Silesian University of Technology
Faculty of Automatic Control, Electronics and Computer Science
Institute of Informatics


# Physics-Based Animation of Articulated Rigid Body Systems for Virtual Environments

**Jakub Stępień**

Gliwice 2013

# Contents

# Chapter 1

# Objectives and scope

## 1.1 Motivation and objectives

The primary professional field of the author of this dissertation is the software technology behind computer graphics and animation along with its application in video games. One of the ubiquitous elements of contemporary video games is physics-based animation which is nowadays usually provided by dedicated software libraries commonly referred to as *physics engines*.

In game-development community, physics engines are commonly known to be specialized, performance-oriented modules which combine practical implementation of complex theoretical concepts with low-level optimization and a lot of *ad hoc* rules which together constitute an impressive know-how. The author felt the urge to investigate and understand details of actual implementations of these engines by analyzing available open-source libraries and digesting various indirect information about closed-source solutions. The flip side of this process was obviously the need to study the theory of vectorial and analytical mechanics[1].

After completing preliminary analysis the author was surprised to learn that despite the over 200-year-old exploits of analytical mechanics the dominating tendency in the computer graphics community (both academic and industrial) is to use simple vectorial description of constrained motion by a redundant set of parameters along with conditions which bind the interdependent quantities to each other rather than describing the system in terms of generalized coordinates whose number can be often reduced so that each of them could be assigned an arbitrary value without producing an invalid configuration. Providing a general rule of such a reduction and thus enforcing all constraints modeled in contempo-

[1] Being forced to become acquainted with theoretical foundations of virtually anything (instead of actually implementing it and seeing what happens) is commonly known to be programmer's worst nightmare.

rary physics engines using reduced coordinates is impossible but it does not really explain the total departure from this approach since the core concept of generalized coordinates naturally includes *hybrid* techniques of eliminating some of the degrees of freedom by reduction and others by appending constraint conditions.

The observation was hardly a discovery: preferring maximal coordinates over the reduced representation seems to be a conscious choice of individual developers and apparently the community as a whole. The problem lies in the fact that in most cases this decision does not seem to be grounded in actual experience or publicly available experimental comparison between these two formulations which would clearly point which of them is a better choice for interactive physics engines. Taking this observation into account the author believes it is justified to claim that the *status quo* is mostly based on the common trend[2] which over the years resulted in establishing an unconfirmed belief that one *should* use maximal coordinates. There is obviously no reason why physics engines should *not* be based on maximal coordinates as long as they produce satisfactory results. Author's intention is not to prove anybody wrong. This dissertation is a humble attempt to verify this trend by discussing and analyzing ways of introducing the reduced-coordinate formulation into a typical physics engine of today while preserving its features such as interactive execution times and flexibility.

## 1.2 Contributions

Contributions of work presented in this thesis include:

- development of *Articulated Islands Algorithm* which extends the *Sequential Impulse* method making it utilize the reduced-coordinate representation of kinematic trees
- development of a method exploiting the *Composite Rigid Body Algorithm* to build the equation of motion of a kinematic tree subject to additional joint, non-penetration and velocity constraints
- discussion of problems characteristic to reduced-coordinate algorithms expressed in link frames when symplectic Euler integration is used along with proposing resolutions
- proposal of a unified constraint description utilizing spatial vector algebra
- rephrasing the *Sequential Impulse* algorithm in terms of spatial vector algebra

[2]This trend was greatly reinforced by development of the hardware which made solving larger systems at interactive rates possible.

- implementation and testing of the discussed algorithms including cross-comparison of the results

## 1.3 Roadmap

Chapter 1 is the current chapter which contains the motivation and thesis of this dissertation.

Chapter 2 introduces the application domains of the methods discussed in this work and describes a high-level architecture of a contemporary interactive physics engine. Moreover, it provides several examples of such engines.

Chapter 3 serves as an introduction of different aspects of rigid body dynamics and its basic mathematical description. Core terms and high-level classifications along with principles of mechanics are provided.

Chapter 4 contains practical interpretations, proposed representations and tools (both mathematical and algorithmic) needed to build numerical simulators capable of generating rigid body systems motion approximating or directly following the principles introduced in the third chapter. It discusses three main approaches to constrained multibody system modeling and simulation, i.e. penalty-based, constraint-based and collision-based methods.

Chapter 5 is a survey of works on constrained multibody system simulation which is focused on the authors whose work has mostly contributed[3] to the evolution from early simulators to a contemporary interactive physics engine. Unilaterally and bilaterally systems are considered in separate sections with the latter introducing the basic notions of the *spatial vector algebra*.

Chapter 6 introduces the representation of the system and its constraints used in the forthcoming chapters: the underlying equation of motion along with methods (test stimuli methods) of building and solving it are provided.

Chapter 7 introduces *Sequential Impulse* - a popular simulation method utilized in contemporary physics engines - and describes how it has been integrated with the system representation described in the sixth chapter. Moreover, results of several test scenarios are presented and discussed.

[3] In the author's opinion.

Chapter 8 introduces the proposed method of modifying the Sequential Impulse method so that it utilizes the reduced-coordinate formulation when needed. Description of the method is followed by a discussion of problems one may encounter and means that should be applied to overcome them. Finally, results of a number of test simulations are presented and discussed.

Appendix A contains additional derivations which would not fit into the main body of this dissertation.

# Chapter 2

# Physics for Computer Graphics and Animation

## 2.1 Physics-based Animation

The obvious objective of researchers and programmers within the computer graphics community is to increase the realism of the generated images. The natural manifestation of this trend is the ongoing commitment to provide better and better rendering techniques which results in the ever increasing quality of lighting, materials and special effects observed in video games. Furthermore, in the case of *interactive* graphics[1] the provided rendering methods must produce the results at interactive rates.

However, if one wishes to keep the overall illusion consistent it is necessary to take the quality and believability of the motion of the rendered objects into account as well. Although theoretically all the animation can be produced manually by animators, it soon becomes intractable as the complexity of the scene grows. The natural alternative is to use physics to model the motion of the rendered objects: such and approach promises realism with minimal effort. The problem is to create software capable of performing simulations of this kind.

The term *physics-based animation* was introduced during the 1987 ACM SIGGRAPH conference in a course organized by Alan H. Barr [33]. Obviously, the laws of physics had been used to model and simulate motion long before but in disciplines hardly related to computer graphics such as robotics and mechanics which shows how interdisciplinary this field is. However, although the general objective and the underlying principles are shared between these distant disciplines, the priorities are different: roboticists and mechanicians demand arbitrarily high

[1] Which is the principal interest of the author and the context in which this dissertation has been conceived and realized.

precision of the simulation, while in the computer graphics community the term precision has been replaced by *visual plausibility* (introduced by Barzel et al. in [9]) which means that the generated animation is only expected to *seem natural* to an average observer. It is obviously not forbidden to generate animations which are results of very accurate simulation but it is extremely rarely possible to combine it with the speed, stability and robustness required in interactive applications.

Over the course of last twenty years physics-based animation has become a well established topic in interactive graphics and each year there are numerous papers considering this subject presented in leading computer graphics conferences and papers. Although subjects such as soft-body, cloth, fluid or gas animation have recently gained a lot of popularity, the case of interacting rigid bodies remains the core subject which is still being actively developed. The task of a researcher in this particular domain is to provide the possibility to simulate as many rigid bodies as possible fast enough to maintain interactivity while minimizing the perceivable errors such as one rigid body penetrating into another or a breaking a permanent connection between them.

## 2.2 Physics engine

Game-development community is a natural receiver of the academic research in the field of interactive graphics which currently includes physics-based animation. As in any innovation-oriented industry, software developers in the game industry analyze the applicability of solutions proposed by the scientists and implement those which they find useful. Since there is a tendency to use the term *engine* when referring to any specialized piece of software[2], it should not be surprising that libraries dedicated to physics-based animation are popularly called *physics engines*. Author believes that the commonness of this name justifies the fact that it will be adopted in this dissertation.

This section introduces high-level description of the constituents of a contemporary physics engine. Author wishes to note that a substantial pat of the presented nomenclature and classification is based on [33].

Contemporary physics engines combine different branches of dynamics to provide physics-based animation for a wide range of objects including soft bodies, materials, hair, ropes and fluids but the core functionality is usually focused on rigid body simulation since it is most commonly used in video games of today. Since this dissertation is entirely devoted to animation of rigid-body systems, we shall restrict the forthcoming discussion to these aspects of physics engines which are important and characteristic for this particular domain.

[2]Examples of software engines include search engines, game engines, database engines and web browser engines.

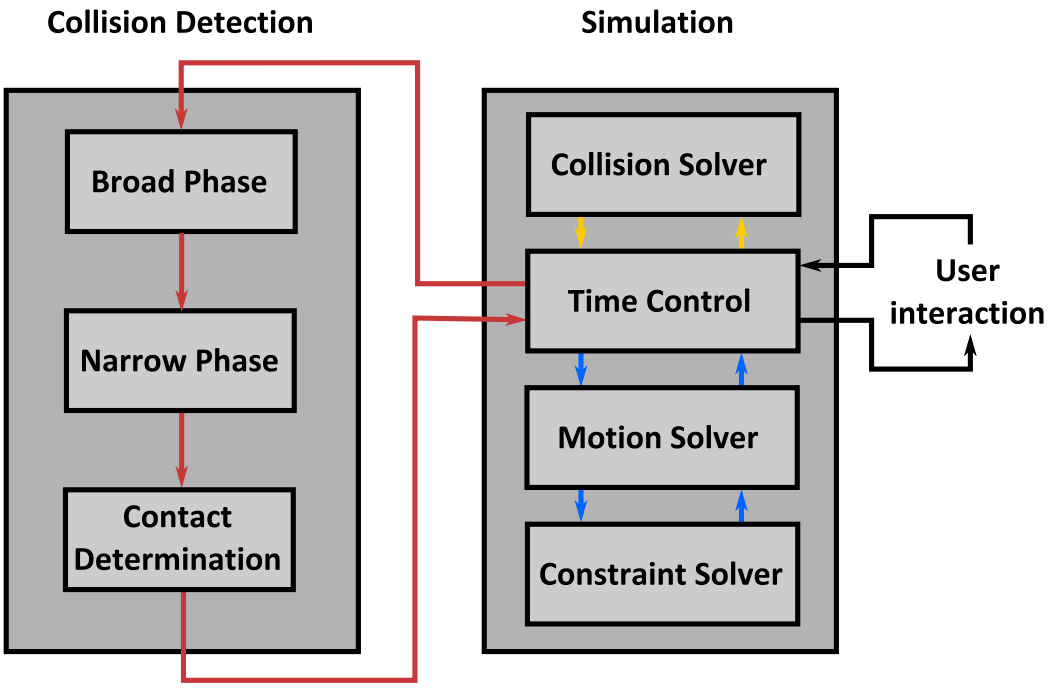



Figure 2.1: A universal architecture of a physics engine according to the modular design described in [33] (note: this figure is a simplified version of Fig. 2.3 in [33].)

### 2.2.1 Anatomy of a physics engine

The actual designs and implementations of physics engines obviously differ so it is hard to discuss them in general, but it is possible to isolate typical or most common elements which can expected to appear in contemporary solutions.

In the coarsest terms, the two main modules that can be easily distinguished are those responsible for collision detection (CD) and simulation. This distinction is fairly natural if one takes into account there are standalone collision detection libraries which can be obviously used in a variety of applications but are most commonly utilized by simulation libraries. Moreover, collision detection and simulation are actually two very different subjects: the former is a purely geometric problem while the latter is concerned with mathematical modeling of motion. The description of the simulated objects reflects this distinction as well since it needs to contain the inertial properties of the object (irrelevant for the collision module) and its shape (irrelevant for the simulator).

#### 2.2.1A Collision detection

The core task of the collision detection module is to supply the information about the existence and properties of *interactions* between the geometric shapes describing the simulated objects. For the CD module to work properly, the

simulation module needs to provide it with the current positions and orientations of the objects.

The traditional and still dominating (in interactive computer graphics) approach is test the scene for intersections at predetermined points in times, most naturally these are the simulation steps. It is called the *discrete collision detection* (DCD). The problem with DCD is that there is no way to avoid objects penetrating into each other and in the extreme may lead to an effect known as *tunneling* when a fast moving object *passes through* another one between consecutive time-steps without any collision being registered. To alleviate this problem different techniques of the so called *continuous collision detection* (CCD) have been proposed. Rather than simply treating the simulated objects as fixed in space in the current instant, CCD analyzes their dynamic state to try to predict its motion instead. However, CCD is obviously more computationally demanding then DCD and thus still less commonly applied in actual implementations.

A naive implementation of the CD module would test each pair of objects for an intersection which implies $O(n^2)$ tests in the case of $n$ objects [87, 33]. To limit the amount of possibly needless computations, the CD process is usually divided into phases:

- broad phase during which bounding boxes or spheres, octrees, hierarchical hash tables and similar methods are applied to discard pairs for which the coarse-grain test is enough to authoritatively state that the objects forming it are *not* intersecting

- narrow phase during which fine-grain (and thus more computationally demanding) tests are performed but only among the pairs that have survived the broad phase

Finally, CD module can provide spatial-temporal coherence analysis which isolates independent contact groups within the overall system and exploits caching which can leverage the efficiency of the CD process itself but it is also becoming more an popular to use the information inside the simulator.

#### 2.2.1B Simulation

The simulation module uses the information about the objects system such as their inertial properties, current configuration and velocity along with the information about the nature of their current interactions to step the system forward in time. Interactions between the objects make their motion inter-dependent, e.g. if the CD reports that two rigid bodies are touching, simulation module needs to restrict their motion so that they do not penetrate into each other when stepping the time forward. If such interactions are present within the system, we call it a *constrained system*, in their absence we say that the system is *unconstrained*.

The heart of the simulation module is the *motion solver* which applies the principles of mechanics to estimate the current accelerations or velocities of the simulated objects. If the scene is represented as an unconstrained system of rigid bodies, the task of the motion solver is pretty simple since the motion of each object is (for the most part) described by equations taught in secondary school. The task becomes much more involved if inter-object interactions come into picture. The general approach to constrain the dynamics of an otherwise unconstrained (or less constrained) multibody system is to express the attributes of motion (velocities or accelerations) we wish to restrict in terms of the chosen coordinates and explicitly state their desired values or acceptable ranges. Then, forces (called reaction or constraint forces) or their impulses need to be found which, when applied, make these attributes attain the desired values: finding these forces is the task of the *constraint solver*. Constraint and motion solvers are often a single module which is then referred to simply as a *solver*[3]. Once the desired accelerations or velocities are determined, motion solver applies a proper numerical integration procedure to update the state of the system.

Collision between rigid bodies is a specific type of interaction because introduces discontinuities in *colliders'* velocities. Therefore, certain simulation paradigms[4] require collisions to be treated separately from other constraints by a *collision solver*.

### 2.2.2 Short review of chosen physics engines

The list of available physics engines is quite long and it is difficult to use or even test all of them - some are presented in the Tab. 2.1. Nevertheless, judging by the popularity and application range, the author believes that the absolutely minimal list of the most famous and often used of these libraries includes Open Dynamics Engine (ODE), Box2D, Bullet, PhysX and Havok.

#### 2.2.2A Open Dynamics Engine

Open Dynamics Engine is one of the earliest open-source physics engines available - its initial release dates back to 2001. Since then it has been used in a number of video games (e.g. *Call of Juarez* series) which is noteworthy since it means it has won the competition with proprietary solutions that dominate the market. ODE is also used by scientific software packages like DANCE or Webots.

ODE is a rigid-body simulator which operates on the force-acceleration level. It offers the user a choice between a precise and iterative solver which enables precision-performance trade-offs.

[3] Actually, the solver terminology is quite informal and different scientists and developers will use these terms differently. Author's intention was to introduce and use nomenclature that is generally acceptable in the community while not being internally conflicted.

[4] Acceleration-force level methods in general.

| Engine | Website | Authors/institution |
|---|---|---|
| Open-source (zlib, BSD, GPL, MIT or unspecified license) | | |
| Bullet | bulletphysics.org | Erwin Coumans et al. |
| Box2D | box2d.org | Erin Catto |
| ODE | ode.org | Russel Smith |
| Tokamak | tokamakphysics.com | David Lam |
| Newton | newtondynamics.com | Julio Jerez and Alain Suero |
| Chipmunk | chipmunk-physics.net | Scott Lembcke |
| Dynamechs | sourceforge.net/projects/dynamechs/ | Scott McMillan |
| IBDS | impulse-based.de | Jan Bender et al. |
| OpenTissue | opentissue.org | Kenny Erleben et al. |
| Moby | physsim.sourceforge.net | Evan Drumwright |
| Closed-source/proprietary | | |
| NVIDIA PhysX | developer.nvidia.com/physx | NVIDIA |
| Havok Physics | havok.com | Havok |
| Vortex | vxsim.com | CM Labs |
| Chrono | chronoengine.info | Alessandro Tasora et al. |

Table 2.1: List of chosen physics engines (based on [95])

#### 2.2.2B Box2D

Box2D offers two-dimensional simulation. It was created by Erin Catto who has been presenting it annually during Game Developers Conference since 2006. Catto is credited for a popular dynamics algorithm called Sequential Impulse used in Box2D (which operates on impulse-velocity level).

Although Box2D is limited to planar simulation (which restricts the application range[5]), Catto uses it as a testing platform and the solutions prototyped in 2D often find their ways to 3D engines.

#### 2.2.2C Bullet

As far as rigid body dynamics is considered, Bullet (originally authored by Erwin Coumans) can be perceived as a 3D version of Box2D since it also uses Catto's Sequential Impulse. However, a different application range has enforced optimization towards simulation of large-scale scenes (exploiting GPUs and multicore CPUs) and high quality collision detection (including CCD). What is more, unlike the previously discussed libraries, Bullet supports soft body simulation.

Via plugins, Bullet is available in 3D modeling tools such as Maya or Blender. It has also been used in a number of recent movies and video games.

[5]Box2D is a popular choice for mobile platforms. A noteworthy exception is Blizzard's famous game franchise: *Diablo*.

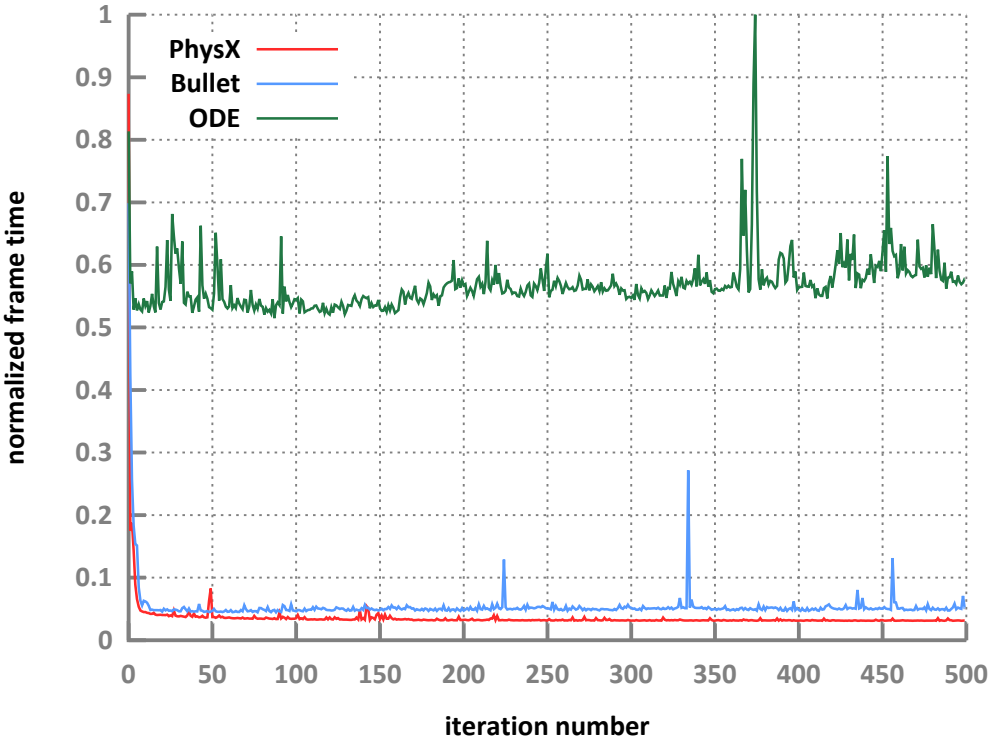



Figure 2.2: Performance-comparison of three popular physics engines (PhysX, Bullet and ODE) performed in [80]: 625 spheres were dropped onto a flat ground and the frame time in consecutive iterations was measured for each of the three engines. We can clearly see ODE being outperformed by the proprietary (closed-source) PhysX engine but it is no longer the case with open-source Bullet (based on Sequential Impulse algorithm) whose execution speed is comparable to PhysX. It is obviously an isolated test-case but shows that the quality offered by proprietary solutions is not completely beyond reach of independent developers.

#### 2.2.2D Closed-source engines

The two best known proprietary physics engines are PhysX and Havok. They have successfully dominated the market and are used in leading game engines and most popular games. Competing with these solutions is obviously a big challenge, especially in terms of feature richness, but the difference between these proprietary packages and community-created open-source solutions is not as great as one could expect (Fig. 2.2).

# Chapter 3

# Simulation of rigid body systems: foundations

## 3.1 Overview

This chapter is devoted to introduction of different aspects of rigid body dynamics and its basic mathematical description. Core terms and different high-level classifications along with principles of mechanics are provided without going into the details of methods or tools (even mathematical) used in simulation.

## 3.2 Non-interacting rigid bodies: projectile motion

A single rigid body in space has six degrees of freedom (DoF) which is equivalent to saying that in order to uniquely describe its configuration (i.e. position and orientation) one needs to provide six parameters: three for translation and three for rotation [78, 112]. What also follows is that the configuration space of a single rigid body is six-dimensional [78, 5]. Sometimes it is more convenient and intuitive to define the degrees of freedom as the number of basic independent motions that are possible to be made.

The dynamics of a single rigid body is provided by the well known Newton-Euler equations; the momentum balance of a single rigid body is given by:

$$\boldsymbol{f} = \frac{d\boldsymbol{l}}{dt} \tag{3.1a}$$

$$\boldsymbol{n} = \frac{d\boldsymbol{h}}{dt}, \tag{3.1b}$$

where $\boldsymbol{l}$ and $\boldsymbol{h}$ are the linear and angular momentum of the rigid body, respectively; $\boldsymbol{f}$ and $\boldsymbol{n}$ are the net force and moment acting on the rigid body,

respectively. It is usually more convenient to use Eq. 3.1 expressed in terms of accelerations rather than momenta, but such a form is dependent on the choice of reference frame. If the equations are expressed about body's center of mass and w.r.t. to an inertial frame, one obtains:

$$\boldsymbol{f} = m\dot{\boldsymbol{v}} \tag{3.2a}$$

$$\boldsymbol{n} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}, \tag{3.2b}$$

where $m$ is the mass, $\boldsymbol{I}$ inertia tensor, $\boldsymbol{v}$ velocity and $\boldsymbol{\omega}$ angular velocity. Eq. 3.2 is sometimes called the *fundamental form* of equations of motion [104].

If we can assume that a multibody system is a collection of rigid bodies that are perfectly independent (i.e. guaranteed not to interact) then we can simply treat each of these bodies as an isolated case. For $n$ bodies, such a system has $6n$ degrees of freedom and configuration space of dimension $6n$. System equations of motion (EOMs) are created by stacking/concatenating the equations for individual bodies:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{f}_x + \boldsymbol{f}_v, \tag{3.3}$$

where $\boldsymbol{M}$ is the system inertia/mass matrix, where $\boldsymbol{q}$ is the vector of generalized coordinates[1], $\boldsymbol{f}_x$ is the vector of external forces acting on individual bodies and $\boldsymbol{f}_v$ represents the velocity-product force terms, i.e. velocity-dependent terms[2] depending on the choice of the reference frame and coordinates.

Equation 3.3 is provided as one, composite system but there is no reason to use the multibody system EOM of in this form directly as long as there is no interaction between the bodies comprising it; each row of this system of equation can be treated interdependently (which has important practical implications when a parallel implementation is considered). The nice feature of multibody systems is lost once interaction between the bodies is considered.

## 3.3 Interaction

If the simulated multibody system is not just a set of permanently independent objects traveling along ballistic trajectories (projectile motion), means must be provided to allow for interactions between these bodies. The commonly distinguished cases of interaction are:

- collisions

[1] It means that $\boldsymbol{q}$ uniquely defines the configuration (i.e. positions and orientations) of bodies in the system. Some authors (e.g. [97]) further restrict the concept of generalized coordinates by requiring their number to be equal to the number of system degrees of freedom (which becomes important once constrained system dynamics are considered). However, we prefer the more general definition as provided by [120, 54, 64].

[2] Terms such as: inertia-variation term appearing as the second summand in the RHS of Eq. 3.2b; Coriolis and centrifugal terms in non-inertial reference frames.

- resting contact
- articulation
- friction

In the most basic terms, the task of the physics engine is to allow for this interactions while preventing inter-penetration.

### 3.3.1 Joints

Joints are permanent connections between bodies which by definition should not be allowed to disconnect. Rephrasing this in simulator terms: forces (called *joint reaction forces*) must be provided that prevent relative motion of the jointed bodies that would lead to violation of conditions imposed by a specific joint. As such, joint permanently reduce the amount of degrees of freedom for the system. Although jointed bodies are not supposed to inter-penetrate, the main consideration when designing joint models is preventing the articulation from falling apart rather than not allowing for the connected links to penetrate into each other. (examples?)

### 3.3.2 Resting contact

Resting contact is the type of interaction occurring when two bodies touch each other for *some time*. Unlike jointed bodies, the ones being in resting contact are allowed to disconnect at any instant, but (as with all the interactions) they are not allowed to penetrate into each other. Thus, the simulator must provide reaction forces that will prevent this from happening. This is a typical non-penetration type of interaction, so the main consideration when modeling it is to prevent overlapping while not restraining separation (i.e. no sticking) of the bodies in contact. Another big issue in this context is handling friction.

### 3.3.3 Friction

The most famous and commonly applied law of friction is the one described back in 1785 [30] by Charles Coulomb. Ubiquity of the Coulomb's law is probably due the simplicity and intuitiveness of its general principle along with the fact that it possesses the quality of being *good-enough* for the vast majority of applications. These features can be misleading though which has been succinctly and vividly summarized by Glocker in [101, Chap. 2]:

> With that friction law, one has chosen one of the most complicated force laws that occur in application problems. It seems to be so easy and so clear at a first view, however, when trying to apply it, or even when just trying to write it down as a mathematical expression, one

immediately encounters a lot of serious and not expected problems of very different nature.

In its basic form, Coulomb's law states that the friction force is bounded by the magnitude of the normal contact force (i.e. the one preventing penetration) times a certain scalar specific to materials of the contacting objects - coefficient of friction. If the contacting bodies are moving with respect to one another tangentially to the contact surface (i.e. sliding) then the friction force acts in the opposite direction of their relative tangential velocity (slip velocity)[3]. If the slip velocity is zero only the magnitude of the friction force is determined by this law. Stated as above, Coulomb's law can be summarized by the following formulae:

$$\begin{cases} \|\boldsymbol{v}_t\| = 0 & \Rightarrow \quad \|\boldsymbol{f}_t\| \leq \mu \|\boldsymbol{f}_n\| \\ \|\boldsymbol{v}_t\| > 0 & \Rightarrow \quad \|\boldsymbol{f}_t\| = \mu \|\boldsymbol{f}_n\|, \quad \boldsymbol{f}_t = -\alpha \boldsymbol{v}_t, \quad \alpha \geq 0, \end{cases} \tag{3.4}$$

where $\mu$ is the coefficient of friction, $\boldsymbol{v}_t$ is the slip velocity, $\boldsymbol{f}_n$ and $\boldsymbol{f}_t$ are the normal and tangential components of the contact force vector, respectively.

In the presence of sliding, friction is described as dynamic or kinetic; otherwise the term static friction (or sticktion) is used. Note that the static scenario includes the case of rolling and these terms are often used synonymously . Static and dynamic cases usually use separate friction coefficients: $\mu_0$ and $\mu$ for static and dynamic case, respectively, where $\mu_0 > \mu$ [5, 50].

The formulae in Eq. 3.4 are stated at the velocity level, but if one wishes to monitor transition from sticking to sliding it is convenient to consider the slip acceleration as well [52, 21]: the transition occurs when $\|\boldsymbol{v}_t\| = 0$ but $\|\dot{\boldsymbol{v}}_t\| \neq 0$. Usually, the direction of friction force in the instant of transition is required to directly oppose the slip acceleration [21, 49, 98], but some authors, e.g. [81, 82, 5, 6] provide a less restrictive version of the law: it is enough if the friction force is partially opposing the tangential acceleration, i.e. $\dot{\boldsymbol{v}}_t \cdot \boldsymbol{f}_t \leq 0$.

### 3.3.3A Set-valuedness

Figure 3.2 depicts the above described dependence between the friction force and slip velocity in one-dimensional case. The first thing to notice is the fact that the illustrated force law is set-valued (at $v_t = 0$ point), i.e. the force can take any value (in the allowed range) when there is no slip velocity.

### 3.3.3B Problematic transitions

Different values of static and dynamic friction coefficients lead to a jump change when the transitions between these two cases occur. It can be avoided if they

[3] The fact that friction forces are allowed to dissipate the kinetic energy due to slip velocity means that they are *not workless* (unlike contact normal and other constraint forces)[5]

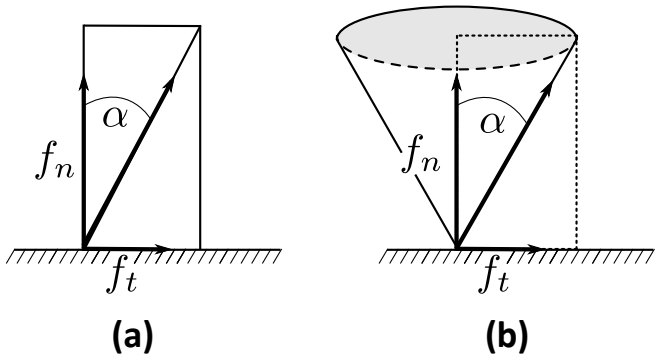



Figure 3.1: (a) Friction angle (2D), (b) Friction cone (3D); the relation with the coefficient of friction is $\mu = tan\,\alpha$

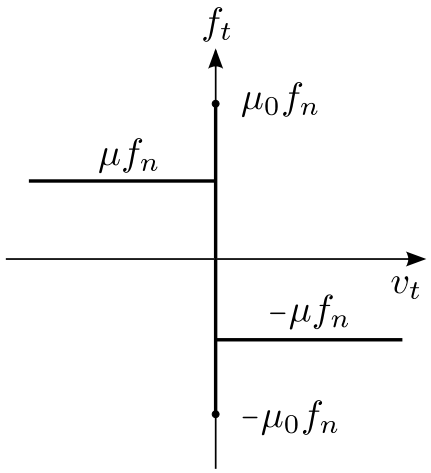



Figure 3.2: Friction force as a function of tangent/slip velocity (planar case)

are assumed to be equal at the instant of transition and more modern theories introduce the relation $\mu = \mu(\|\boldsymbol{v}_t\|)$ thus allowing the coefficient of friction to smoothly depend on the slip velocity (Fig. 3.3) [50, 52, 113], which is experimentally grounded for most material pairings [52].

This approach eliminates another interesting/problematic feature of the model : while it is intuitive to assume that the friction force needs to reach the allowed bound before the stick-slip transition occurs, it is not the case for the opposite transition during which the force can step-change to any value in the allowed interval. This behavior is described as hysteretic by Glocker [50].

### 3.3.3C Non-linearity

More problems arise if we move to three-dimensional scenarios where the friction force is planar. In this case, the allowed contact forces (comprising both normal and friction components) are bounded by a so called *friction cone* (Fig. 3.1b)

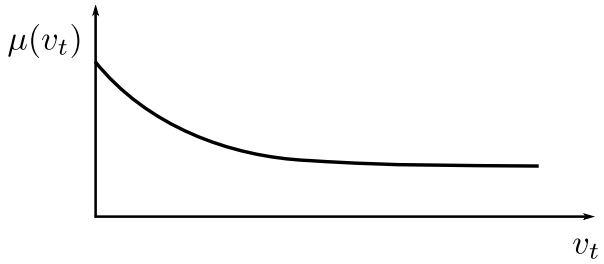



Figure 3.3: Possible dependence of the friction coefficient, $\mu$, on the slip velocity magnitude, $v_t$ [113]

containing the admissible contact force vectors (i.e. sums of normal and frictional contact forces), which is usually given as follows:

$$FC(\boldsymbol{q}) = \{\boldsymbol{f}_t + \boldsymbol{f}_n \mid \|\boldsymbol{f}_t\| \geq \mu\|\boldsymbol{f}_n\|,\ \boldsymbol{f}_t \perp \boldsymbol{f}_n\}, \tag{3.5}$$

where $\boldsymbol{q}$ is the current system configuration , $\boldsymbol{f}_t$ and $\boldsymbol{f}_n$ are the tangential and normal components of the contact force, respectively and $\mu$ is the friction coefficient. Since the relation 3.5 incorporates the magnitudes of the 2D force vectors it is obviously non-linear, which makes handling of friction in 3D systems much more complex than in the planar case. Moreover, anisotropic friction is also possible in 3D scenarios[4] [114].

### 3.3.3D Coupling

While the models describing normal and frictional contact forces are specified separately, the coupling between them is their intrinsic feature - a precise overall contact law cannot treat them separately[5]. Apart from being more difficult to model in general, the coupling can lead to feedback effects between these components so strong that one needs allow for impulsive forces *in the absence of collisions* to provide a solution (Painlevé's paradox) [5, 113].

## 3.3.4 Collisions

In reality, all bodies are deformable [23], but the idealistic assumption of perfect rigidity is very much justified for modeling bodies whose deformations and their influence on the motion of a body as a whole are negligible most of the time. However, the situation is quite different when it comes to collisions for which the fact of disallowing deformations changes the dynamics to an extent which needs special treatment (velocity is no longer a continuous function of time). To

[4]The friction cone is then elliptic as opposed to the circular cone corresponding to isotropic friction [2].

[5]However, many approximate methods are based on the principle of handling these components separately.

alleviate this without scarifying the assumption of rigidity, it is usually extended by a set of rules which *guide* the simulation through collisions [87, 23]. Such extensions are called impact or restitution models or hypotheses.

When two rigid bodies *come into contact* (collide) an instantaneous step-change of their velocities occurs which results in a discontinuity that cannot be expressed using an acceleration-level EoM[6] due to the level of indirection between applied force and resulting velocity alteration. Step-changes in velocity/momentum can be expressed, however, using impulses - time-integrals of forces:

$$p = \int_{t_1}^{t_0} f(t)dt, \tag{3.6}$$

which have the effect of modifying the momentum as follows:

$$p = \int_{t_1}^{t_0} f(t)dt = \int_{t_0}^{t_1} \frac{dh}{dt}dt = h(t_1) - h(t_0). \tag{3.7}$$

This relation is general and simply summarizes the effect of any force acting over a specific period. However, in the case of rigid collisions it allows for escaping the problem of an infinite force active over an infinitesimal time period; it is achieved by representing the effect of such a force by the following finite impulse:

$$p = \lim_{\delta t \to 0} \int_{t_0}^{t_0+\delta t} \frac{f(t)}{\delta t}dt. \tag{3.8}$$

What is also important in the case of rigid body collisions, as the time of application tends to zero, the effect of all finite forces is negligible. Impulses of this specific kind are sometimes called *collision impulses* and this convention shall be adopted in this work to differentiate them from *any* impulse as defined by Equation 3.6. Additionally, the name *instantaneous impulses* shall be used as well to stress the fact of the infinitesimal time of application characterizing them. .

Like in the case resting contact, this is a non-penetration type of interaction: the main consideration when modeling collision is preventing overlapping without sticking and trying to incorporate friction. What is more, restitution law must be chosen and properly modeled which allows to resolve the collision in order to determine post-collision velocities of the colliders.

#### 3.3.4A Restitution laws

Let us first quote Chatterjee & Ruina [25] in order to define a restitution law (which they call a *collision law*)

> Given a pair of colliding rigid bodies (or mechanisms composed of rigid bodies connected with frictionless geometric constraints) in known

[6]The force needed to produce the large acceleration diverges to infinity while the period of its application converges to zero

> configurations and with known pre-collision velocities, and given the relevant parameters (which could be coefficients of friction, restitution, material properties, geometric information like local surface shape, etc.) a collision law predicts the contact impulse.

The quoted work along with [24] by the same authors provide a richly referenced classification of different restitution law, which is however beyond the scope of this dissertation.

As it was mentioned before, when two physical bodies collide, they keep deforming until the normal component of their relative velocity, $v_n$, vanishes and the reaction force reaches its peak value[7]; this is called the compression phase (time interval $[t^-, t^c]$). When the velocity changes sign, the restitution/decompression phase (time interval $(t^c, t^+]$) begins when bodies deform toward their original shapes, but will generally not reach it due to smaller compliance than during compression [117]. It is important to consider energy transformations as well:

- during compression contact force changes the kinetic energy of relative motion into the internal energy of body deformation
- during restitution the elastic strain energy accumulated in the previous phase generates force which drives the bodies apart which restores a fraction of the initial kinetic energy (rebound) [117].

This is obviously a very complex process and the restitution laws provide means to approximate it. The most famous and simplest among these laws is the one due to Newton:

$$v_n(t^+) = -\epsilon v_n(t^-), \tag{3.9}$$

where the scalar value $\epsilon$ is called the coefficient of restitution. As one can see, the division of a collision into phases is not manifested in this formulation: we get a simple mapping from pre- to post-collision velocity.

An alternative proposition known as Poisson's restitution law/hypothesis can be perceived as a slightly more realistic model since it explicitly utilizes the concepts of compression and restitution phases by relating the impulse generated during their courses as follows:

$$p_n(t^+) - p_n(t^c) = \epsilon p_n(t^c), \tag{3.10}$$

where $p_n(t_i)$ represents the impulse of the normal force acting over the time interval $[t^-, t^c]$. Eq. 3.10 states that the normal component of the collision impulse delivered during restitution is $\epsilon$ times what has been delivered during compression. However, the practical difference between Newton's and Poisson's hypotheses is

[7] These events coincide in time [117].

not that big since they are actually equivalent for frictionless collisions and when friction is present they can both cause the total energy of the colliding bodies to increase during a collision [87].

The energetic reformulation of the coefficient of restitution due to Stronge [117] relates the energy released during restitution with the energy that has been stored/absorbed by deformation during compression [124]; expressing it in terms of work done by normal components of contact impulses, $W_n(t)$, one obtains [87]:

$$W_n(t^+) - W_n(t^c) = -\epsilon^2 W_n(t^c). \tag{3.11}$$

Eq. 3.11 ensures that normal contact forces will not add energy to the system, i.e. they are always dissipative (like frictional forces).

#### 3.3.4B Collision with friction

When resolving collisions it is often necessary to take friction into account and the usual choice is the Coulomb model. However, Coulomb law is formulated for the instantaneous values of contact forces and it cannot be simply assumed to hold for impulses which represent the total force acting over the collision timespan and thus a dedicated procedure for analyzing this process is needed.

The over a century old treatment of frictional collisions by Routh is still commonly cited and one often encounters statements that not much has been done in this field since then [89, 124]. Routh used differential equations to describe the collision process [87] and developed a graphical method of analyzing it.

Friction models used in contemporary interactive physics engines rarely go beyond the (simplified) Coulomb law and thus detailed description of frictional collisions is beyond the scope of this dissertation, but the relatively recent works like [25, 24] are a good resource on this and related subjects.

# Chapter 4

# Simulation of rigid body systems: methodology

## 4.1 Overview

The current chapter is an approach at extracting the relevant essence of the rigid body dynamics simulation state of the art without focusing on the works of specific authors who have contributed to it. It presents practical interpretations, proposed representations and tools (both mathematical and algorithmic) needed to build numerical simulators capable of generating rigid body systems motion approximating or directly following the principles introduced in Chap. 3. It discusses three main approaches to constrained multibody system modeling and simulation, i.e. penalty-based, constraint-based and collision-based methods.

## 4.2 Penalty-based methods

This approach provides the simplest means to introduce inter-body interactions into the model. Although particular realizations will vary in details, reaction forces are generally determined using stiff springs (Hooke's law): the greater the configuration error the stronger the correcting/restoring reaction force. The spring is usually combined with a damper/dashpot to make the restoring force penalize velocity and thus reduce oscillations[1]. Due to the simplicity of the underlying principles, ease of implementation and visually plausible animations penalty-based methods were widely used in the past [128, 91, 118, 53, 77, 105, 71] and are sometimes encountered in more recent works as well [88, 59, 130, 32,

[1]Moreover, the damper has the additional effect of eliminating invalid velocity, e.g. if two bodies have already penetrated into each other it is incorrect for their velocities to keep increasing the inter-penetration.

58, 84, 75]. Another advantage of penalty-based methods is that they generate physically valid reaction forces which cannot be always guaranteed by more advanced techniques: a canonical example is a symmetric table positioned on a flat ground[2][5, 87].

Naturally, if penalty-based methods met all the needs of the computer animation community, the research on interactive rigid body simulation could have been abandoned years ago. The following paragraphs briefly discuss the reasons why it did not happen.

#### Configuration errors

Since penalty-based methods generate reaction forces *after* invalid configurations have been detected, the simulation/animation based on them will suffer from from errors by definition. However, it is very often tolerable (to a certain extent) in the computer animation domain[3] and can rarely be eliminated entirely, even if much more complex methods are used. On the other hand, the error must be kept relatively small to maintain the illusion of correctness which requires high spring constants; stiff springs make maintaining simulation stability harder.

#### Tedious tuning

It is very hard to provide a generally applicable and robust way of determining spring constants; they are usually found using ad hoc rules of a thumb which essentially boils down to tedious trial and error:

- if constants are too low the visual artifacts become more and more visible
- if constants are too high the resulting equations can become too stiff and thus hard to handle numerically (integration);

the problem becomes more challenging with the growing number of interactions and can easily become intractable. Moreover, if for any reason the configuration error has grown very high (e.g. discrete collision detection along with a large simulation step), the parameters that are tuned to keep the error small can make the simulation unstable.

#### Oscillation

The fact that a configuration error must occur before any reaction force can be generated, the simulated bodies will tend to oscillate around the proper configuration (see Fig. 4.1) [32, 16]; this problem can be tackled by introducing the

[2]The same acceleration of the chair's center of mass can result from infinitely many distributions of the net ground reaction force on the legs and penalty-based methods produce physically intuitive results which is not guaranteed by the constraint-based methods [87].

[3]This is obviously not a general claim and surely not applicable in other domains, e.g. mechanical engineering.

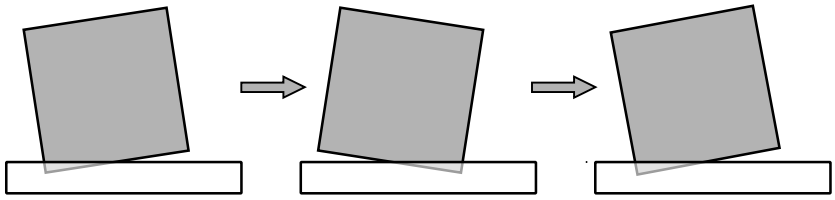


Figure 4.1: The problem of a body oscillating around the proper configuration - inherent problem of penalty-based methods

aforementioned dampers but it is hardly physical and generates even more parameters to tune.

## 4.3 Constraint-based methods

The interaction between the bodies comprising the systems limits their mobility so in general each body can (at least temporarily) have less than 6 DoFs. Those mobility limitations can be expressed in terms of constraint equations/inequalities, algebraic or differential equations/inequalities on position/orientation variables and/or their derivatives, augmenting the basic (i.e. projectile motion/ballistic) ODEs.

The effect of introducing these constraint equations/inequalities is the additional coupling between chosen dimensions of the system configuration space[4] which makes them mutually dependent thus decreasing the number of system DoFs. In terms of system dynamics, this means that specific forces must act on the system to enforce those relations.

### 4.3.1 Classification of constraints

A constraint is described by an algebraic equation in the configuration and/or motion variables of the body or system and can be classified basing on the exact form of these equations. The general classification that is about to be provided is mainly based on [40, 10, 3] and is one of at least two commonly used in the literature. The choice is somewhat arbitrary and definitely subjective. Alternatives are briefly discussed afterwards. .

We will start by discriminating between so called *bilateral* (given by equations) and *unilateral* (given by inequalities)[5] constraints. The former correspond to permanent interaction between bodies which is never allowed to be interrupted; the latter arise when bodies can interact but can also become independent by

[4]To be exact: constraint equations/inequalities which consider one DoF are not uncommon; joint limits are a perfect example: single DoFs can be temporarily locked/disabled whenever a joint is about to leave the admissible region of its configuration space

[5]In fact, bilateral constraints are often called *equality* constraint and unilateral - *inequality* constraints.

separation (as far as the constraint condition is considered). Thus, a body subject only to a unilateral constraint undergoes a free motion in certain portions of its configuration space but becomes constrained when entering others . The name unilateral comes from the fact if bodies are supposed to be allowed to separate constraint force must not prevent it while still allowing for interaction if need be: it must act in *one direction* only. Correspondingly, bilateral constraint generates a reaction force which acts in both directions.

Next we consider *what* gets constrained: if these are certain configurations that we wish to disallow, then the corresponding constraints are called *geometric*; if these are certain types of motion that are prohibited - we need to apply *kinematic* constraints.

It is important to note that when disallowing configurations (by applying geometric constraints), we also eliminate certain types of motion but it does not necessarily work the other way round. In order to put it in mathematical terms, let us introduce geometric constraint function, $\phi_h$:

$$\phi_h(\boldsymbol{q}, t) = 0 \tag{4.1}$$

and let us differentiate it with respect to time to obtain:

$$\frac{d}{dt}\phi_h(\boldsymbol{q}, t) = \frac{\partial \phi_h}{\partial t} + \sum_{i=1}^{n} \frac{\partial \phi_h}{\partial q_i}\dot{q}_i, \tag{4.2}$$

which can be perceived as kinematic/velocity version of Eq. 4.1[6] Now, let us introduce a different constraint function, $\phi_{nh}$, which (unlike $\phi_h$) is linear in the velocity variables before it is differentiated:

$$\phi_{nh}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = \phi^0_{nh} + \sum_{i=1}^{n} \phi^i_{nh}\dot{q}_i, \tag{4.3}$$

where $\phi^i_{nh}$, are functions of time and configuration variables, $\boldsymbol{q}$, only [40]. Equations 4.2 and 4.3 have the same algebraic form but the latter cannot be integrated to resemble the former in Eq. 4.1 because it is simply not a derivative of any function. Constraints which can be both geometric and kinematic are called *holonomic* while those which are only kinematic - *nonholonomic*.

Finally, we shall discuss the dependence on time, or, to be more precise, explicit dependence on time. A constraint is called rheonomic if it is explicitly dependent on time and thus enforces prescribed motions. If there is no such explicit dependence, the constraint is called scleronomic.

Unfortunately, various authors classify constraints differently and it is quite hard to choose the *proper* approach. Very often inequality constraints are treated as non-holonomic (by assumption that constraints that are not bilateral and holonomic are simply non-holonomic) [108, 33, 12, 3]. Furthermore, it is not always

[6] Providing the kinematic version of the constraint equation in respected at all times.

clear whether one should use terms scleronomic and rheonomic in unilateral cases [90, 40]. Certain authors disassemble the classification hierarchy completely and propose to use the classes as a set of independent labels which (presumably) are applicable in any combination [48].

Luckily, this struggle for proper and clear distinction between different constraint types is not our concern. Thus, to avoid the needless confusion introduced by generalizing classifications, we will provide three classes of constraints typically encountered in physics-engines domain and only refer them to the formerly provided formal definitions.

#### 4.3.1A Joints

Joints generate bilateral holonomic constraint because the corresponding reaction forces can both push and pull in order to block the motion which would lead to disallowed configurations. Usually joints yield scleronomic constraints, but it is conceivable to treat them as rheonomic in kinematically-driven motion cases.

#### 4.3.1B Non-penetration

Non-penetration is obviously defined by providing valid and invalid configurations; furthermore, it is enforced by preventing motion in one direction of a chosen dimension since the reaction force can only repel, not attract. These two features mean that non-penetration requires holonomic unilateral constraints. Since time-controlled contact sounds grotesque, one can safely assume that these constraints are also scleronomic.

#### 4.3.1C Joint limits

To maintain generality, joint constraints are usually chosen from a moderate set of canonical models like hinge or ball-and-socket joints which correspond to relatively uncomplex configuration or motion subspaces. Joint limits enable customization of the standard models by explicitly specifying valid ranges of the configurations allowed by the joint. Human knee can serve as an example here (Fig. 4.2).

General properties of joint limits are mostly the same as those characterizing non-penetration constraints.

#### 4.3.1D Velocity constraints

A good example of a velocity constraint is a requirement that wheels of a vehicle rotate with a specific angular velocity; velocity constraint says nothing about the configuration of these wheels. Therefore, when discussing velocity constraints, we need to consider valid and invalid velocities rather than configurations which clearly means we are dealing with nonholonomic constraints. However, it is hard

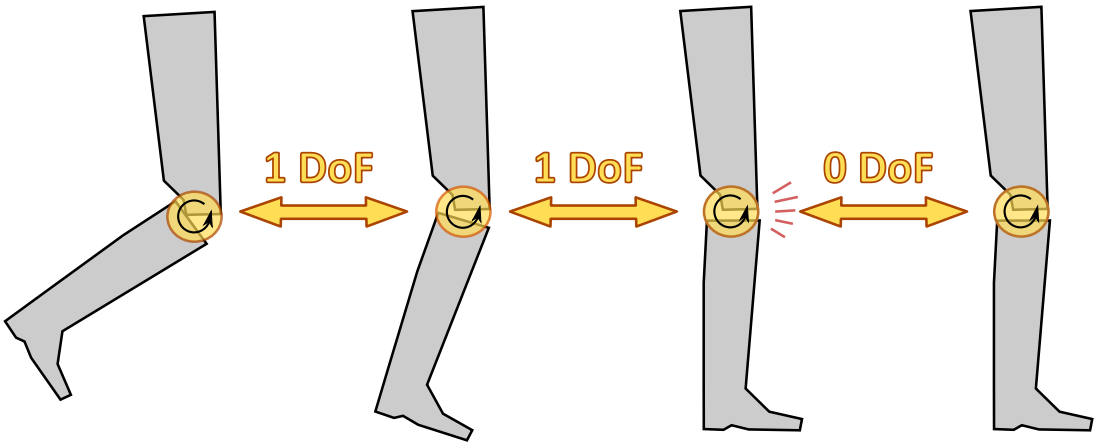



Figure 4.2: Knee modeled by a hinge joint and an additional limit

to characterize this class any further since both unilateral and bilateral velocity constraints seem reasonable and, similarly to joints, in the case of kinematically-driven motion they could well become rheonomic.

