Aspects of Computational Contact Dynamics

Tomasz Koziara

A thesis submitted for the degree of Doctor of Philosophy

Department of Civil Engineering
University of Glasgow

June 2008

Abstract

This work summarises a computational framework for dealing with dynamic multi-body frictional contact problems. It is in fact a detailed account of an instance of the Contact Dynamics method by Moreau and Jean. Hence the title. Multi-body systems with contact constraints are common. Some of them, such as machines or arrangements of particulate media, need to be predictable. Predictions correspond to approximate solutions of mathematical models describing interactions within such systems. The models are implemented as computational algorithms.

The main contributions of the author are in an improved time integration method for rigid rotations, and in a robust Newton scheme for solving the frictional contact problem. A simple and efficient way of integrating rigid rotations is presented. The algorithm is stable, second order accurate, and in its explicit version involves evaluation of only two exponential maps per time step. The semi-explicit version of the proposed scheme improves upon the long term stability, while it retains the explicitness in the force evaluation. The algebraic structure of both schemes makes them suitable for the analysis of constrained multi-body systems. The explicit algorithm is specifically aimed at the analysis involving small incremental rotations, where its modest computational cost becomes the major advantage. The semi-explicit scheme naturally broadens the scope of possible applications. The semismooth Newton approach is adopted in the context of the frictional contact between three-dimensional pseudo-rigid bodies, proposed by Cohen and Muncaster. The Signorini-Coulomb problem is formulated according to the formalism of Contact Dynamics. Hybrid linearisation, parameter scaling and line search techniques are combined as the global convergence enhancements of the Newton algorithm. Quasi-static simulations of dry masonry assemblies exemplify performance of the presented framework.

Acknowledgements

I would like to thank Prof. Nenad Bićanić for his constant support, encouragement and patience over the last years. I would be nowhere close to finishing this thesis without his help. I would also like to take this opportunity to express my deepest gratitude to all of my friends, without whom doing this work, or doing anything for that matter, would be vain and lonely. This work is dedicated to my family.

Contents

Chapter 1. Introduction 7
Style 7
Topic 7
Basics 8
Map 9
Contributions 11

Chapter 2. Shape 12

Chapter 3. Kinematics 14
3.1. Rigid body 14
3.2. Pseudo-rigid body 21
3.2.1. Kinematics of vectors and tensors 23
3.2.2. Pseudo-rigid motion and convexity 27
3.3. Matrix notation 27
3.4. Literature 28

Chapter 4. Dynamics 29
4.1. Pseudo-Rigid body 29
4.1.1. Referential formulation 33
4.1.2. Constitutive equation 35
4.2. Rigid body 35
4.3. Matrix notation 37
4.4. Literature 38

Chapter 5. Time stepping 39
5.1. Pseudo-rigid dynamics 39
5.1.1. Accuracy 40
5.1.2. Stability 41
5.1.3. Conservation 42
5.2. Rigid dynamics 44
5.2.1. Preliminaries 44
5.2.2. Scheme 45
5.2.3. Conservation and stability 47
5.2.4. Free rotation 52
5.2.5. Efficiency 55
5.3. Quasi-statics 56
5.4. Literature 58

Chapter 6. Local frames 60
6.1. From generalised to local velocities 62
6.2. Rigid kinematics 62
6.3. Pseudo-rigid kinematics 63
6.4. Dynamics and quasi-statics 63

Chapter 7. Local dynamics 64
7.1. Many bodies and local frames 66
7.2. Constraints 68
7.3. Quasi-statics 68

Chapter 8. Joints 70
8.1. Back to the discrete case 71
8.2. Single-body joints 72
8.3. Multi-body joints 73
8.4. Configuration space 73

Chapter 9. Contact points 74
9.1. Auxiliary data structures 74
9.1.1. Tuple 75
9.1.2. Pointer 75
9.1.3. List 75
9.1.4. Hash table 77
9.1.5. Priority queue 79
9.1.6. Binary search tree 79
9.1.7. Priority search tree 80
9.1.8. Segment and interval trees 87
9.2. The optimal data structure 88
9.3. Finding contact candidates 90
9.3.1. Axis aligned bounding boxes 91
9.3.2. 1D interval overlap 94
9.3.3. 2D rectangle overlap 98
9.3.4. The reference approach 106
9.3.5. Spatial hashing 108
9.3.6. Plane-sweep approach 111
9.4. Finding points and normals 115
9.4.1. Finding a common point 119
9.4.2. Computing the convex hull 121
9.4.3. No gaps? 122
9.5. Literature 123
9.5.1. Collision detection 123
9.5.2. Polyhedra 124

Chapter 10. The frictional contact problem 126
10.1. The contact problem 126
10.1.1. From gaps to velocity constraints 127
10.1.2. Moreau's sweeping 128
10.1.3. Velocity jumps 130
10.1.4. Back to the discrete case 131
10.1.5. From inequalities to equalities 132
10.1.6. Non-smoothness 133
10.1.7. Existence of solutions 134
10.1.8. Contact problem as root finding 135
10.1.9. Contact problem as minimisation 136
10.1.10. Quasi-statics 138
10.2. The friction problem 138
10.2.1. Retrieving the projection formula 139
10.2.2. Friction problem as root finding 140
10.2.3. Friction problem as minimisation 140

10.3. The frictional contact problem 141
10.3.1. Projection formulae 141
10.3.2. Potentials, normality, monotonicity and association 141
10.3.3. Lack of potential, normality, monotonicity and association 143
10.3.4. The Bipotential Method 144
10.3.5. Frictional contact as root finding 145
10.3.6. The well-behaved juxtaposed simplification 147
10.3.7. Can the frictional contact operator be monotone? 147
10.4. Cohesion 148
10.5. Energetic consistency 149
10.5.1. Contacts, ideally plastic impacts, and friction 150
10.5.2. Frictionless impacts and contacts 150
10.5.3. Impacts, contacts and friction 151
10.6. Literature 152

Chapter 11. Solvers 154
11.1. Properties of **C** 154
11.1.1. Invertibility 154
11.1.2. Semi-smoothness 154
11.2. Newton method 156
11.2.1. Unilateral contact 157
11.2.2. Frictional tangents 158
11.2.3. Complete algorithm 159
11.2.4. Inclusion of joints 165
11.3. Gauss-Seidel method 166
11.4. Literature 166

Chapter 12. Implementation 168

Chapter 13. Examples 170
13.1. Rigid rotations 170
13.1.1. Unstable rotation 170
13.1.2. Heavy top 172
13.1.3. Rotating plate 173
13.2. Contact search 175
13.3. Newton solvers 178
13.4. Some benchmarks 182
13.4.1. Pendulum 182
13.4.2. Sphere impacting a plate 184
13.4.3. Cube impacting a plate 186
13.4.4. Double pendulum impacting a rigid wall 187
13.4.5. Block sliding on a frictional table 189
13.4.6. Newton's cradle 190
13.4.7. Masonry arch 192
13.4.8. Box-kite push until lockup 195
13.4.9. Lourenco's wall 200

Chapter 14. Conclusions 203

Bibliography 205

CHAPTER 1

Introduction

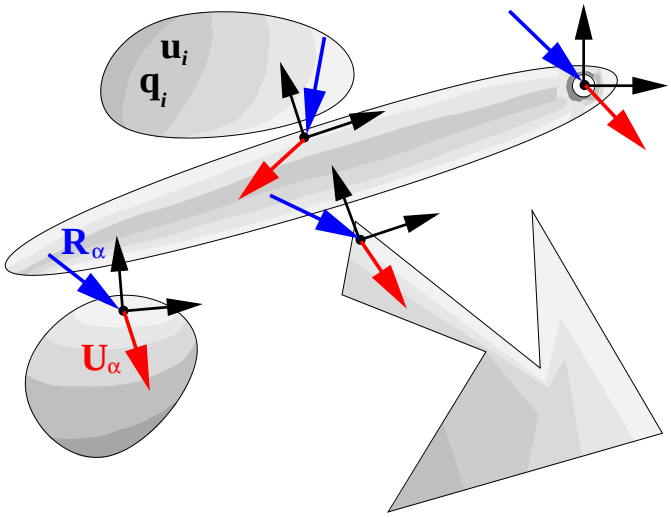
I once watched an interview with the Dutch computer scientist Edsger Wybe Dijkstra. In the flow of the interview, he told a story about a lecture he gave at a software company somewhere in Brussels. The lecture was about writing correct code, and it turned out to be a complete failure. According to Dijkstra's judgement, the programmers were not interested in learning how to code, because they "derived their intellectual excitement from the fact that they didn't quite know what they were doing". At that time I was working at a software company and I could indeed observe this type of excitement in my own manner of work. Sometime later I started doctoral studies in Glasgow. In fact I was not that much interested in the topic itself, but rather I wanted to find a way of turning the "intellectual excitement" into something more useful. This thesis gives a snapshot of an ongoing effort towards realisation of this aim. It would be far fetched to claim, that the state of mind mentioned by Dijkstra did not accompany me occasionally in the course of this work. Nevertheless, upon reflection I have to admit that maybe it is rather a kind of balance that should be sought.

Style. This is the only chapter written in the first person. The remaining ones use a mixture of the passive voice and the "royal we", which I have found most convenient and versatile. I did not manage to avoid expressions like: "It is easy to see ...", or "It is not difficult ...", or "Clearly ... " etc. This has to be brought down to my linguistic limitations, rather than mathematical skills. I did seriously consider removing them all after finishing writing, but then I gave up, foreseeing too much trouble. I trust the reader will accept my apologies here. I did make an effort to deliver some mathematical rigour, more for my own use and as an exercise, rather than because it was unavoidable. For this reason, I suppose, a mathematician would find this text not only overblown but also lacking precision, while an engineer could find it at times formidable. I do tend to include lengthy derivations whenever necessary, as I would like them to serve me (or someone else) as a reference at a later point. At the end of some chapters I have included concise literature reviews. This might seem like a strange choice at first. I reckon it is not so, as it seems more natural to become curious of related developments, after having some taste of the main body of a chapter. Also for me it was often easier to summarise additional references, after the foregoing material had taken its final shape.

Topic. This work outlines a computational method aimed at tracing motion of bodies coming into contact with each other. As such, the motion of contacting bodies is among the most common physical phenomena. By merely looking around, one can easily register a number of "multi-body systems with contact constraints". Almost every human activity involves some kind of "contact dynamics". For example, typing this very text. Of course, most of every day actions do not require to be abstracted in the language of mathematics in order to be executed. But in general, there is a need for such abstraction. It is both practical (driven by industry) and purely cognitive. Several years ago, when reviewing literature related to the issues

of contact, my attention was drawn to the works of Moreau [**156**] and Jean [**102**], describing basics of their *Contact Dynamics* method (CD). I did not understand much of those papers at first. Over time, I have filled most (but not all) of the gaps in my understanding. In the following I have described a particular instance of a CD algorithm. I preferred not to repeat dully those aspects of the mathematical formulation, which are still beyond my grasp (e.g. *measure theory*). Hence, I tend to resort to discretisation. I do not deliberate much on the convergence of the discrete scheme. Relevant references are mentioned in the due time. My intention is to deliver a self-contained summary for a programmer interested in getting into grips with CD.

Basics. It will be useful to introduce some basic notions here. It can be best done by drawing a figure. Let us have a look



There are four bodies in the figure. Placement of each point of every body is determined by the configuration $\mathbf{q}_i$. Velocity of each point of every body is determined by the velocity $\mathbf{u}_i$. If the time history of velocity is known, the configuration can be computed as

$$\mathbf{q}(t) = \mathbf{q}(0) + \int_0^t \mathcal{F}(\mathbf{u}(t))\, dt \tag{1.0.1}$$

where $\mathcal{F}$ is a general function, usually an identity, $\mathcal{F}(x) = x$. The velocity is determined by integrating Newton's law

$$\mathbf{p}(t) = \mathbf{p}(0) + \int_0^t \mathbf{f}(\mathbf{q}, \mathbf{u}, t)\, dt \tag{1.0.2}$$

$$\mathbf{u}(t) = \mathcal{G}(\mathbf{p}(t)) \tag{1.0.3}$$

where $\mathbf{p}$ is the momentum, $\mathcal{G}$ is another general function, and $\mathbf{f}$ is the resultant force. While integrating the motion of bodies, one keeps track of a number of local coordinate systems. These will be called *local frames*. There are four of them in the figure. Each local frame is related to a pair of points, belonging to two distinct bodies. An observer embedded in a local frame calculates the local relative velocity $\mathbf{U}$ of one of the points, viewed from the perspective of the other point. If necessary,

the observer applies some force $\mathbf{R}$[1]. An action of each observer can be implicitly described as

$$\mathbf{C}(\mathbf{U},\mathbf{R}) = \mathbf{0} \tag{1.0.4}$$

Actions of observers are local. They only know how to react to a change of velocity $\mathbf{U}$ at the point of their residence. At the same time, as many of them act collectively, the effect of their work influences one another. A global observer can see this happening and is able to transform the relation

$$\mathbf{u}(t) = \mathcal{G}\left(\mathbf{p}(0) + \int_0^t \mathbf{f}(t)\,dt\right) \tag{1.0.5}$$

into a formula describing what will be called *local dynamics*

$$\mathbf{U} = \mathbf{W}\mathbf{R} + \mathbf{B} \tag{1.0.6}$$

$\mathbf{W}$ and $\mathbf{B}$ determine a linear transformation between the local forces $\mathbf{R}$ and velocities $\mathbf{U}$ for every instant of time. The global observer can then force his local peers to act in harmony by stating

$$\mathbf{C}(\mathbf{W}\mathbf{R} + \mathbf{B}, \mathbf{R}) = \mathbf{0} \tag{1.0.7}$$

In a sense, there is no more to it. In the above, when using symbols $\mathbf{q}$, $\mathbf{u}$, $\mathbf{U}$ and $\mathbf{R}$ without indices, collections of relevant variables were meant.

Map. The remaining chapters decompose the above figure into more or less independent modules. An experienced reader should be able to skip uninteresting bits, and move right to the one of his or her interest. In order to make this easier, I have summarised below all, but the last[2], of the forthcoming chapters.

Chapter 2: Shape

The class of shapes considered in the implementation is described. In short, these are arbitrary unions of convex polyhedrons. These include finite-element like meshes, etc. A class of *surface elements* is distinguished. These are adjacent to the surface of discretised bodies and will be used later for contact detection.

Chapter 3: Kinematics

This chapter deals with formula (1.0.1). It contains quite a detailed account of what $\mathbf{q}$ and $\mathbf{u}$ are in the case of *rigid* and so called *pseudo-rigid* bodies. Notions of the configuration and tangent spaces are introduced, to which respectively $\mathbf{q}$ and $\mathbf{u}$ belong. Issues of parametrisation of $\mathbf{q}$ by a reduced number of variables are discussed for rigid bodies. Some basics of relevant tensor calculus are given.

Chapter 4: Dynamics

Formulae (1.0.2) and (1.0.3) are a focus of attention here. The classical Newtonian balance principles are worked out for rigid and pseudo-rigid bodies. The matrix notation given at the end of the chapter will be of use for a reader interested in the implementation.

[1] Let us is temporarily abandon the traditional notion of a *passive* observer.

[2] Is there a point in summarising conclusions?

Chapter 5: Time stepping

Time integration, that is a numerical equivalent of what happens in (1.0.1) and (1.0.2), is the "work horse" of the complete scheme. My intention here was to use the simplest possible methods. For dynamics, explicit second order schemes are employed (equivalent to the central difference method). Their accuracy reduces to the first order in the presence of impacts. First order implicit Euler scheme is utilised for the quasi-static case.

Chapter 6: Local frames

In this chapter the notion of the local frame is given a precise definition. The main point here is in introduction of a linear operator $\mathbf{H}$, a role of which is to transform $\mathbf{u}$ into $\mathbf{U}$. That is $\mathbf{U} = \mathbf{H}\mathbf{u}$. Specific forms of $\mathbf{H}$ for rigid and pseudo-rigid bodies are given.

Chapter 7: Local dynamics

The notion of the local frame and the $\mathbf{H}$ mapping are employed in order to derive the equations of local dynamics (1.0.6). Some links with convexity and conjugacy are explained. As a byproduct, the numerical integration in time is given a wrapping of unconstrained convex minimisation. Also, some remarks about the structure of the $\mathbf{W}$ operator are included.

Chapter 8: Joints

A joint is pictured in the top-right part of the figure given few pages earlier. The slender body can only rotate around this point. Implementation of this and other kinds of joints is described in this chapter. Specific actions of the local observer in form (1.0.4) are given. In other words, this chapter is about the *equality constraints*.

Chapter 9: Contact points

This chapter summarises algorithms, aimed at finding candidate contact points. Contrary to joints, these are usually not known in advance. A geometrical search needs to be done to identify pairs of points, where local frames are later placed. Efficient methods for performing this task are given. One of the characteristic features is the derivation of local frames from the volumetric intersections between pairs of surface elements. This is of use in the presence of nonsmooth geometry.

Chapter 10: The frictional contact problem

Once the local frames related to contacts have been found, the frictional contact problem can be defined. This is done in a standard manner, that is in stages. The frictionless non-penetration problem is discussed at greatest length. Then the friction problem, not coupled with non-penetration is summarised. The frictional contact problem is given and difficulties related to its solution pointed out. In the meantime, the equality form (1.0.4) of the contact constraints is worked out. When applicable, analogies with constrained minimisation are mentioned, although in the end only the root finding problem (1.0.7) prevails.

Chapter 11: Solvers

Numerical methods for solving problem (1.0.7) are discussed. The classical fixed-point iteration is described, together with a semi-smooth Newton method and a hybrid method based on heuristic improvements. The block Gauss-Seidel scheme, traditionally used in CD, is also summarised.

Chapter 12: Implementation

In this brief chapter, the foregoing developments are summarised in two algorithms. One for dynamics and one for quasi-statics.

Chapter 13: Examples

A number of examples is given here. This include integration of rigid rotations, contact detection and Newton solvers. Several benchmarks are included, comparing the results with previously documented figures.

Contributions. The biggest gain from this work is of course personal. It was undoubtedly a privilege to have several years for discovering and improving a method of work that suits me best. One the other hand, I should mention some papers as these seem to be the agreed upon measure of performance.

My first journal paper [**123**] described a Newton method for solving (1.0.7). The main scheme was developed earlier by Hüeber *et al.* [**96**]. My contribution was only in translating that work into the context of CD and developing some heuristic improvements (cf. Section 11.2). The Signorini-Coulomb problem is formulated according to the formalism of Contact Dynamics. Hybrid linearisation, parameter scaling and line search techniques are combined as the global convergence enhancements of the Newton algorithm. Quasi-static simulations of dry masonry assemblies exemplify performance of the presented framework.

The second paper [**124**] described a new time integration scheme for rigid rotations (cf. Section 5.2). It arose as a byproduct of an interaction with our industrial partner. Papers by Krysl *et al.* [**126, 163, 128, 127**] were of great help and served as inspiration. The scheme given in Section 5.2 is simple and efficient. It is also stable, second order accurate, and in its explicit version involves evaluation of only two exponential maps per time step. The semi-explicit version of the proposed scheme improves upon the long term stability, while it retains the explicitness in the force evaluation. The algebraic structure of both schemes makes them suitable for the analysis of constrained multi-body systems. The explicit algorithm is specifically aimed at the analysis involving small incremental rotations, where its modest computational cost becomes the major advantage. The semi-explicit scheme naturally broadens the scope of possible applications.

During the first year of studies I was still biased by my programming background. It was easier to work on contact detection, rather than study CD. The work presented in Chapter 9 is quite laborious, although it does not seem to be adding much to the saturated field of geometrical algorithms. Some of the results presented there I have improved only recently, while writing up. More time is needed to test them thoroughly.

CHAPTER 2

Shape

Shapes are approximated by volumetric meshes identical with those used in the finite element analysis[1]. This serves a double purpose. Within the adopted, simplified representation of motion, mesh density corresponds to the accuracy of contact resolution. At the same time, an extension to the finite element case is made easier. Nevertheless, the extension is not pursued within this work. The shape of a body is then represented by a convex decomposition (discretisation) into hexahedrons, wedges, pyramids and tetrahedrons (Figure 2.0.1). Those are composed of nodes, edges and faces in a manner suitable for identification of topological adjacency relations. The volumetric convex cells are called elements. All of those issues are rather elementary and need no further explanation. The only notion specific to the current context corresponds to the set of *surface elements*. The faces of those elements have nonempty intersections with the discretised surface of a body. Figure 2.0.2 illustrates the idea.

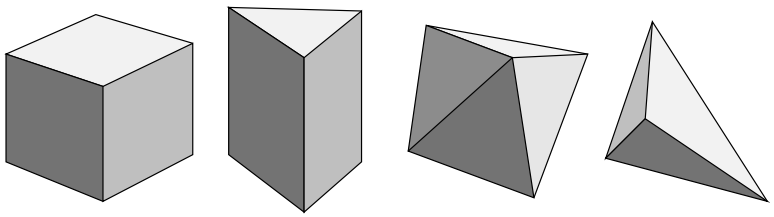


FIGURE 2.0.1. Hexahedron, wedge, pyramid and tetrahedron. Basic elements used for the discretisation of a body shape.

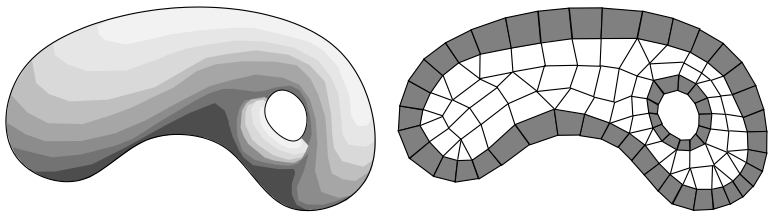


FIGURE 2.0.2. Torus shaped body and a planar slice of its discretisation. The surface elements have been darkened.

The surface elements will play a role in the contact detection process described in Chapter 9. As far as the present framework is concerned, the remaining elements are only used to calculate characteristics of mass distribution. It is relevant to point

[1]This is assumed only to simplify the presentation. In the actual implementation, apart from the mesh representation, arbitrary unions of convex shapes are admitted.

out that the convexity of elements is a necessary condition for the correctness of some of the subsequently employed algorithms. Within the class of motions considered here, convexity is naturally preserved (Chapter 3). An eventual generalisation admitting a greater degree of deformability ought to account for the possibility of severe element distortion. This can be for example achieved, by employing an exclusively tetrahedral mesh within the set of the surface elements.

CHAPTER 3

Kinematics

Placement of a three-dimensional body can be identified with a subset of the Euclidean point space E^3. The open nonempty set occupied by the body at time t_0 is denoted by $\mathcal{B}_0$. The closure of $\mathcal{B}_0$ bears the name of the *reference configuration*. Accordingly, at any time t the closure of an open nonempty set $\mathcal{B}$ is referred to as the *current configuration*. Boundaries of those sets are denoted by $\partial\mathcal{B}_0$ and $\partial\mathcal{B}$. An invertible mapping χ carrying points of $\mathcal{B}_0$ into corresponding points of $\mathcal{B}$ is called a *motion*. Thus $\mathbf{x} = \chi(\mathbf{X}, t)$, where $\mathbf{x} \in \mathcal{B}$ and $\mathbf{X} \in \mathcal{B}_0$.

In order to express the motion in an explicit form, it is necessary to select coordinate systems $\{x^i\}$ and $\{X^i\}$, covering respectively the current and the reference configuration. This is most naturally done by an introduction of two Cartesian coordinate systems, where both points and vectors are represented by triplets of real numbers. Let $\mathbf{e}_i$ and $\mathbf{E}_i$ be two sets of orthonormal vectors (with respect to the standard inner product on R^3). Keeping in mind the notional difference between a point (location) and a vector (equivalence class of location differences), the spatial and referential points can be expressed in coordinates as $\mathbf{x} = x^i\mathbf{e}_i$ and $\mathbf{X} = X^i\mathbf{E}_i$. The real numbers x^i, X^i are the components of $\mathbf{x}, \mathbf{X}$ with respect to the bases $\mathbf{e}_i, \mathbf{E}_i$. The zero origins of the two coordinate systems need not coincide in the physical space.

It should be noted, that $\mathcal{B}_0$ and $\mathcal{B}$, being open subsets of the Euclidean space, are by definition *manifolds*. In general a *differentiable manifold* can be defined as a set in which neighbourhoods of all points can be mapped in a smooth and invertible manner onto open subsets of R^n. A *tangent space* $T_{\mathbf{x}}\mathcal{B}$ is a vector space spanned at a point $\mathbf{x} \in \mathcal{B}$ of the manifold and composed of all possible velocities of the point. The set of all tangent spaces at all points is called the *tangent bundle* $T\mathcal{B}$. As all tangent spaces of $\mathcal{B}_0$ and $\mathcal{B}$ are identical, vector bases $\mathbf{e}_i, \mathbf{E}_i$ can be used to parametrise the tangent bundles $T\mathcal{B}_0$ and $T\mathcal{B}$. More precise definitions can be found in Arnold [**12**, pp. 76-81] or Marsden and Hughes [**147**, pp. 35-36].

3.1. Rigid body

The motion of a rigid body reads

$$\mathbf{x}(\mathbf{X}, t) = \mathbf{\Lambda}(t)\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \bar{\mathbf{x}}(t) \tag{3.1.1}$$

where $\mathbf{\Lambda}(t)$ is a 3×3 rotation operator, $\bar{\mathbf{X}}$ is a selected referential point, and $\bar{\mathbf{x}}(t)$ is a spatial point. It is seen that $\bar{\mathbf{x}}(t) = \mathbf{x}\left(\bar{\mathbf{X}}, t\right)$ is the motion of the selected point $\bar{\mathbf{X}}$. The term $\mathbf{\Lambda}(t)\left(\mathbf{X} - \bar{\mathbf{X}}\right)$ represents the rotation of $\mathbf{X}$ about the point $\bar{\mathbf{X}}$. Thus, the rigidity condition follows $\|\mathbf{x} - \bar{\mathbf{x}}\| = \left\|\mathbf{X} - \bar{\mathbf{X}}\right\|$, where the standard Euclidean norm is assumed. The linear operator $\mathbf{\Lambda}$ acts between the tangent bundles $\mathbf{\Lambda} : T\mathcal{B}_0 \to T\mathcal{B}$. In order to represent rotations, $\mathbf{\Lambda}$ must be orthogonal $\mathbf{\Lambda}^T\mathbf{\Lambda} = \mathbf{I}$, where $\mathbf{I}$ is the 3×3 identity on $T\mathcal{B}_0$. It is physically meaningful to assume that $\mathbf{\Lambda}$ preserves orientation, so that $\det(\mathbf{\Lambda}) = 1$. The set of all 3×3 matrices with the assumed properties forms a group under matrix multiplication, called the *special orthogonal group* $SO(3)$ [**12**, p. 126]. The configuration space of a rigid body can be then defined as

$$\mathcal{Q}^{rig} = R^3 \times SO(3) \tag{3.1.2}$$

The set $\mathcal{Q}^{rig}$ has the structure of a six-dimensional manifold. The first three coordinates are simply those of the point $\bar{\mathbf{x}}$. The remaining three coordinates correspond to the parametrisation of the rotation space. As $\mathbf{\Lambda}^T\mathbf{\Lambda}$ is a symmetric matrix, the condition $\mathbf{\Lambda}^T\mathbf{\Lambda} = \mathbf{I}$ induces six independent constraints on nine entries of the rotation matrix. It can be shown that the Jacobian of the constraints has full rank everywhere, and thus the implicit function theorem implies existence of locally smooth and invertible maps from $SO(3)$ into R^3. Hence, the special orthogonal group is a manifold and so is the configuration space $\mathcal{Q}^{rig}$.

In fact it will be useful to extend a bit the discussion related to the constraint function $f(\mathbf{\Lambda}) = \mathbf{\Lambda}^T\mathbf{\Lambda} - \mathbf{I}$. The surface $f(\mathbf{\Lambda}) = \mathbf{0}$ is embedded in the nine-dimensional space of all 3×3 matrices. On the part where $\det(\mathbf{\Lambda}) = 1$, it is composed of the points of the manifold $SO(3)$. A selected point $\mathbf{\Lambda} \in SO(3)$ travels on $SO(3)$ along the directions tangent to the surface: $\dot{\mathbf{\Lambda}} \in T_{\mathbf{\Lambda}}SO(3)$. Thus $\dot{\mathbf{\Lambda}}$ must be orthogonal to the gradients of all six scalar constraints in f. In other words $Df(\mathbf{\Lambda}) \cdot \dot{\mathbf{\Lambda}} = \mathbf{0}$ or equivalently

$$\dot{\mathbf{\Lambda}}^T\mathbf{\Lambda} + \mathbf{\Lambda}^T\dot{\mathbf{\Lambda}} = \mathbf{0} \tag{3.1.3}$$

Let us define an anti-symmetric 3×3 operator as

$$\hat{\mathbf{\Omega}} = \mathbf{\Lambda}^T\dot{\mathbf{\Lambda}} \tag{3.1.4}$$

so that (3.1.3) states $\hat{\mathbf{\Omega}}^T = -\hat{\mathbf{\Omega}}$. If $\mathbf{\Lambda} = \mathbf{I}$, there follows that $\hat{\mathbf{\Omega}} = \dot{\mathbf{\Lambda}}$, hence the tangent space $T_{\mathbf{I}}SO(3)$ is composed of anti-symmetric 3×3 matrices. In the remaining case $\mathbf{\Lambda} \neq \mathbf{I}$, the tangent space $T_{\mathbf{\Lambda}}SO(3)$ is composed of matrix products $\mathbf{\Lambda}\hat{\mathbf{\Omega}}$. It should be noted, that the *three* independent components of $\hat{\mathbf{\Omega}}$ are exactly the reason why Df was assumed to have full rank in the previous paragraph (the dimension of the null space of Df, cf. [**1**]).

The operator $\hat{\mathbf{\Omega}}$ deserves further attention. From (3.1.1) and (3.1.4) the velocity of a spatial point can be computed as follows

$$\dot{\mathbf{x}} = \mathbf{\Lambda}\hat{\mathbf{\Omega}}\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \dot{\bar{\mathbf{x}}} \tag{3.1.5}$$

thus $\hat{\mathbf{\Omega}}$ acts between the spaces $\hat{\mathbf{\Omega}} : T\mathcal{B}_0 \to T\mathcal{B}_0$. Let $\mathbf{y} = \mathbf{x} - \bar{\mathbf{x}}$ and $\mathbf{Y} = \mathbf{X} - \bar{\mathbf{X}}$. Noting that $\mathbf{Y} = \mathbf{\Lambda}^T\mathbf{y}$, equation (3.1.5) can be rewritten as

$$\dot{\mathbf{y}} = \mathbf{\Lambda}\hat{\mathbf{\Omega}}\mathbf{\Lambda}^T\mathbf{y} \tag{3.1.6}$$

Obviously, transformation $\mathbf{\Lambda}\hat{\mathbf{\Omega}}\mathbf{\Lambda}^T$ preserves anti-symmetry of $\hat{\mathbf{\Omega}}$. It is convenient to define the following operator

$$\hat{\omega} = \mathbf{\Lambda}\hat{\mathbf{\Omega}}\mathbf{\Lambda}^T \tag{3.1.7}$$

acting between the spaces $\hat{\omega} : T\mathcal{B} \to T\mathcal{B}$. Equation (3.1.6) reads now

$$\dot{\mathbf{y}} = \hat{\omega}\mathbf{y} \tag{3.1.8}$$

The above formula gives the velocity of a spatial vector caused *solely* by the rotational motion. The operator $\hat{\omega}$ bears the name of the *spatial angular velocity* tensor. By analogy $\hat{\mathbf{\Omega}}$ is called the *referential*[1] *angular velocity* tensor.

Let $\hat{\omega}$ be constant (whereas $\hat{\mathbf{\Omega}}$ need not be), so that (3.1.8) becomes the homogeneous system of linear ordinary differential equations with constant coefficients. The solution to (3.1.8) can be expressed the following form [**14**, pp. 110-111]

$$\mathbf{y}(t) = \exp(t\hat{\omega})\,\mathbf{y}(0) \tag{3.1.9}$$

where $\exp(\cdot)$ is the matrix exponential, yet to be commented on. Equation (3.1.8) can be also rewritten in the referential form

$$\dot{\mathbf{Y}} = \hat{\mathbf{\Omega}}\mathbf{Y} \tag{3.1.10}$$

where $\dot{\mathbf{Y}} = \mathbf{\Lambda}^T\dot{\mathbf{y}}$. Similarly, if one assumes $\hat{\mathbf{\Omega}}$ to be constant (whereas $\hat{\omega}$ need not be), the solution to (3.1.10) follows

$$\mathbf{Y}(t) = \exp\left(t\hat{\mathbf{\Omega}}\right)\mathbf{Y}(0) \tag{3.1.11}$$

For both cases, one can compute $\mathbf{y}(t)$ as

$$\mathbf{y}(t) = \left[\exp(t\hat{\omega})\,\mathbf{\Lambda}(0)\right]\mathbf{Y}(0) \tag{3.1.12}$$

$$\mathbf{y}(t) = \left[\mathbf{\Lambda}(0)\exp\left(t\hat{\mathbf{\Omega}}\right)\right]\mathbf{Y}(0) \tag{3.1.13}$$

where the terms in brackets $[\cdot]$ are respectively called the spatial and the referential *compound rotations* [**146**, p. 29]. In case neither $\hat{\omega}$ nor $\hat{\mathbf{\Omega}}$ are constant, the above formulae still provide a good (first order) estimate of the rotation update for $t \to 0$. This feature is often utilised in the numerical context.

The matrix exponential $\exp(\cdot)$ is defined as follows

$$\exp(\mathbf{A}) = \mathbf{I} + \mathbf{A} + \frac{\mathbf{A}^2}{2!} + \frac{\mathbf{A}^3}{3!} + ... \tag{3.1.14}$$

where $\mathbf{A} : R^n \to R^n$ is a linear operator and $\mathbf{I}$ is the identity. It is easy to show that the above series converges uniformly (at rate independent of the argument) if only $\mathbf{A}$ is bounded (it does not stretch the unit ball in R^n infinitely) [**14**, p. 105]. One can consider a one-parameter family of linear operators $\exp(t\mathbf{A}) : R^n \to R^n$. It can be shown that this is a one-parameter *group* of linear operators [**14**, p. 109], that is

$$\exp((t+s)\,\mathbf{A}) = \exp(t\mathbf{A})\exp(s\mathbf{A}) \tag{3.1.15}$$

and

$$\frac{d}{dt}\exp(t\mathbf{A}) = \mathbf{A}\exp(t\mathbf{A}) \text{ is defined for all } t \tag{3.1.16}$$

The above defined group is *commutative*: $\exp(t\mathbf{A})\exp(s\mathbf{A}) = \exp(s\mathbf{A})\exp(t\mathbf{A})$. Another useful property follows from the definition of the matrix exponential (3.1.14) and the group property (3.1.15): action of $\exp(\cdot)$ on skew-symmetric matrices produces orthogonal operators. This could be anticipated from (3.1.12) and (3.1.13),

[1] *material*, *convected* or *body-frame* are also used in the literature

although now it is clear that $\mathbf{\Lambda}^T\mathbf{\Lambda} = \exp\left(\hat{\mathbf{\Omega}}^T\right)\exp\left(\hat{\mathbf{\Omega}}\right) = \exp\left(-\hat{\mathbf{\Omega}}\right)\exp\left(\hat{\mathbf{\Omega}}\right) = \exp(\mathbf{0}) = \mathbf{I}$, where $\mathbf{\Lambda} = \exp\left(\hat{\mathbf{\Omega}}\right)$ was assumed. It is easy to realise that rotations do not commute in general ($\mathbf{\Lambda}_1\mathbf{\Lambda}_2 \neq \mathbf{\Lambda}_2\mathbf{\Lambda}_1$, rotate a pencil about the horizontal and then the vertical axes and then swap the order). The rotation group $SO(3)$ is not commutative. Nevertheless, the one-parameter group $\mathbf{\Lambda}^t = \exp(t\hat{\omega})$ is commutative. Experience suggests that this corresponds to the rotation about a fixed axis, where indeed the final effect does not depend on the order in which the rotations are being applied. $\mathbf{\Lambda}^t$ can be interpreted as a curve on the surface of $SO(3)$, starting at the point $\mathbf{I}$. After (3.1.16) the velocity of $\mathbf{\Lambda}^0$ along $\mathbf{\Lambda}^t$ reads $\frac{d}{dt}\Lambda^t\big|_{t=0} = \hat{\omega}$ which confirms that $\hat{\omega} \in T_{\mathbf{I}}SO(3)$ (note that $\hat{\omega} \equiv \hat{\mathbf{\Omega}}$ at $\mathbf{I}$). Generally, the velocity along $\mathbf{\Lambda}^t$ at some point $\mathbf{\Lambda}^s$ reads $\frac{d}{dt}\Lambda^t\big|_{t=s} = \hat{\omega}\mathbf{\Lambda}^s$. By definition of the tangent space $\hat{\omega}\mathbf{\Lambda}^s \in T_{\mathbf{\Lambda}^s}SO(3)$. Indeed, as $\hat{\omega} = \mathbf{\Lambda}\hat{\mathbf{\Omega}}\mathbf{\Lambda}^T$, there holds $\hat{\omega}\mathbf{\Lambda} = \mathbf{\Lambda}\hat{\mathbf{\Omega}}$ and it was already demonstrated, that $\mathbf{\Lambda}\hat{\mathbf{\Omega}} \in T_{\mathbf{\Lambda}}SO(3)$.

The matrix exponential (3.1.14) applied in the context of the group $SO(3)$ is also called the *exponential map*. This term is traditionally used in the theory of *Lie groups* (groups L, where the internal operation $L \times L \to L$ is continuous and differentiable), where the exponential map acts on the elements of the tangent space at identity (called *Lie algebra*) and produces elements of the Lie group. This is exactly the case with $SO(3)$ [**146**, pp. 27-32]. The practical utility of the exponential map results here from the fact that (3.1.14) enjoys a closed form sum

$$\exp[\mathbf{\Psi}] = \mathbf{I} + \frac{\sin\|\mathbf{\Psi}\|}{\|\mathbf{\Psi}\|}\hat{\mathbf{\Psi}} + \frac{1-\cos\|\mathbf{\Psi}\|}{\|\mathbf{\Psi}\|^2}\hat{\mathbf{\Psi}}^2 \tag{3.1.17}$$

due to Euler and Rodrigues [**100**]. The above expression is often addressed as the Rodrigues formula. The argument $\mathbf{\Psi}$ and the operator $\hat{[\cdot]}$ require some further explanation. Anti-symmetric matrices in R^n generally have $n(n-1)/2$ components. It happens that $3(3-1)/2 = 3$, so that there is a one-to-one correspondence between the 3×3 anti-symmetric matrices and vectors in R^3. Namely

$$\hat{\mathbf{\Psi}} = \begin{bmatrix} 0 & -\Psi_3 & \Psi_2 \\ \Psi_3 & 0 & -\Psi_1 \\ -\Psi_2 & \Psi_1 & 0 \end{bmatrix} \tag{3.1.18}$$

where $\mathbf{\Psi} \in TE^3$ and $\hat{\mathbf{\Psi}} \in T_{\mathbf{I}}SO(3)$. Note, that $\mathbf{\Psi}$ acts as an argument to the mapping outputting a *point* on the surface of $SO(3)$. Thus it is natural to interpret $\mathbf{\Psi}$ as a point in E^3 (rotations do not commute - one does not add points). On the other hand, $\mathbf{\Psi}$ remains in correspondence with the skew-symmetric matrices $\hat{\mathbf{\Psi}} \in T_{\mathbf{I}}SO(3)$ and in this context it is most conveniently interpreted as a vector. This notional duality needs to be kept in mind. Vector $\mathbf{\Psi}$ is called the *axial vector* of the skew-symmetric matrix $\hat{\mathbf{\Psi}}$. This convention allows to interpret ω and $\mathbf{\Omega}$ as respectively the spatial and the referential *angular velocity vectors*. It is easy to notice, that action of the skew-symmetric operator (3.1.18) on a vector parallels the usual *vector product* formula

$$\hat{\omega}\mathbf{y} = \omega \times \mathbf{y} \tag{3.1.19}$$

Formulae (3.1.18) and (3.1.19) establish an *isomorphism* (invertible, structure preserving map) between the spaces TE^3 and $T_{\mathbf{I}}SO(3)$ (denoted as $TE^3 \cong T_{\mathbf{I}}SO(3)$). In practice it is often more efficient to operate on vectors, rather than skew-symmetric matrices. For example, formula (3.1.7) takes the simple form

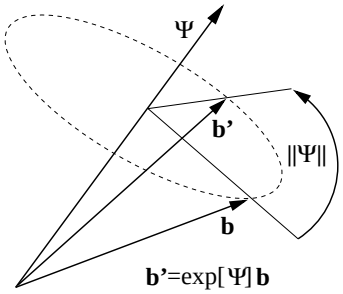


FIGURE 3.1.1. The finite rotation vector $\boldsymbol{\Psi}$ and the action of the exponential map $\exp[\boldsymbol{\Psi}]$.

$$\omega = \boldsymbol{\Lambda}\boldsymbol{\Omega} \tag{3.1.20}$$

when vectors are used instead of the skew matrices.

