowns are force $\mathbf{f}$ or impulse $\mathbf{p}$, the body velocities $\mathbf{u}$ and the body configurations $\mathbf{q}$. Let n_{bdy}, n_{jnt} and n_{cnt} be the number of bodies, one-DOF joints and unilateral contacts, respectively. Then, the sizes of the models are given in the following table.

Model	Number of unknowns
Model-dNCP	$7n_{\text{jnt}} + 4n_{\text{cnt}} + 13n_{\text{bdy}}$
Model-mNCP	$7n_{\text{jnt}} + 4n_{\text{cnt}} + 13n_{\text{bdy}}$
Model-mLCP	$8n_{\text{jnt}} + (2+n_{\text{d}})n_{\text{cnt}} + 6n_{\text{bdy}}$
Model-LCP	$3n_{\text{jnt}} + (2+n_{\text{d}})n_{\text{cnt}}$

Several of the terms warrant some explanation. The term $7n_{\text{jnt}}$ appears (rather than $6n_{\text{jnt}}$), because in addition to the six components of impulse acting at each joint, there is also an unknown slack variable that determines if the joint is sticking or sliding. The 7 changes to 8 when the model is linearized, because the friction space in each one-DOF joint has two friction directions, and the slack variable is still needed. The 8 then reduces to 3 in Model-LCP, because the five components of normal impulse are solved when converting from an mLCP to an LCP.

In the term $4n_{\text{cnt}}$, the 4 corresponds to the three contact impulse directions and the slack variable that differentiates between sticking and sliding. The 4 changes to $2 + n_{\text{d}}$ in the linearization process. Recall that two friction directions are replaced by n_{d} directions.

The term $13n_{\text{bdy}}$ appears, because we have assumed that the orientations of the bodies are represented by unit quaternions. This term drops to $6n_{\text{bdy}}$ for Model-mLCP, because the linearization process decouples the update of $\mathbf{q}$ from the others, so the impulses and velocities are found by solving the mLCP, then $\mathbf{u}$ is updated. While $\mathbf{q}$ is still technically an unknown, its update is extremely simple in comparison to solving the mLCP.

3.7. Reduced coordinate models

The models presented so far are of a family known as 'maximal coordinate' models, so called, because all six DOFs of every body are represented in the Newton–Euler equation. Maximal coordinate models use bilateral constraints, ${}^{b}\mathbf{C}_{in} = \mathbf{0}$, to eliminate the DOFs removed by joint structures. In maximal coordinate models, one solves for all impulses, velocities and configurations simultaneously. By contrast, minimal coordinate models express as many of the unknowns as possible as functions of independent coordinates, known as generalized coordinates. Minimal coordinate formulations of multi-body dynamics have the added benefit of eliminating all of the bilateral constraints so that constraint stabilization (required in simulation with maximal coordinate formulations) is unnecessary.

The equations of motion for a reduced coordinate model 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 total energy of the system which should be conserved. By the Euler–Lagrange equation

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

we get a system of differential equations for the motion of the bodies which can be solved numerically.

Several reduced coordinate formulations for multi-body systems with unilateral contacts and frictionless joints were developed by Bhalerao *et al.* in [BAT09]; two mLCPs and one LCP. Unlike Model-mLCP presented above, an interesting feature of these formulations is that their sizes are independent of the number of bodies n_{bdy}, but one is dependent on the number of bodies with at least one unilateral contact. This independence derives from the use of Featherstone's Divide and Conquer Algorithm (DCA) [Fea99].

Model	Number of unknowns
Model-DCA-mLCP1	$(2+n_{\text{d}})n_{\text{c}} + 12 + 6n_{\text{u-bdy}}$
Model-DCA-mLCP2	$(2+n_{\text{d}})n_{\text{c}} + 12$
Model-DCA-LCP	$(2+n_{\text{d}})n_{\text{c}}$

Note that these formulations did not incorporate dry joint friction. However, since all bilateral constraints are eliminated by the DCA algorithm, the number of unknowns in the joints should be $3n_{\text{jnt}}$ for all three models.

Comparing these formulations to Model-mLCP and Model-LCP, we find that all the quantities relating to dry friction and unilateral contacts ($\mathbf{x}^{\ell+1}$, $\mathbf{F}$ and $\mathbf{a}$) are unchanged. The normal constraint impulse of the bilateral contacts (${}^b\mathbf{p}_n^{(\ell+1)}$) is absent, which eliminates the second block row of $\mathbf{D}$, $\mathbf{C}$ and $\mathbf{r}$, and the second block column of $\mathbf{D}$ and $\mathbf{B}$. The remaining changes are in the inertia matrix $\mathbf{M}$, the Jacobian matrices in $\mathbf{C}$ (and $\mathbf{B}$), the body velocity vector $\mathbf{u}$ and the kinematic velocity map $\mathbf{H}$. In the case of Model-DCA-LCP, $\mathbf{M}$ and $\mathbf{u}$ become vacuous, and the Jacobians become very dense. For model-DCA-mLCP2, $\mathbf{M}$ and $\mathbf{u}$ become (12×12) and (12×1), respectively. In Model-DCA-mLCP1, the sizes grow to $6n_{\text{u-bdy}} + 12$, and the Jacobians are the least dense.

So, one might wonder, why bother with maximal coordinate models at all. The main reasons are ease of implementation and sparseness of the matrices of the model, which can be exploited for more efficient matrix inversion and parallel implementation (see Section 5).

4. The Numerical Solution Methods

Once discrete-time 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 4.1- 4.4 we cover numerical methods for computing solutions of the discrete LCP contact model from Section 3.1 and approaches for simulating articulated bodies. The methods are presented in a general setting; hence, $\mathbf{A}$ and $\mathbf{b}$ are arbitrary as defined in (23).