### 4.3.2 Constrained system EOMs

The unconstrained rigid body equations of motion presented by Eq. 3.6 will now be modified to (explicitly) contain the vector representing reaction forces of the constraints, $\boldsymbol{f}_c$:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{f}_x + \boldsymbol{f}_v + \boldsymbol{f}_c. \tag{4.4}$$

By the principles of virtual work (or power) [78, 112, 129, 65, 40] we can always express the constraint forces as follows:

$$\boldsymbol{f}_c = \boldsymbol{J}_c^T \boldsymbol{\lambda}_c, \tag{4.5}$$

where $\boldsymbol{J}_c$ is the *Jacobian matrix* of the constraint equation treated as a vector-valued function of the system configuration vector, $\boldsymbol{q}$ . This Jacobian matrix is commonly referred to as a *constraint Jacobian matrix* (or shortly *constraint Jacobian*[7]); if all the constraints in the system are bilateral $\boldsymbol{J}_c$ is given by:

$$\boldsymbol{J}_c = \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial \phi_1}{\partial q_1} & \cdots & \frac{\partial \phi_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \phi_m}{\partial q_1} & \cdots & \frac{\partial \phi_m}{\partial q_n} \end{bmatrix}, \tag{4.6}$$

[7]The nomenclature concerning Jacobian matrices is ambiguous: they are commonly referred to simply as Jacobians (especially among roboticists [31]), but so are their determinants! However, since Jacobian matrix determinants do not appear in this work, we shall adopt the convention of using the terms Jacobian matrix and Jacobian as synonyms.

where $\boldsymbol{\phi}(\boldsymbol{q})$ is a bilateral constraints function. Let rows of $\boldsymbol{J}_c$ be grouped by type and stored in smaller Jacobians: $\boldsymbol{\Gamma}$, $\boldsymbol{Z}$ and $\boldsymbol{\Upsilon}$ for holonomic, non-holonomic and unilateral constraints, respectively; the same holds for $\boldsymbol{\lambda}$. These assumptions yield:

$$\boldsymbol{J}_c = \begin{bmatrix} \boldsymbol{\Gamma} \\ \boldsymbol{Z} \\ \boldsymbol{\Upsilon} \end{bmatrix}, \quad \boldsymbol{\lambda}_c = \begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\zeta} \\ \boldsymbol{\upsilon} \end{bmatrix}. \tag{4.7}$$

The deeper meaning behind the principle of virtual power and Eq. 4.5 is that constraint forces act so as to eliminate any illegal motions without influencing the legal ones. This condition is naturally fulfilled if in the system configuration space constrain forces, no matter their magnitudes, are oriented orthogonally to any valid velocity vector[8].

Getting back to the equation of motion: inspecting the Eq. 4.5 we can say that if the constraint equations are known, so are the directions of the corresponding reaction forces. Therefore, what remains to be determined are *only* their magnitudes, $\boldsymbol{\lambda}$. Now, substituting Eq. 4.5 into Eq. 4.4 one obtains:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{f}_x + \boldsymbol{f}_v + \boldsymbol{J}_c^T \boldsymbol{\lambda}_c, \tag{4.8}$$

which is an undetermined system of $n_{dof}$ equations in $n_{dof}+n_c$ unknowns: system accelerations and the magnitudes of reaction forces. Therefore, in order to fully determine it one needs to supply $n_c$ additional relations, which are specific to constraint types.

### 4.3.3 Joints

In case of joints their constraint equations seem like a natural candidate for making the system in Eq. 4.8 fully determined:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{f}_x + \boldsymbol{f}_v + \boldsymbol{\Gamma}^T \boldsymbol{\gamma}, \tag{4.9a}$$

$$\boldsymbol{\phi}(\boldsymbol{q}) = \boldsymbol{0}, \tag{4.9b}$$

where Eq. 4.9b represents the joint constraints. This choice yields a set of Differential Algebraic Equations (DAEs). DAEs combine differential and algebraic equations and are often viewed as differential equations on manifolds [47]. The formal definition of a DAE (in implicit form) is:

$$F(t, z(t), \dot{z}(t)) = 0, \tag{4.10}$$

where $z$ denotes the state. If $\frac{\partial F}{\partial \dot{z}}$ is non-singular then the above equation could be converted into an ODE [28, 47]. What follows is that explicit and implicit ODEs are special cases of DAEs [100, 47].

[8] It also means that constraint forces are not allowed to do work on the system - they are *workless*.

A DAE in a semi-explicit form can be written as:

$$\dot{x}(t) = f(t, x(t), y(t)) \tag{4.11a}$$

$$0 = g(t, x(t), y(t)), \tag{4.11b}$$

where the state $z$ is now split into components $x$ and $y$ called the differential and algebraic variable, respectively [47]. DAEs in this form are sometimes viewed as ODEs with constraints.

In order to measure the difficulty (in terms of stability and accuracy) in the numerical treatment of different DAEs a notion of an *index* has been introduced which quantifies the degree of regularity of a DAE [47]. DAEs of index 2 or higher are called *higher index* DAEs.

Several different index definitions have been proposed for general systems but they are way beyond the scope of this work. In our particular case, EOMs for a mechanical system subject to holonomic constraints form semi-explicit DAEs of index 3 [65, 21, 69, 19, 16] or, equivalently, Hessenberg DAE of index 3 [47], which means that the various index definitions coincide and thus it is typical to focus on the so called *differential index* only which is defined alternately as:

- the minimum number of differentiations with respect to time of the algebraic equation 4.11b (followed by substitution of 4.11a for $\dot{x}$) needed for the resulting equation to become solvable for $\dot{y}$ [47, 65, 28]
- the number of times the constraint equations need to be differentiated to obtain an ODE (called the *underlying ODE*)[9]

Solving general DAE systems is still a very active research area and the methods proposed so far are from achieving the maturity of corresponding ODE approaches [100, 21, 69, 65]. Numerical methods employed for DAEs belong to the group of either backward-difference (BDF) or implicit Runge-Kutta (IRK) methods but these approaches a far from being general or suited for real-time/interactive purposes [62, 65, 74].

A popular alternative for dealing with higher index DAEs is to reformulate them in order to obtain a more tractable lower index problem [69, 47], which can be achieved by differentiating the constraint equations with respect to time[10]; this method is called an *index reduction* of a DAE. However, although this simplification is *analytically* sound since the lower-index DAE has the same solution as the original one, it is no longer so in the case of a *numerical* solution [60, 47]. This results in accumulation of the error on configuration/position and/or velocity levels in the course of simulation and thus demands applying techniques

[9] Note: since ODEs are actually special cases of DAEs [47] they are sometimes described as index-0 DAEs [47, 62, 65, 18].

[10] Each differentiation of the constraint/algebraic equations reduces the index by one [47].

minimizing/eliminating the visible artifacts [8, 60, 47, 27].

Let us apply the index reduction technique to the EOMs in Eq. 4.9; it requires us to differentiate the constraint equations with respect to time twice, which yields:

$$\boldsymbol{\Gamma}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{\Gamma}}\dot{\boldsymbol{q}} = \mathbf{0} \qquad \Longleftrightarrow \qquad \boldsymbol{\Gamma}\ddot{\boldsymbol{q}} = -\dot{\boldsymbol{\Gamma}}\dot{\boldsymbol{q}}, \tag{4.12}$$

and combine it with Eq. 4.8, to finally obtain index-1 DAE of the form :

$$\begin{bmatrix} \boldsymbol{M} & -\boldsymbol{\Gamma}^T \\ \boldsymbol{\Gamma} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_x + \boldsymbol{f}_v \\ -\dot{\boldsymbol{\Gamma}}\dot{\boldsymbol{q}} \end{bmatrix}. \tag{4.13}$$

This equation is often called the *descriptor form* of the equations of motion [60] and the constraint force magnitudes are usually referred to as Lagrange multipliers thus referring to the well known mathematical optimization method dealing with extremizing a function subject to equality constraints. Another popular convention is to call this form of EOMs *Lagrange equations of the first kind* [17].

The system in Eq. 4.13 can be obviously solved directly, especially if the of constraint Jacobians and inertia matrix introduce a substantial level of sparsity [8] or it is possible to prune out of linearly-dependent rows [40]. Alternatively, one can subtract $\boldsymbol{\Gamma}\boldsymbol{M}^{-1}$ times the first row from the second row to obtain equation:

$$\boldsymbol{\Gamma}\boldsymbol{M}^{-1}\boldsymbol{\Gamma}^T\boldsymbol{\gamma} = -\dot{\boldsymbol{\Gamma}}\dot{\boldsymbol{q}} - \boldsymbol{\Gamma}\boldsymbol{M}^{-1}(\boldsymbol{f}_x + \boldsymbol{f}_v). \tag{4.14}$$

The coefficient matrix $\boldsymbol{\Gamma}\boldsymbol{M}^{-1}\boldsymbol{\Gamma}^T$ is closely related to the inverse of the operational space inertia matrix introduced by [67] but a common convention for its name or symbol does not really seem to exist[11]. For joint constraints, this coefficient matrix is always symmetric positive-definite [40], but not necessarily full-rank in which case least-squares solution is determined. Once the Eq. 4.14 has been solved for the multipliers $\boldsymbol{\gamma}$, we can move back to Eq. 4.13 to compute accelerations.

Another way to avoid DAEs arising due to holonomic constraints is to express the system dynamics in terms of its degrees of freedom; this approach is sometimes called the *embedding technique* [112] and the set of independent parameters used - the *reduced coordinates*[12]. Generally speaking, embedding eliminates the constraint equations and the corresponding reaction forces at the cost of increased complexity and nonlinearity of the equations[13] [112]

[11] To the best of author's knowledge.

[12] Coordinates resulting from treating all bodies as free objects restrained by reaction forces are sometimes called *the maximal coordinates*

[13] The increased complexity/nonlinearity should not be surprising since the same information must be expressed using a smaller set of equations [112]

The first step is to express the dependent coordinates in terms of the independent ones

$$\boldsymbol{q} = \boldsymbol{b}(\boldsymbol{q}^r), \tag{4.15}$$

where $\boldsymbol{q}$ and $\boldsymbol{q}^r$ denote the maximal- and reduced-coordinate configuration vectors, respectively. A single time-derivation yields the following relation between velocities yields:

$$\dot{\boldsymbol{q}} = \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}^r = \boldsymbol{B}\dot{\boldsymbol{q}}^r, \tag{4.16}$$

where $\boldsymbol{B}$ is the *velocity transformation matrix*[112]. The important feature of $\boldsymbol{B}$ is the fact that if $\boldsymbol{f}_c$ denotes the reaction forces of the constraints enforcing the relation in Eq. 4.15, then using the principle of virtual power one can prove that $\boldsymbol{B}^T \boldsymbol{f}_c = \boldsymbol{0}$.

Let us differentiate Eq. 4.15 w.r.t. time once again to obtain acceleration-level relation:

$$\ddot{\boldsymbol{q}} = \boldsymbol{B}\ddot{\boldsymbol{q}}^r + \dot{\boldsymbol{B}}\dot{\boldsymbol{q}}^r, \tag{4.17}$$

substitute it into Eq. 4.4 for $\ddot{\boldsymbol{x}}$:

$$\boldsymbol{M}\boldsymbol{B}\ddot{\boldsymbol{q}}^r = \boldsymbol{f}_x + \boldsymbol{f}_v + \boldsymbol{f}_c - \boldsymbol{M}\dot{\boldsymbol{B}}\dot{\boldsymbol{q}}^r \tag{4.18}$$

and left-multiply it the resulting expression by $\boldsymbol{B}^T$ to eliminate the reaction force vector, $\boldsymbol{f}_c$:

$$\begin{aligned} \boldsymbol{B}^T\boldsymbol{M}\boldsymbol{B}\ddot{\boldsymbol{q}}^r &= \boldsymbol{B}^T\boldsymbol{f}_x + \boldsymbol{B}^T(\boldsymbol{f}_v - \boldsymbol{M}\dot{\boldsymbol{B}}\dot{\boldsymbol{q}}^r) = \\ &= \boldsymbol{M}^r\ddot{\boldsymbol{q}}^r = \boldsymbol{f}_x^r + \boldsymbol{f}_v^r, \end{aligned} \tag{4.19}$$

which is an EOM of the system in terms of its degrees of freedom.

The Lagrange-multiplier-based methods and the embedding technique have their advantages and disadvantages which in many cases complement each other.

#### Lagrange multipliers: pros and cons

The multiplier approach is intuitive, yields relatively simple and thus legible equations; adding/removing constraints is straightforward which facilitates software design. On the other hand, there are two basic problems with this approach:

- reducing system degrees of freedom makes the system to be solved larger; in cases when most DoFs are permanently eliminated this approach may not a good choice performance-wise
- index reduction when faced with the inherently imprecise/approximate nature of the numerical integration process causes the configuration error to accumulate over time resulting in what is popularly known as the drifting problem of multiplier approaches

#### Reduced coordinates: pros and cons

The situation is perfectly opposite in the case of reduced coordinates: they yield more complex equations and they are commonly believed to be more difficult to understand and implement. Furthermore:

- means must be provided to define the proper set of independent variables; while this is quite easily achieved if the underlying topology is a tree, but providing a general solution is very difficult and rarely practical for general purpose simulators
- determination of the coefficients of the EOM will in most cases require more complex algorithms than in the multiplier approaches
- reduced coordinates cannot be used to express non-holonomic constraints (as opposed to multiplier methods) [8]

The advantages of reduced coordinates are:

- they are free from the drifting problem
- more constrained systems yield smaller EOMs: every new constraint decreases the number of DoFs which reduces the dimension of the EOM to be determined and solved

### 4.3.4 Velocity constraints

Introducing this type of constraints into the EOM is done very similarly to the joints' case:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{f}_x + \boldsymbol{f}_v + \boldsymbol{Z}^T\boldsymbol{\zeta}, \tag{4.20}$$

$$\boldsymbol{Z}\dot{\boldsymbol{q}} = \boldsymbol{0}. \tag{4.21}$$

Again, with this approach we end up with a semi-explicit DAE, only this time it is index-2 [69, 62].

### 4.3.5 Non-penetration

If one wishes to introduce non-penetration constraints turning the Eq. 4.8 into a fully determined system requires more consideration. The basic problem here can be summarized by the following relation between the value of a scalar non-penetration constraint function, $\psi(\boldsymbol{q})$[14], and the corresponding normal contact force preventing the penetration (given by its scalar Lagrange multiplier, $\lambda$):

$$\lambda \geq 0, \quad \psi \geq 0, \quad \lambda\psi = 0, \tag{4.22}$$

[14]Dependency on the system configuration vector, $\boldsymbol{q}$, is often dropped for convenience; we shall follow this convention.

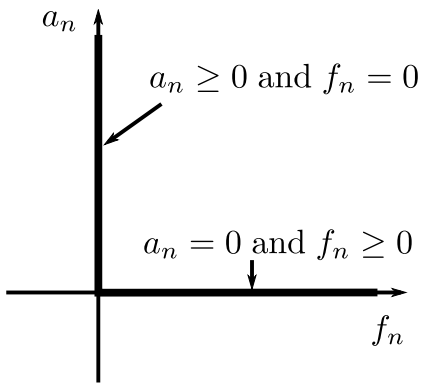



Figure 4.3: Corner law illustrating the complementarity between the normal components of the acceleration, $a_n$, and contact reaction force, $f_n$.

which is often referred to as Signorini's conditions/law or a corner law for normal contacts [20, 102, 21, 103, 127] (Fig. 4.3). The two inequalities in Eq. 4.22 manifest a convention stating that valid (by their physical interpretations, see below) values of both $\lambda$ and $\psi$ are non-negative, which must guaranteed by proper choice of coordinate frames and constraint equation/function. The third relation is known as a *complementarity condition* which requires at least one the two factors, $\lambda$ and $\psi$, to be zero at all times. The intuitive physical interpretation of Eq. 4.22 is that either:

(i) the bodies are in contact and thus $\psi = 0$ and the reaction force is allowed to act ($\lambda \geq 0$), or

(ii) there is no contact which yields $\psi > 0$ and therefore the reaction force must be zero ($\lambda = 0$).

It is thus apparent that the condition Eq. 4.22 can be used as an indicator of the contact transition phases. If the normal relative distance between two formerly non-contacting bodies becomes zero, the contact is said to have become active and an appropriate normal contact force must be provided. If, on the other hand, the constraint force preventing penetration of two formerly contacting bodies becomes negative (change from repulsive to attractive) it means that detachment is occurring and the contact is said to have become passive. [102]

Simply monitoring these indicators is an obvious approach for single-contact scenarios but it is known to become intractable as the number of interdependent contacts grows [102, 21]. As noted by Delassus, this leads to a massive combinatorial problem of testing $2^m$ combinations ($m$ is the number of contacts). Therefore, a non-naive approach is necessary.

In the case of multiple contacts Eq. 4.22 becomes:

$$\boldsymbol{\lambda} \geq \mathbf{0}, \quad \boldsymbol{\psi} \geq \mathbf{0}, \quad \boldsymbol{\lambda}^{\boldsymbol{T}} \boldsymbol{\psi} = 0, \tag{4.23}$$

where the vector inequalities should be understood component-wise. Eq. 4.23 is often phrased briefly as:

$$\mathbf{0} \leq \boldsymbol{\lambda} \perp \boldsymbol{\psi} \geq \mathbf{0}. \tag{4.24}$$

Posing the conditions in Eq. 4.24 is equivalent to defining the set of admissible normal contact forces [44]. If there are only unilateral constraints in the system its dynamics are given by:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{f}_x + \boldsymbol{f}_v + \boldsymbol{\Upsilon}^T \boldsymbol{v}, \tag{4.25a}$$

$$\mathbf{0} \leq \boldsymbol{v} \perp \boldsymbol{\psi} \geq \mathbf{0} \tag{4.25b}$$

Complementarity between the position-level constraint, $\psi$, and corresponding reaction forces is always true but different conditions are also possible and often more convenient for analytic or numerical treatment of multi-contact scenarios [20, 21].

Let us first define a non-penetration constraint at velocity and acceleration levels. The former is obtained by differentiating $\psi$ once with respect to time:

$$\dot{\psi} = \boldsymbol{\Upsilon}\dot{\boldsymbol{q}} \geq 0, \tag{4.26}$$

and the latter by doing it twice:

$$\ddot{\psi} = \boldsymbol{\Upsilon}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{\Upsilon}}\dot{\boldsymbol{q}} \geq 0. \tag{4.27}$$

$\dot{\psi} = \frac{d\psi}{dt}$ and $\ddot{\psi} = \frac{d^2\psi}{dt^2}$ are the relative normal velocity and acceleration corresponding to the non-penetration/contact constraint $\psi$, respectively.

Let us assume that there was a contact between two bodies and it was broken at time $t_0$:

$$\psi(t < t_0) = 0, \quad \psi(t_0) > 0. \tag{4.28}$$

From the definition of a derivative[15] it follows that for a non-zero time interval $[t_0 - \epsilon_v, t_0)$ the normal contact velocity must have been positive [81]. Using the same logic, assuming that $\dot{\psi}(t)$ is continuous and:

$$\psi(t < t_0) = 0, \quad \dot{\psi}(t < t_0 - \epsilon_v) = 0, \quad \psi(t_0) > 0, \quad \dot{\psi}(t_0 - \epsilon_v) > 0, \tag{4.29}$$

one can conclude that for a non-zero time interval $[t_0 - \epsilon_a, t_0 - \epsilon_v)$, $\epsilon_a < \epsilon_v$ the relative normal acceleration must have been positive. In fact, this logic can be extended to derivatives or arbitrarily high order [20, Remark 5.14]. It is therefore possible to state the complementarity condition between force and velocity or acceleration (which is very popular in the literature) but it is important to note that different level complementarity conditions are not exactly equivalent to the original position-level one [20, Remark 5.14].

[15] And because $\boldsymbol{x}(t)$ (and thus $\psi(\boldsymbol{x})$) is obviously continuous

Let us limit this general discussion to acceleration-level complementarity for now (other options will be covered later). For multiple interdependent contacts it can be written as follows:

$$\mathbf{0} \leq \boldsymbol{\upsilon} \perp \ddot{\boldsymbol{\psi}} \geq \mathbf{0}. \tag{4.30}$$

The main advantage of applying complementarity at this level is the fact that it can be directly used to augment the constrained system EOM (Eq. 4.8) in order to obtain a Linear Complementarity Problem (LCP). Indeed, performing the same substitution as in the case of holonomic constraints, i.e. using Eq. 4.8 to eliminate $\ddot{x}$ from Eq. 4.27, and then combining the result with the conditions 4.30 leads to:

$$\begin{cases} \ddot{\boldsymbol{\psi}} = \boldsymbol{\Upsilon}\boldsymbol{M}^{-1}(\boldsymbol{f}_x + \boldsymbol{f}_v) + \boldsymbol{\Upsilon}\boldsymbol{M}^{-1}\boldsymbol{\Upsilon}^T\boldsymbol{\upsilon} + \dot{\boldsymbol{\Upsilon}}\dot{\boldsymbol{q}} \\ \boldsymbol{\upsilon} \geq \mathbf{0}, \quad \ddot{\boldsymbol{\psi}} \geq \mathbf{0}, \quad \boldsymbol{\upsilon}^T\ddot{\boldsymbol{\psi}} = \mathbf{0}, \end{cases} \tag{4.31}$$

which has the structure

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \quad \boldsymbol{y} \geq \mathbf{0}, \quad \boldsymbol{x} \geq \mathbf{0}, \quad \boldsymbol{y}^T\boldsymbol{x} = \mathbf{0}, \tag{4.32}$$

known as an LCP in the standard form [52, 94, 29]. Obviously, the coefficient matrix $\boldsymbol{A}$ has the same meaning as its *holonomic counterpart* in Eq. 4.14

### 4.3.6 Acceleration- or velocity-level

The basis of the traditional modeling techniques is the usage of second order differential equations of motion. Since this view relates forces/moments acting on the system with system accelerations it is sometimes called the *force-acceleration scheme* [21] or *model* [116].

EOM stated at this level provides an instantaneous view on the system: the simulation sets up these equations at each discrete moment in time in order to solve them for system accelerations (and possibly unknown reaction forces). The determined accelerations are then used to advance the simulation in time by means of a numerical integrator [2], which is roughly equivalent to *stretching* the discrete instant over a non-zero time interval.

The obvious problem with acceleration-level approaches is that they cannot really handle collisions - the continuity of velocities is implicit into this formulation/only finite reaction forces can be determined . In order to enable simulator based on this approach to handle collisions a special treatment is needed: whenever a collision event is detected the flow of the simulation is interrupted to handle it using a collision impulse and an impulse-momentum/velocity relation; once all collisions are resolved this way, the simulation is *restarted* with the modified state. This technique, which is sometimes called an *event-driven* approach

to collisions/nonsmooth motion [93, 21], may seem like an *ad hoc* approach but is actually well-based in terms mathematical descriptions of nonsmooth motion (see Sec. A.2). This is sometimes pointed as on of the arguments against using acceleration-level formulations.

The main drawback of acceleration-level approach occurs when modeling contact using LCP. For frictionless contact, an acceleration-force level LCP is known to have a solution (not necessarily unique) [5]. However, in the case of frictional contact, only certain special cases are guaranteed to be solvable; there are well-known configurations (including single-contact scenarios, e.g. Painlevé's paradox) for which accelerations consistent with the contact constraints do not exist [2].

One way to overcome the problem of acceleration-level formulation is to discretize the EOM before solving it for accelerations (instead of solving them as a differential equation and discretizing/numerically integrating then ). In other words, the acceleration determination step is combined with the velocity update which is probably why methods based on this approach are very often called *time-steppers*.

In this formulation, system dynamics are expressed by a difference equation relating impulses (time integrals of forces) and velocity increments rather than a differential equation relating forces and accelerations. Therefore, we are no longer observing the system at a single time instant but rather over the time-interval of integration. Advantages of the velocity-level formulation over the acceleration-level include:

- it solves a much wider class of problems thanks to allowing friction to cause discontinuous velocity jumps in the absence of collisions
- they handle finite and infinite forces in a unified fashion[16]
- velocity-level algorithms are amendable for convergence analysis, which is not the case for acceleration-level alternatives [21]

### 4.3.7 Handling friction

Since the dominating approach in constraint-based methods is to formulate an LCP, the chosen friction law, Coulomb or its approximation in majority of cases, also needs to be tailored in order to fit the formulation. Means used to achieve this vary considerably as different authors allow different levels of approximation and generality. Therefore, in the following section we shall try to provide the main tendencies without going into details of specific realizations.

The first commonly applied adjustment of the Coulomb law is the approximation of the friction cone (Eq. 3.5) with a polyhedral one, which is necessary

[16] There is no direct way to tell whether an impulsive force has acted over the time-step [115].

to avoid non-linearities yielding an NCP rather than LCP. The simplest possible polyhedral cone is a four-facet pyramid; the more facets are introduced, the closer we get to the original cone at the expense of simplicity (i.e. in general, more computations need to be performed).

#### 4.3.7A Acceleration-level

**Dynamic friction**

Let us first assume that all the contacts in the system yield dynamic friction only. Therefore, just as in the frictionless case, the only independent values that need to be determined are the normal components of contact forces since the tangential ones are then uniquely determined by slip velocities (provided by the system state) and coefficients of friction. What is more, the entire problem can still be expressed in the general form of an LCP.

The problem which arises in such a setting is that the coefficient matrix is no longer symmetric which means that the LCP is no longer convex: finding solutions to non-convex LCPs is NP-hard. Furthermore, the existence and uniqueness of reaction forces solving the problem at the given time is not guaranteed.

**Static friction**

Including the static friction requires more preparation. First of all, when working at the acceleration level the case of zero slip velocity requires further specification w.r.t. slip acceleration, $\dot{\boldsymbol{v}}_t$:

- $\dot{\boldsymbol{v}}_t = \mathbf{0}$, i.e. the friction remains static
- $\dot{\boldsymbol{v}}_t \neq \mathbf{0}$, i.e. transition to sliding is occurring

As discussed before (see 3.3.3B), it is commonly (realistically) assumed that the static and dynamic friction coefficients are equal at the instants of stick-slip/slip-stick transitions.

We can illustrate the slip acceleration-friction force relation using the Fig. 4.5 it is apparent that the relation is more complex than the corner law presented before[17] (Fig. 4.3). A common approach to deal with this, is to introduce slack variables in order to decompose the characteristic so that each component is equivalent to the corner law, but their intersection yields the original relation for static friction. Different decompositions are possible, but the simplest one is enough to convey the essence of these approaches:

- friction is isotropic and its cone is approximated by a pyramid (i.e. its basis is a square and the slip accelerations/friction forces are represented by 2D vectors)

[17] It is sometimes called the *double-corner law* [103].

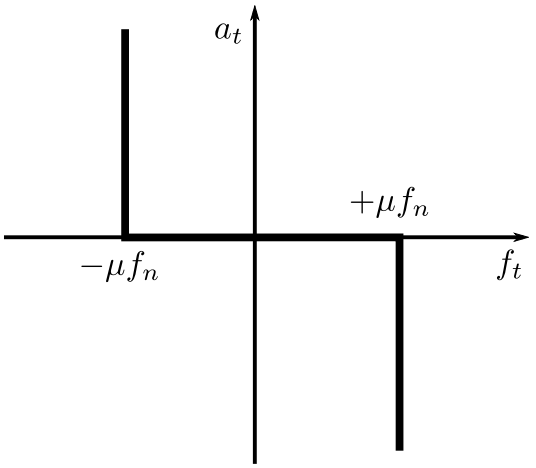



Figure 4.4: Tangential characteristic illustrating the relation between the tangential components of relative acceleration, $a_t$, and and contact force, $f_t$

- each component of $\dot{\boldsymbol{v}}_t$ is decomposed into its non-negative and non-positive parts, $\dot{v}_t^+ \geq 0$ and $\dot{v}_t^- \geq 0$, respectively such that $\dot{v}_t = \dot{v}_t^+ - \dot{v}_t^-$
- for each component of the friction force two slack variables, $s^+ \geq 0$ and $s^- \geq 0$, are introduced such that $s^\pm = \mu f_n \pm f_t$; $s^+$ ($s^-$) is positive as long as $f_t$ is negative (positive) and inside the friction pyramid; when it vanishes sliding can begin[18]

Under the above assumptions, static friction can be stated using the following complementarity relations which should be appended to the normal component conditions:

$$0 \leq s^+ \quad \perp \quad \dot{v}_t^+ \geq 0 \quad \text{and} \quad 0 \leq s^- \quad \perp \quad \dot{v}_t^- \geq 0, \tag{4.33}$$

for each static contact and both of its tangential directions. The decomposition process is illustrated in Fig. 4.5.

Still, the ability to cast the frictional contact problem as an LCP does not change the fact that the class of scenarios solvable at the acceleration-force level is very limited. Let us thus proceed to velocity-level formulations.

#### 4.3.7B Velocity-level

For velocity-level formulations it is typical to resolve friction conditions at the end of the time step, so there is no initial discrimination into static and dynamic friction contacts. On the other hand, there is no slip acceleration to be

[18]These slack variables are sometimes called the *friction saturations* [52] and, colloquially speaking, quantify how much the current friction force is away from its maximal or boundary value. The transition to sliding is possible if they vanish.

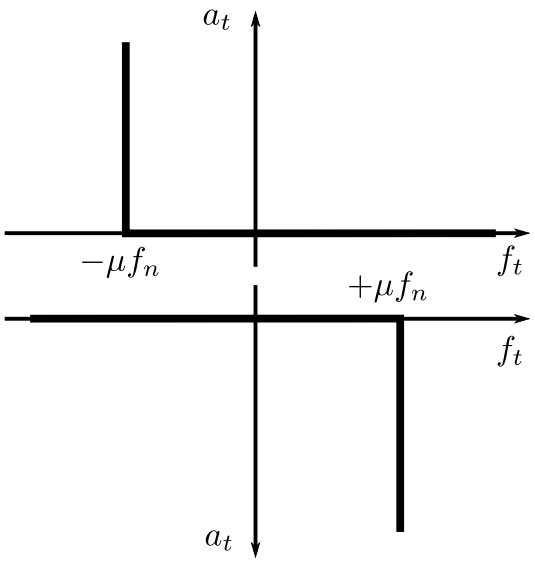



Figure 4.5: One of the possible decompositions of the tangential characteristic in Fig. 4.4: tangential component of the relative acceleration, $a_t$, is split into its non-negative and non-positive parts

tracked. Therefore, although decomposition into complementarity conditions is also applied, its role is a bit different: it must encompass the entire velocity-level Coulomb law. What may be surprising is that if we drop the distinction into dynamic and static friction coefficients (a common simplification [115, 2, 1, 110, 33]), the simple decomposition rule described in the previous section can be directly applied at the velocity level by substituting the slip accelerations with slip velocities.

## 4.4 Collision-based methods

Unlike constraint-based approaches, collision-based methods[19] do not apply explicit algebraic constraints on the configuration of bodies in the simulated system. *All* interactions are handled using *collision impulses*. Between the collisions/interactions bodies act independently following ballistic trajectories.

The need for algebraic constraint conditions on the configurations of interacting bodies is omitted by treating interactions locally, i.e.:

- only a single pair of bodies is analyzed/handled at a time
- no macroscopic assumptions about the nature of the currently analyzed interaction are assumed; the proper macroscopic behavior is supposed to

[19]These methods are most commonly known in the literature as *impulse-based*, which is however (in the author's opinion) quite misleading since impulses by their definition (i.e. time-integrals of forces) are not specific neither to this approach nor to the case of rigid body collisions.

result from proper model/application of impact laws

The above description defines what one might call a *purely* collision-based simulator and is provided in this manner to distinguish it among other approaches (the need which is often overlooked in the literature). As such, however, despite being valid and practically proven approach, has a limited application domain and one needs to relax some of the assumptions to be able handle broader range of systems. Therefore, collision-based simulators are usually equipped with elements typical for the constraint-based methods to make the best of both worlds, but their exact classification is hard since the borders between different paradigms become blurry. We shall call such hybrids collision-constraint-based simulators.

### 4.4.1 Resting contact and joints

The problem that naturally arises with collision-based methods are the ways of treating the resting contact. While in principle it could be resolved by series/-trains of collision impulses, it is computationally inefficient in most implementations[20] whose authors are thus forced to discriminate between the cases of resting and colliding contact and treat them separately.

The situation is even worse with joints: a simulator which expects bodies to be connected by precise joint geometries in order for it to provide articulation based on detecting and resolving collisions is easily conceivable, but chances for making it work in interactive rates for massive multibody systems is not likely to become feasible in the near future.

Collision-constraint-based simulators can treat the system in the collision-free intervals as an articulated structure rather than a set of independent rigid bodies [86, 87] by applying any of the well known algorithms suitable for modeling the continuous motion of such systems. For animating jointed structures this technique is definitely a more tractable approach than purely collision-based methods, but may raise questions on how it differs from acceleration-level simulation restarted after collision is detected to apply impact dynamics (see Sec. 4.3.6). Alternatively, joints and contact can be simulated by applying impulses to eliminate disallowed velocities, which can naturally lead to error accumulation due to drift. Another hybrid technique is to apply collision impulses at the beginning of the time step such that the resultant velocities guarantee no configuration error (within tolerance) at its end [13, 126].

### 4.4.2 Handling friction

As mentioned in section 3.3.4B, Coulomb law is formulated for contact forces and does not simply generalize to impulses. However, many authors dealing with

[20] It becomes apparent when one considers collision-based methods using CCD: for *continuous* contact the time of impact estimates tend to zero bringing the simulator to a halt [87].

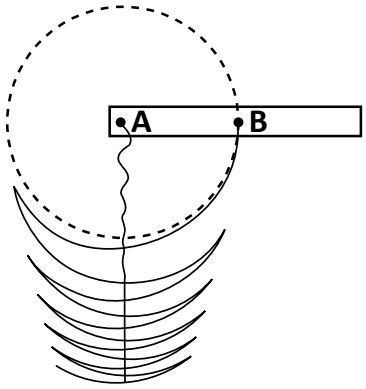



Figure 4.6: An unstabilized simulation of a swinging pendulum [27]; the solid curves illustrate the trajectories of the pivot point, A, and center of mass, B

frictional collisions ignore the fact which is usually enough to provide plausible animations, but can lead to energetic inconsistencies [21].

The *impulsive Coulomb law* along with another simplification - assumption of constant sliding velocity direction across the collision - form the basis for most of the simulators designed for computer graphics with the noteworthy exception of [87], who approached the collision resolution process utilizing methods similar to Routh and applied the Coulomb model for the time-derivatives of impulses (i.e. forces).

## 4.5 Fighting configuration errors

Minor configuration errors are sure to occur no matter which simulation paradigm is chosen, either due to the core properties of the chosen methods or simply because of to the finite precision the computer arithmetic. Therefore, the problem discussed in this section is thus not whether incorrect configurations will occur but whether the chosen simulator realization is capable of eliminating it before it is noticeable without deteriorating the stability or quality of the simulation.

### 4.5.1 Penalty-based methods

Configuration error is an inherent feature of penalty-based methods and the only thing one can do is to keep it as small as possible by properly adjusting spring and damper parameters or decreasing the simulation time-step.

### 4.5.2 Constraint-based methods

The main source of error in the case of popular constraint-based methods is the approximate nature of numerical integration. Most of the interactive constraint-based simulators enforce the geometric/configuration constraints only by ensuring

the velocities/accelerations does not lead to their violation[21] (see Sec. 4.3.3); once those *correct* velocities/accelerations are numerically integrated to update system state, configuration errors are introduced [8, 27]. One can obviously decrease the integration time-interval or choose a higher order integration scheme but firstly: it will never eliminate the error entirely, and secondly: it increases the time of execution making it inapplicable in the interactive simulation realities [27]. The process of tackling constraint configuration errors is commonly called the *constraint stabilization*.

The simplest and most common constraint stabilization technique is the so called *Baumgarte stabilization*: let us assume that we are dealing with a joint constraint and thus its acceleration-level equation is given by Eq. 4.12; if we assume that $\boldsymbol{e}$ denotes the configuration error then Eq. 4.12 is the following differential equation:

$$\ddot{\boldsymbol{e}} = \boldsymbol{0}. \tag{4.34}$$

Baumgarte stabilization replaces the acceleration-level constraint above equation by [40, 112, 65]:

$$\ddot{\boldsymbol{e}} + 2\alpha\dot{\boldsymbol{e}} + \beta^2\boldsymbol{e} = \boldsymbol{0}, \tag{4.35}$$

which combined with by Eq. 4.12 leads to:

$$\boldsymbol{\Gamma}\ddot{\boldsymbol{q}} = -\dot{\boldsymbol{\Gamma}}\dot{\boldsymbol{q}} - 2\alpha\dot{\boldsymbol{e}} - \beta^2\boldsymbol{e} \tag{4.36}$$

and is equivalent to introducing a damped-spring-based penalty force acting so as to eliminate the error term $\boldsymbol{e}$. Therefore, Baumgarte stabilization suffers from the same problems penalty-based methods do. One final note is that the Eq. 4.35 applies to acceleration-level formulations; the velocity-level counterpart uses only the $\beta^2\boldsymbol{e}$ term since the solver itself corrects the velocities.

The problem with the Baumgarte stabilization is the fact that it introduces energy into the system. To avoid correcting the error by modifying the EOM one can project the solution onto the valid manifold in the configuration space after each time step [125, 1, 116]. This technique is sometimes called *post-stabilization* [125, 27]. Let us assume that the configuration level constraint equation is $g(\boldsymbol{q}) = 0$; the idea is to find a smallest possible corrective term, $d\boldsymbol{q}$, such that while $g(\boldsymbol{q}) \neq 0$ (i.e. $\boldsymbol{q}$ has drifted away from the constraint manifold), $g(\boldsymbol{q} + d\boldsymbol{q}) = 0$. This approach relates forces/impulses with configuration increments directly and is thus sometimes compared to a so called *first-order world* which assumes velocities to be driven by forces [7, 33]. Researchers have proposed to determine the corrective terms solving non-linear equations [1], an approximate linear system [7] or an LCP [27].

Finally, one can try to create a solver *aware of* the configuration-level constraint conditions and thus reacts to configuration errors before they manage

[21] Except the constraints introduced by the embedding technique.

to accumulate. This may sound like the return to higher-index DAEs which does not seem a practical approach for interactive simulators but is not necessarily the case. Certain researchers have turned their attention to the concept of *pre-stabilization*: the configuration error for each constraint at the end of the time-step is first predicted and then a correcting impulse is determined. This, however, requires solving a set of nonlinear equations [125, 126] or a less precise linear approximation [15, 14]. See Sections 5.2.7 and 5.3.7 for more details.

Moreover, one of the best-known time-steppers [115] enforces non-penetration at the configuration level which results in a non-linear complementarity problem (See Sec. 5.2.3) which seems to deter the interactive simulation community.

### 4.5.3 Collision-based methods

Purely collision-based simulators are not really designed to handle penetration: the entire idea is to model local interactions very precisely. Therefore, different approaches at continuous collision detection (CCD) are typically applied, which essentially boils down to either predicting the time of impact or *backing-up* the simulation to the approximate time of collision once penetration is detected.

If discrete collision detection is used or in the case of collision-constraint-based hybrids one can obviously apply the same techniques as those described in the previous section.

# Chapter 5

# Literature survey

## 5.1 Overview

Basing on the general introduction of the methodology provided in Chap. 4, several prominent works in the field of rigid body simulations will be summarized and categorized in the current chapter. Note that the surveyed works will be discussed in varying detail since they differ in both their impact and fitness for brief presentation[1].

## 5.2 Unilaterally constrained systems

Frictional contact and collision seem to be the single most difficult interaction types to simulate and have thus attracted a lot of scientific attention among researchers in the fields of mathematics, mechanics, robotics and computer graphics. If, however, one looks upon related publications from the last three decades it turns out that the path to the contemporary interactive simulator can be *interpolated* using the works of less than ten authors.

Formulating contact problems in terms of LCP has gained a lot of popularity in 1980s and 1990s. Casting the constrained system dynamics as an LCP enabled unilateral constraints to be posed directly rather than by bilateral constraints switched whenever attractive contact reaction force was detected [61]. The earliest description of a simulator using constraint-based methods to calculate reaction forces (by quadratic programming) appears to be by Lötstedt [81, 82]. The next prominent contributor in this stream of research is definitely Baraff who after several earlier research and survey papers wrote an easy-to-read paper on modified Dantzig's LCP solver [6] applied to enforce frictional

[1] These are obviously subjective criteria.

non-penetration constraints. Due to the limited range of problems solvable by acceleration-level formulation, certain authors ([115, 2, 110]) have turned their attention to velocity-level constraint-based methods whose origins date back to the works of Moreau [93] and Monteiro-Marquez [85].

In recent years, there has been a noticeable shift towards collision-based approaches and their fusions with velocity-level constraint-based methods [89, 86, 55, 111, 13, 15, 14, 126, 11] which was strengthened by the fact the the general simplicity and robustness of these algorithms resonated with physics engine developers.

### 5.2.1 Lötstedt

The works of Per Lötstedt starting from late 1970's and through early 1980's are often referred to as pioneer in the field of rigid body systems simulation [6, 87], but it is not immediately clear (from these referrals) which aspects of his propositions were actually novel. Nevertheless, it seems that one can safely state that (at the very least) Lötstedt popularized the LCP-formulation of unilaterally constrained rigid body systems, particularly in combination with Coulomb friction and Quadratic Programming (QP).

#### 5.2.1A General

An important note needs to be made before we actually begin: [82] is predominantly focused on planar systems. The fact does not really matter that much when discussing frictionless contacts at certain level of abstraction and thus the section 5.2.1B can be assumed to represent both spatial and planar cases. However, there are major differences when friction enters the picture and Lötstedt did not really provide a description of how his solutions could be generalized to 3D systems with friction[2]. Therefore, the section 5.2.1C is limited to planar cases (i.e. friction in one-dimensional).

Let $n$ denote the number of DoFs of the modeled system; we assume there are no non-holonomic constraints and thus the vector $\boldsymbol{q} \in \mathbb{R}^n$ the configuration vector. Let us further assume that the system in question is additionally subject to $p$ contact constraints, the $i^{th}$ of which is given by the condition $\psi_i(\boldsymbol{q}) \geq 0$, collectively given by $\boldsymbol{\psi}(\boldsymbol{q}) \geq \mathbf{0}$ where $\boldsymbol{\psi}(\boldsymbol{q}) \in \mathbb{R}^p$. Constraint inequalities parameters will be often dropped for easier presentation.

There are four time-dependent index sets defined (containing chosen indices of bodies in the system) which summarize the current interactions within the system; for the $j^{th}$ step:

- all (active) contacts set: $N_j = \{i | \psi_i = 0\}$

[2] Although the problem is acknowledged in the concluding section of this paper.

- static friction contacts set: $S_j = \{i | i \in N_j, \boldsymbol{J}\dot{\boldsymbol{q}} = \boldsymbol{0}\}$
- dynamic friction contacts set: $D_j = \{i | i \in N_j, \boldsymbol{J}\dot{\boldsymbol{q}} \neq \boldsymbol{0}\}$
- all contacts with friction: $F_j = (S_j \cup D_j) \subset N_j$

#### 5.2.1B Contact without friction

Equations of motion are explicitly discretized using linear multistep methods; for the $j^{th}$ step:

$$\boldsymbol{q}_j = \frac{1}{\alpha_0^1} b_j^1 \tag{5.1a}$$

$$\dot{\boldsymbol{q}}_j = \frac{1}{\alpha_0^2}[h\beta_0^2 \boldsymbol{M}^{-1}(\boldsymbol{f}_j + \boldsymbol{J}_j^T \boldsymbol{\lambda}_j) + b_j^2], \tag{5.1b}$$

where $h$ is the step size, $r$ is the number of steps; quantities known from previous steps, stored for convenience in $b_j^*$, are given by:

$$b_j^1 = \sum_{k=1}^{r} \beta_k^1 \dot{\boldsymbol{q}}_{j-k} - \sum_{k=1}^{r} \alpha_k^1 \boldsymbol{q}_{j-k}, \tag{5.2}$$

$$b_j^2 = \sum_{k=1}^{r} \beta_k^2 \boldsymbol{M}^{-1}(\boldsymbol{f}_{j-k} + \boldsymbol{J}_{j-k}\boldsymbol{\lambda}_{j-k}) - \sum_{k=1}^{r} \alpha_k^2 \dot{\boldsymbol{q}}_{j-k}, \tag{5.3}$$

with coefficients $\alpha_k^*$ and $\beta_k^*$ assumed to be known and constant: those corresponding to configuration update are determined by Adams-Bashforth explicit formulae of order one or two; velocity update equation is a first or second order backward difference method, i.e. $\beta_0^2 = 1$, $\beta_k^2 = 0$, for $k = 1, 2, \ldots, r$ [65].

For intervals when the set $N$ remains unchanged (or indices are removed from it[3]), Eq. 5.1b is augmented with force-velocity contact complementarity[4] conditions corresponding to the constraint set determined at the beginning of step $j$. The resulting LCP is rephrased as a QP problem, for the $j^{th}$ step:

$$\text{minimize } \frac{1}{2}\boldsymbol{\lambda}_j^T \boldsymbol{J}_j \boldsymbol{M}^{-1} \boldsymbol{J}_j^T \boldsymbol{\lambda}_j + \boldsymbol{\lambda}_j^T \boldsymbol{J}_j (\frac{b_j^2}{h_j} + \boldsymbol{M}^{-1}\boldsymbol{f}_j) \tag{5.4a}$$

$$\text{subject to } \boldsymbol{\lambda}_j \geq \boldsymbol{0}. \tag{5.4b}$$

The above problem is solved for $\boldsymbol{\lambda}_j$, which is then used to update $\dot{\boldsymbol{q}}_j$ by Eq. 5.1b. Configuration update can be done independently because Eq. 5.1a is explicit, i.e. independent of $\dot{\boldsymbol{q}}_j$. It is important to note that although the underlying EOM has been explicitly discretized, the above QP is actually posed at the acceleration-force level[5].

[3] Lötstedt consciously ignores the resulting discontinuity of the jerk

[4] Differentiated complementarity condition is a natural choice taking into account the discretized EOM: the index is lower and one obtains $\boldsymbol{J}\dot{\boldsymbol{q}}$ - relation linear in velocity. Therefore the problem can be cast as an LCP.

[5] Roughly speaking, the acceleration, $\ddot{\boldsymbol{q}}_j$, is replaced by $\frac{\Delta \dot{\boldsymbol{q}}_j}{h}$, where the term in numerator is evaluated using BDF

As it has been previously discussed, changes in $N$ can give rise to discontinuities and need to be handled explicitly. The proposed method considers discontinuities of the:

- velocities due to external impulse or contact gain (tracked by comparing $\boldsymbol{\psi}_{j-1}$ and $\boldsymbol{\psi}_j$)
- accelerations due discontinuous external or reaction forces (it is not clear how one should track them)

If a discontinuity is detected, the simulation is backed up to the estimated time of the event using inverse linear interpolation (if there are several such events the earliest is chosen) and impulsive equations are used to alter the system state. The problem to be solved is again cast as a QP equivalent to 5.4 only this time it relates impulses and the resultant change in velocities/momenta. The procedure is reset with the modified state so this step won't be passed until the are no more discontinuity event detected along the interval.

Once there are no discontinuities, it is checked whether any of the constraints that were active at the beginning of the time-step/interval have become passive along its course; if so - they are removed from $N_j$ and the processing is finished.

#### 5.2.1C Contact with friction (planar systems)

Lötstedt aimed at providing Coulomb-based friction which would allow for the above described frictionless problem to be a special case of the more general algorithm handling friction.

Since we are discussing planar systems, the friction model given in section 3.3.3 is simplified; Lötstedt used the following:

$$\begin{cases} v_t = 0 & \Rightarrow \quad |f_t| \leq \mu f_n \\ v_t \neq 0 & \Rightarrow \quad |f_t| = \mu |f_n|, \quad f_t v_t \leq 0 \end{cases} \tag{5.5}$$

The vector of normal force multipliers needed is renamed to $\boldsymbol{\lambda}_N$ and an additional vector for tangential forces, $\boldsymbol{\lambda}_F$ is defined. However, since the friction forces for contacts in $D_j$ are fully determined in terms of the corresponding normal force, coefficient of friction and a the sign/direction of slip velocity, they are not included in $\boldsymbol{\lambda}_F$ so contains all normal and those tangential contact force magnitudes which correspond to sticking. The concatenation of $\boldsymbol{\lambda}_N$ and $\boldsymbol{\lambda}_F$ is called $\boldsymbol{\lambda}_1$.

Constraint Jacobian, $\boldsymbol{J}$, is also augmented to contain the tangent directions and then split into two parts: rows corresponding to entries in the extended force vector form $\boldsymbol{J}_1$ and the remaining ones - $\boldsymbol{H}$. Additionally, there is an auxiliary matrix $\boldsymbol{U}$ (containing friction coefficients with proper signs and padded with

zeros)[6] such that we can finally write the rephrased EOM:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} = \boldsymbol{f} + (\boldsymbol{J}_1 + \boldsymbol{H}\boldsymbol{U})^T \boldsymbol{\lambda}_1. \tag{5.6}$$

Accordingly, the Eq. 5.1b becomes, for the $j^{th}$ step:

$$\dot{\boldsymbol{q}}_j = \frac{1}{\alpha_0^2}[h\beta_0^2 \boldsymbol{M}^{-1}(\boldsymbol{f}_j + (\boldsymbol{J}_{1,j} + \boldsymbol{H}\boldsymbol{U}_j)^T \boldsymbol{\lambda}_{1,j}) + b_j^2]. \tag{5.7}$$

Computations for the dynamic and static friction are also split. The former is approximated by simple extrapolation using two previous normal contact force evaluations (assuming they are available[7]):

$$\tilde{\boldsymbol{\lambda}}_{F,j} = \boldsymbol{\lambda}_{F,j-1} + \frac{h_j(\boldsymbol{\lambda}_{F,j-1} - \boldsymbol{\lambda}_{F,j-2})}{h_{j-1}}. \tag{5.8}$$

Once evaluated, dynamic friction forces are included into $b_j^2$, which makes the Eq. 5.7 have the same algebraic form as Eq. 5.1b[8].

What remains is the static friction multipliers and, obviously, normal forces, i.e. $\boldsymbol{\lambda}_1$ need to be determined. Since the aim was to obtain a generalization of the frictionless case (Eq. 5.4), the problem to be solved is posed as follows:

$$\text{minimize } \frac{1}{2}\boldsymbol{\lambda}_{1,j}^T \boldsymbol{J}_{1,j}\boldsymbol{M}^{-1}\boldsymbol{J}_{1,j}^T \boldsymbol{\lambda}_{1,j} + \frac{1}{h_j}\boldsymbol{\lambda}_{1,j}^T \boldsymbol{J}_{1,j}(b_j^2 + h_j \boldsymbol{M}^{-1}\boldsymbol{f}_j) \tag{5.9a}$$

$$\text{subject to } \boldsymbol{\lambda}_{N,j} \geq \boldsymbol{0}, \quad -\boldsymbol{\mu}\tilde{\boldsymbol{\lambda}}_{F,j} \leq \boldsymbol{\lambda}_{F,j} \leq \boldsymbol{\mu}\tilde{\boldsymbol{\lambda}}_{F,j}. \tag{5.9b}$$

It is important to note that the bounds on the static friction forces are specified using the approximate normal forces. Lötstedt claimed that the Kuhn-Tucker conditions satisfied at the optimum of 5.9 are the same as the Coulomb friction conditions given by 5.5 except for the approximate bounds.

Thanks to the above-described algebraic representation and applied approximations, the modifications to the main procedure are fairly small:

- there is an additional acceleration discontinuity event that needs to taken into account: the one accompanying a slip-stick transition
- after discontinuities have been handled, constraints in $S_j$ are checked to see if any of them has undergone a stick-slip transition; if that is the case, index sets are aptly updated

[6] Friction coefficients with proper signs reflect the sliding case of the Coulomb law and padding with zeros makes it possible to perform a multiplication with only those entries of $\boldsymbol{\lambda}_1$ which represent normal contact forces corresponding to sliding contacts (and skip $\boldsymbol{\lambda}_F$ entirely).