Recall, that the curve $\Lambda^t = \exp(t\hat{\omega})$ was interpreted as the rotation along a fixed axis. $\boldsymbol{\Psi}$ acting as an argument of (3.1.17) can be interpreted as the *rotation vector*, collinear with the fixed axis. Note, that $\exp[\boldsymbol{\Psi}]\,\boldsymbol{\Psi} = \boldsymbol{\Psi}$ (as $\hat{\boldsymbol{\Psi}}\boldsymbol{\Psi} = \mathbf{0}$) so that $\boldsymbol{\Psi}$ does not rotate vectors coaxial with itself. Geometrically, the action of the operator $\exp[\boldsymbol{\Psi}]$ can be interpreted as the rotation of magnitude $\|\boldsymbol{\Psi}\|$ about the axis collinear with $\boldsymbol{\Psi}$ (Figure **3.1.1**). Thus, the rotation vector based parametrisation of $SO(3)$ is singular on spheres $\|\Psi\| = 2\pi n$, $n \in \{1, 2, ...\}$ in the sense that these subsets of the Euclidean 3−space are mapped into the single identity element of the rotation space. Nevertheless, the singularity can be avoided either by the adaptation of the incremental formulation (3.1.12), (3.1.14) (the magnitudes of the rotation increments need to be smaller than 2π), or by a suitable re-parametrisation [**146**, p. 26]. The singularity of the map $\exp[\boldsymbol{\Psi}] : E^3 \to SO(3)$ is related to the intrinsic incompatibility between the shapes of subsets of E^3 and the manifold $SO(3)$. Although $SO(3)$ is locally Euclidean (looks like E^3 in the neighbourhood of each point) it cannot be spread in E^3 without making a hole in it. Or conversely, one cannot wrap E^3 around $SO(3)$ without having some bits of E^3 overlapping (somewhat more rigorous discussion can be found in [**146**, pp. 25-26]). Similarly, a sphere in E^3 can be locally deformed into a flat area, although there is no way to spread it over a planar surface without some damage. The sphere analogy is in fact quite adequate, as the *quaternion* parametrisation allows to interpret $SO(3)$ as the unit sphere embedded in E^4 [**100**].

One more thing to discuss is the relation between tangent spaces at different points of $SO(3)$. Let us consider $\boldsymbol{\Lambda} = \exp[\boldsymbol{\Psi}]$. One can perceive $\boldsymbol{\Psi}$ as a point in E^3 for which a corresponding point in $SO(3)$ can be found through $\exp[\cdot]$. It is natural to ask, how a perturbation of the point $\boldsymbol{\Psi}$ affects the point $\boldsymbol{\Lambda}$. Hence, a linearisation of the relation $\delta\boldsymbol{\Lambda} = \exp[\boldsymbol{\Psi} + \delta\boldsymbol{\Psi}]$ is sought. We already know that $\delta\boldsymbol{\Lambda} = \delta\hat{\omega}\boldsymbol{\Lambda} = \boldsymbol{\Lambda}\delta\hat{\boldsymbol{\Omega}} \in T_{\boldsymbol{\Lambda}}SO(3)$, which is simply another way of writing down the velocity relations. The linear variation of $\exp[\boldsymbol{\Psi} + \delta\boldsymbol{\Psi}]$ with respect to the perturbation vector $\delta\boldsymbol{\Psi}$ is delivered by the differential of $\exp[\cdot]$. There holds

$$\mathrm{dexp}[\boldsymbol{\Psi}] = \frac{\partial \exp[\boldsymbol{\Psi}]}{\partial \boldsymbol{\Psi}}\delta\boldsymbol{\Psi} = \left.\frac{d}{ds}\exp[\boldsymbol{\Psi} + s\delta\boldsymbol{\Psi}]\right|_{s=0} \tag{3.1.21}$$

where $\delta\boldsymbol{\Psi} \in TE^3 \cong T_{\mathbf{I}}SO(3)$ is arbitrary. One can write now

$$\delta\hat{\mathbf{\Omega}} = \mathbf{\Lambda}^T \frac{\partial \exp[\mathbf{\Psi}]}{\partial \mathbf{\Psi}} \delta\mathbf{\Psi} \tag{3.1.22}$$

or equivalently

$$\delta\hat{\mathbf{\Omega}} = \exp[-\mathbf{\Psi}] \frac{d}{ds} \exp[\mathbf{\Psi} + s\delta\Psi]\Big|_{s=0} \tag{3.1.23}$$

It is seen that $\mathbf{\Lambda}^T \frac{\partial \exp[\mathbf{\Psi}]}{\partial \mathbf{\Psi}}$ is a third-order object, which contracted with $\delta\mathbf{\Psi}$ delivers a skew-symmetric operator. One can thus use only three components of (3.1.22) in order to create a relation between the axial vectors

$$\delta\mathbf{\Omega} = \mathbf{T}^T \delta\mathbf{\Psi} \tag{3.1.24}$$

A lengthy and somewhat tedious calculation (Ibrahimbegović [**99**], Crisfield [**144**], Ritto-Corrêa [**145**]) leads to the following simple form of $\mathbf{T}$

$$\mathbf{T} = \mathbf{I} + \frac{1 - \cos\|\mathbf{\Psi}\|}{\|\mathbf{\Psi}\|^2} \hat{\mathbf{\Psi}} + \frac{\|\mathbf{\Psi}\| - \sin\|\mathbf{\Psi}\|}{\|\mathbf{\Psi}\|^3} \hat{\mathbf{\Psi}}^2 \tag{3.1.25}$$

One can also establish a relation between the spatial perturbation $\delta\omega$ and $\delta\mathbf{\Psi}$ by processing the relation $\delta\hat{\omega} = \frac{\partial \exp[\mathbf{\Psi}]}{\partial \mathbf{\Psi}} \delta\mathbf{\Psi}\mathbf{\Lambda}^T$. The resultant formula reads [**99, 145**]

$$\delta\omega = \mathbf{T}\delta\mathbf{\Psi} \tag{3.1.26}$$

Due to (3.1.24), (3.1.26) and the transformation between the spatial and referential angular velocity vectors (3.1.20), there holds

$$\mathbf{T}^T = \mathbf{\Lambda}^T \mathbf{T} \tag{3.1.27}$$

Transposition and right-multiplication by $\mathbf{\Lambda}^T$ leads to

$$\mathbf{T}^T = \mathbf{T}\mathbf{\Lambda}^T \tag{3.1.28}$$

so that $\mathbf{\Lambda}$ and $\mathbf{T}$ commute. One can see from (3.1.17) and (3.1.25) that $\mathbf{\Lambda}$ and $\mathbf{T}$ share eigenvectors and thus commute [**99**].

The operator $\mathbf{T}$ establishes a connection between the tangent spaces $T_{\mathbf{I}}SO(3)$ and $T_{\mathbf{\Lambda}}SO(3)$, where $\mathbf{\Lambda}$ is a point at $t = 1$ on the curve $\mathbf{\Lambda}^t = \exp[t\mathbf{\Psi}]$. In order to picture this graphically, it is convenient to make a notional distinction between the spatial and the material tangent spaces at $\mathbf{\Lambda}$. In reference to the algebraic form of the constraint $\mathbf{\Lambda}^T\mathbf{\Lambda} - \mathbf{I}$, it is natural to speak about velocities $\dot{\mathbf{\Lambda}}$ fulfilling (3.1.3) as elements of the tangent space $T_{\mathbf{\Lambda}}SO(3)$. It then follows that pre-multiplying $\mathbf{\Lambda}$ by an anti-symmetric spatial angular velocity ω (or perturbation $\delta\omega$), or post-multiplying it by a referential (material) angular velocity $\mathbf{\Omega}$ (or perturbation $\delta\mathbf{\Omega}$), creates an element of the tangent space $T_{\mathbf{\Lambda}}SO(3)$. This is in analogy with the addition of vectors to points in E^3, although the lack of commutativity of the rotation group makes it necessary to speak about the left-multiplication and the right-multiplication separately. Thinking about anti-symmetric operators acting on a point $\mathbf{\Lambda}$, one can than introduce the notion of a spatial and material tangent spaces $T^{spa}_{\mathbf{\Lambda}}SO(3)$ and $T^{mat}_{\mathbf{\Lambda}}SO(3)$, composed respectively of all spatial and referential (material) angular velocities (perturbations) acting on $\mathbf{\Lambda}$. In this respect, there holds $\mathbf{T} : T_{\mathbf{I}}SO(3) \rightarrow T^{spa}_{\mathbf{\Lambda}}SO(3)$ and $\mathbf{T}^T : T_{\mathbf{I}}SO(3) \rightarrow T^{mat}_{\mathbf{\Lambda}}SO(3)$. This is illustrated in Figure 3.1.2.

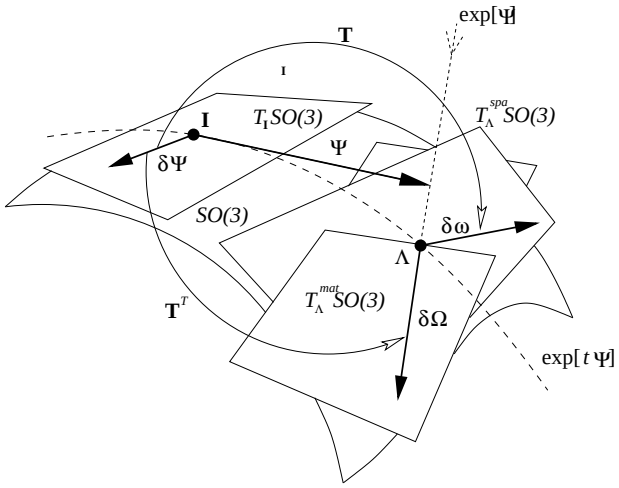


FIGURE 3.1.2. Interpretation of tangent spaces on $SO(3)$ together with the actions of $\exp[\cdot]$ and $\mathbf{T}$ operators.

It is finally relevant to comment on the practical utility of the operator $\mathbf{T}$. Elements of tangent vector spaces $T_{\mathbf{\Lambda}}SO(3)$ can be added to one another only if they are all of the same kind (spatial or material) and all based at the same point $\mathbf{\Lambda}$. That is, for $\mathbf{\Omega}_1 \in T^{mat}_{\mathbf{\Lambda}_1}SO(3)$ and $\mathbf{\Omega}_2 \in T^{mat}_{\mathbf{\Lambda}_2}SO(3)$ it is meaningful to consider the sum $\mathbf{\Omega}_1 + \mathbf{\Omega}_2$ only if $\mathbf{\Lambda}_1 = \mathbf{\Lambda}_2$. In case $\mathbf{\Lambda}_1 \neq \mathbf{\Lambda}_2$, one of the vectors needs to be brought into the tangent space of the other one. $\mathbf{T}$ provides a specific instance of such operation, of importance in numerical realisations.

Rigid kinematics

(1) Motion

$$\mathbf{x}(\mathbf{X},t) = \mathbf{\Lambda}(t)\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \bar{\mathbf{x}}(t)$$

$$\mathbf{x}, \bar{\mathbf{x}} \in \mathcal{B}, \;\; \mathbf{X}, \bar{\mathbf{X}} \in \mathcal{B}_0, \;\; \mathbf{\Lambda} \in SO(3)$$

(2) Velocity

$$\dot{\mathbf{x}}(\mathbf{X},t) = \dot{\mathbf{\Lambda}}(t)\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \dot{\bar{\mathbf{x}}}(t)$$

$$\dot{\mathbf{x}}, \dot{\bar{\mathbf{x}}} \in T\mathcal{B}, \;\; \dot{\mathbf{\Lambda}} = \mathbf{\Lambda}\hat{\mathbf{\Omega}} = \hat{\omega}\mathbf{\Lambda} \in T_{\mathbf{\Lambda}}SO(3)$$

$$\hat{\omega} = \mathbf{\Lambda}\hat{\mathbf{\Omega}}\mathbf{\Lambda}^T \Leftrightarrow \omega = \mathbf{\Lambda}\mathbf{\Omega}$$

(3) Parametrisation

$$\mathbf{\Lambda}(\mathbf{\Psi}) = \exp[\mathbf{\Psi}], \;\; \mathbf{\Psi} \in E^3, \;\; \mathbf{\Lambda} \in SO(3)$$

$$\delta\mathbf{\Omega} = \mathbf{T}^T\delta\mathbf{\Psi}, \;\; \delta\omega = \mathbf{T}\delta\mathbf{\Psi}$$

$$\mathbf{\Lambda}\delta\hat{\mathbf{\Omega}}, \delta\hat{\omega}\mathbf{\Lambda} \in T_{\mathbf{\Lambda}}SO(3), \;\; \delta\mathbf{\Psi} \in TE^3 \cong T_ISO(3)$$

$$\exp[\mathbf{\Psi}] = \mathbf{I} + \frac{\sin\|\mathbf{\Psi}\|}{\|\mathbf{\Psi}\|}\hat{\mathbf{\Psi}} + \frac{1-\cos\|\mathbf{\Psi}\|}{\|\mathbf{\Psi}\|^2}\hat{\mathbf{\Psi}}^2$$

$$\mathbf{T} = \mathbf{I} + \frac{1-\cos\|\mathbf{\Psi}\|}{\|\mathbf{\Psi}\|^2}\hat{\mathbf{\Psi}} + \frac{\|\mathbf{\Psi}\| - \sin\|\mathbf{\Psi}\|}{\|\mathbf{\Psi}\|^3}\hat{\mathbf{\Psi}}^2$$

$$\hat{\mathbf{\Psi}} = \begin{bmatrix} 0 & -\Psi_3 & \Psi_2 \\ \Psi_3 & 0 & -\Psi_1 \\ -\Psi_2 & \Psi_1 & 0 \end{bmatrix}, \;\; \|\mathbf{\Psi}\| = \sqrt{\langle \mathbf{\Psi}, \mathbf{\Psi} \rangle}$$

3.2. Pseudo-rigid body

The motion of a pseudo-rigid body reads

$$\mathbf{x}(t) = \mathbf{F}(t)\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \bar{\mathbf{x}}(t) \tag{3.2.1}$$

where $\mathbf{x}$ is the current image of a referential point $\mathbf{X}$, $\mathbf{F}$ is a spatially homogeneous deformation gradient ($\mathbf{F} = \partial\chi/\partial\mathbf{X}$), $\bar{\mathbf{X}}$ is a selected referential point and $\bar{\mathbf{x}} = \bar{\mathbf{x}}(t)$ is the current image of $\bar{\mathbf{X}}$. Deformation gradient $\mathbf{F}$, being an invertible and orientation preserving ($\det(\mathbf{F}) > 0$) operator, belongs to the subgroup $GL_+(3,R)$ of the general linear group $GL(3,R)$ (group of real, invertible, 3×3 matrices). The constraint $\det(\mathbf{F}) > 0$ indicates that $GL_+(3,R)$ is an open subset of the twelve-component space of all 3×3 matrices, trivially isomorphic with the Euclidean space E^9. Hence, the configuration space of a pseudo-rigid body

$$\mathcal{Q}^{prb} = GL_+(3,R) \times E^3 \tag{3.2.2}$$

is a smooth manifold of dimension twelve. The velocity reads

$$\dot{\mathbf{x}}(t) = \dot{\mathbf{F}}(t)\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \dot{\bar{\mathbf{x}}}(t) \tag{3.2.3}$$

where, contrary to the rigid body case (3.1.5), no special treatment of $\dot{\mathbf{F}}$ is necessary. This results from the fact, than the inequality constraint $\det(\mathbf{F}) > 0$ does not reduce the dimension of the configuration space. By definition, every point $\mathbf{F}$ has an open neighbourhood contained in $GL_+(3,R)$. Thus all velocities $\dot{\mathbf{F}}$ are eligible, as an instantaneous departure from $GL_+(3,R)$ is not possible. This can be shown on the following example. Assume $\mathbf{F}(0) = \mathbf{I}$ and $\dot{\mathbf{F}}(0) = \mathbf{A}$. Then $\det(\mathbf{I} + t\mathbf{A}) = 1 + t\sum_i A_{ii} + O\left(t^2\right)$ [**14**, p. 116], so that $\det(\mathbf{I} + t\mathbf{A}) > 0$ for sufficiently small t.

Instead of using the $\mathbf{a} = [a_1, a_2, ..., a_9]^T$ notation for the coordinates of points in E^9, one can arrange them into the matrix form

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \tag{3.2.4}$$

and define a binary operation $E^9 \times E^9 \to E^9$ equivalent to the matrix product. This establishes the isomorphism $\phi : E^9 \to GL_+(3,R)$, so that $\mathbf{A} = \phi(\mathbf{a})$ and $\mathbf{a} = \phi^{-1}(\mathbf{A})$. Similarly as for rotations and the $\exp[\cdot]$ mapping, one can ask what is the linearised relation between the perturbations $\delta\mathbf{A} = \phi(\mathbf{a} + \delta\mathbf{a})$. Obviously $\delta\mathbf{A} = \frac{d}{dt}\phi(\mathbf{a} + t\delta\mathbf{a})\big|_{t=0} = \phi(\delta\mathbf{a})$. As all tangent spaces of E^9 are identical and can be parametrised by the standard base $\mathbf{e}_i = [0, 0, ..., 1_i, 0, 0...]^T$, one is free to add vectors within TE^9. One can define $\delta\mathbf{A} + \delta\mathbf{B} = \phi\left(\phi^{-1}(\delta\mathbf{A}) + \phi^{-1}(\delta\mathbf{B})\right)$, which is in fact the usual matrix addition. It follows that all tangent spaces of $GL_+(3,R)$ are identical. Thus, one can add velocities $\dot{\mathbf{F}}(t) + \dot{\mathbf{F}}(s)$ for all t, s.

The structure of the configuration space $\mathcal{Q}^{prb}$ is then simpler than that of $\mathcal{Q}^{rig}$. The above summary exhausts most of the points previously discussed for the rigid body. One more analogy can be drawn. As $\mathbf{x} - \bar{\mathbf{x}} \in T\mathcal{B}$ and $\mathbf{X} - \bar{\mathbf{X}} \in T\mathcal{B}_0$, it follows that $\mathbf{F}, \dot{\mathbf{F}} : T\mathcal{B}_0 \to T\mathcal{B}$ are two-point objects. One can define a spatial object $\mathbf{L} : T\mathcal{B} \to T\mathcal{B}$, similar to the spatial angular velocity $\hat{\omega}$

$$\mathbf{L} = \dot{\mathbf{F}}\mathbf{F}^{-1} \tag{3.2.5}$$

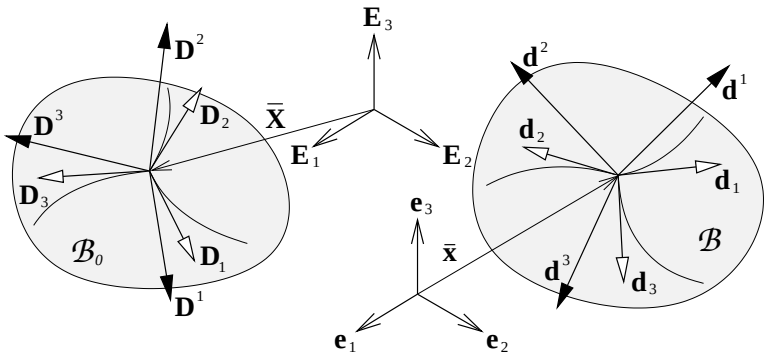


FIGURE 3.2.1. Two sets of base vectors $\mathbf{D}_i, \mathbf{D}^i$ and $\mathbf{d}_i, \mathbf{d}^i$ spanning tangent and cotangent spaces $T\mathcal{B}_0, T^*\mathcal{B}_0$ and $T\mathcal{B}, T^*\mathcal{B}$.

$\mathbf{L}$ is called the *deforming tensor* [**46**, p. 19]. One can decompose $\mathbf{L}$ into the anti-symmetric *spin tensor* $\mathbf{O}$ and the symmetric *stretching tensor* $\mathbf{D}$.

$$\mathbf{L} = \mathbf{O} + \mathbf{D} \tag{3.2.6}$$

$$\mathbf{O} = \frac{1}{2}\left(\mathbf{L} - \mathbf{L}^T\right), \quad \mathbf{D} = \frac{1}{2}\left(\mathbf{L} + \mathbf{L}^T\right) \tag{3.2.7}$$

By utilising the polar decomposition of $\mathbf{F}$

$$\mathbf{F} = \mathbf{\Lambda}\mathbf{U} \tag{3.2.8}$$

where $\mathbf{\Lambda}$ is orthogonal and $\mathbf{U}$ is symmetric positive-definite, it is possible to express the spin and stretching tensors as [**46**, p. 19]

$$\mathbf{O} = \dot{\mathbf{\Lambda}}\mathbf{\Lambda}^T + \frac{1}{2}\mathbf{\Lambda}\left(\dot{\mathbf{U}}\mathbf{U}^{-1} - \mathbf{U}^{-1}\dot{\mathbf{U}}\right)\mathbf{\Lambda}^T \tag{3.2.9}$$

$$\mathbf{D} = \frac{1}{2}\mathbf{\Lambda}\left(\dot{\mathbf{U}}\mathbf{U}^{-1} + \mathbf{U}^{-1}\dot{\mathbf{U}}\right)\mathbf{\Lambda}^T \tag{3.2.10}$$

Clearly, in the absence of deformation $\mathbf{U} = \mathbf{I}$, the stretch tensor $\mathbf{D}$ is zero. For $\mathbf{U} = \mathbf{I}$, the spin tensor becomes anti-symmetric and equal to the spatial angular velocity $\mathbf{O} \equiv \hat{\omega}|_{\mathbf{U}=\mathbf{I}}$.

Pseudo-rigid kinematics

(1) Motion

$$\mathbf{x}(\mathbf{X}, t) = \mathbf{F}(t)\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \bar{\mathbf{x}}(t)$$

$$\mathbf{x}, \bar{\mathbf{x}} \in \mathcal{B}, \;\; \mathbf{X}, \bar{\mathbf{X}} \in \mathcal{B}_0, \;\; \mathbf{F} \in GL_+(3, R)$$

(2) Velocity

$$\dot{\mathbf{x}}(\mathbf{X}, t) = \dot{\mathbf{F}}(t)\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \dot{\bar{\mathbf{x}}}(t)$$

$$\dot{\mathbf{x}}(\mathbf{X}, t) = \mathbf{L}(t)\left(\mathbf{x} - \bar{\mathbf{x}}\right) + \dot{\bar{\mathbf{x}}}(t)$$

$$\dot{\mathbf{x}}, \dot{\bar{\mathbf{x}}} \in T\mathcal{B}, \;\; \dot{\mathbf{F}} = \mathbf{L}\mathbf{F} \in T_{\mathbf{F}}GL_+(3, R)$$

3.2.1. Kinematics of vectors and tensors. It is useful to investigate, how the assumed motion carries over geometrical objects attached to a body. Let $T^*\mathcal{B}_0, T^*\mathcal{B}$ be the covector (dual) spaces of $T\mathcal{B}_0, T\mathcal{B}$, where the *dual space* is composed of all linear functionals (one-forms, covectors) acting on the elements of the tangent space (vectors). The distinction between vectors and covectors is made for the sake of clarity, while it is acknowledged that R^n is equal to its dual [**125**, p. 121]. Let base vectors $\mathbf{D}_i, \mathbf{d}_i$ and covectors $\mathbf{D}^i, \mathbf{d}^i$ respectively span the tangent spaces $T\mathcal{B}_0, T\mathcal{B}$ and their duals $T^*\mathcal{B}_0, T^*\mathcal{B}$ (Figure 3.2.1). It is assumed that $\mathbf{D}_i\mathbf{D}^j = \mathbf{d}_i\mathbf{d}^j = \delta_i^j$, where the Kronecker delta is defined as $\delta_i^j = \{0 \text{ if } i \neq j \text{ or } 1 \text{ if } i = j\}$. The deformation gradient can be expressed as

$$\mathbf{F}(t) = \mathbf{d}_i(t) \otimes \mathbf{D}^i \tag{3.2.11}$$

where $\otimes$ stands for the dyadic product ($\mathbf{a} \otimes \mathbf{b} = a^i b^j \mathbf{e}_i \otimes \mathbf{e}_j$) and the Einstein summation convention for repeated lower and upper indices is adopted (this holds in the remaining part of this section) [**46**, p. 44]. Note that usually it is convenient to assume $\mathbf{D}_i = \mathbf{D}^i = \mathbf{E}_i = \mathbf{E}^i$ so that $\mathbf{F}$ can be viewed as composed of column vectors $\mathbf{d}_i$. Through (3.2.1) and (3.2.11) the motion can be regarded as superposition of translation and distortion of a coordinate system attached to a selected material point. One can conceptually associate various tensor entities with the frames $\mathbf{d}_i(t), \mathbf{d}^i(t)$. The time dependent distortion of $\mathbf{d}_i, \mathbf{d}^i$ gives rise to several forms of tensor rates, depending on the selection of an observer and the nature of involved objects.

Let us recall that the action of a covector $\mathbf{n} = \eta_i \mathbf{d}^i$ on a vector $\mathbf{a} = \alpha^i \mathbf{d}_i$ is defined as $\langle \mathbf{n}, \mathbf{a} \rangle = \eta_i \mathbf{d}^i \alpha^j \mathbf{d}_j = \eta_i \alpha^j \delta_j^i = \eta_i \alpha^i$. This, together with (3.2.11) allows to deduce the basic transformation laws for vectors and covectors

$$\mathbf{a} = \mathbf{F}\mathbf{A} \tag{3.2.12}$$

$$\mathbf{N} = \mathbf{F}^T \mathbf{n} \tag{3.2.13}$$

where

$$\mathbf{A} = \alpha^i \mathbf{D}_i, \quad \mathbf{a} = \alpha^i \mathbf{d}_i \tag{3.2.14}$$

$$\mathbf{n} = \eta_i \mathbf{d}^i, \quad \mathbf{N} = \eta_i \mathbf{D}^i \tag{3.2.15}$$

Note, that the actual coordinates with respect to the bases $\mathbf{D}, \mathbf{d}$ do not change, therefore $\mathbf{a}, \mathbf{A}$ and $\mathbf{n}, \mathbf{N}$ are called the *codeforming* vectors and covectors. From (3.2.12) and (3.2.13) it is seen that while $\mathbf{F} : T\mathcal{B}_0 \to T\mathcal{B}$, its transpose $\mathbf{F}^T : T^*\mathcal{B} \to T^*\mathcal{B}_0$. Let us consider $\mathbf{A}$ and $\mathbf{N}$, fixed in the reference configuration. The velocity of current images of $\mathbf{A}$ and $\mathbf{N}$ reads

$$\dot{\mathbf{a}} = \dot{\mathbf{F}}\mathbf{A} = \dot{\mathbf{F}}\mathbf{F}^{-1}\mathbf{a} = \mathbf{L}\mathbf{a} \tag{3.2.16}$$

$$\dot{\mathbf{n}} = \dot{\mathbf{F}}^{-T}\mathbf{N} = \dot{\mathbf{F}}^{-T}\mathbf{F}^T\mathbf{n} = -\left(\mathbf{F}^{-1}\dot{\mathbf{F}}\mathbf{F}^{-1}\right)^T \mathbf{F}^T\mathbf{n} = -\mathbf{L}^T\mathbf{n} \tag{3.2.17}$$

where differentiation of $\mathbf{F}\mathbf{F}^{-1}$ was exploited in the third step of (3.2.17). The deforming tensor $\mathbf{L}$ can be expressed as

$$\mathbf{L} = \dot{\mathbf{F}}\mathbf{F}^{-1} = \dot{\mathbf{d}}_i \otimes \mathbf{D}^i\mathbf{D}_j \otimes \mathbf{d}^j = \dot{\mathbf{d}}_j \otimes \mathbf{d}^j = L_j^i \mathbf{d}_i \otimes \mathbf{d}^j \tag{3.2.18}$$

where L_j^i are the components of the velocity of $\mathbf{d}_j$ expressed in (the same) basis $\mathbf{d}_i$. The deforming tensor can be used to obtain velocities of vectors (3.2.16) and

covectors (3.2.17) convected with the body. In a more specific situation of an ant walking on the surface of a pseudo-rigid body, the velocity of the ant will follow from a chain rule of differentiation. The relative motion of the ant can be parametrised by a vector-valued function

$$\mathbf{a}(t) = \alpha^i(t)\,\mathbf{d}_i \tag{3.2.19}$$

and thus the velocity reads

$$\dot{\mathbf{a}} = \dot{\alpha}^i\mathbf{d}_i + \alpha^i\dot{\mathbf{d}}_i = \overset{\diamond}{\mathbf{a}} + \mathbf{L}\mathbf{a} \tag{3.2.20}$$

The symbol $\overset{\diamond}{\mathbf{a}}$ stands for the *codeforming derivative* [**46**, p. 45] of the vector function $\mathbf{a}(t)$. The codeforming derivative describes the velocity of a point travelling through a moving pseudo-rigid body, viewed from the perspective of an observer embedded in the deforming frame $\mathbf{d}_i$. In other words

$$\overset{\diamond}{\mathbf{a}} = \dot{\alpha}^i\mathbf{d}_i = \dot{\mathbf{a}} - \mathbf{L}\mathbf{a} \tag{3.2.21}$$

A similar exercise can be made for a covector-valued function

$$\mathbf{n}(t) = \eta_i(t)\,\mathbf{d}^i \tag{3.2.22}$$

Now the codeforming derivative reads

$$\overset{\diamond}{\mathbf{n}} = \dot{\eta}_i\mathbf{d}^i = \dot{\mathbf{n}} + \mathbf{L}^T\mathbf{n} \tag{3.2.23}$$

It is quite easy to see that the codeforming derivative is in fact the *Lie derivative* with respect to the flow defined by the motion (3.2.1). For this to hold one needs to conceptually extend the (co)vector function $\mathbf{t}(t)$ into a constant (co)vector field $\mathbf{t}(\mathbf{x},t) = \mathbf{t}(t)$, defined on $\mathcal{B}$. The *flow* based on the motion $\chi(t)$ can be in general defined as $\chi_{s,t} = \chi(t)\chi(s)^{-1}$ [**147**, p. 95]. Note that $\chi_{r,t}\chi_{s,r} = \chi(t)\chi(r)^{-1}\chi(r)\chi(s)^{-1} = \chi_{s,t}$ and $\chi_{t,t}$ is the identity. Let $\mathbf{w} = \frac{d}{dt}\chi_{t,s}$ be the spatial velocity field on $\mathcal{B}$. The Lie derivative of $\mathbf{t}$ with respect to $\mathbf{w}$ is defined as

$$L_{\mathbf{w}}\mathbf{t} = \frac{d}{dt}\left(\chi^*_{t,s}\mathbf{t}(t)\right)\Big|_{t=s} \tag{3.2.24}$$

where $\chi^*_{t,s}$ is the *pull-back* operator related to $\mathbf{t}$. For fixed t and s the flow $\chi_{s,t}$ becomes a point mapping $\chi_{s,t} : \mathcal{B}_s \to \mathcal{B}_t$. For vectors, the *push-forward* $\chi_{s,t*}$ and the *pull-back* $\chi^*_{s,t}$ operators are respectively the Jacobian and the inverse Jacobian of $\chi_{s,t}$. Thus $\chi_{s,t*} : T\mathcal{B}_s \to T\mathcal{B}_t$ and $\chi^*_{s,t} : T\mathcal{B}_t \to T\mathcal{B}_s$ for vectors. The relevant operators for covectors are obtained in analogy to (3.2.13). For the pseudo-rigid case the flow related formulae read

$$\chi^{-1}(s) = \mathbf{F}^{-1}(s)\left(\mathbf{x} - \bar{\mathbf{x}}(s)\right) + \bar{\mathbf{X}} \tag{3.2.25}$$

$$\chi(t) = \mathbf{F}(t)\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \bar{\mathbf{x}}(t) \tag{3.2.26}$$

$$\chi_{s,t} = \mathbf{F}(t)\,\mathbf{F}^{-1}(s)\left(\mathbf{x} - \bar{\mathbf{x}}(s)\right) + \bar{\mathbf{x}}(t) \tag{3.2.27}$$

Thus for vectors, the push-forward and pull-back operators take the form

$$\chi_{s,t*} = \frac{\partial \chi_{s,t}}{\partial \mathbf{x}} = \mathbf{F}(t)\,\mathbf{F}^{-1}(s) \tag{3.2.28}$$

$$\chi^*_{s,t} = \left(\frac{\partial \chi_{s,t}}{\partial \mathbf{x}}\right)^{-1} = \mathbf{F}(s)\,\mathbf{F}^{-1}(t) \tag{3.2.29}$$

while for covectors, there holds

$$\chi_{s,t*} = \left(\frac{\partial \chi_{s,t}}{\partial \mathbf{x}}\right)^{-T} = \mathbf{F}^{-T}(t)\,\mathbf{F}^{T}(s) \tag{3.2.30}$$

$$\chi^*_{s,t} = \left(\frac{\partial \chi_{s,t}}{\partial \mathbf{x}}\right)^{T} = \mathbf{F}^{-T}(s)\,\mathbf{F}^{T}(t) \tag{3.2.31}$$

Formula (3.2.21) can be rewritten as

$$\begin{aligned} \overset{\diamond}{\mathbf{a}} &= \frac{d}{dt}\left(\mathbf{F}(s)\,\mathbf{F}^{-1}(t)\,\mathbf{a}(t)\right)\Big|_{t=s} \\ &= \left[\mathbf{F}_s\dot{\mathbf{F}}_t^{-1}\mathbf{a}_t + \mathbf{F}_s\mathbf{F}_t^{-1}\dot{\mathbf{a}}_t\right]_{t=s} \\ &= \mathbf{F}_s\mathbf{F}_s^{-1}\dot{\mathbf{a}}_s - \mathbf{F}_s\mathbf{F}_s^{-1}\dot{\mathbf{F}}_s\mathbf{F}_s^{-1}\mathbf{a}_s \\ &= \dot{\mathbf{a}} - \mathbf{L}\mathbf{a} \end{aligned} \tag{3.2.32}$$

Similarly (3.2.23) reads

$$\begin{aligned} \overset{\diamond}{\mathbf{n}} &= \frac{d}{dt}\left(\mathbf{F}^{-T}(s)\,\mathbf{F}^{T}(t)\,\mathbf{n}(t)\right)\Big|_{t=s} \\ &= \left[\mathbf{F}_s^{-T}\dot{\mathbf{F}}_t^{T}\mathbf{n}_t + \mathbf{F}_s^{-T}\mathbf{F}_t^{T}\dot{\mathbf{n}}_t\right]_{t=s} \\ &= \dot{\mathbf{n}} + \mathbf{L}^T\mathbf{n} \end{aligned} \tag{3.2.33}$$

Note that due to (3.2.16) and (3.2.17), $\overset{\diamond}{\mathbf{t}} = \mathbf{0}$ implies that the the (co)vector $\mathbf{t}$ is being convected with the flow of the motion. In this sense, the Lie derivative measures how much a time dependent (co)vector field fails to be convected with the motion.