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 numerical integration methods. In contrast to the well-known explicit Euler, the semi-implicit Euler uses the velocity at time $t_0 + \Delta t$ instead of time t_0 for the integration of the position vector:

$$\mathbf{u}(t_0 + \Delta t) = \mathbf{u}(t_0) + \Delta t\, \mathbf{M}^{-1}\, \mathbf{g}(\mathbf{q}, \mathbf{u}, t_0),$$
$$\mathbf{q}(t_0 + \Delta t) = \mathbf{q}(t_0) + \Delta t\, \mathbf{H}\mathbf{u}(t_0 + \Delta t),$$

where $\mathbf{M}$, $\mathbf{g}(\mathbf{q}, \mathbf{u}, t)$ and $\mathbf{H}$ are defined by Equations (1), (10) and (11). 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] to solve the initial value problem given by the equation of motion. For more details, we refer to [BETC12].

4.1. Direct methods

Direct solution methods are known to be computationally heavy. Therefore, they are often not preferred for interactive simulation. However, their ability to deliver accurate solutions makes them ideal to handle problems such as large mass ratios. Thus, for some applications, direct methods are the only option. Among direct methods for LCPs based on pivoting are the Lemke method and the Keller method [CPS92], [Lac03].

Here, we will present a core idea of most pivoting methods, namely a guessing approach that exploits the fact that an LCP is a combinatorial problem. The LCP can be written as

$$\mathbf{0} \le \mathbf{y} \;\perp\; \mathbf{x} \ge \mathbf{0},$$

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

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

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

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

We assume strict complementarity holds meaning that 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 submatrices given by the column indices $\mathcal{F}$ and $\mathcal{A}$. Our problem is simplified to verifying if the linear programming (LP) problem

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

has a solution. This can be done by first computing $\mathbf{x}_{\mathcal{A}} = -\mathbf{A}_{\mathcal{A}\mathcal{A}}^{-1}\mathbf{b}_{\mathcal{A}}$, and verify if $\mathbf{x}_{\mathcal{A}} \ge 0$. 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 $\mathbf{y}_{\mathcal{F}} \ge 0$. If that last verification succeeds, then a solution has been found. Observe that during the verification processes, we only need to compute $\mathbf{A}_{\mathcal{A}\mathcal{A}}^{-1}$. If $\| \mathcal{A} \| \ll n$, then verification will be fast.

In the worst case, the time complexity of guessing would be $\mathcal{O}(n^3 2^n)$, which is not computationally efficient. Another strategy is to be clever in making new guesses. For instance, by applying a pivoting or other strategy that builds up the index sets incrementally. One such algorithm was introduced by Baraff [Bar94]. For this algorithm, one can exploit incremental matrix factorizations and prove that no more than n pivot steps are needed when $\mathbf{A}$ has certain matrix properties. The result is an algorithm running in $\mathcal{O}(n^3)$. For full algorithm detail and complexity analysis, we refer to [BETC12].

The pivoting method is capable of finding an accurate solution for the LCP, whereas the iterative methods we cover in Sections 4.2 and 4.3 only find approximate solutions. However, the accuracy is at the expense of having to form the $\mathbf{A}$-matrix, whereas the iterative methods often exploit a factorization of the $\mathbf{A}$-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

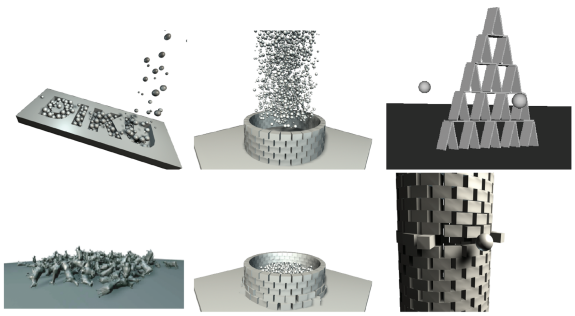


Figure 8: *More complex interacting geometry showing robustness.*

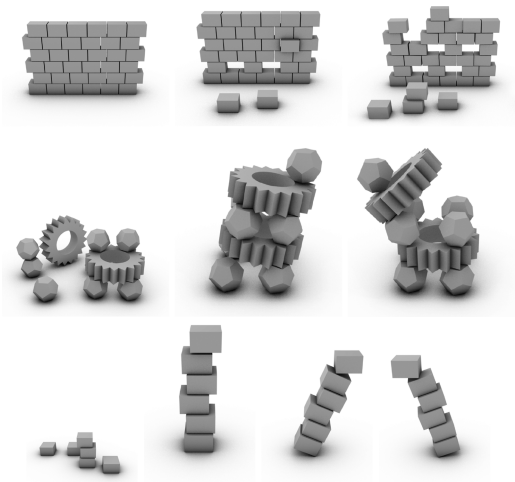


Figure 9: *Goal-oriented task-based testing for interactivity. Stacks of objects are to be created and tipped over without falling down.*

A-matrix which can be very dense even if it consists of products of sparse matrices.

4.2. 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 ODE. PGS is computationally very efficient with an iteration cost of $\mathcal{O}(n)$, using a careful memory layout of sparse matrices allows for a memory footprint of $\mathcal{O}(n)$. In addition to being computationally and memory-wise efficient, PGS is very robust (see Figure 8) and can deal gracefully with ill-conditioned problems (due to many redundant constraints) or ill-posed problems (due to badly defined constraints). For these reasons, PGS is well suited for interactive applications like computer games. Figure 9 shows different interactive tasks, which are accomplished in a simulation with a PGS solver.

4.2.1. *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

$$\mathbf{0} \leq \mathbf{x} \perp \mathbf{A}\mathbf{x} + \mathbf{b} \geq \mathbf{0}, \tag{54}$$

becomes

$$\mathbf{0} \leq \mathbf{x}^{k+1} \perp \mathbf{M}\mathbf{x}^{k+1} + \mathbf{c}^k \geq \mathbf{0}. \tag{55}$$

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 (55).
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 where the corresponding LCP has a solution for all vectors $\mathbf{c}^k$ [Mur88], [CPS92]. Clearly, if $\mathbf{x}^{k+1}$ is a solution for (55) and we have $\mathbf{x}^{k+1} = \mathbf{x}^k$, then by substitution into the subproblem given by (55), we see that $\mathbf{x}^{k+1}$ is a solution of the original problem (54).