[7] Lötstedt provides *ad hoc* rules on how to deal with discontinuities or the first steps in the simulation, but w skip them since they don't seem essential here.

[8] This operation does not seem to have any physical meaning, it is just an algebraic *trick*.

The procedure of handling discontinuities is also an extension of the one for frictionless contact, however, it is important to note that Lötstedt has chosen to use Coulomb law at the impulsive level which may seem natural, but can actually lead to a energetic inconsistencies, e.g. inverting sliding direction due to the collision [21].

### 5.2.2 Baraff

The results of David Baraff's research published from late 1980's through 1990's remains a must-read literature for the rigid-body systems simulation researchers, particularly from the computer graphics community. Baraff, unlike authors of mathematical or mechanical backgrounds, wrote avoiding excessive mathematical abstraction which may easily overwhelm a reader with *less formalized* knowledge of mechanics.

Baraff iteratively provided what may now be called a *typical* acceleration-level simulation handling joints, contact, friction and impact. The complementarity conditions were initially handled as an LP and a heuristic method to escape NP-hardness in frictional contact cases [4], but eventually direct LCP formulation was chosen. LCP is solved using procedures based on Lemke's [5] and Dantzig algorithms [6]. Little attention is paid to impulses, but it is clear that they are handled separately  so Baraff's methods can be classified as event-driven.

#### 5.2.2A Contact without friction

In the case of contact constraints, complementarity condition are expressed between accelerations and forces in order to formulate an LCP, which is solved by a customized Dantzig's algorithm sketched below.

Contacts between bodies are assigned indices. Two disjoint index sets are maintained by the procedure:

- already processed detaching contacts: $D = \{i | a_{N,i} > 0\}$
- already processed non-detaching contacts: $ND = \{i | a_{N,i} = 0\}$,

where $a_{N,i}$ denotes the magnitude of the normal relative acceleration at the $i^{th}$ contact point. For convenience, we define three auxiliary sets:

- all contacts: $C$
- already processed contacts: $P = D \cup ND$
- unprocessed contacts: $NP = C \setminus P$

In each simulation step, we set off with $D$ and $ND$ empty and as long as among the contacts in $NP$ there is at least one, denoted with index $d$, such that $a_{N,d} < 0$, it is processed as follows:

- **step 1**: create system comprising the contacts in $P$ plus $d$ and for every non-detaching contact $i$ determine the additional force, $\Delta f_{N,i}$, needed to keep the corresponding $a_{N,i}$ equal to zero (i.e.$\Delta a_{N,i} = 0$) assuming for every detaching contact $j$, $f_{N,j}$ remains zero (i.e. $\Delta f_{N,j} = 0$) and $\Delta f_{N,d}$ is set to some known value[9]; this requires solving a system of linear equations (SOLE) of dimension equal to $|ND|$, i.e. cardinality of the non-detaching set)

- **step 2**: find a tentative scalar multiplier $s$ for the determined $\Delta \boldsymbol{f}$ which ensures that $a_{N,d}$ is driven to zero; that value of $s$ is accepted and this step is over if $s$ *does not* cause penetrating acceleration or attracting force at any contact in $P$; if it *does*, $s$ is scaled down so that its value is as high as possible, but the non-negativity conditions are satisfied for contacts in $P$ and one of the following holds for the $i^{th}$ of them[10]:

  - if $i \in ND$ then $f_{N,i} = 0$ and $i$ is moved to $D$
  - if $i \in D$ then $a_{N,i} = 0$ and $i$ is moved to $ND$

- **step 3**: update force and acceleration and go back do step 1 unless $a_{N,d}$ has been driven zero

- **step 4**: add index $d$ to the set $ND$

The above description of the second step has been composed so as to precisely provide all the information in least formal terms, but since it may not be easy to follow, we additionally quote Baraff himself whose brief description is likely to make it easier to comprehend:

> For each member $j$ in an index set, we compute the minimum step size $s$ that causes $j$ to need to change to another set. For the driving index $d$, we compute the step size that causes us to reach $a_{N,d} = 0$ [...]. The minimum step size that can be taken, along with the constraint $j$ responsible for that limit, is returned.

Baraff has managed to prove that this procedure will always yield a solution .

[9] Baraff proposes $\Delta f_{N,d} = 1$. This just a way to establish the linear relation between $\Delta f_d$ and $\Delta a_d$, which is later used to find the proper scaling factor, $s$; it is reminiscent of *test methods* known in the literature [87, 79, 43]

[10] Baraff's procedure *maxstep* considers only one transfer between index sets per call; we claim that such a situation allows for contacts to temporarily remain in a set non-corresponding to the their state (the case when $s' = s$ in *maxstep*). This seems incorrect but Baraff may have ignored the option since closer analysis reveals that it wouldn't really influence the results.

#### 5.2.2B Contact with friction

Baraff shows that the frictionless procedure is fairly easily generalized to handle static friction, but he failed to prove that a solution can be always determined in this fashion.

Baraff provided procedures handling friction for both planar and 3D systems, but the former ones can be described with far less caveats, while the changes needed to obtain the latter are not substantial for understanding the underlying principle.

Three additional index sets are defined for contacts with friction:

- already processed non-accelerating contacts: $NA = \{i | a_{T,i} = 0\}$
- already processed contacts with positive friction force: $A^+ = \{i | a_{T,i} < 0\}$
- already processed contacts with negative friction force: $A^- = \{i | a_{T,i} > 0\}$,

where $a_{T,i}$ denotes the magnitude of the tangential relative acceleration at the $i^{th}$ contact point. Naturally, the set of all the already processed contacts, $P$, is redefined to include the new sets.

In each simulation step, we set off with $D$, $ND$, $A^+$, $A^-$ and $NA$ empty and as long as among the contacts in $NP$ there is at least one, denoted with index $d$, such that $a_{N,d} < 0$, it is processed as follows :

- **step $1_n$**: step 1 in frictionless case extended to handle the new sets by requiring that $a_{F,i}$ remains zero (i.e. $\Delta a_{F,i} = 0$) while assuming that $f_{F,i}$ for $i \in A^{\mp}$ is updated to remain maximal following the changes in the corresponding normal force (i.e. $\Delta f_{F,i} = \pm\mu\Delta f_{N,i}$)[11]
- **step $2_n$**: step 2 in frictionless case extended to assure that multiplier $s$ does not make the friction force exceed its bounds (set $NA$) or tangential acceleration change its sign (sets $A^{\pm}$) by scaling it down so that for the $i^{th}$ contact one of the following holds:
  - if $i \in A^{\pm}$ then $a_{F,i} = 0$ and $i$ is moved to $NA$
  - if $i \in NA$ then $f_{F,i} = \pm\mu f_{N,i}$ and $i$ is moved to $A^{\pm}$
- **step $3_n$** step 3 in frictionless case but it is looped-back to step $1_n$
- **step $4_n$** step 4 in frictionless case
- **step $1_f$**: step $1_n$

[11] Baraff calls the requirement $\Delta f_{F,i} = \pm\mu\Delta f_{N,i}$ for $i \in A^{\mp}$ a *side condition* and, although he is not really explicit on how it should be enforced, it seems natural that the each such relation is used to eliminate one equation from the system. Thus, since as before the normal forces in the detaching set are assumed zero and the new force, $f_{*,d}$, is set to a known value, one ends up with a SOLE of dimension equal to $|ND \cup NA|$

- **step $\mathbf{2_f}$**: step $2_n$ but the driving index $d$ is treated differently; multiplier $s_1$ is evaluated which makes $a_{F,d}$ reach zero and another one, $s_2$, which makes $f_{F,d}$ reach its bound; the tentative value of $s$ is the smaller of $s_1$ and $s_2$

- **step $\mathbf{3_f}$**: step $3_n$ but it is looped-back to step $1_f$ and the stop condition is different - either $a_{F,d}$ has reached zero or $f_{F,d}$ its bound, $\pm\mu f_{N,d}$

- **step $\mathbf{4_f}$**: add index $d$ either to $A^{\pm}$ or $NA$ depending on the stop condition in step $3_f$

#### Chosen aspects of the 3D case

For 3D cases, Baraff uses the less restrictive version of Coulomb's law for the static friction in 3D, i.e. the relation between the direction of the friction of force and slip acceleration (whenever it is non-zero) is only given by their dot product which is required to be non-positive (see Sec. 3.3.3).

The troublesome fact of the non-linearity of $\|f_{F,i}\| \leq \mu f_{N,i}$ is not really an issue for Baraff's procedures since it can be easily dealt with in step $2_f$.

The direction of the friction force is preserved as long as the contact is in the *accelerating* set. When it is pivoted to $NA$ the force is allowed to act in any direction.

#### Briefly on dynamic friction

Dynamic friction contacts spoil the symmetricity of matrix $\boldsymbol{A}$ , which may result in a situation when driving the normal or friction force up to infinity will not be enough to zero the corresponding acceleration or cause pivoting between index sets. In the given procedures, such a scenario results in the multiplier $s$ being infinite which means that an impulsive force must be applied

### 5.2.3 Stewart & Trinkle

Stewart's and Trinkle's paper [115] (and its shortened yet refreshed version [116]) is among the most often cited texts in the rigid body simulation literature. The proposed time-stepping algorithm (colloquially referred to as the *Stewart-Trinkle time-stepper*) has become the basis of many later propositions and implementations.

As described in the chapter on methodology (4), time-stepping algorithms analyze the system by observing its behavior along the entire time-step, which is naturally realized by discretized EOMs where velocity changes along the step are related to the integrals of applied forces. Collisions are implicitly covered by this approach, but it needs to be noted that Stewart-Trinkle time-stepper in its original form treats all of them as plastic/inelastic.

#### 5.2.3A General

The state of the system is given by its configuration, i.e. $n$ generalized coordinates $\boldsymbol{q}(t) \in \mathbb{R}^n$, and its velocities, $\boldsymbol{v}(t) \in \mathbb{R}^n$. System dynamics are given by:

$$\boldsymbol{M}(\boldsymbol{q})\frac{d\boldsymbol{v}}{dt} = \boldsymbol{k}(\boldsymbol{q}, \boldsymbol{v}), \tag{5.10}$$

where matrix $\boldsymbol{M}(\boldsymbol{q})$ describes the inertial properties of the bodies comprising the system; Coriolis, centrifugal and external force terms are contained in the vector $\boldsymbol{k}(\boldsymbol{q}, \boldsymbol{v}) \in \mathbb{R}^n$. EOM 5.10 is discretized using explicit Euler scheme, which yields, for the $l + 1^{st}$ step:

$$\boldsymbol{M}(\boldsymbol{q}^l + h\boldsymbol{v}^l)(\boldsymbol{v}^{l+1} - \boldsymbol{v}^l) = h\boldsymbol{k}(\boldsymbol{q}^l + h\boldsymbol{v}^l/2, \boldsymbol{v}^l) \tag{5.11a}$$

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^l = \boldsymbol{v}^{l+1}, \tag{5.11b}$$

where $h$ is the integration step size; the aim is to find $\boldsymbol{q}^{l+1}$ and $\boldsymbol{v}^{l+1}$. Including 5.11b in the above may not seem inevitable in this from of Eq. 5.11, but it becomes necessary to clearly define $\boldsymbol{q}^{l+1}$ when contact constraints are introduced.

Note that simple predictions $\boldsymbol{q}^l + h\boldsymbol{v}^l$ and $\boldsymbol{q}^l + h\boldsymbol{v}^l/2$ are used for evaluating $\boldsymbol{M}$ and $\boldsymbol{k}$, respectively, to preserve linearity of the resulting complementarity problem.

#### 5.2.3B Contact without friction

There are $p$ contacts which restrict the admissible region of configuration space, $C$, and each is introduced by a non-penetration constraint of the general form:

$$C = \{\boldsymbol{q} | f(\boldsymbol{q}) \geq 0\} \subset \mathbb{R}^n. \tag{5.12}$$

The boundary of $C$ is denoted by $\partial C$. However, it is assumed that in neighborhoods of configurations corresponding to a contact situation $\partial C$ is well approximates by a half-space and thus the constraint 5.12 becomes:

$$\boldsymbol{n}^T\boldsymbol{q} \geq \alpha_0, \quad \boldsymbol{n} = \nabla f(\boldsymbol{q}), \tag{5.13}$$

which is linear and thus suitable for LCP.

Naturally, the EOM 5.11 needs to be extended to include the contact reaction forces and complementarity conditions. The characteristic feature of the Stewart-Trinkle time-stepper is that it states the complementarity directly between the configuration-level constraint 5.12 or 5.13 and the reaction force/impulse. What is also very important is that the constraint condition is evaluated *at the end of the time step* (i.e. for $\boldsymbol{q}^{l+1}$), which ensures that the determined solution produces no penetration.

Multiple contacts are introduced by means of simple concatenation so, in order to avoid introduction of additional symbols, the complementarity conditions are provided for the $j^{th}$ contact but are meant for all $p$ of them. For the $l+1^{st}$ step:

$$\boldsymbol{M}(\boldsymbol{q}^l + h\boldsymbol{v}^l)(\boldsymbol{v}^{l+1} - \boldsymbol{v}^l) = \sum_{j=1}^{p} \boldsymbol{n}^{(j)} c_n^{(j)} + h\boldsymbol{k}(\boldsymbol{q}^l + h\boldsymbol{v}^l/2, \boldsymbol{v}^l) \tag{5.14a}$$

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^l = \boldsymbol{v}^{l+1}, \tag{5.14b}$$

$$\boldsymbol{n}^{(j)T}\boldsymbol{q}^{l+1} \geq \alpha_0^{(j)} \quad \perp \quad c_n^{(j)} \geq 0, \tag{5.14c}$$

where $c_n^j$ is the impulse summarizing the influence of the normal reaction force associated with contact $j$ along the time step. It should now be clear that 5.14b is needed to evaluate 5.14c at the end of the time step.

For simpler notation, let us temporarily assume a single contact scenario : Eq. 5.14 can be expressed in the following form (functional dependencies dropped ):

$$\begin{bmatrix} \boldsymbol{M} & -\boldsymbol{n} \\ \boldsymbol{n}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^{l+1} \\ c_n \end{bmatrix} + \begin{bmatrix} -\boldsymbol{M}\boldsymbol{v}^l - h\boldsymbol{k} \\ (\boldsymbol{n}^T\boldsymbol{q}^l - \alpha_0)/h \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}, \tag{5.15a}$$

$$r_1 = \boldsymbol{0}, \tag{5.15b}$$

$$0 \leq r_2 \quad \perp \quad c_n \geq 0, \tag{5.15c}$$

where $r_1$ and $r_2$ denote the first and second block-row of the system 5.15a, respectively. Clearly, the Eq. 5.15 represents a Mixed Linear Complementarity Problem (MLCP) [29].One may solve $r_1$ for $\boldsymbol{v}^{l+1}$ in order to eliminate it in the remaining conditions[12] in order to convert the MLCP into a standard LCP [29]:

$$\boldsymbol{n}^T\boldsymbol{M}^{-1}\boldsymbol{n}c_n + (\boldsymbol{n}^T\boldsymbol{q}^l - \alpha_0)/h + \boldsymbol{n}^T(\boldsymbol{v}^l + \boldsymbol{M}^{-1}h\boldsymbol{k}) = r, \tag{5.16a}$$

$$0 \leq r \quad \perp \quad c_n \geq 0, \tag{5.16b}$$

where $r$ denotes the left-hand side of 5.16a. The equational part of the MLCP corresponding to $\boldsymbol{v}^{l+1}$ has been eliminated and we have obtained an LCP[13].

### 5.2.3C Contact with friction

To keep the LCP tools applicable, the friction cone needs to be linearized and the conditions imposed by the Coulomb model need to be tailored to fit the complementary formulation.

[12] Or, equivalently, subtract $\boldsymbol{n}^T\boldsymbol{M}^{-1}$ times $r_1$ from $r_2$.

[13] Schur complement of the coefficient matrix with respect to $\boldsymbol{M}$, i.e. $\boldsymbol{0} - \boldsymbol{n}^T\boldsymbol{M}^{-1}(-\boldsymbol{n}) = \boldsymbol{n}^T\boldsymbol{M}^{-1}\boldsymbol{n}$, naturally arises in this conversion and some authors name the resulting LCP the *Schur complement system* of the original MLCP [115].

In order to achieve these goals Stewart and Trinkle use a polyhedral cone with $k$ facets at each contact:

$$\widehat{FC}(\boldsymbol{q}) = \{c_n\boldsymbol{n} + \boldsymbol{D}\boldsymbol{\beta} | c_n \geq 0, \boldsymbol{\beta} \geq \boldsymbol{0}, \boldsymbol{e}^T\boldsymbol{\beta} \leq \mu c_n\}, \tag{5.17}$$

where $\mu$ is the coefficient of friction; $\boldsymbol{e} \in \mathbb{R}^k$ is a vector of ones, $\boldsymbol{D}$ is a $n \times k$ matrix whose columns are direction vectors[14] defining the polygonal base, $\boldsymbol{d}_i,\ i = 1, ..., k$, which span a plane tangent to $\partial C$ at the configuration $\boldsymbol{q}$; $\boldsymbol{\beta}$ is tangent subspace vector corresponding to the friction force. Columns of $\boldsymbol{D}$ are defined so that for every $i$ there is a $j$ such that $\boldsymbol{d}_i = -\boldsymbol{d}_j$.

Since there are many contacts in general, the individual cones are simply summed; the corresponding vectors are, as before, concatenated[15].

In order to handle friction, the EOM in Eq. 5.14 are adjusted by augmenting 5.14a to include the tangent forces and there are two new complementarity conditions added, thus for the $l + 1^{st}$ step (the complementarity conditions are again provided for the $j^{th}$ contact but are meant for all $p$ of them):

$$\boldsymbol{M}(\boldsymbol{q}^l + h\boldsymbol{v}^l)(\boldsymbol{v}^{l+1} - \boldsymbol{v}^l) = \sum_{j=1}^{p}(\boldsymbol{n}^{(j)}c_n^{(j)} + \boldsymbol{D}^{(j)}\boldsymbol{\beta}^{(j)}) + h\boldsymbol{k}(\boldsymbol{q}^l + h\boldsymbol{v}^l/2, \boldsymbol{v}^l) \tag{5.18a}$$

$$\boldsymbol{q}^{l+1} - \boldsymbol{q}^l = \boldsymbol{v}^{l+1}, \tag{5.18b}$$

$$\boldsymbol{n}^{(j)T}\boldsymbol{q}^{l+1} \geq \alpha_0^{(j)} \quad \perp \quad c_n^{(j)} \geq 0, \tag{5.18c}$$

$$\lambda^{(j)}\boldsymbol{e}^{(j)} + \boldsymbol{D}^{(j)T}\boldsymbol{v}^{l+1} \geq \boldsymbol{0} \quad \perp \quad \boldsymbol{\beta}^{(j)} \geq \boldsymbol{0}, \tag{5.18d}$$

$$\mu^{(j)}c_n^{(j)} - \boldsymbol{e}^{(jT)}\boldsymbol{\beta}^{(j)} \geq 0 \quad \perp \quad \lambda^{(j)} \geq 0, \tag{5.18e}$$

where the vector condition 5.18d is understood row-wise .

The complementarity conditions 5.18c - 5.18e map to the possible contact situations *at the end of the time-step* as follows:

- **detachment**: if there is no contact, then by 5.18c $c_n = 0$ and thus by 5.18e $\boldsymbol{\beta}$ vanishes as well; since there is no contact, $\boldsymbol{D}^T\boldsymbol{v}^{l+1}$ is also zero and so $\lambda$ is left to take any positive value, but has no physically meaningful interpretation

- **sliding**: sliding occurs if the velocity $\boldsymbol{v}^{l+1}$ has a component in $range(\boldsymbol{D})$; due to the way direction vectors, $\boldsymbol{d}_i$, are defined, there must be at least one of them for which $\boldsymbol{d}_i^T\boldsymbol{v}^{l+1}$ is negative which by 5.18d means that $\lambda$ is positive (it must be at least as high as the *most negative* entry of $\boldsymbol{D}^T\boldsymbol{v}^{l+1}$); this in turn means that 5.18e becomes $\mu c_n = \boldsymbol{e}^T\boldsymbol{\beta}$ which clearly corresponds to the case of dynamic friction

[14] They are not necessarily of unit length (due to generalized coordinates or in anisotropic friction scenarios)

[15] There is no certainty whether such a formulation satisfies Coulomb model as the step size tends to zero [21].

- **sticking/rolling**: if the contact exists but there is no sliding $\boldsymbol{v}^{l+1}$ has no component in $range(\boldsymbol{D})$, in which case 5.18d becomes $\lambda \geq 0$
  - if $\lambda = 0$ then we are left with $\mu c_n \geq \boldsymbol{e}^T\boldsymbol{\beta}$ for non-negative values of $\boldsymbol{\beta}$
  - if $\lambda > 0$ then $\boldsymbol{\beta} = \boldsymbol{0}$ which corresponds to a situation when no friction force/impulse was needed to prevent sliding

Again, let us temporarily assume a single contact scenario: the MLCP corresponding to 5.18 is:

$$\begin{bmatrix} \boldsymbol{M} & -\boldsymbol{n} & -\boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{n}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{D}^T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{e} \\ \boldsymbol{0} & \mu & -\boldsymbol{e}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^{l+1} \\ c_n \\ \boldsymbol{\beta} \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} -\boldsymbol{M}\boldsymbol{v}^l - h\boldsymbol{k} \\ (\boldsymbol{n}^T\boldsymbol{q}^l - \alpha_0)/h \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}, \tag{5.19a}$$

$$r_1 = \boldsymbol{0}, \tag{5.19b}$$

$$0 \leq r_2 \quad \perp \quad c_n \geq 0, \tag{5.19c}$$

$$0 \leq r_3 \quad \perp \quad \boldsymbol{\beta} \geq \boldsymbol{0}, \tag{5.19d}$$

$$0 \leq r_4 \quad \perp \quad \lambda \geq 0, \tag{5.19e}$$

where $r_1$, $r_2$, $r_3$ and $r_4$ denote consecutive rows of the system 5.19a. Using the same approach as in the frictionless case, the above MLCP can be easily turned into an LCP.

### 5.2.3D Notes on non-linearity

In general, there will obviously be scenarios where the simple version of the non-penetration constraint (Eq. 5.13) cannot be applied. In cases like this, Stewart and Trinkle advise to replace the complementarity condition with the following non-linear one:

$$f(\boldsymbol{q}^{l+1}) \geq 0 \quad \perp \quad c_n \geq 0, \tag{5.20}$$

and evaluate matrices and vectors dependent on $\boldsymbol{q}$ at $l+1$. This givens rise to a non-linear complementarity problem. The authors propose to solve it iteratively (starting from a simple prediction of $\boldsymbol{q}^{l+1}$) by a fixed point scheme.

It should be noted that even without the non-linear form of the non-penetration constraint, evaluating matrices and vectors for $\boldsymbol{q}^{l+1}$ corresponds to an implicit discretization scheme and also results in an NCP [92].

### 5.2.4 Anitescu & Potra

Despite their earlier work, Anitescu and Potra are most often recognized for their modification of the Stewart-Trinkle time-stepper presented in [2]. The introduced alterations included a different form of the normal contact constraint (which has a major impact on the solvability) and introduction of partially-elastic collisions. It may be worth noting that these authors did not abandon that research course and provided further propositions of developing the basic time-stepper.

In most aspects, the time-stepper due to Anitescu and Potra is exactly the same as the one originally proposed by Stewart and Trinkle: what will follow is the description of the differences, omitting the parts where these two propositions are identical.

#### 5.2.4A General

The general assumptions are the same as those described for Stewart-Trinkle time-stepper in Sec. 5.2.3A.

#### 5.2.4B Normal contact condition

The main achievement of [2] is that they managed to prove that a minor modification in the Stewart-Trinkle time-stepper guarantees a solution for any number and configuration of contacts. The solution is not guaranteed to be unique (which is however obvious), but Lemke's algorithm will always be able to find the solution.

The aforementioned modification is to replace the non-penetration constraint of Eq. 5.13 with its velocity-level version:

$$\boldsymbol{n}^T\boldsymbol{v}, \tag{5.21}$$

which reduces the index . The obvious and expected drawback of this method is the drift since the the constraint is no longer represented directly; projection is proposed as means to fight the drift.

The discretized EOM becomes, for the $l+1^{st}$ step (the complementarity conditions are again provided for the $j^{th}$ contact but are meant for all $p$ of them):

$$\boldsymbol{M}(\boldsymbol{q}^l + h\boldsymbol{v}^l)(\boldsymbol{v}^{l+1} - \boldsymbol{v}^l) = \sum_{j=1}^{p}(\boldsymbol{n}^{(j)}c_n^{(j)} + \boldsymbol{D}^{(j)}\boldsymbol{\beta}^{(j)}) + h\boldsymbol{k}(\boldsymbol{q}^l + h\boldsymbol{v}^l/2, \boldsymbol{v}^l) \tag{5.22a}$$

$$\boldsymbol{n}^{(j)T}\boldsymbol{v}^{l+1} \geq 0 \quad \perp \quad c_n^{(j)} \geq 0, \tag{5.22b}$$

$$\lambda^{(j)}\boldsymbol{e}^{(j)} + \boldsymbol{D}^{(j)T}\boldsymbol{v}^{l+1} \geq \boldsymbol{0} \quad \perp \quad \boldsymbol{\beta}^{(j)} \geq \boldsymbol{0}, \tag{5.22c}$$

$$\mu^{(j)}c_n^{(j)} - \boldsymbol{e}^{(jT)}\boldsymbol{\beta}^{(j)} \geq 0 \quad \perp \quad \lambda^{(j)} \geq 0. \tag{5.22d}$$

As one can see by inspecting 5.22b, the non-penetration constraint is evaluated at $l+1$, which means that the solution of the above problem will not generate penetration at the end of the time-step, but now it is ensured only in terms of normal velocity, i.e. the bodies will not tend to further penetrate.

#### 5.2.4C Partially elastic collisions

Anitescu & Potra have also managed to endow Stewart-Trinkle time-stepper with (optional) partially elastic collision. The restitution law used is based on the Poisson's hypothesis, so there are two distinct phases, named compression and decompression, and the coefficient of restitution relates their corresponding impulses. Each phase is assumed to be instantaneous and while that implies that impulses that arise are known to represent impulsive forces[16], the impulse-velocity relation of Eq. 5.22 can still be applied to model these phases after the components that vanish as $h \to 0$ are dropped.

It is important to note that in order to resolve collisions using a restitution law of choice, means must be provided so that they can be detected and the corresponding system configuration determined. The procedure proposed in [2] is to solve the basic problem of Eq. 5.22, check whether any collisions have occurred along the corresponding time-step and if so - estimate time of the earliest, back-up the simulation to that instant and resolve the collision.

Let $\boldsymbol{q}^i$ denote the system configuration for which a collision is detected and $\boldsymbol{v}^{i-}$, $\boldsymbol{v}^{i0}$ and $\boldsymbol{v}^{i+}$ be the pre-collision, post-compression and post-collision system velocities, respectively; $c_n^c$ and $c_n^d$ represent normal impulses in the compression and decompression phase, respectively.

The compression phase is modeled by:

$$\boldsymbol{M}(\boldsymbol{q}^i)(\boldsymbol{v}^{i0} - \boldsymbol{v}^{i-}) = \sum_{j=1}^{p}(\boldsymbol{n}^{(j)}c_n^{c(j)} + \boldsymbol{D}^{(j)}\boldsymbol{\beta}^{(j)}), \tag{5.23a}$$

$$\boldsymbol{n}^{(j)T}\boldsymbol{v}^{i0} \geq 0 \quad \perp \quad c_n^{c(j)} \geq 0, \tag{5.23b}$$

along with the complementarity conditions 5.22c and 5.22d. Similarly, the decompression phase is modeled by:

$$\boldsymbol{M}(\boldsymbol{q}^i)(\boldsymbol{v}^{i+} - \boldsymbol{v}^{i0}) = \sum_{j=1}^{p}(\boldsymbol{n}^{(j)}c_n^{d(j)} + \boldsymbol{D}^{(j)}\boldsymbol{\beta}^{(j)}) + \boldsymbol{I}_r, \tag{5.24a}$$

$$\boldsymbol{n}^{(j)T}\boldsymbol{v}^{i+} \geq 0 \quad \perp \quad c_n^{d(j)} \geq 0, \tag{5.24b}$$

along with the complementarity conditions 5.22c and 5.22d. As one can see, the only difference between these phases is the presence of the free term $\boldsymbol{I}_r$ for

[16] There is no way of telling whether impulsive forces have arisen if the time-window is non-zero [115].

decompression - it is an external restitution impulse due to the release of the energy absorbed by compressive deformation; Anitescu & Potra assumed:

$$\boldsymbol{I}_r = \sum_{j=1}^{p} \epsilon_j \boldsymbol{n}^{(j)} c_n^{c(j)}, \tag{5.25}$$

where $\epsilon_j$ is the coefficient of restitution for the $j^{th}$ collision, but a more general form is also briefly discussed.

### 5.2.5 Hahn

Hahn is very often referred to as a pioneer of collision-based simulation. His proposition of a simulator in [56] was not very detailed, several features are poorly discussed, but the presented paradigm of treating *all* interactions between rigid bodies using collision impulses was fresh and visibly different from other publications of that time. Therefore, while Hahn's simulator itself was far from perfect (as described in [56]), it could be treated as a root from which the now-popular collision-based methods have stemmed out.

#### 5.2.5A General

Let us assume that the current simulation time is given by $t_0$ and the system at $t_0$ consists of $n$ rigid bodies following projectile motion. Hahn's idea was quite simple: first, configurations of all the rigid bodies in the system are independently updated basing on their state, Newton-Euler equations and a numerical integration scheme of choice which tentatively advances the simulation by a constant step $dt$; then, all pairs of these bodies are checked to see whether a contact has occurred in $[t_0, t_0 + dt]$ between them - if so, impact dynamics are applied to determine new state.

#### 5.2.5B Resolving collisions and resting contact

For each pair of colliding bodies, Hahn applies the impact dynamics equations with Newton's restitution law which is enough equations to handle frictionless cases. If friction is present, two additional equations are needed. In order to provide them, it is first assumed that the post-collision slip velocity is zero and the resulting system of 15 equations in 15 unknowns (6 velocity variables for each body and the 3D collision impulse) is solved. If for the contact normal vector $\boldsymbol{N}$ and the determined impulse $\boldsymbol{P}$ relation

$$\|\boldsymbol{N} \times (\boldsymbol{P} \times \boldsymbol{N})\| < \mu \|\boldsymbol{P} \cdot \boldsymbol{N}\| \tag{5.26}$$

holds then the no-slip assumption is accepted[17]. Otherwise, the two equations for slip velocity are replaced by another two which assign zero to the component of the collision impulse along $\boldsymbol{P} \times \boldsymbol{N}$ and $\mu\|\boldsymbol{P} \cdot \boldsymbol{N}\|$ to the one along $\boldsymbol{N} \times (\boldsymbol{P} \times \boldsymbol{N})$, which yields a new system to be solved for the slipping case. Clearly, this is an approach at the Coulomb model of friction.

For resting contact, Hahn proposed to use the very same procedure and as means to detect resting cases - a small threshold for the relative normal velocity between bodies in contact[18].

#### 5.2.5C Handling interpenetration

Due to discrete collision detection (i.e. performed at pre-defined times, independent of the simulation course), the detected contacts will, in most cases, correspond to inter-penetration. As means to alleviate this, Hahn discusses the need to *back up* the colliders in time to the instant of collision, but it is really hard to say if this method is actually applied as opposed to simple geometric operation of moving the colliding bodies apart until the inter-penetration vanishes (which seems more probable in the context of mutli-body contact discussed in the next section).

#### 5.2.5D Multiple contacts

Multiple simultaneous contacts/collisions are dealt with in an ordered manner using a scene graph whose edges are contacts, with a common node for all immovable bodies and separate node for each movable one. Hahn seems concerned only with the cases of pushing a movable object into an immovable one (since such object cannot be *backed up*) - breadth-first graph traversal starting from the immobile node eliminates such a possibility (with the unresolved exception of cycles).

### 5.2.6 Mirtich

It seems that where Hahn has left, Mirtich has taken over . In his work of mid-1990s [89, 86, 87] he clearly defined and formalized the collision-based simulation; commented on its advantages and disadvantages and firmly established this paradigm in the spectrum of approaches. Author's subjective opinion is that Mirtich's ideas as described in the referenced works constitute a model collision-based simulator.

[17] The vector triple product formulation can be easily rephrased into a more intuitive form: by (vector) triple product expansion we have $\boldsymbol{N} \times (\boldsymbol{P} \times \boldsymbol{N}) = (\boldsymbol{N} \cdot \boldsymbol{N})\boldsymbol{P} - (\boldsymbol{N} \cdot \boldsymbol{P})\boldsymbol{N}$ which due to $\|\boldsymbol{N}\| = 1$ becomes $\boldsymbol{P} - (\boldsymbol{N} \cdot \boldsymbol{P})\boldsymbol{N}$.

[18] For reasons unclear to the author, Hahn is not really credited for applying series of impulses to handle resting contact; see e.g. [89, 33]. The only reasonable explanation of this fact may be due to limited detail and dubious generality of [56].

Mirtich's proposition in its entirety was presented in his PhD dissertation [87][19], but we shall restrict most of the discussion here to his slightly earlier work [89], which conveys the core ideas yet requires far less explanation.

#### 5.2.6A General

The highest-level concept of Mirtich's simulator is to

- estimate the time interval over which no contact/collision will occur; this collision-free interval ends at so called *time of impact* (TOI); this is a conservative estimation, i.e. TOI can be underestimated, but never overestimated; the pair of bodies which are predicted to collide at the TOI are called the *critical pair*
- treat the system as a collection of independent bodies and use basic Newton-Euler equations along with numerical integration to advance the simulation by the estimated TOI
- check if the critical pair has actually collided; if so - resolve the collision

The problem of multiple simultaneous collisions is not really considered and it is hard to say, whether it is an issue or not.

#### 5.2.6B Resolving collisions

Let there be a 3D Cartesian coordinate frame, called a *collision frame*, with its origin in the point of detected contact and z-axis aligned with the contact normal. Furthermore, let $\boldsymbol{u}$ be the relative velocity of the contacting bodies expressed in collision frame coordinates. Obviously, if $u_z$ is negative then bodies are not only in *geometric* contact, but are also traveling into each other, which implies that a collision occurs and a collision impulse, $\boldsymbol{p}$, must be determined and applied to prevent inter-penetration. As usual, it is done by first evaluating $\Delta\boldsymbol{u}$ due to the collision and then inverting/solving for $\boldsymbol{p}$:

$$\Delta\boldsymbol{u} = \boldsymbol{M}\boldsymbol{p}, \tag{5.27}$$

where $\boldsymbol{M}$ is a $3 \times 3$ inertia matrix representing inertial properties of the colliding bodies. Collision duration is assumed to be infinitesimal which means that body configurations remain constant over its course which in turn make $\boldsymbol{M}$ fixed as well.

Despite assuming infinitesimal collision duration, Miritch does not use a simple mapping from pre- to post-collision velocity, but rather performs what he

[19] It consisted of a detailed descriptions of collision detection procedures, collision resolution process using Stronge's hypothesis and introduction of *hybrid simulation* understood as collision- and constrained-based methods fused in order to simulate articulated systems. The last of these will be described in the subsequent section on bilaterally constrained systems.

calls *collision integration*, i.e. tracks relative velocity evolution over the collision time span. It is important if one considers Coulomb friction model, which discriminates different cases depending on $\boldsymbol{u}$.

Now, since we are considering instantaneous collisions, their duration is infinitesimal and thus parameterizing the collision duration with time reintroduces all the problems characteristic to acceleration-level formulations. To remedy this, Mirtich proposed to use a different parameter, $\gamma$, which (like time) monotonically increases during the collision and w.r.t. to which (unlike time) the integrated quantities are continuous. The natural (but not the only one) choice for $\gamma$ is the normal component of the collision impulse.

Let the sliding velocity be given by $\boldsymbol{u}_t$, i.e. $\boldsymbol{u}_t = [u_x \; u_y]^T$. While $\boldsymbol{u}_t$ is non-zero, by the sliding case of the Coulomb law and the fact that the time-derivative of an impulse is force, it can be easily proven that:

$$\frac{d\boldsymbol{p}}{d\gamma} = \begin{bmatrix} -\mu\, u_x/|\boldsymbol{u}_t| \\ -\mu\, u_y/|\boldsymbol{u}_t| \\ 1 \end{bmatrix}, \tag{5.28}$$

where $\mu$ is the coefficient of friction. Therefore, if 5.27 is differentiated with respect to $\gamma$, one obtains:

$$\frac{d\boldsymbol{u}}{d\gamma} = \boldsymbol{M} \begin{bmatrix} -\mu\, u_x/|\boldsymbol{u}_t| \\ -\mu\, u_y/|\boldsymbol{u}_t| \\ 1 \end{bmatrix}, \tag{5.29}$$

which is a differential equation describing the evolution of relative slip velocity, $\boldsymbol{u}$, as long as it is non-zero. Eq. 5.29 is numerically integrated w.r.t. $\gamma$ which allows for tracking $\boldsymbol{u}$ throughout the collision, but we do not really know when this process should be terminated. In [89] Poisson's hypothesis fills this gap:

- the integration is performed until $u_z$ reaches zero which means that the compression phase is over and we can determine the corresponding impulse
- Poisson's law is used to determine the normal component of the total collision impulse using its value after compression
- the integration continues until $\gamma$ reaches this total value

If, at some point, the collision integration process yields $\boldsymbol{u}_t = \boldsymbol{0}$ then sticking occurs and friction force is not uniquely given by the model. What follows is that the sliding case relation in Eq. 5.28 is no longer valid. The procedure applied in such cases is reminiscent of Hahn's approach: the differential version of 5.27 is solved for $\boldsymbol{p}_t = [dp_x/d\gamma \; dp_y/d\gamma]^T$ assuming that $\boldsymbol{u}_t = \boldsymbol{0}$ and $dp_z/d\gamma = 1$; if for the resultant $\boldsymbol{p}_t$ relation:

$$\left(\frac{dp_x}{d\gamma}\right)^2 + \left(\frac{dp_y}{d\gamma}\right)^2 \leq \mu^2 \tag{5.30}$$

holds then we conclude that friction is sufficient for sticking to continue till the collision ends; moreover - the determined $\boldsymbol{p}_t$ will remain unchanged as well.

If the relation does not hold, then we conclude that $\boldsymbol{u}_t = \mathbf{0}$ was only a transient state and sliding will resume. Although the case was not classified as sticking, Eq. 5.29 cannot be applied (division by zero) and thus $d\boldsymbol{u}_t/d\gamma$ must be found by other means. Mirtich claims that it can be determined by solving a quartic equation [89]

#### 5.2.6C Resolving resting contact

Mirtich wished to keep the simple contact-local nature of the algorithm to unambiguously differentiate it from constraint-based methods so no system-wide information is used not only for collision, but also when handling resting contact cases.

If the relative normal velocity is below a certain threshold, the contact is classified as resting and thus treated differently. Since reaction forces are supposed to be workless, Mirtich proposed to apply perfectly elastic collisions to model them: contacting bodies' configurations remain the same throughout the collision and the relative velocity, $\boldsymbol{u}$, gets inverted so potential and kinetic energy is conserved[20] . The resulting impulse is applied only if it lies within the friction cone; if it does not, the usual collision integration process is launched.

The problem with applying the collision resolution procedure is that it is not really suitable for handling resting contact: the fact of bodies being continuously within their collision envelopes[21] with smaller relative velocity after each collision (and thus hardly bouncing off each other) will ultimately lead to reporting TOIs closer and closer to the current time, which will in turn bring the simulator to a halt. To alleviate this, Mirtich proposed to use a dynamic coefficients of restitution, whose values grow as the penetration into the collision envelop becomes larger (with the original coefficient as a lower bound).

### 5.2.7 Bender/Weinstein

The most recent major development of the impulse-based simulation methods manifested in the literature are due to Bender (with numerous colleagues) and to Weinstein. Their contributions are very similar (although the proposed simulators generally differ) and it is hard to say whether they were mutually inspired in any fashion, since Bender started to publish several years earlier but in German.

[20] A noteworthy side-feature of this approach is that the usual relatively complex collision integration process is skipped.

[21] Mirtich used a non-zero distance between bodies at which they were already reported as contacting (*collision epsilon*), which constitutes a *collision envelope* surrounding the actual geometry of an object. If two bodies are continuously in their collision envelopes, they are always reported as contacting by the collision detection module.

In this section, we shall mostly refer to Bender's rather than Weinstein's work since he has been more concerned with collisions and resting contact[22].

### 5.2.7A General

Although Bender's articles published in mid-2000s are notably different from one another in details (they present alternatives rather than augmentations), the core characteristics remain the same. For presentation of how Bender proposes to treat unilaterally constraints systems we chose [14].

Like Mirtich, Bender uses impulses both for colliding and resting contact, but while the collision case is roughly comparable (apart from the lack of collision integration), Bender's treatment of resting contact is remotely reminiscent of Stewart-Trinkle time-stepper since it determines impulse that satisfies a configuration constraint at the end of the time step. As far as collision detection is considered, Bender does not really clearly state whether continuous or discrete collision detection is used, but the repeated brief descriptions of CCD [13, 14] and the fact that means of correcting for the existing inter-penetration is not considered, seem to suggest that CCD *should be* applied.

A high-level view of the simulation procedure is as follows, for the time step $[t_0, t_1]$:

- detect contacts
- for contacts corresponding to colliding cases: resolve iteratively one at a time until all achieve proper post-collision velocities (within a certain tolerance); the time remains $t_0$
- for contacts corresponding to resting cases: resolve iteratively one at a time until all are guaranteed not to inter-penetrate at the end of the time step (within a certain tolerance); the time advances to $t_1$

### 5.2.7B Resolving collisions

Although, unlike Mirtich, Bender does consider multiple simultaneous collisions and resolves them at a time, it is done in a sequential manner, one at a time and hence the procedure for each single collision is quite the same (with the exception of collision integration).

Let us start with frictionless cases: first, Newton's law is applied to map from the relative normal pre- to post-impact velocity and then Eq. 5.27 is used to determine the collision impulse necessary to get a corresponding velocity change; this initial impulse is determined for all colliding pairs; we shall call that stage the $0^{th}$ iteration. Since a single body may participate in multiple colliding contacts,

[22] Weinstein merely provides a description on how her articulation-enforcement procedures can be combined with a simulator based on [55], calling it a *hybridization*.

velocity correction is now performed sequentially for consecutive pairs, which may invalidate the previous ones and thus must be repeated iteratively to reach the desired precision for all of them: in each iteration $i$ where $i > 0$, for each colliding contact a corrective normal impulse is computed and applied under the condition that it does not cause the overall/aggregate normal impulse for the current pair to become attractive, i.e. for the $i^{th}$ iteration:

$$\boldsymbol{n} \cdot \sum_{j=1}^{i-1} \boldsymbol{p}_{n,j} \geq 0 \quad \text{and} \quad \boldsymbol{n} \cdot \sum_{j=1}^{i-1} \boldsymbol{p}_{n,j} \geq -\boldsymbol{n}\boldsymbol{p}_{n,i}, \tag{5.31}$$

where $\boldsymbol{n}$ is the contact normal vector for the current pair. If $\boldsymbol{p}_{n,i}$ violates the condition 5.31 it is *clamped* to $-\boldsymbol{n} \cdot \sum_{j=1}^{i-1} \boldsymbol{p}_{n,j}$ so that it causes aggregate impulse to vanish, but remain non-negative.

If friction enters the picture, each iteration of the above-described loop is extended by steps based on the Coulomb law *at the impulse-level* . Again - the mutual tangential-normal impulse and inter-contact correlations are handled by iterations rather than enforced in a single step. Although Bender is not really explicit about this, it seems natural that friction impulses should be evaluated *after* the corresponding normal component has been processed in the given iteration. Naturally, there are two distinct cases, if in the $i^{th}$ iteration the relative tangential velocity for a colliding contact pair is:

- non-zero then both the direction and magnitude of the friction force are uniquely defined by the Coulomb law; however, the resultant friction impulse is clamped so that it will not invert the direction of the slip velocity (most it can do is cause the velocity to vanish), i.e.

$$\boldsymbol{p}_{t,i} \leq -\boldsymbol{M}\boldsymbol{u}_{rel,t} \tag{5.32}$$

- zero then the contact pair is labeled as static and the Coulomb law provides the upper bound on the current aggregate friction impulse (friction coefficient times the magnitude of the current aggregate normal impulse); as long as a pair is labeled as static, the corresponding friction impulse is expected to cause the current slip velocity to vanish, but under the condition that the upper bound is not exceeded ; if it is - the static label is removed.

Dynamic friction does not add to the overall iteration count since there is no *good enough* criterion for it, consecutive iterations only update its value so that it matches the magnitude of the current aggregate normal impulse. On the other hand, static friction may generate the need for more iterations since it must cause the slip velocity to vanish (within a certain tolerance).

#### 5.2.7C Resolving resting contact

Resting contact is resolved very similarly to collisions, but it must first be adjusted to fit the iterative velocity correction procedure. In order to do so, there is an initial *prediction stage*: the system is stepped ahead *as if* non-penetration was not enforced and the resultant tentative configuration is used to compute the error for each resting contact pair (i.e. penetration depth). Now, Bender notes that determining a corrective impulse, which would eliminate the error and thus get a proper configuration at the end of the time step would require solving a non-linear equation[23], but proposes a different procedure. The penetration depth is disguised as a velocity by simply dividing it by the time step and and impulse is estimated which would cause the normal relative velocity to be equal to it if applied at $t_0$. This intuitively feels like we were eliminating the error at $t_f$ by applying an impulse at $t_0$ to prevent the motion which causes, but since the relative motion at the contact will in general not be linear, this is an approximation and may require several iterations to reach the desired accuracy.

Apart from that, the normal impulses are determined in the same fashion as in the colliding cases and so are the tangential impulses for dynamic friction. For static friction, it must be ensured that contact points do not move tangentially with respect to each other across the time step and this is achieved similarly to the elimination of inter-penetration.

## 5.3 Bilaterally constrained systems

One of the earliest propositions of dynamics algorithms emerged in the mid-1960s with solutions based on both Newtonian [63, 107] and Lagrangian mechanics [121]. By the mid-1970s solutions utilizing Lagrangian Multipliers for joint constraints [22, 96] appeared. What is more, Orlandea's program (ADAMS) exploited matrix sparsity (Sparse Tableaux Formulation) to improve performance. These early works are well reviewed in [99]. The *composite-rigid-body algorithm* (CRBA), first proposed in [123], harnessed the $O(n)$ recursive Newton-Euler inverse dynamics algorithm (RNEA) presented in [83] as a means of calculating the joint-space inertia matrix of a kinematic tree much faster than earlier methods. CRBA is most commonly presented as an $O(n^2)$ solution, but obviously once the entries of the inertia matrix have been determined a system of linear equations needs to be solved thus the dynamics algorithms using CRBA are collectively referred to as $O(n^3)$ algorithms.

The earliest known forward dynamics algorithm having linear time complexity was proposed by [122] but it was not until Featherstone's works on *articulated-body algorithm* (ABA) [34, 35, 36] that propagation methods became widely ac-

[23] And this is what Weinstein does to enforce articulation constraints [126].

knowledged. An alternative linear time solution is due to [8] who has coupled an *always sparse* formulation using Lagrangian Multipliers with linear-time factorization. This approach has been particularly popular within the computer graphics community where rigid body dynamics algorithms are used to build general purpose physics engines and thus solutions based on Lagrangian Multipliers, which provide natural means for mixing articulation and contact constraints, are preferred. A recent, thorough comparison of multibody dynamics algorithms can be found in [64].

### 5.3.1 Prerequisites

It is often difficult to present the development of a methodology applied (even in a very specific domain) using a unified notation, since different authors tend to use different notational conventions and assumptions. In the case of three most commonly known reduced-coordinate rigid-body algorithms, however, the work has already been done: Roy Featherstone has used spatial vector algebra to present them in a unified fashion. To avoid reinventing the wheel, we shall use the same conventions to describe these algorithms and their derivatives presented in later chapters. Therefore, we devote the current section to introduce the basics of spatial vector algebra used by Featherstone.

#### 5.3.1A Introduction

Thanks to spatial vector algebra the traditional separate treatment of the linear and angular aspects of dynamics are merged into a uniform notational form consisting of both. It should be noted that, to a certain extent, a combined representation like this can be achieved without the mathematical abstraction employed by Featherstone, e.g. [31].

#### 5.3.1B Spatial vectors

The dominating convention in rigid-body dynamics is to consider its linear and angular aspects separately by using two sets of 3D vectors and two equations of motion (often concatenated into a single system for convenience, though). This approach results from the assumption that linear and angular quantities are considered physically different [36].

Spatial algebra is a successful approach at systematic unification of these quantities; the notation is based on $6 \times 1$ vectors and corresponding $6 \times 6$ matrices which considerably simplify the equations[24]. The notational convention adopted in this work is to mark spatial quantities with a hat symbol.

[24] Which obviously comes at the cost of considerable amount of abstraction one has to bear with, more complex definitions and transformations rules.

The are two *types* of spatial vectors: (i) spatial motion vectors describing attributes of motion, i.e. velocities and accelerations; they belong to vector space $M^6$; (ii) spatial force vectors describing forces, impulses and momenta; they belong to vector space $F^6$.

$M^6$ and $F^6$ are not inner-product spaces, but there is a scalar product defined *between* them (i.e. it takes one argument from each space); the operation is order-insensitive. The physical interpretation of that product is *power*[25]. Since $M^6$ and $F^6$ have the same dimensions and scalar product between them, they are formally said to be *dual* [40].

One can define equivalent pair of spaces for different dimensions, e.g. $M^{6n}$ and $F^{6n}$ to describe the motion of and forces acting on the system of $n$ independent bodies.

### 5.3.1C Spatial velocity, force and acceleration

#### Velocity

Conventionally, the velocity of a rigid body can be momentarily described by an (arbitrarily chosen) point $O$ in the body and a pair of $3 \times 1$ vectors: the linear velocity, $\boldsymbol{v}_O$, of $O$ and an angular velocity about an axis passing through $O$, $\boldsymbol{\omega}$, of the body as a whole. Unlike $\boldsymbol{\omega}$, $\boldsymbol{v}_O$ applies explicitly to $O$ and the linear velocity of any other point, $P$, in the body is given by

$$\boldsymbol{v}_P = \boldsymbol{v}_O + \boldsymbol{\omega} \times \overrightarrow{OP} = \boldsymbol{v}_O - \boldsymbol{\omega} \times \overrightarrow{PO}. \tag{5.33}$$

The choice of the point and axis passing through it is a virtual action of choosing a frame of reference in which the linear velocity of the chosen point is a non-redundant trait of its motion since it is independent of the angular velocity of the body[26].