A bilinear form $\mathbf{E} : \mathcal{N} \times \mathcal{K} \to R$, where $\mathbf{n} \in \mathcal{N}$, $\mathbf{k} \in \mathcal{K}$ and $\mathcal{N}, \mathcal{K} \in \{T\mathcal{B}_s, T^*\mathcal{B}_s\}$ is a linear form with respect to each of its arguments, $\mathbf{E}(\alpha\mathbf{a} + \beta\mathbf{b}, \cdot) = \alpha\mathbf{E}(\mathbf{a}, \cdot) + \beta\mathbf{E}(\mathbf{b}, \cdot)$, analogously for $\mathbf{E}(\cdot, \alpha\mathbf{c} + \beta\mathbf{d})$. One can define it as follows

$$\mathbf{E}(\mathbf{n}, \mathbf{k}) = \mathbf{n} \cdot \mathbf{E}\mathbf{k} \tag{3.2.34}$$

Conventionally [**147**, p. 65], if $\mathbf{n}, \mathbf{k} \in T^*\mathcal{B}_s$ then $\mathbf{E}$ is called a *contravariant* tensor, while for $\mathbf{n}, \mathbf{k} \in T\mathcal{B}_s$ it is called a *covariant* tensor. It is a mixed tensor otherwise. Let us focus on the contravariant case, as it will be of use in the next chapter. Let $\mathbf{E}$ be given by (3.2.34) for all $\mathbf{n}, \mathbf{k} \in T^*\mathcal{B}_s$. For any $\mathbf{p}, \mathbf{q} \in T^*\mathcal{B}_t$ one can obtain $\mathbf{n} = \chi^*_{s,t}\mathbf{p}$ and $\mathbf{k} = \chi^*_{s,t}\mathbf{q}$ and thus

$$\begin{aligned} \mathbf{E}(\mathbf{p}, \mathbf{q})|_{\mathbf{p},\mathbf{q} \in T^*\mathcal{B}_t} &= \mathbf{E}\left(\chi^*_{s,t}\mathbf{p}, \chi^*_{s,t}\mathbf{q}\right) \\ &= \left(\mathbf{F}_s^{-T}\mathbf{F}_t^{T}\mathbf{p}\right) \cdot \mathbf{E}\mathbf{F}_s^{-T}\mathbf{F}_t^{T}\mathbf{q} \\ &= \mathbf{p} \cdot \mathbf{F}_t\mathbf{F}_s^{-1}\mathbf{E}\mathbf{F}_s^{-T}\mathbf{F}_t^{T}\mathbf{q} \\ &= \mathbf{p} \cdot \chi_{s,t*}\mathbf{E}\mathbf{q} \end{aligned} \tag{3.2.35}$$

where

$$\chi_{s,t*}\mathbf{E} = \mathbf{F}_t\mathbf{F}_s^{-1}\mathbf{E}\mathbf{F}_s^{-T}\mathbf{F}_t^T \tag{3.2.36}$$

defines the push-forward of $\mathbf{E} : T^*\mathcal{B}_s \times T^*\mathcal{B}_s \to R$ into $\chi_{s,t*}\mathbf{E} : T^*\mathcal{B}_t \times T^*\mathcal{B}_t \to R$. Conversly, one can define the pull-back of $\mathbf{E} : T^*\mathcal{B}_t \times T^*\mathcal{B}_t \to R$ into $\chi^*_{s,t}\mathbf{E} : T^*\mathcal{B}_s \times T^*\mathcal{B}_s \to R$ as follows

$$\chi^*_{s,t}\mathbf{E} = \mathbf{F}_s\mathbf{F}_t^{-1}\mathbf{E}\mathbf{F}_t^{-T}\mathbf{F}_s^T \tag{3.2.37}$$

It is possible to calculate now the Lie derivative of $\mathbf{E}$ with respect to the flow defined by the pseudo-rigid motion

$$\begin{aligned}\overset{\diamond}{\mathbf{E}} &= \frac{d}{dt}\left(\mathbf{F}_s\mathbf{F}_t^{-1}\mathbf{E}\mathbf{F}_t^{-T}\mathbf{F}_s^T\right)\Big|_{t=s} \\ &= \left[\mathbf{F}_s\dot{\mathbf{F}}_t^{-1}\mathbf{E}\mathbf{F}_t^{-T}\mathbf{F}_s^T + \mathbf{F}_s\mathbf{F}_t^{-1}\dot{\mathbf{E}}\mathbf{F}_t^{-T}\mathbf{F}_s^T + \mathbf{F}_s\mathbf{F}_t^{-1}\mathbf{E}\dot{\mathbf{F}}_t^{-T}\mathbf{F}_s^T\right]_{t=s} \\ &= \dot{\mathbf{E}} - \mathbf{L}\mathbf{E} - \mathbf{E}\mathbf{L}^T\end{aligned} \tag{3.2.38}$$

where $\dot{\mathbf{F}}^{-1} = -\mathbf{F}^{-1}\dot{\mathbf{F}}\mathbf{F}^{-1}$ was utilised. The same formula can be worked out in components

$$\begin{aligned}\dot{\mathbf{E}} &= \frac{d}{dt}\left(E^{ij}\mathbf{d}_i \otimes \mathbf{d}_j\right) \\ &= \dot{E}^{ij}\mathbf{d}_i \otimes \mathbf{d}_j + E^{ij}\left(\mathbf{L}\mathbf{d}_i\right) \otimes \mathbf{d}_j + E^{ij}\mathbf{d}_i \otimes \left(\mathbf{L}\mathbf{d}_j\right) \\ &= \dot{E}^{ij}\mathbf{d}_i \otimes \mathbf{d}_j + E^{ij}\mathbf{L}\left(\mathbf{d}_i \otimes \mathbf{d}_j\right) + E^{ij}\left(\mathbf{d}_i \otimes \mathbf{d}_j\right)\mathbf{L}^T\end{aligned} \tag{3.2.39}$$

where a direct analogy to the codeforming derivative for vectors can be observed. That is $\overset{\diamond}{\mathbf{E}} = \dot{E}^{ij}\mathbf{d}_i \otimes \mathbf{d}_j$. Codeforming rates can be similarly computed for other types of tensors.

The last discussed rate is related to the linear map $\mathbf{H} : T^*\mathcal{B}_t \to T\mathcal{B}_t$. The action of $\mathbf{H}$ can be described as

$$\mathbf{p} = \mathbf{H}\mathbf{n} \tag{3.2.40}$$

where $\mathbf{p} \in T\mathcal{B}_t$ and $\mathbf{n} \in T^*\mathcal{B}_t$. One is then interested in computing the action of $\mathbf{H}$ on covectors $\mathbf{k} \in T^*\mathcal{B}_s$. Any $\mathbf{k} \in T^*\mathcal{B}_s$ can be pushed forward into $\chi_{s,t*}\mathbf{k} \in T^*\mathcal{B}_t$ so that

$$\begin{aligned}\mathbf{q} &= \mathbf{H}\chi_{s,t*}\mathbf{k} \\ &= \mathbf{H}\mathbf{F}_t^{-T}\mathbf{F}_s^T\mathbf{k} \\ &= \chi^*_{s,t}\mathbf{H}\mathbf{k}\end{aligned} \tag{3.2.41}$$

where the pull-back of $\mathbf{H}$ is defined as

$$\chi^*_{s,t}\mathbf{H} = \mathbf{H}\mathbf{F}_t^{-T}\mathbf{F}_s^T \tag{3.2.42}$$

The codeforming derivative of $\mathbf{H}$ follows

$$\begin{aligned}\overset{\diamond}{\mathbf{H}} &= \frac{d}{dt}\left(\mathbf{H}\mathbf{F}_t^{-T}\mathbf{F}_s^T\right)\Big|_{t=s} \\ &= \dot{\mathbf{H}} - \mathbf{H}\mathbf{L}^T\end{aligned} \tag{3.2.43}$$

Vector and tensor kinematics

(1) Flow gradient

$$\mathbf{F}_{s,t} = \mathbf{F}(t)\,\mathbf{F}^{-1}(s) \Rightarrow \mathbf{F}_{0,t} = \mathbf{F},\ \ \mathbf{F}_{s,s} = \mathbf{I}$$

(2) Vectors

$$\mathbf{a} = \mathbf{F}_{s,t}\mathbf{b},\ \ \mathbf{b} = \mathbf{F}_{s,t}^{-1}\mathbf{a},\ \ \mathbf{a} \in T\mathcal{B}_t,\ \ \mathbf{b} \in T\mathcal{B}_s$$

$$\overset{\diamond}{\mathbf{a}} = \dot{\mathbf{a}} - \mathbf{L}\mathbf{a}$$

(3) Covectors

$$\mathbf{k} = \mathbf{F}_{s,t}^{-T}\mathbf{n},\ \ \mathbf{n} = \mathbf{F}_{s,t}^{T}\mathbf{k},\ \ \mathbf{n} \in T^*\mathcal{B}_s,\ \ \mathbf{k} \in T^*\mathcal{B}_t$$

$$\langle \mathbf{n}, \cdot \rangle : T\mathcal{B}_s \to R$$

$$\overset{\diamond}{\mathbf{n}} = \dot{\mathbf{n}} + \mathbf{L}^T\mathbf{n}$$

(4) Contravariant tensors

$$\mathbf{G} = \mathbf{F}_{s,t}\mathbf{E}\mathbf{F}_{s,t}^{T}$$

$$\mathbf{E} = \mathbf{F}_{s,t}^{-1}\mathbf{G}\mathbf{F}_{s,t}^{-T}$$

$$\mathbf{E} : T^*\mathcal{B}_s \times T^*\mathcal{B}_s \to R,\ \ \mathbf{G} : T^*\mathcal{B}_t \times T^*\mathcal{B}_t \to R$$

$$\overset{\diamond}{\mathbf{E}} = \dot{\mathbf{E}} - \mathbf{L}\mathbf{E} - \mathbf{E}\mathbf{L}^T$$

(5) Contravariant linear maps

$$\mathbf{G} = \mathbf{H}\mathbf{F}_{s,t}^{T}$$

$$\mathbf{H} = \mathbf{G}\mathbf{F}_{s,t}^{-T}$$

$$\mathbf{H} : T^*\mathcal{B}_s \to T\mathcal{B}_s,\ \ \mathbf{G} : T^*\mathcal{B}_t \to T\mathcal{B}_t$$

$$\overset{\diamond}{\mathbf{H}} = \dot{\mathbf{H}} - \mathbf{H}\mathbf{L}^T$$

3.2.2. Pseudo-rigid motion and convexity. The following trivial fact ensures correctness of some of the subsequently employed algorithms.

FACT 3.2.1. *Pseudo-rigid motion preserves convexity.*

PROOF. Let $\mathcal{B}_0$ be convex. Then $\lambda\mathbf{X} + (1-\lambda)\,\mathbf{Y} \in \mathcal{B}_0$ for all $\mathbf{X}, \mathbf{Y} \in \mathcal{B}_0$, $\lambda \in [0,1]$.

$$\lambda\left(\mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \bar{\mathbf{x}}\right) + (1-\lambda)\left(\mathbf{F}\left(\mathbf{Y} - \bar{\mathbf{X}}\right) + \bar{\mathbf{x}}\right) =$$
$$= \mathbf{F}\left(\lambda\mathbf{X} + (1-\lambda)\,\mathbf{Y} - \bar{\mathbf{X}}\right) + \bar{\mathbf{x}} \in \mathcal{B}$$

□

3.3. Matrix notation

Whenever it is not necessary to be specific about the underlying kinematics, it is convenient to adopt a unified notation. The generalised configuration of a body will be denoted by $\mathbf{q}$ while the velocity will be denoted by $\mathbf{u}$. Thus $\mathbf{q} \in \mathcal{Q}$ and $\mathbf{u} \in T_{\mathbf{q}}\mathcal{Q}$, where $\mathcal{Q}$ is a generalised configuration space. From several possible choices, the following one is made for the rigid body

$$\mathbf{q} = \begin{bmatrix} \Lambda_{11} \\ \Lambda_{21} \\ \dots \\ \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{bmatrix},\ \ \mathbf{u} = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \\ \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \\ \dot{\bar{x}}_3 \end{bmatrix} \tag{3.3.1}$$

This is due to consistency with the algorithmic developments of Chapter 5. It should be noted that in (3.3.1) $\dot{\mathbf{q}} \neq \mathbf{u}$, but rather $\dot{\mathbf{q}} = \left[\begin{array}{cc} \mathbf{\Lambda} & \mathbf{I} \end{array} \right] \mathbf{u}$. One the other hand, for the pseudo-rigid body, the most natural choice reads

$$\mathbf{q} = \begin{bmatrix} F_{11} \\ F_{12} \\ \cdots \\ \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \dot{F}_{11} \\ \dot{F}_{12} \\ \cdots \\ \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \\ \dot{\bar{x}}_3 \end{bmatrix} \tag{3.3.2}$$

where obviously $\dot{\mathbf{q}} = \mathbf{u}$. The row-wise ordering of $\mathbf{F}$ in $\mathbf{q}$ is related to the computational efficiency of some algebraic operations, to be commented on at a later point.

3.4. Literature

Starting with Euler in the third quarter of the eighteenth century, kinematics of rigid motion has been studied for over two hundred years now. A review paper by Dai [**52**] gives a good summary in that respect. A typical textbook exposition, like in Arnold [**12**], usually contains a brief statement of kinematics followed by an exhaustive discussion on dynamics. From the numerical point of view though, parametrisation of rigid rotations is rather important. In the literature on the integration of rigid motion, three major methods of updating the rotation operator can be named. *Cayley formula* is used for example in an old paper by Benson and Hallquist [**23**], as well as in one of the algorithms recently investigated by Nukala and Shelton [**171**]. *Rodrigues formula* is employed in the explicit scheme by Simo and Wong [**100**], also by Krysl and Endres [**163**], Krysl [**126, 128, 127**], Nukala and Shelton [**171**]. *Quaternion* based update is utilised in the implicit scheme by Simo and Wong [**100**], also by Park and Chiou [**110**], Omelyan [**162**], Shivarama [**189**], Johnson *et al.* [**187**]. On a somewhat more theoretical level, recent quaternion based developments include Kosenko [**120**] and Rico-Martinez and Gallardo-Alvarado [**179**]. On the other hand, the incremental rotation angle and the Rodrigues formula seem to be often exploited within the field of geometrically exact beam theories. Papers by Ibrahimbegović *et al.* [**99**], Crisfield and Jelenic [**144**], Ritto-Corrêa and Camotim [**145**], and the doctoral thesis by Mäkinen [**146**] provide a good reference here.

The pseudo-rigid body model was derived by Cohen and Muncaster [**46**] as a simplified counterpart of finite elastodynamics. Kinematically, it does not differ much from the point level description of the classical continuum. Thus, apart from the monograph [**46**], textbooks on continuum mechanics might be of use. For example, chapters on kinematics in Marsden and Hughes [**147**] and Belytschko *et al.* [**22**] seem to be complementary in terms of the balance between theory and practice. As discussed by Nordenholz and O'Reilly [**160, 161**], pseudo-rigid bodies are equivalent to Cosserat points. As shown by Solberg and Papadopoulos [**193**], pseudo-rigid bodies are also equivalent to constant strain finite elements. An extension of the pseudo-rigid body concept, admitting second order deformation effects, has been proposed by Papadopoulos [**167**].

CHAPTER 4

Dynamics

For a body $\mathcal{B}$, the conservation of mass and the balance of linear and angular momentum respectively read

$$\frac{d}{dt}\int_{\mathcal{B}} \rho dv = 0 \tag{4.0.1}$$

$$\frac{d}{dt}\int_{\mathcal{B}} \rho \dot{\mathbf{x}} dv = \int_{\partial\mathcal{B}} \mathbf{t} da + \int_{\mathcal{B}} \rho \mathbf{b} dv \tag{4.0.2}$$

$$\frac{d}{dt}\int_{\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}}) \times \rho \dot{\mathbf{x}} dv = \int_{\partial\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}}) \times \mathbf{t} da + \int_{\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}}) \times \rho \mathbf{b} dv \tag{4.0.3}$$

where t is time, ρ is the mass density, $\dot{\mathbf{x}}$ is the point velocity, $\mathbf{t}$ is the surface traction, $\mathbf{b}$ is the body force, and $\bar{\mathbf{x}}$ is a selected point. All of the mentioned quantities are spatial and so is the integration domain $\mathcal{B}$, being the current configuration of the body. One can work out a specific form of the above principles, by considering kinematic models presented in the previous chapter. This time it is more convenient to start with the pseudo-rigid case (Section 4.1), and eventually simplify the obtained equations in order to embrace the rigid body model (Section 4.2).

4.1. Pseudo-Rigid body

The scalar mass of a body is

$$m = \int_{\mathcal{B}} \rho dv \tag{4.1.1}$$

and the conservation of mass states

$$\dot{m} = 0 \tag{4.1.2}$$

The useful consequence of the conservation of mass is that

$$\rho J = \rho_0 \tag{4.1.3}$$

where $J = \det(\mathbf{F})$ is the Jacobian (with respect to the Cartesian coordinates $\{x^i\}$ and $\{X^i\}$), and ρ_0 is the referential mass density. This follows from the fact, that $\int_{\mathcal{B}} \rho dv = \int_{\mathcal{B}_0} \rho J dV = \int_{\mathcal{B}_0} \rho_0 dV$. One can now move the time derivative under the integral in the standard way

$$\frac{d}{dt}\int_{\mathcal{B}} a\rho dv = \frac{d}{dt}\int_{\mathcal{B}_0} a\rho_0 dV = \int_{\mathcal{B}_0} \frac{da}{dt}\rho_0 dV \tag{4.1.4}$$

The motion of the pseudo-rigid body is employed, in order to rewrite the linear momentum balance as

$$\ddot{\mathbf{F}} \int_{\mathcal{B}_0} \rho_0 \left(\mathbf{X} - \bar{\mathbf{X}}\right) dV + m\ddot{\bar{\mathbf{x}}} = \int_{\partial\mathcal{B}} \mathbf{t} da + \int_{\mathcal{B}} \rho \mathbf{b} dv \tag{4.1.5}$$

Clearly, it is advantageous to select $\bar{\mathbf{X}}$ so that

$$\int_{\mathcal{B}_0} \rho_0 \left(\mathbf{X} - \bar{\mathbf{X}}\right) dV = \mathbf{0} \tag{4.1.6}$$

From now on $\bar{\mathbf{X}}$ is considered to be the referential *mass centre* of the body. The linear momentum balance reads then

$$m\ddot{\bar{\mathbf{x}}} = \mathbf{f} \tag{4.1.7}$$

where

$$\mathbf{f} = \int_{\partial\mathcal{B}} \mathbf{t} da + \int_{\mathcal{B}} \rho \mathbf{b} dv \tag{4.1.8}$$

is the resultant force. The angular momentum conservation can be worked out as follows. First note, that

$$\begin{aligned}
\frac{d}{dt} \int_{\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \times \rho \dot{\mathbf{x}} dv &= \\
\frac{d}{dt} \int_{\mathcal{B}_0} \left[\mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right)\right] \times \left[\dot{\mathbf{F}}\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \dot{\bar{\mathbf{x}}}\right] \rho_0 dV &= \\
\int_{\mathcal{B}_0} \left[\dot{\mathbf{F}}\left(\mathbf{X} - \bar{\mathbf{X}}\right)\right] \times \left[\dot{\mathbf{F}}\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \dot{\bar{\mathbf{x}}}\right] \rho_0 dV &+ \\
\int_{\mathcal{B}_0} \left[\mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right)\right] \times \left[\ddot{\mathbf{F}}\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \ddot{\bar{\mathbf{x}}}\right] \rho_0 dV &= \\
\int_{\mathcal{B}_0} \left[\dot{\mathbf{F}}\left(\mathbf{X} - \bar{\mathbf{X}}\right)\right] \times \dot{\bar{\mathbf{x}}} \rho_0 dV + \int_{\mathcal{B}_0} \left[\mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right)\right] \times \ddot{\bar{\mathbf{x}}} \rho_0 dV &+ \\
\int_{\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \times \left[\ddot{\mathbf{F}}\mathbf{F}^{-1}\left(\mathbf{x} - \bar{\mathbf{x}}\right)\right] \rho dv &= \\
-\text{skew}\left[\dot{\bar{\mathbf{x}}}\right] \dot{\mathbf{F}} \int_{\mathcal{B}_0} \left(\mathbf{X} - \bar{\mathbf{X}}\right) \rho_0 dV - \text{skew}\left[\ddot{\bar{\mathbf{x}}}\right] \mathbf{F} \int_{\mathcal{B}_0} \left(\mathbf{X} - \bar{\mathbf{X}}\right) \rho_0 dV &+ \\
\int_{\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \times \left[\ddot{\mathbf{F}}\mathbf{F}^{-1}\left(\mathbf{x} - \bar{\mathbf{x}}\right)\right] \rho dv &= \\
\int_{\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \times \left[\ddot{\mathbf{F}}\mathbf{F}^{-1}\left(\mathbf{x} - \bar{\mathbf{x}}\right)\right] \rho dv &
\end{aligned} \tag{4.1.9}$$

where $\mathbf{a} \times \mathbf{a} = \mathbf{0}$ was used in the transition from line three to five, spatial homogeneity of $\mathbf{F}$ and the fact that $\mathbf{a} \times \mathbf{b} = -\text{skew}\left[\mathbf{b}\right] \mathbf{a}$ (skew $[\cdot]$ makes a skew symmetric operator of a vector) were utilised in the transition from line five to seven, and formula (4.1.6) was exploited in order to reach the last line. The angular momentum balance can now be phrased as

$$\int_{\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \times \left[\ddot{\mathbf{F}}\mathbf{F}^{-1}\left(\mathbf{x} - \bar{\mathbf{x}}\right)\right] \rho dv = \int_{\partial\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \times \mathbf{t} da + \int_{\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \times \rho \mathbf{b} dv \tag{4.1.10}$$

Let vecskew $[\cdot]$ make a 3-vector out of a 3×3 skew symmetric matrix. By noticing that $\mathbf{a} \times \mathbf{b} = \text{vecskew}\left[\mathbf{b} \otimes \mathbf{a} - \mathbf{a} \otimes \mathbf{b}\right]$, and the fact that $\mathbf{A} = \mathbf{B}$ implies $\mathbf{A} - \mathbf{A}^T = \mathbf{B} - \mathbf{B}^T$, one can rewrite the above as

$$\int_{\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}}) \otimes \left[\ddot{\mathbf{F}}\mathbf{F}^{-1}(\mathbf{x}-\bar{\mathbf{x}})\right] \rho dv = \int_{\partial\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}}) \otimes \mathbf{t} da + \int_{\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}}) \otimes \rho\mathbf{b} dv \tag{4.1.11}$$

or equivalently

$$\int_{\mathcal{B}} \left[\ddot{\mathbf{F}}\mathbf{F}^{-1}(\mathbf{x}-\bar{\mathbf{x}})\right] \otimes (\mathbf{x}-\bar{\mathbf{x}}) \rho dv = \int_{\partial\mathcal{B}} \mathbf{t} \otimes (\mathbf{x}-\bar{\mathbf{x}}) da + \int_{\mathcal{B}} \rho\mathbf{b} \otimes (\mathbf{x}-\bar{\mathbf{x}}) dv \tag{4.1.12}$$

This can be further simplified, by making use of the deforming tensor $\mathbf{L} = \dot{\mathbf{F}}\mathbf{F}^{-1}$

$$\dot{\mathbf{L}} = \ddot{\mathbf{F}}\mathbf{F}^{-1} + \dot{\mathbf{F}}\dot{\mathbf{F}}^{-1}, \;\; \dot{\mathbf{F}}^{-1} = -\mathbf{F}^{-1}\dot{\mathbf{F}}\mathbf{F}^{-1}, \;\; \ddot{\mathbf{F}}\mathbf{F}^{-1} = \dot{\mathbf{L}} + \mathbf{L}^2 \tag{4.1.13}$$

and the following relations for the tensor product

$$\mathbf{a} \otimes (\mathbf{L}\mathbf{a}) = (\mathbf{a} \otimes \mathbf{a})\mathbf{L}^T, \;\; (\mathbf{L}\mathbf{a}) \otimes \mathbf{a} = \mathbf{L}(\mathbf{a} \otimes \mathbf{a}) \tag{4.1.14}$$

and

$$\begin{aligned} \frac{d}{dt}(\mathbf{L}\mathbf{a} \otimes \mathbf{a}) &= \dot{\mathbf{L}}\mathbf{a} \otimes \mathbf{a} + \mathbf{L}\dot{\mathbf{a}} \otimes \mathbf{a} + \mathbf{L}\mathbf{a} \otimes \dot{\mathbf{a}} \\ &= \dot{\mathbf{L}}\mathbf{a} \otimes \mathbf{a} + \mathbf{L}\mathbf{L}\mathbf{a} \otimes \mathbf{a} + \mathbf{L}\mathbf{a} \otimes \mathbf{L}\mathbf{a} \\ &= \left(\dot{\mathbf{L}} + \mathbf{L}^2\right)\mathbf{a} \otimes \mathbf{a} + \mathbf{L}\mathbf{a} \otimes \mathbf{a}\mathbf{L}^T \end{aligned} \tag{4.1.15}$$

so that

$$\begin{aligned} \int_{\mathcal{B}} \left[\ddot{\mathbf{F}}\mathbf{F}^{-1}(\mathbf{x}-\bar{\mathbf{x}})\right] \otimes (\mathbf{x}-\bar{\mathbf{x}}) \rho dv &= \\ \left(\dot{\mathbf{L}} + \mathbf{L}^2\right) \int_{\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}}) \otimes (\mathbf{x}-\bar{\mathbf{x}}) \rho dv &= \\ \left(\dot{\mathbf{L}} + \mathbf{L}^2\right)\mathbf{E} &= \frac{d}{dt}(\mathbf{L}\mathbf{E}) - \mathbf{L}\mathbf{E}\mathbf{L}^T \end{aligned} \tag{4.1.16}$$

where

$$\mathbf{E} = \int_{\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}}) \otimes (\mathbf{x}-\bar{\mathbf{x}}) \rho dv \tag{4.1.17}$$

is the *spatial Euler tensor*. Finally, the tensor equation

$$\frac{d}{dt}(\mathbf{L}\mathbf{E}) - \mathbf{L}\mathbf{E}\mathbf{L}^T = \int_{\partial\mathcal{B}} \mathbf{t} \otimes (\mathbf{x}-\bar{\mathbf{x}}) da + \int_{\mathcal{B}} \rho\mathbf{b} \otimes (\mathbf{x}-\bar{\mathbf{x}}) dv \tag{4.1.18}$$

describes the balance of the angular momentum.

Formula (4.1.18) implicitly represents a set of nonlinear ordinary differential equations with respect to the components of the deformation gradient $\mathbf{F}$ and Euler tensor $\mathbf{E}$. It is incomplete though, in the sense that an integration of the above equation would allow a body to deform without a bound. Intuitively, such a bound comes from the internal forces, opposing any deformation. In our case, this opposition must be rather specific, so that the homogeneity of deformations is preserved. This issue has risen some controversy in the literature, see Steigmann [**196**] and Casey [**36, 37**]. Nevertheless, we shall not be concerned with this rather philosophical discourse, as it does not affect the practical utility of the pseudo-rigid model. In order to bridge (4.1.18) with the deformation induced forces, we need to recall that as a consequence of the Cauchy's theorem (cf. Marsden and Hughes [**147**, pp.

127-134]), there exists a second order contravariant *Cauchy stress* tensor σ, such that

$$\mathbf{t} = \sigma \mathbf{n} \tag{4.1.19}$$

where $\mathbf{n}$ is the unit outward normal to $\partial\mathcal{B}$. The following evaluation is now possible

$$\begin{aligned} \int_{\partial\mathcal{B}} \mathbf{t} \otimes (\mathbf{x} - \bar{\mathbf{x}})\, da &= \int_{\partial\mathcal{B}} \sigma\mathbf{n} \otimes (\mathbf{x} - \bar{\mathbf{x}})\, da \\ &= \int_{\mathcal{B}} \mathrm{div}\sigma \otimes (\mathbf{x} - \bar{\mathbf{x}})\, dv + \int_{\mathcal{B}} \sigma dv \end{aligned} \tag{4.1.20}$$

where the divergence theorem $\int_{\mathcal{B}} \mathrm{div}\mathbf{a} dv = \int_{\partial\mathcal{B}} \mathbf{a} \cdot \mathbf{n} da$ has been applied with $\mathbf{a} = \mathrm{row}_i [\sigma] (x_j - \bar{x}_j)$ for all i, j. In the next step, the local form of the linear momentum balance (4.0.2)

$$\rho\ddot{\mathbf{x}} = \rho\mathbf{b} + \mathrm{div}\sigma \tag{4.1.21}$$

and the *mean* Cauchy stress tensor defined as

$$\bar{\sigma} = \frac{1}{V} \int_{\mathcal{B}} \sigma dv \tag{4.1.22}$$

are plunged back into (4.1.20), so that

$$\int_{\mathcal{B}} \rho\ddot{\mathbf{x}} \otimes (\mathbf{x} - \bar{\mathbf{x}})\, dv + V\bar{\sigma} = \int_{\partial\mathcal{B}} \mathbf{t} \otimes (\mathbf{x} - \bar{\mathbf{x}})\, da + \int_{\mathcal{B}} \rho\mathbf{b} \otimes (\mathbf{x} - \bar{\mathbf{x}})\, dv \tag{4.1.23}$$

Equivalently, by (4.1.9) and (4.1.16) there holds

$$\frac{d}{dt} (\mathbf{LE}) - \mathbf{LEL}^T + V\bar{\sigma} = \int_{\partial\mathcal{B}} \mathbf{t} \otimes (\mathbf{x} - \bar{\mathbf{x}})\, da + \int_{\mathcal{B}} \rho\mathbf{b} \otimes (\mathbf{x} - \bar{\mathbf{x}})\, dv \tag{4.1.24}$$

where $V = \int_{\mathcal{B}} dv$ is the current volume. It should be noted, that the balance principle (4.1.24) implies the angular momentum conservation (4.0.3), provided that the Cauchy stress tensor σ is *symmetric*. The stress term $V\bar{\sigma}$ prevents an unbounded growth of deformation, although for this one needs to declare a physically plausible relationship between $\bar{\sigma}$ and $\mathbf{F}$. The relation $\bar{\sigma} = \bar{\sigma}(\mathbf{F})$ bears the name of a *constitutive equation*. Let us not specify this relation yet, but rather summarise the current derivations. This is done in the box below.

Spatial pseudo-rigid dynamics

(1) Mass conservation

$$\dot{m} = 0, \;\; \dot{\mathbf{E}} - \mathbf{LE} - \mathbf{EL}^T = \mathbf{0}$$

(2) Linear momentum balance

$$m\ddot{\bar{\mathbf{x}}} = \int_{\partial\mathcal{B}} \mathbf{t} da + \int_{\mathcal{B}} \rho\mathbf{b} dv$$

(3) Angular momentum balance

$$\frac{d}{dt} (\mathbf{LE}) - \mathbf{LEL}^T + V\bar{\sigma} = \int_{\partial\mathcal{B}} \mathbf{t} \otimes (\mathbf{x} - \bar{\mathbf{x}})\, da + \int_{\mathcal{B}} \rho\mathbf{b} \otimes (\mathbf{x} - \bar{\mathbf{x}})\, dv$$

$$\mathbf{L} = \dot{\mathbf{F}}\mathbf{F}^{-1}, \;\; \bar{\sigma} = \bar{\sigma}(\mathbf{F})$$

In the first point of the box, $\dot{\mathbf{E}} - \mathbf{LE} - \mathbf{EL}^T = \mathbf{0}$ has been added in an ad-hoc manner. This corresponds to the conservation of the spatial Euler tensor, in the sense that, as viewed from the point of view of a co-deforming frame, it should not change with time. Recall from the previous chapter, that $\overset{\diamond}{\mathbf{E}} = \dot{\mathbf{E}} - \mathbf{LE} - \mathbf{EL}^T$ is a suitable Lie derivative, if only $\mathbf{E}$ can be regarded as a contravariant object. It is so, because $\mathbf{n} \cdot \mathbf{Ek}$ can be interpreted as measuring the amount of matter away from a pair of planes $\mathbf{n}, \mathbf{k}$ passing through $\bar{\mathbf{x}}$. Similarly, if one defines a generalised angular momentum $\mathbf{H} = \mathbf{LE}$, and realises that this is a contravariant map assigning to each plane $\mathbf{k}$ (passing through $\bar{\mathbf{x}}$) the net linear momentum $\mathbf{Hk}$ orthogonal to the plane, the term $\frac{d}{dt}(\mathbf{LE}) - \mathbf{LEL}^T$ becomes $\overset{\diamond}{\mathbf{H}} = \dot{\mathbf{H}} - \mathbf{HL}^T$. This connects our derivation with that pursued by Cohen and Muncaster [**46**, pp. 23-31], where a more constructive approach was undertaken. Regardless of those subtleties, it is quite clear that the nonlinearities of the spatial equations render them quite useless for practical purposes. In fact, the punch line of pseudo-rigid dynamics is in the simplicity of its referential formulation.

4.1.1. Referential formulation. We wish to simplify $\frac{d}{dt}(\mathbf{LE}) - \mathbf{LEL}^T$. Let us first define the *referential Euler tensor* as

$$\mathbf{E}_0 = \int_{\mathcal{B}_0} \left(\mathbf{X} - \bar{\mathbf{X}}\right) \otimes \left(\mathbf{X} - \bar{\mathbf{X}}\right) \rho_0 dV \tag{4.1.25}$$

so that

$$\begin{aligned} \mathbf{E} &= \int_{\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \otimes (\mathbf{x} - \bar{\mathbf{x}}) \rho dv \\ &= \int_{\mathcal{B}_0} \mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right) \otimes \mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right) \rho_0 dV = \mathbf{F}\mathbf{E}_0\mathbf{F}^T \end{aligned} \tag{4.1.26}$$

We can now write

$$\mathbf{LEL}^T = \dot{\mathbf{F}}\mathbf{F}^{-1}\mathbf{E}\mathbf{F}^{-T}\dot{\mathbf{F}}^T = \dot{\mathbf{F}}\mathbf{E}_0\dot{\mathbf{F}}^T \tag{4.1.27}$$

$$\frac{d}{dt}(\mathbf{LE}) = \dot{\mathbf{L}}\mathbf{E} + \mathbf{L}\dot{\mathbf{E}} \tag{4.1.28}$$

$$\dot{\mathbf{L}} = \ddot{\mathbf{F}}\mathbf{F}^{-1} + \dot{\mathbf{F}}\dot{\mathbf{F}}^{-1} \tag{4.1.29}$$

$$\begin{aligned} \dot{\mathbf{E}} &= \frac{d}{dt}\int_{\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \otimes (\mathbf{x} - \bar{\mathbf{x}}) \rho dv \\ &= \frac{d}{dt}\int_{\mathcal{B}_0} \mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right) \otimes \mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right) \rho_0 dV \\ &= \int_{\mathcal{B}_0} \dot{\mathbf{F}}\left(\mathbf{X} - \bar{\mathbf{X}}\right) \otimes \mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right) \rho_0 dV \\ &+ \int_{\mathcal{B}_0} \mathbf{F}\left(\mathbf{X} - \bar{\mathbf{X}}\right) \otimes \dot{\mathbf{F}}\left(\mathbf{X} - \bar{\mathbf{X}}\right) \rho_0 dV \\ &= \dot{\mathbf{F}}\mathbf{E}_0\mathbf{F}^T + \mathbf{F}\mathbf{E}_0\dot{\mathbf{F}}^T \end{aligned} \tag{4.1.30}$$

so that

$$\begin{aligned}\frac{d}{dt}(\mathbf{LE}) - \mathbf{LEL}^T &= \left(\ddot{\mathbf{F}}\mathbf{F}^{-1} + \dot{\mathbf{F}}\dot{\mathbf{F}}^{-1}\right)\mathbf{F}\mathbf{E}_0\mathbf{F}^T + \dot{\mathbf{F}}\mathbf{F}^{-1}\left(\dot{\mathbf{F}}\mathbf{E}_0\mathbf{F}^T + \mathbf{F}\mathbf{E}_0\dot{\mathbf{F}}^T\right) - \dot{\mathbf{F}}\mathbf{E}_0\dot{\mathbf{F}}^T \\ &= \ddot{\mathbf{F}}\mathbf{E}_0\mathbf{F}^T + \dot{\mathbf{F}}\dot{\mathbf{F}}^{-1}\mathbf{F}\mathbf{E}_0\mathbf{F}^T + \dot{\mathbf{F}}\mathbf{F}^{-1}\dot{\mathbf{F}}\mathbf{E}_0\mathbf{F}^T + \dot{\mathbf{F}}\mathbf{E}_0\dot{\mathbf{F}}^T - \dot{\mathbf{F}}\mathbf{E}_0\dot{\mathbf{F}}^T \\ &= \ddot{\mathbf{F}}\mathbf{E}_0\mathbf{F}^T - \dot{\mathbf{F}}\mathbf{F}^{-1}\dot{\mathbf{F}}\mathbf{F}^{-1}\mathbf{F}\mathbf{E}_0\mathbf{F}^T + \dot{\mathbf{F}}\mathbf{F}^{-1}\dot{\mathbf{F}}\mathbf{E}_0\mathbf{F}^T \\ &= \ddot{\mathbf{F}}\mathbf{E}_0\mathbf{F}^T \qquad (4.1.31)\end{aligned}$$

where $\dot{\mathbf{F}}^{-1} = -\mathbf{F}^{-1}\dot{\mathbf{F}}\mathbf{F}^{-1}$ was utilised. This allows to rewrite the angular momentum balance as

$$\ddot{\mathbf{F}}\mathbf{E}_0\mathbf{F}^T + V\bar{\sigma} = \int_{\partial\mathcal{B}} \mathbf{t}\otimes(\mathbf{x}-\bar{\mathbf{x}})\,da + \int_{\mathcal{B}} \rho\mathbf{b}\otimes(\mathbf{x}-\bar{\mathbf{x}})\,dv \qquad (4.1.32)$$

and after right-multiplying by $\mathbf{F}^{-T}$ obtain

$$\ddot{\mathbf{F}}\mathbf{E}_0 + V\bar{\sigma}\mathbf{F}^{-T} = \int_{\partial\mathcal{B}} \mathbf{t}\otimes\mathbf{F}^{-1}(\mathbf{x}-\bar{\mathbf{x}})\,da + \int_{\mathcal{B}} \rho\mathbf{b}\otimes\mathbf{F}^{-1}(\mathbf{x}-\bar{\mathbf{x}})\,dv \qquad (4.1.33)$$

The term $V\bar{\sigma}\mathbf{F}^{-T}$ can be worked out as follows

$$V\bar{\sigma}\mathbf{F}^{-T} = \int_{\mathcal{B}} \sigma\mathbf{F}^{-T}dv = \int_{\mathcal{B}_0} \sigma\mathbf{F}^{-T}JdV = \int_{\mathcal{B}_0} \mathbf{P}dV \qquad (4.1.34)$$

where

$$\mathbf{P} = J\sigma\mathbf{F}^{-T} \qquad (4.1.35)$$

is the *first Piola-Kirchhoff stress tensor* [**147**, p. 135]. The average referential stress is defined as

$$\bar{\mathbf{P}} = \frac{1}{V}\int_{\mathcal{B}_0} \mathbf{P}dV \qquad (4.1.36)$$

which together with the surface element transformation

$$\mathbf{t} = \sigma\mathbf{n},\ \ \mathbf{N} = \mathbf{F}^T\mathbf{n},\ \ \mathbf{p} = \mathbf{P}\mathbf{N} = J\sigma\mathbf{n} = J\mathbf{t} \Rightarrow\ \ \mathbf{t}da = \mathbf{p}dA \qquad (4.1.37)$$

leads to the final form of the referential angular momentum balance

$$\ddot{\mathbf{F}}\mathbf{E}_0 + V\bar{\mathbf{P}} = \int_{\partial\mathcal{B}_0} \mathbf{p}\otimes\left(\mathbf{X}-\bar{\mathbf{X}}\right)dA + \int_{\mathcal{B}_0} \rho_o\mathbf{b}\otimes\left(\mathbf{X}-\bar{\mathbf{X}}\right)dV \qquad (4.1.38)$$

From the computational point of view, the above equations improves much upon the spatial formulation. Clearly, deformations of the pseudo rigid body can be conveniently integrated in the reference frame, as soon as a constitutive relation $\bar{\mathbf{P}} = \bar{\mathbf{P}}(\mathbf{F})$ is given. This is summarised in the box below.