Next, we will use the minimum map reformulation on the complementarity subproblem [Erl13], this is equivalent to

$$\min(\mathbf{x}^{k+1}, \mathbf{M}\mathbf{x}^{k+1} + \mathbf{c}^k) = \mathbf{0}. \tag{56}$$

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

$$\max(\mathbf{0}, -\mathbf{M}\mathbf{x}^{k+1} - \mathbf{c}^k + \mathbf{x}^{k+1}) = \mathbf{x}^{k+1}. \tag{57}$$

Again, we re-discover a fixed-point formulation. Let us perform a case-by-case analysis of the ith component. If

$$\left(\mathbf{x}^{k+1} - \mathbf{M}\mathbf{x}^{k+1} - \mathbf{c}^k\right)_i < 0, \tag{58}$$

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

$$\left(\mathbf{x}^{k+1} - \mathbf{M}\mathbf{x}^{k+1} - \mathbf{c}^k\right)_i = \mathbf{x}_i^{k+1}. \tag{59}$$

That is,

$$(\mathbf{M}\mathbf{x}^{k+1})_i = \mathbf{c}_i^k. \tag{60}$$

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

$$\left(\mathbf{M}^{-1}\left(\mathbf{N}\mathbf{x}^k - \mathbf{b}\right)\right)_i = \mathbf{x}_i^{k+1}. \tag{61}$$

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

$$\max\left(\mathbf{0}, \left(\mathbf{M}^{-1}\left(\mathbf{N}\mathbf{x}^k - \mathbf{b}\right)\right)\right) = \mathbf{x}^{k+1}. \tag{62}$$

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

$$\mathbf{x}^{k+1} = \max\left(\mathbf{0}, \mathbf{z}^k\right). \tag{63}$$

That is, the $k+1$ iteration is obtained by projecting the vector $\mathbf{z}^k$ onto the non-negative orthant. In a practical implementation, one would rewrite (63) into a for loop that sweeps over the vector components and updates the $\mathbf{x}$-vector in place.

One would want to use a clever splitting such that the inversion of $\mathbf{M}$ is computationally inexpensive. 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. For PSOR, $\mathbf{b}$ is replaced by $\gamma\mathbf{b}$ and $\gamma\mathbf{A} = \mathbf{M} - \mathbf{N}$. This is because the linear relation is written as $\gamma\,(\mathbf{A}\mathbf{x} + \mathbf{b})$.

Using the PSOR variant (with PGS as a special case of $\gamma = 1$) and rewriting the matrix update equation into a for loop over the ith component, then one obtains the iterative method

1 : **method PSOR**(N, γ, $\mathbf{x}$, $\mathbf{A}$, $\mathbf{b}$)
2 : **for** $k = 1$ **to** N
3 : **for all** i
4 : $\mathbf{r}_i \leftarrow \mathbf{A}_{i*}\mathbf{x} + \mathbf{b}_i$
5 : $\mathbf{x}_i \leftarrow \max\left(0, \mathbf{x}_i - \gamma \dfrac{\mathbf{r}_i}{\mathbf{A}_{ii}}\right)$
6 : **next** i
7 : **next** k

where N is the maximum number of allowed iterations and γ is the relaxation parameter. In [BETC12], it is shown how quadratic programming problems can be used to derive this method. In the case of $\mathbf{A}$ being symmetric positive semi-definite, it can be shown that the method will always converge to a solution.

It is worthwhile to note that $\mathbf{A}$ must at least have non-zero diagonal for these splittings to work. In general for non-symmetric matrices, one may experience divergence. This means we cannot 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 subspace minimization [SNE10b] or a non-smooth non-linear conjugate gradient method [SNE10a].

It seems that all hope of using matrix splitting for the LCP model is lost. However, as we show in Sections 4.2.2 and 4.2.3 a blocked version of the matrix splittings can be used for the LCP model.

4.2.2. *The blocked Gauss–Seidel (BGS) method*

The matrix splitting and QP reformulation approaches imply that Gauss–Seidel methods cannot 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 ith 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 $\left[\mathbf{x}\right]_i = \left[\mathbf{x}_{n,i}\ \mathbf{x}_{t_1,i}\ \cdots\ \beta_i\right]^T$. Similar $\left[\mathbf{A}\right]_{ij}$ is the sub-block of $\mathbf{A}$ corresponding to the ith and jth contact point variables. Thus, the blocked LCP can be written:

$$\left[\mathbf{y}\right]_i = \sum_j \left[\mathbf{A}\right]_{ij}\left[\mathbf{x}\right]_j + \left[\mathbf{b}\right]_i \geq \mathbf{0} \quad \forall i, \tag{64a}$$

$$[-2mm]\left[\mathbf{x}\right]_i \geq \mathbf{0} \quad \forall i, \tag{64b}$$

$$\left[\mathbf{y}\right]_i^T\left[\mathbf{x}\right]_i = 0 \quad \forall i. \tag{64c}$$

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

1 : **method BGS**(N, $\mathbf{x}$, $\mathbf{A}$, $\mathbf{b}$)
2 : **for** $k = 1$ **to** N
3 : **for all** i
4 : $\left[\mathbf{b}\right]_i' \leftarrow \left[\mathbf{b}\right]_i - \sum_{j\neq i}\left[\mathbf{A}\right]_{ij}\left[\mathbf{x}\right]_j$
5 : **solve-sub-lcp**($\left[\mathbf{x}\right]_i$, $\left[\mathbf{A}\right]_{ii}$, $\left[\mathbf{b}\right]_i'$)
6 : **next** i
7 : **next** k

The intuition behind the numerical method is that all contact point variables other than the ith block are momentarily frozen while solving for the variables of the ith 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. Usually, one would apply another splitting to divide 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) 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 [BETC12]. The main difference is that this is a numerical method for solving a simultaneous contact model, whereas the other is a model in itself. The former solves for force impulses, whereas the latter solves for collision impulses. The similarity with

the propagation model also gives intuition to some of the traits of the numerical method. One may see propagation effects even though one is using a simultaneous model.