Now, as long a $O$ has the motion identical as the body, it does not really have to be *within it*. What is more: we are free to choose different points to describe the velocity of a rigid body at different instants. Using these two facts, we can imagine the entire space moving with the rigid body and in consecutive moments of time *keep choosing* the point in this space that is instantaneously coincident with the origin, $O$, of a chosen fixed coordinate frame. As a result, we can now say that the body is moving with velocity $\boldsymbol{v}_O = \boldsymbol{v}_P + \overrightarrow{OP} \times \boldsymbol{\omega}$ while simultaneously rotating about an axis passing through the origin with the angular velocity $\boldsymbol{\omega}$. Thus, once the coordinate frame is chosen, the spatial velocity vector describing the motion of a rigid body in the chosen frame is:

$$\widehat{\boldsymbol{v}}_O = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_O \end{bmatrix}. \tag{5.34}$$

[25] Let $\widehat{\boldsymbol{m}} \in M^6$ be velocity of a rigid body and $\widehat{\boldsymbol{f}} \in F^6$ be a force acting on it; the product $\widehat{\boldsymbol{m}} \cdot \widehat{\boldsymbol{f}}$ is the power delivered by force $\widehat{\boldsymbol{f}}$.

[26] Axis of rotation passes thorough it so the perpendicular distance between the axis and chosen point is zero.

### Force

Using the same arbitrarily chosen reference point $O$, a body can be acted upon by a linear force, $\boldsymbol{f}$, whose line of action passes through $O$ and the total moment, $\boldsymbol{n}_O$, about $O$ of all the remaining forces acting on this body. We are again performing the virtual action of choosing a frame of reference in which we combine all the active linear forces into these acting along a line passing through $O$, while the remaining ones generate the moment about it.

The point $O$ and vectors $\boldsymbol{f}$ and $\boldsymbol{n}_O$ fully describe the stimuli acting on the body and can be used to determine the total moment about any other point, $P$:

$$\boldsymbol{n}_P = \boldsymbol{n}_O + \boldsymbol{f} \times \overrightarrow{OP} = \boldsymbol{n}_O - \boldsymbol{f} \times \overrightarrow{PO}. \tag{5.35}$$

Once a coordinate frame is chosen, the spatial force vector describing the stimuli acting on the rigid body in the chosen frame is:

$$\widehat{\boldsymbol{f}}_O = \begin{bmatrix} \boldsymbol{n}_O \\ \boldsymbol{f} \end{bmatrix}. \tag{5.36}$$

### Acceleration

Spatial acceleration of a rigid body is defined as the time-derivative of its spatial velocity:

$$\widehat{\boldsymbol{a}}_O = \frac{d}{dt} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_O \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{v}}_O \end{bmatrix}. \tag{5.37}$$

This may sound pretty obvious and natural, but actually it is one of the places where spatial algebra is vividly different from conventional description of motion.

The difference becomes apparent when one considers a scenario of a rigid body rotating about an axis with a constant angular velocity - the velocity of any point in this body not on the axis of rotation is obviously not constant and thus an acceleration occurs. In the very same scenario the spatial velocity is constant and thus spatial acceleration is zero, since it does not describe the motion of the body by providing motion of a point in it, but consider the entire space moving with the body.

In more practical terms it means that what is conventionally provided as an acceleration of a rigid body differs from the *linear* component of the spatial acceleration and cannot be treated synonymously. Luckily, Featherstone proved [36, 40] that this difference boils down to:

$$\widehat{\boldsymbol{a}} = \begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \ddot{\boldsymbol{r}} - \boldsymbol{\omega} \times \dot{\boldsymbol{r}} \end{bmatrix}, \tag{5.38}$$

where $\boldsymbol{\omega}$ is the angular velocity of the body and $\boldsymbol{r}$ is the position of the point in the body conventionally used to describe its motion. Featherstone calls $\boldsymbol{\omega} \times \dot{\boldsymbol{r}}$ the *Coriolis term.*

Such a definition of spatial acceleration allows for operations like:

$$\widehat{\boldsymbol{a}}_{rel} = \widehat{\boldsymbol{a}}_2 - \widehat{\boldsymbol{a}}_1 \tag{5.39}$$

where $\widehat{\boldsymbol{a}}_{rel}$ is the relative acceleration of body 2 w.r.t. to body 1, assuming that they travel with accelerations given by $\widehat{\boldsymbol{a}}_2$ and $\widehat{\boldsymbol{a}}_1$, respectively. Conventional accelerations are not that simple to handle in general.

### 5.3.1D Spatial vectors' coordinate transforms

Transforming spatial vectors from one coordinate frame to another encapsulates both the conventional coordinate change due to differently oriented axes of the associated Cartesian frame, but also the change of the origin/reference point which is a part of the description of rigid body motion or stimuli acting on it. The spatial vector transformed this way describes the same motion/force but with respect to different reference point and in different coordinates.

If $A$ and $B$ are Cartesian frames and the reference points $O$ and $P$ are their respective origins, the spatial motion vector transform from $A$ to $B$ is:

$$^{B}\boldsymbol{X}_{A} = \begin{bmatrix} \boldsymbol{E} & \boldsymbol{0} \\ -\boldsymbol{E}\boldsymbol{r}\times & \boldsymbol{E} \end{bmatrix}, \tag{5.40}$$

where $\boldsymbol{E}$ is a $3 \times 3$ alias rotation matrix which changes the coordinates of $3 \times 1$ vectors from $A$ to $B$, $\boldsymbol{r}$ expresses $\overrightarrow{OP}$ in $A$ coordinates and the symbol $\times$ is the cross product operator. Under the same assumptions, the spatial force vector transform from $A$ to $B$ is:

$$^{B}\boldsymbol{X}^{*}_{A} = \begin{bmatrix} \boldsymbol{E} & -\boldsymbol{E}\boldsymbol{r}\times \\ \boldsymbol{0} & \boldsymbol{E} \end{bmatrix}, \tag{5.41}$$

### 5.3.1E Spatial vectors' cross products

Cross product is defined between two motion vectors and between a motion and force vector. Let $\widehat{\boldsymbol{v}} = [\boldsymbol{\omega}^T\, \boldsymbol{v}_O^T]^T$ and $\widehat{\boldsymbol{\nu}} = [\boldsymbol{\nu}^T\, \boldsymbol{\nu}_O^T]^T$ be two spatial motion vectors and $\widehat{\boldsymbol{f}} = [\boldsymbol{f}_O^T\, \boldsymbol{f}^T]^T$ be spatial force vector; the spatial cross products are defined as:

$$\widehat{\boldsymbol{v}}_O \times \widehat{\boldsymbol{\nu}}_O = \begin{bmatrix} \boldsymbol{\omega} \times \boldsymbol{\nu} \\ \boldsymbol{\omega} \times \boldsymbol{\nu}_O + \boldsymbol{v}_O \times \boldsymbol{\nu} \end{bmatrix} \tag{5.42}$$

and

$$\widehat{\boldsymbol{v}}_O \times^{*} \widehat{\boldsymbol{f}}_O = \begin{bmatrix} \boldsymbol{\omega} \times \boldsymbol{f}_O + \boldsymbol{v}_O \times \boldsymbol{f} \\ \boldsymbol{\omega} \times \boldsymbol{f} \end{bmatrix}. \tag{5.43}$$

#### 5.3.1F Spatial momentum and inertia

**Momentum**

If a rigid body has mass $m$ and rotational inertia, $I_{CM}$, about its center of mass (which currently coincides with the point $C$), then with respect to $C$ its linear and angular momenta are given by:

$$\boldsymbol{l} = m\boldsymbol{v}_C \tag{5.44}$$

and

$$\boldsymbol{h}_C = \boldsymbol{I}_{CM}\boldsymbol{\omega}, \tag{5.45}$$

respectively, where $\boldsymbol{v}_C$ is the (linear) velocity of $C$ and $\boldsymbol{\omega}$ is the angular velocity of the body. Since $C$ currently coincides with the center of mass, $\boldsymbol{h}_C$ is the spin angular momentum which is an intrinsic feature of a rotating body. Should we wish to determine the angular momentum of this body about any point $P$ different than $C$:

$$\boldsymbol{h}_P = \boldsymbol{h}_C + \boldsymbol{p} \times \overrightarrow{CP} = \boldsymbol{h}_C + \overrightarrow{PC} \times \boldsymbol{l}, \tag{5.46}$$

which means that the linear momentum, spin angular momentum and the chosen reference point uniquely describe the dynamic state of the body. If we again allow the entire space to move with the body, fix coordinate frame within this space and choose its origin (as a reference point) to be a $C$, we can define the spatial momentum of the rigid body with respect to that frame as:

$$\widehat{\boldsymbol{h}}_C = \begin{bmatrix} \boldsymbol{h}_C \\ \boldsymbol{l} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{CM}\boldsymbol{\omega} \\ m\boldsymbol{v}_C. \end{bmatrix} \tag{5.47}$$

If we choose any other point, $O$, as the origin for this frame, then the spatial momentum of the body with respect to that frame is:

$$\widehat{\boldsymbol{h}}_O = \begin{bmatrix} \boldsymbol{h}_O \\ \boldsymbol{l} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{CM}\boldsymbol{\omega} + \overrightarrow{OC} \times m\boldsymbol{v}_C \\ m\boldsymbol{v}_C \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \overrightarrow{OC}\times \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{h}_C. \tag{5.48}$$

As one can see, it is a spatial force vectors transformation from the coordinate frame with the origin at $C$ to one with origin at $C$ assuming their axes have the same orientations[27].

**Inertia**

Spatial inertia, $\widehat{\boldsymbol{I}}$, of a rigid body is defined as a transformation of its spatial velocity into its spatial momentum:

$$\widehat{\boldsymbol{h}} = \widehat{\boldsymbol{I}}\widehat{\boldsymbol{v}}. \tag{5.49}$$

[27] Not to blur the formulae, these transformations are limited to the change of origin/reference point, i.e. the coordinate frames are assumed to have identical orientations.

Let us first resolve the above general formula in coordinate frame with its origin ate body's center of mass:

$$\widehat{\boldsymbol{h}}_C = \widehat{\boldsymbol{I}}_C \widehat{\boldsymbol{v}}_C = \begin{bmatrix} \boldsymbol{I}_{CM}\boldsymbol{\omega} \\ m\boldsymbol{v}_C \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{CM} & \mathbf{0} \\ \mathbf{0} & m \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_C \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{CM} & \mathbf{0} \\ \mathbf{0} & m \end{bmatrix} \boldsymbol{v}_C. \tag{5.50}$$

Thus:

$$\widehat{\boldsymbol{I}}_C = \begin{bmatrix} \boldsymbol{I}_{CM} & \mathbf{0} \\ \mathbf{0} & m \end{bmatrix} \tag{5.51}$$

is the spatial inertia of the body if and only if the coordinate frame origin coincides with the center of mass of the body and will thus be sometimes denoted as $\widehat{\boldsymbol{I}}_{CM}$. In other cases, a double transformation is needed[27]:

- spatial velocity transformation into the coordinate frame at $C$
- spatial force/momentum transformation back to the coordinate frame at $O$

which yields an expression:

$$\widehat{\boldsymbol{h}}_O = \widehat{\boldsymbol{I}}_O \widehat{\boldsymbol{v}}_O = \begin{bmatrix} \mathbf{1} & \overrightarrow{OC}\times \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \widehat{\boldsymbol{I}}_C \begin{bmatrix} \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} \\ \overrightarrow{OC}\times^T & \mathbf{1}_{3\times3} \end{bmatrix} \widehat{\boldsymbol{v}}_O. \tag{5.52}$$

By rephrasing the above, we obtain the general form of the spatial inertia matrix, namely:

$$\widehat{\boldsymbol{I}}_O = \begin{bmatrix} \boldsymbol{I}_{CM} + m\overrightarrow{OC} \times \overrightarrow{OC}\times^T & m\overrightarrow{OC}\times \\ m\overrightarrow{OC}\times^T & m\mathbf{1}_{3\times3} \end{bmatrix}. \tag{5.53}$$

### 5.3.1G Motion subspace

If a rigid body is free to move then n order to express its velocity one needs all dimensions of $M^6$. However, if the body is attached to some other element of the system with a joint (which we shall call this body's *inboard joint*), then its velocity is limited to a subspace $S \subset M^6$. This subspace is called *motion subspace* and it is defined for each body in the system by matrix $\widehat{\boldsymbol{S}}_i$ whose columns span the motion subspace of the $i^{th}$ body. What follows is that the following relation holds for all bodies in the system:

$$\widehat{\boldsymbol{v}}_i = \widehat{\boldsymbol{S}}_i \dot{\boldsymbol{q}}_i, \tag{5.54}$$

where $\widehat{\boldsymbol{v}}_i \in M^6$ is the spatial velocity of the $i^{th}$ body and $\dot{\boldsymbol{q}}_i$ is a vector containing its coordinates in $S_i \subseteq M^6$. In other words, matrix $\widehat{\boldsymbol{S}}_i$ is the velocity transformation matrix of the embedding technique described in Sec. 4.3.3 and $\dot{\boldsymbol{q}}_i$ is the vector of independent velocity variables of the $i^{th}$ body.

Note that the concept of motion subspace includes free bodies as a special case when $S = M^6$.

#### 5.3.1H Joint-space coordinates

The notation for the vectors of generalized configuration, velocity and acceleration variables is maintained, i.e. they are denoted by $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$, respectively. However, they are now explicitly assumed to be joint-space vectors: each of them is a concatenation of the corresponding reduced-coordinate vectors describing individual bodies in the system *as long as it is a kinematic tree*. The joint-space EOM is a the same as Eq. 4.19 but will be denoted by:

$$\boldsymbol{\tau} = \boldsymbol{H}\ddot{\boldsymbol{q}} + \boldsymbol{C}, \tag{5.55}$$

where $\boldsymbol{H}$ is a joint-space inertia matrix, $\boldsymbol{\tau}$ is a joint-space vector of force variables and $\boldsymbol{C}$ is a joint-space bias vector representing velocity-product and external force terms.

If loops appear in the topology joint-space coordinates become redundant, but can be still used as the generalized coordinates as long as the additional constraints' conditions are specified.

### 5.3.2 Recursive Newton-Euler Algorithm

The Recursive Newton-Euler Algorithm (RNEA) was originally proposed in [83] and is essentially the same as the one described by Featherstone [40] whose input boils down to rephrasing RNEA in terms of spatial rather than conventional 3D vectors.

RNEA is principally used as an inverse dynamics (ID) algorithm, which is actually beyond the scope of this work: the forthcoming brief presentation is provided since RNEA is used (more or less directly) by the two, reduced-coordinates forward dynamics (FD) algorithms, i.e. CRBA and ABA.

#### 5.3.2A The ID algorithm

In coarse terms, the algorithm consists of two traversals of the kinematic tree: outward (from the root to the leaves) and inward (from the leaves to the root).

**Outward iterations**

In this stage, spatial velocities and accelerations of the bodies are evaluated basing on the system configuration, joint-space velocity and acceleration; one they have been determined, the net force acting on the body can be calculated using EOM of each individual body. Starting from the root, the following formulae are evaluated for each body $i$:

$$\widehat{\boldsymbol{v}}_i = \widehat{\boldsymbol{v}}_{\lambda(i)} + \widehat{\boldsymbol{S}}_i\dot{\boldsymbol{q}}_i \tag{5.56}$$

$$\widehat{\boldsymbol{a}}_i = \widehat{\boldsymbol{a}}_{\lambda(i)} + \widehat{\boldsymbol{S}}_i\ddot{\boldsymbol{q}}_i + \dot{\widehat{\boldsymbol{S}}}_i\dot{\boldsymbol{q}}_i \tag{5.57}$$

$$\widehat{\boldsymbol{f}}_{B,i} = \widehat{\boldsymbol{I}}_i\widehat{\boldsymbol{a}}_i + \widehat{\boldsymbol{v}}_i \times^* \widehat{\boldsymbol{I}}_i\widehat{\boldsymbol{v}}_i, \tag{5.58}$$

where $\widehat{\boldsymbol{f}}_{B,i}$ denotes the net force acting on body $i$, $\widehat{\boldsymbol{v}}_i = \boldsymbol{0}$ and $\widehat{\boldsymbol{a}}_0$ is assumed to be the negated gravity vector (if gravitational acceleration is considered). Notice, that in order to determine the spatial velocity and acceleration for any body, the corresponding vectors for its parent need to be known - this implies outward traversal of the topology.

#### Inward iterations

In this stage, the force acting on each body via its inboard joint is calculated from the net forces. Starting from the leaves, the following formulae are evaluated for each body $i$:

$$\widehat{\boldsymbol{f}}_i = \widehat{\boldsymbol{f}}_{B,i} - \widehat{\boldsymbol{f}}_{x,i} + \sum_{j \in \mu(i)} \widehat{\boldsymbol{f}}_j \tag{5.59}$$

$$\boldsymbol{\tau}_i = \widehat{\boldsymbol{S}}_i^T \widehat{\boldsymbol{f}}_i, \tag{5.60}$$

where $\widehat{\boldsymbol{f}}_i$ is the spatial force that body $i$ is acted upon via its inboard joint, $\boldsymbol{\tau}_i$ is its component generated by the inboard joint of the $i^{th}$ body, $\widehat{\boldsymbol{f}}_{x,i}$ is a *known* external force and $\mu(i)$ symbolizes an index set of the $i^{th}$ body's children. Notice, that in order to determine the spatial force for any body, the corresponding vectors of its children need to be known - this implies inward traversal of the hierarchy.

### 5.3.3 Composite Rigid Body Algorithm

The Composite Rigid Body Algorithm (CRBA) has been first introduced in [123], but the name is a bit younger and due to Featherstone. CRBA is not really a forward dynamics algorithm by itself: its purpose is to calculate the joint-space inertia matrix of the system. This allows system EOM to be formulated explicitly and then solved for system accelerations by any appropriate method. Therefore, we shall often use the term *CRBA-based (forward) dynamics algorithm.*

#### 5.3.3A Composite rigid body

A composite rigid body (CRB) is a collection of individual rigid bodies treated as a single rigid object itself (i.e. any joint connections between the constituent bodies are ignored). If from the overall system kinematic tree we isolate a subtree rooted at body $i$ and treat it as a CRB, its composite inertia can be calculated as follows[28]:

$$\widehat{\boldsymbol{I}}_i^C = \widehat{\boldsymbol{I}}_i + \sum_{j \in \mu(i)} \widehat{\boldsymbol{I}}_j^C, \tag{5.61}$$

where $\widehat{\boldsymbol{I}}_i^C$ is called the composite inertia of the subtree rooted at $i$ and $\mu(i)$ is an auxiliary symbol representing the index set of all topological children of body $i$.

[28] Ancestor-descendant coordinate transforms skipped not to blur the formulae.

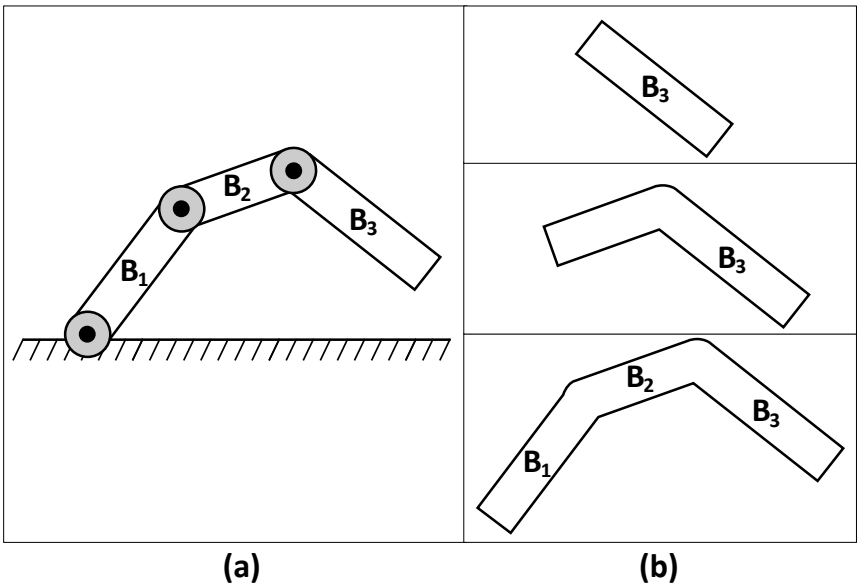



Figure 5.1: Illustration of the concept of composite rigid bodies: (a) the entire system; (b) three successive (top to bottom) composite rigid bodies considered by CRBA; notice that the non-root joints of each subtree are *frozen* and the root joint is ignored body which makes it a floating compound rigid body

#### 5.3.3B The FD algorithm

The entire FD procedure is:

1. determine the joint-space bias force vector $\boldsymbol{C}$
2. use CRBA to determine the joint-space inertia matrix $\boldsymbol{H}$
3. solve Eq. 5.55 for $\ddot{q}$

**Step 1**

If we look at the EOM we can see that if all joint accelerations are zero it takes the simple form:

$$\boldsymbol{\tau} = \boldsymbol{H}\boldsymbol{0} + \boldsymbol{C} = \boldsymbol{C} \tag{5.62}$$

which makes it apparent that $\boldsymbol{C}$ vector is the vector of joint-forces which generate zero joint accelerations. If we pose it as an inverse dynamics problem where the question asked is: *what joint-space forces need to be applied to achieve zero joint-space accelerations given the system state*, the solution will be the vector $\boldsymbol{C}$. Thus, we can employ RNEA to handle the first step.

**Step 2**

The CRBA algorithm is based upon the fact that the inertial properties of the system depend only on its configuration, which means it does not change when

the joint velocities or external forces acting on the system are modified. This fact allows us to make the assumption that the system is at rest (Coriolis and centrifugal are zero) and not influenced by any forces other than those in , which is very convenient since it allows for yet another simplification we can simplify the main equation, namely:

$$\boldsymbol{\tau} = \boldsymbol{H}\ddot{\boldsymbol{q}} + \mathbf{0} = \boldsymbol{H}\ddot{\boldsymbol{q}} \tag{5.63}$$

We now introduce a $\boldsymbol{\delta}_i$ vector which is the same size as all the joint-space vectors and contains a 1 in its $i^{th}$ element and zeros elsewhere. $\boldsymbol{\delta}_i$ is a unit accelerations at the $i^{th}$ degree of freedom of the system. We can again make use of the inverse dynamics to answer a differently posed question – what joint-space forces are needed to achieve an acceleration given by $\boldsymbol{\delta}_i$. The result will be the $i^{th}$ column of the generalized inertia matrix $\boldsymbol{H}$. Implementing this approach directly is far from optimal but serves well for illustration purposes.

CRBA is a more efficient alternative to the above approach proposed by Featherstone : it works by treating the consecutive subtrees of the overall kinematic tree as composite rigid bodies (hence the name) accelerating about a single axis of the inboard joint of the subtree root body.

Using the notion of composite inertia, the entries of $\boldsymbol{H}$ are filled as follows[28]:

$$\boldsymbol{H}[i,j] = \begin{cases} \widehat{\boldsymbol{S}}_i^T \widehat{\boldsymbol{I}}_i^C \widehat{\boldsymbol{S}}_j & \text{if body } j \text{ is an ancestor of body } i \\ \widehat{\boldsymbol{S}}_i^T \widehat{\boldsymbol{I}}_j^C \widehat{\boldsymbol{S}}_j & \text{if body } i \text{ is an ancestor of body } j \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{5.64}$$

**Step 3**

What remains, is to solve the EOM which thanks to the matrix $\boldsymbol{H}$ begin symmetric positive-definite can be done efficiently using Cholesky decomposition.

To further optimize the third step, Featherstone proposed a custom factorization to exploit the sparsity patterns of $\boldsymbol{H}$ and avoid needless fill-in [39, 40], but it is not sure whether the actual implementations of generic sparse factorization algorithms harnessing vector instructions can be easily outperformed using this customized approach.

### 5.3.3C Beyond kinematic trees

Neither CRBA, nor the above-described FD algorithm handle cycles in the system topology by themselves. In order to model closed-loop systems (which includes all but the simplest contact scenarios) while still utilizing reduced coordinates, it is common to remove some of the system constraints to obtain a subsystem with a tree topology[29] (the *spanning tree* of the system [40]) and then re-apply them by

[29] So CRBA can be applied to it.

appending equations to the EOM of the spanning tree. What one obtains is the descriptor form of the EOM [66] (see Sec. 4.3.3) which can be reduced to the form of Eq. 4.14. Further treatment depends on the constraints (bilateral/unilateral, friction) but is similar to maximal coordinate approaches in all aspects that are not related to the spanning tree[30]. A detailed description of CRBA-based procedure utilizing this technique is presented in Chap. 6.

### 5.3.4 Articulated Body Algorithm

The Articulated Body Algorithm (ABA) has been first introduced by Featherstone in [34], but the more general and efficient version presented in [36] is the one he is commonly recognized for. ABA is a standalone forward dynamics algorithm, i.e. it determines system accelerations basing on its state, inertial properties, geometry and topology. Rather than using the intermediate representation of the system by its EOM, ABA determines body accelerations directly one by one, in an iterative manner.

#### 5.3.4A Articulated body

An articulated body (AB) is a collection of *jointed* rigid bodies (i.e. the joints between the constituent bodies are *not* ignored). AB interacts with the rest of the system through one of its bodies called the handle (of this AB). In order to define what AB inertia is, let us express the acceleration of the handle body, $\widehat{\boldsymbol{a}}_H$, using the joint-space EOM:

$$\begin{aligned}\widehat{\boldsymbol{a}}_H &= \boldsymbol{J}_H\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_H\dot{\boldsymbol{q}} = \\ &= \boldsymbol{J}_H\boldsymbol{H}^{-1}(\boldsymbol{\tau} - \boldsymbol{C}) + \dot{\boldsymbol{J}}_H\dot{\boldsymbol{q}} = \\ &= \boldsymbol{J}_H\boldsymbol{H}^{-1}(\boldsymbol{J}_H^T\widehat{\boldsymbol{f}}_H - \boldsymbol{C}) + \dot{\boldsymbol{J}}_H\dot{\boldsymbol{q}}\end{aligned} \tag{5.65}$$

where $\boldsymbol{J}_H$ is the Jacobian of the handle and $\widehat{\boldsymbol{f}}_H$ is the spatial force acting on it. By grouping the terms in the above equation we arrive at:

$$\widehat{\boldsymbol{a}}_H = \widehat{\boldsymbol{\Phi}}_H^A\widehat{\boldsymbol{f}}_H + \widehat{\boldsymbol{b}}_H^A, \tag{5.66}$$

where the inverse inertia of the AB (having body $H$ as a handle) $\widehat{\boldsymbol{\Phi}}_H^A$ and its bias acceleration term $\widehat{\boldsymbol{b}}_H^A$ are:

$$\widehat{\boldsymbol{\Phi}}_H^A = \boldsymbol{J}_H\boldsymbol{H}^{-1}\boldsymbol{J}_H^T \qquad \text{and} \qquad \widehat{\boldsymbol{b}}_H^A = \dot{\boldsymbol{J}}_H\dot{\boldsymbol{q}} - \boldsymbol{H}^{-1} - \boldsymbol{C}), \tag{5.67}$$

which finally provides us with the definition of the articulated body inertia:

$$\widehat{\boldsymbol{I}}_H^A = (\boldsymbol{J}_H\boldsymbol{H}^{-1}\boldsymbol{J}_H^T)^{-1}. \tag{5.68}$$

[30] These are mainly: determination of inertia/mass matrix and body/constraint Jacobians.

Naturally, while $\widehat{\boldsymbol{\Phi}}_H^A$ always exists, $\widehat{\boldsymbol{I}}_H^A$ can be determined only if the handle is free to move in space (six DoFs of motion freedom)[31]. Notice that the articulated body inertia has the form of the operational space inertia matrix [67, 41].

### 5.3.4B The FD algorithm

The AB algorithm (in its standard form) applies the so called *assembly method*: it considers successive articulated bodies within the hierarchy created by any subtree of the overall kinematic tree, starting with the leaves (which form trivial articulated bodies) and then using the already processed links to assemble the ABs containing more and more individual bodies; roots of these subtrees are chosen as corresponding ABs' handles. This process is illustrated in Fig. 5.2. This step-by-step construction of articulated bodies and their inertias allows for $O(n)$ dynamics algorithm under the following assumptions:

- every AB has only one handle
- connection of one AB to another requires a single joint only
- articulated-body handles' motion is unrestricted (6 DoFs)

which taken together mean that every AB created to solve the FD problem is a floating kinematic tree [40].

Since the derivation of the assembly formulae is quite tedious, we shall only provide the final expressions for articulated inertia and bias force (the coordinate transforms are skipped).

The algorithm is organized into three passes briefly discussed below. Notice the similarities between the first two of them and RNEA executed for $\ddot{\boldsymbol{q}} = \boldsymbol{0}$ - they are less obvious than in the case of CRBA but still apparent.

#### Pass 1

The first pass starts at the base and goes down the tree calculating velocity-product accelerations and rigid-body (i.e. not articulated body) bias forces, which are needed in later passes:

$$\widehat{\boldsymbol{v}}_i = \widehat{\boldsymbol{v}}_{\lambda(i)} + \widehat{\boldsymbol{S}}_i \dot{\boldsymbol{q}}_i \tag{5.69}$$

$$\widehat{\boldsymbol{c}}_i = \widehat{\dot{\boldsymbol{S}}}_i \dot{\boldsymbol{q}}_i + \widehat{\boldsymbol{v}}_i \times \widehat{\boldsymbol{S}}_i \dot{\boldsymbol{q}}_i \tag{5.70}$$

$$\widehat{\boldsymbol{p}}_i = \widehat{\boldsymbol{v}}_i \times^* \widehat{\boldsymbol{I}}_i \widehat{\boldsymbol{v}}_i - \widehat{\boldsymbol{f}}_{x,i}, \tag{5.71}$$

where $\widehat{\boldsymbol{v}}_0$ is assumed to be zero and $\widehat{\dot{\boldsymbol{S}}}_i + \widehat{\boldsymbol{v}}_i \times \widehat{\boldsymbol{S}}_i$ is the derivative of $\widehat{\boldsymbol{S}}_i$ taken in a coordinate frame moving with velocity $\widehat{\boldsymbol{v}}_i$[32].

[31] Six DoFs of motion freedom imply that $\widehat{\boldsymbol{\Phi}}_H^A$ is invertible.

[32] Refer to Sec. A.1 for a similar convention applied to non-spatial vectors.

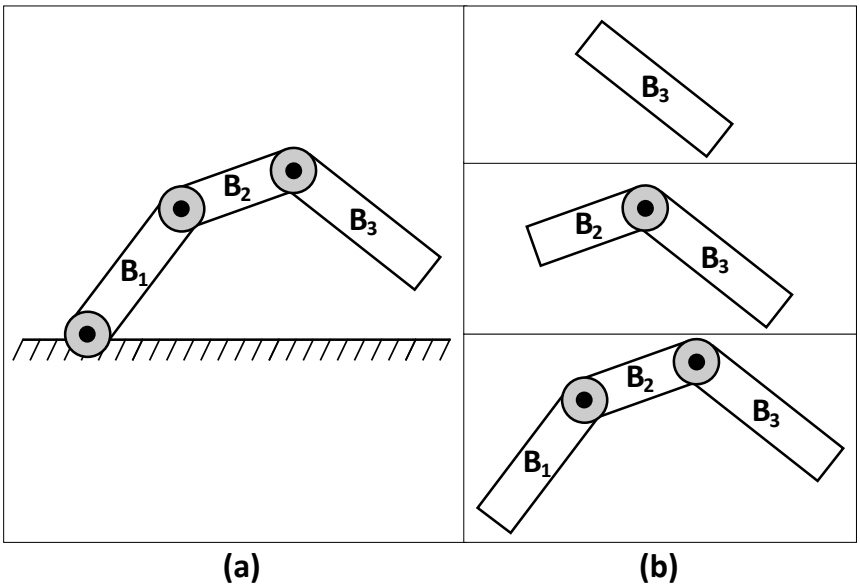



Figure 5.2: Illustration of the articulated-body assembly process: (a) the entire system; (b) three successive (top to bottom) subtrees/articulated bodies considered by the assembly process; notice that the root joint of each subtree is *not* included into the corresponding articulated body which makes it a floating kinematic tree

**Pass 2**

The second pass travels from the tree leaves up to the base and calculates articulated-body inertias and bias forces (assembly method), like so:

$$\widehat{\boldsymbol{I}}_i^A = \widehat{\boldsymbol{I}}_i + \sum_{j \in \mu(i)} \widehat{\boldsymbol{I}}_j^a \tag{5.72}$$

$$\widehat{\boldsymbol{p}}_i^A = \widehat{\boldsymbol{p}}_i + \sum_{j \in \mu(i)} \widehat{\boldsymbol{p}}_j^a \tag{5.73}$$

$$\widehat{\boldsymbol{I}}_j^a = \widehat{\boldsymbol{I}}_j^A - \widehat{\boldsymbol{I}}_j^A \widehat{\boldsymbol{S}}_j (\widehat{\boldsymbol{S}}_j^T \widehat{\boldsymbol{I}}_j^A \widehat{\boldsymbol{S}}_j)^{-1} \widehat{\boldsymbol{S}}_j^T \widehat{\boldsymbol{I}}_j^A \tag{5.74}$$

$$\widehat{\boldsymbol{p}}_j^a = \widehat{\boldsymbol{p}}_j^A + \widehat{\boldsymbol{I}}_j^a \widehat{\boldsymbol{c}}_j + \widehat{\boldsymbol{I}}_j^A \widehat{\boldsymbol{S}}_j (\widehat{\boldsymbol{S}}_j^T \widehat{\boldsymbol{I}}_j^A \widehat{\boldsymbol{S}}_j)^{-1} (\boldsymbol{\tau}_j - \widehat{\boldsymbol{S}}_j^T \widehat{\boldsymbol{p}}_j^A) \tag{5.75}$$

**Pass 3**

The final pass calculates joint-space and spatial articulated body handles (and thus tree bodies) accelerations, starting at the base and going toward the leaves:

$$\ddot{\boldsymbol{q}}_i = (\widehat{\boldsymbol{S}}_i^T \widehat{\boldsymbol{I}}_i^A \widehat{\boldsymbol{S}}_i)^{-1} (\boldsymbol{\tau}_i - \widehat{\boldsymbol{S}}_i^T \widehat{\boldsymbol{I}}_i^A (\widehat{\boldsymbol{a}}_{\lambda(i)} + \widehat{\boldsymbol{c}}_i) - \widehat{\boldsymbol{S}}_i^T \widehat{\boldsymbol{p}}_i^A) \tag{5.76}$$

$$\widehat{\boldsymbol{a}}_i = \widehat{\boldsymbol{a}}_{\lambda(i)} + \widehat{\boldsymbol{c}}_i + \widehat{\boldsymbol{S}}_i \ddot{\boldsymbol{q}}_i, \tag{5.77}$$

with $\widehat{\boldsymbol{a}}_0$ is assumed to be the negated gravity vector (if gravitational acceleration is considered).

#### 5.3.4C Beyond kinematic trees

The assembly method described in the previous section is but one of FD algorithms utilizing the concept of articulated bodies. Its obvious advantage is the $O(n)$ time complexity, but the alternatives are applicable to a wider class of problems [40].

Among these methods the one that has gained some popularity in the literature (its open-loop version has been used in e.g. [106, 46, 68]) is the *Divide-and-Conquer Articulated-Body Algorithm* (DCA) first described in [37, 38].

In practice, however, implementing the basic DCA handling only unbranched, open-loop systems does not really seem worth the effort (if one has the linear-time ABA at hand) and the description of the general-case algorithm involves a lot of caveats making it harder to follow; therefore, we have decided not to discuss DCA in this work and only provide its high-level features:

- it is applicable to systems containing any number of bodies, connected into any topology by joints of any type (i.e. it supports arbitrary numbers of loops)
- it is designed for parallel processing ($O(nlog(n))$ time complexity on $O(n)$ processors)
- its accuracy deteriorates with the increase in the number of bodies
- its performance deteriorates with the increase in the density of kinematic loops

### 5.3.5 Mirtich/Kokkevis & Metaxas

Alternatively to both the EOM-based (Sec. 5.3.3C) and ABA-based (Sec. 5.3.4C) approaches one may apply a hybrid technique popularized independently by Brian Mirtich and Evaneglos Kokkevis. The idea is to utilize the basic articulated-body algorithm to build system like Eq. 4.14 instead of using CRBA as in Sec. 5.3.3C. The unified interpretation of Mirtich's and Kokkevis' algorithms is presented in the next chapter.

It is important to keep in mind that these propositions reduce to ABA for loop-free topologies and thus there contribution in this context is none; their essence is a smart application of ABA to handle collisions, contact and loop-closure constraints.

#### 5.3.5A Mirtich

Apart from proposing a free-body collision-based simulator (Sec. 5.2.6), Mirtich has also devised a way to fuse it with the efficient articulation-oriented algorithm - ABA. The overall idea remains the same, collisions are handled one at a time and the impulses are determined from the basic relation in Eq. 5.27, but the matrix $\boldsymbol{M}$ is calculated differently.

Mirtich has modified/simplified ABA to support impact dynamics by dropping all the finite-force terms which vanish in the limit as the collision duration tends to zero. The collision matrix is determined column by column by consecutively applying test impulses, which are the basis vectors for the collision frame. The procedure for the $i^{th}$ column is:

- at the collision point, apply the $i^{th}$ basis vector of the collision frame as the test impulse
- run the modified ABA to determine the velocity response of the system at the collision point, $\boldsymbol{\Delta u}$
- fill the $i^{th}$ column of $\boldsymbol{M}$ with $\boldsymbol{\Delta u}$

#### 5.3.5B Kokkevis & Metaxas

Kokkevis has also used a modified version of ABA, but to impose acceleration constraints. The formulation was based upon the fact that at any time instant the generalized force applied to an articulated system has a net effect on its joint and body accelerations which is proportional to the magnitude of the applied generalized force. This formulation was based on one-dimensional (scalar) acceleration-force relations of the form :

$$h(\ddot{\boldsymbol{q}}^f) - h(\ddot{\boldsymbol{q}}^0) = kf, \tag{5.78}$$

where $\ddot{\boldsymbol{q}}^f$ and $\ddot{\boldsymbol{q}}^0$ are the vectors of joint accelerations *before* and *after* the force is applied, $f$ is the magnitude of force, $k$ is a scalar constant and $h$ is a linear scalar function of joint accelerations. In order to use Equation 5.78 to find (constraint) forces needed to generate a desired acceleration change, we first need to determine $k$. Kokkevis proposed to first evaluate the value of $h(\ddot{\boldsymbol{q}}^0)$ by running ABA with no force acting, then apply any known scalar *test force* $f_t$ at the point of interest in the system and run ABA again to evaluate the resultant value of $h(\ddot{\boldsymbol{q}}^f)$ to finally calculate $k$ from:

$$k = \frac{1}{f^t}(h(\ddot{\boldsymbol{q}}^t) - h(\ddot{\boldsymbol{q}}^0)) \tag{5.79}$$

To make the approach support any number $m$ of constraints, all the one-dimensional force-acceleration relations are stacked into a single system:

$$\boldsymbol{h}^f - \boldsymbol{h}^0 = \boldsymbol{K}\boldsymbol{f}, \tag{5.80}$$

where $\boldsymbol{h}^0$, $\boldsymbol{h}^f$ and $\boldsymbol{f}$ are $m \times 1$ vectors and $\boldsymbol{K}$ is an $m \times m$ matrix, called a *constraint system matrix*. Entries of $\boldsymbol{K}$ are determined by using the same test-force procedure, only now it needs to be performed for each constraint (i.e. $m$ times); after each such run, all the entries of the corresponding column of $\boldsymbol{K}$ are evaluated by applying Equation 5.79 for each scalar function $h$ (i.e. $m$ times).

### 5.3.6 Baraff

One of the most commonly cited representatives of the maximal-coordinate formulations is David Baraff's 1996 paper [8]. The general method applied by Baraff is the index reduction (Sec. 4.3.3) to obtain the descriptor form of system EOMs. The drift is compensated using Baumgarte's stabilization.

The hallmarks of Baraff's proposition are:

- linear-time method for handling bilateral, pairwise constraints forming acyclic sets (called *primary constraints*) exploiting the sparsity of the corresponding descriptor form EOM
- method for handling loop-closure, unilateral and multibody constraints (called *auxiliary* constraints) build on top of the main procedure

An interesting observation is to note that the discrimination into primary and secondary constraints is similar to the techniques of extracting a spanning tree from the overall system (see Sec. 5.3.3C).

#### 5.3.6A Handling primary constraints

The primary constraints are handled using the descriptor form of the system EOM (Eq. 4.13) in a slightly modified form:

$$\begin{bmatrix} \boldsymbol{M} & -\boldsymbol{J}^T \\ -\boldsymbol{J} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{b} \end{bmatrix}. \tag{5.81}$$

Obviously, the fact of applying the above equation was nothing new; Baraff's contribution in this aspect was to prove that relatively simple sparse factorization and solving procedures are enough to exploit the sparsity of the coefficient matrix in Eq. 5.81 and produce a simulation algorithm linear in the number of bodies. It made the Lagrange multipliers approach comparable to the more complex and less flexible reduce-coordinate algorithms in terms of the asymptotic time complexity.

It is worth noting that the general approach of creating larger EOMs and sacrificing certain desirable properties (like the positive-definiteness of the $\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{J}^T$) to exploit the sparsity of alternative formulations, gained popularity quite fast and has been becoming more and more justified since then along with the increasing significance of GPGPU computing.

#### 5.3.6B Handling auxiliary constraints

Baraff's calls his procedure for handling auxiliary constraints the *constraint anticipation*. If a system is assumed to be subject to a set of primary constraints, then any force acting on it *can* lead to accelerations violating these constraints and will thus make them generate appropriate reaction forces which will in turn make eliminate the invalid acceleration components; this obviously includes reaction forces of the auxiliary constraints. That modified response will obviously influence the system (inverse) inertia matrix since it now describes an articulated system rather than a collection of independent bodies.

Let $\boldsymbol{k}_i$ be the direction of the $i^{th}$ auxiliary constraint reaction force and $\mu_i$ it magnitude/unknown multiplier. Matrix $\boldsymbol{K}$ is defined by:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}_1 & \boldsymbol{k}_2 & \cdots & \boldsymbol{k}_k \end{bmatrix}, \tag{5.82}$$

where $k$ is the number of auxiliary one-dimensional constraints[33]. If we place all the multipliers, $\mu_i$, into a single vector $\boldsymbol{\mu}$ then the total force due to all the auxiliary constraints is $\boldsymbol{K}\boldsymbol{\mu}$.

##### Tentative accelerations

First, tentative accelerations are evaluated, i.e. the accelerations in the absence of any auxiliary constraints. This step is equivalent to solving the system subject to primary constraints only so its cost is $O(n)$.

##### Modified inverse inertia matrix

In order to determine the auxiliary constraint's reaction forces a system like 5.81 needs to be solved

$$\begin{bmatrix} \boldsymbol{M}_M & -\boldsymbol{K} \\ -\boldsymbol{J}_a & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{b} \end{bmatrix}, \tag{5.83}$$

where $\boldsymbol{M}_M$ denotes the modified inertia matrix of the system. The problem is, the coefficient matrix of the above system is not guaranteed to be sparse which eliminates the only benefit of this formulation and it is wiser to solve it using the dense equivalent which requires determining the matrix $\boldsymbol{J}_a\boldsymbol{M}_M^{-1}\boldsymbol{K}$ (see 4.14).

Baraff proposed to calculate $\boldsymbol{J}_a\boldsymbol{M}_M^{-1}\boldsymbol{K}$ by determining $\boldsymbol{M}_M^{-1}\boldsymbol{K}$ column by column and multiplying the resulting matrix by $\boldsymbol{J}_a$. The procedure is as follows:

- for each auxiliary constraint/column $j$
  1. determine $\boldsymbol{F}_j^{resp}$ - the response of the primary constraints to $\boldsymbol{k}_j$
  2. fill the $j^{th}$ column of the matrix $\boldsymbol{M}_M^{-1}\boldsymbol{K}$ with $\boldsymbol{M}^{-1}(\boldsymbol{F}_j^{resp} + \boldsymbol{k}_j)$

[33] Should all the auxiliary constraint be workless then $\boldsymbol{K} = \boldsymbol{J}_a^T$, where $\boldsymbol{J}_a$ is the stacked Jacobian of the auxiliary constraints, but Baraff made no such assumption

As we can see, the $j^{th}$ column of $\boldsymbol{M}_M^{-1}\boldsymbol{K}$ is the acceleration response of the system subject to primary constraints when acted upon by $\boldsymbol{k}_j$. Now, since each evaluation of point 1. in the above algorithm requires a single solution of the sparse system in Eq. 5.81, its asymptotic cost is $O(nk)$. The eventual multiplication of the resulting matrix by $\boldsymbol{J}_a$ is an $O(nk^2)$ operation if $\boldsymbol{J}_a$ is dense and $O(k^2)$ otherwise.

Once the coefficient matrix $\boldsymbol{J}_a\boldsymbol{M}_M^{-1}\boldsymbol{K}$ has been determined, it is used to to determine the multipliers $\boldsymbol{\mu}$. The exact method applied depends on the nature of the auxiliary constraints, i.e. if the resulting system is a SOLE (additional joints, loop-closure constraints) or rather an LCP (contact and friction). For the purpose of estimating the overall time-complexity, Baraff summarizes this step as an $O(n^3)$ operation.

#### Net accelerations

Once the total auxiliary constraints reaction force, $\boldsymbol{K}\boldsymbol{\mu}$, is known the system 5.81 is solved one last time to determine the net accelerations of a system subject to primary, auxiliary reaction and external forces.

### 5.3.7 Bender/Weinstein

As it has been already discussed in Sec. 5.2.7 Bender (and Weinstein) enforces contact constraints by predicting unconstrained system configuration in order to determine the interpenetration at the end of the time-step. The same approach can be generalized to joint-constraints, the only difference is the evaluation of the configuration error which will obviously differ between joint types [13, 15, 11, 126].

# Chapter 6

# Description of the system and its constraints

## 6.1 Overview

The current chapter introduces the representation the constraints, form of the system equations of motion and the the way its coefficients can be determined sing *test-stimuli methods.* along with a description of a popular method of solving it. This notions form a basis for the forthcoming chapters which introduce, discuss and test specific forward dynamics algorithms.

### 6.1.1 Terms and symbols

- additional constraint: a constraint that is not included in the definition of a kinematic tree; it can be any type of constraint but for performance reasons we propose them as means to introduce constraints that are problematic to express by an explicit constraint (*a joint*)
- underconstrained system: the system before any additional constraint is applied (i.e. less constrained or unconstrained in the extreme[1])
- $\dot{\boldsymbol{q}}_o$, $\ddot{\boldsymbol{q}}_o$: joint-space velocity and acceleration vectors of the underconstrained system
- $n$: underconstrained system degrees of freedom number
- constraint dimension: the number of system degrees of freedom a specific constraint restricts

[1] The system is underconstrained if the generalized coordinates used chosen for its description introduce any constraints by themselves (reduced coordinates) but further ones must be applied. Otherwise - the system is obviously unconstrained.

- $n_c$: the cumulative dimension of all additional constraints acting in the system
- constraint coordinate frame: a Cartesian frame associated with each constraint; the exact meaning of its origin and orientation depends on the constraint type

## 6.2 Equation of Motion

The basis for the described simulator are the methods of extending CRBA-based open-loop simulation algorithm to handle loops, which has been described in [40] and briefly discussed in Sec. 5.3.3C. Therefore, as it has been already presented in Sec. 4.3.3, the constraint equations are differentiated w.r.t. time until they can be posed at the acceleration level and appended to the EOM which after trivial substitutions becomes:

$$\boldsymbol{A}\boldsymbol{f} = \boldsymbol{b}, \qquad \boldsymbol{A} = \boldsymbol{J}_c\boldsymbol{H}^{-1}\boldsymbol{J}_c^T, \qquad \boldsymbol{b} = -\boldsymbol{J}_c\ddot{\boldsymbol{q}}_o - \dot{\boldsymbol{J}}_c\dot{\boldsymbol{q}}, \tag{6.1}$$

where $\boldsymbol{f}$ is a vector of additional constraints' reaction forces, $\boldsymbol{b}$ is the change in the constrained motion attributes that the forces $\boldsymbol{f}$ are expected to cause. $\boldsymbol{A}$ characterizes the constraints and related inertial properties of the system and is closely related to the inverse of operational space inertia matrix [67]. At the impulse-velocity level, Eq. 6.1 becomes:

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b}, \qquad \boldsymbol{A} = \boldsymbol{J}_c\boldsymbol{H}^{-1}\boldsymbol{J}_c^T, \qquad \boldsymbol{b} = -\boldsymbol{J}_c\dot{\boldsymbol{q}}_o, \tag{6.2}$$

where $\boldsymbol{p}$ is the vector additional constraints' reaction impulses.

However, since our simulator is not only required to handle bilateral constraints but also contact/collision, we need to be able to combine constraint equations with contact complementarity conditions and a popular way of achieving this is by casting the Eq. 6.1 as a Mixed Linear Complementarity Problem (MLCP) stated as follows: find $\boldsymbol{a}$ and $\boldsymbol{f}$ satisfying:

$$\boldsymbol{a} = \begin{bmatrix}\boldsymbol{a}_u \\ \boldsymbol{a}_b\end{bmatrix} = \begin{bmatrix}\boldsymbol{A}_{uu} & \boldsymbol{A}_{ub} \\ \boldsymbol{A}_{bu} & \boldsymbol{A}_{bb}\end{bmatrix}\begin{bmatrix}\boldsymbol{f}_u \\ \boldsymbol{f}_b\end{bmatrix} - \begin{bmatrix}\boldsymbol{b}_u \\ \boldsymbol{b}_b\end{bmatrix} = \boldsymbol{A}\boldsymbol{f} - \boldsymbol{b}, \tag{6.3a}$$

$$\boldsymbol{0} \leq \boldsymbol{a}_u \perp \boldsymbol{f}_u \geq \boldsymbol{0} \quad \text{and} \quad \boldsymbol{a}_b = \boldsymbol{0}, \tag{6.3b}$$

where the vectors and matrices have the same meaning as their equivalents in the previous formulation; subscript $u$ indicates the unilateral portion of the constraints, while $b$ the bilateral one. The general algebraic form is the same as in Eq. 6.1 but the differently posed conditions require altered solution methods and additional bookkeeping. The relation at between reaction impulses and velocities

is identical (definition of $\boldsymbol{b}$ is more general than in Eq. 6.2 to enable post-collision bounce):

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_u \\ \boldsymbol{v}_b \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{uu} & \boldsymbol{A}_{ub} \\ \boldsymbol{A}_{bu} & \boldsymbol{A}_{bb} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_u \\ \boldsymbol{p}_b \end{bmatrix} - \begin{bmatrix} \boldsymbol{b}_u \\ \boldsymbol{b}_b \end{bmatrix} = \boldsymbol{A}\boldsymbol{p} - \boldsymbol{b}, \tag{6.4a}$$

$$\mathbf{0} \leq \boldsymbol{v}_u \perp \boldsymbol{p}_u \geq \mathbf{0} \qquad \boldsymbol{v}_b = \mathbf{0}. \tag{6.4b}$$

### 6.2.1 MLCP solver

We shall now describe an iterative approach to solving the MLCP in Equations 6.3 and 6.4 since iterative schemes are a very popular choice in interactive simulation [70, 76, 33, 119], due to:

- their simplicity when compared to exact methods
- the possibility to trade off execution-time for accuracy
- the possibility to reuse the previous solution as the current-step staring point (warm-starting)

Obviously, iterative methods can converge slowly to a high-precision solution, but it is not really the main concern in the case of animation where one expects plausible solutions to be determined at interactive rates. Moreover, the numerical integration and limited precision of floating-point operations introduce errors themselves so paying the price for highly precise (M)LCP solver[2] would not guarantee a substantial improvement over the iterative one [33].