Referential pseudo-rigid dynamics

$$\ddot{\mathbf{F}}\mathbf{E}_0 + V\bar{\mathbf{P}} = \int_{\partial\mathcal{B}_0} \mathbf{p}\otimes\left(\mathbf{X}-\bar{\mathbf{X}}\right)dA + \int_{\mathcal{B}_0} \rho_o\mathbf{b}\otimes\left(\mathbf{X}-\bar{\mathbf{X}}\right)dV$$

$$\bar{\mathbf{P}} = \bar{\mathbf{P}}(\mathbf{F})$$

4.1.2. Constitutive equation. Cohen and Muncaster [**46**, pp. 52-58] discuss a pseudo-rigid adaptation of the classical constitutive theory and examine several well established material models. In the current work a hyperelastic pseudo-rigid continuum is considered, admitting the strain energy function Ψ, such that

$$\bar{\mathbf{P}} = \partial_{\mathbf{F}} \Psi (\mathbf{F}) \tag{4.1.39}$$

$$\Psi = \frac{1}{4} \left[\mathbf{F}^T \mathbf{F} - \mathbf{I} \right] : \mathbf{C} : \left[\mathbf{F}^T \mathbf{F} - \mathbf{I} \right] \tag{4.1.40}$$

$$C_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu \left[\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk} \right] \tag{4.1.41}$$

where the Saint Venant - Kirchhoff material was adopted. In the above λ and μ are Lamé constants, while δ_{ij} is the Kronecker delta. The Lamé constants can be expressed in terms of the Young modulus E and the Poisson ratio ν as

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{4.1.42}$$

$$\mu = \frac{E}{2+2\nu} \tag{4.1.43}$$

4.2. Rigid body

The mass conservation and the linear momentum balance do not change for the rigid body case. Some work must be done however, in order to work out the balance of the angular momentum. It is convenient to start from equation (4.1.18) for pseudo-rigid bodies

$$\frac{d}{dt} (\mathbf{LE}) - \mathbf{LEL}^T = \int_{\partial \mathcal{B}} \mathbf{t} \otimes (\mathbf{x} - \bar{\mathbf{x}}) \, da + \int_{\mathcal{B}} \rho \mathbf{b} \otimes (\mathbf{x} - \bar{\mathbf{x}}) \, dv \tag{4.2.1}$$

The deforming tensor $\mathbf{L}$ assigns velocities to spatial vectors, $\mathbf{L} : T\mathcal{B} \to T\mathcal{B}$. As it was shown in Section 3.1 of the previous chapter, for rigid bodies the same role is played by the spatial angular velocity $\hat{\omega}$. In other words, assuming an orthogonal deformation gradient $\mathbf{F} = \mathbf{\Lambda}$, there follows

$$\mathbf{L} = \hat{\omega}|_{\mathbf{F}=\mathbf{\Lambda}} \tag{4.2.2}$$

so that

$$\frac{d}{dt} (\hat{\omega}\mathbf{E}) + \hat{\omega}\mathbf{E}\hat{\omega} = \int_{\partial \mathcal{B}} \mathbf{t} \otimes (\mathbf{x} - \bar{\mathbf{x}}) \, da + \int_{\mathcal{B}} \rho \mathbf{b} \otimes (\mathbf{x} - \bar{\mathbf{x}}) \, dv \tag{4.2.3}$$

Let us now take the skew part of the above and come back to the vector form of the equation. One can see that

$$\operatorname{skew} [\hat{\omega}\mathbf{E}] = \hat{\omega}\mathbf{E} + \mathbf{E}\hat{\omega} \tag{4.2.4}$$

$$\operatorname{skew} [\hat{\omega}\mathbf{E}\hat{\omega}] = \hat{\omega}\mathbf{E}\hat{\omega} - \hat{\omega}\mathbf{E}\hat{\omega} = \mathbf{0} \tag{4.2.5}$$

$$\operatorname{skew} [\mathbf{a} \otimes \mathbf{b}] = \mathbf{a} \otimes \mathbf{b} - \mathbf{b} \otimes \mathbf{a} \tag{4.2.6}$$

where skew $[\cdot]$ was now used with respect to matrices, and $\hat{\omega}^T = -\hat{\omega}$ was utilised. When retrieving a vector from (4.2.4), it can be noticed that

$$\text{vecskew}\left[\hat{\omega}\mathbf{E} + \mathbf{E}\hat{\omega}\right] = \left[\text{tr}\left(\mathbf{E}\right)\mathbf{I} - \mathbf{E}\right]\omega \tag{4.2.7}$$

By definition

$$\mathbf{j} = \text{tr}\left(\mathbf{E}\right)\mathbf{I} - \mathbf{E} \tag{4.2.8}$$

is the *spatial inertia* tensor ($\mathbf{I}$ is the 3×3 identity). By noticing further

$$\mathbf{b} \times \mathbf{a} = \text{vecskew}\left[\mathbf{a} \otimes \mathbf{b} - \mathbf{b} \otimes \mathbf{a}\right] \tag{4.2.9}$$

we can arrive at a vector form of the spatial angular momentum conservation for rigid bodies

$$\frac{d}{dt}\left(\mathbf{j}\omega\right) = \int_{\partial\mathcal{B}} \left(\mathbf{x} - \bar{\mathbf{x}}\right) \times \mathbf{t} da + \int_{\mathcal{B}} \left(\mathbf{x} - \bar{\mathbf{x}}\right) \times \rho\mathbf{b} dv \tag{4.2.10}$$

For the complete picture, it remains to expand the $\frac{d}{dt}\left(\mathbf{j}\omega\right)$ term and compute the referential form of the equation. Notice that

$$\begin{aligned} \dot{\mathbf{E}} &= \int_{\mathcal{B}} \hat{\omega}\left(\mathbf{x} - \bar{\mathbf{x}}\right) \otimes \left(\mathbf{x} - \bar{\mathbf{x}}\right) \rho dv + \int_{\mathcal{B}} \left(\mathbf{x} - \bar{\mathbf{x}}\right) \otimes \hat{\omega}\left(\mathbf{x} - \bar{\mathbf{x}}\right) \rho dv \\ &= \hat{\omega}\mathbf{E} - \mathbf{E}\hat{\omega} \end{aligned} \tag{4.2.11}$$

Now it is convenient to start again with

$$\frac{d}{dt}\left(\hat{\omega}\mathbf{E}\right) = \dot{\hat{\omega}}\mathbf{E} + \hat{\omega}\dot{\mathbf{E}} = \dot{\hat{\omega}}\mathbf{E} + \hat{\omega}\hat{\omega}\mathbf{E} - \hat{\omega}\mathbf{E}\hat{\omega} \tag{4.2.12}$$

and take the vector representation its skew part. First and third components have already been evaluated in (4.2.4) and (4.2.5). The remaining one can be computed as

$$\text{skew}\left[\hat{\omega}\hat{\omega}\mathbf{E}\right] = \hat{\omega}\hat{\omega}\mathbf{E} - \mathbf{E}\hat{\omega}\hat{\omega} \tag{4.2.13}$$

which happens to coincide with the vector form

$$\text{vecskew}\left[\hat{\omega}\hat{\omega}\mathbf{E} - \mathbf{E}\hat{\omega}\hat{\omega}\right] = \omega \times \mathbf{j}\omega \tag{4.2.14}$$

The local form of the spatial angular momentum balance reads now

$$\mathbf{j}\dot{\omega} + \omega \times \mathbf{j}\omega = \int_{\partial\mathcal{B}} \left(\mathbf{x} - \bar{\mathbf{x}}\right) \times \mathbf{t} da + \int_{\mathcal{B}} \left(\mathbf{x} - \bar{\mathbf{x}}\right) \times \rho\mathbf{b} dv \tag{4.2.15}$$

The referential form of the above equation follows by noticing that

$$\omega = \boldsymbol{\Lambda}\boldsymbol{\Omega} \Leftrightarrow \hat{\omega} = \boldsymbol{\Lambda}\hat{\boldsymbol{\Omega}}\boldsymbol{\Lambda}^T \tag{4.2.16}$$

and hence

$$\omega \times \mathbf{j}\omega = \hat{\omega}\mathbf{j}\omega = \boldsymbol{\Lambda}\hat{\boldsymbol{\Omega}}\boldsymbol{\Lambda}^T\mathbf{j}\boldsymbol{\Lambda}\boldsymbol{\Omega} \tag{4.2.17}$$

which together with

$$\dot{\omega} = \dot{\boldsymbol{\Lambda}}\boldsymbol{\Omega} + \boldsymbol{\Lambda}\dot{\boldsymbol{\Omega}} = \boldsymbol{\Lambda}\hat{\boldsymbol{\Omega}}\boldsymbol{\Omega} + \boldsymbol{\Lambda}\dot{\boldsymbol{\Omega}} = \boldsymbol{\Lambda}\dot{\boldsymbol{\Omega}} \tag{4.2.18}$$

allows to write

$$\mathbf{j}\boldsymbol{\Lambda}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Lambda}\hat{\boldsymbol{\Omega}}\boldsymbol{\Lambda}^T\mathbf{j}\boldsymbol{\Lambda}\boldsymbol{\Omega} = \int_{\partial\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}})\times\mathbf{t}da + \int_{\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}})\times\rho\mathbf{b}dv \tag{4.2.19}$$

and after left-multiplication by $\boldsymbol{\Lambda}^T$ becomes the desired referential form of the balance

$$\mathbf{J}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega}\times\mathbf{J}\boldsymbol{\Omega} = \boldsymbol{\Lambda}^T\left[\int_{\partial\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}})\times\mathbf{t}da + \int_{\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}})\times\rho\mathbf{b}dv\right] \tag{4.2.20}$$

where by definition

$$\mathbf{J} = \boldsymbol{\Lambda}^T\mathbf{j}\boldsymbol{\Lambda} \tag{4.2.21}$$

is the *referential inertia* tensor (also called the *body frame* inertia tensor). It is not difficult to verify that

$$\mathbf{J} = \mathrm{tr}\,(\mathbf{E}_0)\,\mathbf{I} - \mathbf{E}_0 \tag{4.2.22}$$

Box below summarises rigid body dynamics. Note, that conservation of the spatial Euler tensor is now automatic ($\overset{\diamond}{\mathbf{E}} = \dot{\mathbf{E}} - \hat{\omega}\mathbf{E} - \mathbf{E}\hat{\omega}^T = \hat{\omega}\mathbf{E} - \mathbf{E}\hat{\omega} - \hat{\omega}\mathbf{E} + \mathbf{E}\hat{\omega} = \mathbf{0}$) and hence it was not stated explicitly. This results from the rigidity, preventing any distortion of the co-deforming frame.

Rigid dynamics

(1) Mass conservation

$$\dot{m} = 0$$

(2) Linear momentum balance

$$m\ddot{\bar{\mathbf{x}}} = \int_{\partial\mathcal{B}} \mathbf{t}da + \int_{\mathcal{B}} \rho\mathbf{b}dv$$

(3) Angular momentum balance

$$\frac{d}{dt}(\mathbf{j}\omega) = \mathbf{j}\dot{\omega} + \omega\times\mathbf{j}\omega = \mathbf{m} \Leftrightarrow \mathbf{J}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega}\times\mathbf{J}\boldsymbol{\Omega} = \boldsymbol{\Lambda}^T\mathbf{m}$$

$$\mathbf{m} = \int_{\partial\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}})\times\mathbf{t}da + \int_{\mathcal{B}} (\mathbf{x}-\bar{\mathbf{x}})\times\rho\mathbf{b}dv$$

$$\omega = \boldsymbol{\Lambda}\boldsymbol{\Omega},\ \ \mathbf{j} = \boldsymbol{\Lambda}\mathbf{J}\boldsymbol{\Lambda}^T,\ \ \mathbf{J} = \mathrm{tr}\,(\mathbf{E}_0)\,\mathbf{I} - \mathbf{E}_0$$

4.3. Matrix notation

We adopt the following uniform matrix notation for the dynamics of rigid and pseudo-rigid bodies

$$\mathbf{M}\dot{\mathbf{u}} = \mathbf{f} \tag{4.3.1}$$

For the rigid body case the inertia operator reads

$$\mathbf{M} = \begin{bmatrix} \mathbf{J} & \\ & m\mathbf{I} \end{bmatrix} \tag{4.3.2}$$

and the generalised out of balance force is

$$\mathbf{f} = \begin{bmatrix} \mathbf{\Lambda}^T \int_{\partial\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \times \mathbf{t} da + \mathbf{\Lambda}^T \int_{\mathcal{B}} (\mathbf{x} - \bar{\mathbf{x}}) \times \rho \mathbf{b} dv - \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} \\ \int_{\partial\mathcal{B}} \mathbf{t} da + \int_{\mathcal{B}} \rho \mathbf{b} dv \end{bmatrix} \tag{4.3.3}$$

For the pseudo-rigid body case the inertia operator reads

$$\mathbf{M} = \begin{bmatrix} \mathbf{E}_0 & & & \\ & \mathbf{E}_0 & & \\ & & \mathbf{E}_0 & \\ & & & m\mathbf{I} \end{bmatrix} \tag{4.3.4}$$

and the generalised out of balance force is

$$\mathbf{f} = \begin{bmatrix} \int_{\partial\mathcal{B}_0} \mathbf{p} \otimes \left(\mathbf{X} - \bar{\mathbf{X}}\right) dA + \int_{\mathcal{B}_0} \rho_o \mathbf{b} \otimes \left(\mathbf{X} - \bar{\mathbf{X}}\right) dV - V\bar{\mathbf{P}} \\ \int_{\partial\mathcal{B}} \mathbf{t} da + \int_{\mathcal{B}} \rho \mathbf{b} dv \end{bmatrix} \tag{4.3.5}$$

It should be noted, that it is the row-wise composition of $\dot{\mathbf{F}}$ in $\mathbf{u}$ (cf. Section 3.3), which allows us to use the computationally convenient block-diagonal form of $\mathbf{M}$ for pseudo-rigid bodies. This results from the fact, that $\ddot{F}_{ij} E_{0jk}$ can be seen as three matrix-vector products, where the vectors are rows of $\ddot{\mathbf{F}}$, and the symmetry of $\mathbf{E}_0$ is utilised.

4.4. Literature

Rigid body dynamics is a classical subject and has been for example comprehensively discussed by Arnold [**12**]. The monograph by Cohen and Muncaster [**46**] provides the essential summary for the pseudo-rigid body case. Although pseudo-rigid bodies seem not to have enjoyed many practical applications, the simple nonlinear form of the governing equations made them specifically attractive for a theoretically grounded research. For example, Lewis and Simo [**135**] studied stability of rotating pseudo-rigid bodies, Cohen and Mac Sithigh formulate a pseudo-rigid impact model [**47**], and discuss the slip reversal problem for frictional impact [**45**], Nordenholz and O'Reilly [**160, 161**] discuss some aspects of motion and stability of Cosserat points, and point out the compatibility of their studies with the pseudo-rigid context. Casey [**36**] gives a Lagrangian formulation of the pseudo-rigid dynamics, and discusses imposition of the homogeneity of deformation as a *global constraint*. This gives rise to the latter discussion with Steigmann [**196, 37**]. Solberg and Papadopoulos [**193**] examine an energy conserving impact of a spherical pseudo-rigid body, and show that multiple impacts occur before rebounding. The chaotic behaviour of the pseudo-rigid impact hinted in [**193**] was further studied by Kanso and Papadopoulos [**113, 112**].

CHAPTER 5

Time stepping

Before deciding upon a preferred time integration scheme, it is useful to realise what our needs are. The general intention is to develop a framework dealing with constrained systems, with an emphasis on multi-body frictional contact problems. The employed kinematic models are quite simple, hence there is not much of the discrepancy between the eigenvalues related to the low and the high vibration modes. We intend to deal with non-smoothness such as shocks, and employ implicit solvers in order to deal with the constraint. Having said that, it seems relevant to look for:

- *A low order scheme.* Because of a specific manner of dealing with the non-smooth constraints, the accuracy of any time-stepping will be reduced to the first order, if such are present. Hence, there is not much point in aiming at high accuracy. We will be satisfied with a second order method, as it will at the same time facilitate an adequate treatment of smooth dynamics.

- *An explicit scheme.* On one hand, the lightness of an explicit scheme is preferred in order to balance out the expenses related to the implicit treatment of the constraints. On the other hand, the issue of stability might seem restrictive. For the dynamics of pseudo-rigid bodies this does not represent a significant compromise. Skipping the few pseudo-rigid vibration modes by employing a large time step and a Newton solver seems vain, as one attempts to extract the rotational motion by means of linearisation unaware of rotations. Why not resort to the rigid model instead? However, the issue of stability does not generally vanish for rigid bodies. In that respect, a new stable scheme is proposed.

Section 5.1 summarises the time integration method employed for the dynamics of pseudo-rigid bodies. This is followed by the exposition of a scheme suitable for integration of constrained rotational motion (Section 5.2). The quasi-static case is briefly treated in Section 5.3. A short literature review follows in Section 5.4.

5.1. Pseudo-rigid dynamics

The time integrator ought to fit well into the structure of the computational code. In the explicit analysis of constrained multi-body dynamics it is convenient to employ the following time stepping

$$\mathbf{q}^{t+\frac{h}{2}} = \mathbf{q}^t + \frac{h}{2}\mathbf{u}^t \tag{5.1.1}$$

$$\mathbf{u}^{t+h} = \mathbf{u}^t + \mathbf{M}^{-1}h\mathbf{f}^{t+\frac{h}{2}} + \mathbf{M}^{-1}\mathbf{H}^T\mathbf{R} \tag{5.1.2}$$

$$\mathbf{q}^{t+h} = \mathbf{q}^{t+\frac{h}{2}} + \frac{h}{2}\mathbf{u}^{t+h} \tag{5.1.3}$$

where $\mathbf{u}$ is the velocity, $\mathbf{q}$ is the configuration, $\mathbf{M}$ is the inertia operator, $\mathbf{f}$ represents the generalised out of balance force, $\mathbf{H}$ incorporates gradients of the constraints, and $\mathbf{R}$ stores the constraints reactions. The utility of the above formulae results from several elementary facts:

(1) Combination of the central difference scheme and the trapezoidal rule maintains good conservation properties (Section 5.1.3) and is second order accurate (Section 5.1.1). Conditional stability (Section 5.1.2) is the only compromise here.
(2) The mid-step configuration $\mathbf{q}^{t+\frac{h}{2}}$ can be utilised for both, calculation of the constraints gradients operator $\mathbf{H}$ and approximation of $\mathbf{f}^{t+\frac{h}{2}}$. In practice, this means that some of the constraints (e.g. contacts) will be discovered at the mid-step configuration. As will be exemplified later, this choice allows to retain the second order accuracy in the presence of smooth constraints.
(3) The momentum balance (5.1.2) can be employed to calculate the constraints reactions $\mathbf{R}$. In particular, the algebraic structure of equation (5.1.2) allows for a convenient reformulation, which will be the subject of discussion in Chapter 7.
(4) If a suitable kinematic formulation is used, the inverse of inertia $\mathbf{M}^{-1}$ is computed only once. Obviously, this is a desirable feature.

The above scheme is applicable if the velocity and the configuration belong to the same vector space. For pseudo-rigid bodies this was shown to be the case (Section 3.2). In fact, equations (5.1.1-5.1.3) are an explicit reformulation and simplification of the implicit scheme given by Simo and Tarnow [**190**]. The scheme (5.1.1-5.1.3) was mentioned by Moreau [**156**], when presenting a "primitive example" of the *sweeping process* (cf. Chapter 10). In a sense, this thesis is merely a variation on the subject of this example. The following three sections show, that the above scheme is identical with the central difference method.

5.1.1. Accuracy. Assume that $\bar{\mathbf{q}}$ and $\bar{\mathbf{u}}$ are the solution of the initial value problem

$$\mathbf{M}\dot{\mathbf{u}} = \mathbf{f}(\mathbf{q}, t) \tag{5.1.4}$$

$$\dot{\mathbf{q}} = \mathbf{u} \tag{5.1.5}$$

$$\mathbf{q}(0) = \mathbf{q}_0, \quad \mathbf{u}(0) = \mathbf{u}_0 \tag{5.1.6}$$

For a general polynomial function $f(x)$ it is easy to see that $f(x + h\dot{x}) = f(x) + h\dot{f}(x) + O(h^2)$, where $O(\cdot)$ denotes terms growing no faster αh^2, $\alpha > 0$. For example

$$\begin{aligned} \bar{\mathbf{P}}\left(\mathbf{F} + \frac{h}{2}\dot{\mathbf{F}}\right) &= \frac{1}{2}\left(\mathbf{F} + \frac{h}{2}\dot{\mathbf{F}}\right)\mathbf{C} : \left\{\left(\mathbf{F} + \frac{h}{2}\dot{\mathbf{F}}\right)^T\left(\mathbf{F} + \frac{h}{2}\dot{\mathbf{F}}\right) - \mathbf{I}\right\} \\ &= \bar{\mathbf{P}}(\mathbf{F}) + \frac{h}{2}\dot{\bar{\mathbf{P}}}(\mathbf{F}) + O(h^2) \end{aligned} \tag{5.1.7}$$

where $\bar{\mathbf{P}}$ is the first Piola-Kirchhoff stress computed for the Saint Venant - Kirchhoff material model ($\mathbf{F}$ is the deformation gradient). Hence, one can write

$$\mathbf{f}\left(\bar{\mathbf{q}}^t+\frac{h}{2}\bar{\mathbf{u}}^t,t+\frac{h}{2}\right)=\mathbf{f}\left(\bar{\mathbf{q}}^t,t\right)+\frac{h}{2}\dot{\mathbf{f}}\left(\bar{\mathbf{q}}^t,t\right)+O\left(h^2\right) \tag{5.1.8}$$

Also

$$\bar{\mathbf{u}}^{t+h}=\bar{\mathbf{u}}^t+h\dot{\bar{\mathbf{u}}}^t+\frac{h^2}{2}\ddot{\bar{\mathbf{u}}}+O\left(h^3\right) \tag{5.1.9}$$

$$\bar{\mathbf{q}}^{t+h}=\bar{\mathbf{q}}^t+h\dot{\bar{\mathbf{q}}}^t+\frac{h^2}{2}\ddot{\bar{\mathbf{q}}}+O\left(h^3\right) \tag{5.1.10}$$

One can now compute the residuals

$$\begin{aligned}\tau_1(h) &= \mathbf{M}\left(\bar{\mathbf{u}}^{t+h}-\bar{\mathbf{u}}^t\right)-h\mathbf{f}\left(\bar{\mathbf{q}}^t+\frac{h}{2}\bar{\mathbf{u}}^t,t+\frac{h}{2}\right)\\ &= h\left[\mathbf{M}\dot{\bar{\mathbf{u}}}^t-\mathbf{f}\left(\bar{\mathbf{q}}^t,t\right)\right]+\frac{h^2}{2}\left[\mathbf{M}\ddot{\bar{\mathbf{u}}}-\dot{\mathbf{f}}\left(\bar{\mathbf{q}}^t,t\right)\right]+O\left(h^3\right)\end{aligned} \tag{5.1.11}$$

$$\begin{aligned}\tau_2(h) &= \bar{\mathbf{q}}^{t+h}-\bar{\mathbf{q}}^t-h\frac{\bar{\mathbf{u}}^t+\bar{\mathbf{u}}^{t+h}}{2}\\ &= h\left[\dot{\bar{\mathbf{q}}}^t-\bar{\mathbf{u}}^t\right]+\frac{h^2}{2}\left[\ddot{\bar{\mathbf{q}}}-\dot{\bar{\mathbf{u}}}^{t+h}\right]+O\left(h^3\right)\end{aligned} \tag{5.1.12}$$

and by assuming a sufficiently regularity of $\bar{\mathbf{q}},\bar{\mathbf{u}},\mathbf{f}$ (terms in $[\cdot]$ vanish) conclude, that $\|\tau_1(h)\|=O\left(h^3\right)$ and $\|\tau_2(h)\|=O\left(h^3\right)$. The method is then of the second order.

5.1.2. Stability. Only the linearised case is considered, which accounts for the necessary but not for the sufficient stability condition (cf. Hughes [**98**, p. 135]). The aim is to show briefly that the linearised stability criterion is the same as for the central difference scheme. Consider the following linearisation of equation (5.1.4)

$$\mathbf{M}\delta\dot{\mathbf{u}}+\mathbf{K}\delta\mathbf{q}=\mathbf{O} \tag{5.1.13}$$

where

$$\mathbf{K}=-\partial\mathbf{f}/\partial\mathbf{q} \tag{5.1.14}$$

is the tangent stiffness operator, and δs denote linear variations of the respective arguments. Provided, that both $\mathbf{M}$ and $\mathbf{K}$ are symmetric and positive definite (semi- for $\mathbf{K}$), standard spectral decomposition related to the eigenproblem $(\mathbf{K}-\lambda\mathbf{M})\,\psi=\mathbf{0}$ can be applied. Assuming normalisation $\mathbf{\Psi}^T\mathbf{M}\mathbf{\Psi}=\mathbf{I}$, where $\mathbf{\Psi}$ is composed of column-wise eigenvectors ψ, equation (5.1.13) can be diagonalised into a number of scalar equations of form

$$\delta\dot{u}+\lambda\delta q=0 \tag{5.1.15}$$

Scheme (5.1.1-5.1.3) is now applied to the above equation. After some simple algebra, there follows

$$\begin{bmatrix}\delta u^{t+h}\\ \delta q^{t+h}\end{bmatrix}=\begin{bmatrix}\left(1-\frac{h^2}{2}\lambda\right) & -h\lambda\\ h\left(1-\frac{h^2}{4}\lambda\right) & \left(1-\frac{h}{2}^2\lambda\right)\end{bmatrix}\begin{bmatrix}\delta u^t\\ \delta q^t\end{bmatrix} \tag{5.1.16}$$

which can be rewritten as

$$\mathbf{y}^{t+h} = \mathbf{A}\mathbf{y}^t \tag{5.1.17}$$

The scheme is stable if

$$\|\mathbf{A}\| \leq 1 \tag{5.1.18}$$

where $\|\cdot\|$ is the natural linear operator norm, defined as the largest stretch of a unit vector, $\|\mathbf{A}\| = \sup_{\mathbf{y}} \|\mathbf{A}\mathbf{y}\| / \|\mathbf{y}\|$. If γ_i are the eigenvalues of $\mathbf{A}$, there holds

$$\rho(\mathbf{A}) = \max_i |\gamma_i| \leq \|\mathbf{A}\| \tag{5.1.19}$$

and hence the stability criterion can be replaced by $\rho(\mathbf{A}) \leq 1$, where ρ is called the *spectral radius* of $\mathbf{A}$. For a 2×2 matrix, the eigenvalues read

$$\gamma = \frac{1}{2}\left(\operatorname{tr}(\mathbf{A}) \pm \sqrt{\operatorname{tr}^2(\mathbf{A}) - 4\det(\mathbf{A})}\right) \tag{5.1.20}$$

and since in our case $\operatorname{tr}(\mathbf{A}) = 2 - h^2\lambda$ and $\det(\mathbf{A}) = 1$, there follows

$$\gamma = 1 - \frac{h^2\lambda \pm h\sqrt{h^2\lambda^2 - 4\lambda}}{2} \tag{5.1.21}$$

and consequently $|\gamma| \leq 1$ reads

$$-1 \leq 1 - \frac{h^2\lambda \pm h\sqrt{h^2\lambda^2 - 4\lambda}}{2} \leq 1 \tag{5.1.22}$$

While the right inequality is satisfied for any $h, \lambda \geq 0$, the left one leads to the constraint on the time step

$$h \leq \frac{2}{\sqrt{\lambda}} \tag{5.1.23}$$

which is the same as for the central difference scheme [**98**, p. 94].

5.1.3. Conservation. The following discussion is largely based upon Simo and Tarnow [**190**]. Let us define the generalised momentum $\mathbf{p} = \mathbf{M}\mathbf{u}$ and rewrite (5.1.4-5.1.6) as

$$\dot{\mathbf{z}} = \mathbf{J}\nabla H \tag{5.1.24}$$

$$\mathbf{z}(0) = \mathbf{z}_0 \tag{5.1.25}$$

where

$$\mathbf{z} = \begin{bmatrix} \mathbf{q} \\ \mathbf{p} \end{bmatrix}, \quad \mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} \tag{5.1.26}$$

and

$$H = E_k + E_p \tag{5.1.27}$$

is the Hamiltonian of the dynamical system. By assumption we are dealing with an autonomous and conservative case, that is the out of balance force in (5.1.4)

reads $\mathbf{f}(\mathbf{q}) = -\partial E_p/\partial \mathbf{q}$, where E_p is the potential energy. The kinetic energy is the quadratic form $E_k = \frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}$. Hence, $\nabla H = \left[-\mathbf{f}, \mathbf{M}^{-1}\mathbf{p}\right]^T$. It is not difficult to notice, that the Hamiltonian remains constant along the integral curves of equation (5.1.24). Namely

$$\langle \nabla H, \dot{\mathbf{z}} \rangle = \langle \nabla H, \mathbf{J}\nabla H \rangle = \frac{\partial H}{\partial \mathbf{q}}\frac{\partial H}{\partial \mathbf{p}} - \frac{\partial H}{\partial \mathbf{p}}\frac{\partial H}{\partial \mathbf{q}} = 0 \tag{5.1.28}$$

In other words, solutions of (5.1.24-5.1.25) are the level curves of the total energy function and thus, the energy is conserved along the flow defined by the vector field $\mathbf{J}\nabla H$. It can be also noticed that (5.1.28) holds, because $w(\mathbf{a}, \mathbf{b}) = \langle \mathbf{a}, \mathbf{J}\mathbf{b} \rangle$ is an anti-symmetric bilinear form (also called the *symplectic* two-form), and thus $w(\mathbf{a}, \mathbf{a}) = 0$. Let M be the configuration space of all $\mathbf{z}$. One distinguishes a class of *symplectic transformations* $\mathbf{G}_t : M \to M$, that preserve w in the sense that

$$\frac{d}{dt} w\left(D\mathbf{G}_t(\mathbf{z})\,\delta\mathbf{z}_1, D\mathbf{G}_t(\mathbf{z})\,\delta\mathbf{z}_2\right) = 0 \tag{5.1.29}$$

for all $\delta\mathbf{z}_1, \delta\mathbf{z}_2 \in TM_{\mathbf{z}}$, where $D\mathbf{G}_t(\mathbf{z}) : TM_{\mathbf{z}} \to TM_{\mathbf{G}_t(\mathbf{z})}$ is the gradient of $\mathbf{G}_t$. Note, that the above condition means, that $\mathbf{G}_t$ moves points of M along some curves in such a way, that the pull-back of w defined at $TM_{\mathbf{G}_t(\mathbf{z})}$ is the same as w defined at $TM_{\mathbf{z}}$. That is

$$D\mathbf{G}_t(\mathbf{z})^T\,\mathbf{J}D\mathbf{G}_t(\mathbf{z}) = \mathbf{J} \tag{5.1.30}$$

In consequence, under the change of coordinates induced by $\mathbf{G}_t$ the Hamiltonian system (5.1.24-5.1.25) looks just the same. In the theory of Hamiltonian systems such changes of coordinates are called *canonical transformations*. The phase flow defined by equation (5.1.24) is composed of canonical transformations (cf. Arnold [**12**, p. 190]). When integrating the dynamical problem numerically, one advances the solution from $\mathbf{z}^t$ to $\mathbf{z}^{t+h}$ by finding roots of some general nonlinear map $\mathbf{G}\left(\mathbf{z}^{t+h}, \mathbf{z}^t\right) = \mathbf{0}$. If $\mathbf{G}$ is symplectic, one hopes to obtain an approximation of the integral curve, close to the level curve of the Hamiltonian. Thus, in the numerical sense, symplectic integrators are energy conserving. Technically, the symplecticity of $\mathbf{G}$ can be verified on the basis of linearisation $\delta\mathbf{z}^{t+h} = \mathbf{A}\delta\mathbf{z}^t$, where the *linearised amplification matrix* $\mathbf{A}$ reads

$$\mathbf{A} = \mathbf{A}_1^{-1}\mathbf{A}_2 \tag{5.1.31}$$

and

$$\mathbf{A}_1 = \frac{\partial \mathbf{G}\left(\mathbf{z}^{t+h}, \mathbf{z}^t\right)}{\partial \mathbf{z}^{t+h}}, \quad \mathbf{A}_2 = -\frac{\partial \mathbf{G}\left(\mathbf{z}^{t+h}, \mathbf{z}^t\right)}{\partial \mathbf{z}^t} \tag{5.1.32}$$

By analogy with (5.1.30), there needs to hold $\mathbf{A}^T\mathbf{J}\mathbf{A} = \mathbf{J}$ for the amplification matrix to be symplectic (and so for $\mathbf{G}$). Note, that the condition implies that $\det(\mathbf{A}) = 1$, and thus the spectral radius $\rho(\mathbf{A}) = 1$ (which was the case in the previous section). At this point it is convenient to notice, that $\mathbf{J} = -\mathbf{J}^T = -\mathbf{J}^{-1}$. Now, the condition $\mathbf{A}^T\mathbf{J}\mathbf{A} = \mathbf{J}$ can be spelt out as $\mathbf{A}_2^T\mathbf{A}_1^{-T}\mathbf{J}\mathbf{A}_1^{-1}\mathbf{A}_2 = \mathbf{J}$, further transformed into $\mathbf{A}_1^{-T}\mathbf{J}\mathbf{A}_1^{-1} = \mathbf{A}_2^{-T}\mathbf{J}\mathbf{A}_2^{-1}$ and inverted, resulting in

$$\mathbf{A}_1\mathbf{J}\mathbf{A}_1^T - \mathbf{A}_2\mathbf{J}\mathbf{A}_2^T = \mathbf{O} \tag{5.1.33}$$

We are finally in the position to verify symplecticity of the scheme (5.1.1-5.1.3). In our case, the operator $\mathbf{G}$ reads

$$\mathbf{G}\left(\mathbf{z}^{t+h},\mathbf{z}^{t}\right)=\mathbf{z}^{t+h}-\mathbf{z}^{t}-h\mathbf{J}\left[\begin{array}{c}\mathbf{f}\left(\mathbf{q}^{t}+\frac{h}{2}\mathbf{M}^{-1}\mathbf{p}^{t}\right)\\ \mathbf{M}^{-1}\left(\mathbf{p}^{t}+\mathbf{p}^{t+h}\right)/2\end{array}\right] \tag{5.1.34}$$

and hence

$$\mathbf{A}_{1}=\mathbf{I}-h\left[\begin{array}{cc}\mathbf{0} & \mathbf{I}\\ -\mathbf{I} & \mathbf{0}\end{array}\right]\left[\begin{array}{cc}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \frac{1}{2}\mathbf{M}^{-1}\end{array}\right]=\left[\begin{array}{cc}\mathbf{I} & -\frac{h}{2}\mathbf{M}^{-1}\\ \mathbf{0} & \mathbf{I}\end{array}\right] \tag{5.1.35}$$

$$\mathbf{A}_{2}=\mathbf{I}+h\left[\begin{array}{cc}\mathbf{0} & \mathbf{I}\\ -\mathbf{I} & \mathbf{0}\end{array}\right]\left[\begin{array}{cc}\mathbf{K} & \frac{h}{2}\mathbf{K}\mathbf{M}^{-1}\\ \mathbf{0} & \frac{1}{2}\mathbf{M}^{-1}\end{array}\right]=\left[\begin{array}{cc}\mathbf{I} & \frac{h}{2}\mathbf{M}^{-1}\\ -h\mathbf{K} & \mathbf{I}-\frac{h^{2}}{2}\mathbf{K}\mathbf{M}^{-1}\end{array}\right] \tag{5.1.36}$$

where $\mathbf{K}=\partial\mathbf{f}\left(\mathbf{q}\right)/\partial\mathbf{q}$. After some algebra there follows, that the two triple products in (5.1.33) read

$$\mathbf{A}_{1}\mathbf{J}\mathbf{A}_{1}^{T}=\mathbf{A}_{2}\mathbf{J}\mathbf{A}_{2}^{T}=\mathbf{J} \tag{5.1.37}$$

and thus the time stepping (5.1.1-5.1.3) is symplectic.

5.2. Rigid dynamics

As far as the linear motion is concerned, the discussion of the previous section applies. The rotational motion is solely of interest here. In principle, the objective is to devise a time integrator, preserving the structure and the qualities of scheme (5.1.1-5.1.3). In the pursuit of this goal it will be necessary to abuse slightly the notion of *geometrical consistency*, although the resulting scheme will have the qualities of modest computational cost, second order accuracy, and stability. Two versions of the new scheme are considered. The fully explicit one requires less computational effort, although it does experience a negative energy drift. The semi-explicit version does not drift, and it retains explicitness in the evaluation of the external force. Nonetheless, solution of a local implicit problem is necessary in order to update the configuration.

After some preliminary remarks in Section 5.2.1, the proposed scheme is specified in Section 5.2.2. Some comments about the conservation and stability properties are given in Section 5.2.3. A single illustrative example is given in Section 5.2.4. This is followed by a brief discussion on efficiency (Section 5.2.5).

5.2.1. Preliminaries. We recall, that the orthogonal rotation operator $\mathbf{\Lambda}\left(t\right)$ belongs to a curved space, the special orthogonal group $SO\left(3\right)$. It is updated in the multiplicative manner

$$\mathbf{\Lambda}\left(t+h\right)=\mathbf{\Lambda}\left(t\right)\exp\left[\mathbf{\Psi}\left(h\right)\right] \tag{5.2.1}$$

where $\mathbf{\Psi}\left(h\right)$ is the incremental rotation vector, and $\exp\left[\cdot\right]$ is the exponential map defined by the Rodrigues formula

$$\exp\left[\mathbf{\Psi}\right]=\mathbf{I}+\frac{\sin\left\|\mathbf{\Psi}\right\|}{\left\|\mathbf{\Psi}\right\|}\hat{\mathbf{\Psi}}+\frac{1-\cos\left\|\mathbf{\Psi}\right\|}{\left\|\mathbf{\Psi}\right\|^{2}}\hat{\mathbf{\Psi}}^{2} \tag{5.2.2}$$

Above, $\mathbf{I}$ is the 3×3 identity operator, $\hat{\mathbf{\Psi}}$ creates the skew symmetric matrix out of a 3-vector $\mathbf{\Psi}$, and $\left\|\cdot\right\|$ stands for the Euclidean norm. As was already discussed in Section 3.1, the increment of rotation $\left\|\mathbf{\Psi}\right\|$ should be smaller then 2π in order to avoid the singularity of the exponential map. In practice, and specifically for the constrained systems, this is a rather realistic assumption.

In the view of the update formula (5.2.1), the finite rotation vector $\mathbf{\Psi}$ can be perceived as belonging to the tangent space $T_{R(t)}SO\left(3\right)$. Operations such as vector

addition $\boldsymbol{\Theta}_1 + \boldsymbol{\Theta}_2$ make sense only if both vectors belong to the same tangent space $\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2 \in T_{R(t)}SO(3)$ *(geometrical consistency)*. When $\boldsymbol{\Theta}_1 \in T_{R(t)}SO(3)$ and $\boldsymbol{\Theta}_2 \in T_{R(t+h)}SO(3)$ the differential of the exponential map is employed in order to shift a selected vector from its own tangent space into the tangent space of the other vector. An example is

$$\left(\mathbf{T}^T \boldsymbol{\Theta}_1\right) + \boldsymbol{\Theta}_2 \tag{5.2.3}$$

where

$$\mathbf{T} = \mathbf{I} + \frac{1 - \cos\|\boldsymbol{\Psi}\|}{\|\boldsymbol{\Psi}\|^2}\hat{\boldsymbol{\Psi}} + \frac{\|\boldsymbol{\Psi}\| - \sin\|\boldsymbol{\Psi}\|}{\|\boldsymbol{\Psi}\|^3}\hat{\boldsymbol{\Psi}}^2 \tag{5.2.4}$$

was already defined as (3.1.25) in Section 3.1. As $\hat{\boldsymbol{\Psi}}\boldsymbol{\Psi} = \boldsymbol{\Psi} \times \boldsymbol{\Psi} = \mathbf{0}$, there follows that $\mathbf{T}^T\boldsymbol{\Psi} = \boldsymbol{\Psi}$, which represents a useful fact.

The balance of the angular momentum, expressed in the body-frame, reads

$$\mathbf{J}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times \mathbf{J}\boldsymbol{\Omega} = \boldsymbol{\Lambda}^T\mathbf{t} \tag{5.2.5}$$

where $\mathbf{J}$ is the constant referential inertia tensor, $\boldsymbol{\Omega}$ is the referential angular velocity, and $\mathbf{t}$ is the spatial torque. It is noteworthy that $\boldsymbol{\Omega}(t) \in T_{\mathbf{R}(t)}SO(3)$, so that an extrapolation $\boldsymbol{\Psi}(h) = h\boldsymbol{\Omega} + \frac{h^2}{2}\dot{\boldsymbol{\Omega}}$ makes sense.