The BGS method offers many possibilities. In Section 4.2.3 we divide an LCP into two sub-blocks: one with normal variables only and the other containing the rest. In fact, one may use any kind of partitioning 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 pre-conditioned 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 4.4 In the extreme case, BGS can be used to partition a configuration into sub-blocks where one can apply specialized solvers for each sub-block. Such approaches have been termed hierarchical solvers by the graphics and gaming community.

4.2.3. *A staggered approach*

One may combine the ideas of splitting the LCP and using QP reformulations. The idea is referred to as staggering [Löt84], [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,

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

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 a BGS 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 a non-symmetric matrix. One may exploit a QP reformulation anyway, because the friction LCP is the first-order optimality conditions of the QP problem

$$\mathbf{x}^*_{\mathcal{F}} = \arg\min \frac{1}{2}\mathbf{x}^T_{\mathcal{F}}\mathbf{A}_{\mathcal{F}\mathcal{F}}\mathbf{x}_{\mathcal{F}} + \mathbf{c}^T_{\mathcal{F}}\mathbf{x}_{\mathcal{F}} \tag{65}$$

subject to

$$\mathbf{x}_{\mathcal{F}} \geq \mathbf{0} \quad \text{and} \quad c_{\mathcal{N}} - \mathbf{e}^T\mathbf{x}_{\mathcal{F}} \geq 0, \tag{66}$$

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 QP subproblems converge to a fixed point is not obvious.

There exist many variations of this staggering scheme [LL11]. One variation is to use a BGS method for the frictional problem rather than a QP reformulation. This is mostly due to performance. Using a QP solver for the normal problem helps to find accurate normal forces, which are important for systems with with large mass ratios among the bodies. In some interactive applications, accurate friction forces are not as important, which means a Gauss–Seidel method is suitable for the friction problem.

4.3. Newton methods

The PGS methods from Section 4.2 may suffer from viscous artefacts due to linear convergence rate. An alternative is to use Newton methods. These can provide quadratic convergence rates and thus offer more accurate solutions at a slightly higher per iteration computational cost than PGS methods. PATH [Pat05] is a well-known Newton-type solver for NCPs 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

$$\phi(a, b) = \sqrt{a^2 + b^2} - (a + b) \quad \text{for } a, b \in \mathbb{R}. \tag{67}$$

If one has the complementarity problem $0 \leq a \perp b \geq 0$, a solution (a^*, b^*) is 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

$$\mathbf{0} \leq \mathbf{x} \quad \perp \quad \mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b} \geq \mathbf{0}, \tag{68}$$

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 non-smooth root search problem

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \phi(\mathbf{x}_1, \mathbf{y}_1) \\ \vdots \\ \phi(\mathbf{x}_n, \mathbf{y}_n) \end{bmatrix} = \mathbf{0}. \tag{69}$$

Thus, our problem is changed to that of finding the root of a non-linear non-smooth equation. This problem may be solved using a generalized Newton method which is an iterative method. In the kth iteration, the Newton method solves the generalized Newton system

$$\mathbf{J}\Delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^k) \tag{70}$$

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 $\partial\mathbf{F}(\mathbf{x})$, for details, see [BETC12]. After having computed the Newton direction, one performs a Newton update to obtain the next iterate,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \tau^k\Delta\mathbf{x}^k. \tag{71}$$

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

The Clarke-generalized Jacobian of the Fischer reformulation (69) can be written as

$$\partial F(\mathbf{x}) \equiv \mathbf{D}_a(\mathbf{x}) + \mathbf{D}_b(\mathbf{x})\mathbf{A}, \tag{72}$$

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

$$a_i(\mathbf{x}) = \frac{\mathbf{x}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1, \quad b_i(\mathbf{x}) = \frac{\mathbf{y}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1, \tag{73}$$

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

$$a_i(\mathbf{x}) = \alpha_i - 1, \quad b_i(\mathbf{x}) = \beta_i - 1, \tag{74}$$

for any $\alpha, \beta \in \mathbb{R}$ such that $\| \begin{bmatrix} \alpha_i & \beta_i \end{bmatrix}^T \| \leq 1$. For proof, see [BETC12].

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 ith unit axis vector as the ith row of $\mathbf{J}$. A more practical implementation approach would simply consist in whenever $\mathbf{x}_i = \mathbf{y}_i = 0$ one would use $\mathbf{x}_i' = \mathbf{x}_i + \varepsilon$ in-place 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 [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

$$\Psi(\mathbf{x}^k + \Delta\mathbf{x}^k) \leq \Psi(\mathbf{x}^k) + c\tau^k \nabla\Psi(\mathbf{x}^k)^T \Delta\mathbf{x}^k, \tag{75}$$

where the sufficient decrease parameter is $c \in (0, 1)$ and 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 (75) is satisfied. The back-tracking approach starts with the guess of $\tau^k = 1$ and then tests if (75) holds. If not, τ^k is reduced by a step reduction factor and the test is repeated. This continues until the test passes and one will have obtained the final value τ^k.

In comparison with the described Fischer–Newton method, we note that PATH is also based on a Fischer function reformulation of a linearized boxed NCP formulation and uses a non-monotome line search method.

4.4. Articulated bodies and jointed mechanics

In the following, we introduce methods to simulate articulated bodies. An articulated body is a system of rigid bodies connected by joints (see Figure 10). Joints define bilateral constraints ${}^b\mathbf{C} = \mathbf{0}$ (see Section 2.1.3). There are two main approaches for the simulation of articulated bodies: the reduced (or generalized) coordinate formulation and the maximal coordinate formulation (see Section 3.7).