Among stationary iterative methods designed to solve SOLEs, the three most commonly met and discussed are:

- Jacobi's method
- Gauss-Seidel method (GS)
- Successive over-relaxation method (SOR)

Out of the three, Jacobi's method is known to exhibit the poorest convergence and the additional computational effort of SOR does not really introduce much advantage over GS. Therefore, we decided to use the Gauss-Seidel method - a common choice in this field [33, 119].

[2]These are mainly pivoting methods among which the one by Lemke is still considered the most robust in the case of problems including frictional contact [76].

#### 6.2.1A Projected Gauss-Seidel method (PGS)

In the case of iterative methods, the matrix $\boldsymbol{A}$ is decomposed into a sum of three:

$$\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}, \tag{6.5}$$

where matrices $\boldsymbol{L}$, $\boldsymbol{D}$ and $\boldsymbol{U}$ are strictly lower triangular, diagonal, and strictly upper triangular, respectively.

If we apply the above decomposition to a coefficient matrix of a sample SOLE in Eq. 6.1, the Gauss-Seidel iterative update of the $i^{th}$ entry of the unknown vector, $\boldsymbol{f}$ is:

$$\boldsymbol{f}[i]^{k+1} = \frac{\boldsymbol{b}[i] - \sum_{j<i} \boldsymbol{L}[i,j]\boldsymbol{f}[j]^{k+1} - \sum_{j>i} \boldsymbol{U}[i,j]\boldsymbol{f}[j]^{k}}{\boldsymbol{A}[i,i]}, \tag{6.6}$$

where the superscript denotes the iteration.

Naturally, we are not really aiming at SOLE but the MLCP in Eq. 6.3; the necessary modification is to project the current approximation of $\boldsymbol{x}[i]$ into the desired range:

$$\boldsymbol{f}[i]^{k+1} = \mathrm{clp}_i\left(\frac{\boldsymbol{b}[i] - \sum_{j<i} \boldsymbol{L}[i,j]\boldsymbol{f}[j]^{k+1} - \sum_{j>i} \boldsymbol{U}[i,j]\boldsymbol{f}[j]^{k}}{\boldsymbol{A}[i,i]}\right), \tag{6.7}$$

where $\mathrm{clp}_i(s)$ clamps the scalar $s$ into the range defined for the $i^{th}$ entry of $\boldsymbol{f}$. Gauss-Seidel method with clamping is commonly known as the Projected Gauss-Seidel method (PGS). Note that the projection operation does not explicitly enforce the complementarity condition, it only ensures that forces are within the desired ranges (which in most scenarios is limited to keeping the contact forces from being attractive). What is more, since (P)GS does not really determine accelerations (unlike e.g. Baraff's LCP solver, see Sec. 5.2.2), vector $\boldsymbol{a}$ needs to be separately evaluated from Eq. 6.3a.

Finally, to cover a broader range of constraints within a single formulation, we introduce an additional acceleration-biasing term, $\boldsymbol{c} = [\boldsymbol{c}_u^T\, \boldsymbol{c}_b^T]^T$ so that the system Eq. 6.3a becomes: $\boldsymbol{a} = \boldsymbol{A}\boldsymbol{f} - \boldsymbol{b} - \boldsymbol{c}$. The same applies to the velocity-level formulation. Vector $\boldsymbol{c}$ can be perceived as the desired constraint-space acceleration. For any bilateral constraint using the name *desired* can be taken literally - the constraining force can take any value to make the motion attribute take the corresponding desired value (i.e. $\boldsymbol{a}_b = \boldsymbol{c}_b$). In the case of unilateral constraints $\boldsymbol{c}$ holds lower bounds on the desired values of motion attributes (i.e. $\boldsymbol{a}_u \geq \boldsymbol{c}_u$). The final version of the iterative update is:

$$\boldsymbol{f}[i]^{k+1} = \mathrm{clp}_i\left(\frac{\boldsymbol{b}[i] + \boldsymbol{c}[i] - \sum_{j<i} \boldsymbol{L}[i,j]\boldsymbol{f}[j]^{k+1} - \sum_{j>i} \boldsymbol{U}[i,j]\boldsymbol{f}[j]^{k}}{\boldsymbol{A}[i,i]}\right). \tag{6.8}$$

Obviously, the evaluation of $\boldsymbol{a}$ is performed using the unmodified system in Eq. 6.3a. To avoid introducing additional algebraically-redundant terms into the equations, vector $\boldsymbol{c}$ is dropped in the further analysis.

```
procedure PGS()
begin
  A := the coefficient matrix
  b := the RHS vector
  n := system dimension
  x : the vector of unknowns
  s : auxiliary variable

  x = initialGuess()

  while stop criteria not met do

    for i = 1 to n do

      h := upper bound of x[i]
      l := lower bound of x[i]

      s = A[i, 1 → i − 1] x[1 → i − 1] + A[i, i + 1 → n] x[i + 1 → n]

      x[i] = (b[i] − s) / A[i, i]

      if x[i] < l then
        x[i] = l
      elseif x[i] > h then
        x[i] = h
      end

    end
  end
end
```

Procedure 6.1: Projected Gauss-Seidel launched for a sample SOLE

## 6.3 Representation of additional constraints

### 6.3.0B Constraining force bases

The single most important piece of information about each additional constraint are the *dimensions* in $F^6$ in which reaction forces can act and, at the same time, the *dimensions* of motion in $M^6$ that they restrain. More formally, we need means to define a constraining force subspace, $T$, and (indirectly) its orthogonal complement - valid motion subspace, $S$. This is done using matrix $\widehat{\boldsymbol{T}}$ whose columns span $T$.

For $\widehat{\boldsymbol{T}}$ to have a simple and intuitive form, it is necessary to provide a properly placed and oriented constraint coordinate frame and express the matrix in constraint-local coordinates. The matrices representing constraining force bases will be useful in two basic situations. Firstly, to represent constraining force vectors in $F^6$ in order to obtain equivalent spatial force vectors:

$$\widehat{\boldsymbol{f}} = \widehat{\boldsymbol{T}}_k \boldsymbol{f}_k, \tag{6.9}$$

where $k$ identifies one of the additional constraints, $\boldsymbol{f}_k$ and $\widehat{\boldsymbol{f}}_k$ are the coordinate vectors representing the constraining force in the constraining force subspace and $F^6$ bases, respectively. Secondly, to *extract* the *forbidden amount* of motion from an arbitrary spatial motion vector or, in other words, to project spatial motion vectors from $M^6$ to the constrained motion subspace:

$$\boldsymbol{v}_k = \widehat{\boldsymbol{T}}_k^T \widehat{\boldsymbol{v}} \quad \text{and} \quad \boldsymbol{a}_k = \widehat{\boldsymbol{T}}_k^T \widehat{\boldsymbol{a}} + \dot{\widehat{\boldsymbol{T}}}_k^T \widehat{\boldsymbol{v}}, \tag{6.10}$$

where $\boldsymbol{v}_k$ and $\widehat{\boldsymbol{v}}$ are coordinate vectors representing velocity in the constrained motion subspace and $M^6$, respectively. The same applies to accelerations: $\boldsymbol{a}_k$ and $\widehat{\boldsymbol{a}}$. This operation, which is a projection followed by a change of basis, reveals an assumption that the bases for each constraining force subspace are orthonormal, more general cases are beyond the scope of this work.

#### 6.3.0C Constrained motion attribute mapping function

Apart from $\widehat{\boldsymbol{T}}$, each constraint type also defines how the underconstrained system motion attributes that it restrains are supposed to *change* after it is applied. We propose to represent this change by a function mapping the underconstrained system motion attribute value to its desired value. If the specified mapping defines ranges rather than exact value, the corresponding constraint is unilateral, otherwise - it is bilateral.

Frictionless non-penetration constraint is a simple yet illustrative example of what this actually means. It restricts only the normal relative motion of the contacting bodies. The constrained acceleration is supposed to become non-negative; in the case of velocity (impact) we may want the bodies to bounce off each other and thus define the mapping function to return a specific departing velocity determined using e.g. a coefficient of restitution or a more advanced *collision integration* technique like in [87].

### 6.3.1 Combining multiple constraints

The main goal of this subsection is to introduce a notion of a *constraint space*, but is it impossible without first presenting constraint Jacobians and stacking.

#### 6.3.1A Body and constraint Jacobians

The $i^{th}$ body Jacobian, $\boldsymbol{J}_i$, determines the $M^6$ velocity of this body:

$$\widehat{\boldsymbol{v}}_i = \boldsymbol{J}_i \dot{\boldsymbol{q}}, \tag{6.11}$$

where $\widehat{\boldsymbol{v}}_i$ is $M^6$ velocity vector of body $i$ expressed at the origin of its coordinate frame. Determining $\boldsymbol{J}_i$ requires traversing the chain starting at body $i$ up the hierarchy till the base, which implies $O(n)$ worst case time complexity. Pseudocode illustrating this procedure is presented in Proc. 6.2.

```
procedure determineBodyJacobian()
begin
  i := index of the body
  a := system DoF offset of the i^th body
  b := DoF count of the i^th body
  c : auxiliary variable
  J_i : Jacobian of the i^th body

  c = a + b
  J_i = 0_{6×6}
  J_i[*, a → c] = Ŝ_i

  X̂ : tranformation matrix from the currently processed to this body's frame
  X̂ = 1_{6×6}
  j = i

  while the j^th body has a parent do
    d := index of the parent of the j^th body
    ^jX̂_d := parent-to-child tranformation matrix of the j^th body

    a = system DoF offset of the j^th body
    b = DoF count of the j^th body
    c = a + b

    X̂ = X̂ ^jX̂_d
    J_i[*, a → c] = X̂ Ŝ_d
  end
end
```

Procedure 6.2: Body Jacobian determination

The $k^{th}$ constraint Jacobian, $\boldsymbol{J}_{c,k}$, is similar to a body Jacobian but it is associated with the attributes of motion the $k^{th}$ constraint restricts, $\boldsymbol{v}_k$ or $\boldsymbol{a}_k$:

$$\boldsymbol{v}_k = \boldsymbol{J}_{c,k}\dot{\boldsymbol{q}} \quad \text{and} \quad \boldsymbol{a}_k = \boldsymbol{J}_{c,k}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{c,k}\dot{\boldsymbol{q}}, \tag{6.12}$$

In order to determine $\boldsymbol{J}_{c,k}$, we need the body Jacobians for the two constrained bodies, $i$ and $j$ (unless it is a single body constraint):

$$\boldsymbol{J}_{c,k} = \widehat{\boldsymbol{T}}_k^T(\boldsymbol{J}_i - \boldsymbol{J}_j), \tag{6.13}$$

which for the total of $m$ simultaneous constraints yields $2mO(n)$ worst-case time complexity. However, in most cases a single body will be constrained multiple times so simple caching of the already determined body Jacobians will save a lot of computation and heavily improve performance.

### 6.3.2 Constraint space

The hereby introduced notion of *constraint space* is a generalization of *contact space* formulation proposed in [109]. An element of the constraint space is a

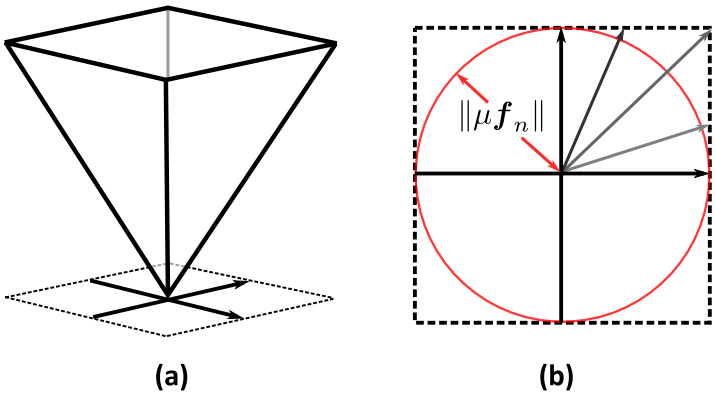



Figure 6.1: Four-facet friction pyramid (a); base of the pyramid with the ideal upper-bound on the friction force magnitude compared to force vectors allowed by the approximate model (b)

concatenation of constrained motion attributes or constraining forces from all individual additional constraints. Similarly, the constraint Jacobian for the entire system, $\boldsymbol{J}_c$, is a result of *stacking* the individual constraint Jacobians. Therefore, constraint space velocity, $\boldsymbol{v}$, and acceleration, $\boldsymbol{a}$, of the system, can be expressed as:

$$\boldsymbol{v} = \boldsymbol{J}_c \dot{\boldsymbol{q}} \quad \text{and} \quad \boldsymbol{a} = \boldsymbol{J}_c \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_c \dot{\boldsymbol{q}}, \tag{6.14}$$

and by the principle of virtual work/power:

$$\boldsymbol{\iota} = \boldsymbol{J}_c^T \boldsymbol{p} \quad \text{and} \quad \boldsymbol{\tau} = \boldsymbol{J}_c^T \boldsymbol{f}, \tag{6.15}$$

where $\boldsymbol{p}/\boldsymbol{f}$ are the constraint-space reaction impulse/force, while $\boldsymbol{\iota}/\boldsymbol{\tau}$ are their joint-space equivalents.

## 6.4 Friction

### 6.4.1 Forces

Since we are not really focused on precise frictional contact simulation, friction is introduced using one of the coarsest approximations of the Coulomb's law, which is however usually enough for visual plausibility:

- friction cone is approximated by a four-facet pyramid (Fig. 6.1a), which means that the approximation of the upper limit for friction force magnitude[3] is quite rough and the actual magnitude will often be allowed to be greater than the upper bound (Fig. 6.1b)
- there is no explicit distinction between static and dynamic friction, nor the corresponding coefficients

[3] Friction coefficient times the normal contact force magnitude.

- friction forces act so as to make the relative acceleration vanish, no matter the value of the slip velocity

### 6.4.2 Impulses

When resolving collisions or velocity-level constraints, friction is handled using the model applied at the force-acceleration level, only now it is realized by impulses (Sec. 4.4.2) which act so as to make the slip velocity vanish, while not exceeding their upper limit[4].

## 6.5 Building the equation

The main focus of this section are the ways of building the system in Eq. 6.3 either by utilizing constraint Jacobians and system generalized inertia matrix explicitly or by making use of efficient recursive methods in order to determine system response to a set of test impulses/forces. Both of these approaches are represented in the literature: the former has been applied by [79, 109] while the latter by [72, 70, 87].

### 6.5.1 Vectors $b$ and $c$

Vector $\boldsymbol{b}$ is an element of the constraint space and holds values that the constrained motion attributes would take if no additional constraining forces acted. To determine its entries we can either use constraint Jacobian (Eq. 6.12) or recursively determine body velocities ($M^6$) and then *extract* their forbidden directions using Eq. 6.10. While underconstrained system velocities, $\dot{\boldsymbol{q}}$, are readily available in the state vector, its accelerations, $\ddot{\boldsymbol{q}}$, need to be determined using the open-loop system solver of choice.

Vector $\boldsymbol{c}$ is also an element of the constraint space and holds the desired values of constrained motion attributes.

### 6.5.2 Matrix $A$

In the Eq. 6.1 we can see that the basic formula for the matrix $\boldsymbol{A}$ is:

$$\boldsymbol{A} = \boldsymbol{J}_c \boldsymbol{H}^{-1} \boldsymbol{J}_c^T, \tag{6.16}$$

where $\boldsymbol{H}$ is the joint-space inertia matrix of the system, which is most efficiently determined using the Composite-Rigid-Body Algorithm (CRBA). One can obviously apply Eq. 6.16 directly, which requires computing $\boldsymbol{J_c}$, $\boldsymbol{H}$ along with its inverse and finally performing the multiplication. The computational complexity of the entire procedure is $O(n^3)$ due to the inversion of an $n \times n$ matrix, but other

[4] Friction coefficient times the normal *impulse* magnitude

terms will be significant as well. Still - for systems of moderate sizes this can be the best choice.

Direct inversion of $\boldsymbol{H}$ in Eq. 6.16 can be avoided by somehow exploiting the linear relationship between the applied stimulus and the resultant step-change of the constrained motion attributes, which is derived below.

Let us we move the term $\boldsymbol{b}$ in the Eq. 6.3a to the RHS:

$$\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{a} - \boldsymbol{J}_c\ddot{\boldsymbol{q}}_o - \dot{\boldsymbol{J}}_c\dot{\boldsymbol{q}} = \boldsymbol{A}\boldsymbol{f}. \tag{6.17}$$

Notice that $\boldsymbol{a} - \boldsymbol{b}$ is actually an increment of $\boldsymbol{a}$ caused by $\boldsymbol{f}$. A corresponding relation can be derived for impulses and velocity increments using the impulsive EOM of the system. Therefore, the key relations are:

$$\Delta\boldsymbol{v} = \boldsymbol{A}\boldsymbol{p} \quad \text{and} \quad \Delta\boldsymbol{a} = \boldsymbol{A}\boldsymbol{f}. \tag{6.18}$$

As discussed in Sec. 5.3.5, Mirtich [87] and Kokkevis [70] have managed to abandon Eq. 6.16 completely by exploiting the linear relation summarized in Eq. 6.18.

Although Mirtich worked exclusively with instantaneous impulses while Kokkevis operated on the acceleration level, they both filled $\boldsymbol{A}$ column by column repetitively applying appropriately placed $F^6$ *test stimuli* and determining the system response using the (slightly modified[5]) Featherstone's Articulated-Body Algorithm (ABA). So, as far as determination of $\boldsymbol{A}$ goes, these approaches are *equivalent*, which is a good thing when considering unification. Moreover, although the idea of constraint space becomes vague, the constraint representation described in Sec. 6.3 still holds. And last but not least, as the Eq. 6.16 is no longer directly used, the costly determination of matrix $\boldsymbol{H}$ and constraint Jacobians can be omitted. However, this approach has its cons as well: the tempting $O(n)$ complexity of ABA can be deceiving since a brief inspection reveals that the procedure is $n_c O(n)$ - a fact that needs to be considered when conducting simulations with large and varying numbers of simultaneous constraints. One can obviously store certain intermediate ABA results to reduce the amount of redundant computations, but there is no equivalent of Jacobian caching since the propagation of each impulse is inseparably merged with topology traversal. Furthermore, the test-stimuli approach requires special handling of constraints that are local to a specific joint (e.g. joint limits) and finally - we have found it relatively cumbersome to implement.

As an alternative, we propose to fill $\boldsymbol{A}$ column by column as well, but this time by directly solving a system of joint-space equations using the factorized inertia matrix, rather than by propagation methods. What is important is that the costly determination and factorization of $\boldsymbol{H}$ can be done *once* per integrator

[5] The modifications were non-substantial and focused on extracting the operations needlessly repeated for each constraint space dimensions and performing them a single time (e.g. determining articulated-body inertias); moreover, Mirtich dropped finite force terms.

```
1  procedure determineMatrixA()
2  begin
3    J_c := constraint-space Jacobian of the constraints
4    H_fact := factorized inertia matrix of the system
5    A : matrix A
6    Δq̇ : auxiliary vector
7
8    foreach dimension of the constraint space do
9      i := dimension index
10     Δq̇ = solveFactorized(H_fact, J_c[i,*]^T)
11     A[*,i] = J_c Δq̇
12   end
13 end
```

Procedure 6.3: Determination of matrix $\boldsymbol{A}$ using joint-space test impulses

step, since $\boldsymbol{H}$ depends on the current system configuration only. We then use this *pre-factorized* coefficient matrix to solve the joint-space equation of motion $n_c$ times efficiently. We shall refer to this procedure as *the joint-space test stimuli method*. A similar procedure has been briefly described in [79] (*A Unit Force Method*) but it considered single constraint cases only and did not use a notion equivalent to our constraint space.

Let us introduce a constraint-space test impulse by means of an $n_c$-dimensional vector $\boldsymbol{\delta}_i$ having a 1 as its $i^{th}$ element and zeros elsewhere. Basing upon Equations 6.14 and 6.16 we can write:

$$\Delta\boldsymbol{v} = \boldsymbol{J}_c\Delta\dot{\boldsymbol{q}} = \boldsymbol{J}_c\boldsymbol{H}^{-1}\boldsymbol{J}_c^T\boldsymbol{\delta}_i = \boldsymbol{A}\boldsymbol{\delta}_i. \tag{6.19}$$

Now, by dropping $\boldsymbol{J}_c$ and left-multiplying both sides by $\boldsymbol{H}$ we obtain:

$$\boldsymbol{H}\Delta\dot{\boldsymbol{q}} = \boldsymbol{J}_c^T\boldsymbol{\delta}_i, \tag{6.20}$$

which gives us the procedure for the $i^{th}$ column of $\boldsymbol{A}$: solve the above equation for $\Delta\dot{\boldsymbol{q}}$ using the $i^{th}$ row of $\boldsymbol{J}_c$ as the joint-space impulse vector. The resulting $\Delta\dot{\boldsymbol{q}}$ is joint-space so it needs to be multiplied by $\boldsymbol{J_c}$ afterwards. The procedure is illustrated by Proc. 6.3.

Performance considerations are presented below, but the obvious advantage in terms of overall functionality is that, thanks to applying joint-space rather than $F^6$ test stimuli, local constraints do *not* require special handling, which fits well into our unified framework.

Factorizing and solving $\boldsymbol{H}$ is obviously $O(n^3)$ in general, but thanks to $\boldsymbol{H}$ being symmetric, positive-definite matrix [40] we can use Cholesky decomposition. The greatest performance gain is, however, due to the fact that this is done only *once* per integrator step; evaluating each column of $\boldsymbol{A}$ requires only a forward- and back-substitution based on two triangular matrices, which asymptotically is an $O(n^2)$ operation but with a very favorable constant factor. Therefore, when

considering execution times, the basic difference between this joint-space procedure and its $F^6$ equivalents is that instead of a number of calls to a fairly fast ABA, we have a single, high-cost CRBA and factorization step followed by a number of fast substitution operations (there is obviously also the cost of determining constraint Jacobian). Although the $F^6$ version of this procedure seems preferable when comparing the asymptotic time complexities alone, the practical considerations and benchmarking often prove very much in favor of its joint-space alternative.

Basing on our earlier research presented in [43], we can say that there is definitely no single best method among the ones described in this section. Inversion-based procedure is obviously mostly dependent on the efficiency and robustness of the matrix inverting procedure but when comparing the test stimuli methods alone, the proportion between the total constraints' dimension, $n_c$, the number of degrees of freedom, $n$, of the kinematic tree is the decisive factor: for great $n$ and small $n_c$ spatial impulse methods could be a better choice but they are badly outperformed in the opposite case (Fig. 6.2) which we believe to be more common if one considers the application domain[6]. Because of that we have chosen the joint-space test-stimuli method and we shall apply it in the forthcoming chapters.

## 6.6 Summary

Although the acceleration-level simulator described in [43] produced plausible results, properly reacted to motor control and exhibited stable behavior and turned out to be competitive in terms of performance when compared to literature-based alternatives, it required fourth-order Runge-Kutta (RK4) integration scheme and relatively small time-steps to achieve that and this obviously means four solver runs per simulation step which is costly. The majority of physics engines available on the market use simple, single-step integration - symplectic/semi-implicit Euler predominantly and closely related Verlet integrator, providing robust, stable and high-performance simulations. Clearly, if the articulated procedures proposed in this work are to be (anyhow) competitive with respect to these solutions, we need a proper industry-proven basis to merge our propositions into/with.

[6] Video games are much likely to need to animate a relatively simple articulated structure undergoing a lot of contact and collision events than a large jointed system subject to just a few non-penetration constraints.

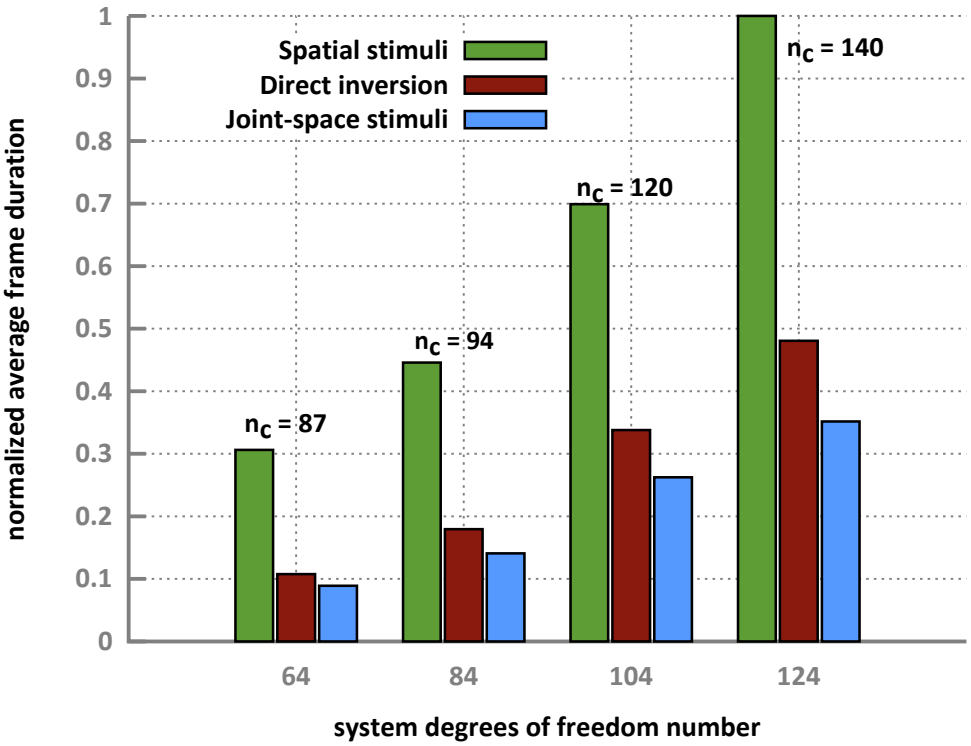



Figure 6.2: Normalized (division by the maximal value) average frame duration per method with the corresponding average constraint-space dimension, $n_c$ (based on [43])

# Chapter 7

# Sequential Impulse

## 7.1 Overview

The method popularly called the *Sequential Impulse* (SI) is known to be implemented in *Box2D* and *Bullet* physics engines. It uses impulses to solve all the constraints, without differentiating between collisions and *permanent* constraints which places it among velocity-level constraint-based methods but the way that it is actually realized makes it similar to simple collision-based simulators. SI is known for its robustness, high-performance (see Fig. 2.2). All of these factors along with the fact that there are two working open-source implementations make SI a very good candidate for the basis we need.

## 7.2 High-level view

Although the name Sequential Impulse refers to the way of processing the constraints, in this work we use this name to identify the entire simulation procedure, which in most abstract terms is realized as follows:

1. discrete collision detection[1]
2. unconstrained velocity prediction: each body independently follows its unconstrained trajectories; only gravity and explicitly given external forces are considered; the resulting acceleration are integrated into tentative velocities
3. if there are any constraints that need to be handled - go to 5; continue in 4 otherwise
4. tentative velocities are accepted as the final ones; go to 7

[1] It could be continuous (CCD) but we shall limit the analysis the the discrete case.

5. preparation for constraint processing (determination of inertial properties, constraint Jacobians and velocity error)

6. for a fixed number of iterations corrective impulses are evaluated for each constraint separately and immediately applied to the bodies (one or two) it restrains

7. final velocities are used to update the configuration; simulation proceeds - go back to 1

## 7.3 Sequential constraint processing and (P)GS

Sequential constraint processing is a method of correcting velocities one constraint-space dimension at a time. Since all these dimensions are now explicitly decoupled matrix $\boldsymbol{A}$ used in the previous section is now non-zero only on its main diagonal and there is a modified nomenclature as well:

- each diagonal entry of $\boldsymbol{A}$ is called the *inverse effective mass* for a given constraint space dimension

- what is actually used by the correcting procedure are *effective masses* (scalar inverses of the diagonal terms of $\boldsymbol{A}$)[2]

To achieve the effect of one constraint space dimension affecting others, the resulting velocity increments for the restrained bodies are shared among them so that when a specific constraint space dimension is being processed, the velocity change generated by *all* corrective impulses so far applied to the restrained bodies (one or two) is corrected towards the desired value.

If the equation 6.3 is solved using an exact method, one cannot really replace $\boldsymbol{A}$ with its diagonal version, but since we decided to use PGS it turns out that the sequential processing is a slightly different way of performing the same operations[3].

The equivalence of SI and (P)GS is easily proven; clamping and the additional bias vector can be safely ignored since they are non-substantial in the following derivations.

Let us start with the Gauss-Seidel iterative update (Eq. 6.6):

$$\boldsymbol{f}[i]^{k+1} = \frac{\boldsymbol{b}[i] - \sum_{j<i} \boldsymbol{L}[i,j]\boldsymbol{f}[j]^{k+1} - \sum_{j>i} \boldsymbol{U}[i,j]\boldsymbol{f}[j]^{k}}{\boldsymbol{A}[i,i]}, \tag{7.1}$$

[2] Actually, they are the diagonal entries of the inverse of the diagonal version of $\boldsymbol{A}$

[3] The results will not be exactly the same due to limited precision of floating-point operations, but in principle - PGS and sequential processing are the same.

extract $\boldsymbol{b}$ into a separate term:

$$\boldsymbol{f}[i]^{k+1} = \frac{\boldsymbol{b}[i]}{\boldsymbol{A}[i,i]} - \frac{\sum_{j<i} \boldsymbol{L}[i,j]\boldsymbol{f}[j]^{k+1} - \sum_{j>i} \boldsymbol{U}[i,j]\boldsymbol{f}[j]^{k}}{\boldsymbol{A}[i,i]}, \tag{7.2}$$

and make the LHS be the $k+1^{st}$ *increment* of $\boldsymbol{f}[i]$ rather than its entire value:

$$\Delta\boldsymbol{f}[i]^{k+1} = \boldsymbol{f}[i]^{k+1} - \boldsymbol{f}[i]^{k} =$$

$$= \frac{\boldsymbol{b}[i]}{\boldsymbol{A}[i,i]} - \frac{\sum_{j<i} \boldsymbol{L}[i,j]\boldsymbol{f}[j]^{k+1} - \sum_{j>i} \boldsymbol{U}[i,j]\boldsymbol{f}[j]^{k}}{\boldsymbol{A}[i,i]} - \boldsymbol{f}[i]^{k}. \tag{7.3}$$

Finally, we get:

$$\Delta\boldsymbol{f}[i]^{k+1} = \frac{\boldsymbol{b}[i]}{\boldsymbol{A}[i,i]} - \frac{\sum_{j<i} \boldsymbol{L}[i,j]\boldsymbol{f}[j]^{k+1} - \sum_{j>i} \boldsymbol{U}[i,j]\boldsymbol{f}[j]^{k} + \boldsymbol{f}[i]^{k}\boldsymbol{A}[i,i]}{\boldsymbol{A}[i,i]}, \tag{7.4}$$

where

- the left term on the RHS is the desired/target impulse in the current dimension (desired velocity increment times the inverse effective mass)
- the right term on the RHS is the overall impulse applied so far in the current dimension (the velocity increment accumulated so far due to impulses in all dimensions times the inverse effective mass)

## 7.4 Integrating SI with the simulator

For SI there is no such thing as system topology and it could be dropped entirely, but in order to integrate it with articulation-oriented simulator, we simply flattened the hierarchy so that all the bodies are root nodes.

What is more, the bodies by definition are free to move so they are all assigned 6DoF inboard joints. Furthermore, SI uses inertial global frame to describe body velocities and accelerations and we can see no reason to do otherwise[4].

Origins of body-local coordinate frame are assumed to be at body center of masses since (i) it simplifies the underlying equations and (ii) there is no gain from placing them elsewhere if to bodies are assumed to be free to move.

### 7.4.1 Velocity prediction

The procedure for determination of tentative velocities can be easily created by modifying RNEA equations with the caveat that there is no specific order in which the topology needs to be traversed since it is flat (all bodies are siblings). The modifications obviously are aimed at creating a Forward rather than Inverse Dynamics algorithm. The modified procedure is presented in Proc. 7.1.

[4]Non-inertial frames in free-body simulation introduce certain difficulties in the absence of any benefits, which becomes apparent in the next chapter (8).

```
1  procedure freeBodyFD()
2  begin
3    ĝ := gravity vector
4
5    foreach body in the system do
6      X̂ := rotation matrix of the body
7      v̂ := velocity of the body
8      Î := inertia of the body
9      f̂_x := external force acting on the body
10     f̂ : net force acting on the body
11     Φ̂ : base-frame inverse inertia of the body
12
13
14     f̂ = −v̂ × X̂^T Î X̂ v̂
15     f̂[4 → 6] = 0
16     f̂ = f̂ + f̂_x
17
18     Φ̂ = X̂^T Î^{-1} X̂
19     â = ĝ + Φ̂ f̂
20   end
21 end
```

Procedure 7.1: Forward dynamics for independent bodies

Once the 6D acceleration vector is determined for each body, it gets integrated into 6D velocity vector (Forward Euler) producing tentative velocities.

### 7.4.2 Preparation for velocity correction

The preparation step includes (for each constraint space dimension) evaluation of:

- constraint Jacobians[5]
- effective masses
- desired velocity increments

Additionally, if configuration error correction is performed - this is a good moment to configure it using the current constraint errors. We shall assume that the velocity-level Baumgarte stabilization[6] is used.

[5] Actually - individual rows of constraint Jacobians since all constraint space dimensions will be processed independently

[6] By velocity-level Baumgarte stabilization we mean using only the term proportional to the error .

```
procedure processConstraints()
begin
  foreach constraint in the system do

    T := reaction force subspace matrix of the constraint
    v̂1 := velocity of the 1st body
    v̂2 := velocity of the 2nd body
    X1/X2 := transf. from the 1st/2nd body's frame to the constraint's frame
    Φ̂1/Φ̂2 := base-frame inverse inertia of the 1st/2nd body


    foreach dimension of the constraint do
      i := dimension index
      j1/j2 : the i^th row of the constraint Jacobian w.r.t. the 1st/2nd body
      χ̂1/χ̂2 : spatial reponse to impulse j1^T/j2^T applied to the 1st/2nd body
      m : effective mass w.r.t. the dimension
      v : constraint-space velocity in the dimension
      Δdes v : desired increment of v

      j1 = T^T[i,*]X1
      j2 = −T^T[i,*]X2

      χ̂1 = Φ̂1 j1^T
      χ̂2 = Φ̂2 j2^T
      m = (j1 χ̂1 + j2 χ̂2)^−1
      v = j1 v̂1 + j2 v̂2
      Δdes v = velocityMap(v) − v
      Δdes v = Δdes v + evalBaumgarteTerm()

    end
  end
end
```

Procedure 7.2: Determination of Jacobians, effective masses and desired velocity increments

### 7.4.3 Velocity correction

Once all the constraints are prepared, the velocity correction can start. Each body has its velocity increment vector which is initially set to zero and is successively updated when the currently processed constraint happens to restrain a specific body. High-level description of the procedure is - for each constraint space dimension:

- evaluate the current constraint space dimension (scalar) velocity increment generated by impulses applied so far

$$\Delta v = \boldsymbol{j}_1 \Delta \widehat{\boldsymbol{v}}_1 + \boldsymbol{j}_2 \Delta \widehat{\boldsymbol{v}}_2, \tag{7.5}$$

  where $\boldsymbol{j}_*$ is the constraint Jacobian row corresponding to the current dimension and $\widehat{\boldsymbol{v}}_*$ is the constrained body's spatial velocity increment generated

by impulses applied to it so far

- determine how much the current value of $v$ needs to change in order to reach the target value and use it to evaluate the (scalar) impulse to be applied:

$$\Delta p = m(\Delta_{des} v - \Delta v), \tag{7.6}$$

where $m$ is the effective mass w.r.t. the current dimension

- clamp $\Delta p$ so that the value of the total impulse applied in this dimension stays within the required range

- update $\Delta \boldsymbol{v}_1$ and $\Delta \boldsymbol{v}_2$ vectors by applying $\Delta p$ to the restrained bodies (using opposite signs)

$$\Delta \widehat{\boldsymbol{v}}_1 = \Delta \widehat{\boldsymbol{v}}_1 + \widehat{\boldsymbol{\Phi}}_1 \boldsymbol{j}_1^T \Delta p \qquad \text{and} \qquad \Delta \widehat{\boldsymbol{v}}_2 = \Delta \widehat{\boldsymbol{v}}_2 - \widehat{\boldsymbol{\Phi}}_2 \boldsymbol{j}_2^T \Delta p, \tag{7.7}$$

where $\widehat{\boldsymbol{\Phi}}_*$ denotes the global-coordinates inverse spatial inertia of the body w.r.t. to its center of mass.

Note that the $\widehat{\boldsymbol{\Phi}}_* \boldsymbol{j}_*^T$ is repeated in every iteration so it could be pre-computed and stored to slightly reduce the amount of computations: Proc. 7.3 uses vectors $\widehat{\boldsymbol{\chi}}_*$ for that purpose (they pre-computed and stored by Proc. 7.2).

It is an exact application of the iterations described in section 7.3. The non-critical but helpful feature of using this approach is the fact that equations are far less abstract than for (P)GS which is easy to notice when inspecting Proc. 7.3.

## 7.5 Simulations

### 7.5.1 Cube

#### 7.5.1A The model

The first scenario is very basic and involves only a single cube falling onto a flat immobile ground under the influence of gravity. Most important simulation parameters are listed in Tab. 7.1. Velocity-level Baumgarte stabilization is enabled for the non-penetration constraints.

#### 7.5.1B Results

The simulation is stable and the resulting animation is visually plausible: snapshots are presented in Fig. 7.1.

```
procedure correctVelocities()
begin
  while stop criteria not met do

    foreach constraint in the system do

      Δv̂1/Δv̂2 := current velocity increment of the 1st/2nd body

      foreach dimension of the constraint do
        i := dimension index
        j1/j2 := the i^th row of the constraint Jacobian w.r.t. the 1st/2nd body
        χ̂1/χ̂2 := spatial reponse to impulse j1^T/j2^T applied to the 1st/2nd body
        m := effective mass w.r.t. the dimension
        Δv : curr. increment of the constraint-space vel. in the dim.
        Δ_des v := desired value of Δv
        p := total corrective impulse in this dimension
        Δp : increment of p
        h/l := upper/lower bound of p

        Δv = j1 Δv̂1 + j2 Δv̂2
        Δp = (Δ_des v − Δv) m

        if p + Δp < l then
          Δp = l − p
          p = l
        elseif p + Δp > h then
          Δp = h − p
          p = h
        else
          p = p + Δp
        end

        Δv̂1 = Δv̂1 + χ̂1 Δp
        Δv̂2 = Δv̂2 − χ̂2 Δp

      end
    end
  end
end
```

Procedure 7.3: Velocity correction procedure; termination criteria are not specified

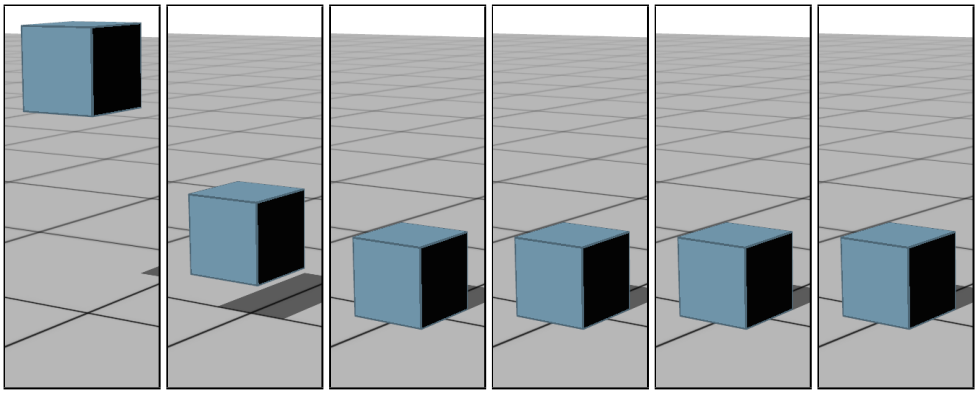
### 7.5.2 Multiple cubes

#### 7.5.2A The model

The individual models in this scenario are again unit cubes which fall onto a float immobile ground under gravity. The difference is that this time we have 100 of them positioned one above the other forming a column of initially non-touching objects. Velocity-level Baumgarte stabilization is enabled for the non-penetration constraints.

| Parameter | Value | Unit |
|---|---|---|
| initial altitude | 3.0 | m |
| initial velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | m/s |
| initial ang. velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | rad/s |
| grav. acceleration | $x = 0.0 \quad y = 0.0 \quad z = -9.81$ | $m/s^2$ |
| simulation step | 1/60 | s |

Table 7.1: Simulation parameters and initial conditions

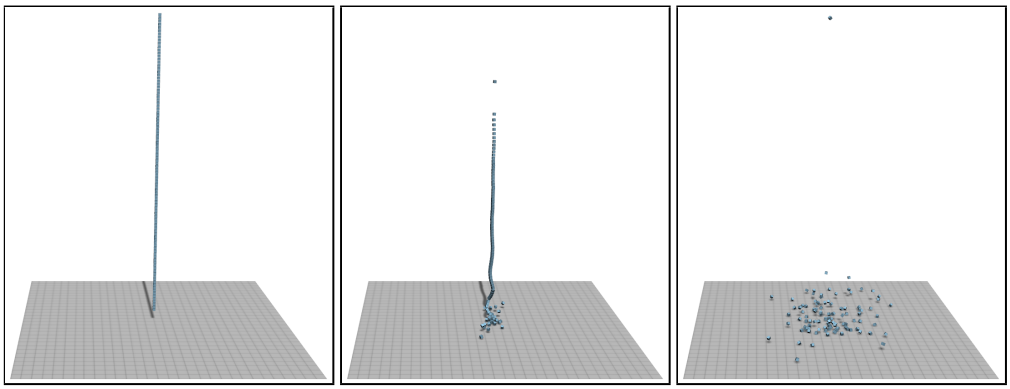


(a) 0.0s (b) 1.0s (c) 2.0s (d) 3.0s (e) 4.0s (f) 5.0s

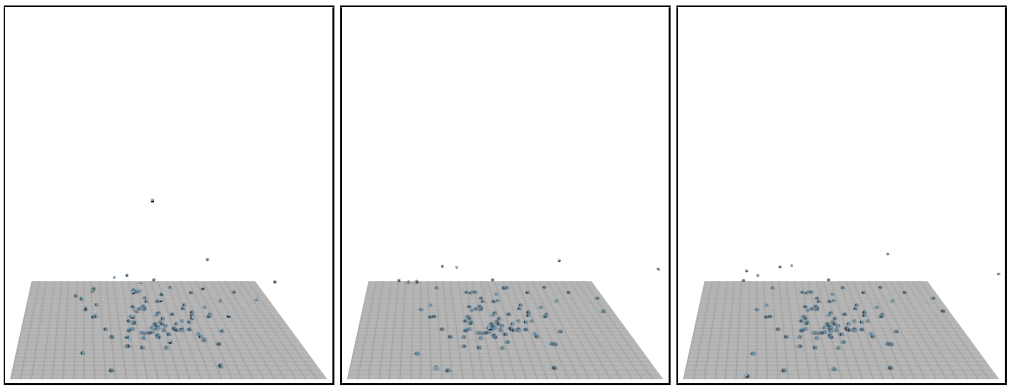
Figure 7.1: Simulation of a single falling cube

| Parameter | Value | Unit |
|---|---|---|
| initial altitude (bottom) | 3.0 | m |
| initial offset | $x = 0.0 \quad y = 0.0 \quad z = 1.1$ | m |
| initial velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | m/s |
| initial ang. velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | rad/s |
| grav. acceleration | $x = 0.0 \quad y = 0.0 \quad z = -9.81$ | $m/s^2$ |
| simulation step | 1/60 | s |
| iteration count | 20 | - |

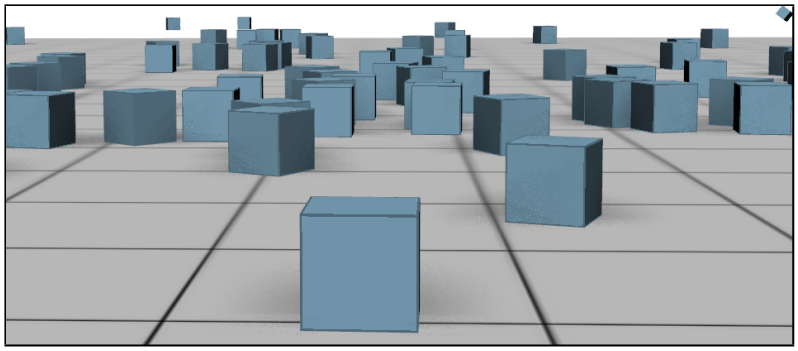
Table 7.2: Simulation parameters and initial conditions



(a) 0.0s (b) 3.0s (c) 6.0s



(d) 9.0s (e) 12.0s (f) 15.0s



(g) 15.0s. zoomed in

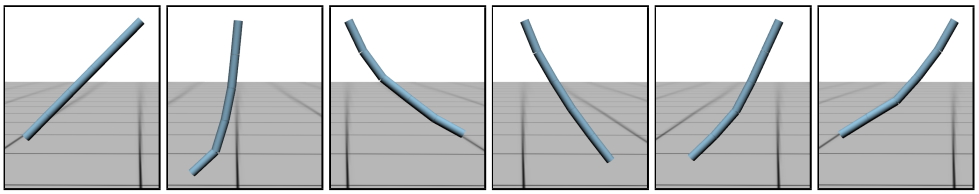
Figure 7.2: Simulation of 100 falling cubes

### 7.5.2B Results

The snapshots presented in Fig. 7.2 show that the simulation is stable and the resulting animation is visually plausible.

| Parameter | Value | Unit |
|---|---|---|
| initial $y$-angle (top) | 0.7854 | rad |
| initial velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | m/s |
| initial ang. velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | rad/s |
| grav. acceleration | $x = 0.0 \quad y = 0.0 \quad z = -9.81$ | $m/s^2$ |
| simulation step | 1/60 | s |
| iteration count | 20 | - |

Table 7.3: Simulation parameters and initial conditions



(a) 0.0s (b) 0.5s (c) 1.0s (d) 1.5s (e) 2.0s (f) 2.5s

Figure 7.3: Simulation of a 5-link pendulum (early stage)

### 7.5.3 Pendulum

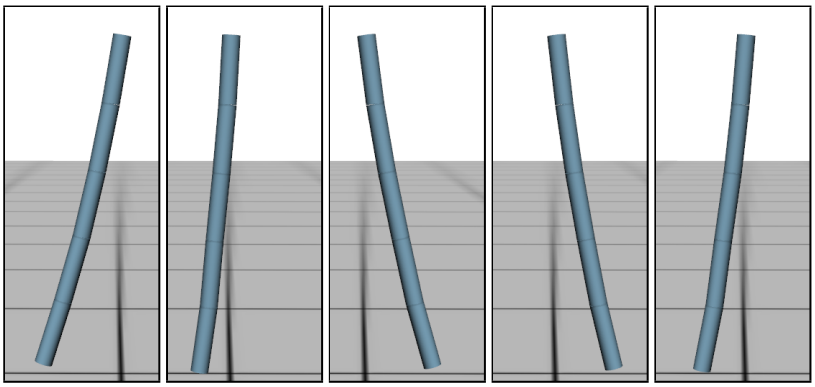
#### 7.5.3A The model

The first scenario testing joint constraints will involve a 5-element pendulum swinging under the influence of gravity. The system is created from cylinders connected to each other by hinge joints. The topmost cylinder is additionally hinged to the *world* at an altitude which eliminates the possibility of any of the bodies colliding with the ground (supposing the simulation does not introduce too much error). Joints error is eliminated using Baumgarte stabilization. Key parameters of the simulation are listed in Tab. 7.3.

#### 7.5.3B Results

The simulation is stable and the produced animation is visually convincing (Fig. 7.3). However, if we let the simulation run for a longer time, it becomes clear that artificial damping is being introduced. It is apparent if one compares the snapshots from the beginning of a long simulation (Fig. 7.3) with those created along its course (Fig. 7.4) - the range of motion has been visibly decreased.

Nevertheless, taking into account the application domain it is not really that great a problem since one cannot really notice the damping without taking the time to observe the motion for tens of seconds: a rare situation in case of video games.



(a) 2040.0s (b) 2040.5s (c) 2041.0s (d) 2041.5s (e) 2042.0s

Figure 7.4: Simulation of a 5-link pendulum (late stage)

| Parameter | Value | Unit |
|---|---|---|
| initial altitude (pelvis) | 1.5 | m |
| initial velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | m/s |
| initial ang. velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | rad/s |
| simulation step | 1/60 | s |
| iteration count | 20 | - |

Table 7.4: Simulation parameters and initial conditions

### 7.5.4 Rag-doll

The first articulated scenario concerns a passive humanoid structure: a simple (headless and armless) rag-doll.

#### 7.5.4A Model

All joints in the system are assumed to be hinges and are modeled by hinge constraints. Key simulation settings are presented in Tab. 7.4.

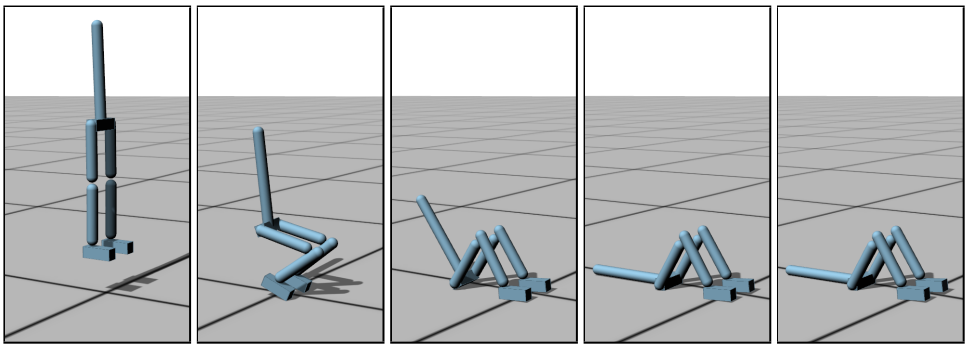
#### 7.5.4B Results

The rag-doll simulation reaches its steady state after about 5 seconds and the final configuration is plausible. The resulting snapshots are presented in the Fig. 7.5.

### 7.5.5 Rag-doll stack

#### 7.5.5A The model

The multibody system in this scenario is consists of two slanted columns of rag-dolls introduced in the previous section all of which fall under the influence of gravity onto a flat ground. The total body count is thus 480.



(a) 0.0s (b) 1.0s (c) 2.0s (d) 3.0s (e) 4.0s

Figure 7.5: Simulation of a single 8-link falling rag-doll

#### 7.5.5B Results

The simulation is stable but we found it did not really reach a steady-state: there is substantial amount of jitter which is mostly due to the method begin iterative and the fact of terminating the solver basing on a fixed iteration count [119] since SI is prone to slow convergence in the case of *big* stacks.

### 7.5.6 Loaded net

#### 7.5.6A The model

To test non-penetration and joint constraints at the same time, we shall suspend a net build of cylinders by hinging it to the *world*[7] and drop several boxes onto it under the influence of gravity. The net is built from cylinders connected using Baumgarte-stabilized hinge constraints. Since it is difficult to describe the model parameters as it was done with previous scenarios, we shall skip the table and only state that the simulation step is again 1/60 and that the iteration count is set to 20.