Another form of the balance of the angular momentum follows from the spatial formula

$$\frac{d}{dt}(\mathbf{j}\omega) = \mathbf{t} \tag{5.2.6}$$

where $\mathbf{j}$ is the time-dependent spatial inertia tensor ($\mathbf{j} = \boldsymbol{\Lambda}\mathbf{J}\boldsymbol{\Lambda}^T$), and ω is the spatial angular velocity ($\omega = \boldsymbol{\Lambda}\boldsymbol{\Omega}$). The above expression can be integrated over the time interval $[t, t+h]$

$$\begin{aligned}
\mathbf{j}\omega|_t^{t+h} &= \mathbf{j}(t+h)\,\omega(t+h) - \mathbf{j}(t)\,\omega(t) \\
&= \boldsymbol{\Lambda}(t+h)\,\mathbf{J}\boldsymbol{\Lambda}^T(t+h)\,\omega(t+h) - \boldsymbol{\Lambda}(t)\,\mathbf{J}\boldsymbol{\Lambda}^T(t)\,\omega(t) \\
&= \boldsymbol{\Lambda}(t)\exp[\boldsymbol{\Psi}(h)]\,\mathbf{J}\boldsymbol{\Omega}^{t+h} - \boldsymbol{\Lambda}(t)\,\mathbf{J}\boldsymbol{\Omega}^t \\
&= \int_t^{t+h} \mathbf{t}dt
\end{aligned} \tag{5.2.7}$$

resulting in

$$\boldsymbol{\Omega}(t+h) = \mathbf{J}^{-1}\exp[-\boldsymbol{\Psi}(h)]\left[\mathbf{J}\boldsymbol{\Omega}(t) + \boldsymbol{\Lambda}^T(t)\int_t^{t+h}\mathbf{t}dt\right] \tag{5.2.8}$$

Discretisations of the above formula give rise to the variety of well-behaved time stepping methods (e.g. Krysl [**126**]). Nevertheless, an implicit dependence of the incremental rotation vector $\boldsymbol{\Psi}$ on the external torque $\mathbf{t}$ precludes a direct algorithmic analogy with (5.1.2).

5.2.2. Scheme. The proposed scheme reads

$$\boldsymbol{\Lambda}^{t+\frac{h}{2}} = \boldsymbol{\Lambda}^t \exp\left[\frac{h}{2}\boldsymbol{\Omega}^t\right] \tag{5.2.9}$$

$$\mathbf{T}^{t+\frac{h}{2}} = \left(\boldsymbol{\Lambda}^{t+\frac{h}{2}}\right)^T \mathbf{t}^{t+\frac{h}{2}} \tag{5.2.10}$$

$$\boldsymbol{\Omega}^{t+\frac{h}{2}} = \mathbf{J}^{-1}\left[\exp\left[-\frac{h}{2}\boldsymbol{\Omega}^{t}\right]\mathbf{J}\boldsymbol{\Omega}^{t} + \frac{h}{2}\mathbf{T}^{t+\frac{h}{2}}\right] \tag{5.2.11}$$

$$\boldsymbol{\Omega}_{1}^{t+h} = \boldsymbol{\Omega}^{t} + \mathbf{J}^{-1}h\left[\mathbf{T}^{t+\frac{h}{2}} - \boldsymbol{\Omega}^{t+\frac{h}{2}} \times \mathbf{J}\boldsymbol{\Omega}^{t+\frac{h}{2}}\right] \tag{5.2.12}$$

If explicit

$$\boldsymbol{\Lambda}^{t+h} = \boldsymbol{\Lambda}^{t+\frac{h}{2}}\exp\left[\frac{h}{2}\boldsymbol{\Omega}_{1}^{t+h}\right] \tag{5.2.13}$$

$$\boldsymbol{\Omega}_{2}^{t+h} = \mathbf{J}^{-1}\exp\left[-\frac{h}{2}\boldsymbol{\Omega}_{1}^{t+h}\right]\left[\exp\left[-\frac{h}{2}\boldsymbol{\Omega}^{t}\right]\mathbf{J}\boldsymbol{\Omega}^{t} + h\mathbf{T}^{t+\frac{h}{2}}\right] \tag{5.2.14}$$

otherwise

$$\textbf{solve}\ \left(\exp\left[\frac{h}{2}\boldsymbol{\Omega}_{3}^{t+h}\right]\mathbf{J}\boldsymbol{\Omega}_{3}^{t+h} = \exp\left[-\frac{h}{2}\boldsymbol{\Omega}^{t}\right]\mathbf{J}\boldsymbol{\Omega}^{t} + h\mathbf{T}^{t+\frac{h}{2}}\right) \tag{5.2.15}$$

$$\boldsymbol{\Lambda}^{t+h} = \boldsymbol{\Lambda}^{t+\frac{h}{2}}\exp\left[\frac{h}{2}\boldsymbol{\Omega}_{3}^{t+h}\right] \tag{5.2.16}$$

In the first formula (5.2.9) the mid-step rotation $\boldsymbol{\Lambda}^{t+\frac{h}{2}}$ is extrapolated with the forward Euler scheme. It is then used to compute the referential torque components in (5.2.10). In equation (5.2.11) the idea of LIEMID[E1] algorithm by Krysl [**126**] is borrowed in order to approximate the mid-step angular velocity $\boldsymbol{\Omega}^{t+\frac{h}{2}}$. Formula (5.2.8) is employed, where the spatial torque integral is approximated by

$$\int_{t}^{t+\frac{h}{2}} \mathbf{t}dt \simeq \frac{h}{2}\mathbf{t}^{t+\frac{h}{2}} \tag{5.2.17}$$

This allows to compute the external force only once and reuse it at a later stage. The central difference scheme is applied to the referential angular momentum balance in formula (5.2.12). This step is somewhat naive, but we need it in order to preserve the algebraic structure of formula (5.1.2). This is also the source of the *geometrical inconsistency*. Due to the collinearity of the incremental rotation vector and the initial angular velocity there holds

$$\mathbf{T}^{T}\left[\frac{h}{2}\boldsymbol{\Omega}^{t}\right]\boldsymbol{\Omega}^{t} = \boldsymbol{\Omega}^{t} \tag{5.2.18}$$

so that the right hand side of (5.2.12) resides in the tangent space $T_{\boldsymbol{\Lambda}(t+h/2)}SO(3)$. The left hand side, however, belongs to $T_{\boldsymbol{\Lambda}(t+h)}SO(3)$. Thus, the equality in (5.2.12) is not formally rigorous. Furthermore, $\boldsymbol{\Omega}_{1}^{t+h}$ generally implies neither the angular momentum conservation, nor energy conservation (cf. Section 5.2.3).

If the fully explicit version of the scheme is to be executed, one would nevertheless like to make some use of $\boldsymbol{\Omega}_{1}^{t+h}$. As the right hand side of (5.2.12) is in $T_{\boldsymbol{\Lambda}(t+h/2)}SO(3)$, and it is supposed to approximate $\boldsymbol{\Omega}(t+h)$, one can notionally interpret (5.2.12) as an assignment to $\boldsymbol{\Omega}_{1}^{t+h}$ of its own pull-back (along the exponential map) to $T_{\boldsymbol{\Lambda}(t+h/2)}SO(3)$. Now formula (5.2.16) becomes a "consistent" backward Euler step, updating the mid-step rotation into $\boldsymbol{\Lambda}^{t+h}$. There also holds

$$\mathbf{T}^{T}\left[\frac{h}{2}\boldsymbol{\Omega}_{1}^{t+h}\right]\boldsymbol{\Omega}_{1}^{t+h} = \boldsymbol{\Omega}_{1}^{t+h} \tag{5.2.19}$$

which happens to alleviate the inconsistency (again, this is only a notional trick). The scheme (5.2.9-5.2.13) has two drawbacks: conservation of the angular momentum is only approximate, and the kinetic energy experiences a positive drift. This is remedied in (5.2.14), where the angular momentum conservation is algorithmically enforced. As will be illustrated, the scheme (5.2.9-5.2.14) has a negative energy drift and becomes strongly dissipative for large time steps.

Although in applications involving small incremental rotations (e.g. constrained systems) the scheme (5.2.9-5.2.14) will be often sufficient, it is useful to have at hand a refined method, that does not experience the energy drift. Formulae (5.2.9-5.2.13) are still of use, although $\boldsymbol{\Omega}_1^{t+h}$ becomes now merely a dummy variable. Equation (5.2.12) needs to be stated only to solve for the constraint reactions (which contribute to $\mathbf{T}^{t+\frac{h}{2}}$). After that, the final implicit Euler half-step is executed more rigorously. As the configuration has already been advanced from $\boldsymbol{\Lambda}^t$ to $\boldsymbol{\Lambda}^{t+\frac{h}{2}}$, we do not wish to undo it. Rather, the following mid-point approximation of (5.2.8) is exercised

$$\exp\left[\frac{h}{2}\boldsymbol{\Omega}^t\right]\exp\left[\frac{h}{2}\boldsymbol{\Omega}^{t+h}\right]\mathbf{J}\boldsymbol{\Omega}^{t+h}=\mathbf{J}\boldsymbol{\Omega}^t+h\mathbf{t}^{t+\frac{h}{2}} \tag{5.2.20}$$

where the first exponential has already been computed, while the second one implicitly involves $\boldsymbol{\Omega}^{t+h}$. It should be noted, that the rotation update $\boldsymbol{\Lambda}(t+h)=\boldsymbol{\Lambda}(t)\exp[\boldsymbol{\Psi}(h)]$ makes sense, provided $\boldsymbol{\Psi}(h)\in T_{\boldsymbol{\Lambda}(t)}SO(3)$. In that respect, while the first update $\boldsymbol{\Lambda}^{t+\frac{h}{2}}=\boldsymbol{\Lambda}^t\exp\left[\frac{h}{2}\boldsymbol{\Omega}^t\right]$ is correct, the consecutive one $\boldsymbol{\Lambda}^{t+h}=\boldsymbol{\Lambda}^{t+\frac{h}{2}}\exp\left[\frac{h}{2}\boldsymbol{\Omega}^{t+h}\right]$ might seem inconsistent. More correctly, there should hold

$$\boldsymbol{\Lambda}^{t+h}=\boldsymbol{\Lambda}^{t+\frac{h}{2}}\exp\left[\mathbf{T}^T\left[-\frac{h}{2}\boldsymbol{\Omega}^{t+h}\right]\frac{h}{2}\boldsymbol{\Omega}^{t+h}\right] \tag{5.2.21}$$

where $\frac{h}{2}\boldsymbol{\Omega}^{t+h}\in T_{\boldsymbol{\Lambda}(t+h)}SO(3)$ was carried over to $T_{\boldsymbol{\Lambda}(t+h/2)}SO(3)$ by means of the reverse half-rotation $\boldsymbol{\Psi}(h)=-\frac{h}{2}\boldsymbol{\Omega}^{t+h}$, and hence $\mathbf{T}^T\left[-\frac{h}{2}\boldsymbol{\Omega}^{t+h}\right]$. Again, by the collinearity argument, there follows $\mathbf{T}^T\left[-\frac{h}{2}\boldsymbol{\Omega}^{t+h}\right]\frac{h}{2}\boldsymbol{\Omega}^{t+h}=\frac{h}{2}\boldsymbol{\Omega}^{t+h}$. The implicit solution (5.2.15) requires few iterations of Newton scheme. The velocity $\boldsymbol{\Omega}_1^{t+h}$ is used as an initial guess. The final configuration update follows in (5.2.16).

In the sequel the scheme (5.2.9-5.2.13) will be addressed as NEW1, the scheme (5.2.9-5.2.14) will be addressed as NEW2 and the scheme (5.2.9-5.2.12, 5.2.15-5.2.16) will be addressed as NEW3.

5.2.3. Conservation and stability. Conservation and stability properties are most conveniently analysed in the space of referential angular momenta, $\boldsymbol{\Pi}=\mathbf{J}\boldsymbol{\Omega}$. Assume, that the external torque $\mathbf{t}\equiv\mathbf{0}$. Conservation of the spatial angular momentum reads then

$$\boldsymbol{\Lambda}(t)\boldsymbol{\Pi}(t)=\boldsymbol{\Lambda}(0)\boldsymbol{\Pi}(0) \tag{5.2.22}$$

which together with the conservation of the kinetic energy implies

$$\frac{1}{2}\boldsymbol{\Pi}^T(t)\boldsymbol{\Pi}(t)=\frac{1}{2}\boldsymbol{\Pi}^T(0)\boldsymbol{\Pi}(0) \tag{5.2.23}$$

$$\frac{1}{2}\boldsymbol{\Pi}^T(t)\mathbf{J}^{-1}\boldsymbol{\Pi}(t)=\frac{1}{2}\boldsymbol{\Pi}^T(0)\mathbf{J}^{-1}\boldsymbol{\Pi}(0) \tag{5.2.24}$$

where the kinetic energy $E_k=\frac{1}{2}\boldsymbol{\Omega}^T\mathbf{J}\boldsymbol{\Omega}$. Free rigid rotation can be then viewed as a purely geometrical problem of intersection between the sphere (5.2.23) and the ellipsoid (5.2.24) in the $\boldsymbol{\Pi}$-space. In general, the intersection curve is of higher

order and cannot be written down in an explicit form. A rotation integrator traces the curve numerically. In particular, let us have a look at formula (5.2.12)

$$\mathbf{\Pi}_1^{t+h} = \mathbf{\Pi}^t + h\mathbf{\Pi}^{t+\frac{h}{2}} \times \mathbf{\Omega}^{t+\frac{h}{2}} \tag{5.2.25}$$

At any time t, $\mathbf{\Pi}(t)$ is normal to the momentum sphere (5.2.23) and $\mathbf{\Omega}(t)$ is normal to the energy ellipsoid (5.2.24). Hence, the product $\mathbf{\Pi}^{t+\frac{h}{2}} \times \mathbf{\Omega}^{t+\frac{h}{2}}$ can be interpreted as an approximation of the tangent to the intersection curve at $t+h/2$, and (5.2.25) becomes a surface intersection tracing scheme. In our case, $\mathbf{\Pi}^{t+\frac{h}{2}}$ is obtained from $\mathbf{\Pi}^t$ by rolling on the surface of the momentum sphere according to the formula

$$\mathbf{\Pi}^{t+\frac{h}{2}} = \exp\left[-\frac{h}{2}\mathbf{\Omega}^t\right]\mathbf{\Pi}^t \tag{5.2.26}$$

This is a first order update, as it results from the solution of a linear ordinary equation of rotation about a fixed axis (cf. remarks on the origin of the exponential map in Section 3.1). Hence, $\mathbf{\Pi}^{t+\frac{h}{2}} \times \mathbf{\Omega}^{t+\frac{h}{2}} = \mathbf{\Pi}\left(t+\frac{h}{2}\right) \times \mathbf{\Omega}\left(t+\frac{h}{2}\right) + O\left(h^2\right)$, where $\mathbf{\Pi}(t), \mathbf{\Omega}(t)$ is the exact solution. In analogy with Section 5.1.1, one can show that (5.2.25) is of second order. Unfortunately, as a tangent to two convex surfaces is used, points generated by (5.2.25) lay outside of both surfaces. Only with $h \to 0$ they approach the actual intersection curve. For large h it is easy to step far outside of both surfaces and rapidly climb up over the increasing energy levels. NEW1 conserves neither the momentum nor the energy and is prone to the catastrophic energy blowup.

By algorithmic enforcement of the momentum conservation (5.2.14), the solution iterates cling to the momentum sphere. There holds

$$\mathbf{\Pi}_2^{t+h} = \exp\left[-\frac{h}{2}\mathbf{\Omega}_1^{t+h}\right]\exp\left[-\frac{h}{2}\mathbf{\Omega}^t\right]\mathbf{\Pi}^t \tag{5.2.27}$$

and thus, one always stays on the surface of the conserved momentum. Staying within a compact set prevents an unbounded growth of the energy. The energy blowup is not possible for NEW2. The dissipative behaviour of the scheme however, is not explained by this fact alone. Generally, a sequence of points on a compact set will have at least one accumulation point. Qualitatively, only three types of behaviour are possible (Figure 5.2.1):

- *Swelling* of the energy ellipsoid until its smallest radius and the radius of the momentum sphere become equal. The final state corresponds to the stable rotation about the axis of the minimum moment of inertia. This behaviour is typical for first order updates of kind (5.2.26), but also for example the explicit scheme by Simo and Wong [**100**].
- *Shrinking* of the energy ellipsoid until its largest radius and the radius of the momentum sphere become equal. The final state corresponds to the stable rotation about the axis of the maximum moment of inertia. This is the case for NEW2.
- *Oscillation* about the intersection curve of the energy ellipsoid and the momentum sphere. This is the case for NEW3, as well as for many other implicit algorithms [**126, 171**].

Swelling is the easiest to analyse. While applying (5.2.26) we would like to know, how much the energy grows from t to $t+h/2$. This can be estimated be the linearisation of the mid-step kinetic energy with respect to the time step. The mid-step energy reads

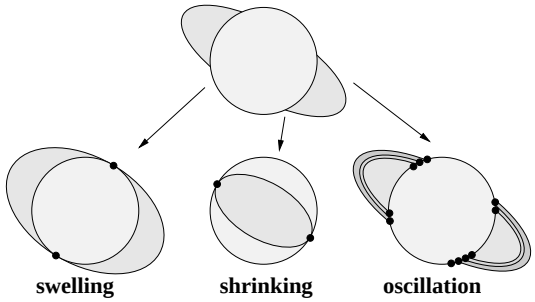


FIGURE 5.2.1. Qualitative behaviour of integration methods enforcing conservation of the spatial angular momentum. Section through the momentum sphere and the energy ellipsoid. The ellipsoid either swells (schemes with positive energy drift), shrinks (schemes with negative energy drift), or oscillates (stable schemes).

$$E_k^{t+\frac{h}{2}} = \frac{1}{2}\left\langle \boldsymbol{\Pi}^t, \exp\left[\frac{h}{2}\boldsymbol{\Omega}^t\right]\mathbf{J}^{-1}\exp\left[-\frac{h}{2}\boldsymbol{\Omega}^t\right]\boldsymbol{\Pi}^t\right\rangle \tag{5.2.28}$$

and its increment is roughly

$$\triangle E_k^{t+\frac{h}{2}} \simeq \dot{E}_k^{t+\frac{h}{2}}(0)\,h + \ddot{E}_k^{t+\frac{h}{2}}(0)\,\frac{h^2}{2} \tag{5.2.29}$$

where $\dot{E}_k(0) = \frac{d}{dh}E_k(h)\big|_{h=0}$. The first derivative of the energy reads then

$$\begin{aligned}
\dot{E}_k^{t+\frac{h}{2}}(0) &= \frac{1}{4}\left\langle \boldsymbol{\Pi}^t, \hat{\boldsymbol{\Omega}}^t\mathbf{J}^{-1}\boldsymbol{\Pi}^t\right\rangle - \frac{1}{4}\left\langle \boldsymbol{\Pi}^t, \mathbf{J}^{-1}\hat{\boldsymbol{\Omega}}^t\boldsymbol{\Pi}^t\right\rangle \\
&= \frac{1}{4}\left\langle \boldsymbol{\Pi}^t, \hat{\boldsymbol{\Omega}}^t\mathbf{J}^{-1}\boldsymbol{\Pi}^t\right\rangle - \frac{1}{4}\left\langle \mathbf{J}^{-1}\boldsymbol{\Pi}^t, \hat{\boldsymbol{\Omega}}^t\boldsymbol{\Pi}^t\right\rangle \\
&= \frac{1}{4}\left\langle \boldsymbol{\Pi}^t, \hat{\boldsymbol{\Omega}}^t\boldsymbol{\Omega}^t\right\rangle - \frac{1}{4}\left\langle \boldsymbol{\Omega}^t, \hat{\boldsymbol{\Omega}}^t\boldsymbol{\Pi}^t\right\rangle = 0
\end{aligned} \tag{5.2.30}$$

where $\hat{\boldsymbol{\Omega}}^t\boldsymbol{\Omega}^t = \mathbf{0}$ and $\boldsymbol{\Omega}^t \perp \hat{\boldsymbol{\Omega}}^t\boldsymbol{\Pi}^t$ were used. The second derivative takes the following form

$$\begin{aligned}
\ddot{E}_k^{t+\frac{h}{2}}(0) &= \frac{1}{8}\left\langle \boldsymbol{\Pi}^t, \hat{\boldsymbol{\Omega}}^t\hat{\boldsymbol{\Omega}}^t\mathbf{J}^{-1}\boldsymbol{\Pi}^t\right\rangle - \frac{1}{4}\left\langle \boldsymbol{\Pi}^t, \hat{\boldsymbol{\Omega}}^t\mathbf{J}^{-1}\hat{\boldsymbol{\Omega}}^t\boldsymbol{\Pi}^t\right\rangle + \frac{1}{8}\left\langle \boldsymbol{\Pi}^t, \mathbf{J}^{-1}\hat{\boldsymbol{\Omega}}^t\hat{\boldsymbol{\Omega}}^t\boldsymbol{\Pi}^t\right\rangle \\
&= \frac{1}{8}\left\langle \boldsymbol{\Pi}^t, \hat{\boldsymbol{\Omega}}^t\hat{\boldsymbol{\Omega}}^t\boldsymbol{\Omega}^t\right\rangle + \frac{1}{4}\left\langle \hat{\boldsymbol{\Omega}}^t\boldsymbol{\Pi}^t, \mathbf{J}^{-1}\hat{\boldsymbol{\Omega}}^t\boldsymbol{\Pi}^t\right\rangle + \frac{1}{8}\left\langle \boldsymbol{\Omega}^t, \hat{\boldsymbol{\Omega}}^t\hat{\boldsymbol{\Omega}}^t\boldsymbol{\Pi}^t\right\rangle \\
&= \frac{1}{4}\left\langle \hat{\boldsymbol{\Pi}}^t\boldsymbol{\Omega}^t, \mathbf{J}^{-1}\hat{\boldsymbol{\Pi}}^t\boldsymbol{\Omega}^t\right\rangle
\end{aligned} \tag{5.2.31}$$

where terms with $\frac{1}{8}$ vanish by similar arguments. Finally

$$\triangle E_k^{t+\frac{h}{2}} \simeq \frac{h^2}{8}\left\langle \hat{\boldsymbol{\Pi}}^t\boldsymbol{\Omega}^t, \mathbf{J}^{-1}\hat{\boldsymbol{\Pi}}^t\boldsymbol{\Omega}^t\right\rangle \tag{5.2.32}$$

The above energy increment is always positive due to the same definiteness of $\mathbf{J}$. Clearly, if update (5.2.26) was to be solely used for advancing the motion, the solution point would climb up the energy levels on the surface of the momentum sphere, until $\boldsymbol{\Pi}^t$ and $\boldsymbol{\Omega}^t$ would become aligned and no more growth could happen. At that stage, the energy ellipsoid would contain the momentum sphere and their intersection would comprise only two opposite points.

The final update of momentum in NEW2 reads

$$\mathbf{\Pi}_2^{t+h} = \exp\left[-\frac{h}{2}\mathbf{\Omega}_1^{t+h}\right]\mathbf{\Pi}^{t+\frac{h}{2}} \tag{5.2.33}$$

Point $\mathbf{\Pi}^{t+\frac{h}{2}}$ corresponds to the energy growth by at least (5.2.32). We shall investigate, whether the energy can be further increased by performing the step (5.2.33). Note, that (5.2.33) describes rotation of $\mathbf{\Pi}^{t+\frac{h}{2}}$ about the fixed axis $\mathbf{\Omega}_1^{t+h}$. At time $t+h/2$ we shall consider the instantaneous linearisation of (5.2.33)

$$\left.\frac{d}{dh}\mathbf{\Pi}_2^{t+h}\right|_{h=0} = \frac{1}{2}\mathbf{\Pi}^{t+\frac{h}{2}} \times \mathbf{\Omega}_1^{t+h} \tag{5.2.34}$$

A linearised stability criterion is that $\left.\frac{d}{dh}\mathbf{\Pi}_2^{t+h}\right|_{h=0}$ should not have a component along the direction of the energy growth. Namely

$$\left\langle \mathbf{\Pi}^{t+\frac{h}{2}} \times \mathbf{\Omega}_1^{t+h}, \mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle \leq 0 \tag{5.2.35}$$

where $\mathbf{\Omega}^{t+\frac{h}{2}}$ is the energy gradient at $t+h/2$, and the factor of $\frac{1}{2}$ was dropped off. The above condition can be expanded as follows

$$\begin{aligned}
\left\langle \mathbf{\Pi}^{t+\frac{h}{2}} \times \left[\mathbf{\Omega}^t - \mathbf{J}^{-1}h\left(\mathbf{\Omega}^{t+\frac{h}{2}} \times \mathbf{J}\mathbf{\Omega}^{t+\frac{h}{2}}\right)\right], \mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle &= \\
\left\langle \mathbf{\Pi}^{t+\frac{h}{2}} \times \left[\mathbf{\Omega}^t + \mathbf{J}^{-1}h\left(\mathbf{\Pi}^{t+\frac{h}{2}} \times \mathbf{\Omega}^{t+\frac{h}{2}}\right)\right], \mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle &= \\
\left\langle \mathbf{\Pi}^{t+\frac{h}{2}} \times \mathbf{\Omega}^t, \mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle + \left\langle \mathbf{\Pi}^{t+\frac{h}{2}} \times \left[\mathbf{J}^{-1}h\left(\mathbf{\Pi}^{t+\frac{h}{2}} \times \mathbf{\Omega}^{t+\frac{h}{2}}\right)\right], \mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle &= \\
\left\langle \hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{\Omega}^t, \mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle + h\left\langle \hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{J}^{-1}\hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{\Omega}^{t+\frac{h}{2}}, \mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle &= \\
-\left\langle \mathbf{\Omega}^t, \hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle - h\left\langle \hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{\Omega}^{t+\frac{h}{2}}, \mathbf{J}^{-1}\hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle &=
\end{aligned} \tag{5.2.36}$$

Let us now define three functions

$$a(h) = -\left\langle \mathbf{\Omega}^t, \hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle \tag{5.2.37}$$

$$b(h) = -\left\langle \hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{\Omega}^{t+\frac{h}{2}}, \mathbf{J}^{-1}\hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{\Omega}^{t+\frac{h}{2}} \right\rangle \tag{5.2.38}$$

$$c(h) = \frac{1}{2}\left(a(h) + hb(h)\right) \tag{5.2.39}$$

where for $c(h)$, the previously dropped factor of $\frac{1}{2}$ was restored. The stability criterion reads now

$$a(h) + hb(h) \leq 0 \tag{5.2.40}$$

Obviously, $b(h) \leq 0$ for any h due to the positive definiteness of $\mathbf{J}^{-1}$. On the other hand, a simple geometric arguments shows that, at least for small h, function $a(h) \geq 0$. In order to see that, one needs to consider circulation of $\mathbf{\Pi}(t)$ along the intersection curve. Due to the interpretation of $\hat{\mathbf{\Pi}}(t)\mathbf{\Omega}(t)$ as the tangent to the curve, $\hat{\mathbf{\Pi}}(t+s)\mathbf{\Omega}(t+s)$ points away from $\mathbf{\Pi}(t)$ for some sufficiently small $s > 0$, because $\mathbf{\Pi}(t+s)$ runs away from $\mathbf{\Pi}(t)$ along $\hat{\mathbf{\Pi}}(t+s)\mathbf{\Omega}(t+s)$. As $\mathbf{\Omega}(t)$ is normal to the tangent plane of the energy ellipsoid at time t, the complete intersection curve lays behind this plane. For sufficiently small s, point $\mathbf{\Pi}(t+s)$

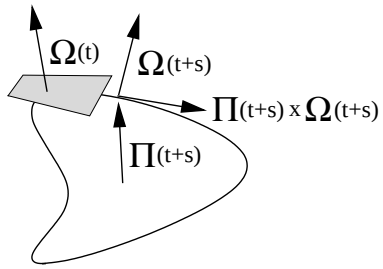


FIGURE 5.2.2. Circulation of $\mathbf{\Pi}(t)$ along the sphere-ellipsoid intersection curve. Due to the convexity of the intersecting surfaces, for small s there holds $\langle \mathbf{\Omega}(t), \mathbf{\Pi}(t+s) \times \mathbf{\Omega}(t+s) \rangle \leq 0$.

runs away from the plane and thus $\hat{\mathbf{\Pi}}(t+s)\,\mathbf{\Omega}(t+s)$ does not have a component aligned with the normal $\mathbf{\Omega}(t)$. Hence, $a(h) \geq 0$ for small h (Figure 5.2.2). In order to verify condition (5.2.40) for $h \to 0$, the following linearisation is considered

$$a(0) + \dot{a}(0)\,h + 0b(0) + b(0)\,h + O\left(h^2\right) \leq 0 \tag{5.2.41}$$

where the over-dot corresponds to $\frac{d}{dh}$. Recalling, that $\mathbf{\Omega}^{t+\frac{h}{2}} = \mathbf{J}^{-1} \exp\left[-\frac{h}{2}\mathbf{\Omega}^t\right] \mathbf{\Pi}^t$, one obtains

$$\frac{d}{dh}\left(\mathbf{\Omega}^{t+\frac{h}{2}}\right)\Big|_{h=0} = -\frac{1}{2}\mathbf{J}^{-1}\hat{\mathbf{\Omega}}^t\mathbf{\Pi}^t = \frac{1}{2}\mathbf{J}^{-1}\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t \tag{5.2.42}$$

and

$$\begin{aligned} \frac{d}{dh}\left(\hat{\mathbf{\Pi}}^{t+\frac{h}{2}}\mathbf{\Omega}^{t+\frac{h}{2}}\right)\Big|_{h=0} &= \frac{d}{dh}\left(\mathbf{\Pi}^{t+\frac{h}{2}}\right)\Big|_{h=0} \times \mathbf{\Omega}^t + \mathbf{\Pi}^t \times \frac{d}{dh}\left(\mathbf{\Omega}^{t+\frac{h}{2}}\right)\Big|_{h=0} \\ &= \frac{1}{2}\left\{\left[\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right] \times \mathbf{\Omega}^t + \mathbf{\Pi}^t \times \mathbf{J}^{-1}\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right\} \end{aligned} \tag{5.2.43}$$

so that

$$\begin{aligned} \dot{a}(0) &= -\left\langle \mathbf{\Omega}^t, \frac{1}{2}\left\{\left[\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right] \times \mathbf{\Omega}^t + \mathbf{\Pi}^t \times \mathbf{J}^{-1}\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right\}\right\rangle \\ &= 0 - \frac{1}{2}\left\langle \mathbf{\Omega}^t, \hat{\mathbf{\Pi}}^t\mathbf{J}^{-1}\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right\rangle = \frac{1}{2}\left\langle \hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t, \mathbf{J}^{-1}\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right\rangle \end{aligned} \tag{5.2.44}$$

As $a(0) = -\left\langle \mathbf{\Omega}^t, \hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right\rangle = 0$ and $b(0) = -\left\langle \hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t, \mathbf{J}^{-1}\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right\rangle$, there holds

$$\begin{aligned} a(0) + \dot{a}(0)\,h + 0b(0) + b(0)\,h + O\left(h^2\right) &= \\ \left(\dot{a}(0) + b(0)\right)h + O\left(h^2\right) &= \\ -\frac{h}{2}\left\langle \hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t, \mathbf{J}^{-1}\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right\rangle + O\left(h^2\right) &\leq 0 \end{aligned} \tag{5.2.45}$$

This shows, that for sufficiently small h, the kinetic energy is always decreased from $t + h/2$ to $t + h$ for the scheme NEW2. The amount of the energy drop can be estimated as

$$\triangle E_k^{t+h} \simeq \frac{h}{2}c(h) \simeq -\frac{h^2}{8}\left\langle \hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t, \mathbf{J}^{-1}\hat{\mathbf{\Pi}}^t\mathbf{\Omega}^t\right\rangle \tag{5.2.46}$$

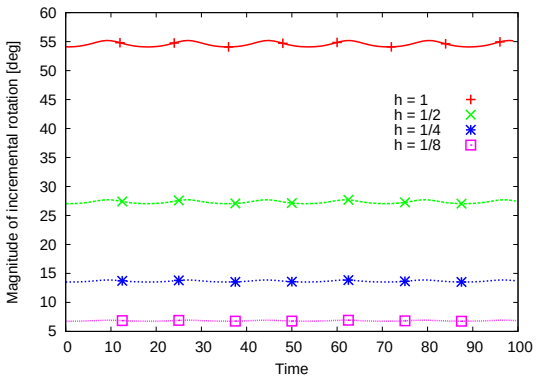


FIGURE 5.2.3. Free rotation. Magnitude of the incremental rotation vector at a range of time steps.

which together with (5.2.32) shows, that up to the second order terms the energy growth and drop cancel out each other. In other words

$$E_k^{t+h} = E_k^t + O\left(h^3\right) \tag{5.2.47}$$

This conclusion is not really significant, as it does not imply that NEW2 is a *shrinking* scheme. In fact, numerical analysis shows that the long term negative drift of NEW2 is overlapped by some up and down oscillations, related to the local curvature of the intersecting surfaces. This suggests, that the local analysis of the above kind cannot be conclusive. We do not attempt further analysis. In the following section we resort instead to the numerical example.

5.2.4. Free rotation. More examples will follow in Chapter 13. The current one is referred to after Krysl [**126**] and is meant to provide a brief summary of the essential features of the proposed schemes. The initial rotation is identity, the initial angular velocity reads $\mathbf{\Omega}^0 = [0.45549, 0.82623, 0.03476]$, and the referential inertia tensor is $\mathbf{J} = \mathrm{diag}\,[0.9144, 1.098, 1.66]$. No external forcing is assumed.

The proposed schemes are compared against LIEMID[EA] by Krysl [**126**], which is one of the best performing schemes today (although its computational cost per time step is rather high). In some of the comparisons the explicit scheme by Simo and Wong [**100**] is also included, as it requires relatively little computational effort per time step. It should be noted that neither the explicit scheme by Simo and Wong, nor LIEMID[EA] comply with the algebraic structure of (5.1.2), which from our point of view is a drawback.

Figure 5.2.3 illustrates the magnitudes of the incremental rotation vector computed with NEW3, at a range of time steps. It is seen that small increments of rotation, say $\|\mathbf{\Psi}\| \ll 10\,\mathrm{deg}$[1], occur for time steps $h < 1/8$. This range of incremental rotations is of the main interest here, although for the sake of illustration this and other examples include larger increments.

Figure 5.2.4 illustrates the characteristic momentum phase space behaviour of the proposed schemes. The plots have been obtained over 500 steps of size $h = 1$ (about 55 deg of incremental rotation per time step). Clearly, NEW1 diverges gradually towards the energy blowup. NEW2 dissipates the energy and after a few tens of steps around the original intersection curve, it switches to the qualitatively new state, asymptotically equivalent to the rotation about the axis of the maximum

[1] $\|\Psi\| = \left\|\frac{h}{2}\mathbf{\Omega}^t\right\| + \left\|\frac{h}{2}\mathbf{\Omega}^{t+h}\right\|$ is used for illustration

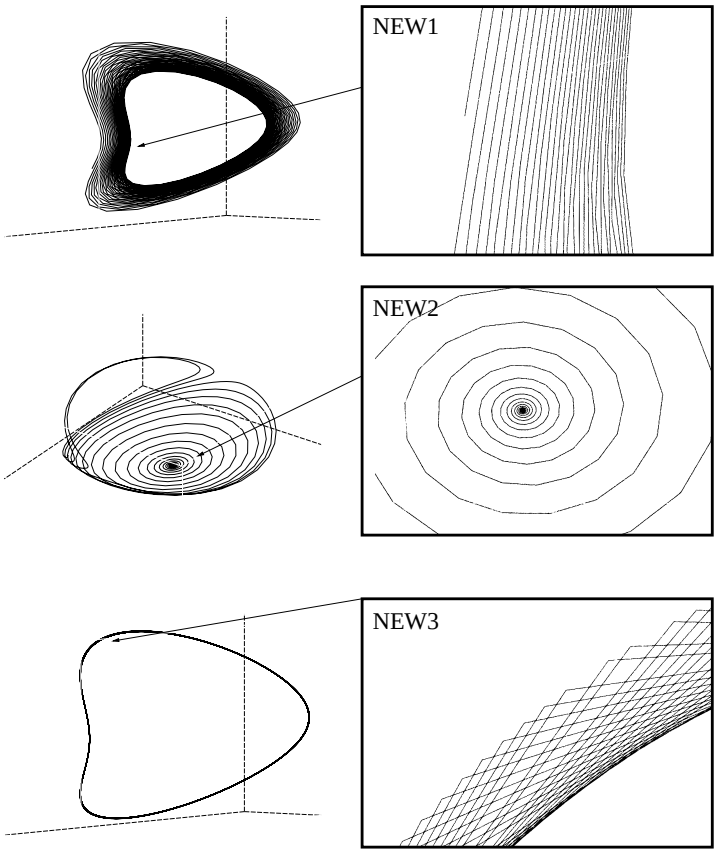


FIGURE 5.2.4. Free rotation. Body-frame angular momentum space plots for 500 steps of size $h = 1$ (about 55 deg of incremental rotation per time step). The large time step allows to capture characteristic behaviour of all three schemes. NEW1 gradually diverges, and it is about to blow up within the next few hundreds of iterations. NEW2 dissipates energy until a stable rotation about the axis of the maximum moment of inertia is reached. NEW3 stably oscillates about the original intersection curve between the momentum sphere and the energy ellipsoid.

moment of inertia. NEW3, on the other hand, oscillates stably about the original intersection curve between the momentum sphere and the energy ellipsoid.

Figure 5.2.5 illustrates the characteristic energy behaviour of the proposed algorithms. NEW1 experiences a positive energy drift, while NEW2 experiences nearly symmetrical negative energy drift. NEW3, similarly to LIEMID[EA] displays excellent stability although the solution in both cases is oscillatory. NEW3 oscillates on the negative side and with larger amplitude then LIEMID[EA]. The latter method oscillates on the positive side.

Figure 5.2.6 illustrates conservation of the spatial angular momentum ($\pi = \mathbf{\Lambda\Pi}$). NEW2, NEW3 and LIEMID[EA] clearly conserve the angular momentum (which is their algorithmic feature). On the other hand, NEW1 displays an oscillatory drift for the large time step. For the smaller step, although not visible in the figure, the drift is still present.

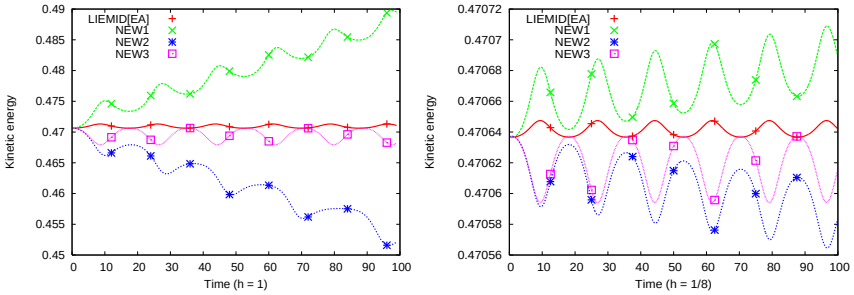


FIGURE 5.2.5. Free rotation. Kinetic energy for step sizes $h = 1$ (left) and $h = 1/8$ (right).