4.4.1. *Maximal coordinate formulation*

Here, we introduce a maximal coordinate formulation and, because solutions of such formulations allow joint constraint errors to build over time, we also present one possible joint error correction method.

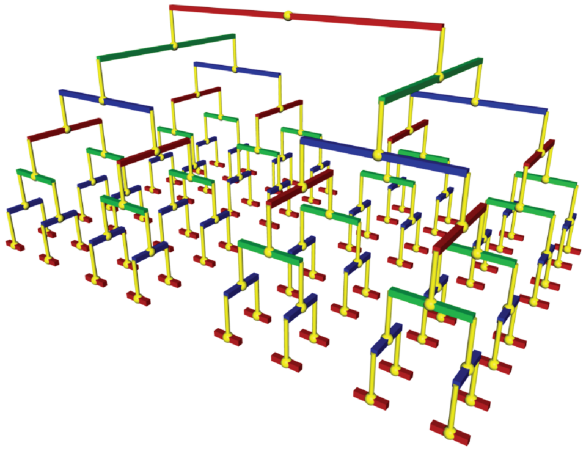


Figure 10: *This articulated body is a tree of rigid bodies which are connected by spherical joints.*

Lagrange multipliers: The bilateral constraints in the system are maintained by reaction forces. These forces can be viewed as Lagrange multipliers (see Section 3.1). To compute the reaction forces, the bilateral constraints are transformed in the general constraint form:

$$\mathbf{J}(\mathbf{q}, \mathbf{u}, t)\,\dot{\mathbf{u}} + \mathbf{k}(\mathbf{q}, \mathbf{u}, t) = \mathbf{0}. \tag{76}$$

In an n-dimensional system with an m-dimensional constraint, the matrix $\mathbf{J}$ has dimension $(m \times n)$ and the vector $\mathbf{k}$ has dimension m. A holonomic constraint is transformed in the general form by differentiating the constraint function ${}^b\mathbf{C}$ twice with respect to time (cf. Equation 28).

The Lagrange multipliers λ are determined by substituting the equation of motion $\dot{\mathbf{u}} = \mathbf{M}^{-1}(\mathbf{f}_{\text{ext}} + {}^b\mathbf{f})$ into the general constraint (76), where ${}^b\mathbf{f}$ are the constraint forces. Regarding D'Alembert's principle [GPS02], it follows that ${}^b\mathbf{f} = \mathbf{J}^T\lambda$. Hence, the constraint forces always act in the constrained directions of a system. Such forces do not influence the motion of the $n - m$ DOFs of an articulated body. Finally, we get a system of linear equations for the Lagrange multipliers which are required to determine ${}^b\mathbf{f}$:

$$\underbrace{\mathbf{J}\,\mathbf{M}^{-1}\,\mathbf{J}^T}_{\mathbf{A}}\,\lambda = \underbrace{-\mathbf{J}\,\mathbf{M}^{-1}\,\mathbf{f}_{\text{ext}} - \mathbf{k}}_{\mathbf{b}}. \tag{77}$$

The matrix $\mathbf{A}$ is positive definite if there are no conflicting or redundant constraints. Furthermore, $\mathbf{A}$ is sparse for most models since it reflects the structure of the articulated body.

David Baraff's method [Bar96] allows the simulation of articulated bodies without closed loops in linear time. The system of linear Equation (77) is rewritten as

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

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. Observe that $\mathbf{A}$ is the Schur matrix of $\mathbf{K}$. $\mathbf{A}$ has a row and column

for each constraint, while $\mathbf{K}$ has a row and column for each DOF and each constraint. The advantage of this 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-first 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, an $\mathbf{LDL}^T$ decomposition is performed. Due to the reordered matrix structure, the decomposition introduces that no new non-zero elements can be stored in linear space and performed in linear time. The decomposition allows for solving the linear system for the Lagrange multipliers in linear time.

The Lagrange multiplier method computes constraint forces in order to prevent a violation of constraints due to external forces. However, if the constraints are violated in a different way (e.g. by errors that occur during numerical integration), the method cannot correct this. The problem is that a constraint is not regarded directly. Instead, it is demanded that its second derivative is zero. Therefore, an additional stabilization method is required to prevent joints from breaking due to numerical errors. The method of Baumgarte [Bau72] adds two penalty terms to Equation (76). These terms depend on the constraint function and its first time derivative to consider position and velocity errors. 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 and propose an enhanced stabilization method [ACPR95]. This has also been explored for NCP and LCP models [ST96], [CP03].

Impulse-based error correction: The impulse-based error correction [BFS05], [BS06b], [Ben07], [WTF06] is similar to the Lagrange multiplier method. The main difference between these methods is that the impulse-based approach determines constraint impulses by using a prediction of the final state, while the Lagrange multiplier method computes additional forces or impulses based on the current state. In this sense, it is like a mid-point method that considers information from both ends of the time step to compute the update.

By differentiating a bilateral constraint function of a joint with respect to time, we get a general constraint form for velocities $\mathbf{Ju} + \mathbf{k} = \mathbf{0}$ which is analogous to the one of Equation (76). Now a system of linear equations for the impulses could be created which is analogous to the one of Equation (77). However, the impulse-based error correction 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 11 shows the predicted state of a ball joint with the bilateral constraint ${}^b\mathbf{C} = \mathbf{P}_1 - \mathbf{P}_2 = \mathbf{0}$. For the prediction, we assume that both rigid bodies are unconstrained. Then, the predicted position of a joint point $\mathbf{P}(t + \Delta t)$ is determined in two steps. First, we solve the differential equation $\dot{\mathbf{r}} = \omega \times \mathbf{r}$ for the vector $\mathbf{r}(t) = \mathbf{P}(t) - \mathbf{x}(t)$, where $\mathbf{x}$ is the centre of mass. Second, we solve the equation of motion for the centre of mass and determine the new