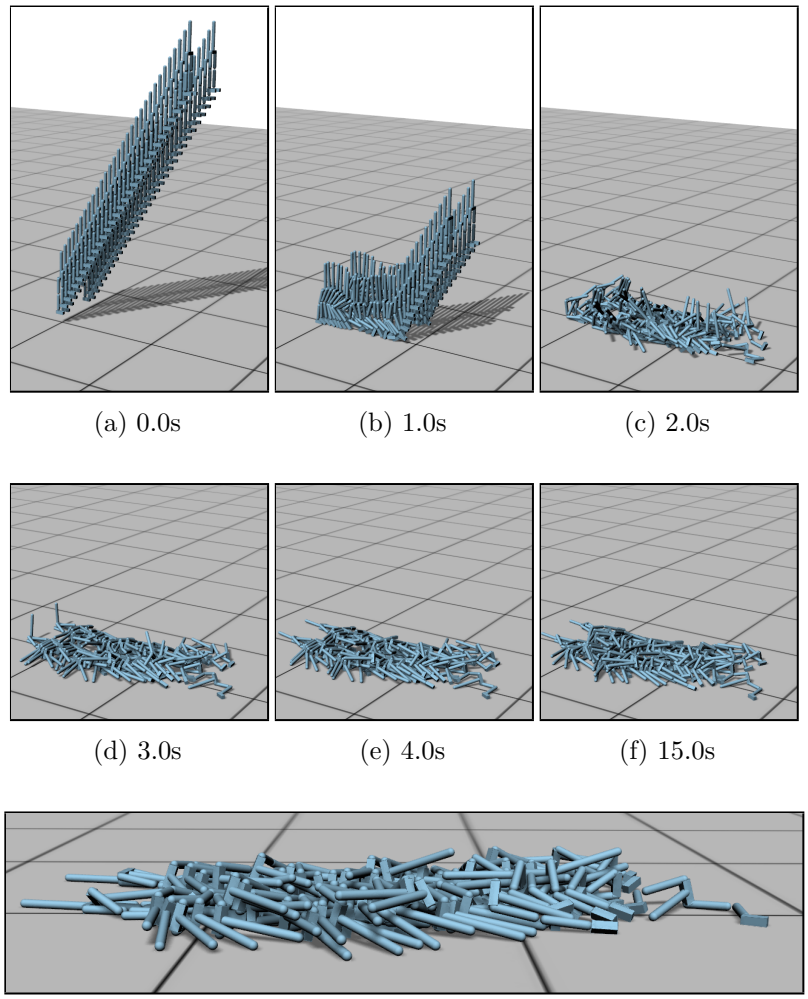
#### 7.5.6B Results

The snapshots of the simulation are presented in Fig. 7.7. We can see that SI has once again produced a visually plausible animation without introducing instabilities.

### 7.5.7 Tracked vehicle

The final test involves a tracked vehicle (which we shall shortly refer to as *a tank*) combines joint- and non-penetration with heavy dependence on friction modeling.

[7] It is equivalent to connecting the net to a virtual body of infinite mass.



(a) 0.0s (b) 1.0s (c) 2.0s

(d) 3.0s (e) 4.0s (f) 15.0s

(g) 15.0s zoomed in

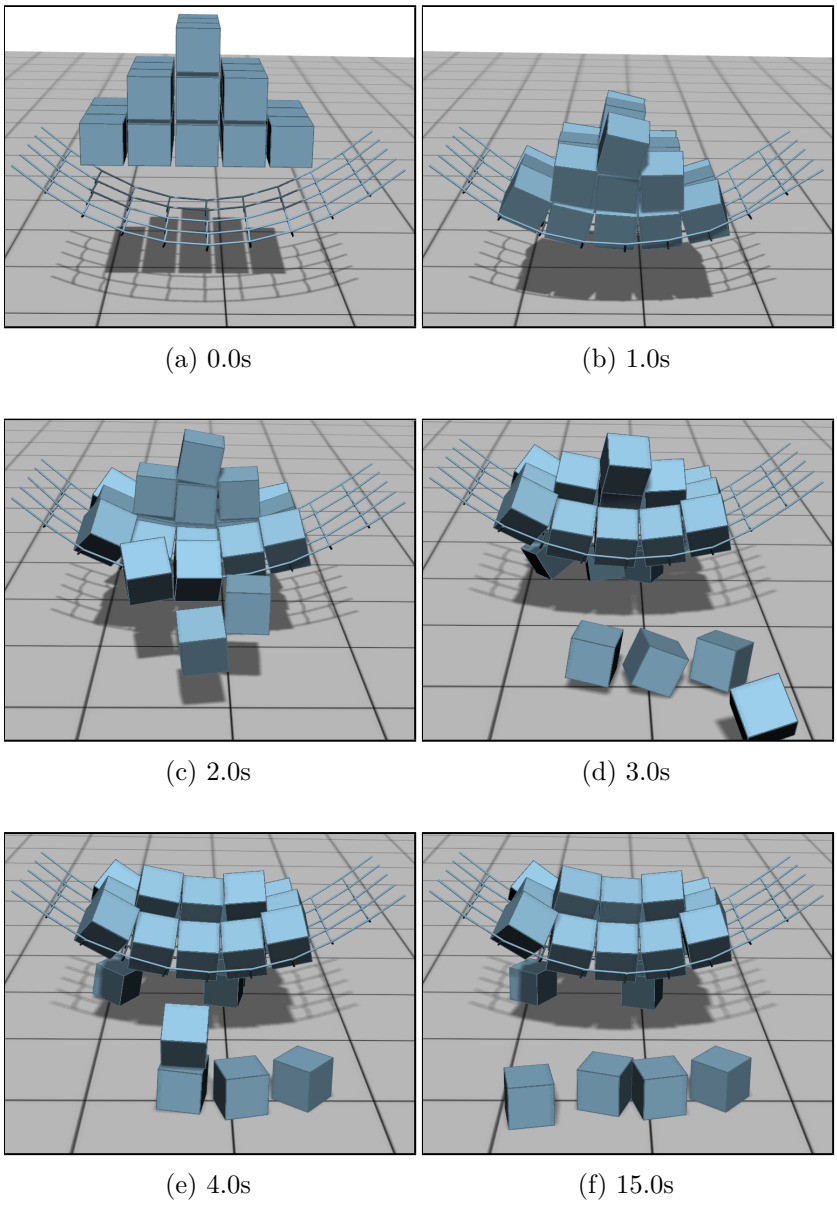
Figure 7.6: Simulation of 60 8-link falling rag-dolls

Tanks appear in modern video games (e.g. *Battlefield 3* or *World of Tanks*) quite often but it is extremely rare to see their motion being fully simulated on the fly.

### 7.5.7A The model

On each side, the tank has one actuated drive sprocket and a set of unactuated *wheels*: three road wheels, an idler and return rollers (Fig. 7.8). Each wheel rolls on the caterpillar track which prevents its direct contact with the ground. The model has been constructed as follows:

- hull is a cuboid and is free to move with respect to an immobile base of the hierarchy



(a) 0.0s (b) 1.0s

(c) 2.0s (d) 3.0s

(e) 4.0s (f) 15.0s

Figure 7.7: Simulation of a net with 27 cubes dropped onto it

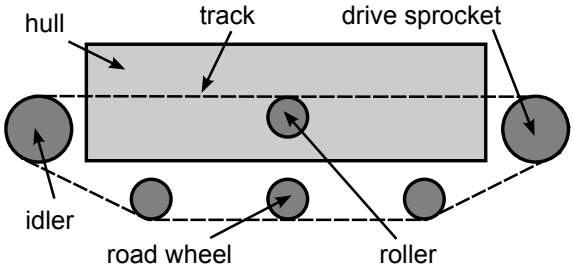



Figure 7.8: Simple tracked vehicle model

| Parameter | Value | Unit |
|---|---|---|
| initial altitude (bottom track plates) | 0.0 | m |
| initial velocity (hull) | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | m/s |
| initial ang. velocity (hull) | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | rad/s |
| simulation step | 1/60 | s |
| iteration count | 20 and 100 | - |

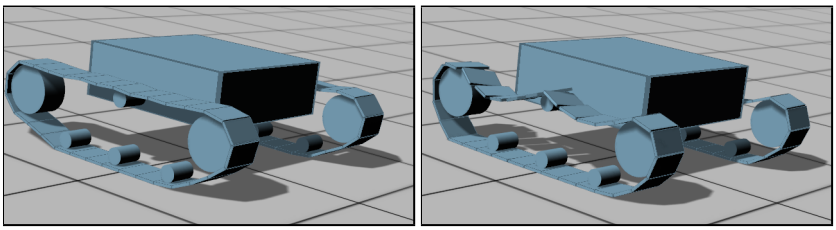
Table 7.5: Simulation parameters and initial conditions

- track chains are built from flat cuboids linked by hinge constraints so that each of them forms a loop
- wheels are attached to the hull using hinge constraints
- contact between the actuated wheels and the track is modeled using non-penetration constraints with infinite friction
- contact between the unactuated wheels and the track is modeled using non-penetration constraints with infinite lateral friction and finite (Coulomb) longitudinal friction
- contact between the track links and the ground is modeled using 3D non-penetration constraints (Coulomb friction)

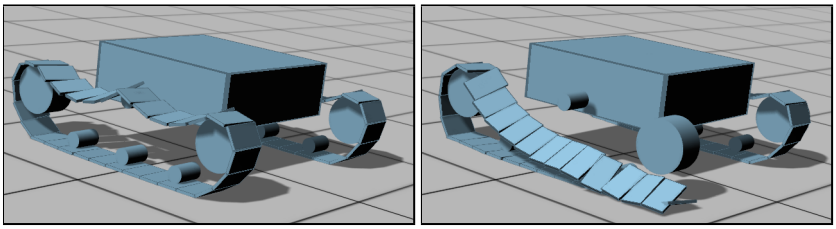
Infinite friction between the sprockets and the track models the meshed sprocket-track connection. Infinite lateral friction between wheels and the tracks is a simple way of modeling guide horns on individual track links used in real-life tracked vehicles to restrict lateral motion of the wheels relative to the track and thus prevent its throwing.

### 7.5.7B Results

Since the tracks kept falling apart for our standard setting of 20 iterations (Fig. 7.9), we increased their count to 100 (which obviously increases the amount

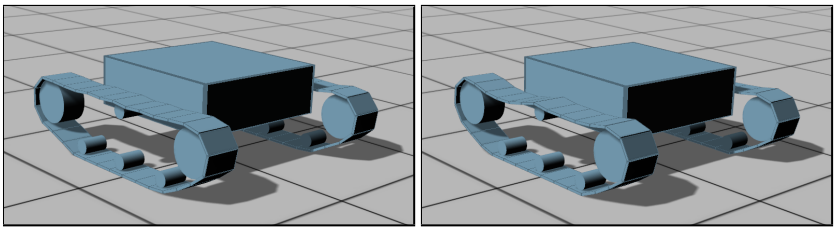


(a) 0.0s (b) 2.0s

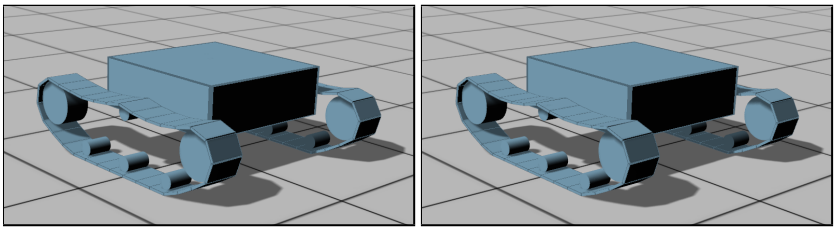


(c) 4.0s (d) 8.0s

Figure 7.9: Simulation of a tracked vehicle (20 iterations)



(a) 0.0s (b) 3.0s



(c) 6.0s (d) 9.0s

Figure 7.10: Simulation of a tracked vehicle (100 iterations)

of computations). It is an occurrence of a problem similar to this encountered with the rag-doll stack scenario only this time we are dealing with a relatively long chain rather than a big stack. Once the higher iteration limit was set, however, the resulting simulation was stable, produced animation visually satisfactory including frictional caterpillar-ground contact which is essential for tracked vehicles motion and turning. Snapshots are presented in Fig. 7.10.

## 7.6 Summary

The results of the various tests confirm the general robustness of SI. Furthermore, the quality of the produced animations is enough to create the illusion of physically-valid motion with the caveat that there is an inherent damping introduced by this method which, however, does not really substantially reduce its applicability in the video gaming domain. A more serious problem is the slow convergence of long chains (or big stacks in the case of non-penetration constraints) but *warm-starting* is known to reduce the iteration count.

# Chapter 8

# Articulated Islands Algorithm

## 8.1 Overview

The current chapter is the climax of this dissertation since it introduces the proposed method of modifying the Sequential Impulse method described in Chap. 7 in order to make it utilize the reduced-coordinate formulation when needed. Description of the method is followed by a discussion of problems one may encounter and means that should be applied to overcome them. Next, results of a number of test simulations are presented. Finally, the limitations of the method and future research challenges are discussed.

## 8.2 Articulated Sequential Impulse

In this section we shall provide a SI-resemblant procedure using CRBA and the joint-space test method (Chap. 6) as the underlying approach to modeling dynamics. Since the proposed approach can be perceived as the articulation-oriented algorithm presented in [43] modified so as to handle additional constraints similarly to SI, we shall call it the *Articulated Sequential Impulse* algorithm.

### 8.2.1 Velocity prediction

We will use an articulated dynamics algorithm to determine the accelerations needed to evaluated tentative velocities. As a result, these velocities will describe an articulated system rather than a set of free bodies and thus kinematic-tree joints will not have to be introduced via velocity correction. For large articulated systems the best choice for the dynamics algorithm would probably be ABA, but basing on the analysis provided in Chap. 6 we know that the CRBA-based method performs much better when there is a high number of additional constraints -

this rule applies also now. Therefore, we believe that the best approach is to use ABA when the individual articulated systems are large, but there are few or no additional constraints (i.e. the tentative velocities do not need correction) and CRBA-based procedure otherwise. In this work, however, we assume that CRBA-based algorithm is used exclusively[1].

Once the joint-space acceleration have been determined, they are Euler-integrated into the current joint-space velocity vector yielding the tentative velocities. If there are no additional constraints that need to be processed, the tentative velocities are accepted as final and integrated into the current configuration vector, which obviously means that the symplectic Euler integration is used.

### 8.2.2 Preparation for velocity correction

#### Body Jacobians

First, we need to determine Jacobians for these bodies which are restrained by at least one constraint. The procedure is obviously the same as the one discussed in Sec. 6.3.1A.

#### Constraints' Jacobian rows

Once body Jacobians have been determined, we can proceed to individual constraint Jacobian rows for each constraint space dimension. The formula is just a single-row version of Eq. 6.13.

#### Effective masses

Again, this stage is very similar to its counterpart in the procedure for determining matrix $\boldsymbol{A}$ presented in Chap. 6:

- for each dimension of the constraint-space solve:

$$\boldsymbol{H}\boldsymbol{\chi}_1 = \boldsymbol{j}_i^T \qquad \text{and} \qquad \boldsymbol{H}\boldsymbol{\chi}_2 = \boldsymbol{j}_2^T, \tag{8.1}$$

  where $\boldsymbol{j}_i$ is the row of the constraints' Jacobian corresponding to the current dimension w.r.t. to the $i^{th}$ body[2] and $\boldsymbol{\chi}_i$ is the joint-space response of the system to impulse $\boldsymbol{j}_i^T$ w.r.t. to the $i^{th}$ body; note that $\boldsymbol{\chi}_i$ is stored since it will save computations in the velocity correction stage. Obviously, matrix $\boldsymbol{H}$ is factorized once in the Forward Dynamics algorithm so we are now doing the substitution step only.

- determine inverse of the scalar effective mass for this dimension using:

$$\boldsymbol{j}_1\boldsymbol{\chi}_1 + \boldsymbol{j}_2\boldsymbol{\chi}_2 \tag{8.2}$$

[1] Since the entire procedure of modeling *additional* constraints does not make much sense in cases when there are none and, basing on our earlier analysis, it is the main case when application of ABA would be justified.

[2] Which means it maps joint-space velocity to $M^6$ velocity of the $i^{th}$ body

#### Target velocity increments

The current constraint space velocity for each of its dimensions is determined from the joint-space tentative velocity vector and then mapped to the desired velocity using the velocity mapping functions described in :

$$\Delta_{des}v = f(\boldsymbol{j}_1\dot{\boldsymbol{q}} - \boldsymbol{j}_2\dot{\boldsymbol{q}}) - (\boldsymbol{j}_1\dot{\boldsymbol{q}} - \boldsymbol{j}_2\dot{\boldsymbol{q}}), \tag{8.3}$$

where $f$ denotes the mapping function.

The entire procedure is presented in Proc. 8.1. For now, we assume velocity-level Baumgarte stabilization-based configuration error elimination which obviously influences the target velocity increment. Alternatives will be discussed later in this chapter

### 8.2.3 Velocity correction

Once the effective masses, velocity errors and constraints Jacobians are determined, iterative velocity correction process starts. The correction effects are accumulated using a velocity increment vector only this time (unlike SI which uses per-body spatial vectors) it is a joint-space vector representing a tree in the system; at the beginning of each iteration the current constraint-space velocity is evaluated from that vector and at the end - the effect of the effective impulse is accumulated in it.

The procedure works as follows, for each constraint-space dimension:

- evaluate the current constraint space dimension (scalar) velocity increment, generated by impulses applied so far

$$\Delta v = \boldsymbol{j}_1\Delta\dot{\boldsymbol{q}} - \boldsymbol{j}_2\Delta\dot{\boldsymbol{q}} = (\boldsymbol{j}_1 - \boldsymbol{j}_2)\Delta\dot{\boldsymbol{q}}, \tag{8.4}$$

- determine how much the current value of $\Delta v$ needs to change in order to reach the target value and use it to evaluate (scalar) impulse to be applied:

$$p = m(\Delta_{des}v - \Delta v) \tag{8.5}$$

- update the $\Delta\dot{\boldsymbol{q}}$ vector by applying p to the to the tree via the constrained bodies (using opposite signs)

$$\begin{aligned}\Delta\dot{\boldsymbol{q}} &= \Delta\dot{\boldsymbol{q}} + \boldsymbol{\chi}_1^T p - \boldsymbol{\chi}_2^T p \\ &= \Delta\dot{\boldsymbol{q}} + (\boldsymbol{\chi}_1^T - \boldsymbol{\chi}_2^T)p.\end{aligned} \tag{8.6}$$

The entire procedure is presented in Proc. 8.2.

## 8.3 Problems with body-local coordinates

The algorithm in its current form uses body-local coordinates for any kind of a simulated system which introduces two major issues that need to be addressed: fictitious force/acceleration terms and applying symplectic Euler integration.

```
procedure processConstraints()
begin
  foreach constraint in the system do

    T := reaction force subspace matrix of the constraint
    q̇ := joint-space velocity of the system
    X_1 := transf. from the 1st body's frame to the constraint's frame
    X_2 := transf. from the 2nd body's frame to the constraint's frame

    foreach dimension of the constraint do
      i := dimension index
      J_1/J_2 := Jacobian of the 1st/2nd body
      a_1/a_2 := system DoF offset of the 1st/2nd body
      b_1/b_2 := DoF count of the 1st/2nd body
      c_1/c_2 : auxiliary variable

      c_1 = a_1 + b_1
      c_2 = a_2 + b_2

      j_1/j_1 : the i^th row of the constraint Jacobian w.r.t. the 1st/2nd body
      χ_1/χ_2 : joint-space reponse to impulse j_1^T/j_2^T applied to the 1st/2nd body
      m : effective mass w.r.t. the dimension
      v : constraint-space velocity in the dimension
      Δ_des v : desired increment of v


      j_1 = T^T[i,*] X_1 J_1[*, a_1 → c_1]
      j_2 = −T^T[i,*] X_2 J_2[*, a_2 → c_2]

      χ_1 = solveFactorized(H_fact, j_1^T)
      χ_2 = solveFactorized(H_fact, j_2^T)

      m = (j_1 χ_1 + j_2 χ_2)^−1
      v = (j_1 + j_2) q̇
      Δ_des v = velocityMap(v) − v
      Δ_des v = Δ_des v + evalBaumgarteTerm()

    end
  end
end
```

Procedure 8.1: Determination of Jacobians, effective masses and desired velocity increments

### 8.3.1 Fictitious force/acceleration terms

First, let us present the problem occurring in a simulation - the multiple cubes scenario. Snapshots of the simulation are presented in Fig. 8.1.

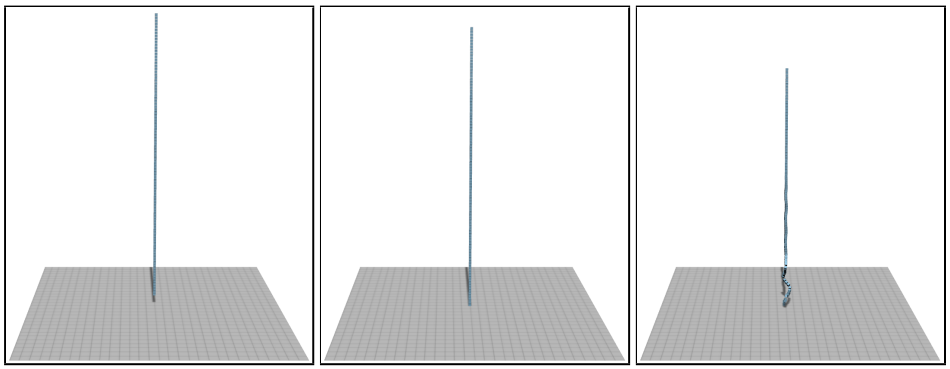
Unfortunately, the simulation is unstable and the generated values become unmanageable after about 3.5 seconds. The most direct cause of simulation crash is that at least one of the bodies gains a very large linear velocity. Before this occurs, there is usually a build-up process which needs to be analyzed and traced

```
1  procedure correctVelocities()
2  begin
3    while stop criteria not met do
4
5      foreach constraint in the system do
6
7        Δq̇ := current increment of the joint-space velocity
8
9        foreach dimension of the constraint do
10         i := dimension index
11         j1/j2 := the i^th row of the constraint Jacobian w.r.t. the 1^st/2^nd body
12         χ1/χ1 := joint-space reponse to impulse j1^T/j2^T applied to the 1^st/2^nd body
13         m := effective mass w.r.t. the dimension
14         Δv : curr. increment of the constraint-space vel. in the dim.
15         Δ_des v := desired value of Δv
16         p := total corrective impulse in this dimension
17         Δp : increment of p
18         h/l := upper/lower bound of p
19
20         Δv = (j1 + j2)Δq̇
21         Δp = (Δ_des v − Δv)m
22
23         if p + Δp < l then
24           Δp = l − p
25           p = l
26         elseif p + Δp > h then
27           Δp = h − p
28           p = h
29         else
30           p = p + Δp
31         end
32
33         Δq̇ = Δq̇ + (χ1 − χ2)Δp
34       end
35     end
36   end
37 end
```

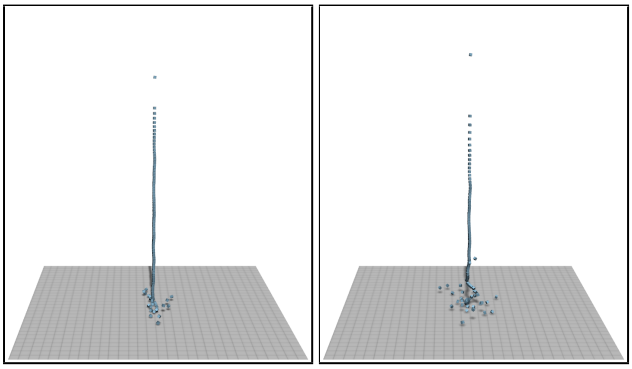
Procedure 8.2: Velocity correction procedure; termination criteria are not specified

back to the original cause[3] if one hopes to eliminate the instability. However, if the aim is to show that an instability occurs, the maximal speed in the system is a convenient measure. Let us take a look at Fig. 8.2 - we can clearly see that the undesired velocity build-up starts after about 3 seconds and soon becomes intractable.

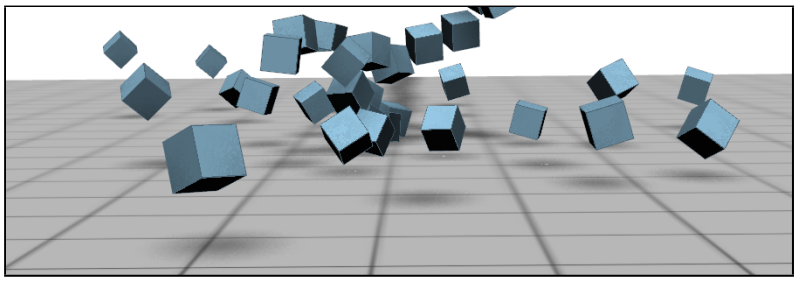
[3] Which often turns out to be a typical *chicken or the egg* causality dilemma.



(a) 0.0s (b) 1.0s (c) 2.0s



(d) 3.0s (e) 3.43s



(f) 3.43s. zoomed in

Figure 8.1: Unstable simulation of 100 falling cubes

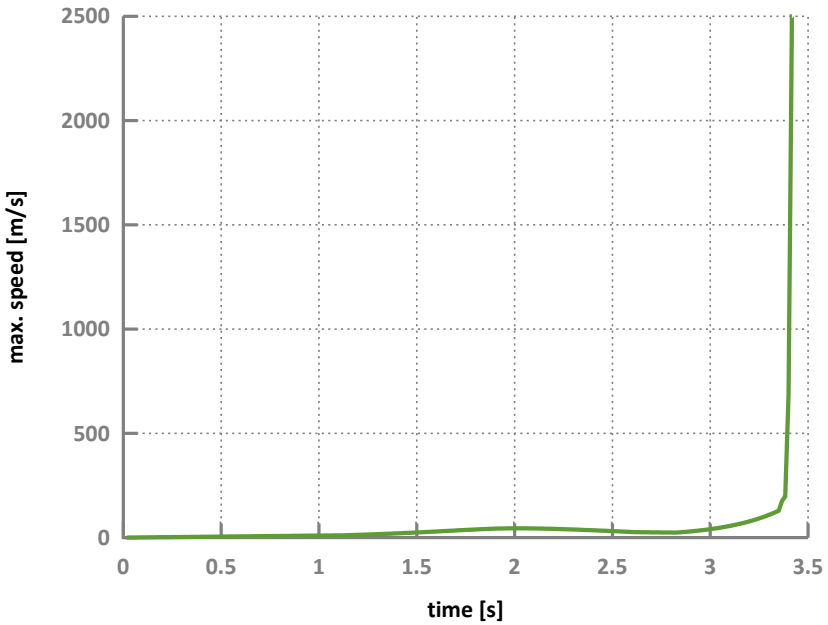



Figure 8.2: Unstable simulation 100 falling cubes: maximal speed in the system in consecutive simulation steps

| Parameter | Value | Unit |
|---|---|---|
| initial altitude | 5.0 | m |
| initial velocity | $x = 15.4355 \quad y = -12.0092 \quad z = 5.99297$ | m/s |
| initial ang. velocity | $x = -12.9186 \quad y = -11.8749 \quad z = 6.68083$ | rad/s |
| simulation step | 1/60 | s |

Table 8.1: Simulation parameters and initial conditions

#### 8.3.1A Tracing the source of the problem

Analyzing the velocity build-up process in a scenario involving 100 bodies is a cumbersome task. Therefore, hoping that the origins of the problem do not lie in the number of simulated objects, we will try to reproduce the instability in a simpler scenario. We have isolated a single cube shortly before the build-up process started, disabled non-penetration constraints and gravity. Key simulation parameters are listed in Tab. 8.1.

Since there are no external forces on the body, its angular and linear momenta should remain constant (within tolerance) which means that the velocities' magnitudes should stay fixed as well. The results clearly show that while the angular speed *behaves* as expected (Fig. 8.4), we can observe a very fast build-up of the linear velocity magnitude (Fig. 8.3). Such unexplained differences are obviously undesired not to mention the fact that cases like the ones we have tested are likely

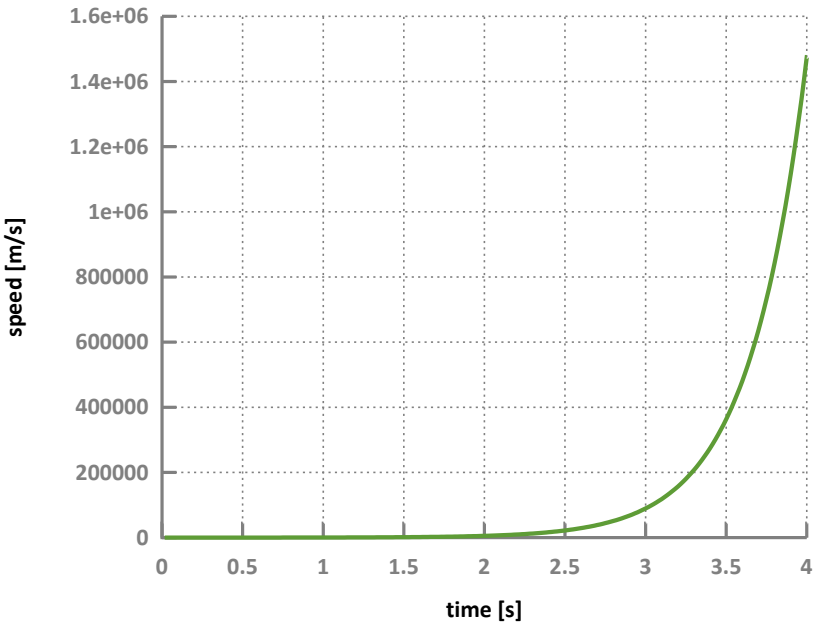



Figure 8.3: Unstable simulation of a spinning cube: maximal speed in the system in consecutive simulation steps

to occur frequently in real-life scenarios which makes our proposition inapplicable to the domain we are targeting. This problem needs to be eliminated.

Tackling this particular issue requires further analysis of the applied articulated algorithm: unlike the FD algorithm we use in SI, CRBA - as proposed by Featherstone - uses local coordinate frames rigidly attached to each body to express its velocity and acceleration[4]. This obviously means that fictitious forces appear: centrifugal, Euler and Coriolis which fill the gap between what *seems* to be physical in non-inertial frames and what really *is* such in inertial frames.

Centrifugal force is not of great concern since it can be directly mapped to an actual centripetal force which makes the coordinate frame/body follow a curved path. The same applies to the Euler term which which results from coordinate transformation of angular acceleration generating tangential linear acceleration. What follows is that no matter if we choose to use non-inertial or inertial frames, tangent and radial force terms will appear. The Coriolis term is different: it is entirely perceptional, does not directly reflect any physical interaction and is thus non-existent in inertial frames. Note that the very same cube-stack scenario was simulated successfully using SI (Sec. 7.5.2B) which expresses the velocities and accelerations in the stationary *world* frame. The conclusion is that this could be the differentiating factor which causes the linear velocity build-up for our

[4]The choice allows for many optimizations and it would not be wise to force a change in this aspect, not to mention our original assumption.

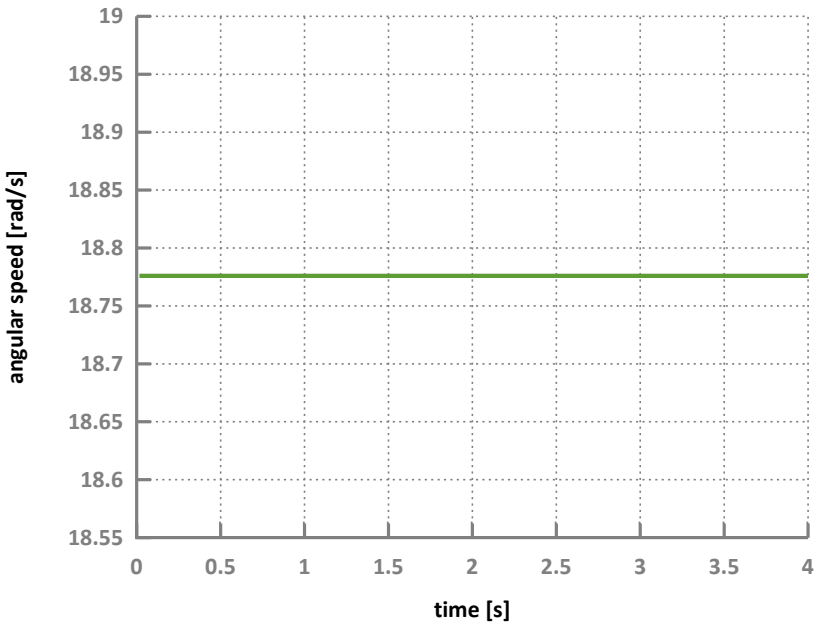



Figure 8.4: Unstable simulation of a spinning cube: maximal angular speed in consecutive simulation steps

simulator.

The formula for the Coriolis inertial force term is (all fictitious terms are derived in Sec. A.1):

$$-2m\boldsymbol{\omega}_{T,i} \times \boldsymbol{v}_{L,i}, \tag{8.7}$$

where vector $\boldsymbol{\omega}_{T,i}$ describes the total angular velocity of the coordinate frame, vector $\boldsymbol{v}_{L,i}$ describes linear velocity of a body as perceived by an observer rigidly attached to the frame and $m$ is mass of the body which turns the velocity-product acceleration[5] term into inertial force. The only things about Eq. 8.7 immediately important in the forthcoming analysis are:

- the Coriolis acceleration term is non-zero only in the case of joints which allow for linear motion since the formula uses the local velocity vector
- $\boldsymbol{\omega}_{T,i}$ describes the *total/absolute* angular velocity of body $i$ frame and thus its coordinate frame
- the factor 2 means that there are two equal contributions to Coriolis force: (i) inertially contact velocities appear to continually change direction in

[5]Acceleration Coriolis term, $2\boldsymbol{\omega} \times \boldsymbol{v}$, added to the acceleration perceived in the non-inertial frame gives us the actual, inertially observed acceleration. The minus sign in inertial force formula in Eq. 8.7 effectively strips the inertially perceived acceleration of this term leaving the non-inertially perceived component.

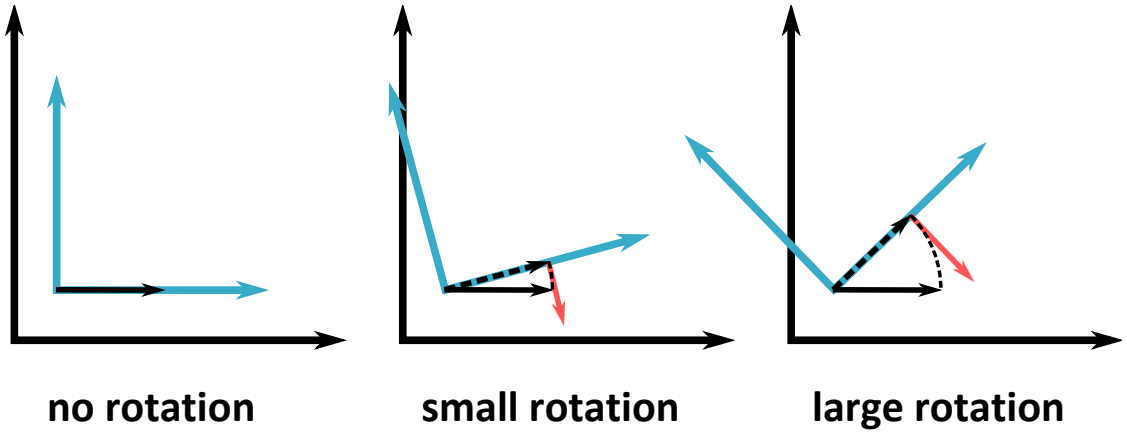



Figure 8.5: Illustration of the error introduced by Euler-integrating the Coriolis acceleration: black and cyan coordinate axes represent the inertial and non-inertial frames of reference, respectively; solid black and red arrows represent inertially constant velocity vector and Coriolis acceleration term (its magnitude is artificially adjusted in this figure), respectively; dashed black arrow depicts what the inertially constant velocity vector would appear to be for an inertial observer in the absence of the Coriolis acceleration. As one can see, the discrepancy between the actual velocity vector and the one approximated using Euler-integrated Coriolis acceleration will grow with larger relative rotation increments between parent and child frames. Large relative orientations result from high angular velocity of the child frame relative to the parent and large integration time-steps.

rotating coordinate frames[6] (ii) if an object traveling in a rotating frame changes its perpendicular distance w.r.t. the the axis of rotation, its velocity in an inertial frame changes even if it seems to be fixed in the rotating frame.

By looking at the first point, one can easily draw the first conclusion that in the case of free body-simulation Coriolis acceleration/force needs to be taken into consideration for all bodies.

As it turns out, the problem with Coriolis acceleration/force term is the way it gets integrated into velocity, especially when simplistic integration schemes and large steps are used, which is exactly the case of interactive simulators. The problem is illustrated in the Fig. 8.5. So, although the acceleration term by itself is evaluated precisely, it gets poorly integrated and in time, especially when dealing with high angular velocities, changes the magnitude of velocity which often leads to overall instability.

[6] And, on the other hand, velocity constant in a rotating frame is actually continually changing its direction.

### 8.3.1B Proposed resolution

Naturally, we cannot simply ignore the Coriolis effect: we will introduce it in a different fashion - by mimicking the phenomenon that causes it, i.e. by rotating the linear velocity (once it is established) in the opposite direction and the same angle that the angular velocity rotates the body and thus its corresponding coordinate frame. This way we can obviously obtain only the first contribution to the Coriolis term, but as it has been previously established it is usually the only one that is non-zero.

In order to introduce the Coriolis acceleration in the proposed way, we first need to establish a Coriolis-free acceleration: the physical meaning of this operation is that the body-local coordinate frame is positioned and oriented as if it has been traveling with the body up till now, but it remains fixed in the current instant. To achieve that, we will now analyze the articulated algorithms with the focus on fictitious forces terms and the corresponding accelerations.

One of the *slight nuisances* of using algorithms expressed with spatial algebra is that without close inspection it is not immediately obvious where exactly particular fictitious terms reside because:

- spatial acceleration is different from the classic acceleration and the difference is what Featherstone calls the *Coriolis terms* but it is not what is conventionally treated as one (see Sec. 5.3.1C)
- the $\widehat{\boldsymbol{v}} \times^* \widehat{\boldsymbol{I}}\widehat{\boldsymbol{v}}$ term conveys more than the conventional non-spatial $\boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}$, i.e. the derivative of angular momentum, $\boldsymbol{I}\boldsymbol{\omega}$, in coordinate frame rotating with angular velocity $\boldsymbol{\omega}$
- there is also the $\widehat{\boldsymbol{v}} \times \widehat{\boldsymbol{S}}\dot{\boldsymbol{q}}$ term due to differentiation of $\widehat{\boldsymbol{S}}\dot{\boldsymbol{q}}$ in coordinate frame moving with the body, i.e. with spatial velocity $\widehat{\boldsymbol{v}}$

Therefore, we will now disassemble each of these terms into its non-spatial constituents in order to make them more intuitive and find the actual, conventional fictitious/inertial forces.

As it is shown in App. A.1, fictitious force terms result from describing the acceleration as perceived by an observer accelerating w.r.t. to an inertial frame of reference. By looking at RNEA we can see that there are two spots where differentiation in moving coordinates occurs: (i) rigid-body spatial EOM (time-differentiated force-momentum balance); (ii) acceleration formula (time-differentiated velocity formula).

Since the forthcoming analysis is limited to a single rigid body (within a multibody system), we shall refer to it as *the analyzed body*. Since in such a case any indications identifying *a* body within the system are needless, we drop them to simplify the notation; the only exception is the topological parent of the

analyzed body - all quantities describing it will be labeled by $\lambda$. We shall now introduce several variables needed to perform the analysis:

- $\widehat{\boldsymbol{v}}_L = [\boldsymbol{\omega}_L^T \, \boldsymbol{v}_L^T]^T = \widehat{\boldsymbol{S}}\dot{\boldsymbol{q}}$ denotes the analyzed body's *local* (i.e. relative to its topological parent) spatial velocity as opposed to its absolute spatial velocity $\widehat{\boldsymbol{v}} = [\boldsymbol{\omega}^T \, \boldsymbol{v}^T]^T$
- $\widehat{\boldsymbol{v}}_\lambda = [\boldsymbol{\omega}_\lambda^T \, \boldsymbol{v}_\lambda^T]^T$ denotes absolute spatial velocity of the analyzed body's topological parent
- $\boldsymbol{c}$ denotes offset from the origin of the analyzed body's coordinate frame (pivot point) to its center of mass
- $\boldsymbol{r}$ denotes offset from the origin of the coordinate frame associated with the parent and the analyzed body (expressed in parent's coordinates)

If we temporarily assume that the coordinate frames of the analyzed body and its parent have identical orientations, the following relation holds:

$$\widehat{\boldsymbol{v}} = \widehat{\boldsymbol{v}}_\lambda + \widehat{\boldsymbol{v}}_L = \begin{bmatrix} \boldsymbol{\omega}_\lambda + \boldsymbol{\omega}_L \\ \boldsymbol{v}_\lambda + \boldsymbol{\omega}_\lambda \times \boldsymbol{r} + \boldsymbol{v}_L \end{bmatrix}. \tag{8.8}$$

Let us finally start the analysis; the spatial equation of motion of the analyzed body:

$$\widehat{\boldsymbol{f}} = \frac{d\widehat{\boldsymbol{I}}\widehat{\boldsymbol{v}}}{dt} = \widehat{\boldsymbol{I}}\widehat{\boldsymbol{a}} + \widehat{\boldsymbol{v}} \times^* \widehat{\boldsymbol{I}}\widehat{\boldsymbol{v}}. \tag{8.9}$$

Since the analyzed body is an element of articulated structure, its spatial acceleration, $\widehat{\boldsymbol{a}}$, can be further decomposed using the acceleration formula:

$$\widehat{\boldsymbol{a}} = \frac{\widehat{\boldsymbol{v}}}{dt} = \frac{d(\widehat{\boldsymbol{v}}_\lambda + \widehat{\boldsymbol{v}}_L)}{dt} = \frac{d(\widehat{\boldsymbol{v}}_\lambda + \widehat{\boldsymbol{S}}\dot{\boldsymbol{q}})}{dt} = \widehat{\boldsymbol{a}}_\lambda + \widehat{\boldsymbol{S}}\ddot{\boldsymbol{q}} + \dot{\widehat{\boldsymbol{S}}}\dot{\boldsymbol{q}}. \tag{8.10}$$

As one can see, the above equation forms a recursive relation since it contains the absolute spatial acceleration of the analyzed body's parent. More importantly, though, the motion subspace matrix $\widehat{\boldsymbol{S}}$ is differentiated in moving coordinates so:

$$\dot{\widehat{\boldsymbol{S}}}\dot{\boldsymbol{q}} = \mathring{\widehat{\boldsymbol{S}}}\dot{\boldsymbol{q}} + \widehat{\boldsymbol{v}} \times \widehat{\boldsymbol{S}}\dot{\boldsymbol{q}}. \tag{8.11}$$

The second term on the RHS in the above formula accounts for the variation of the motion subspace matrix due to the fact that the frame of reference is moving with velocity $\widehat{\boldsymbol{v}}$. By the definition of spatial cross product (Sec. 5.3.1E):

$$\widehat{\boldsymbol{v}} \times \widehat{\boldsymbol{S}}\dot{\boldsymbol{q}} = \widehat{\boldsymbol{v}} \times \widehat{\boldsymbol{v}}_L = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{\omega}_L \\ \boldsymbol{v}_L \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega} \times \boldsymbol{\omega}_L \\ \boldsymbol{\omega} \times \boldsymbol{v}_L + \boldsymbol{v} \times \boldsymbol{\omega}_L \end{bmatrix}. \tag{8.12}$$

We are naturally interested in the linear component of Eq. 8.12:

$$\begin{aligned}&\boldsymbol{\omega}\times\boldsymbol{v}_L+\boldsymbol{v}\times\boldsymbol{\omega}_L\\&=(\boldsymbol{\omega}_\lambda+\boldsymbol{\omega}_L)\times\boldsymbol{v}_L+(\boldsymbol{v}_\lambda+\boldsymbol{\omega}_\lambda\times\boldsymbol{r}+\boldsymbol{v}_L)\times\boldsymbol{\omega}_L\\&=\boldsymbol{\omega}_\lambda\times\boldsymbol{v}_L+\boldsymbol{\omega}_L\times\boldsymbol{v}_L+\boldsymbol{v}_\lambda\times\boldsymbol{\omega}_L+(\boldsymbol{\omega}_\lambda\times\boldsymbol{r})\times\boldsymbol{\omega}_L+\boldsymbol{v}_L\times\boldsymbol{\omega}_L,\end{aligned}\tag{8.13}$$

which by the relation $\boldsymbol{\omega}_L\times\boldsymbol{v}_L=-\boldsymbol{v}_L\times\boldsymbol{\omega}_L$ simplifies to:

$$\boldsymbol{\omega}_\lambda\times\boldsymbol{v}_L+\boldsymbol{v}_\lambda\times\boldsymbol{\omega}_L+(\boldsymbol{\omega}_\lambda\times\boldsymbol{r})\times\boldsymbol{\omega}_L.\tag{8.14}$$

Finally, in order to return to the EOM in Eq. 8.9, let us multiply the interesting component of the spatial acceleration by the inertia matrix. By the definition of spatial inertia about an arbitrary point (Sec. 5.3.1F) we obtain:

$$\begin{aligned}&\widehat{\boldsymbol{I}}(\widehat{\boldsymbol{v}}\times\widehat{\boldsymbol{v}}_L)=\\&=\begin{bmatrix}(\boldsymbol{I}_C+m\boldsymbol{c}\times\boldsymbol{c}\times^T)(\boldsymbol{\omega}\times\boldsymbol{\omega}_L)+m\boldsymbol{c}\times(\boldsymbol{\omega}_\lambda\times\boldsymbol{v}_L+\boldsymbol{v}_\lambda\times\boldsymbol{\omega}_L+(\boldsymbol{\omega}_\lambda\times\boldsymbol{r})\times\boldsymbol{\omega}_L)\\m\boldsymbol{c}\times^T(\boldsymbol{\omega}\times\boldsymbol{\omega}_L)+m(\boldsymbol{\omega}_\lambda\times\boldsymbol{v}_L+\boldsymbol{v}_\lambda\times\boldsymbol{\omega}_L+(\boldsymbol{\omega}_\lambda\times\boldsymbol{r})\times\boldsymbol{\omega}_L)\end{bmatrix},\end{aligned}\tag{8.15}$$

which shows that the expression in Eq. 8.14 gets scalar-multiplied by the mass and thus preserves its form which turns out very convenient in further analysis.

Let us now proceed to the second term on the RHS of Eq. 8.9: similarly to Euler's equation (Eq. 3.2b), it represents the variation of the inertia tensor due to the fact that the frame of reference is moving with velocity $\widehat{\boldsymbol{v}}$. By the definitions of spatial cross product and spatial inertia about an arbitrary point we get:

$$\widehat{\boldsymbol{v}}\times^*\widehat{\boldsymbol{I}}\widehat{\boldsymbol{v}}=\begin{bmatrix}\boldsymbol{\omega}\times\boldsymbol{I}_C\boldsymbol{\omega}+\boldsymbol{\omega}\times m\boldsymbol{c}\times\boldsymbol{c}\times^T\boldsymbol{\omega}+\boldsymbol{\omega}\times m\boldsymbol{c}\times\boldsymbol{v}\\\boldsymbol{\omega}\times m\boldsymbol{c}\times^T\boldsymbol{\omega}+\boldsymbol{\omega}\times m\boldsymbol{v}\end{bmatrix}.\tag{8.16}$$

Both angular and linear components are $\boldsymbol{\omega}\times$ times the corresponding momentum vector about the pivot point which is obviously correct but not very informative in the context of fictitious forces. What is important in our current analysis is the $\boldsymbol{\omega}\times\boldsymbol{v}$ term which is clearly the *Coriolis term* in Featherstone's nomenclature (Sec. 5.3.1C). Let us further disassemble the cross-product:

$$\begin{aligned}&\boldsymbol{\omega}\times\boldsymbol{v}=(\boldsymbol{\omega}_\lambda+\boldsymbol{\omega}_L)\times(\boldsymbol{v}_\lambda+\boldsymbol{\omega}_\lambda\times\boldsymbol{r}+\boldsymbol{v}_L)\\&=\boldsymbol{\omega}_\lambda\times\boldsymbol{v}_\lambda+\boldsymbol{\omega}_\lambda\times(\boldsymbol{\omega}_\lambda\times\boldsymbol{r})+\boldsymbol{\omega}_\lambda\times\boldsymbol{v}_L+\boldsymbol{\omega}_L\times\boldsymbol{v}_\lambda+\\&\quad+\boldsymbol{\omega}_L\times(\boldsymbol{\omega}_\lambda\times\boldsymbol{r})+\boldsymbol{\omega}_L\times\boldsymbol{v}_L.\end{aligned}\tag{8.17}$$

The $\boldsymbol{\omega}_\lambda\times\boldsymbol{v}_\lambda$ term is simply the very same expression but evaluated for the parent of the analyzed body, which means that the formula is recursive; $\boldsymbol{\omega}_\lambda\times(\boldsymbol{\omega}_\lambda\times\boldsymbol{r})$ is obviously the radial/centripetal acceleration; finally, $\boldsymbol{\omega}_L\times\boldsymbol{v}_L$ and $\boldsymbol{\omega}_\lambda\times\boldsymbol{v}_L$ are Coriolis accelerations due to the rotation of the analyzed body's and its parent's coordinate frames, respectively (without the factor 2, though). The two remaining terms do not have an obvious interpretation. However, since in the

EOM (Eq. 8.9) both the expression in Eq. 8.17 and Eq. 8.14 get only scalar-multiplied by the mass, we can simply add them which results in reduction of these two troublesome terms:

$$\begin{aligned} &\boldsymbol{\omega}_\lambda \times \boldsymbol{v}_\lambda + \boldsymbol{\omega}_\lambda \times (\boldsymbol{\omega}_\lambda \times \boldsymbol{r}) + \boldsymbol{\omega}_\lambda \times \boldsymbol{v}_L + \boldsymbol{\omega}_L \times \boldsymbol{v}_\lambda + \\ &+ \boldsymbol{\omega}_L \times (\boldsymbol{\omega}_\lambda \times \boldsymbol{r}) + \boldsymbol{\omega}_L \times \boldsymbol{v}_L + \boldsymbol{\omega}_\lambda \times \boldsymbol{v}_L + \boldsymbol{v}_\lambda \times \boldsymbol{\omega}_L + (\boldsymbol{\omega}_\lambda \times \boldsymbol{r}) \times \boldsymbol{\omega}_L \\ &= \boldsymbol{\omega}_\lambda \times \boldsymbol{v}_\lambda + \boldsymbol{\omega}_\lambda \times (\boldsymbol{\omega}_\lambda \times \boldsymbol{r}) + 2\boldsymbol{\omega}_\lambda \times \boldsymbol{v}_L + \boldsymbol{\omega}_L \times \boldsymbol{v}_L. \end{aligned} \tag{8.18}$$

The only fact that may seem surprising is the Coriolis acceleration whose form is clearly incompatible with the Eq. 8.7: the factor of 2 has appeared but only w.r.t. the angular velocity of the parent while we are left with the single *dangling* term w.r.t. to the analyzed body's local angular velocity. The explanation lies in the fact that we are using a coordinate frame that is traveling with the body and so the distance between the body and the axis of its *local rotation* does not change[7]. As a result, Coriolis acceleration due to local rotation relative to the parent has only one term since the second contribution does not exist: the factor 2 vanishes.