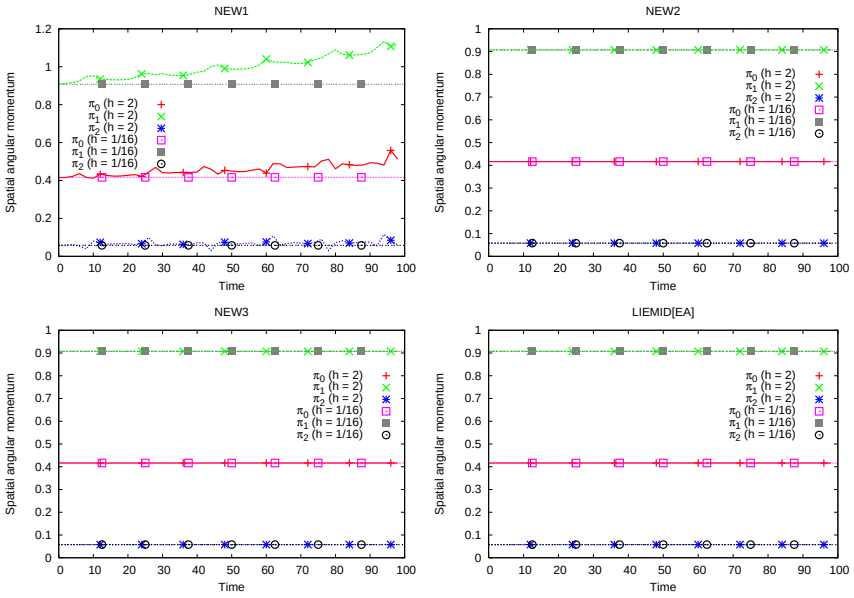


FIGURE 5.2.6. Free rotation. Spatial angular momentum for step sizes $h = 2$ and $h = 1/8$.

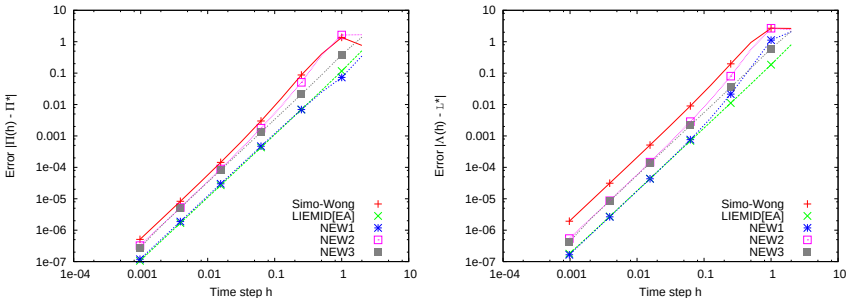


FIGURE 5.2.7. Free rotation. Convergence of the body-frame angular momentum $\mathbf{\Pi} = \mathbf{J\Omega}$ (left), and the rotation operator $\mathbf{\Lambda}$ (right). The reference solutions $\mathbf{\Pi}*$ and $\mathbf{\Lambda}*$ have been computed with LIEMID[EA] and the time step $h = 2^{-15}$ at time $t = 100$. The solutions $\mathbf{\Pi}(h)$ and $\mathbf{\Lambda}(h)$ were computed for time steps $h \in \{1, 2^{-1}, ..., 2^{-10}\}$.

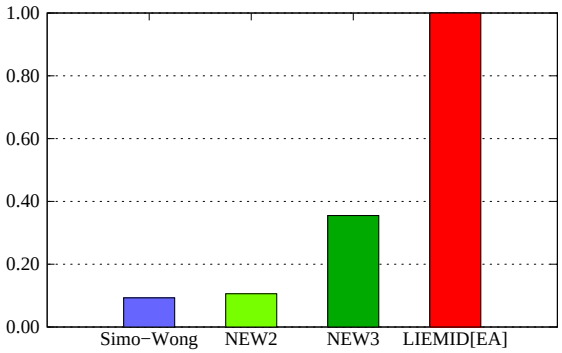


FIGURE 5.2.8. Free rotation. Normalised runtimes comparison for ten million steps of size $h = \frac{1}{8}$.

Figure 5.2.7 illustrates the convergence in the L_2 norm, of the referential angular momentum $\Pi = \mathbf{J\Omega}$ and the rotation operator $\mathbf{\Lambda}$. The reference solutions $\mathbf{\Pi}*$ and $\mathbf{\Lambda}*$ have been computed with LIEMID[EA] and the time step $h = 2^{-15}$ at time $t = 100$. The solutions $\mathbf{\Pi}(h)$ and $\mathbf{\Lambda}(h)$ were computed for time steps $h \in \{1, 2^{-1}, ..., 2^{-10}\}$ at time $t = 100$. It is seen that all of the compared algorithms are second order accurate. All versions of the new scheme outperform the explicit algorithm by Simo and Wong [**100**]. Interestingly NEW1 displays excellent accuracy of the body-frame angular momentum and performs on a par with LIEMID[EA]. For small time steps the accuracy of the rotation operator obtained with NEW1 also compares well with the one reached by LIEMID[EA].

5.2.5. Efficiency. Many of the recently proposed algorithms [**126, 163, 171**] posses excellent stability properties and can pursue their tasks with extremely large $O(\pi)$ incremental rotations. The price for those advantages lies in the necessity for solving local implicit problems, for which Newton iterations are usually employed. For large time steps, the local solutions involve evaluations of the exponential map at the magnitudes of the rotation angle, for which the truncated Taylor expansion of $\exp[\cdot]$ is not effective. Thus, although sparse steps can be performed, the cost of an individual step is high.

In the explicit multi-body analysis with contacts and joints the possibility of performing $O(\pi)$ steps does not seem practical. The time step has to be small enough in order to capture the geometrical nonlinearities of the multi-body interactions. This is why a lightweight, but well behaved time-stepper is usually a better choice. In this respect, NEW2 involves evaluation of only two exponential maps per step. For small incremental rotations this can be well dealt with by the truncated Taylor expansion of $\exp[\cdot]$.

For long term simulations, where the negative drift of NEW2 cannot be accepted, NEW3 seems to be a good alternative, as it retains the explicitness of the force evaluation and improves much upon the stability. Nevertheless, the single implicit problem needs to be solved. In order to evaluate and compare the relative efficiency of the proposed schemes, ten millions time steps of size $\frac{1}{8}$ has been performed for the free rotation example of the previous section. Figure 5.2.8 summarises the normalised runtimes. The explicit scheme by Simo and Wong [**100**] computes only one exponential map and hence requires least time. NEW2 with its two exponential map evaluations places itself right after the scheme by Simo and Wong. NEW3 on the other hand takes roughly half of the time needed by Krysl's LIEMID[EA] [**126**]. This is because the latter method involves solution of two implicit problems per time step.

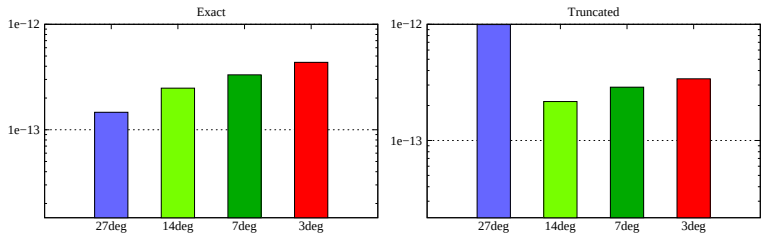


FIGURE 5.2.9. Free rotation. Loss of orthogonality illustrated by the $\left\|\mathbf{I}-\mathbf{\Lambda}^T\mathbf{\Lambda}\right\|$ norms computed with NEW3 after one million steps with $h \in \left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\right\}$. The left graph summarises the results computed with the numerically exact routines. The right graph corresponds to the truncated expansion of $\exp[\cdot]$.

As the truncated Taylor expansion of $\exp[\cdot]$ has been mentioned above, it is relevant to verify whether the orthogonality of the rotation operator has not been compromised. Figure 5.2.9 illustrates the norms $\left\|\mathbf{I}-\mathbf{\Lambda}^T\mathbf{\Lambda}\right\|$ computed with NEW3 after one million steps of the free rotation test at a range of time steps $h = \frac{1}{2}, ..., \frac{1}{16}$. The left graph corresponds to the numerically exact computations (library routines has been used). The right graph corresponds to the same computations employing the truncated expansion of the scalar terms in (5.2.2). It is seen that only for the largest incremental rotation magnitude (27 deg) some loss of orthogonality can be observed. Six terms in the expansions were used.

5.3. Quasi-statics

Quasi-static multi-body simulations with contact constraints represent a subtle issue. During this sort of simulation individual members of a multi-body structure can undergo limited rigid motion affecting the global deformation mode, while the dynamic effects related to elastic deformability can remain negligible. In those circumstances purely static formulation does not provide sufficient information, as the contact forces are transmitted mainly due to the freedom of rigid motion. Tangent stiffness operator resulting from the static formulation of a multi-body system has a vast null-space, making it necessary to introduce some sort of regularisation. While this regularisation is expected to provide a meaningful representation of rigid modes, it seems most natural to adopt the dynamic formulation for that purpose. The classical dynamic relaxation technique by Underwood [**205**], constructed around the central difference scheme, was already successfully applied to statics of granular materials (cf. Bardet and Proubet [**18**]). This was possible in the context of smoothed (penalty based) discrete element formulation, where availability of contact stiffness provides means for identifying globally dominant modes. Such information is not explicitly available in a non-smooth formulation. Nevertheless the dynamic approach is still of use. For assemblies of stiff bodies, which are mostly of interest here, optimally one would like to solve the quasi-static contact problem "on rigid modes" and only update the stresses on the way. The elastic deformability of such assemblies is limited, although presence of frictional sliding or rigid rocking does not necessarily render the structure unstable. In general some amount of slow rigid dislocations can happen before the onset of a dynamic failure mechanism. Quasi-static rigid motion was analysed to some extent in the field of robotics. For example Pang *et al.* [**165**] developed a linear programing technique to solve an uncoupled complementary problem resulting from a planar formulation. This work was later extended to three dimensions [**204**], where polyhedral discretisation of the friction cone allowed to preserve the original algebraic structure. The uncoupled

structure of the complementary problem resulted from the fact, that equilibrium of contact reactions and external forces was sought. Lack of the inertial term left the diagonal zero and the amount of rigid motion across a time step resulted directly from the assumed value of the step size and the velocity of time-dependent constraints. More generally, in a quasi-static simulation of a multi-body system, the amount of stepwise rigid motion is merely a rational modelling choice, adjusted to the velocity of a control mechanism. In the current development we decided not to discard the inertia regularisation. Instead the inertial term will be manipulated in order to deliver an expected behaviour. In the context of the Contact Dynamics method, Acary and Jean [**3**] discuss adaptation of a dynamic framework for the needs of a quasi-static simulation. A straightforward relaxation technique results from assuming velocities to be zero at the beginning of each iteration. This approach is adopted here. The modified backward Euler step follows

$$\mathbf{u}^{t+h} = \mathbf{A}^{-1} h \mathbf{f}\left(t+h, \mathbf{q}^t\right) + \mathbf{A}^{-1}\mathbf{H}^T\mathbf{R} \tag{5.3.1}$$

$$\mathbf{q}^{t+h} = \mathbf{q}^t + h\mathbf{u}^{t+h} \tag{5.3.2}$$

where

$$\mathbf{A} = \mathbf{M} + h^2 \left.\frac{\partial^2 \Psi}{\partial \mathbf{q} \partial \mathbf{q}}\right|_{\mathbf{q}^t} \tag{5.3.3}$$

$$\mathbf{H} = \mathbf{H}\left(\mathbf{q}^t\right) \tag{5.3.4}$$

Above, Ψ is the hyperelastic potential of the system and the remaining terms are interpreted as in (5.1.1-5.1.3). Equations (5.3.1) and (5.3.2) apply directly to the pseudo-rigid continuum case. One can write down a similar time stepping for rigid rotations, by obtaining an auxiliary extrapolation of the angular velocity $\mathbf{\Omega}^{t+h}$ with an analogue of (5.2.9), and then plunging it into $\mathbf{f}\left(t+h, \mathbf{q}^t\right)$. This way a consistent linearisation with respect to $\mathbf{\Omega}^{t+h}$ can be avoided, as it seems superfluous in this simplified setting. It has to be noted, that equation (5.3.1) holds true under the strong assumption of the constraints geometry remaining unchanged over the time interval $[t, t+h]$. Again, this simplifies implementation, as the linearisation with respect to $\mathbf{H}$ is avoided.

For the quasi-static simulation to make sense, it has to be assumed that a steady state solution exists at $t = 0$. After that instant some sort of control mechanism is executed at a slow rate. The displacement control seems most appropriate, as it introduces inertia-independent velocity. On the other hand, the existence of inertial terms allows force control to be utilised to some extent. In this case though, despite the fact that the above relaxation scheme rules out acceleration, once some unconstrained rigid motion occurs, the kinetic energy remains proportional to the out of balance portion of the applied force. Few additional remarks can be made:

(1) The operator $\mathbf{A}$ should be positive definite. Since it is equal to $\mathbf{M} + h^2\mathbf{K}$ (where $\mathbf{K}$ is the current tangent) it follows that the assumed time step h must be small enough. This will depend on material parameters, body volume and mass properties.

(2) Regarding the deformable part of the motion, it is desirable to impose uniform convergence of the Euler scheme for all bodies. At the same time information about their shape should not be discarded (so to account for rotations). Hence, the inertia matrices ought to be appropriately scaled. A reasonable amount of numerical damping can be obtained for $\lambda h \geq 4$,

where λ is a selected eigenvalue of $\mathbf{M}^{-1}\mathbf{K}$ [**98**]. For the pseudo-rigid model, a sufficient heuristic is to impose a uniform (across all bodies) distribution of $\lambda_{max}h$, where λ_{max} is the maximum eigenvalue.

(3) The amount of stepwise rigid motion (say, l_{max}) should be constrained. Even if the control mechanism introduces a bounded amount of rigid displacement, the possibility of free scattering of an assembly exists. One would expect a rational behaviour of the numerical scheme in such case. Appropriate scaling is possible for the linear part of the motion, as the dynamics of the mass centre is decoupled. A simple relation for the scalar mass follows $m = \frac{f_{max}h^2}{l_{max}}$, where f_{max} is the maximum magnitude of the resultant external force over all bodies. This choice provides a uniform bound for the stepwise linear displacement.

(4) At constraint points, velocity contributions of the rigid and deformable motion should be separated ($v_{deformable} \ll v_{rigid}$). Assuming an equilibrium configuration exists, numerical scheme (5.3.1-5.3.2) will converge no faster than the constraints reactions. Solution for the constraints delivers reactions adjusted to the dynamics of the overall system. In order to encourage fast identification of rigid modes, the velocity of those should dominate the stretch velocity at constraint points. If this condition is not satisfied, stretch velocities affect the constraint solver, which considerably slows down the convergence.

5.4. Literature

Selected developments, specific to the integration of rigid rotation are considered. In this respect, one of the early contributions is due to Benson and Hallquist [**23**], where the central difference scheme was applied to the spatial angular momentum balance. This simple scheme seems to have survived until recently in LS-DYNA software [**2**]. Simo and Vu-Quoc [**191**] apply the Newmark method to the body-frame angular momentum balance and develop an implicit scheme for the dynamics of rods undergoing large rotations. Nevertheless, the mid-point version of their algorithm conserves neither the energy nor the momentum. In a classical paper today, Simo and Wong [**100**] address this shortcoming by algorithmically enforcing conservation of the spatial angular momentum. This leads to an implicit scheme that conserves both the momentum and the energy. As a side-effect their main result, an explicit scheme examined in Section 5.2.4 is also given. An idea of discrete momentum conservation is also exploited by Park and Chiou [**110**], where the spatial central difference scheme is combined with the quaternion parametrisation based rotation update, in order to deliver an explicit scheme with good stability characteristics. An inexperienced reader should be warned however, that this paper contains some confusing notation flaws. In a short and informative paper, Omelyan [**162**] has proposed a lightweight semi-explicit leap-frog integrator, targeted at the molecular dynamics simulations. Krysl and Endres [**163**] developed a semi-explicit Newmark scheme with good stability properties, although not conserving the spatial angular momentum. Krysl [**126**] has derived a mid-point approximation of the incremental rotation angle, which gave rise to a well behaved implicit scheme and an explicit scheme LIEMIED[EA], examined in Section 5.2.4. Both conserve the spatial angular momentum (exactly) and the energy (in a stable, but oscillatory manner). In the following paper [**128**], Kyrsl discusses a family of implicit trapezoidal rule based integrators, which to some extent resemble the methods presented in Section 5.2.2. A fourth order Runge-Kutta method in the quaternion space has been given by Johnson *et al.* [**187**]. Kumar *et al.* [**171**] present several semi-implicit integrators, including a partitioned Runge-Kutta scheme (good

conservation properties, although relatively poor accuracy in the rotation space) and sub-cycling based method (very accurate and good conservation). It should be noted, that none of the listed methods directly complies with the algebraic structure of equations (5.1.1-5.1.3), which was the reason behind the developments of Section 5.2.2.

CHAPTER 6

Local frames

Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two bodies. Let us

(1) Pick spatial points $\mathbf{x}_1 \in \bar{\mathcal{B}_1}$ and $\mathbf{x}_2 \in \bar{\mathcal{B}_2}$ at some time t.
(2) Pick a coordinate system $\{\alpha^i\}$, with base $\mathbf{a}_i$ attached to $\mathbf{x}_1$, and deforming with $\mathcal{B}_1$ from t onwards.

We would like observe the motion of $\mathbf{x}_2$ from the perspective of the deforming *local frame* $\{\alpha^i\}$. For the relative displacement $\mathbf{d}_{\{x\}} = \mathbf{x}_2 - \mathbf{x}_1$, expressed in $\{\alpha^i\}$, there holds

$$\{\mathbf{a}_i\}\, \mathbf{d}_{\{\alpha\}} = \mathbf{d}_{\{x\}} \tag{6.0.1}$$

where $\{\mathbf{a}_i\}$ is the 3×3 matrix of column-wise base vectors. Above, both the base $\mathbf{a}_i$ and the relative displacement $\mathbf{d}_{\{x\}}$ change in time. From the point of view of an observer embedded in the local frame $\{\alpha^i\}$ however, $\mathbf{d}_{\{\alpha\}}$ changes only as far, as $\mathbf{x}_2$ fails to be convected with the motion of $\mathcal{B}_1$. The rate of such change is described by the Lie derivative of $\mathbf{d}_{\{\alpha\}}$ with respect to the flow induced by the motion of $\mathcal{B}_1$. In order to show that, let us first notice that

$$\mathbf{d}_{\{\alpha\}} = \{\mathbf{a}_i\}^{-1}\, \mathbf{d}_{\{x\}} = \left\{\mathbf{a}^i\right\}^T \mathbf{d}_{\{x\}} \tag{6.0.2}$$

where $\mathbf{a}^i$ are elements of the dual base, and the fact that $\left\{\mathbf{a}^i\right\}^T \{\mathbf{a}_i\} = \mathbf{I}$ was exploited. Without loss of generality, taking as an example the pseudo-rigid motion, one obtains

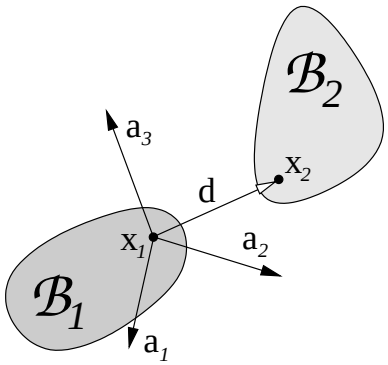


FIGURE 6.0.1. A local frame.

$$\begin{aligned}
\overset{\diamond}{\mathbf{d}}_{\{\alpha\}} &= \left\{\overset{\diamond}{\mathbf{a}}^{i}\right\}^{T}\mathbf{d}_{\{x\}} + \left\{\mathbf{a}^{i}\right\}^{T}\overset{\diamond}{\mathbf{d}}_{\{x\}} \\
&= \left[\left\{\dot{\mathbf{a}}^{i}\right\}^{T} + \left\{\mathbf{a}^{i}\right\}^{T}\mathbf{L}\right]\mathbf{d}_{\{x\}} + \left\{\mathbf{a}^{i}\right\}^{T}\left[\dot{\mathbf{d}}_{\{x\}} - \mathbf{L}\mathbf{d}_{\{x\}}\right] \\
&= \left\{\mathbf{a}^{i}\right\}^{T}\dot{\mathbf{d}}_{\{x\}} \qquad (6.0.3)
\end{aligned}$$

where $\overset{\diamond}{\mathbf{a}}^{i} = \dot{\mathbf{a}}^{i} + \mathbf{L}^{T}\mathbf{a}^{i}$ (as $\mathbf{a}^{i}$ are covectors), $\overset{\diamond}{\mathbf{d}}_{\{x\}} = \dot{\mathbf{d}}_{\{x\}} - \mathbf{L}\mathbf{d}_{\{x\}}$ (as $\mathbf{d}_{\{x\}}$ is a vector), and $\left\{\dot{\mathbf{a}}^{i}\right\}^{T} = \mathbf{0}$ by definition ($\mathbf{a}_{i}$ and hence $\mathbf{a}^{i}$ do not explicitly depend on t). It might be noted that $\mathbf{a}^{i}$ are interpreted as dual vectors, also because their action on $\mathbf{d}_{\{x\}}$ results in scalar components of $\mathbf{d}_{\{\alpha\}}$. We recall, that the notion of Lie derivative was briefly described in Section 3.2.1. It is not difficult to see that $\overset{\diamond}{\mathbf{d}}_{\{\alpha\}}$ is an objective rate in the sense that rigid rotations of the coordinate system $\left\{x^{i}\right\}$ do not affect its components. Indeed $\left\{\mathbf{\Lambda}\mathbf{a}^{i}\right\}^{T}\mathbf{\Lambda}\dot{\mathbf{d}}_{\{x\}} = \left\{\mathbf{a}^{i}\right\}^{T}\mathbf{\Lambda}^{T}\mathbf{\Lambda}\dot{\mathbf{d}}_{\{x\}} = \left\{\mathbf{a}^{i}\right\}^{T}\dot{\mathbf{d}}_{\{x\}}$, where $\mathbf{\Lambda}$ is an orthogonal operator.

Admittedly, the frame-picking method from points 1-2 is somewhat simplistic. This results from the pragmatism related to computer implementation. The considered class of shapes (Chapter 2) and motions (Chapter 3) allows to discard the case of curved geometry. Furthermore, we wouldn't like to be constrained by a specific manner of selecting points $\mathbf{x}_1, \mathbf{x}_2$ and local frames $\left\{\alpha^{i}\right\}$. For example, classically in contact problems, points $\mathbf{x}_1$ and $\mathbf{x}_2$ are related through proximity mapping, which together with some curvilinear structure gives rise to the local basis $\mathbf{a}_i$. The curvilinear structure needs not to be locally Euclidean and hence $\mathbf{a}_i$ is not necessarily orthonormal (in the tangent plane). This can be of use for anisotropic problems. In our applications it will be usually enough to use an orthonormal base. It might be useful to notice that equation (6.0.3) can be rephrased as

$$\{\mathbf{a}_i\}\,\overset{\diamond}{\mathbf{d}}_{\{\alpha\}} = \dot{\mathbf{d}}_{\{x\}} \qquad (6.0.4)$$

and, after left multiplying by $\{\mathbf{a}_i\}^{T}$, again expressed as

$$\overset{\diamond}{\mathbf{d}}_{\{\alpha\}} = \mathcal{A}^{-1}\{\mathbf{a}_i\}^{T}\dot{\mathbf{d}}_{\{x\}} \qquad (6.0.5)$$

where $\mathcal{A} = \{\mathbf{a}_i\}^{T}\{\mathbf{a}_i\}$ is the metric tensor. The last equation parallels formula (4.21) from Wriggers [**211**, p. 64], provided $\mathbf{d}_{\{x\}} = \mathbf{0}$ at t. This corresponds to the "zero gap" case in a contact problem. In the predominately dynamic framework presented in this thesis, and not resorting solely to penalisation, we try to avoid "gaps". This will be further commented on in Chapters 9, 10 and 11 dealing with the formulation and solution of the contact problem.

In this context, one should mention the paper by Laursen [**130**] where a specific treatment of the convected local description is developed. Similarly as in [**211**], the author assumes that the material point corresponding to $\mathbf{x}_2$ is chosen once and for all, and this in turn allows to select $\mathbf{x}_1$ at any given time. Paths of $\mathbf{x}_1$ on the surface $\partial\mathcal{B}_1$ are recognised as integral curves of an abstract flow, with respect to which one can take required derivatives. The author favours the material description, which is a minor nuance. The major conceptual difference is that, whereas in [**130**] the observer travels over the body $\mathcal{B}_1$ chasing the shadow of $\mathbf{x}_2$, in our case the observer catches $\mathbf{x}_2$ red-handed (usually there will hold $\mathbf{x}_1 = \mathbf{x}_2$ at t) and then watches its escape, while staying at $\mathbf{x}_1$.

6.1. From generalised to local velocities

Let us rewrite the motion in the general form

$$\mathbf{x}(\mathbf{X}, t) = \chi(\mathbf{X}, \mathbf{q}(t)) \tag{6.1.1}$$

where $\mathbf{x}$ is the spatial point, $\mathbf{X}$ is the referential point, and $\mathbf{q}$ is the configuration. One can compute the material velocity

$$\dot{\mathbf{x}}(\mathbf{X}, t) = \frac{\partial \chi(\mathbf{X}, \mathbf{q}(t))}{\partial \mathbf{q}} \mathbf{u}(t) \tag{6.1.2}$$

The components of $\mathbf{x}$ and $\dot{\mathbf{x}}$ are expressed in the spatial coordinate system $\{x^i\}$. After the preliminary discussion, it is not difficult to express the velocity $\dot{\mathbf{x}}$ in a local frame $\{\alpha^i\}$, with dual base $\mathbf{a}^i$. Namely

$$\mathbf{U} = \{\mathbf{a}^i\}^T \frac{\partial \chi(\mathbf{X}, \mathbf{q}(t))}{\partial \mathbf{q}} \mathbf{u} \tag{6.1.3}$$

where $\mathbf{U}$ comprises the components of the *local velocity* of the spatial point $\mathbf{x}$, with respect to the base $\mathbf{a}_i$. This can be rephrased as

$$\mathbf{U} = \mathbf{H}\mathbf{u} \tag{6.1.4}$$

where

$$\mathbf{H} = \{\mathbf{a}^i\}^T \frac{\partial \chi(\mathbf{X}, \mathbf{q}(t))}{\partial \mathbf{q}} \tag{6.1.5}$$

is a linear operator, acting between the spaces of generalised and local velocities $\mathbf{H} : T\mathcal{Q} \to TE^3$. The operator $\mathbf{H}$ takes a specific form, depending on the underlying kinematic model.

6.2. Rigid kinematics

For rigid bodies, there holds

$$\dot{\mathbf{x}} = \mathbf{\Lambda}\hat{\mathbf{\Omega}}(\mathbf{X} - \bar{\mathbf{X}}) + \dot{\bar{\mathbf{x}}} \tag{6.2.1}$$

$$\mathbf{u} = \begin{bmatrix} \mathbf{\Omega} \\ \dot{\bar{\mathbf{x}}} \end{bmatrix} \tag{6.2.2}$$

and hence

$$\mathbf{H} = \{\mathbf{a}^i\}^T \begin{bmatrix} \mathbf{\Lambda}\left(\hat{\bar{\mathbf{X}}} - \hat{\mathbf{X}}\right) & \mathbf{I} \end{bmatrix} \tag{6.2.3}$$

because

$$\hat{\mathbf{\Omega}}(\mathbf{X} - \bar{\mathbf{X}}) = \mathbf{\Omega} \times (\mathbf{X} - \bar{\mathbf{X}}) = (\bar{\mathbf{X}} - \mathbf{X}) \times \mathbf{\Omega} = \left(\hat{\bar{\mathbf{X}}} - \hat{\mathbf{X}}\right)\mathbf{\Omega} \tag{6.2.4}$$

Above, $\mathbf{I}$ is the 3×3 identity.

6.3. Pseudo-rigid kinematics

For pseudo-rigid bodies, there holds

$$\dot{\mathbf{x}} = \dot{\mathbf{F}}\left(\mathbf{X} - \bar{\mathbf{X}}\right) + \dot{\bar{\mathbf{x}}} \tag{6.3.1}$$

$$\mathbf{u} = \begin{bmatrix} \dot{F}_{11} \\ \dot{F}_{12} \\ \cdots \\ \dot{\bar{\mathbf{x}}} \end{bmatrix} \tag{6.3.2}$$

and hence

$$\mathbf{H} = \left\{\mathbf{a}^i\right\}^T \begin{bmatrix} \mathbf{X}^T - \bar{\mathbf{X}}^T & & & 1 & \\ & \mathbf{X}^T - \bar{\mathbf{X}}^T & & & 1 \\ & & \mathbf{X}^T - \bar{\mathbf{X}}^T & & & 1 \end{bmatrix} \tag{6.3.3}$$

because

$$\dot{\mathbf{F}}\left(\mathbf{X} - \bar{\mathbf{X}}\right) = \dot{F}_{ij}\left(X_j - \bar{X}_j\right) \tag{6.3.4}$$

6.4. Dynamics and quasi-statics

For dynamics, when the time integration is executed from a known step t to an unknown $t + h$, it is further assumed that evaluation of $\mathbf{H}$ involves

$$\left\{\mathbf{a}^i\right\}^T = \left\{\mathbf{a}^i\right\}^T \left(\mathbf{q}^t + \frac{h}{2}\mathbf{u}^t\right) \tag{6.4.1}$$

Similarly, for quasi-statics there holds

$$\left\{\mathbf{a}^i\right\}^T = \left\{\mathbf{a}^i\right\}^T \left(\mathbf{q}^t\right) \tag{6.4.2}$$

CHAPTER 7

Local dynamics

Let us consider the following function

$$L(\mathbf{u}) = \frac{1}{2}\langle \mathbf{M}\mathbf{u}, \mathbf{u}\rangle - \langle \mathbf{b}, \mathbf{u}\rangle \tag{7.0.3}$$

where

$$\mathbf{u} = \mathbf{u}^{t+h} \tag{7.0.4}$$

$$\mathbf{b} = h\mathbf{f}^{t+\frac{h}{2}} + \mathbf{M}\mathbf{u}^{t} \tag{7.0.5}$$

The velocity update of the dynamic time-stepping given in Chapter 5 can now be expressed as

$$\frac{\partial L}{\partial \mathbf{u}} = \mathbf{0} \tag{7.0.6}$$

The unknown velocity $\mathbf{u}$ is obtained as a stationary point of L and hence, in a sense, L can be regarded as a discrete *Lagrangian* of our mechanical system. From the geometrical point of view L is a strictly convex function, $L(\lambda\mathbf{u}_1 + (1-\lambda)\mathbf{u}_2) < \lambda L(\mathbf{u}_1) + (1-\lambda)L(\mathbf{u}_2)$ for any $\mathbf{u}_1$, $\mathbf{u}_2$ and $\lambda \in (0,1)$, which follows from the positive definiteness of $\mathbf{M}$, preventing the graph of L from having linear slopes. The stationary point in (7.0.6) is then unique. Such a wrapping of the time integration formula might seem somewhat overblown. Nevertheless, it allows us to put the formulation of local dynamics into a broader context of duality. In the first place, it is of use to interpret the case

$$\frac{\partial L}{\partial \mathbf{u}} = \mathbf{r} \neq \mathbf{0} \tag{7.0.7}$$

A velocity update formula derived from the above condition adds $\mathbf{r}$ on the right hand side

$$\mathbf{M}\mathbf{u} = \mathbf{b} + \mathbf{r} \tag{7.0.8}$$

Although $\mathbf{r}$ cannot be readily interpreted as a force at a particular time t, it is correct to view it as an integral of some force over the time interval $[t, t+h]$

$$\mathbf{r} = \int_t^{t+h} d\mathbf{r} \tag{7.0.9}$$

If $\mathcal{Q}$ is the configuration space of the mechanical system, it is then easy to see that $\mathbf{u} \in T\mathcal{Q}$, while $\mathbf{b}, \mathbf{r} \in T^*\mathcal{Q}$ and $\mathbf{M} : T\mathcal{Q} \to T^*\mathcal{Q}$. In the previous chapter, the mapping $\mathbf{H}$ was defined

$$\mathbf{U} = \mathbf{H}\mathbf{u} \tag{7.0.10}$$

acting between the spaces of generalised and local velocities $\mathbf{H} : T\mathcal{Q} \to TE^3$. Rows of $\mathbf{H}$ can be interpreted as elements of the generalised force space $T^*\mathcal{Q}$ and hence the transpose map $\mathbf{H}^T$ acts between the spaces of local and generalised forces $\mathbf{H}^T : T^*E^3 \to T^*\mathcal{Q}$. This is also seen from the duality pairing between local and global variables (power conjugacy)

$$\langle \mathbf{U}, \mathbf{R} \rangle = \langle \mathbf{H}\mathbf{u}, \mathbf{R} \rangle = \left\langle \mathbf{u}, \mathbf{H}^T\mathbf{R} \right\rangle \tag{7.0.11}$$

where $\mathbf{R} \in T^*E^3$ is by definition a *local net force* over $[t, t+h]$. Every local force $\mathbf{R}$ corresponds then to some generalised force $\mathbf{r}$

$$\mathbf{r} = \mathbf{H}^T\mathbf{R} \tag{7.0.12}$$

Note, that while $\mathbf{H}$ is a surjection, $\mathbf{H}^T$ happens to be an injection, covering only a subset of $T^*\mathcal{Q}$. Nevertheless, for each $\mathbf{R}$ we can obtain a corresponding $\mathbf{u}$ as a solution of

$$\frac{\partial L}{\partial \mathbf{u}} = \mathbf{H}^T\mathbf{R} \tag{7.0.13}$$

which corresponds to the maximum, with respect to $\mathbf{u}$, of the following saddle function

$$G(\mathbf{u}, \mathbf{R}) = \left\langle \mathbf{u}, \mathbf{H}^T\mathbf{R} \right\rangle - L(\mathbf{u}) \tag{7.0.14}$$

The maximum can be computed as

$$\mathbf{u}(\mathbf{R}) = \mathbf{M}^{-1}\left(\mathbf{b} + \mathbf{H}^T\mathbf{R}\right) \tag{7.0.15}$$

and plunged back into G, resulting in

$$\begin{aligned} L^*_{\mathbf{H}}(\mathbf{R}) &= G(\mathbf{u}(\mathbf{R}), \mathbf{R}) \\ &= \frac{1}{2}\langle \mathbf{W}\mathbf{R}, \mathbf{R} \rangle + \langle \mathbf{B}, \mathbf{R} \rangle + b \end{aligned} \tag{7.0.16}$$

where

$$\mathbf{W} = \mathbf{H}\mathbf{M}^{-1}\mathbf{H}^T \tag{7.0.17}$$

$$\mathbf{B} = \mathbf{H}\mathbf{M}^{-1}\mathbf{b} \tag{7.0.18}$$

$$b = \frac{1}{2}\left\langle \mathbf{M}^{-1}\mathbf{b}, \mathbf{b} \right\rangle \tag{7.0.19}$$

$L^*_{\mathbf{H}}$ is the *local conjugate* function of L, defined by the Legendre-Fenchel transform

$$L^*_{\mathbf{H}}(\mathbf{R}) = \sup_{\mathbf{u}} \{\langle \mathbf{H}\mathbf{u}, \mathbf{R} \rangle - L(\mathbf{u})\} \tag{7.0.20}$$

We call it *local* and index with $\mathbf{H}$, as it corresponds to the duality between the local variables $\mathbf{U}$ and $\mathbf{R}$, related to their generalised counterparts through $\mathbf{H}$. In general,

a *conjugate* function of a convex function L reads (cf. Rockafellar and Wets [**183**, p. 473-474])

$$L^*(\mathbf{r}) = \sup_{\mathbf{u}} \{\langle \mathbf{u}, \mathbf{r}\rangle - L(\mathbf{u})\} \tag{7.0.21}$$

which in our case takes the form

$$L^*(\mathbf{r}) = \frac{1}{2}\left\langle \mathbf{M}^{-1}\mathbf{r}, \mathbf{r}\right\rangle + \left\langle \mathbf{M}^{-1}\mathbf{b}, \mathbf{r}\right\rangle + b \tag{7.0.22}$$

Clearly

$$L^*_{\mathbf{H}}(\mathbf{R}) = L^*\left(\mathbf{H}^T\mathbf{R}\right) \tag{7.0.23}$$

so that $L^*_{\mathbf{H}}$ corresponds to the restriction of L^* to a domain generated by the row space of $\mathbf{H}$. The gradient of $L^*_{\mathbf{H}}$ reads

$$\frac{\partial L^*_{\mathbf{H}}}{\partial \mathbf{R}} = \mathbf{W}\mathbf{R} + \mathbf{B} \tag{7.0.24}$$

and from the algebraic structure of $\mathbf{W}$ and $\mathbf{B}$ it is seen that it corresponds to some local velocity

$$\mathbf{U} = \mathbf{W}\mathbf{R} + \mathbf{B} \tag{7.0.25}$$

As it can be deduced from (7.0.7) and (7.0.22), while the gradient of L at $\mathbf{u}$ is $\mathbf{r}$, the gradient of L^* at $\mathbf{r}$ is $\mathbf{u}$, which is characteristic for conjugate functions. It should be noted, that a conjugate function L^* is always convex. In our case, it is strictly convex as the eigenvalues of $\mathbf{M}^{-1}$ are positive and bounded away from zero. The local conjugate $L^*_{\mathbf{H}}$ might or might not be strictly convex, depending on the particular shape of the $\mathbf{H}$ mapping. This issue becomes clear, when more than one local frame is considered. It is relevant to mention, that as for $\mathbf{R} = \mathbf{0}$ there follows $\mathbf{U} = \mathbf{B}$, vector $\mathbf{B}$ is sometimes called the *local free velocity*.