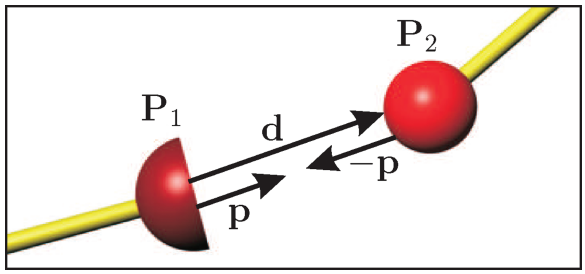


Figure 11: *Predicted state of a ball joint. The points* $\mathbf{P}_1$ *and* $\mathbf{P}_2$ *have different positions which must be corrected by a pair of impulses* $\mathbf{p}$ *and* $-\mathbf{p}$.

position as $\mathbf{P}(t + \Delta t) = \mathbf{r}(t + \Delta t) + \mathbf{x}(t + \Delta t)$. For the predicted state, we evaluate the constraint function and get the drift vector $\mathbf{d}(t + \Delta t) = {}^b\mathbf{C}(t + \Delta t)$. 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 joint points so that the constraint ${}^b\mathbf{C}(t + \Delta t) = \mathbf{0}$ will be fulfilled.

The required impulses for a constraint can be determined by solving a non-linear equation by Newton iteration [WTF06]. In a system with multiple constraints, there exist dependencies between constraints with a common body. These dependencies are handled in an iterative way by Weinstein *et al*. In contrast, Bender *et al*. linearize the equation by approximating the required velocity change as $\Delta\mathbf{v} = \mathbf{d}(t + \Delta t)/\Delta t$. Bender *et al*. use this value as an approximation for the non-linear 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 of velocity changes for all constraints. This system of linear equations can be solved in linear time for articulated bodies without loops [Bar96], [Ben07]. Loops can be handled by splitting the model in acyclic parts [BB08]. In [BS06a], impulse-based correction is extended by inequality constraints in order to simulate collisions and resting contacts. Bayer *et al*. [BDB09] present different performance optimizations for the impulse-based method. Numerical comparisons of the impulse-based approach with other methods can be found in [SB05] and [SBP05].

4.4.2. *Reduced coordinate formulation*

We introduced the reduced coordinate model in Section 3.7 In the following, we will discuss some methods for the simulation with this model.

Since methods based on Lagrange formulation (see Section 3.7) have a complexity of $\mathcal{O}(n^4)$ [FO00], more efficient approaches were investigated. An overview over such approaches 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 three phases. In the first phase, the velocity and bias terms are

determined in a top–down traversal of the tree. Then, the tree is traversed in reverse to compute articulated-body inertias and bias forces. Finally, the accelerations are determined in a second top–down traversal. To provide a compact notation, Featherstone introduced the spatial vector algebra. Spatial vectors are six-dimensional and combine the linear and angular aspects of rigid body motions and forces. A detailed introduction to the spatial vector algebra and the ABA of Featherstone can be found in [Mir96] 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 one to reduce the numbers of DOFs (at the cost of accuracy), while it automatically determines the best set of active joints.

The DOFs of a closed-loop model can vary and forces in such a model can be indeterminate when the system is over-constrained. Therefore, these models need 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].

5. Parallel Processing and Optimizations

Parallelization is an important topic since multi-core systems and massively parallel GPUs are very common today. For the simulation of bilateral constraints, one has to solve a system of linear equations (see Section 4.4). 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], [BCL09] and [WBS*13]. To achieve a high performance, a good memory layout of these structures is very important. Weber *et al*. [WBS*13] demonstrated that reducing the kernel calls can further improve the performance. The optimized matrix operations allow the efficient solution of sparse systems which generally occur in multi-body simulations with bilateral constraints. Another approach was presented by Bayer *et al*. [BBD09]. They create groups of independent constraints in a pre-computation step. Then, all constraints in a group are solved 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].

The parallel computation of contact forces was 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 response, Harada used a discrete element method where repulsive, damping and shear forces are computed for colliding particles. In [CA09], a parallel Gauss–Seidel iteration method for dense matrices is introduced. This method works on multi-core processors and GPUs. Tasora *et al*. [TNA08], [TNA*10] used a cone complementarity problem formulation instead of an NCP. They 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 and ignored race conditions completely. Harada showed how to efficiently solve this problem by partitioning, synchronizing and scheduling the operations using *local atomics* within each *compute unit* [Har11]. More recently, Tonge *et al*. [TBV12] approached the problem differently by decomposing rigid bodies into smaller parts, thereby eliminating race conditions altogether.

6. Conclusion and Future Work

Interactive rigid body simulations have become an important part in different application areas. Such simulations require 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 environment by simulating bilateral and unilateral constraints 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. However, controlling the simulation is still a problem where much work has to be done.

Acknowledgements

The work of Jan Bender was supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School CE at TU Darmstadt. The work of Trinkle was supported in part by NSF CCF-1208468, DARPA W15P7T-12-1-0002 and Lockheed-Martin's Advanced Technologies Laboratory.

References

[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.

[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.

[AP97] 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.

[Bar89] Baraff D.: Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proceedings of SIGGRAPH* (1989).

[Bar93] Baraff D.: Non-penetrating rigid body simulation. In *State of the Art Reports: Proceedings of Eurographics '93* (September 1993). Eurographics Association, Barcelona, Spain.

[Bar94] Baraff D.: Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of SIGGRAPH* (1994).

[Bar96] Baraff D.: Linear-time dynamics using lagrange multipliers. In *Proceedings of SIGGRAPH* (New York, NY, USA, 1996), ACM Press, pp. 137–146.

[BAT09] Bhalerao K., Anderson K., Trinkle J.: A recursive hybrid time-stepping scheme for intermittent contact in multi-rigid-body dynamics. *ASME Journal of Computational and Nonlinear Dynamics 4*, 4 (2009), 1–11.

[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.