As one can see, there is a lot of coupling in these formulae and one cannot easily eliminate the individual fictitious terms in general, but in the particular case of free-body simulation most of these terms vanish: if we realize that all parent velocities are zero and that the spatial inertia matrix is expressed at the center of mass the only term we are left with is $\boldsymbol{\omega}_L \times \boldsymbol{v}_L$. Therefore, we propose to leave the articulated algorithm unchanged and once the unaltered accelerations have been established simply traverse all the bodies and correct their accelerations by adding the term $\boldsymbol{\omega}_L \times \boldsymbol{v}_L$ effectively making the body-local coordinate frames be inertial but still sharing the momentary orientation of the corresponding body. We can now integrate these modified accelerations into velocities, correct them if there are any additional constraints, update configuration and then rotate the velocity vectors so that their coordinates correspond to the updated frames.

### 8.3.1C Applying the resolution - results

The results of applying the modifications in the spinning cube test scenario are presented in Fig. 8.6. As we can see, although the coordinates of the velocity vector are continuously changing, the undesired non-physical growth of its magnitude has been eliminated.

We are now ready to re-run the box stack scenario. The resulting snapshots provided in Fig. 8.7 prove that the proposed resolution solved the problem - the simulation is now stable.

[7] To be precise: using the convention that origin of the frame is at the current location of body pivot - this distance is always zero

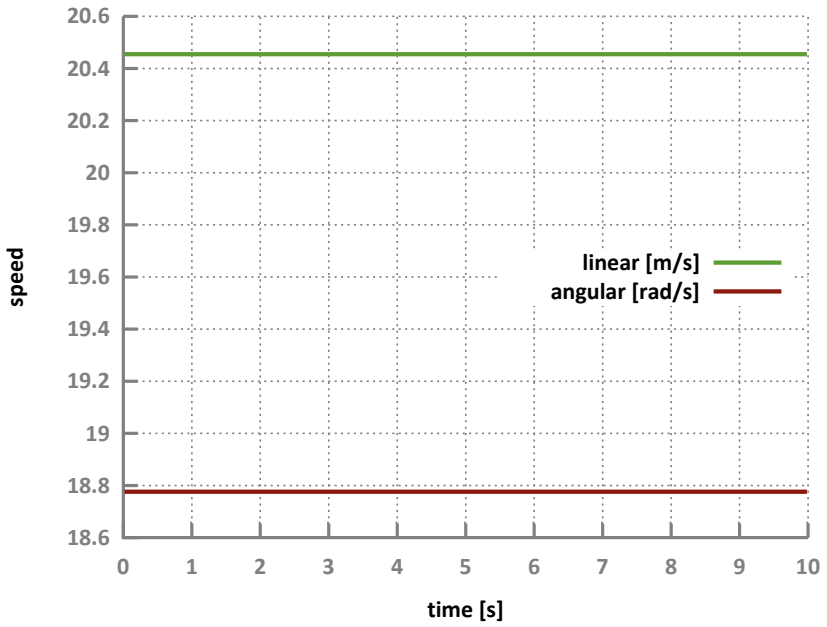



Figure 8.6: Simulation of a spinning cube after Coriolis-term correction has been applied: maximal speed in the system (linear and angular) in consecutive simulation steps

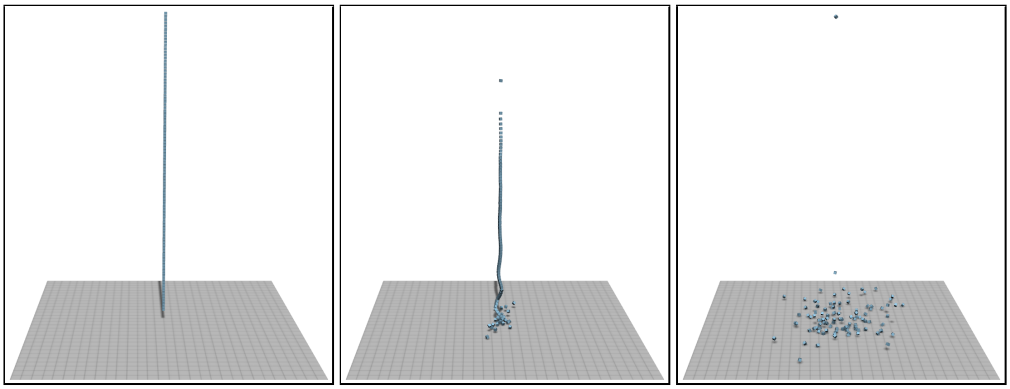
### 8.3.2 Symplectic Euler integration

The isolated scenario that will be used to introduce the problem and help us trace its origins will be the 5-link pendulum model introduced in the previous chapter (Sec. 7.5.3). This may seem surprising since a pendulum forms a kinematic chain so in the proposed algorithm it is simulated simply by applying CRBA and then solving the equation of motion without having to worry about additional constraints and velocity correction. However, it turns out that the basic articulated system simulation is very unstable which is illustrated using the previously introduced maximal speed graph (Fig. 8.8).

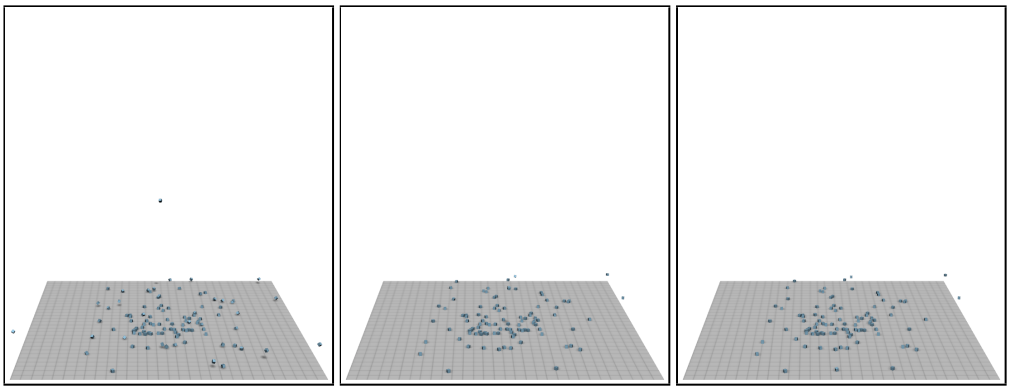
#### 8.3.2A Tracing the source of the problem

Simulation of similar models is perfectly stable in the acceleration-level simulator presented in [43] and - as long as there are no loops or contacts (which is the case) - the currently discussed approach is identical. The only difference is the integration procedure: fourth order Runge-Kutta versus symplectic Euler which means this is what we need to focus attention on. Simply applying RK4 instead of Euler is not really an option since we are aiming at interactive simulation so repeated solver calls per step need to be avoided.

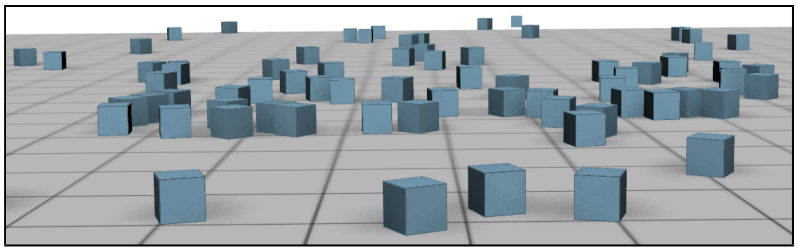
Seeing RK4 being more stable than Euler integrator should not be surprising at all, however: symplectic Euler is known to be a computationally cheap way of



(a) 0.0s (b) 3.0s (c) 6.0s



(d) 9.0s (e) 12.0s (f) 15.0s



(g) 15.0s. zoomed in

Figure 8.7: Simulation of 100 falling cubes after Coriolis-term correction has been applied

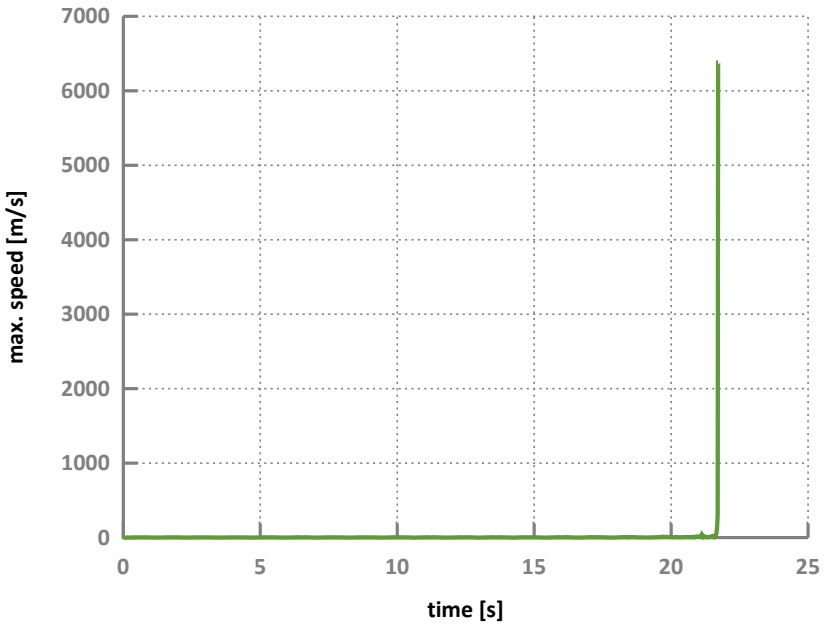



Figure 8.8: Unstable simulation of a 5-link pendulum: maximal speed in the system in consecutive simulation steps

introducing stability and should rather damp than boost velocities; what is more - we have seen Sequential Impulse successfully simulate pendulums using this integration scheme[8]. Compared to SI, there are obviously multiple differences but the most basic one is the fact of using reduced coordinates along with body-local coordinate frames. This will be the second point of our focus in the forthcoming analysis.

Let us see what happens when the symplectic Euler integrator is applied to a pendulum represented using reduced coordinates and local frames:

1. we start with the vector containing configuration scalars (angles) of each segment, $\boldsymbol{q}^i$, and a corresponding velocity, $\dot{\boldsymbol{q}}^i$ (superscript identifies the simulation step)

2. for each body, RNEA computes total angular and linear velocity vectors (body-local coordinates), $\boldsymbol{\omega}^i$, $\boldsymbol{v}^i$, respectively, which are then used to determine the contents of vector $\boldsymbol{C}^i$

3. generalized inertia matrix is determined using system configuration information *only*

4. the resulting EOM is solved for $\ddot{\boldsymbol{q}}^i$:

$$\boldsymbol{H}\ddot{\boldsymbol{q}}^i = -\boldsymbol{C}^i \tag{8.19}$$

[8]Reminder: SI damped pendulum's motion.

5. accelerations get Euler-integrated into velocity vector:

$$\dot{\boldsymbol{q}}^{i+1} = \dot{\boldsymbol{q}}^{i} + \ddot{\boldsymbol{q}}^{i}\Delta t \tag{8.20}$$

6. new velocities get Euler-integrated into configuration vector:

$$\boldsymbol{q}^{i+1} = \boldsymbol{q}^{i} + \dot{\boldsymbol{q}}^{i+1}\Delta t \tag{8.21}$$

As we can see in point 2.: RNEA needs to reevaluate the angular and linear velocities in each step since they are given implicitly by the system topology, pivot locations, the current configuration vector and finally - current joint-space velocity. In an inertial frame - only acceleration can change velocity; however: we are dealing with body-local non-inertial frames which means that changing configuration will, in general, influence the total velocities *seen* by RNEA. Therefore, what happens in point 6. effectively changes the velocities, although the $\dot{\boldsymbol{q}}$ remains untouched.

Does that mean that the reduced coordinates algorithms using local-frames produce improper results by definition? Not at all: the motion of the coordinate frames which in turn modifies the locally/non-inertially perceived total velocities is compensated by Coriolis acceleration terms. The thing is that they are evaluated from velocities at the instant $i$, and symplectic Euler updates the configuration using the velocities at instant $i+1$.

The facts that this issue is naturally non-existent for SI and that it is obviously related to the application of symplectic Euler integrator, make it a good candidate for being the original cause of the observed instabilities.

### 8.3.2B Proposed resolution

Considering the coupling of fictitious terms concluded from derivations in the previous section (Sec. 8.3.1B) we wished to minimize the modifications in the original algorithm and chose a more tractable approach. The proposed resolution is to keep the absolute $i+1$ velocity vectors expressed in a base/world frame[9] evaluated in point 5. so that in point 1. of the next simulation step they can be recast into body-local coordinate frames while still representing the same total velocities.

For floating articulated systems, one can additionally apply the Coriolis-related modification presented in the context of free-body simulation (Sec. 8.3.1B) since every tree root is actually a free body. The stage where free-body accelerations are altered remains the same, but the final rotation of velocity vectors is no longer needed since we actually do the same thing by storing the global absolute velocities across the simulation step. Non-free roots will naturally generate zero

[9] Any fixed frame would do, but the base frame seems to be a natural choice.

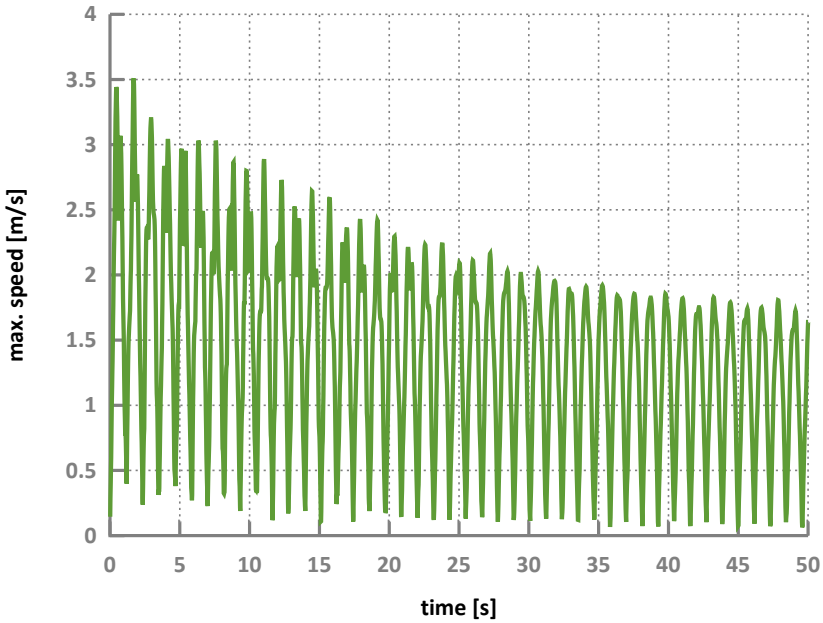



Figure 8.9: Simulation of a spinning cube after global-velocity correction has been applied: maximal speed in the system in consecutive simulation steps

correction term, but it is more efficient to simply skip them by checking inboard joint types[10].

### 8.3.2C Applying the resolution - results

As one can see in Fig. 8.9, the pendulum is no longer unstable: the velocity build-up process does not appear. In fact, apparently the range of fluctuations of the maximal speed in the system is decreasing which means that the that the velocities are damped. This is similar to to the equivalent experiment performed using SI (i.e. maximal coordinates) but the damping it now visibly stronger. The problem is further discussed in Sec. 8.7.1.

## 8.4 Articulated Islands Algorithm

Although it is obviously correct to conduct free-body simulations using an underlying articulation-oriented algorithm, it makes little to no sense to do the additional processing it introduces while the same could be achieved with far less computation. What we need is a way of switching between the articulated and free-body versions of SI.

[10]Whether it is a 6DoF joint or not.

### 8.4.1 Common dynamics algorithm

This part of the fusion is not that difficult to achieve: we can simply pick the childless hierarchy root free-bodies and pass them to the free-body algorithm, while the rest is treated as individual trees and passed to the articulated procedure.

### 8.4.2 Common constraint processing

Common processing of the constraints requires more thought, but thanks to the fact that we have already used SI-like processing for trees rather than the exact PGS makes the fusion much simpler to conduct and in most aspects boils down to additional bookkeeping.

The differences between free-body and tree processing are:

- trees require body Jacobian evaluation, free-bodies do not
- constraint Jacobians map from joint-space velocity in the case of trees while for free-bodies they always map from a 6D spatial velocity vector
- effective mass determination for trees requires solving an EOM while for free-bodies it is evaluated directly from the inverse of (transformed) spatial inertia matrix
- the current velocity increment is evaluated from a joint-space vector for trees and from spatial velocity of the constrained bodies (one or two)
- each impulse application affects the entire joint-space velocity vector of a tree, but only a single body spatial velocity in the case of free-bodies

This differences cannot be fused and need to be performed differently if the currently processed constraint a subtree/subtrees, a free-body/bodies or one of each. However, apart from that - the constraint processing procedure is perfectly compatible and allows for combining these two types of objects in a single simulation.

The resultant algorithm will act as SI if the simulated scenario is limited to free-bodies and as Articulated SI if there are kinematic trees only. Otherwise - the jointed linkages can be perceived as *articulated islands* simulated along with other such islands and free bodies there for we will name this method the *Articulated Islands Algorithm* (AIA).

## 8.5 Free-body simulations

There is no real point in performing simulations involving only free bodies since they are handed by the SI-potion of AIA which has already been tested in the previous chapter.

| Parameter | Value | Unit |
|---|---|---|
| initial altitude (pelvis) | 1.5 | m |
| initial velocity | $x = 0.0$ $y = 0.0$ $z = 0.0$ | m/s |
| initial ang. velocity | $x = 0.0$ $y = 0.0$ $z = 0.0$ | rad/s |
| simulation step | 1/60 | s |
| iteration count | 20 | - |

Table 8.2: Simulation parameters and initial conditions

## 8.6 Articulated systems simulations

In the case of articulated systems, the topology is obviously no longer flat: each tree in the system has its root body *stemming* from a 6DoF *connection* to the world/basis; the root is a starting point of the topological which can conveniently presented using a graph. The locations of body frames' origins are now chosen in joint positions/pivots, which is the natural setting for CRBA[11].

During the tentative step, individual trees are handled separately, i.e. their EOMs are solved in isolation, inertia matrices (which are submatrices of the overall system inertia matrix) are pre-factorized and stored for prospective usage in the constraint-processing stage. Note that the previously discussed free-body scenarios are just a special case of articulated simulation.

The possible interactions can occur between:

- a tree body and immobile environment/world
- two bodies in a single tree
- two bodies in two different trees

All these cases are naturally handled by the articulated SI algorithm and are going to tested in the forthcoming sections.

### 8.6.1 Rag-doll

#### 8.6.1A Model

The first articulated scenario is the 8-link rag-doll introduced in the previous chapter (Sec. 7.5.4) only this time the articulation is introduced using reduced-coordinate representation - the corresponding graph is presented in Fig. 8.10.

[11] Using body-local frames is the reason why RNEA (as proposed in [83]) has gained recognition for efficiency [36, 42].

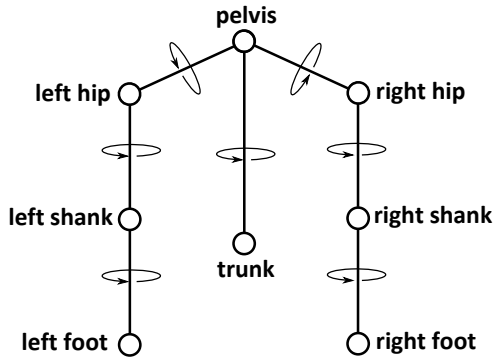
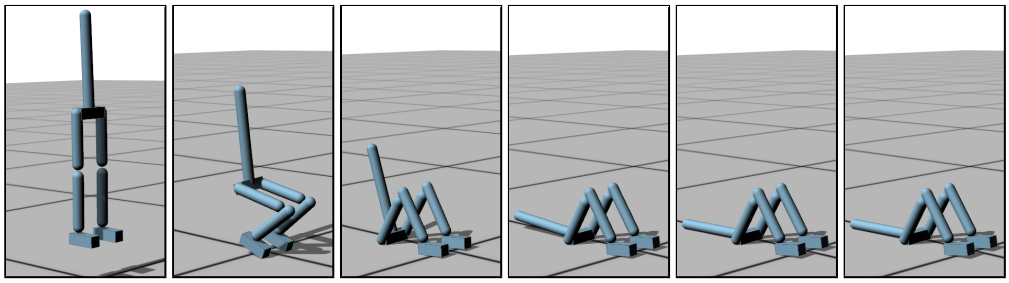



Figure 8.10: Graph illustrating the topology of an 8-link rag-doll. Vertices represent bodies, edges represent joints (*bent* arrow identifies a hinge joint).



(a) 0.0s (b) 1.0s (c) 2.0s (d) 3.0s (e) 4.0s (f) 5.0s

Figure 8.11: Simulation of a single 8-link falling rag-doll

#### 8.6.1B Results

Simulation snapshots provided in Fig. 8.11 show that the simulation is stable and that non-penetration constraints are not violated (within visual tolerance). Joints are obviously error-free due to their reduced-coordinate representation.

### 8.6.2 Rag-doll stack

#### 8.6.2A Model

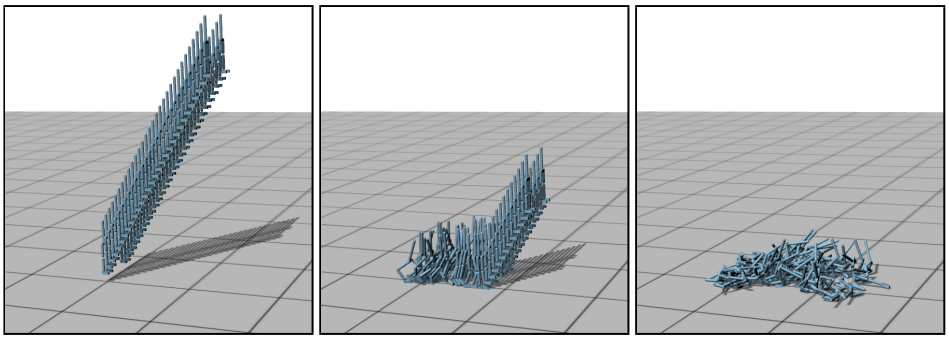
Similarly to the free-body case, we shall now increase the complexity of the simulation by stacking multiple instances of the basic model: the rag-doll. To obtain a high body count but avoid building a very high column, we shall divide it into two slanted ones of 30 rag-dolls each. This gives a total of 480 bodies.

#### 8.6.2B Results

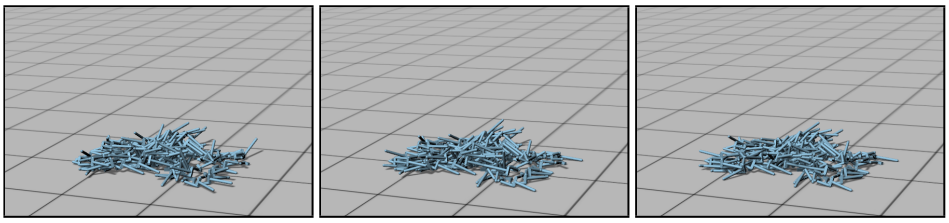
Again, the simulation is stable and the resulting animation seems natural. Snapshots are presented in Fig. 8.12.

| Parameter | Value | Unit |
|---|---|---|
| initial altitude (bottom pelvis) | 1.5 | m |
| initial offset | $x = 0.15$ $y = 0.0$ $z = 0.75$ | m |
| initial velocity | $x = 0.0$ $y = 0.0$ $z = 0.0$ | m/s |
| initial ang. velocity | $x = 0.0$ $y = 0.0$ $z = 0.0$ | rad/s |
| simulation step | 1/60 | s |
| iteration count | 20 | - |

Table 8.3: Simulation parameters and initial conditions



(a) 0.0s (b) 1.0s (c) 2.0s



(d) 3.0s (e) 4.0s (f) 10.0s

Figure 8.12: Simulation of a 60 8-link falling rag-dolls

### 8.6.3 Loaded net

#### 8.6.3A The model

Net model is the same as the one tested in the previous chapter (Sec. 7.5.6) but this time it is built of articulated open-loop *subnets*, each of them constructed using reduced-coordinate representation. Hinge constraints are used to create the looped net from open-loop subnets (Fig. 8.13). Cubes that are dropped onto the net are obviously free bodies. Similarly to the SI test (Sec. 7.5.6), we shall skip the parameters' table and only state that the simulation step is again 1/60 and that the iteration count is set to 20.

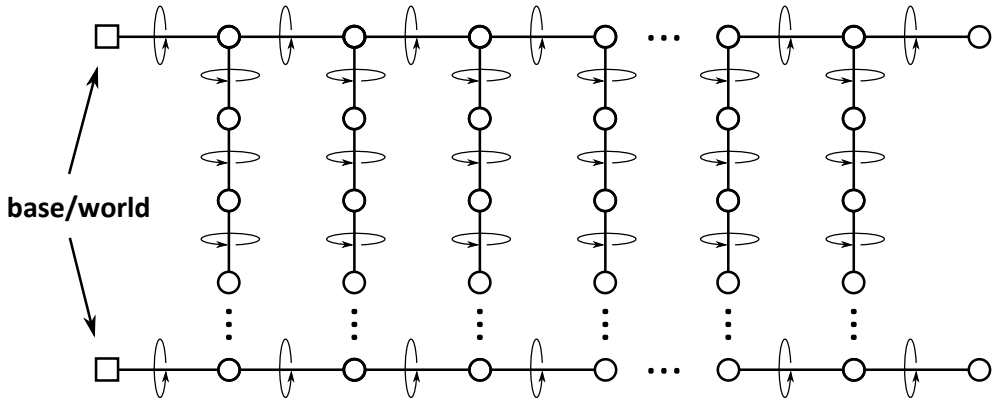



Figure 8.13: Graph illustrating the topology of subtrees creating the net model (there are connected into *one whole* by additional hinge constraints)

### 8.6.4 Results

Snapshots of the simulation are presented in Fig. 8.14. We can see that the simulation is stable and the produced motion believable though visibly different from the one produced by SI (Sec. 7.5.6B) which is a good example of what visual plausibility is: in real world there would obviously be only a single *correct* course of actions[12], but we can easily accept two different animations as natural-looking.

An easily noticeable difference between the animation produced by SI and AIA is that in the former case more cubes stay on the net: imperfect hinge constraints allow for (hopefully small) violations corrected using Baumgarte stabilization which, as we now, acts as a damped spring. As a result, the net created using hinge constraints is in general more *springy* than the one built from articulated subnets for which neither joint errors nor spring-like stabilization occurs.

### 8.6.5 Tracked vehicle

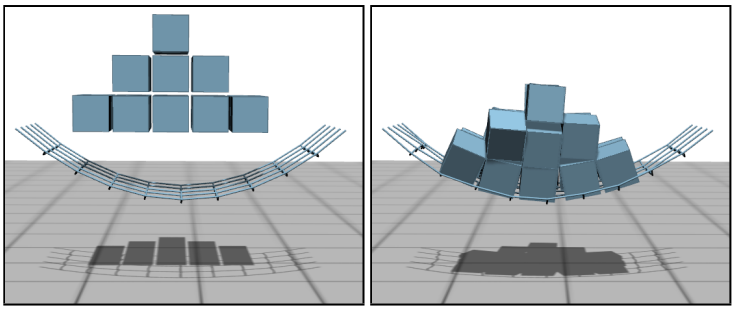
The single remaining scenario is the tracked vehicle which has also been used to test SI in the previous chapter (Sec. 7.5.7)..

#### 8.6.5A The model

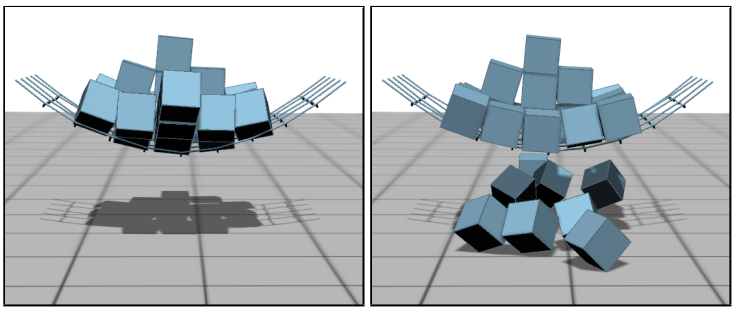
On each side, the tank has one actuated drive sprocket and a set of unactuated *wheels*: three road wheels, an idler and optional return rollers (Fig. 7.8). Each wheel rolls on the caterpillar track which prevents its direct contact with the ground. The model has been constructed as follows (Fig. 8.15):

- hull is a cuboid and is free to move with respect to an immobile base of the hierarchy

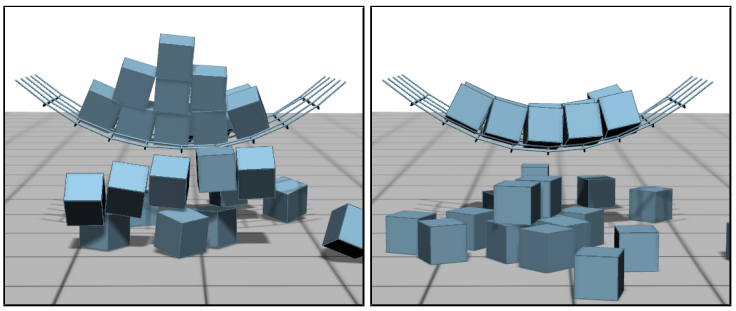
[12] Although repeating identical conditions for a mechanical experiment of this kind would be obviously hard.



(a) 0.0s (b) 1.0s



(c) 2.0s (d) 3.0s



(e) 4.0s (f) 15.0s

Figure 8.14: Simulation of a net with 27 cubes dropped onto it

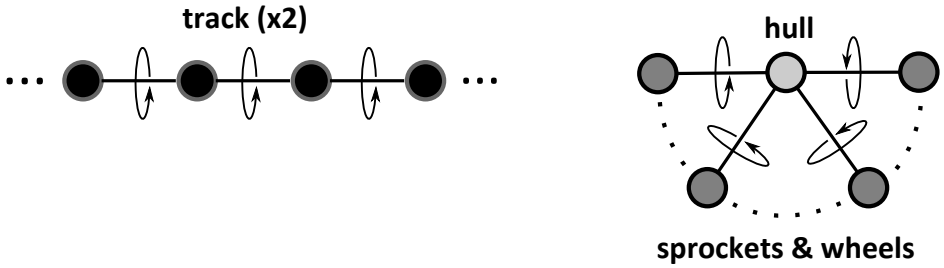



Figure 8.15: Tank system topology. Color-coding of bodies is the same as in Fig. 7.8.

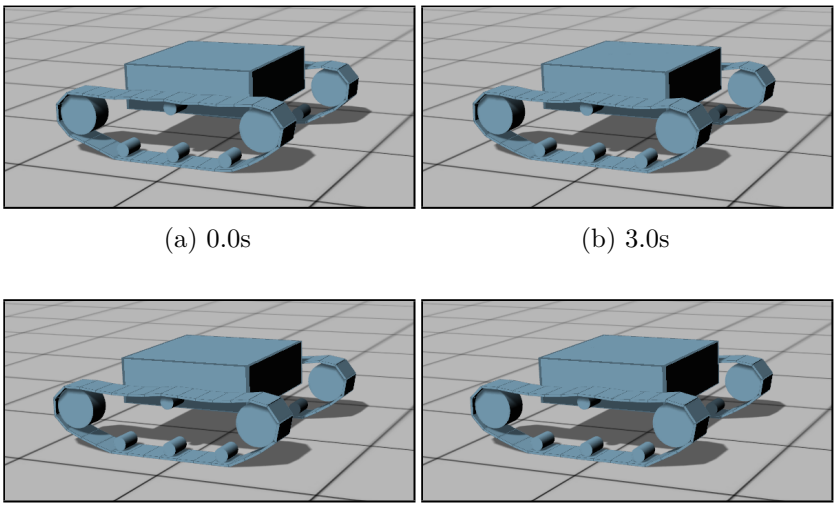
- track chains are built from flat cuboids linked by 1DoF hinge joints
- wheels are attached to the hull using 1DoF hinge joints
- each track chain is *looped* by an additional hinge constraint between the first and last link
- each track is *attached* to the hull by a single planar constraint (relative motion limited to a single plane)
- contact between the actuated wheels and the track is modeled using 2D non-penetration constraints (bilateral in tangent direction)
- contact between the unactuated wheels and the track is modeled using 2D non-penetration constraints (Coulomb friction)
- contact between the track links and the ground is modeled using 3D non-penetration constraints (Coulomb friction)

Bilateral tangential component of the contact constraint between the sprockets and the track models the meshed sprocket-track connection. The planar constraint between the hull and each track is a global equivalent of *real-life* guide horns (see Sec. 7.5.7). This planar constraints eliminates lateral motion of tracks relative to the hull which allowed allowed to use simplified (2D rather than 3D) constraints between the wheels and the track, which resulted in smaller overall constraints' dimension. This is an obvious advantage over the infinite lateral friction approach we had to use in the maximal-coordinate representation[13].

[13]Theoretically, one could apply the very same approach in the SI test but chains creating the tracks are much less *stiff* when using maximal coordinates to model them and a single planar constraint was not enough to prevent their lateral motion w.r.t. the hull. One could probably solve the problem at the cost of yet more iterations. Reduced-coordinate tracks are clearly more fit for the task.

| Parameter | Value | Unit |
|---|---|---|
| initial altitude (bottom track plates) | 0.0 | m |
| initial velocity (hull) | $x = 0.0$ $y = 0.0$ $z = 0.0$ | m/s |
| initial ang. velocity (hull) | $x = 0.0$ $y = 0.0$ $z = 0.0$ | rad/s |
| simulation step | 1/60 | s |
| iteration count | 20 | - |

Table 8.4: Simulation parameters and initial conditions



(a) 0.0s (b) 3.0s

(c) 6.0s (d) 9.0s

Figure 8.16: Simulation of a tracked vehicle

#### 8.6.5B Results

Unlike in the experiment with SI, this time 20 iterations was perfectly enough to keep the tracks intact: the simulation (snapshots in Fig. 8.16) is stable and the constraints are sustained within visual tolerance.

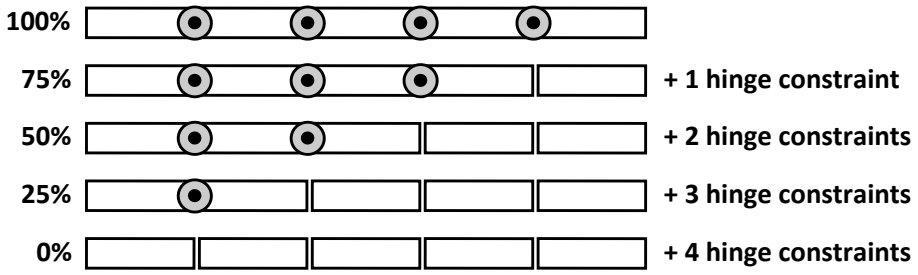



Figure 8.17: The chains used for execution-time tests; starting from the top: four reduced-coordinate joints (100 %), three reduced coordinate joints (75%) and a single additional hinge constraint, two reduced coordinate joints (50%) and two additional hinge constraints, one reduced coordinate joint (25%) and three additional hinge constraints, no reduced-coordinate joints (0%) and four additional hinge constraints. The *percentages* will be used to identify the corresponding test cases.

| Parameter | Value | Unit |
|---|---|---|
| initial altitude (bottom chain) | 2.0 | m |
| initial velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | m/s |
| initial ang. velocity | $x = 0.0 \quad y = 0.0 \quad z = 0.0$ | rad/s |
| simulation step | 1/60 | s |
| iteration count | 20 | - |

Table 8.5: Simulation parameters and initial conditions

### 8.6.6 Performance assessment

To properly assess the effect of using the proposed reduced-coordinate model of joints on performance (understood as a reciprocal of frame execution time), we shall conduct a series of tests designed solely for the purpose. Appropriately parameterized dedicated test scenarios will bring out and thus make it easier to grasp the key trilateral relation between the overall constraints dimension, the ratio between joints modeled using maximal and reduced coordinates and the frame execution time.

The proposed test scenario involves capsules hinged into 5-link chains and dropped onto a flat ground. If one applies coordinate reduction to all the joints in the system, then each such chain forms an articulated structure and the additional constraints are used only to prevent inter-penetration. If coordinate reduction is not used at all, then all joints and contact/collisions are treated as additional constraints augmenting a system of free bodies. Between these two extremes, we shall make each chain be composed by a combination of reduced-coordinate joints and additional constraints (Fig. 8.17).

The results of the test performed for a set of 100, 300 and 500 bodies are

presented in Figures 8.18, 8.19 and 8.20, respectively, where the normalized[14] frame execution times are plotted against the total dimension of constraints in the system (which includes additional constraints and reduced-coordinate joints). There main conclusion one can draw having inspected those figures the case when all joints are introduced by coordinate reduction ("100 %" case) is clearly leading but it is also apparent that its advantage over the perfectly opposite ("0%" case) approach is decreasing with the growing number of simulated bodies.

Let us analyze that observation. The main advantage (in terms of execution times) of using reduced-coordinate joints is the smaller number of constraints which results in less preparatory (Sec. 7.4.2 vs Sec. 8.2.2) and velocity-correction processing (Sec. 7.4.3 vs Sec. 8.2.3). However, performing the CRBA-factorize-solve triplet for a 5-body chain/subtree takes (in general) more CPU time than evaluating accelerations of the five constituent bodies independently and while execution times of these operations are related by a constant factor (for a fixed number of bodies and degrees of freedom), performing these operations more times (more chains in the scene) increases the *difference* in the total execution time. The same applies to the relation between solving the factorized SOLE and a simple matrix multiplication[15] in the preparatory stage. Furthermore, higher chain count means there are potentially more body Jacobians to evaluate - the cost absent when using maximal-coordinates.

Summarizing, the per-subtree cost of operations characteristic for the reduced-coordinate approach is greater than the *price* (in terms of execution time) paid for each additional constraint dimension and thus increasing the subtree count is advantageous for the maximal-coordinate formulation when comparing frame execution times. Nonetheless, we can see that there is wide variety of simulation scenarios for which the introduction of reduced-coordinate joints substantially reduces frame execution times, i.e. improves performance.

## 8.7 Limitations and further challenges

Although all the presented test scenarios were successfully simulated there have been few specific cases when the proposed method failed to provide visual plausibility or stability. As of now, solutions to these problems have not been found and thus they are isolated and described in this section.

[14] It simply means that all the frame execution times are divided by the highest measured value.

[15] It obviously depends on the implementation as well as system and matrix dimensions but in general this claim is valid.

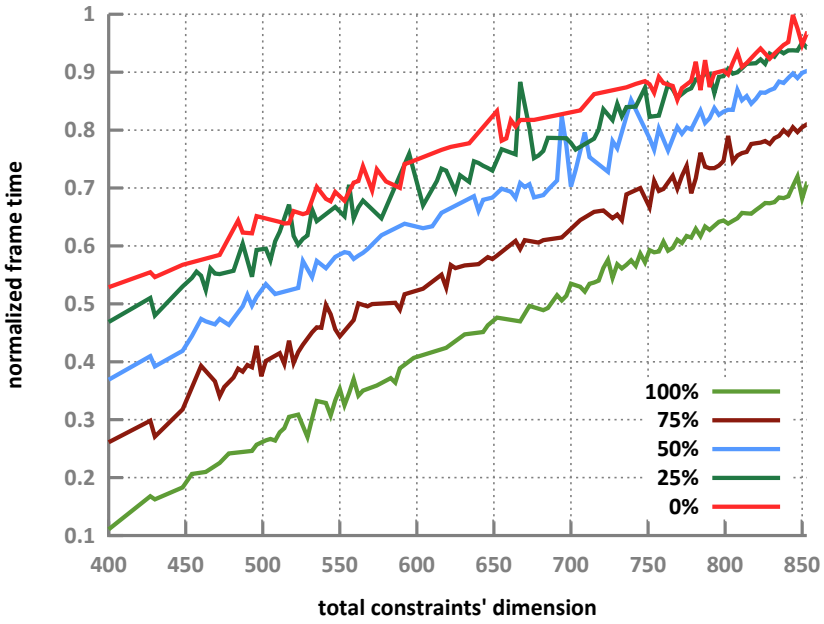



Figure 8.18: Normalized frame execution time plotted against the total dimension of the constraints in the system of **100** bodies connected into 20 5-link chains; different curves illustrate the dependence for the five test cases presented in Fig. 8.17

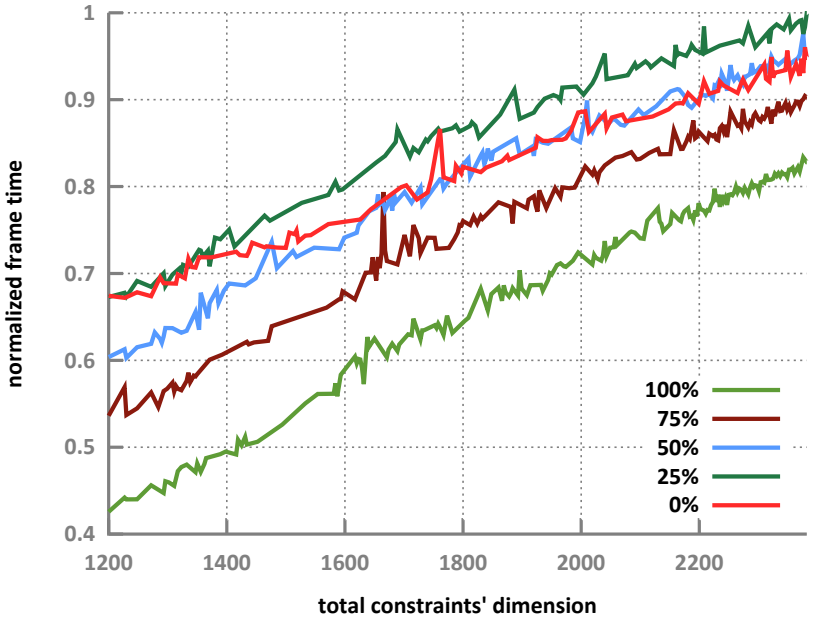



Figure 8.19: Normalized frame execution time plotted against the total dimension of the constraints in the system of **300** bodies connected into 60 5-link chains; different curves illustrate the dependence for the five test cases presented in Fig. 8.17

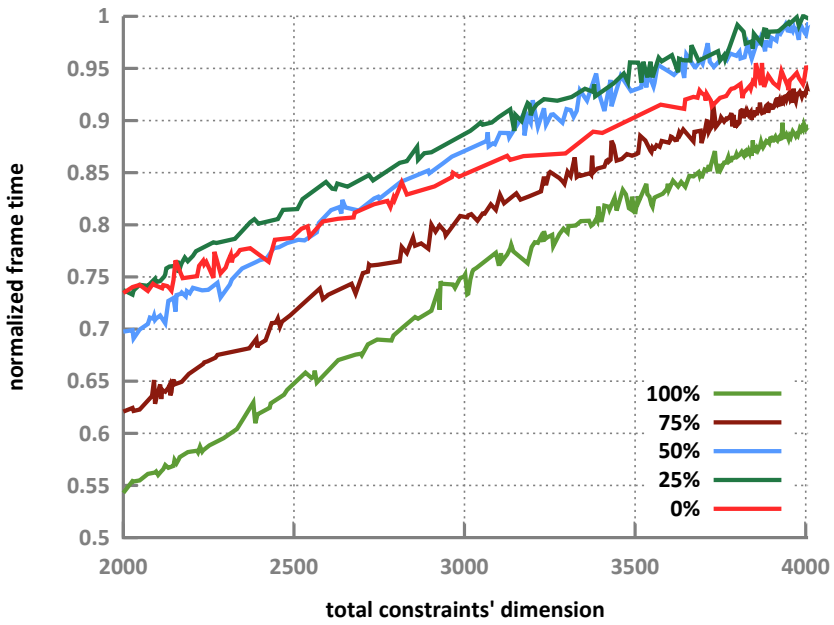



Figure 8.20: Normalized frame execution time plotted against the total dimension of the constraints in the system of **500** bodies connected into 100 5-link chains; different curves illustrate the dependence for the five test cases presented in Fig. 8.17

### 8.7.1 Damping

The first problem was discovered when resolving the problem of applying symplectic Euler integration in local coordinates which is described in Sec. 8.3.2A. As a result of the proposed resolution, the problem of instability was solved but at the same time visible damping was introduced which is clearly *not* introduced by the underlying model and its intensity makes it difficult to ignore[16].

One could obviously blindly assume that this phenomenon is caused by the symplectic Euler integrator[17], but we have seen that the same 5-link pendulum model simulated using SI (i.e. maximal coordinates) is damped much less (Sec. 7.5.3B). However, one could further argue that the energy artificially injected into the maximal-coordinate model by the Baumgarte constraint stabilization balances (accidentally and to a certain extent only) the damping of the applied integrator scheme, an effect absent in the reduced-coordinate simulation[18], but the author has not been able to prove it so far.

[16] Mind you, we are aiming at visual plausibility so *minor* imperfections are often ignored.
[17] And thus not really solvable under the restriction of using that integrator scheme.
[18] As long as there are no additional constraints modeled by explicitly applied reaction forces which is the case of a swinging pendulum scenario.

### 8.7.2 Inadequate velocity correction

Another problem is that the velocity of a point on body, which is an element of a larger chain or tree, evaluated using:

$$\boldsymbol{J}\dot{\boldsymbol{q}}, \tag{8.22}$$

where $\boldsymbol{J}$ is the Jacobian of this body point and $\dot{\boldsymbol{q}}$ is the joint-space velocity of the chain/tree, poorly approximates the corresponding translation which is particularly painful for high angular velocities. The effect is amplified by the simple numerical integration scheme which is, however, a must in an interactive physics engine. Now, if one inspects the velocity correction phase of the proposed method (Proc. 8.2), Jacobian-based mapping from joint-space to constraint-space velocity is the only way solver can evaluate the current velocity error and apply corrective impulses accordingly. Therefore, if the constraint-space velocity is evaluated imprecisely, the precision of the entire correction stage is jeopardized.

A possible solution to this problem could be a modified constraint-space velocity evaluation method, e.g.:

- after the current corrective impulse has been applied and thus the joint-space velocity increment, $\Delta\dot{\boldsymbol{q}}$, has been updated, tentatively integrate $\Delta\dot{\boldsymbol{q}}$ into the configuration vector and use it to determine the predicted post-step position of the constrained body-point
- use the current and predicted positions of the constrained-body point to evaluate the actual[19] velocity increment (simple approximation using difference quotient would be *compatible* with the Euler integration scheme)
- map the evaluated velocity increment to constraint-space

The above proposition is but a rough sketch and needs much refining and verification, but even at this point it is clear that it would require more CPU-time to process than the existing method.

## 8.8 Summary

The presented results prove that the Articulated Islands Algorithm is a viable simulation method. Naturally, the fact of introducing reduced-coordinate joints eliminates the configuration error which would occur in corresponding additional constraints but it has also been shown to be capable of improving performance by reducing the amount of necessary computations or iteration count needed for convergence.

[19] Actual in this context means *not* evaluated by Jacobian-based mapping from joint-space velocity increment.

Nevertheless, AIA in its current form has its limitations and must be applied with caution because: (i) its performance can be worse than SI for very large systems; (ii) due to imprecise constraint-space velocity evaluation, AIA is more prone to instability caused by high angular velocities.

# Chapter 9

# Conclusions and plans

Development of the forward dynamics algorithm combining reduced-coordinate joint models with an existing interactive simulator utilizing maximal coordinates (SI) required solving several non-trivial problems:

- determination of inertial properties of individual bodies in articulated subsystems of the simulated scene
- handling the problems related to resolving the system dynamics in moving coordinate frames
- unified representation and dual processing of the constraints which enabled fine-grain integration of the two formulations resulting in the possibility of employing the optimal approach for free-body and articulated-systems simulations

Once the above mentioned issues have been addressed and the proposed method implemented and thoroughly tested, it seems justified to claim that obtained results confirm the thesis of the dissertation since the Articulated Islands Algorithm, which has been created as a direct low-level extension of SI, has been proven to offer comparable visual plausibility of generated animations. Moreover, certain problems of the basic SI algorithm encountered when simulating articulated structures have been either eliminated or reduced in the proposed method which resulted in:

- zero configuration error for the joints within the kinematic tree
- less constraints to process which often allows to reduce the time spent on determining reaction impulses
- less iterations needed for convergence

Obviously, the author does not claim that AIA is simply *better* than SI: the intention of this dissertation was to prove that fine-grain compatibility between reduced- and maximal-coordinate formulations can be achieved in a contemporary physics engine and that there are numerous simulation scenarios in which such a fusion is beneficial.

Nevertheless, there are still certain problems that require further analysis, i.e. the artificial damping and imprecise mapping from the joint-space to constraint-space velocity. Apart from eliminating these problems, the author plans to further improve the proposed method by experimenting with solver warm-starting, dynamically switched system topologies and developing multi-core versions of the described algorithms.