7.1. Many bodies and local frames

Let $\{\mathcal{B}_i\}$ be a set of bodies and $\{\mathcal{C}_\alpha\}$ be a set of local frames. To each local frame $\mathcal{C}_\alpha$ there corresponds a pair of bodies $\mathcal{B}_i$ and $\mathcal{B}_j$. Let $\mathcal{B}_j$ be the body, to which the local frame is attached. $\mathcal{B}_j$ will be called the *master* in $\mathcal{C}_\alpha$ and denoted by $\mathcal{M}_\alpha$. Consequently, $\mathcal{B}_i$ will be called the *slave* in $\mathcal{C}_\alpha$ and denoted by $\mathcal{S}_\alpha$. Of course, the choice is arbitrary. Considering evolution of a multi-body system over an interval $[t, t+h]$, an analogue of equation (7.0.25) can be written down for each of the local frames

$$\mathbf{U}_\alpha = \mathbf{B}_\alpha + \sum_\beta \mathbf{W}_{\alpha\beta}\mathbf{R}_\beta \tag{7.1.1}$$

where

$$\mathbf{U}_\alpha = \mathbf{H}_{i\alpha}\mathbf{u}_i - \mathbf{H}_{j\alpha}\mathbf{u}_j \tag{7.1.2}$$

$$\mathbf{B}_\alpha = \mathbf{H}_{i\alpha}\mathbf{M}_i^{-1}\mathbf{b}_i - \mathbf{H}_{j\alpha}\mathbf{M}_j^{-1}\mathbf{b}_j \tag{7.1.3}$$

$$\mathbf{W}_{\alpha\beta}|_{\alpha\neq\beta} = s_{\alpha\beta}\mathbf{H}_{k_\beta\alpha}\mathbf{M}_{k_\beta}^{-1}\mathbf{H}_{k_\beta\beta}^T \tag{7.1.4}$$

$$\mathbf{W}_{\alpha\alpha} = \mathbf{H}_{i\alpha}\mathbf{M}_i^{-1}\mathbf{H}_{i\alpha}^T + \mathbf{H}_{j\alpha}\mathbf{M}_j^{-1}\mathbf{H}_{j\alpha}^T \tag{7.1.5}$$

$$k_\beta = \begin{cases} i & \text{if } \mathcal{B}_i \in \mathcal{C}_\beta \\ j & \text{if } \mathcal{B}_j \in \mathcal{C}_\beta \end{cases} \tag{7.1.6}$$

$$s_{\alpha\beta} = \begin{cases} -1 & \text{if } \mathcal{B}_{k_\beta} \text{ is } (\mathcal{M}_\alpha \wedge \mathcal{S}_\beta) \vee (\mathcal{S}_\alpha \wedge \mathcal{M}_\beta) \\ 1 & \text{otherwise} \end{cases} \tag{7.1.7}$$

The above formulae can be conveniently applied in a computer implementation. They stem from the following algebra of the multi-body dynamics. Let $\mathbf{q}$, $\mathbf{u}$, $\mathbf{f}$, $\mathbf{M}$ gather the suitable vectors and matrices as

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \dots \\ \mathbf{q}_n \end{bmatrix}, \mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \dots \\ \mathbf{u}_n \end{bmatrix}, \mathbf{f} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \dots \\ \mathbf{f}_n \end{bmatrix}, \mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & & & \\ & \mathbf{M}_2 & & \\ & & \dots & \\ & & & \mathbf{M}_n \end{bmatrix} \tag{7.1.8}$$

To each local frame $\mathcal{C}_\alpha$, there corresponds a block-row of the global $\mathbf{H}$ operator

$$\mathbf{H} = \begin{bmatrix} \dots & -\mathbf{H}_{j1} & \dots & \mathbf{H}_{i1} & \dots & & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & \dots & \mathbf{H}_{i\alpha} & \dots & -\mathbf{H}_{j\alpha} & \dots & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & & \dots & \mathbf{H}_{im} & \dots & -\mathbf{H}_{jm} & \dots \end{bmatrix} \tag{7.1.9}$$

where

$$\mathbf{H}_{k\alpha} = \mathbf{H}\left(\{\mathbf{a}^i\} \in \mathcal{C}_\alpha, \mathbf{X} \in \mathcal{B}_k\right) \tag{7.1.10}$$

is evaluated according to the formula (6.1.5) of the previous chapter. All of the derivations from the introductory section of this chapter apply without change and lead to the formulae (7.1.1-7.1.7). From now on a distinction between the single-body or the multi-body as well as between the single-frame or the multi-frame cases will be made only, if it is not clear from the context.

Operator $\mathbf{W}$ maps local covariant quantities into the contravariant ones. Algebraically, it is represented by a sparse matrix, composed of dense 3×3 blocks $\mathbf{W}_{\alpha\beta}$. The sparsity pattern of $\mathbf{W}$ corresponds to the vertex connectivity in the graph of local frames. Vertices of this graph are the local frames $\{\mathcal{C}_\alpha\}$, while the edges comprise a subset of all bodies $\{\mathcal{B}_i\}$, such that $\mathcal{B}_i \in \mathcal{C}_\alpha$ and $\mathcal{B}_i \in \mathcal{C}_\beta$ for $\alpha \neq \beta$. This has been illustrated in Figure 7.1.1. Operator $\mathbf{W}$ derives from the formula

$$\mathbf{W} = \mathbf{H}\mathbf{M}^{-1}\mathbf{H}^T \tag{7.1.11}$$

where $\mathbf{M}$ is a $n \times n$ symmetric and positive definite matrix, and $\mathbf{H}$ is an $m \times n$ transformation operator (m and n in (7.1.8) and (7.1.9) are respectively equal n/k and $m/3$ here, where k is the dimension of $T\mathcal{Q}$). Clearly, $\mathbf{W}$ is an $m \times m$ symmetric matrix. It is positive definite, provided rows of $\mathbf{H}$ are linearly independent. This is easiest to see from the flow of the actions in the above formula. A local force $\mathbf{R}$ is first mapped by $\mathbf{H}^T$ into a generalised force $\mathbf{r}$. If rows of $\mathbf{H}$ are not linearly independent, then there exist $\mathbf{R}_1 \neq \mathbf{R}_2$ such that $\mathbf{H}^T\mathbf{R}_1 = \mathbf{H}^T\mathbf{R}_2$ and hence $\mathbf{W}$ fails to be a bijection. This means, that the null space of $\mathbf{W}$ is larger than $\{\mathbf{0}\}$, so that it is not invertible in the usual sense. $\mathbf{W}$ becomes singular whenever $m > n$, which is trivially related to the number of considered bodies. On the other

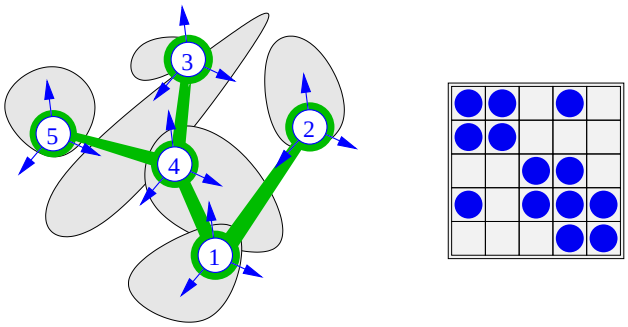


FIGURE 7.1.1. A graph of local frames and the corresponding pattern of $\mathbf{W}$.

hand, one can always introduce singularity of $\mathbf{W}$ by using local frames between the same pair of bodies, whose $\mathbf{H}$ operators are linearly dependent. This can be easily related to the deformability of our kinematic models. For example, the pseudo-rigid body has a linear distribution of the instantaneous velocity over an arbitrary flat surface. Thus, the relative velocity between two bodies over a flat surface is fully parametrised by three points. A larger number of local frames results in the singularity of $\mathbf{W}$. So does their collinearity. One can then generally speak about the global and local over-restraining of the system. In practice, $\mathbf{W}$ often becomes numerically singular for some particular configurations of local frames. Indeterminacy of local forces is then an unavoidable consequence of the kinematic simplicity, and as so it needs to be accepted in numerical practise. It might be noted, that semi-positive definiteness of $\mathbf{W}$ implies non-strict convexity of L^*_H.

7.2. Constraints

As it was already mentioned, L is strictly convex and hence, it admits a unique minimum at a root of its gradient. Formula 7.0.6 describes this case, which by construction corresponds to the numerical integration of an *unconstrained* motion. It is not hard to guess however, that the local dynamical equation (7.0.25) was introduced here in order to bring into the picture the notion of *constraints*. Within the current formulation, these will be phrased in the form of some local equalities

$$\mathbf{C}(\mathbf{U}, \mathbf{R}) = \mathbf{0} \tag{7.2.1}$$

which, combined with (7.0.25), result in

$$\mathbf{C}(\mathbf{W}\mathbf{R} + \mathbf{B}, \mathbf{R}) = \mathbf{0} \tag{7.2.2}$$

The above is an implicit, nonlinear and usually nonsmooth equation, numerically solved for $\mathbf{R}$. In some particular cases, it does correspond to the solution of a constrained minimisation problem. More often however, it can only be viewed as the root finding problem. These subtleties will become clearer, when particular kinds of constraints are introduced in Chapters 8 and 10.

7.3. Quasi-statics

Take

$$\mathbf{M} = \mathbf{A} \tag{7.3.1}$$

$$\mathbf{b} = h\mathbf{f}\left(t+h, \mathbf{q}^t\right) \tag{7.3.2}$$

where $\mathbf{A}$ was as (5.3.3) in Section 5.3. The foregoing discussion applies without further changes.

CHAPTER 8

Joints

It is often necessary to confine the motion of a material point $\mathbf{X}$ within a prescribed manifold

$$\mathbf{x}(\mathbf{X}, t) \in \mathcal{C}(t) \tag{8.0.3}$$

This can be written down as an implicit equation

$$\mathbf{c}(\mathbf{x}, t) = \mathbf{0} \tag{8.0.4}$$

where $\mathbf{c}$ is a k-component vector function, with $1 \leq k \leq 3$. The Jacobian $[\partial \mathbf{c}/\partial \mathbf{x}]$ is then a $k \times 3$ matrix, of full rank for well defined constraints. In our case, in order to fit into the 3×3 block structure of the local dynamics equations, it would be convenient use a 3-component $\mathbf{c}$. This is easily achieved. Note that for each t, rows of $[\partial \mathbf{c}/\partial \mathbf{x}]$ can be interpreted as some covectors $\mathbf{a}^k$ in the Euclidean 3-space E^3, and

$$[\partial \mathbf{c}/\partial \mathbf{x}]\, \delta \mathbf{x} = \mathbf{0} \tag{8.0.5}$$

for all vectors $\delta \mathbf{x}$ in the tangent space $T_{\mathbf{x}}\mathcal{C}(t)$. Note also, that covectors $\mathbf{a}^k$ span the orthogonal complement space $T_{\mathbf{x}}^{\perp}\mathcal{C}(t)$ (Figure 8.0.1). For each $\mathbf{x}$ and t, one can always define $3-k$ covectors $\mathbf{a}^j$ $(k < j \leq 3)$, spanning $T_{\mathbf{x}}\mathcal{C}(t)$. This way a complete dual base $\{\mathbf{a}^i\}$ is defined in E^3. The constraint equation (8.0.4) can now be suitably extended to

$$\mathbf{c}(\mathbf{x}, t) = \begin{bmatrix} c^1(\mathbf{x}, t) \\ \langle \mathbf{a}^2, \mathbf{x} - \mathbf{x}(\mathbf{X}, t) \rangle \\ \langle \mathbf{a}^3, \mathbf{x} - \mathbf{x}(\mathbf{X}, t) \rangle \end{bmatrix}, \text{ when } \mathcal{C}(t) \text{ is a surface} \tag{8.0.6}$$

and to

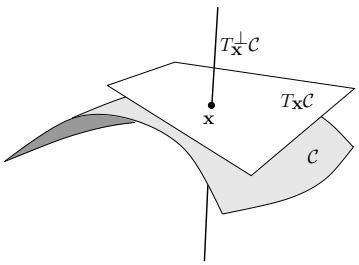


FIGURE 8.0.1. Geometrical interpretation of a constraint manifold $\mathcal{C}$, the tangent space $T_{\mathbf{x}}\mathcal{C}$ and the orthogonal complement $T_{\mathbf{x}}^{\perp}\mathcal{C}$.

$$\mathbf{c}(\mathbf{x},t)=\begin{bmatrix} c^1(\mathbf{x},t) \\ c^2(\mathbf{x},t) \\ \langle \mathbf{a}^3, \mathbf{x}-\mathbf{x}(\mathbf{X},t)\rangle \end{bmatrix}, \text{ when } \mathcal{C}(t) \text{ is a curve.} \tag{8.0.7}$$

The velocity form of so extended (8.0.4) reads now

$$[\partial \mathbf{c}/\partial \mathbf{x}]\,\dot{\mathbf{x}}+\dot{\mathbf{c}}=\mathbf{0} \tag{8.0.8}$$

where the last j equations are satisfied by construction. This somewhat redundant derivation was made here in order to rewrite (8.0.8) as

$$\left\{\mathbf{a}^i\right\}^T\dot{\mathbf{x}}=-\dot{\mathbf{c}} \quad \text{or} \quad \mathbf{U}(t)=-\dot{\mathbf{c}}(t) \tag{8.0.9}$$

where $\mathbf{U}$ is a local velocity with respect to the base $\{\mathbf{a}_i\}$. Hence, a point constraint of form (8.0.4) naturally defines a local frame.

8.1. Back to the discrete case

Resorting back to the time-stepping scheme and the local dynamics of the previous chapter, it is seen that the integration of motion constrained in the above manner can be stated as

$$\begin{array}{c} \min_{\mathbf{u}} L(\mathbf{u}) \\ \mathbf{H}\mathbf{u}+\dot{\mathbf{c}}=\mathbf{0} \end{array} \tag{8.1.1}$$

where by convention $\mathbf{u}=\mathbf{u}^{t+h}$ and $\dot{\mathbf{c}}=\dot{\mathbf{c}}(t+h)$. The local force $\mathbf{R}$ can now be interpreted as a vector of Lagrange multipliers corresponding to the affine constraints $\mathbf{H}\mathbf{u}+\dot{\mathbf{c}}=\mathbf{0}$. The Lagrangian of (8.1.1) reads

$$L_{\mathbf{c}}(\mathbf{u})=L(\mathbf{u})-\langle \mathbf{H}\mathbf{u}+\dot{\mathbf{c}},\mathbf{R}\rangle \tag{8.1.2}$$

and its optimality conditions lead to the saddle point problem

$$\begin{bmatrix} \mathbf{M} & -\mathbf{H}^T \\ \mathbf{H} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \mathbf{u} \\ \mathbf{R} \end{bmatrix}=\begin{bmatrix} \mathbf{b} \\ -\dot{\mathbf{c}} \end{bmatrix} \tag{8.1.3}$$

This can be further transformed into

$$-\dot{\mathbf{c}}=\mathbf{H}\mathbf{M}^{-1}\mathbf{H}^T\mathbf{R}+\mathbf{H}\mathbf{M}^{-1}\mathbf{b} \tag{8.1.4}$$

or

$$-\dot{\mathbf{c}}=\mathbf{W}\mathbf{R}+\mathbf{B} \tag{8.1.5}$$

Because of the redundant components in the constraints of type (8.0.6-8.0.7) and due to (7.0.25) and (8.0.9), some of the rows in the above system might be identities. The motion along the corresponding directions $\mathbf{a}_i$ should not be constrained, and hence these rows are replaced by diagonal terms $R_i=0$. This leads to the following uniform notation for a constraint equation, specified at some point $\mathbf{x}$

$$\mathbf{C}(\mathbf{U},\mathbf{R})=\mathbf{0} \tag{8.1.6}$$

where $\mathbf{C}$ is a 3-component vector function

$$C_i(\mathbf{U},\mathbf{R}) \quad = \quad \begin{cases} U^i+\dot{c}^i & \text{for} \quad \mathbf{a}_i \in T_{\mathbf{x}}^{\perp}\mathcal{C}^{t+h} \\ R_i & \text{for} \quad \mathbf{a}^i \in T_{\mathbf{x}}\mathcal{C}^{t+h} \end{cases} \tag{8.1.7}$$

Symbolically, (8.1.6) describes also the system of $3m$ equations for a multi-body system with m joints. In order to find the constraint reactions $\mathbf{R}$, one then solves the system

$$\mathbf{C}\left(\mathbf{WR}+\mathbf{B},\mathbf{R}\right)=\mathbf{0} \tag{8.1.8}$$

where U^i are expressed by the suitable rows of $\mathbf{U}=\mathbf{WR}+\mathbf{B}$.

8.2. Single-body joints

Let a *dummy* body $\mathcal{B}_i$ be defined as follows

$$\mathbf{q}_i\left(t\right)=\mathbf{q}_i\left(0\right) \tag{8.2.1}$$

$$\mathbf{u}_i\left(t\right)=\mathbf{0} \tag{8.2.2}$$

$$\mathbf{M}_i^{-1}=\mathbf{0} \tag{8.2.3}$$

so it does not move and has a zero inverse inertia operator. One can use a dummy body in order to introduce a single-body constraint $\mathcal{C}_\alpha$ within the framework of local dynamics. For this, one picks a body of interest $\mathcal{B}_j$ and attaches to it a local frame. Together with the dummy body $\mathcal{B}_i$, this allows to formulate a block-row of the $\mathbf{U}=\mathbf{WR}+\mathbf{B}$ relation. Assumption (8.2.2) implies, that in the absence of other constraints, the relative local velocity $\mathbf{U}_\alpha$ is solely due to the motion of $\mathcal{B}_j$. The following assumption (8.2.3) allows to reuse the same dummy body in order to impose other constraints. The zero inverse inertia operator breaks the off-diagonal couplings in the block-row $\mathbf{W}_{\alpha\beta}$, so that the reactions of other constraints using $\mathcal{B}_i$ do not contribute to $\mathbf{U}_\alpha$. In other words, a dummy body does not correspond to an edge in the graph of local frames.

Assume that $\mathbf{X}$ is a referential point, to which the local frame with base $\{\mathbf{a}_i\}$ is attached at $t=0$. In the view of (8.1.7), it is not difficult to come up with several typical constraints

Fixed point. Motion of $\mathbf{X}$ is precluded.	$\mathbf{C}\left(\mathbf{U},\mathbf{R}\right)=\mathbf{U}$
Fixed line. Motion of $\mathbf{X}$ is allowed along a line aligned with $\mathbf{a}_3$ and passing through $\mathbf{X}$ at $t=0$.	$\mathbf{C}\left(\mathbf{U},\mathbf{R}\right)=\begin{bmatrix} U_1 \\ U_2 \\ R_3 \end{bmatrix}$
Fixed plane. Motion of $\mathbf{X}$ is allowed within a plane normal to $\mathbf{a}_3$ and passing through $\mathbf{X}$ at $t=0$.	$\mathbf{C}\left(\mathbf{U},\mathbf{R}\right)=\begin{bmatrix} R_1 \\ R_2 \\ U_3 \end{bmatrix}$
Prescribed velocity. The local velocity of $\mathbf{X}$ reads $\mathbf{V}\left(t\right)$.	$\mathbf{C}\left(\mathbf{U},\mathbf{R}\right)=\mathbf{U}-\mathbf{V}$

One important subtlety needs to be mentioned for the dynamic time stepping. As the configuration update is of the kind

$$\mathbf{q}^{t+h} = \mathbf{q}^t + \frac{h}{2}\left(\mathbf{u}^t + \mathbf{u}^{t+h}\right) \tag{8.2.4}$$

and the above constraint definitions correspond to the velocity $\mathbf{u}^{t+h}$, an $O(h)$ violation of the constraint is possible between $t = 0$ to $t = h$. If the constraints cannot be prescribed in a way, which prevents the local velocities from having components along the orthogonal complement space $T^{\perp}\mathcal{C}$, it is possible to enforce this condition by solving

$$\mathbf{C}(\mathbf{WR} + \mathbf{B}, \mathbf{R}) = \mathbf{0} \tag{8.2.5}$$

at $t = 0$, followed by the update of velocity

$$\mathbf{u}^0 = \mathbf{u}^0 + \mathbf{M}^{-1}\mathbf{H}^T\mathbf{R} \tag{8.2.6}$$

8.3. Multi-body joints

Joints between pairs of bodies can be defined in a natural manner. The discussion of the previous section applies without changes, although the dummy body needs to be replaced by a regular one. For example, the *fixed point* constraint can now be reinterpreted as a *spherical joint*, as soon as the referential points $\mathbf{X} \in \mathcal{B}_i$ and $\mathbf{Y} \in \mathcal{B}_j$ are assumed to coincide at $t = 0$. As an example of a more elaborate constraint, let us consider a *rigid weightless rod*, inserted between an arbitrary pair of points $\mathbf{X} \in \mathcal{B}_i$, $\mathbf{Y} \in \mathcal{B}_j$ at $t = 0$. The rigid rod constraint corresponds to the below statement

$$\|\mathbf{x}_i(\mathbf{X}, t) - \mathbf{x}_j(\mathbf{Y}, t)\| = \|\mathbf{X} - \mathbf{Y}\| \tag{8.3.1}$$

Let us define the dual base vector $\mathbf{a}^1$ as

$$\mathbf{a}^1(t) = \left[\mathbf{x}_i(\mathbf{X}, t) - \mathbf{x}_j(\mathbf{Y}, t)\right] / \|\mathbf{x}_i(\mathbf{X}, t) - \mathbf{x}_j(\mathbf{Y}, t)\| \tag{8.3.2}$$

and select the remaining covectors $\left\{\mathbf{a}^2, \mathbf{a}^3\right\} \perp \mathbf{a}^1$. With this definition of the local frame, the rod constraint can be expressed as

$$\mathbf{C}(\mathbf{U}, \mathbf{R}) = \begin{bmatrix} U_1 \\ R_2 \\ R_3 \end{bmatrix} \tag{8.3.3}$$

8.4. Configuration space

A multi-body system without constraints has freedom to move inside of its configuration space $\mathcal{Q}$. Enforcement of some *equality constraints* (joints) reduces this space to a subset of $\mathcal{Q}$. In the following this fact will be implicitly acknowledged. Nevertheless, from the point of view of our implementation it is more convenient to think about $\mathcal{Q}$ as intact. This is because all of the constraints will be dealt with in a uniform manner, and no formal reduction of the configuration space will be performed. At times, it will be convenient though to re-frame our thinking and treat some of the bodies as "moving boundaries". This will be emphasised by writing $\mathcal{Q}(t)$.

CHAPTER 9

Contact points

Bodies never come into contact at a single point. At some level of observation, one can usually speak about a smooth contact surface. Yet, from the computational point of view it is convenient to consider instead the set of "oriented points" (Figure 9.0.1). Here the contact corresponds to a point and a normal direction attached to it. It is customary to refer by the single notion of "contact" to the totality of entities attached to a contact point. The multi-body framework has to cope with identification and maintenance of a representative set of contacts. Optimally, the cost of those activities should be comparable with the one pertinent to other essential computations (e.g. the time stepping). In order to accomplish this goal, it is necessary to resort to some of the methods studied within the field of computer science. This requires a temporary departure from mechanics into the realm of algorithms and data structures.

Let the set of bodies $\{\mathcal{B}_i\}$ be called a configuration. Let $\{\mathcal{B}_i\}^t, \{\mathcal{B}_i\}^{t+h}$ be two consecutive configurations, possibly admitting small interpenetrations (the time indexing is used at convenience). The motion affects shapes and positions of bodies in $\{\mathcal{B}_i\}$. Additional operations cause structural changes to $\{\mathcal{B}_i\}$ (e.g. insertion or deletion of bodies). Let the tuple $\mathcal{C}_\alpha = (\mathbf{x}, \mathbf{n}, \mathcal{B}_i, \mathcal{B}_j)_\alpha$ store the point, the normal direction and the pairing of bodies involved in a contact. The goal is to efficiently maintain $\{\mathcal{C}_\alpha\}$, under possible changes of $\{\mathcal{B}_i\}$.

What precisely *efficiently* means, will be the matter of discussion in Section 9.2. Before that, Section 9.1 introduces a number of auxiliary data structures, setting the background for the forthcoming developments. Section 9.3 discusses the approximate contact search methods. Section 9.4 deals with the detection of contact points and normals. Brief literature review follows in the last section.

9.1. Auxiliary data structures

The notion of a *data structure* will not be explicitly introduced. It will emerge as a result of presentation of a number of beings belonging to this category. Let us discuss some general properties instead. Data structures require *space* in order to store their elements. A basic question is how much space is required in order to store n elements? Structures are accompanied by *algorithms* operating on them. Another elementary question is then how much *time* is necessary for an algorithm

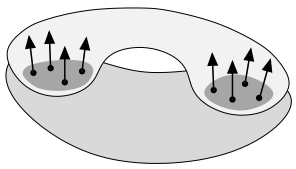


FIGURE 9.0.1. An admittedly telephone-like example of contact between two bodies. The set of oriented points represents two disjoint contact surfaces.

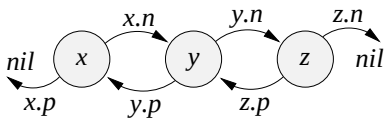


FIGURE 9.1.1. A doubly-linked list.

to accomplish its goal? Both issues can be briefly addressed as the space and the time *complexity*, and examined on a case by case basis. Without resorting to any particular example, it is adequate to recall some notations commonly employed in the analysis of algorithms. Let $g(x)$ be a known function. The growth of any function $f(x)$ can be related to the growth of $g(x)$ in a number of ways. Definitions below are quoted after Wilf **[209]**

DEFINITION 9.1.1. We say that $f(x) = O(g(x))$ $(x \to \infty)$ if $\exists C, x_0$ such that $|f(x)| < Cg(x)$ $(\forall x > x_0)$.

DEFINITION 9.1.2. We say that $f(x) = \Theta(g(x))$ if there are constants $c_1 > 0$, $c_2 > 0$, x_0 such that for all $x > x_0$ it is true that $c_1 g(x) < f(x) < c_2 g(x)$.

DEFINITION 9.1.3. We say that $f(x) = \Omega(g(x))$ if there is an $\epsilon > 0$ and a sequence $x_1, x_2, x_3, ... \to \infty$ such that $\forall_j : |f(x_j)| > \epsilon g(x_j)$.

Thus, $f(x) = O(g(x))$ implies that $f(x)$ grows no faster than $g(x)$, $f(x) = \Theta(g(x))$ states that $f(x)$ and $g(x)$ grow at the same rate, while $f(x) = \Omega(g(x))$ means that $f(x)$ grows at least at the rate of $g(x)$.

Considering an algorithm operating on n elements of a data structure, it is now easier to describe its space and time demands. In practice one is mostly interested in data structures with $O(n)$ space complexity. Fast algorithms will usually have runtime complexity similar to $O\left(n^a \log^b n\right)$, where $a, b \geq 0$ will depend on the state of the ordering and dimensionality of the input data. The runtime is measured by the number of discrete steps. In the following, unless specified otherwise, the logarithm to the base 2 is considered.

9.1.1. Tuple. Tuple is a grouping of elements. For example (b, c, d) is a tuple composed of elements (members) b, c, d. Any of the elements can be a tuple itself. Let $a = (b, c, d)$ be a variable storing the tuple. We can refer to the members of a by $a.b$, $a.c$, $a.d$.

9.1.2. Pointer. A pointer is a symbolic link to a tuple. For example let the tuple (d, n) be composed of an arbitrary data d, and a pointer n. Then it is fair to create a variable $a = (d, n)$ and assign the pointer value $a.n = a$. The member $a.n$ behaves now as it was a. For example $a.n.d$ is the same as $a.d$ and the infinite reference $a.n.n.n...$ makes sense. If the pointer value was not assigned, the default value *nil* is assumed. It is also valid to assign the *nil* value $a.n = nil$ explicitly. Note, that pointers do not stand out in the adopted notation. It is enough to mention them, when a tuple is being first defined.

9.1.3. List. Each item of a (doubly-linked) list is composed of three elements (d, p, n), where d stores an arbitrary data, p is a pointer to the previous list item, and n is a pointer to the next list item (Figure 9.1.1). There are as many list items as there are data items, so that the space demand of the list structure is $O(n)$. A list is represented by a pointer to the first element, say l. A data item d is inserted into the list l as follows

ALGORITHM 9.1.4. *List_ Insert* (l, d)
```
1  a = (d, p, n)
2  if l ≠ nil then l.p = a
3  a.n = l
4  l = a
```

It is seen that the newly created list item a replaces the head of the list l. Pointers are updated accordingly. The complexity of this operation is $O(1)$. Another elementary operation is deletion. Let us delete an item a from l

ALGORITHM 9.1.5. *List_ Delete* (l, a)
```
1  if a.p ≠ nill then a.p.n = a.n
2  else l = a.n
2  if a.n ≠ nil then a.n.p = a.p
```

The $O(1)$ deletion comprises obvious updates of pointers. The following routine finds a list item associated with a specific data d

ALGORITHM 9.1.6. *List_ Find_ Item* (l, d)
```
1  a = l
2  while a ≠ nil do
3     if a.d = d then
4        return a
5     end if
6     a = a.n
7  end while
8  return nil
```

As there is no other way to identify the list item storing d, the $O(n)$ search is necessary. Combining the two above algorithms allows to delete the list item associated with d

ALGORITHM 9.1.7. *List_ Delete_ Data* (l, d)
```
1  a = List_ Find_ Item (l, d)
2  if a ≠ nil then List_ Delete (l, a)
```

Somewhat more interesting code can be written down, once the order of data is taken into account. The classical *merge sort* algorithm can be implemented as follows

ALGORITHM 9.1.8. *List_ Merge_ Sort* (l)
```
1  o = 1
2  while true do
3     h = t = nil, j = l
4     while true do
5        i = j, m = 0
6        while m < o ∧ j ≠ nil do j = j.n, m = m + 1
7        k = j, n = 0
8        while n < o ∧ k ≠ nil do k = k.n, n = n + 1
9        if j = nil ∧ i = l then
10          for i = l until nil do i.p = j, j = i, i = i.n
11          return l
12       else if m + n = 0 break
13       if h = nil then if i.d ≤ j.d then h = i else h = j
14       while m > 0 ∧ n > 0 do
15          if i.d ≤ j.d then
16             if t ≠ nil then t.n = i
```

```
17            t = i, i = i.n, m = m − 1
18          else
19            if t ≠ nil then t.n = j
20            t = j, j = j.n, n = n − 1
21          end if
22        end while
23        while m > 0 do t.n = i, t = i, i = i.n, m = m − 1
24        while n > 0 do t.n = j, t = j, j = j.n, n = n − 1
25      end while
26      t.n = nil, l = h, o = 2o
27    end while
```

Algorithm 9.1.8 has $O(n \log n)$ runtime complexity. This is easy to see, once the idea of the merge sort becomes clear. Let us consider a simple illustration. The sequence $7, 2, 6, 1, 4, 5, 9, 3$ is to be sorted. First adjacent pairs of numbers are grouped $(7, 2)$, $(6, 1)$, $(4, 5)$, $(9, 3)$ and the numbers within the pairs sorted $(2, 7)$, $(1, 6)$, $(4, 5)$, $(3, 9)$. In the next step the pairs are merged into the groupings of four numbers, while the order is being preserved. $(1, 2, 6, 7)$, $(3, 4, 5, 9)$. The merge operation is performed again and the final sorted sequence results. Each merge operation can be done in $O(n)$ time and there is at most $\log n$ groupings, thus the runtime complexity follows. In the above algorithm the outer loop controls the current length of grouping $(o + 1)$, while the merge operation is performed in lines 14 - 25.

One can imagine the situation when a long and initially sorted list is altered in the way, that each item is shifted by few places to the right or to the left. The list remains sorted in the average sense. That is to say, if we could assign a colour to the magnitude of each data item, then the altered list observed from a distance would seem very similar to the sorted one. Under a closer look it appears most natural to restore the right order by inspecting each item and shifting it back into the right position. This can be done in a fast manner, provided the alteration is small compared to the length of the list. This idea is utilised by the *insertion sort* algorithm

ALGORITHM 9.1.9. *List_Insertion_Sort* (l)

```
1   p = l
2   while p ≠ nil
3     q = p, p = p.n
4     while q.p ≠ nil ∧ q.p.d > q.d do
5       o = q.p
6       if o.p ≠ nil then o.p.n = q
7       else l = q
8       if q.n ≠ nil then q.n.p = o
9       q.p = o.p, o.n = q.n, q.n = o, o.p = q
10    end while
11  end while
```

The insertion sort has complexity $O(an)$ where a is the average shift length in the originally sorted list. For $a \ll n$ it becomes $O(n)$, which is a useful result at times. The fact that the average shift is enough to asses the complexity follows from the simple observation that $(c_1 + c_2 + ... + c_n)/n = a$, where c_i is the number of comparisons necessary to bring an altered item back into its right place.

9.1.4. Hash table. Let $h[\cdot]$ be a table of pointers to lists (d, p, n), defined in the previous section. Assume $h[\cdot]$ is of size m. Let $f(d)$ be a surjective *hashing* function such that

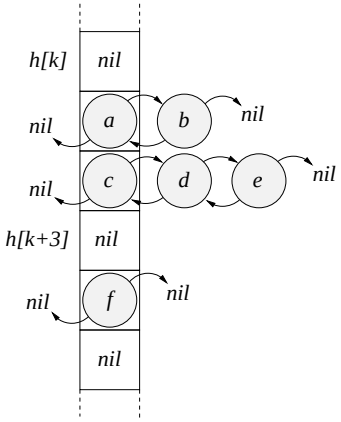


FIGURE 9.1.2. Hash table with lists.

$$\forall d : f(d) \in \{1, 2, ..., m\} \tag{9.1.1}$$

and the evaluation time of $f(d)$ takes $O(1)$ time. One can define the following operations

ALGORITHM 9.1.10. *Hash_Table_Insert* (h, f, d)
1 *List_Insert* $(h[f(d)], d)$

and

ALGORITHM 9.1.11. *Hash_Table_Delete* (h, f, d)
1 *List_Delete_Data* $(h[f(d)], d)$

as well as

ALGORITHM 9.1.12. *Hash_Table_Find* (h, f, d)
1 ***return*** *List_Find_Item* $(h[f(d)], d)$

The insertion into the hash table has $O(1)$ complexity. The hash table deletion and search on the other hand, have the complexity proportional to the length of the list stored at the table element $h[f(d)]$. This length depends on the *quality* of the hashing function. By definition it is possible that

$$\exists x, y : f(x) = f(y) \tag{9.1.2}$$

which is called a *collision*. A hash table can be efficient, provided collisions happen rarely. This is in general the case, if the probability of collision reads

$$\forall_{x,y} : P|_{f(x)=f(y)} = O\left(\frac{1}{m}\right) \tag{9.1.3}$$

with a small ($\ll m$) constant in $O(\cdot)$. In this case collisions are distributed uniformly over $h[\cdot]$, with the probability proportional to $\frac{n}{m}$, where n is the number of stored data items. Thus, for a good hashing function the average length of the list stored at any element $h[\cdot]$ is $O\left(\frac{n}{m}\right)$. If all data items d are known, one can always index them from 1 to m in such a way, that no more than $\left\lfloor \frac{n}{m} \right\rfloor + 1$ share an index, where $\lfloor \cdot \rfloor$ indicates the nearest smaller or equal integer. If the set of d is not known in advance, existence of good hashing functions is not assured. In practice though, reasonably efficient functions can be found. It should be noted, that the presented

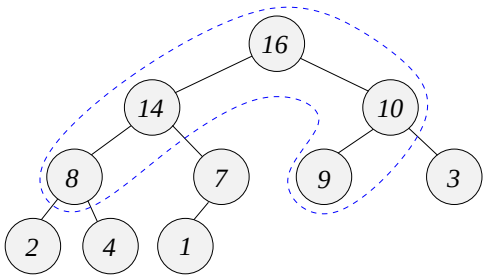


FIGURE 9.1.3. Priority queue arranged into a tree-like heap [**50**]. The dashed lines shows that all k nodes with $y \geq 8$ can be reported by simply descending down the tree in $O(k)$ time.

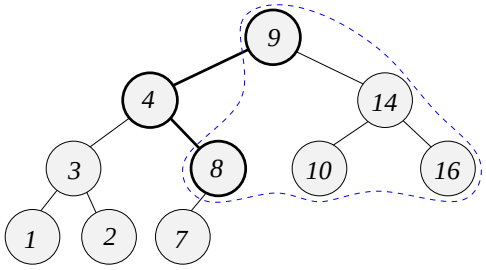


FIGURE 9.1.4. A binary search tree. The thickened path has to be traversed in order to enumerate all k nodes with $x \geq 8$, which for a balanced tree can be done in $O(\log n + k)$.

variant of hashing is not among the most subtle versions of this technique. One might like to consult Knuth [**119**, pp. 552-601] for a more insightful exposition.

9.1.5. Priority queue. Let us consider a set Q of n elements (d, y), where d represents an arbitrary data and y describes a priority assigned to this data. The interest is in maintaining Q in such a way, that the subset of k elements such that $y \geq y_0$ can be accessed in $O(k)$ time. The maintenance operations include insertions and deletions of elements and eventually, updates of their priorities. A data structure facilitating the mentioned operations bears the name of the *priority queue*. A typical efficient implementation of the priority queue exploits the *heap* structure as its skeleton. A through description of both structures can be found in Cormen *et al.* [**50**, pp. 127-144]. A specific implementation of the priority queue will be outlined in Section 9.1.7. Here instead, let us illustrate that the elements of a priority queue can be arranged into a tree-like heap structure. Let us expand the tuple (d, y) into (d, y, p, l, r), where p is the pointer to a parent node in the tree, l is the pointer to the left sub-tree (left child), and r is the pointer to the right sub-tree (right child). One can arrange the elements of Q into a tree-like structure satisfying the *heap property*: for each $v \in Q$, if $v.p \neq nil$ then $v.p.y \geq v.y$. Consider elements of Q with priorities $\{1, 2, 3, 4, 7, 8, 9, 10, 14, 16\}$. An example of such arrangement is given in Figure 9.1.3. The dashed line bounds 5 elements with priorities $y \geq 8$. They can be enumerated by descending down the tree in the $O(5)$ time.

9.1.6. Binary search tree. Similarly as in the previous section, let us consider a set Q of n elements (d, x), where d represents an arbitrary data and x stands for a coordinate assigned to this data. The objective will be to maintain Q in such a way, that for a given x_0 the set of all k elements $v \in Q$ such that $v.x \geq x_0$ can be identified in $O(\log n + k)$ time. It is possible to arrange the elements of Q into

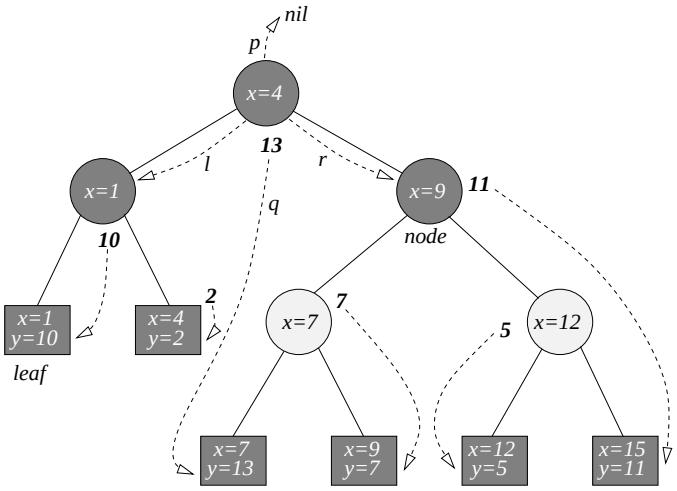


FIGURE 9.1.5. Priority search tree based on the red-black tree structure. Red nodes are light-grey. Numbers outside of the rectangular leaves and circular nodes represent the priority queue. Dashed lines indicate pointers.

a tree-like structure (d, x, p, l, r) satisfying the *binary search tree property*: for each $v \in Q$, if $v.l \neq nil$ then $v.l.x \leq v.x$ and if $v.r \neq nil$ then $v.x \leq v.r.x$. Of course p, l, r denote respectively, the pointer to a parent node in the tree, the pointer to the left sub-tree, and the pointer to the right sub-tree. The n elements of Q can be arranged into a binary search tree in such a way that the number of nodes along the longest path in the tree is $O(\log n)$. Such a tree is called *balanced*. A balanced binary search tree corresponding to the coordinate set $\{1, 2, 3, 4, 7, 8, 9, 10, 14, 16\}$ is illustrated in Figure 9.1.4. A specific instance of the balanced binary search tree will be detailed in the next section.