[BB08] Bender J., Bayer D.: Parallel simulation of inextensible cloth. In *Proceedings of the Workshop on Virtual Reality Interactions and Physical Simulations* (Grenoble, France, November 2008), pp. 47–56.

[BBD09] Bayer D., Bender J., Diziol R.: Impulse-based dynamic simulation on the GPU. In *Proceedings of Computer Graphics and Visualization* (Algarve, Portugal, 2009).

[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, San Francisco, CA, USA, 1991.

[BCL09] Buatois L., Caumon G., Levy B.: Concurrent number cruncher: A GPU implementation of a general sparse linear solver. *International Journal of Parallel, Emergent and Distributed Systems 24* (June 2009), 205–223.

[BDB09] Bayer D., Diziol R., Bender J.: Optimized impulse-based dynamic simulation. In *Proceedings of the Workshop on Virtual Reality Interactions and Physical Simulations* (2009), pp. 125–133.

[Ben07] Bender J.: Impulse-based dynamic simulation in linear time. *CAVW 18*, 4–5 (2007), 225–233.

[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.

[BETC12] Bender J., Erleben K., Trinkle J., Coumans E.: Interactive simulation of rigid body dynamics in computer graphics. In *Proceedings of EG 2012—State of the Art Reports* (2012), Eurographics Association, pp. 95–134.

[BFGS03] Bolz J., Farmer I., Grinspun E., Schröder P.: Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. *ACM Transactions on Graphics 22* (2003), 917–924.

[BFS05] Bender J., Finkenzeller D., Schmitt A.: An impulse-based dynamic simulation system for VR applications. In *Proceedings of Virtual Concept 2005* (2005), Springer.

[BG09] Bell N., Garland M.: Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *Proceedings of the Conference for High Performance Computing, Networking, Storage and Analysis* (2009).

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

[BS06a] Bender J., Schmitt A.: Constraint-based collision and contact handling using impulses. In *Proceedings of the Conference on Computer Animation and Social Agents* (2006), pp. 3–11.

[BS06b] Bender J., Schmitt A.: Fast dynamic simulation of multibody systems using impulses. In *Proceedings of the Workshop on Virtual Reality Interactions and Physical Simulations* (2006), pp. 81–90.

[BWAK03] Baraff D., Witkin A., Anderson J., Kass M.: Physically based modeling. Siggraph Course Notes, 2003.

[CA09] Courtecuisse H., Allard J.: Parallel dense Gauss-Seidel algorithm on many-core processors. In *Proceedings of High-Performance Computing and Communications* (2009), IEEE CS Press.

[CBAT13] Chakraborty N., Berard S., Akella S., Trinkle J.: A geometrically implicit time-stepping method for multibody systems with intermittent contact. *International Journal of Robotics Research 32* (2013). doi: 10.1177/0278364913501210.

[CMT04] Carlson M., Mucha P. J., Turk G.: Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics 23* (August 2004), 377–384.

[Con62] Contensou P.: Couplage entre frottement de glissement et frottement de pivotement dans la théorie de la toupie. In

Kreiselprobleme und Gyrodynamics, IUTAM Symposium Celerina. Springer-Verlag, Berlin (1962), pp. 201–216.

[Cou13] COUMANS E.: The bullet physics library. http://www.bulletphysics.org, 2013. Accessed on December 2013.

[CP03] CLINE M., PAI D.: Post-stabilization for rigid body simulation with contact and constraints. In *Proceedings of IEEE ICRA* (2003), pp. 3744–3751.

[CPN11] CPNET: Complementarity problem net. http://www.cs.wisc.edu/cpnet, 2011. Accessed on December 2011.

[CPS92] COTTLE R., PANG J.-S., STONE R. E.: *The Linear Complementarity Problem*. Academic Press, Boston, MA, USA, 1992.

[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.

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

[Erl07] ERLEBEN K.: Velocity-based shock propagation for multibody dynamics animation. *ACM Transactions on Graphics 26*, 2 (2007), 1–20.

[Erl11] ERLEBEN K.: num4lcp. Published online at code.google.com/p/num4lcp/, October 2011. Numerical methods for LCPs in physics-based animation.

[Erl13] ERLEBEN K.: Numerical methods for linear complementarity problems in physics-based animation. In *Proceedings of ACM SIGGRAPH 2013 Courses* (2013), pp. 1–42.

[ESHD05] ERLEBEN K., SPORRING J., HENRIKSEN K., DOHLMANN H.: *Physics-Based Animation*. Charles River Media, Hingham, MA, USA, Aug. 2005.

[Fea87] FEATHERSTONE R.: *Robot Dynamics Algorithms*. Kluwer International Series in Engineering and Computer Science: Robotics. Kluwer Academic Publishers, Boston/Dordrecht/Lancaster, 1987.

[Fea99] FEATHERSTONE R.: A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics. Part 1: Basic algorithm. *The International Journal of Robotics Research 18*, 9 (1999), 867–875.

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

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

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

[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.

[GPS02] GOLDSTEIN H., POOLE C., SAFKO J.: *Classical Mechanics*. Addison Wesley, Reading, MA, USA, 2002.

[Had06] HADAP S.: Oriented strands: Dynamics of stiff multi-body system. In *Proceedings of SCA* (2006), pp. 91–100.

[Hah88] HAHN J. K.: Realistic animation of rigid bodies. In *Proceedings of SIGGRAPH* (1988).

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

[Har11] HARADA T.: A parallel constraint solver for a rigid body simulation. In *Proceedings of SIGGRAPH Asia Sketches* (2011).

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

[KEP05] KAUFMAN D. M., EDMUNDS T., PAI D. K.: Fast frictional dynamics for rigid bodies. *ACM Transactions on Graphics 24*, 3 (2005), 946–956.

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

[KSJP08] KAUFMAN D. M., SUEDA S., JAMES D. L., PAI D. K.: Staggered projections for frictional contact in multibody systems. *ACM Transactions on Graphics 27*, 5 (2008), 164:1–164:11.