# Bibliography


[1] Mihai Anitescu, Florian A. Potra A, and David E. Stewart. Time-stepping for three-dimensional rigid body dynamics. *Computer methods in applied mechanics and engineering*, 177(3):183–197, 1999. [cited at p. 40, 43]

[2] Mihai Anitescu and Florian A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14(3):231–247, 1997. [cited at p. 18, 36, 37, 40, 46, 58, 59]

[3] G. Aruldhas. Classical mechanics, 2008. [cited at p. 25, 26]

[4] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *SIGGRAPH Comput. Graph.*, 23(3):223–232, July 1989. [cited at p. 50]

[5] David Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10(2-4):292–352, 1993. [cited at p. 13, 16, 18, 24, 37, 50]

[6] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 23–34, New York, NY, USA, 1994. ACM. [cited at p. 16, 45, 46, 50]

[7] David Baraff. Interactive simulation of solid rigid bodies. *Computer Graphics and Applications, IEEE*, 15(3):63–75, 1995. [cited at p. 43]

[8] David Baraff. Linear-time dynamics using lagrange multipliers. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 137–146, New York, NY, USA, 1996. ACM. [cited at p. 31, 33, 43, 68, 83]

[9] Ronen Barzel, John R. Hughes, and Daniel N. Wood. Plausible motion simulation for computer graphics animation. In Ronan Boulic and Gerard Hégron, editors, *Computer Animation and Simulation '96*, Eurographics, pages 183–197. Springer Vienna, 1996. [cited at p. 6]

[10] Millard F. Beatty. *Principles of Engineering Mechanics: Volume 2: Dynamics - The Analysis of Motion*, volume 33. Springer, 2005. [cited at p. 25]

[11] Jan Bender. Impulse-based dynamic simulation in linear time. *Computer Animation and Virtual Worlds*, 18(4-5):225–233, 2007. [cited at p. 46, 85]

[12] Jan Bender, Kenny Erleben, Jeff C. Trinkle, and Erwin Coumans. Interactive simulation of rigid body dynamics in computer graphics. *STAR Proceedings of Eurographics*, 2012. [cited at p. 26]

[13] Jan Bender, Dieter Finkenzeller, and Alfred Schmitt. An impulse-based dynamic simulation system for vr applications. In *Proceedings of Virtual Concept*, volume 2005, 2005. [cited at p. 41, 46, 65, 85]

[14] Jan Bender and Alfred Schmitt. Constraint-based collision and contact handling using impulses. In *Proceedings of the 19th international conference on computer animation and social agents*, pages 3–11, 2006. [cited at p. 44, 46, 65]

[15] Jan Bender and Alfred Schmitt. Fast dynamic simulation of multi-body systems using impulses. In *VRIPHYS*, pages 81–90, 2006. [cited at p. 44, 46, 85]

[16] Stephen G. Berard. *Using simulation for planning and design of robotic systems with intermittent contact*. PhD thesis, Rensselaer Polytechnic Institute, 2009. [cited at p. 24, 30]

[17] Wojciech Blajer. A geometric unification of constrained system dynamics. *Multibody System Dynamics*, 1(1):3–21, 1997. [cited at p. 31]

[18] Wojciech Blajer, Krzysztof Dziewiecki, Krzysztof Kołodziejczyk, and Zenon Mazur. Inverse dynamics of underactuated mechanical systems: a simple case study and experimental verification. *Communications in Nonlinear Science and Numerical Simulation*, 16(5):2265–2272, 2011. [cited at p. 30]

[19] Wojciech Blajer and Krzysztof Kołodziejczyk. A geometric approach to solving problems of control constraints: theory and a dae framework. *Multibody System Dynamics*, 11(4):343–364, 2004. [cited at p. 30]

[20] Bernard Brogliato. *Nonsmooth Mechanics: Models, Dynamics and Control*. Springer, 1999. [cited at p. 34, 35, 171, 172]

[21] Bernard Brogliato, AA Ten Dam, Laetitia Paoli, Frank Génot., and Michel Abadie. Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. *Applied Mechanics Reviews*, 55(2):107–150, 2002. [cited at p. 16, 30, 34, 35, 36, 37, 42, 50, 56, 174]

[22] Milton A. Chace and P. N. Sheth. Adaptation of computer techniques to the design of mechanical dynamic machinery. ASME Paper 73-DEPT-58, 1973. [cited at p. 67]

[23] Anindya Chatterjee. On the realism of complementarity conditions in rigid body collisions. *Nonlinear Dynamics*, 20(2):159–168, 1999. [cited at p. 18, 19]

[24] Anindya Chatterjee and Andy Ruina. A new algebraic rigid-body collision law based on impulse space considerations. *Transactions-American Society Of Mechanical Engineers Journal Of Applied Mechanics*, 65:939–951, 1998. [cited at p. 20, 21]

[25] Anindya Chatterjee and Andy Ruina. Two interpretations of rigidity in rigid-body collisions. *Transactions-American Society Of Mechanical Engineers Journal Of Applied Mechanics*, 65:894–900, 1998. [cited at p. 19, 21]

[26] Steve Cheng. *A Crash Course on the Lebesgue Integral and Measure Theory*. Published online, 2008. [cited at p. 171, 172]

[27] Michael B. Cline and Dinesh K. Pai. Post-stabilization for rigid body simulation with contact and constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA '03*, volume 3, pages 3744–3751, 2003. [cited at p. 31, 42, 43, 179]

[28] Ramon Costa-Castelló, Robert Griñó, and Luis Basañez. Dae methods in constrained robotics system simulation. *Computación y Sistemas*, 1(3), 1998. [cited at p. 29, 30]

[29] Richard W. Cottle, Jong-Shi Pang, and Richard E. Stone. *The Linear Complementarity Problem*, volume 60 of *Classics in Applied Mathematics*. SIAM, 2009. [cited at p. 36, 55]

[30] Charles Coulomb. Théorie des machines simples, 1785. [cited at p. 15]

[31] John J. Craig. *Introduction to Robotics: Mechanics and Control (3rd Edition)*. Prentice Hall, 2005. [cited at p. 28, 68, 170]

[32] Evan Drumwright. A fast and stable penalty method for rigid body simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 14(1):231–240, 2008. [cited at p. 23, 24]

[33] Kenny Erleben. *Stable, robust, and versatile multibody dynamics animation*. PhD thesis, University of Copenhagen, Copenhagen, 2004. [cited at p. 5, 6, 7, 8, 26, 40, 43, 61, 89, 178]

[34] Roy Featherstone. The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research*, 2(1):13–30, 1983. [cited at p. 67, 78]

[35] Roy Featherstone. *Robot dynamics algorithms*. PhD thesis, University of Edinburgh Scotland, 1984. [cited at p. 67]

[36] Roy Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987. [cited at p. 67, 68, 70, 78, 139]

[37] Roy Featherstone. 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):867–875, 1999. [cited at p. 81]

[38] Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics. part 2: Trees, loops, and accuracy. *The International Journal of Robotics Research*, 18(9):876–892, 1999. [cited at p. 81]

[39] Roy Featherstone. Efficient factorization of the joint-space inertia matrix for branched kinematic trees. *The International Journal of Robotics Research*, 24(6):487–500, 2005. [cited at p. 77]

[40] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, 2008. [cited at p. 25, 26, 27, 28, 31, 43, 69, 70, 74, 77, 79, 81, 88, 97, 167]

[41] Roy Featherstone and Oussama Khatib. Load independence of the dynamically consistent inverse of the jacobian matrix. *International Journal of Robotics Research*, 16(2):168–170, 1997. [cited at p. 79]

[42] Roy Featherstone and David Orin. Robot dynamics: Equations and algorithms. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 826–834. IEEE, 2000. [cited at p. 139]

[43] Jakub Stępień. A unified constraint framework for physical animation of articulated rigid bodies. In *Motion in Games 2012*, pages 90–101. Springer, 2012. [cited at p. 51, 98, 99, 119, 133, 179]

[44] Martin Förg, Friedrich Pfeiffer, and Heinz Ulbrich. Simulation of unilateral constrained systems with many bodies. *Multibody System Dynamics*, 14(2):137–154, 2005. [cited at p. 35]

[45] David Heaver Fremlin. *Measure theory*, volume 4. Torres Fremlin, 2000. [cited at p. 171]

[46] Russell Gayle, Ming C. Lin, and Dinesh Manocha. Adaptive dynamics with efficient contact handling for articulated robots. In *Robotics: Science and Systems*, pages 231–238, 2006. [cited at p. 81]

[47] Matthias Gerdts. Optimal control of ordinary differential equations and differential-algebraic equations, 2006. Habilitation thesis. [cited at p. 29, 30, 31]

[48] Dipak Chandra Ghosh, Nripesh Chandra Ghosh, and Prabir Kumar Haldar. *Engineering Physics*. Firewall Media, 2008. [cited at p. 27]

[49] Ch Glocker. Formulatin of spatial contact situations in rigid multibody systems. *Computer Methods in Applied Mechanics and Engineering*, 177(3):199–214, 1999. [cited at p. 16]

[50] Christoph Glocker. Scalar force potentials in rigid multibody systems. *Courses and Lectures - International Centre for Mechanical Sciences*, pages 69–146, 2000. [cited at p. 16, 17]

[51] Christoph Glocker. *Set-valued force laws: dynamics of non-smooth systems*, volume 1. Springer-Verlag, 2001. [cited at p. 171, 172]

[52] Christoph Glocker and Friedrich Pfeiffer. *Multibody dynamics with unilateral contacts*. Wiley series in nonlinear science. John Wiley, 1996. [cited at p. 16, 17, 36, 39]

[53] Jean-Paul Gourret, Nadia Magnenat-Thalmann, and Daniel Thalmann. Simulation of object and human skin formations in a grasping task. *SIGGRAPH Comput. Graph.*, 23(3):21–30, July 1989. [cited at p. 23]

[54] Donald T. Greenwood. *Classical Dynamics*. Dover Publications, 1997. [cited at p. 14]

[55] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Transactions on Graphics (TOG)*, 22(3):871–878, 2003. [cited at p. 46, 65, 174]

[56] James K. Hahn. Realistic animation of rigid bodies. *SIGGRAPH Comput. Graph.*, 22(4):299–308, June 1988. [cited at p. 60, 61, 174]

[57] Paul R. Halmos. *Measure theory*, volume 4. Springer-Verlag, 1974. [cited at p. 172]

[58] Takahiro Harada, Masayuki Tanaka, Seiichi Koshizuka, and Yoichiro Kawaguchi. Real-time rigid body simulation using gpus. *IPSJ SIG Technical Reports*, 2007(13):79–84, 2007. [cited at p. 24]

[59] Shoichi Hasegawa and Makoto Sato. Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects. *Computer Graphics Forum*, 23(3):529–538, 2004. [cited at p. 23]

[60] E. J. Haug, D. Negrut, and C. Engstler. Implicit runge-kutta integration of the equations of multibody dynamics in descriptor form. *Mechanics of Structures and Machines*, 27(3):337–364, 1999. [cited at p. 30, 31]

[61] Edward J. Haug, Shih C. Wu, and Shih M. Yang. Dynamics of mechanical systems with coulomb friction, stiction, impact and constraint addition-deletion—I Theory. *Mechanism and Machine Theory*, 21(5):401–406, 1986. [cited at p. 45]

[62] William B. Heard. *Rigid body mechanics*. Wiley, 2008. [cited at p. 30, 33]

[63] W.W. Hooker and G. Margulies. The dynamical attitude equations for an n-body satellite. *Journal of Astronautical Sciences*, 12:123–128, 1965. [cited at p. 67]

[64] Abhinandan Jain. *Robot and Multibody Dynamics: Analysis and Algorithms*. Springer, 2011. [cited at p. 14, 68]

[65] Javier Garcia De Jalon and Eduardo Bayo. *Kinematic and dynamic simulation of multibody systems*. Springer-Verlag New York, USA, 1994. [cited at p. 28, 30, 43, 47]

[66] Waseem Ahmad Khan, Venkat N. Krovi, Subir Kumar Saha, and Jorge Angeles. Modular and recursive kinematics and dynamics for parallel manipulators. *Multibody System Dynamics*, 14(3-4):419–455, 2005. [cited at p. 78]

[67] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of*, 3(1):43–53, 1987. [cited at p. 31, 79, 88]

[68] Sujeong Kim, Stephane Redon, and Young J. Kim. View-dependent dynamics of articulated bodies. *Computer Animation and Virtual Worlds*, 19(3-4):223–233, 2008. [cited at p. 81]

[69] Evangelos Kokkevis. An index reduction method in holonomic system dynamics. In K.J. Bathe, editor, *Second MIT Conference on Compurational Fluid and Solid Mechanics*, 2003. [cited at p. 30, 33]

[70] Evangelos Kokkevis. Practical physics for articulated characters. In *Game Developers Conference*, volume 2004, 2004. [cited at p. 89, 95, 96]

[71] Evangelos Kokkevis, Dimitri Metaxas, and Norman I. Badler. User-controlled physics-based animation for articulated figures. In *Computer Animation'96. Proceedings*, pages 16–26. IEEE, 1996. [cited at p. 23]

[72] Evangelos Kokkevis and Dimitris Metaxas. Efficient dynamic constraints for animating articulated figures. *Multibody system dynamics*, 2(2):89–114, 1998. [cited at p. 95]

[73] Andrei Nikolaevich Kolmogorov and Sergei Vasilievitch Fomin. *Introductory real analysis*. Courier Dover Publications, 1975. [cited at p. 171, 172]

[74] Thomas R. Kurfess. *Robotics and automation handbook*. CRC press, 2010. [cited at p. 30]

[75] Taesoo Kwon and Jessica Hodgins. Control systems for human running using an inverted pendulum model and a reference motion capture sequence. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 129–138, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association. [cited at p. 24]

[76] Claude Lacoursiere. Splitting methods for dry frictional contact problems in rigid multibody systems: Preliminary performance results. In *Conference Proceedings from SIGRAD2003*, pages 11–16. Citeseer, 2003. [cited at p. 89]

[77] Benoit Lafleur, Nadia Magnenat-Thalmann, and Daniel Thalmann. Cloth animation with self-collision detection. In *Modeling in Computer Graphics*, pages 179–187. Springer, 1991. [cited at p. 23]

[78] Cornelius Lanczos. *The variational principles of mechanics*, volume 4. Dover Publications, 1970. [cited at p. 13, 28]

[79] Kathryn W. Lilly. *Efficient dynamic simulation of robotic mechanisms*. Kluwer Academic Boston, 1993. [cited at p. 51, 95, 97]

[80] Stephen Longshaw. *Numerical Modelling And Visualization Of The Evolution Of Extensional Fault Systems*. PhD thesis, The University of Manchester, 2011. [cited at p. 11, 178]

[81] Per Lötstedt. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM Journal on Applied Mathematics*, 42(2):281–296, 1982. [cited at p. 16, 35, 45]

[82] Per Lötstedt. Numerical simulation of time-dependent contact and friction problems in rigid body mechanics. *SIAM journal on scientific and statistical computing*, 5(2):370–393, 1984. [cited at p. 16, 45, 46]

[83] John Y. S. Luh, Michael W. Walker, and R. P. C. Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76, 1980. [cited at p. 67, 74, 139]

[84] Adriano Macchietto, Victor Zordan, and Christian R. Shelton. Momentum control for balance. *ACM Trans. Graph.*, 28(3):80:1–80:8, July 2009. [cited at p. 24]

[85] Manuel Duque Pereira Monteiro Marques. *Differential inclusions in nonsmooth mechanical problems: Shocks and dry friction*. Birkhäuser Boston, 1993. [cited at p. 46]

[86] Brian Mirtich. Hybrid simulation: combining constraints and impulses. In *Proceedings of First Workshop on Simulation and Interaction in Virtual Environments*, 1995. [cited at p. 41, 46, 61]

[87] Brian Mirtich. *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California, 1996. [cited at p. 8, 19, 21, 24, 41, 42, 46, 51, 61, 62, 92, 95, 96, 174]

[88] Brian Mirtich. Timewarp rigid body simulation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 193–200. ACM Press/Addison-Wesley Publishing Co., 2000. [cited at p. 23]

[89] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 181–ff. ACM, 1995. [cited at p. 21, 46, 61, 62, 63, 64]

[90] Jorge Cortés Monforte. *Geometric, Control and Numerical Aspects of Nonholonomic Systems*, volume 1793 of *Lecture Notes in Mathematics*. Springer-Verlag, 2002. [cited at p. 27]

[91] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *SIGGRAPH Comput. Graph.*, 22(4):289–298, June 1988. [cited at p. 23]

[92] Jean Jacques Moreau. Facing the plurality of solutions in nonsmooth mechanics. *Nonsmooth/Nonconvex Mechanics with Applications in Engineering, II. NNMAE*, 2006:3–12, 2006. [cited at p. 57]

[93] Jean Jacques Moreau and Panagiotis D. Panagiotopoulos. *Nonsmooth Mechanics and Its Applications*, volume 302. Springer, 1988. [cited at p. 37, 46, 173, 174]

[94] Katta G. Murty. *Linear complementarity, linear and nonlinear programming*. Heldermann Berlin, 1988. [cited at p. 36]

[95] Binh Nguyen. *Locally Non-Convex Contact Models And Solution Methods For Accurate Physical Simulation In Robotics*. PhD thesis, Rensselaer Polytechnic Institute, 2011. [cited at p. 10, 182]

[96] Nicolae Orlandea, Milton A. Chace, and Donald A. Calahan. A sparsity-oriented approach to the dynamic analysis and design of mechanical systems - part 1. *Transaction of ASME Journal of Engiennering for Industry*, 99(3):773–779, 1977. [cited at p. 67]

[97] Zbigniew Osiński. *Mechanika Ogólna*. Wydawnictwo Naukowe PWN, 2000. [cited at p. 14]

[98] Jong-Shi Pang and Jeff C. Trinkle. Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with coulomb friction. *Mathematical Programming*, 73(2):199–226, 1996. [cited at p. 16]

[99] B. Paul. Analytical dynamics of mechanisms–a computer oriented overview. *Mechanisms and Machine Theory*, 10:481–507, 1975. [cited at p. 67]

[100] Linda Petzold. Differential/algebraic equations are not ode's. *SIAM Journal on Scientific and Statistical Computing*, 3(3):367–384, 1982. [cited at p. 29, 30]

[101] Friederich Pfeiffer and Christoph Glocker, editors. *Multibody Dynamics with Unilateral Contacts*. Number 421 in CISM Courses and Lectures. Springer-Verlag, 2000. [cited at p. 15]

[102] Friedrich Pfeiffer. Unilateral problems of dynamics. *Archive of Applied Mechanics*, 69(8):503–527, 1999. [cited at p. 34]

[103] Friedrich Pfeiffer. The idea of complementarity in multibody dynamics. *Archive of Applied Mechanics*, 72(11-12):807–816, 2003. [cited at p. 34, 38]

[104] Scott R. Ploen, Fred Y. Hadaegh, and Daniel P. Scharf. Rigid body equations of motion for modeling and control of spacecraft formations. part 1: Absolute equations of motion. In *American Control Conference, 2004. Proceedings of the 2004*, volume 4, pages 3646–3653. IEEE, 2004. [cited at p. 14]

[105] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. *SIGGRAPH Comput. Graph.*, 25(4):349–358, July 1991. [cited at p. 23]

[106] Stephane Redon, Nico Galoppo, and Ming C. Lin. Adaptive dynamics of articulated bodies. *ACM Trans. Graph.*, 24(3):936–945, July 2005. [cited at p. 81]

[107] R.E. Roberson and Jens Wittenburg. A dynamical formalism for an arbitrary number of interconnected rigid bodies.with reference to the problem of satellite attitude control. In *Proceedings of the 3rd IFAC congress*, pages 46D.2–46D.9, 1966. [cited at p. 67]

[108] Reinhardt M. Rosenberg. *Analytical dynamics of discrete systems*. Plenum Press, 1977. [cited at p. 26]

[109] Diego Ruspini and Oussama Khatib. Collision/contact models for the dynamic simulation of complex environments. In *9th International Symposium of Robotics Research*, volume 97, pages 185–195. Citeseer, 1997. [cited at p. 93, 95]

[110] Jörg Sauer and Elmar Schömer. A constraint-based approach to rigid body dynamics for virtual reality applications. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 153–162. ACM, 1998. [cited at p. 40, 46]

[111] Alfred Schmitt and Jan Bender. Impulse-based dynamic simulation of multibody systems: numerical comparison with standard methods. *Proc. automation of discrete production engineering*, pages 324–329, 2005. [cited at p. 46]

[112] Ahmed A. Shabana. *Computational dynamics*. Wiley, 2009. [cited at p. 13, 28, 31, 32, 43]

[113] David Stewart. *Dynamics with Inequalities: Impacts and Hard Constraints*. SIAM, 2011. [cited at p. 17, 18, 178]

[114] David E Stewart. Rigid-body dynamics with friction and impact. *SIAM review*, 42(1):3–39, 2000. [cited at p. 18]

[115] David E. Stewart and Jeff C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673–2691, 1996. [cited at p. 37, 40, 44, 46, 53, 55, 59]

[116] David E. Stewart and Jeff C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 162–169. IEEE, 2000. [cited at p. 36, 43, 53]

[117] William J. Stronge. *Impact mechanics*. Cambridge university press, 2004. [cited at p. 20, 21]

[118] Demetri Terzopoulos and Kurt Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988. [cited at p. 23]

[119] Richard Tonge, Feodor Benevolenski, and Andrey Voroshilov. Mass splitting for jitter-free parallel rigid body simulation. *ACM Transactions on Graphics*, 31(4):105–112, 2012. [cited at p. 89, 112]

[120] Firdaus E. Udwadia and Robert E. Kalaba. *Analytical Dynamics: A New Approach*. Cambridge University Press, 1996. [cited at p. 14]

[121] John J. Uicker. *On the Dynamic Analysis of Spatial Linkages Using 4 by 4 Matrices*. PhD thesis, Northwestern University, 1965. [cited at p. 67]

[122] A.F. Vereshchagin. Computer simulation of the dynamics of complicated mechanisms of robot manipulators. *Engineering Cybernetics*, 12(6):65–70, 1974. [cited at p. 67]

[123] Michael. W. Walker and David. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *Transaction of ASME Journal of Dynamic Systems, Measurement and Control*, 104:205–211, 1982. [cited at p. 67, 75]

[124] Y-T Wang, Vijay Kumar, and Jacob Abel. Dynamics of rigid bodies undergoing multiple frictional contacts. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2764–2769. IEEE, 1992. [cited at p. 21]

[125] Rachel Weinstein. *Simulation and Control of Articulated Rigid Bodies*. PhD thesis, Stanford University, 2007. [cited at p. 43, 44]

[126] Rachel Weinstein, Joseph Teran, and Ronald Fedkiw. Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):365–374, May 2006. [cited at p. 41, 44, 46, 67, 85]

[127] Theodore J. Wendt. *Mixed Complementarity Formulations and Energy Balance in Dynamic Contact Problems*. PhD thesis, University of Iowa, 2008. [cited at p. 34]

[128] Jane Wilhelms. Toward automatic motion control. *Computer Graphics and Applications, IEEE*, 7(4):11–22, 1987. [cited at p. 23]

[129] Andrew Witkin and David Baraff. Physically based modeling: Principles and practice. *SIGGRAPH 2001 Course Notes*, 2001. [cited at p. 28]

[130] Katsu Yamane and Yoshihiko Nakamura. Stable penalty-based model of frictional contacts. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1904–1909. IEEE, 2006. [cited at p. 23]

# Appendices

# Appendix A

# Auxiliary derivations

## A.1 Differentiation in moving coordinates

Let us quote [40] in order to provide a short introduction to this section:

> The problem of differentiation in moving coordinates can be stated as follows: given the coordinate vector that represents a vector $\boldsymbol{v}$, find the coordinate vector that represents $\dot{\boldsymbol{v}}$. If the basis vectors are constants, then the solution is simply the componentwise derivative of the coordinate vector; otherwise, a formula is required that takes into account the motion of the basis vectors.

So the problem with such differentiation is that not only can the analyzed vector change in time, but also the coordinate system by means of which the coordinates of that vector are expressed. Both of these changes will influence the derivative and must be taken into account.

First, a distinction needs to be made between a (Cartesian) coordinate system/frame and frame of reference: the former defines a set or orthonormal vectors (axes) and an origin while the latter defines the point of observation of a system along with the information whether and how that point is moving. The choice of the point of observation (in general) influences the perceived characteristics of motion, i.e. modifies the vectors describing it; the change of a the coordinate frame merely *rephrases* the motion, i.e. describes the same vectors using different coordinates. The same perceived motion can be described in different coordinate frames and *vice versa*.

When describing motion in non-inertial frames of reference (i.e. one which is undergoing acceleration w.r.t. an inertial frame) it is customary/convenient to introduce an auxiliary inertial frame, which is typically assigned a dedicated, stationary coordinate frame. We shall adopt this convention, but a note needs to

be made that the dedicated stationary coordinate frame is not really necessary since one could always use one coordinate frame to describe motion in both inertial and non-inertial frame.

Let there be two frame of reference along with two Cartesian coordinate frames, $A$ and $B$. $A$ represents an inertial frame of reference, while $B$ represents a non-inertial one. Let the current axes of $B$ be given by unit vectors, $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3$, expressed in $A$ coordinates.

We are analyzing the coordinates of the point $\boldsymbol{x}$ represented in frames $A$ and $B$ by coordinate vectors $\boldsymbol{x}^A$ and $\boldsymbol{x}^B$, respectively. **Let us start with an assumption that origins of $A$ an $B$ are initially coincident and remain so** (i.e. $B$ is not translating in time w.r.t. $A$). Then:

$$\boldsymbol{x}^B = [x_1^B \; x_2^B \; x_3^B]^T, \tag{A.1}$$

and

$$\boldsymbol{x}^A = (x_1^B \boldsymbol{u}_1^A + x_2^B \boldsymbol{u}_2^A + x_3^B \boldsymbol{u}_3^A) = \sum_{j=1}^{3} x_j^B \boldsymbol{u}_j^A, \tag{A.2}$$

which can be perceived as a change of coordinates from the representation in coordinate frame $B$ to $A$ using a transformation matrix:

$$\boldsymbol{x}^A = \begin{bmatrix} \boldsymbol{u}_1^A & \boldsymbol{u}_2^A & \boldsymbol{u}_3^A \end{bmatrix} \boldsymbol{x}^B. \tag{A.3}$$

If the above equation is differentiated w.r.t. time, one obtains:

$$\frac{d\boldsymbol{x}^A}{dt} = \sum_{j=1}^{3} \frac{d' x_j^B}{dt} \boldsymbol{u}_j^A + \sum_{j=1}^{3} x_j^B \frac{d\boldsymbol{u}_j^A}{dt}. \tag{A.4}$$

Let us analyze terms of the above equation:

- $\frac{d\boldsymbol{x}}{dt}$ describes the total/absolute rate of change of $\boldsymbol{x}$ as time passes and this change, when expressed along the axes of frame $A$, is given by $\frac{d\boldsymbol{x}^A}{dt}$
- $\frac{d'\boldsymbol{x}^B}{dt} = [\frac{d' x_1^B}{dt} \; \frac{d' x_2^B}{dt} \; \frac{d' x_3^B}{dt}]^T$ is the componentwise rate of change of $\boldsymbol{x}$ coordinates perceived by the viewer moving with the frame of reference $B$ and expressed along axes of frame $B$[1]. It is a component-wise derivative valid only within frame $B$. If the coordinates $\boldsymbol{x}^B$ do not change in time, then $\frac{d'\boldsymbol{x}^B}{dt}$ is zero. Note that $\sum_{j=1}^{3} \frac{d' x_j^B}{dt} \boldsymbol{u}_j^A$ is the same apparent change of $\boldsymbol{x}$ coordinates perceived in frame of reference $B$ but expressed along axes of $A$
- if the frame $B$ is moving, then (in general) the coordinates representing the total/absolute $\boldsymbol{x}$ (e.g. $\boldsymbol{x}^A$) will change and this is given by the second term

[1] Note that it is denoted by $\frac{d'(\,)}{dt}$ to discriminate it from the absolute derivative $\frac{d(\,)}{dt}$.

which describes rate of change of the coordinate vectors (in $A$ coordinates) representing axes of $B$ as time passes. Their change will generate different coordinates $\boldsymbol{x}^A$ even if coordinates $\boldsymbol{x}^B$ remain the same. Note that if $\frac{d\boldsymbol{u}_j^A}{dt}$ is zero for each axis then frame $B$ is not rotating and the component-wise derivative of $\boldsymbol{x}^B$ is the total/absolute derivative.

From now on, we shall adopt Newton's notation for convenience and better readability, i.e.:

$$\frac{d\boldsymbol{x}^A}{dt} = \dot{\boldsymbol{x}}^A \quad \text{and} \quad \frac{d\boldsymbol{u}_j^A}{dt} = \dot{\boldsymbol{u}}_j^A \tag{A.5}$$

and for the derivative observed in the moving coordinate frame:

$$\frac{d'\boldsymbol{x}^B}{dt} = \mathring{\boldsymbol{x}}^B. \tag{A.6}$$

If the differentiation w.r.t. time is carried out once again, one obtains for the first term:

$$\frac{d}{dt}(\sum_{j=1}^{3} \frac{d'x_j^B}{dt}\boldsymbol{u}_j^A) = \sum_{j=1}^{3} \frac{d'^2x_j^B}{dt^2}\boldsymbol{u}_j^A + \sum_{j=1}^{3} \frac{d'x_j^B}{dt}\frac{d\boldsymbol{u}_j^A}{dt}, \tag{A.7}$$

and for the second term

$$\frac{d}{dt}(\sum_{j=1}^{3} x_j^B \frac{d\boldsymbol{u}_j^A}{dt}) = \sum_{j=1}^{3} \frac{d'x_j^B}{dt}\frac{d\boldsymbol{u}_j^A}{dt} + \sum_{j=1}^{3} x_j^B \frac{d^2\boldsymbol{u}_j^A}{dt^2}. \tag{A.8}$$

So, we can finally write:

$$\begin{aligned}\ddot{\boldsymbol{x}}^A &= \sum_{j=1}^{3} \mathring{\mathring{x}}_j^B \boldsymbol{u}_j^A + \sum_{j=1}^{3} \mathring{x}_j^B \dot{\boldsymbol{u}}_j^A + \sum_{j=1}^{3} \mathring{x}_j^B \dot{\boldsymbol{u}}_j^A + \sum_{j=1}^{3} x_j^B \ddot{\boldsymbol{u}}_j^A = \\ &= \sum_{j=1}^{3} \mathring{\mathring{x}}_j^B \boldsymbol{u}_j^A + 2\sum_{j=1}^{3} \mathring{x}_j^B \dot{\boldsymbol{u}}_j^A + \sum_{j=1}^{3} x_j^B \ddot{\boldsymbol{u}}_j^A.\end{aligned} \tag{A.9}$$

Returning to what we have assumed at the beginning:

- if we drop the assumption of the origins of $A$ and $B$ being coincident we need an additional translation transformation when evaluating $\boldsymbol{x}^A$ but the forms of derivatives remain the same
- if we drop the assumption that the origin of $B$ is fixed w.r.t. to $A$ then an additional velocity and acceleration term will appear in the formulae for $\dot{\boldsymbol{x}}^A$ and $\ddot{\boldsymbol{x}}^A$ (should $B$ be accelerating), respectively; the formula for $\boldsymbol{x}^A$ remains the same.

Since relaxing the assumptions does not substantially influence the derivations, we will keep the discussion narrowed down to the case of $B$ changing only its orientation w.r.t. A in time (i.e. $B$ is a rotating frame). Let us apply the

formula for a total/absolute derivative in a coordinate frame currently rotating with angular velocity $\boldsymbol{\omega}$:

$$\dot{\boldsymbol{c}}^B = \mathring{\boldsymbol{c}}^B + \boldsymbol{\omega}^B \times \boldsymbol{c}^B, \tag{A.10}$$

where $\boldsymbol{c}$ is any vector. Obviously, coordinate vectors in Eq. A.10 can be transformed to the coordinate frame $A$ yielding:

$$\dot{\boldsymbol{c}}^A = \mathring{\boldsymbol{c}}^A + \boldsymbol{\omega}^A \times \boldsymbol{c}^A. \tag{A.11}$$

The first term on the right hand side vanishes if the differentiated vector is fixed for an observer moving with $B$ (i.e. its apparent derivative in $B$ is zero). Note that this is the case for axes of $B$, so:

$$\dot{\boldsymbol{u}}_j^A = \boldsymbol{\omega}^A \times \boldsymbol{u}_j^A. \tag{A.12}$$

This relation can be proven by analyzing the derivative of rotation matrix presented in Eq. A.3 [31]. If time-differentiation is repeated we get:

$$\ddot{\boldsymbol{u}}_j^A = \dot{\boldsymbol{\omega}}^A \times \boldsymbol{u}_j^A + \boldsymbol{\omega}^A \times \dot{\boldsymbol{u}}_j^A = \dot{\boldsymbol{\omega}}^A \times \boldsymbol{u}_j^A + \boldsymbol{\omega}^A \times (\boldsymbol{\omega}^A \times \boldsymbol{u}_j^A). \tag{A.13}$$

If we apply the above to Eq. A.1 we obtain[2]:

$$\ddot{\boldsymbol{x}}^A = \sum_{j=1}^{3} \mathring{\mathring{x}}_j^B \boldsymbol{u}_j^A + 2\sum_{j=1}^{3} \mathring{x}_j^B \boldsymbol{\omega}^A \times \boldsymbol{u}_j^A + \sum_{j=1}^{3} x_j^B \dot{\boldsymbol{\omega}}^A \times \boldsymbol{u}_j^A + \sum_{j=1}^{3} x_j^B \boldsymbol{\omega}^A \times \boldsymbol{\omega}^A \times \boldsymbol{u}_j^A = \tag{A.14}$$

$$= \mathring{\mathring{\boldsymbol{x}}}^A + 2\boldsymbol{\omega}^A \times \mathring{\boldsymbol{x}}^A + \dot{\boldsymbol{\omega}}^A \times \boldsymbol{x}^A + \boldsymbol{\omega}^A \times \boldsymbol{\omega}^A \times \boldsymbol{x}^A.$$

What we have obtained is the absolute second derivative of $\boldsymbol{x}$ presented using terms describing its motion as perceived by an observer traveling with frame $B$ and expressed in the coordinates of coordinate frame $A$. As we know, the exact same thing could be done for coordinate frame $B$. No matter the choice of coordinates, the advantage of this representation is the fact that the apparent acceleration $\mathring{\mathring{\boldsymbol{x}}}$ appears explicitly and thus we can *switch places* and observe the motion while moving with the non-inertial frame:

$$\mathring{\mathring{\boldsymbol{x}}}^A = \ddot{\boldsymbol{x}}^A - 2\boldsymbol{\omega}^A \times \mathring{\boldsymbol{x}}^A - \dot{\boldsymbol{\omega}}^A \times \boldsymbol{x}^A - \boldsymbol{\omega}^A \times \boldsymbol{\omega}^A \times \boldsymbol{x}^A. \tag{A.15}$$

Apparently, it requires three acceleration terms which influence the change of coordinates of $\boldsymbol{x}$ but do not result from any *real* forces[3]. Therefore, in order to apply principles of dynamics in a non-inertial frame we need to introduce fictitious (or inertial) forces:

- Coriolis force which corresponds to the $2\boldsymbol{\omega}^A \times \mathring{\boldsymbol{x}}^A$ acceleration term
- Euler force which corresponds to the $\dot{\boldsymbol{\omega}}^A \times \boldsymbol{x}^A$ acceleration term
- centrifugal force which corresponds to the $\boldsymbol{\omega}^A \times \boldsymbol{\omega}^A \times \boldsymbol{x}^A$ acceleration term

[2] Cross product property $(r\boldsymbol{a}) \times \boldsymbol{b} = \boldsymbol{a} \times (r\boldsymbol{b}) = r(\boldsymbol{a} \times \boldsymbol{b})$ is used, where $r$ is any scalar.
[3] Unlike $\ddot{\boldsymbol{x}}^A$ which is **assumed** to result from the applied forces.

## A.2 Nonsmooth motion

Full comprehension of the forthcoming material requires background in the measure and set theories. The author himself has gained the basic theoretical knowledge necessary to digest and present these derivations mainly from three sources: [20, 51, 26].

### A.2.1 Configuration, velocities and accelerations in nonsmooth motion

The simple rule that the consecutive differentiation with respect to time takes one from system configuration through its velocity to the acceleration is obviously based on the assumption that the configuration and velocity is everywhere differentiable. While this is enough for continuous motion, it is no longer correct when dealing with nonsmooth motion resulting from e.g. collisions.

We are considering motion of an $n_{dof}$ degrees of freedom holonomic[4] system on a compact time interval $I = [t_A, t_E]$. Let us denote the instant of collision by $t_i$, and the velocities at the instant of collision, directly before and directly after it by $\boldsymbol{u}(t_i)$, $\boldsymbol{u}^-(t_i)$ and $\boldsymbol{u}^+(t_i)$, respectively. Now, while $\boldsymbol{u}^-(t_i)$ and $\boldsymbol{u}^+(t_i)$ have intuitive physical sense, $\boldsymbol{u}(t_i)$ is meaningless [51].

We can safely assume that the configuration should be a continuous function of time, $\boldsymbol{q} : I \to \mathbb{R}^{n_{dof}}$. However, we relax the restriction that $\boldsymbol{q}$ should be differentiable everywhere and make it differentiable *almost everywhere* which in turn yields its absolute continuity and thus the following holds for every interval $[t_k, t_l] \subset I$:

$$\boldsymbol{q}(t_l) - \boldsymbol{q}(t_k) = \int_{t_k}^{t_l} \dot{\boldsymbol{q}}(\tau) d\tau. \tag{A.16}$$

In order to enable a countable number of finite velocity jumps, velocities are assumed to be functions of bounded variations on $I$, i.e. instead of using the time-derivative of the generalized coordinate vector, $\dot{\boldsymbol{q}}$, as the velocity we introduce $\boldsymbol{u} : I \to \mathbb{R}^{n_{dof}}, \boldsymbol{u} \in BV(I, \mathbb{R}^{n_{dof}})$. Due to $\boldsymbol{u}$ being of bounded variations, the left limit $\boldsymbol{u}^+(t)$ exists at every point $t \in (t_A, t_E]$; so does the right limit $\boldsymbol{u}^-(t)$ at every $t \in [t_A, t_E)$. The points at which $\boldsymbol{u}$ is discontinuous (finite jumps) are denoted by the set $\{t_i\}$. Another important property of *every* $\boldsymbol{u}(t) \in BV$ is that it can be decomposed into three functions[5], each of them also of bounded variations, as follows:

$$\boldsymbol{u} = \boldsymbol{u}_L + \boldsymbol{u}_A + \boldsymbol{u}_C, \tag{A.17}$$

where [51]

- $\boldsymbol{u}_L$ is absolutely continuous ($\dot{\boldsymbol{u}}_L$ exists almost everywhere and $\dot{\boldsymbol{u}}_L \in L^1$)

[4] This simplification does not really change anything substantial for this section of the work, but will definitely make the formulae easier to read and closer to the referred sources.

[5] Lebesgue decomposition of a function of bounded variation [73, 45].

- $\boldsymbol{u}_A$ is a step function (i.e. a piecewise constant function with a countable number of discontinuities which implies that its derivative vanishes for $dt$-almost every $t$)
- $\boldsymbol{u}_C$ is singular (i.e. a continuous function whose derivative vanishes almost everywhere [73])

Thanks to the assumption of at most countable number of velocity jumps and since the Lebesgue integral is not affected by changing the values of the integrated function on at most countable set of points (their Lebesgue-measure is zero), one may choose any finite values of $\boldsymbol{u}$ at points in $\{t_i\}$ (where it is not defined) without changing the value of the integral. Therefore, the following holds:

$$\forall t \in I: \quad \boldsymbol{q}(t) = \boldsymbol{q}(t_A) + \int_{t_A}^{t} \boldsymbol{u}(\tau) d\tau. \tag{A.18}$$

Although $\dot{\boldsymbol{u}}$ exists for $[dt]$-almost every t (thanks to $\boldsymbol{u}$ being of bounded variations), this integration approach cannot be repeated for the velocities; due to the discontinuities of $\boldsymbol{u}$ the velocities cannot be determined by simply integrating $\dot{\boldsymbol{u}}$ - we would only obtain the absolutely continuous component, $\boldsymbol{u}_L$, because the step and singular functions do not contribute[6]:

$$\int_{[t_k,t_l]} \dot{\boldsymbol{u}} d\tau = \boldsymbol{u}_L(t_l) - \boldsymbol{u}_L(t_k) \neq \boldsymbol{u}_L^+(t_l) - \boldsymbol{u}_L^-(t_k). \tag{A.19}$$

Therefore, the Lebesgue measure $dt$ must be replaced by a differential measure of $\boldsymbol{u}$, $d\boldsymbol{u}$, such that:

$$\int_{[t_k,t_l]} d\boldsymbol{u} = \boldsymbol{u}^+(t_l) - \boldsymbol{u}^-(t_k), \tag{A.20}$$

for every compact interval $[t_k, t_l]$ of $I$, which by narrowing the integral to a single instant set, $\{t_k\}$, becomes:

$$\int_{\{t_k\}} d\boldsymbol{u} = \boldsymbol{u}^+(t_k) - \boldsymbol{u}^-(t_k). \tag{A.21}$$

The differential measure, $d\boldsymbol{u}$ has a decomposition[7] corresponding to that of $\boldsymbol{u}$, i.e.:

$$d\boldsymbol{u} = d\boldsymbol{u}_L + d\boldsymbol{u}_A + d\boldsymbol{u}_C, \tag{A.22}$$

where the measure

- $d\boldsymbol{u}_L$ is absolutely continuous with respect to the (1-dimensional) Lebesgue measure $dt$ with $\dot{\boldsymbol{u}}$ as the density function, i.e. $d\boldsymbol{u}_L = \dot{\boldsymbol{u}} dt$ [26, 20]

$$d\boldsymbol{u}_L = \dot{\boldsymbol{u}} dt \tag{A.23}$$

[6] Recall that $\dot{\boldsymbol{u}}_C = 0$ and $\dot{\boldsymbol{u}}_A = 0$ almost everywhere [51].

[7] Lebesgue decomposition of a measure (also known as Radon-Nikodym decomposition) [57].

- $d\boldsymbol{u}_A$ is (purely) atomic and 0-dimensional:

$$d\boldsymbol{u}_A = (\boldsymbol{u}^+ - \boldsymbol{u}^-)d\eta \tag{A.24}$$

- $d\boldsymbol{u}_C$ is singular with respect to $dt$ and is assumed to be 0 and thus ignored.

Finally, one obtains:

$$d\boldsymbol{u} = \dot{\boldsymbol{u}}dt + (\boldsymbol{u}^+ - \boldsymbol{u}^-)d\eta. \tag{A.25}$$

Integration of measures $d\boldsymbol{u}_A$ and $d\boldsymbol{u}_L$ is given by

$$\int_{[t_k,t_l]} d\boldsymbol{u}_A = \sum_{\tau\in[t_k,t_l]\cap\{t_i\}} (\boldsymbol{u}^+(\tau) - \boldsymbol{u}^-(\tau)) = \boldsymbol{u}_A^+(t_l) - \boldsymbol{u}_A^-(t_k), \tag{A.26}$$

and

$$\int_{I_{kl}} d\boldsymbol{u}_L = \int_{t_k}^{t_l} \dot{\boldsymbol{u}}dt = \boldsymbol{u}_L(t_l) - \boldsymbol{u}_L(t_k) \tag{A.27}$$

respectively, where $I_{kl}$ is an interval between $t_k$ and $t_l$ that could closed, open or half-open.

### A.2.2 Nonsmooth EOMs

In order to be able to describe the system motion both in its continuous intervals and discontinuity points [93] has proposed to replace the EOM in the form of differential equations with the corresponding equality of measures, often referred to as a *measure differential equation* (MDE):

$$\boldsymbol{M}(\boldsymbol{q},t)d\boldsymbol{u} - \boldsymbol{h}(\boldsymbol{q},\boldsymbol{u},t)dt = d\boldsymbol{R}. \tag{A.28}$$

EOM in this form holds for every $t \in I$, as opposed to the standard acceleration-force relation which excludes points of discontinuity of $\boldsymbol{u}$ and $\dot{\boldsymbol{u}}$ and is thus true only for $[dt]$-almost every $t \in I$. Translating the abstraction into *mechanical terms*, the above EOM at the impact relates impulses and velocities/momenta at times of impact; in the absence of impacts the relation is between changes in momenta and in forces.

The force measure, $d\boldsymbol{R}$, is decomposed in the same manner as $d\boldsymbol{u}$, i.e. it may consist of Lebesgue-measurable forces $\boldsymbol{f}$, atomic impact impulses $\boldsymbol{F}$ and singular force measure $d\boldsymbol{R}_C$:

$$d\boldsymbol{R} = \boldsymbol{f}dt + \boldsymbol{F}d\eta + d\boldsymbol{R}_C, \tag{A.29}$$

where the last, singular term is again assumed to be zero.

After substituting Equations A.25 and A.29 into Eq. A.28 one obtains:

$$\boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{u}}dt + \boldsymbol{M}(\boldsymbol{q},t)(\boldsymbol{u}^+ - \boldsymbol{u}^-)d\eta - \boldsymbol{h}(\boldsymbol{q},\boldsymbol{u},t)dt = \boldsymbol{f}dt + \boldsymbol{F}d\eta. \tag{A.30}$$

Now, there are two approaches Eq. A.30 can be dealt with. One can evaluate the differential measure $d\boldsymbol{u}$ for a given integration interval leading to the relation between both impulsive and non-impulsive forces and accelerations in a unified manner at the velocity level. Alternatively, one can decompose this equation into parts corresponding to impact instants and smooth motion in order to treat them separately.

**Unified handling**

Due to the following relations

$$\int_{\Delta t} d\boldsymbol{u} = \Delta\boldsymbol{u}, \qquad \int_{\Delta t} \boldsymbol{h}dt \approx \boldsymbol{h}\Delta t, \qquad \int_{\Delta t} d\boldsymbol{R} = \boldsymbol{R}, \tag{A.31}$$

integrating Eq. A.30 over a finite time interval $\Delta t$ gives:

$$\boldsymbol{M}(\boldsymbol{q},t)\Delta\boldsymbol{u} - \boldsymbol{h}(\boldsymbol{q},\boldsymbol{u},t)\Delta t = \boldsymbol{R}, \tag{A.32}$$

which yields a discretized version of Eq. A.30.

**Separate handling**

One may split the Eq. A.30 into atomic and Lebesgue-measurable parts, i.e.:

$$\boldsymbol{M}(\boldsymbol{q},t)(\boldsymbol{u}^+ - \boldsymbol{u}^-)d\eta = \boldsymbol{F}d\eta \tag{A.33a}$$

$$\boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{u}}dt - \boldsymbol{u}^-)d\eta - \boldsymbol{h}(\boldsymbol{q},\boldsymbol{u},t)dt = \boldsymbol{f}dt, \tag{A.33b}$$

respectively. Evaluating Eq. A.33a for any discontinuity point, $t_i$, of $\boldsymbol{u}$ becomes

$$\boldsymbol{M}(\boldsymbol{q}(t_i),t_i)(\boldsymbol{u}^+(t_i) - \boldsymbol{u}^-(t_i)) = \boldsymbol{F}(t_i), \tag{A.34}$$

which is the impact equation for the system.

On the other hand, Eq. A.33b naturally leads us back to:

$$\boldsymbol{M}(\boldsymbol{q},t)\dot{\boldsymbol{u}} - \boldsymbol{h}(\boldsymbol{q},\boldsymbol{u},t) = \boldsymbol{f}, \tag{A.35}$$

which describes the system in times of smooth motion (i.e. it is defined on $I \setminus \{t_j\}$). Any motion $\boldsymbol{q}(t)$ which fulfills the impact equation A.34 and A.35 is also a solution of the MDE in Eq. A.28.

Simulation based on this interpretation is performed by integrating the accelerations obtained from the smooth motion EOMs for as long as there are no discontinuities in velocities or accelerations; when a discontinuity *event* is discovered the smooth simulation is interrupted, the exact time of the event is usually estimated and the system state is brought back to this instant (it is sometimes called *backing up* [56, 87])[8] and impact equations are applied, which changes the system state without advancing the simulation time; the smooth simulation is restarted from that instant with the modified state. This paradigm is often referred to as event-driven or event-based simulation [93, 21].

[8] Backing up is sometimes simply ignored [55].

# Symbols and notational conventions

$\boldsymbol{M}$: matrix $\boldsymbol{M}$

$\boldsymbol{\nu}$: vector $\boldsymbol{\nu}$

$\boldsymbol{M}^T$: the transpose of matrix $\boldsymbol{M}$

$\boldsymbol{M}^{-1}$: the inverse of matrix $\boldsymbol{M}$

$\mathbf{0}_{i \times j}$: an $i$ by $j$ matrix of zeros

$\mathbf{1}_{i \times j}$: an $i$ by $j$ identity matrix

$\boldsymbol{\nu}[i]$: the $ith$ entry of vector $\boldsymbol{\nu}$

$\boldsymbol{M}[i, j]$: the entry in the $i^{th}$ row and $j^{th}$ column of matrix $\boldsymbol{M}$

$\boldsymbol{M}[i, *]$: the $i^{th}$ row of matrix $\boldsymbol{M}$

$\boldsymbol{M}[*, j]$: the $j^{th}$ column of matrix $\boldsymbol{M}$

$\boldsymbol{M}[i_1 \to i_2, j_1 \to j_2]$: a submatrix of $\boldsymbol{M}$ consisting of the intersection of rows $i_1$ through $i_2$, and columns $j_1$ through $j_2$

$\boldsymbol{\nu}[i_1 \to i_2]$: a subvector of $\boldsymbol{\nu}$ consisting of entries $i_1$ through $i_2$

$\lambda(i)$: the index of the topological parent of body $i$

$\boldsymbol{J}_i$: $i^{th}$ body Jacobian

$\boldsymbol{J}_c$: constraint-space constraint Jacobian

$\boldsymbol{J}_{c,i}$: $i^{th}$ constraint Jacobian

$\boldsymbol{v}$: $3 \times 1$ vector denoting (linear) velocity **or** an arbitrarily-sized vector denoting a constraint-space velocity

$\boldsymbol{\omega}$: $3 \times 1$ vector denoting angular velocity

$\boldsymbol{a}$: $3 \times 1$ vector denoting (linear) acceleration **or** an arbitrarily-sized vector denoting a constraint-space acceleration

$\boldsymbol{\alpha}$: $3 \times 1$ vector denoting angular acceleration

$\boldsymbol{f}$: $3 \times 1$ vector denoting a force **or** an arbitrarily-sized vector denoting a constraint-space reaction force

$\boldsymbol{\tau}$: $3 \times 1$ vector denoting a torque/moment **or** an arbitrarily-sized vector denoting a joint-space force

$\boldsymbol{\iota}$: an arbitrarily-sized vector denoting a joint-space impulse

$\boldsymbol{I}$: a $3 \times 3$ matrix denoting an inertia tensor

$\widehat{\boldsymbol{v}}$: $6 \times 1$ vector denoting spatial velocity

$\widehat{\boldsymbol{a}}$: $6 \times 1$ vector denoting spatial acceleration

$\widehat{\boldsymbol{f}}$: $6 \times 1$ vector denoting spatial force

$\widehat{\boldsymbol{I}}$: $6 \times 6$ matrix denoting spatial inertia tensor

$\widehat{\boldsymbol{I}}^A$: $6 \times 6$ matrix denoting spatial articulated-body inertia

$S$, $T$: motion freedom ($S$) and constraining reaction force ($T$) vector subspaces

$\widehat{\boldsymbol{S}}$, $\widehat{\boldsymbol{T}}$: matrices representing $S$ and $T$

$\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$: vectors of generalized or joint-space position, velocity and acceleration variables

$T$ in subscript: total velocity/acceleration, i.e. relative to the immobile base of the system

$L$ in subscript: local velocity/acceleration, i.e. relative to the velocity/acceleration of the topological parent

$CM$ in subscript: concerns the body center of mass

$PV$ in subscript: concerns the body's inboard joint pivot point

$v$ : ... in pseudocode: declaration of a variable $v$

$v$ := ... in pseudocode: definition of a variable $v$

$v$ = ... in pseudocode: assignment of a variable $v$

# Acronyms and abbreviations

| Abbreviation | Description | Introduction |
|---|---|---|
| CD | collision detection | page 7 |
| DCD | discrete collision detection | page 8 |
| CCD | continuous collision detection | page 8 |
| DoF | degree of freedom | page 13 |
| EOM | equation of motion | page 14 |
| DAE | differential-algebraic equation | page 29 |
| LCP | linear complementarity problem | page 36 |
| QP | quadratic programming | page 46 |
| SOLE | system of linear equations | page 51 |
| MLCP | mixed linear complementarity problem | page 55 |
| CRBA | composite rigid body algorithm | page 67 |
| ABA | articulated body algorithm | page 67 |
| RNEA | recursive Newton-Euler algorithm | page 67 |
| CRB | composite rigid body | page 75 |
| AB | articulated body | page 78 |
| RK4 | fourth-order Runge-Kutta | page 98 |
| SI | sequential impulse | page 101 |
| AIA | articulated islands algorithm | page 138 |
| MDE | measure differential equation | page 173 |
| w.r.t. | with respect to | - |
| LHS | left-hand side | - |
| RHS | right-hand side | - |
| CPU | central processing unit | - |
| GPU | graphics processing unit | - |
| GPGPU | general-purpose computing on graphics processing units | - |

# List of Figures

# List of Tables

# List of Procedures