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

[Lac03] LACOURSIERE C.: Splitting methods for dry frictional contact problems in rigid multibody systems: Preliminary performance results. In *Proceedings of the Annual SIGRAD Conference* (Linköping University, Electronic Press, 2003), Ollila M. (Ed.).

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

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

[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.

[LO08] LIN M. C., OTADUY M. (Eds.): *Haptic Rendering: Foundations, Algorithms, and Applications*. A K Peters/CRC Press, July 2008.

[Löt84] Lötstedt P.: Numerical simulation of time-dependent contact and friction problems in rigid body mechanics. *SIAM Journal of Scientific and Statistical Computing 5*, 2 (1984), 370–393.

[Mei70] Meirovitch L.: *Methods of Analytical Dynamics*. McGraw-Hill, New York, USA, 1970.

[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.

[MHHR07] Müller M., Heidelberger B., Hennix M., Ratcliff J.: Position based dynamics. *Journal of Visual Communication and Image Representation 18*, 2 (2007), 109–118.

[Mir96] Mirtich B. V.: Impulse-Based Dynamic Simulation of Rigid Body Systems. PhD thesis, University of California, Berkeley, 1996.

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

[Mur88] Murty K. G.: *Linear Complementarity, Linear and Nonlinear Programming*. Helderman-Verlag, Berlin, Germany, 1988.

[MW88] Moore M., Wilhelms J.: Collision detection and response for computer animation. In *Proceedings of SIGGRAPH* (1988).

[MZS09] Macchietto A., Zordan V., Shelton C. R.: Momentum control for balance. *ACM Transactions on Graphics 28* (2009), 80:1–80:8.

[NW99] Nocedal J., Wright S. J.: *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, USA, 1999.

[Pat05] Path: Path cpnet software, 2005. http://pages.cs.wisc.edu/~ferris/path.html. Accessed on December 2013.

[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).

[PSE03] Popović J., Seitz S. M., Erdmann M.: Motion sketching for control of rigid-body simulations. *ACM Transactions on Graphics 22* (October 2003), 1034–1054.

[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.

[RGL05] Redon S., Galoppo N., Lin M. C.: Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics 24* (July 2005), 936–945.

[RMEF09] Robinson-Mosher A., English R. E., Fedkiw R.: Accurate tangential velocities for solid fluid coupling. In *Proceedings of SCA* (2009), pp. 227–236.

[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 Transactions on Graphics 27* (August 2008), 46:1–46:9.

[SB05] Schmitt A., Bender J.: Impulse-based dynamic simulation of multibody systems: Numerical comparison with standard methods. In *Proceedings of Automation of Discrete Production Engineering* (2005).

[SBP05] Schmitt A., Bender J., Prautzsch H.: On the Convergence and Correctness of Impulse-Based Dynamic Simulation. Internal Report 17, Universität Karlsruhe, 2005.

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

[SKV*12] Smith B., Kaufman D. M., Vouga E., Tamstorf R., Grinspun E.: Reflections on simultaneous impact. *ACM Transactions on Graphics 31*, 4 (July 2012), 106:1–106:12.

[Smi00] Smith R.: Open dynamics engine. http://www.ode.org, 2000. Accessed on December 2013.

[SNE10a] Silcowitz M., Niebe S., Erleben K.: A nonsmooth nonlinear conjugate gradient method for interactive contact force problems. *The Visual Computer 26* (2010), 893–901.

[SNE10b] Silcowitz M., Niebe S., Erleben K.: Projected Gauss-Seidel subspace minimization method for interactive rigid body dynamics. In *Proceedings of Computer Graphics Theory and Applications* (2010), INSTICC Press.

[SSF08] Shinar T., Schroeder C., Fedkiw R.: Two-way coupling of rigid and deformable bodies. In *Proceedings of SCA* (2008), pp. 95–103.

[SSIF07] Sifakis E., Shinar T., Irving G., Fedkiw R.: Hybrid simulation of deformable solids. In *Proceedings of SCA* (2007), pp. 81–90.

[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 39* (1996), 2673–2691.

[Stu08] Studer C.: Augmented Time-Stepping Integration of Non-Smooth Dynamical Systems. PhD thesis, ETH Zürich, 2008.

[TBV12] Tonge R., Benevolenski F., Voroshilov A.: Mass splitting for jitter-free parallel rigid body simulation. *ACM Transactions on Graphics 31*, 4 (July 2012), 105:1–105:8.

[TJ07] Twigg C., James D. L.: Many-worlds browsing for control of multibody dynamics. *ACM Transactions on Graphics 26* (2007), 14:1–14:8.

[TJ08] Twigg C., James D. L.: Backward steps in rigid body simulation. *ACM Transactions on Graphics 27* (2008), 25:1–25:10.

[TNA08] Tasora A., Negrut D., Anitescu M.: Large-scale parallel multi-body dynamics with frictional contact on the graphical processing unit. In *Proceedings of Institution of Mechanical Engineering, Part K: Journal of Multi-body Dynamics* (2008), pp. 315–326.

[TNA*10] Tasora A., Negrut D., Anitescu M., Mazhar H., Heyn T. D.: Simulation of massive multibody systems using GPU parallel computation. In *Proceedings of WSCG* (2010).

[TP97] Trinkle J., Pang J.: Dynamic multi-rigid-body systems with concurrent distributed contacts. In *Proceedings of IEEE ICRA* (1997), pp. 2276–2281.

[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.

[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 (2001), 2575–2593.

[WBS*13] Weber D., Bender J., Schnoes M., Stork A., Fellner D.: Efficient GPU data structures and methods to solve sparse linear systems in dynamics applications. *Computer Graphics Forum 32*, 1 (2013), 16–26.

[WTF06] Weinstein R., Teran J., Fedkiw R.: Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE TVCG 12*, 3 (2006), 365–374.

[WW90] Witkin A., Welch W.: Fast animation and control of non-rigid structures. In *Proceedings of SIGGRAPH* (1990), pp. 243–252.