9.1.7. Priority search tree. Priority search tree has been proposed by McCreight [**148**] as a combination of the priority queue and the balanced binary search tree. The structure operates on tuples (d, x, y), where an arbitrary data d is associated with two coordinates x and y. Priority search tree allows to efficiently process a number of *range queries*, one of which is of particular interest in the current context (Section 9.3.2):

PROBLEM 9.1.13. For a set Q of n tuples (d, x, y), given x_0 and y_0, report all $v \in Q$ such that $v.x > x_0$ and $v.y > y_0$.

The priority search tree presented in this section is based on the *red-black tree* structure, invented by Bayer [**21**] (the name used by him was the *symmetric binary B-tree*, while Guibas and Sedgewick [**80**] have introduced the *red-black* colouring convention). A comprehensive description of the data structure can be found in Cormen *et al.* [**50**, pp. 273-301]. For the sake of completeness, a rather detailed extraction from Cormen *et al.* is included here. It is further completed by embedding the priority queue structure within the red-black tree.

An element of the priority search tree comprises (u, t, c, p, l, r, q), where u is the tuple (d, x, y), $t \in \{node, leaf\}$ describes the type of the element, $c \in \{red, black\}$ is the colour of the element, p is the pointer to the parent of the element, l is the pointer to the left sub-tree (left child), r is the pointer to the right sub-tree (right child), and q is the pointer to the element of the priority queue (Figure 9.1.5). Tree

elements v for which $v.t = node$ are called *nodes*, while those where $v.t = leaf$ are called *leaves*. The following properties are quoted after Cormen *et al.* [**50**, p. 273]:

(1) Every node is either red or black.
(2) The root is black.
(3) Every leaf is black.
(4) If a node is red, then both its children are black.
(5) For each node, all paths from the node to descendant leaves contain the same number of black nodes.

It should be noted, that in general it is not necessary to employ separate tree elements for all leaves in the red-black tree. In a typical application only nodes store data. Nevertheless, the priority search tree requires the additional leaf space. Let us define $\mathrm{bh}(v)$, the *black height* of a node v, as the number of black nodes (not including v) on the way from v down to a leaf. Similarly, let $\mathrm{h}(v)$ bet the height of the sub-tree rooted at v, that is the maximal number of nodes (including v) on the way down from v to a leaf. Also, let $\mathrm{n}(v)$ denote the number of nodes of a sub-tree rooted at v. An empty tree contains no nodes, that is if v is the root, then $v = nil$. Lemma below states a basic result about the efficiency of red-black trees.

LEMMA 9.1.14. *Red-black tree with n nodes has height at most* $2\log(n+1)$.

PROOF. First one needs to show that $\mathrm{n}(v) \geq 2^{\mathrm{bh}(v)} - 1$. If $\mathrm{h}(v) = 0$ then $\mathrm{bh}(v) = 0$. Thus $\mathrm{n}(v) \geq 2^0 - 1 = 0$, which is correct. Now assume $\mathrm{h}(v) = k$ and $\mathrm{n}(v) \geq 2^{\mathrm{bh}(v)} - 1$. Take w, such that $\mathrm{h}(w) = k + 1$. If $w.c = read$ then (from property 4) $\mathrm{bh}(w) = \mathrm{bh}(w.l) + 1 = \mathrm{bh}(w.r) + 1$, otherwise $\mathrm{bh}(w) = \mathrm{bh}(w.l) = \mathrm{bh}(w.r)$. Thus, by the inductive hypothesis $\mathrm{n}(w) \geq \left(2^{\mathrm{bh}(w)-1} - 1\right) + \left(2^{\mathrm{bh}(w)-1} - 1\right) + 1 = 2^{\mathrm{bh}(w)} - 1$. Let h be the height of the tree. From property 5 there follows that the black height of the root is at least $h/2$ (try to insert as many red nodes as possible). It results that $h \geq 2^{h/2} - 1$, or in other words $h \leq 2\log(n+1)$. □

The height of the proposed priority search tree is then $O(\log n)$. This implies that, as long as the properties 1-5 can be maintained, all operations traversing the tree along its height and performing on the way some constant time actions will have $O(\log n)$ complexity. Three basic operations will be considered: insertion, deletion, and the already mentioned two-sided range query. It will be useful to define the comparison of tuples (d, x, y) first

$$\begin{gathered}(d_i, x_i, y_i) < (d_j, x_j, y_j) \text{ iff } x_i < x_j \vee (x_i = x_j \wedge d_i < d_j) \\ (d_i, x_i, y_i) = (d_j, x_j, y_j) \text{ iff } x_i = x_j \wedge d_i = d_j \\ (d_i, x_i, y_i) > (d_j, x_j, y_j) \text{ otherwise.}\end{gathered} \tag{9.1.4}$$

The following two routines will be utilised to maintain the priority search tree.

ALGORITHM 9.1.15. *Pst_Push* (v, q)

```
1  while v.q ≠ nil
2     if v.q.u.y < q.u.y then
3        s = v.q, v.q = q, q = s
4     end if
5     if q.u ≤ v.u then v = v.l
6     else v = v.r
7  end while
```

The above algorithm descends down from the root v comparing the current queue coordinates $v.q.u.y$ against the candidate $q.u.y$ (lines 2-4). If the currently

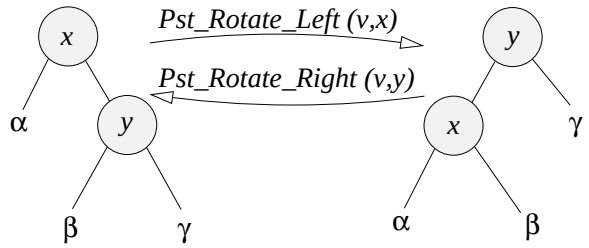


FIGURE 9.1.6. Left and right rotations. The left to right ordering of data tuples according to comparison (9.1.4) is preserved in nodes x, y and sub-trees α, β, γ.

stored $v.q.u.y$ is smaller than the candidate coordinate, $v.q$ and q are swapped (line 3) and the descend continues along the binary search path of q (lines 5-6). If the comparison (9.1.4) is $O(1)$ then the runtime of Algorithm 9.1.15 is $O(\log n)$. The Pst_Push routine does not affect the structure of the red-black tree. A reverse operation follows.

ALGORITHM 9.1.16. *Pst_Pull* (v)

```
1  do
2     if v.t = leaf then
3        v.q = nil
4        return
5     else s = v.l
6     if s.q = nil ∨ (v.r.q ≠ nil ∧ v.r.q.y > s.q.y) then s = v.r
7     v.q = s.q
8     v = s
9  while v.q ≠ nil
```

Algorithm 9.1.16 descends down the tree v. For each tree node v, its priority queue link $v.q$ is replaced with either $v.l.q$ or $v.r.q$ (line 7), depending on whether respectively $v.l.q \geq v.r.q$ or the opposite holds (lines 5-6). The search continues down the path of the maximal priority choice (line 8). The loop terminates either at a leaf element (lines 2-4), or at the end of the queue (line 9). The runtime is $O(\log n)$. The structure of the red-black tree remains unaffected.

Structural changes to the red-black tree will be caused by insertions and deletions. Before these can be analysed, the following two auxiliary routines need to be considered.

ALGORITHM 9.1.17. *Pst_Rotate_Left* (v, x)

```
1  y = x.r
2  x.r = y.l
3  y.l.p = x
4  y.p = x.p
5  if x.p = nil then v = y
6  else if x = x.p.l then x.p.l = y
7  else x.p.r = y
8  y.l = x
9  x.p = y
10 s = y.q, y.q = x.q, x.q = s
11 Pst_Pull (x)
12 if s ≠ nil then Pst_Push (y, s)
```

and

ALGORITHM 9.1.18. *Pst_Rotate_Right* (v, x)
Rewrite Algorithm 9.1.17 with .l and .r swapped.

Above v is the root of the tree, and x is the node about which the rotation is supposed to happen. The left rotation and the right rotation are pictured in Figure 9.1.6. Lines 1-9 in Algorithm 9.1.17 basically update pointers in compliance with Figure 9.1.6. This is a standard part of left-rotation, exactly as in Cormen *et al.* [**50**]. Lines 10-12 update the priority search tree structure. It is seen that as a result of the left rotation y, a former child of x, becomes the parent of x. Thus certainly the queue pointers in x and y need to be swapped. This happens in line 10. The pointer $y.q$ maintains the priority queue property with respect to its both children ($y.q \geq \gamma.q$ is preserved, and the swap in line 10 ensures $y.q \geq x.q$). Nevertheless, although $x.q \geq \beta.q$ (inherited after y), there does not necessarily hold $x.q \geq \alpha.q$. This is remedied by pulling $x.q$ out of the queue in line 11, followed by pushing it back down the queue in line 12 (s is pushed down the $y-$rooted tree, as it actually might have been coming from the γ sub-tree). Because of the priority queue update, the runtime of Algorithms 9.1.17 and 9.1.18 is $O(\log n)$.

Rotations will be utilised as one of the actions aimed at restoring the red-black tree properties 1-5 after an insertion or a deletion has taken place. Let us consider the insertion first.

ALGORITHM 9.1.19. *Pst_Insert* (v, x, y, d)

```
1  if v = nil then
2     v = ((d, x, y), leaf, black, nil, nil, nil, nil), v.l = v.r = v.q = v
3     return
4  end if
5  p = q = v
6  u = (d, x, y)
7  while p.t ≠ leaf
8     q = p
9     if u < p.u then p = p.l
10    else if u > p.u then p = p.r
11    else return
12 end while
13 if u = p.u then return
14 if p ≠ q then p.p = q
15 p.t = node, p.c = red
16 p.l = (nil, leaf, black, nil, nil, nil, nil), p.l.l = p.l.r = p.l
17 p.r = (nil, leaf, black, nil, nil, nil, nil), p.r.l = p.r.r = p.r
18 if u < p.u then
19    p.r.u = p.u
20    p.l.u = p.u = u
21    for q = p while q.q ≠ p do q = q.p
22    q.q = p.r
23    Pst_Push (v, p.l)
24 else
25    p.r.u = u
26    p.l.u = p.u
27    for q = p while q.q ≠ p do q = q.p
28    q.q = p.l
29    Pst_Push (v, p.r)
30 end if
31 Pst_Insert_Fixup (v, p)
```

Algorithm 9.1.19 takes as the arguments the tree root v, the two coordinates x and y and a data item d. In case the tree is empty, a single leaf element is created as a root (lines 1-4). Otherwise the tree is traversed down the (d, x, y) comparison path, until a leaf is found (lines 5-12). It is assumed, that all data items d are distinct. Thus, the insertions exits in lines 11 and 13, rather than updating the y priority. The parent pointer is updated in line 14, for all but the initial root leaf. Then the found leaf is transformed into a node, and its colour changed from black into red in line 15. Two leaf children of the new node are created in lines 16-17. Note that the pointers are set characteristically for leaves (for the sake of correctness of rotation routines). In the binary search ordering of the red-black tree structure the convention is used, that all $\leq$ data is stored to the left of a node. It follows that the former data of node p is now moved to its right child in line 19. Then the new data takes place of the one in node p and in its left child (line 20). Lines 21-23 deal with the update of priority queue. We wish to preserve the principle, that queue pointers point to the data stored at leaves. Thus, the search is done up the tree in line 21, in order to locate the queue pointer, pointing at p. As ps data has moved to its right child, the pointer is now reset to $p.r$ (line 22). The newly inserted data is pushed down the priority queue in line 23. The same procedure is repeated symmetrically in lines 25-29. Finally, as the colour of the new node was changed to red in line 15, the red-black tree structure needs to be maintained in order to preserve properties 1-5. This is done inside of the fix-up routine listed below.

ALGORITHM 9.1.20. *Pst_Insert_Fixup* (v, x)

```
1   while x ≠ v ∧ x.p.c = red
2     if x.p = x.p.p.l then
3       y = x.p.p.r
4       if y.c = red then
5         x.p.c = black
6         y.c = black
7         x.p.p.c = red
8         x = x.p.p
9       else
10        if x = x.p.r then
11          x = x.p
12          Pst_Rotate_Left (v, x)
13        end if
14        x.p.c = black
15        x.p.p.c = red
16        Pst_Rotate_Right (v, x.p.p)
17      end if
18    else
...       Rewrite lines 3-17 with .l and .r swapped.
34    end if
35  end while
36  v.c = black
```

A detailed analysis of Algorithm 9.1.20 can be found in Cormen *et al.* [**50**, pp. 280-287]. As it is rather lengthy, it would be excessive to repeat it here. It is enough to note, that the properties of the red-black tree are restored by Algorithm 9.1.20 in $O(\log n)$ time. The only point where an additional comment is necessary concerns rotations. Due to the priority queue related modifications the time complexity of rotations is $O(\log n)$ rather than $O(1)$. Nevertheless, as commented in [**50**, p. 287], the insertion fix-up performs at most two rotations. As a result the total runtime

of the insertion Algorithm 9.1.19 remains $O(\log n)$. Let us resort to the deletion now.

ALGORITHM 9.1.21. *Pst_Delete* (v, x, y, d)

```
1  u = (d, x, y)
2  r = v, p = q = nil
3  while r.t ≠ leaf
4     if r.q ≠ nil ∧ u = r.q.u then p = r
5     q = r
6     if u ≤ r.u then r = r.l
7     else r = r.r
8  end while
9  if u ≠ r.u then return
10 if p ≠ nil then Pst_Pull (p)
11 if q ≠ nil then
12    if r = q.l then p = q.r
13    else p = q.l
14    p.p = q.p
15    if q.p = nil then v = p
16    else if q = q.p.l then q.p.l = p
17    else q.p.r = p
18    if q.q ≠ nil ∧ q.q ≠ q then Pst_Push (p, q.q)
19    if q.c = black then Pst_Delete_Fixup (v, p)
20    free q
21 else v = nil
22 free r
```

Algorithm 9.1.21 takes as arguments the tree root v, the coordinates x and y, and the data item d. It descends down the tree until a leaf r holding $u = (d, x, y)$ is found (lines 3-8). Along the way a pointer p to the tree node holding the priority queue element associated with u is recorder (line 4). If the right leaf was not found, the algorithm exits in line 9. If a priority queue element associated with u was found in a tree node, it is pulled out of the queue in line 10. For a tree not composed of a single root leaf (line 11), the usual binary tree deletion is performed on node q (otherwise the root is set to *nil* in line 21). First the qs parent branch is set to the child of q which is not being deleted (lines 12-17). As q itself is to be removed an eventual queue element is push down the remaining sub-tree in line 18. If q is black, then its deletion is likely to alter the balance of black nodes across the tree height (principle 5). An appropriate fix-up is performed in line 19. It is marked in the code, that the storage of q and r can be deleted (lines 20 and 22).

ALGORITHM 9.1.22. *Pst_Delete_Fixup* (v, x)

```
1  while x ≠ v ∧ x.c = black
2     if x = x.p.l then
3        y = x.p.r
4        if y.c = red then
5           x.p.c = red
6           y.c = black
7           Pst_Rotate_Left (v, x.p)
8           y = x.p.r
9        end if
10       if y.l.c = black ∧ y.r.c = black then
11          y.c = red
12          x = x.p
```

```
13        else
14          if y.r.c = black then
15            y.l.c = black
16            y.c = red
17            Pst_Rotate_Right (v, y)
18            y = x.p.r
19          end if
20          y.c = x.p.c
21          x.p.c = black
22          y.r.c = black
23          Pst_Rotate_Left (v, x.p)
24          x = v
25        end if
26      else
...         Rewrite lines 3-25 with .l and .r swapped.
42      end if
43    end while
44    x.c = black
```

Similarly as in case of insertion, the purpose of Algorithm 9.1.22 is to maintain properties and therefore balance of the underlying red-black tree structure. The procedure is included here for the sake of completeness. For a through analysis the reader is referred to the comments in [**50**, pp. 288-293]. At most three rotations can take place during the deletion fix-up, thus the usual $O(\log n)$ deletion time is maintained, even though rotations take $O(\log n)$ in the current case.

It remains to discuss an algorithm answering the query defined as Problem 9.1.13. The following routine takes as the arguments the tree root v, the minimal coordinates x_0 and y_0, an arbitrary data pointer δ, and a callback routine $Report(\delta, d)$.

ALGORITHM 9.1.23. *Pst_Query* $(v, x_0, y_0, \delta, Report)$

```
1  p = nil
2  while v ≠ p ∧ v.q ≠ nil
2    p = v
4    if v.q.u.x > x_0 ∧ v.q.u.y > y_0 then Report (δ, v.q.u.d)
5    if x_0 ≤ v.u.x then
6      if v ≠ v.r then Pst_Report_Down (v.r, y_0, δ, Report)
7      v = v.l
8    else v = v.r
9  end while
```

Algorithm 9.1.23 descends down the tree along the path indicated solely by the x coordinate of the stored data (lines 5, 8) and the range limit x_0. If the current x coordinate is larger or equal to x_0, the right sub-tree can only store data with $x > x_0$. Thus, a complete branch of the priority queue is recursively reported as long as $y > y_0$ (line 6, and Algorithm 9.1.24 below). Similarly, queued data items encountered on the way down are eventually reported in line 4.

ALGORITHM 9.1.24. *Pst_Report_Down* $(v, y_0, \delta, Report)$

```
1  if v.q = nil then return
2  else if v.q.u.y ≤ y_0 then return
3  Report (δ, v.q.u.d)
4  if v.t = leaf then return
5  Pst_Report_Down (v.l, y_0, δ, Report)
6  Pst_Report_Down (v.r, y_0, δ, Report)
```

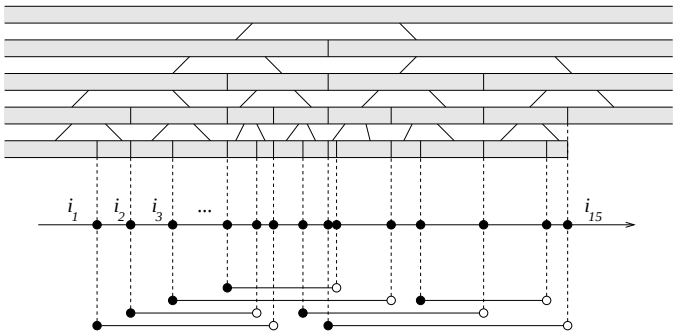


FIGURE 9.1.7. Binary tree on atomic intervals

9.1.8. Segment and interval trees. All of the presented so far data structures had $O(n)$ space complexity. The structures outlined in this section demand more space, which is also the reason why they are rarely implemented in practice. A suitable, practical generalisation will be detailed in Section 9.3.4.

A data structure called the *segment tree* [**25**] allows to solve the following

PROBLEM 9.1.25. (*Stabbing query*) Given a set of intervals S and a query point, report all intervals that contain the point.

On the other hand, the *range tree* [**27**] can be used to solve

PROBLEM 9.1.26. (*Range search*) Given a set of points P and a query interval, report all points that belong to the interval.

It is easy to notice that Problems 9.1.25 and 9.1.26 are dual in the following sense: The same set of pairs $X \subset S \times P$ is reported, when solving either the stabbing problem on S with points from P, or then range search problem on P with intervals from S.

Let us have a look into the segment tree first. Endpoints of intervals from S subdivide the real line into the set of so called *atomic intervals* (intervals $i_1, i_2, ..., i_{15}$ in Figure 9.1.7). They can be assumed half-open, say at their right endpoints. It is not difficult to create a balanced binary search tree, such that atomic intervals are leaves and each node is a union of its offspring intervals. Consequently, the root node spans the entire real line. This is a convenient search structure for point queries, but not yet a segment tree. To obtain the segment tree, one needs to store information about the intervals of S in tree nodes. Assume that tree nodes are supplied with auxiliary lists, storing some of the intervals from S. Let tree node u be associated with an interval I_u. The following rule is applied: An interval $s \in S$ is stored in u if and only if $I_u \subset s$ and $I_{parent(u)} \not\subset s$ (Figure 9.1.8). This ensures, that an interval is stored *at most twice* at each level of the tree. For if this wouldn't be the case and there would exist n nodes $u_1, u_2, ..., u_n$ storing an interval s at one level of the tree, then $I_{parent(u_2)} \subset s, I_{parent(u_3)} \subset s, ..., I_{parent(u_{n-1})} \subset s$ (due to the binary tree structure), which contradicts the assumed manner of storing intervals at tree nodes. Hence, each interval is stored no more than $O(\log n)$ times in the tree. It follows, that the $O(n \log n)$ space is necessary for the segment tree.

Segment tree can be constructed in *bottom-up* or *top-down* manner. In the former case, a good algorithm finding an approximate median of a set of points is necessary [**44, 20**]. The tree is begin built by descending down and splitting point sets according to the median. Building the tree this way requires $O(n \log n)$ steps on average. In case of the bottom-up approach one first sorts P, and then builds the tree climbing up from the leaves level. This results in the well balanced tree, built in at most $O(n \log n)$ steps.

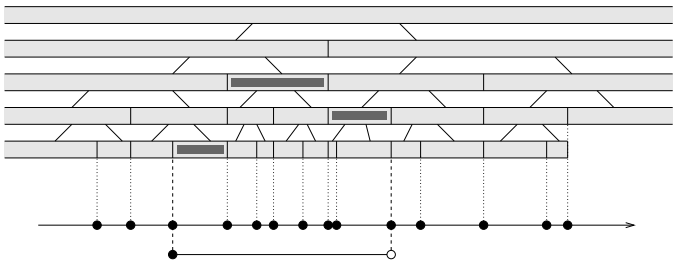


FIGURE 9.1.8. Storing an interval into a segment tree.

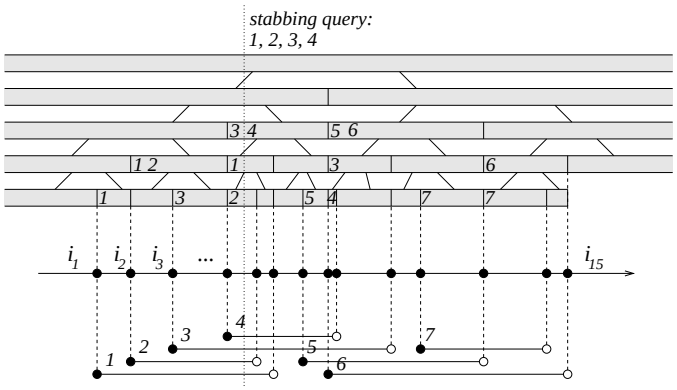


FIGURE 9.1.9. A complete segment tree with a *stabbing query* example.

It remains to discuss is the stabbing query itself (Figure 9.1.9). One needs $O(\log n)$ steps to query the segment tree with a point. For each node u the intervals stored in the nodal list L_u are reported. This way each of the intervals stabbed by the point is reported once, which gives $O(\log n + k)$ query complexity. Once the tree has been built, the stabbing query can be answered efficiently. Nevertheless, we are not satisfied with the space requirements of the segment tree. In Section 9.3.4 it will be shown, how to avoid storing of the complete tree in computer memory, which considerably relaxes the theoretical space requirements of the structure.

The range tree structure is quite similar to the segment tree. The only difference is that in auxiliary nodal lists one stores points, contained within the nodal intervals. Root of the tree stores then the complete set P. One queries the range tree with an interval s, and during this process splits s into $O(\log n)$ parts (exactly as it was done, while storing s in the segment tree). As the query descends down the tree, all points stored in nodes whose intervals I_u are contained in partitions of s are reported. The remarks relevant to the space and time requirements of the segment tree can be repeated for the range tree without change.

9.2. The optimal data structure

When implementing a computer code, one naturally realises what are the desired features of an algorithm. Simplicity, speed and modest usage of space are among the sought qualities. At the same time, one realises that these goals are at times mutually exclusive. Some need to be traded off against others. This is not an exception for the algorithms related to contact search. The purpose of this section is to discuss an *imaginary*, optimal data structure suitable for contact search in dynamic multi-body simulations. It is relevant to realise what an ideal is, before compromising some of its aspects on the way to the practical implementation.

It was already assumed in Chapter 2 that bodies are subdivided into elements. It is arbitrarily decided here to use elements as irreducible geometric atoms. Hence for a contact point, there follows

DEFINITION 9.2.1. A single oriented contact point results from an overlap of two surface elements.

To be somewhat more precise, let $\bar{\mathcal{B}}_i = \cup_j \bar{e}_{ij}$ be the configuration of body i, where e_{ij} is the jth surface element of body i, and the upper dash stands for the set closure. Assume that there exist two bodies $\mathcal{B}_i$ and $\mathcal{B}_k$, such that $\bar{\mathcal{B}}_i \cap \bar{\mathcal{B}}_k \neq \emptyset$. Then, the set of contacts $(\mathbf{x}_{ikjl}, \mathbf{n}_{ikjl}, \mathcal{B}_i, \mathcal{B}_k)$ is defined by points and normals corresponding to all nonempty intersections of elements $\bar{e}_{ij} \cap \bar{e}_{kl} \neq \emptyset$. Details on calculating $(\mathbf{x}, \mathbf{n})_{ikjl}$ for a pair of elements (e_{ij}, e_{kl}) are provided in Section 9.4. For the moment, it is enough to say that this operation takes $O(1)$ time (with a rather large constant factor), which results from the finite variety of element shapes.

Within the above model of acquiring contacts, it is natural to think about a data structure storing elements, and possessing the following qualities:

(1) Insertion of new elements is possible and fast. This is related to the scenario, when new bodies enter an active simulation. For example a granular flow simulation with a source requires insertions.
(2) Deletion of elements is possible and fast. For example scattering of bodies might require deletions, when some prescribed boundaries are crossed. Insertion and deletion together allow for modelling of cracking and separation.
(3) Insertions and deletions should take at most $O(\log n)$ steps, where n is the number of stored elements.
(4) Overlap creation between pairs of elements should be efficiently reported. This includes both the overlaps resulting from element insertions and the overlaps created after an update of element positions.
(5) Overlap release between pairs of elements should be efficiently reported. This includes both the overlaps released after element deletions and the overlaps released after an update of element positions.
(6) Overlap creation/release reports should take at most $O(n+k)$ steps, where k is the number of creation/release events.
(7) Elements directly, topologically adjacent in a mesh should be excluded from overlap reports. The self-contact case is still included.
(8) Exclusion of selected pairs of elements should be possible. This might be of use in the vicinity of joints, where mesh overlaps are sometimes tolerated.
(9) The space complexity of the data structure should be $O(n)$.

The insertion and deletion times listed in point 3 is quite stringent. Without having in mind yet any specific realisation of the data structure, it is acknowledged that data should be stored in some *order*. The fastest purely combinatorial linear structures allowing for dynamic insertions and deletions are balanced binary search trees. Thus, although our hypothetical structure operates in three dimensions, we wish to retain the $O(\log n)$ insertion and deletion times. The overlap report complexity listed in point 6 is in fact even more stringent. It is assumed that the ordering maintained during insertions, deletions and updates of the structure is sufficient to trace creation and release of overlaps in $O(n+k)$ time. If this could be assured, complexity of the contact search would not exceed that of the time stepping. Nevertheless, algorithms detecting overlaps between geometrical objects in three dimensions are slower. This will be demonstrated for shapes as

simple as rectilinear boxes in Section 9.3.1.1. Still, our hope is in the improvement resulting from processing *nearly ordered* data, similarly as it was the case with sorting (Algorithm 9.1.9). A contact search algorithm operates in between of the time integration steps. Therefore it is quite legitimate to assume that ordering of data corresponding to adjacent time frames is similar. This is called *time coherence.* Several overlap search algorithms and related data structures will be investigated in Section 9.3. Few of them will take advantage of the time coherence.

It will be useful to sketch the interface routines for the hypothetical data structure s. Let insertion and deletion of elements e read

ALGORITHM 9.2.2. *Imaginary_Insert* (s, e)
Insert e into s while preserving an implicit ordering.

and

ALGORITHM 9.2.3. *Imaginary_Delete* (s, e)
Delete e from s while preserving an implicit ordering.

Somewhat more can be said about the actions taken inside of the update routine. All of the overlap event reports happen as a consequence of the update of the structure. This means that overlap events related to insertions and deletions are postponed and executed on the occasion of an update. This is related to the anticipated difficulties with an efficient reporting of overlaps during the insertion process. In consequence it is more elegant to assume that all overlap events are reported during the update. This is just a pragmatic choice, dictated by experience. The update routine follows below.

ALGORITHM 9.2.4. *Imaginary_Update* $(s, \delta, Created, Released)$
Find overlaps released due to deletions.
Find overlaps released due to motion of elements.
For each overlap release call $Released(\delta, e_{ij}, e_{kl})$.
Find overlaps created due to insertions.
Find overlaps created due to motion of elements.
For each overlap creation call $Created(\delta, e_{ij}, e_{kl})$,
if and only if the element pair is not topologically adjacent or it was not explicitly excluded.

Equipped with the above structure and the knowledge on how to extract contact points and normals from the pairwise element overlaps, one can easily execute a variety of contact detection tasks. Of course, in practice some aspects of this general idea (typically efficiency) will need to be compromised.

9.3. Finding contact candidates

The data structure outlined in the previous section operated on the surface elements, defined in Chapter 2. In fact, this is not the most convenient approach. It is understandable, that overlaps between objects of simple shapes can be found more rapidly, then between those of intricate shapes. Although element shapes are quite simple, designing a data structure operating directly on them is still too cumbersome. In terms of maintaining an ordering or testing for intersections, rather than using the elements, it is much easier to deal with their axis-aligned extents. The three axis-aligned intervals form a box, called the *axis aligned bounding box* (Figure 9.3.1). Obviously, if two bounding boxes do not overlap, their related elements cannot intersect. Thus, the rejection test is simple and conclusive. On the other hand, the overlap of boxes only indicates a potential intersection of the underlying elements. It will be shown in Section 9.3.1, that under some practical assumptions, the bounding box overlaps reflect quite well the actual element overlaps.

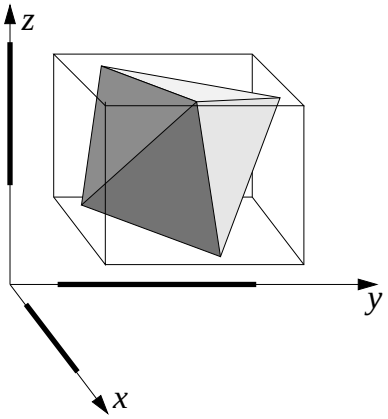


FIGURE 9.3.1. An axis aligned bounding box around a pyramid element. Projections on the coordinate axes have been thickened.

For taxonomic reasons it is relevant to mention, that the presented framework belongs to the broader category of *two-phase* collision/contact/interface detection methods. Section 9.3 corresponds to the *broad phase*, usually involving some sort of *space partitioning* and/or *bounding volume* strategy, aimed at reporting the contact candidate object pairs. Section 9.4 corresponds to the *narrow phase*, pursuing conclusive intersection tests between the reported pairs of objects. Interface detection methods will be briefly reviewed in Section 9.5.

9.3.1. Axis aligned bounding boxes. Although the axis aligned bounding boxes are a rather simple geometrical device, they proved to be effective in many applications (e.g. computer graphics, geometric modelling, interface detection). The reasons behind this effectiveness have been studied by Suri *et al.* [**201**] and Zhou and Suri [**215**]. It will be useful to recall some of their results. For a set $\mathcal{P}$ of n objects in $d-$dimensional space, the following ratio was considered

$$\rho(\mathcal{P}) = \frac{K_b(\mathcal{P})}{n + K_o(\mathcal{P})} \tag{9.3.1}$$

where $K_b(\mathcal{P})$ denotes the number of intersecting bounding boxes, and $K_o(\mathcal{P})$ corresponds to the number of actual intersections among the objects. Formula (9.3.1) describes the efficiency of the bounding box heuristic. The number of objects n added in the denominator allows to consider also the case when $K_o = 0$. At the same time $O(n + K_0)$ corresponds to the complexity of an optimal algorithm finding all intersections between the n objects. If $\rho(\mathcal{P})$ is a small constant, then one can conclude that the bounding box heuristic performs well, i.e. the overhead of fictitious overlap reports is small. Although it is not hard to picture a situation where $\rho(\mathcal{P}) = O(n)$ (e.g. Figure 9.3.2), it generally corresponds to some pathological shapes arranged in a rather special way. After all, in the current case, the bounding boxes enclose the convex surface elements and thus, it is not possible to end up with a configuration similar to the one in Figure 9.3.2. In [**201, 215**] the object shapes are characterised by their *aspect ratio* and *scale factor*. For an object P, its aspect ratio is defined as

$$\alpha(P) = \frac{\mathrm{vol}(b(P))}{\mathrm{vol}(c(P))} \tag{9.3.2}$$

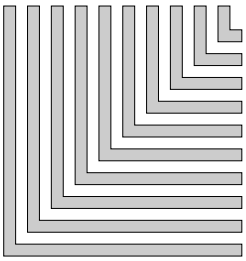


FIGURE 9.3.2. Example of objects shapes where $K_b = O\left(n^2\right)$ and $K_0 = O\left(1\right)$.

where $b\left(P\right)$ and $c\left(P\right)$ are respectively the *enclosing box* and the *core* of P. The enclosing box is defined as the smallest L_∞ ball (or simply, a smallest box) containing P. Core on the other hand, is the largest L_∞ ball contained in P. The aspect ratio of the set $\mathcal{P}$ reads

$$\alpha\left(\mathcal{P}\right) = \max_i \alpha\left(P_i\right) \tag{9.3.3}$$

while the average aspect ratio for the set $\mathcal{P}$ is defined as

$$\alpha_{avg}\left(\mathcal{P}\right) = \frac{1}{n}\sum_{i=1}^{n} \alpha\left(P_i\right) \tag{9.3.4}$$

It is clear that the aspect ratio measures the elongation of an object. The aspect ratio of the set from Figure 9.3.2 is high. The scale factor measures the disparity of object sizes. It is defined then for the set $\mathcal{P}$ as

$$\sigma\left(\mathcal{P}\right) = \max_{i,j} \frac{\mathrm{vol}\left(b\left(P_i\right)\right)}{\mathrm{vol}\left(b\left(P_j\right)\right)} \tag{9.3.5}$$

The difference between the smallest and the largest enclosing box in Figure 9.3.2 is relatively large. Hence, one can see that the large aspect ratio and so the scale factor of the objects in Figure 9.3.2 notably contribute to the possibility of an arrangement resulting in $\rho\left(\mathcal{P}\right) = O\left(n\right)$.

In the chronologically first paper [**201**], Suri *et al.* analyse the ratio (9.3.1) in terms of the maximal bounds (9.3.3) and (9.3.5). The following theorem is quoted without proof

THEOREM 9.3.1. *Let $\mathcal{P}$ be a set of objects in d dimensions, with aspect bound α and scale factor σ, where d is a constant. Then, $\rho = O\left(\alpha\sqrt{\sigma}\log^2\sigma\right)$. Asymptotically, this bound is almost tight, as we can show a family $\mathcal{P}$ achieving $\rho = \Omega\left(\alpha\sqrt{\sigma}\right)$.*

Thus, if α and σ are small constants, there holds $K_b = O\left(K_o\right) + O\left(n\right)$, which shows that the number of box overlaps does not grow faster than the number of actual object intersections (plus an extra $O\left(n\right)$ factor, related to the work that anyhow has to be done if n objects are to be examined). In many practical applications α, σ are small constants. Eventually, objects can be subdivided in order to reduce α and σ, which should increase the effectiveness of the heuristic. Note, that the aspect ratio affects the result in a greater degree than the scale factor.

The lower bound $\rho = \Omega\left(\alpha\sqrt{\sigma}\right)$, described in Theorem 9.3.1, is indeed quite tight for small σ. The authors construct a rather peculiar family of non-convex objects in order to exemplify it. The bound will not be reached in our setting, where convex elements are enclosed by the boxes. In [**215**], Zhou and Suri manage to improve

the upper bound, which together with the already mentioned lower bound allows to refine Theorem 9.3.1 into the following

THEOREM 9.3.2. *Let $\mathcal{P}$ be a set of n objects in d dimensions, where each object has aspect ratio at most α and the family has the scale factor σ, where d is a constant. Then $\rho(\mathcal{P}) = \Theta(\alpha\sqrt{\sigma})$.*

In the same paper [**215**], the value of ρ is estimated with respect to the average aspect ratio α_{avg}. The following result is proven.

THEOREM 9.3.3. *Let $\mathcal{P}$ be a set of n objects in d dimensions , with the average aspect ratio α_{avg} and the scale factor σ, where d is constant. Then $\rho(\mathcal{P}) = \Theta\left(\alpha_{avg}^{2/3}\sigma^{1/3}n^{1/3}\right)$.*

It is seen, that if only the average aspect ratio is bounded (rather than the maximal one), a somewhat less optimistic estimate of the performance is achieved. Nevertheless, $n^{1/3}$ grows slowly and still - a relatively good performance is expected. Proofs of all of the above theorems are too long and technical to be included. However, it is fair to say that the techniques applied in [**201, 215**] seem potentially applicable in the analysis of other geometrical algorithms.

In the context of the results brought up in the above, it is relevant to mention the paper by de Berg *et al.* [**54**], where the idea of *realistic input models* is discussed. The authors notice, that the worst-case performance of geometric algorithms often corresponds to some ill-conditioned and quite unlikely configurations of objects (Figure 9.3.2). By formalisation of the shape and arrangement characteristics, a more realistic analysis becomes possible (e.g. introduction of the aspect ratio and the scale factor led to the practical bounds on ρ). The following notion defined in [**54**] will be of use in our case.

DEFINITION 9.3.4. Let $\mathcal{P} = \{P_1, ..., P_n\}$ be a set of $d-$dimensional objects, and let $\lambda \geq 1$ be a parameter. We say that $\mathcal{P}$ is $\lambda-$low-density if for any L_∞ ball B, the number of objects $P_i \in \mathcal{P}$ with $\text{radius}(b(P_i)) \geq \text{radius}(B)$ that intersect B is at most λ. The density of $\mathcal{P}$ is defined as the smallest λ for which $\mathcal{P}$ is $\lambda-$low-density.

9.3.1.1. *Remarks on finding overlaps.* The above discussion allows to conclude merely, that the axis aligned boxes are useful. The potential of this observation depends however on the availability of an efficient algorithm for the box intersection problem. Several results can be found in the literature in this respect. A two-dimensional version of the problem was solved by Six and Wood [**192**] in $O(n \log n + k)$ time and $O(n \log n)$ space, where n is the number of boxes and k is the number of intersections. The fastest $d-$dimensional result is due to Edelsbrunner and Maurer [**64**] and Edelsbrunner [**65, 66**], where $O\left(n \log^{d-1} n + k\right)$ time and $O\left(n \log^{d-2} n\right)$ space was used. Nonetheless, the algorithm is too complicated to be practical for $d > 2$. Edelsbrunner and Overmars [**67**] discuss a batched version of the intersection problem, enjoying an optimal $O\left(n \log^{d-1} n + k\right)$ time and $O(n)$ space complexity. Zomorodian and Edelsbrunner [**216**] give a fast and practical refinement of this approach, to be discussed in Section 9.3.4.

The brief review of the state of the art allows to conclude, that the algorithm discussed in Section 9.2 is not attainable in general. Hence, when none previous solution is known, the intersection search has to take at least $O\left(n \log^2 n + k\right)$ time. For the consecutive runs though, one hopes to reduce the runtime by exploiting the time coherence.

9.3.1.2. *Bounding box data type.* Let the following tuple (d, lo, hi) describe the axis aligned bounding box. The members of the tuple are respectively: d pointing

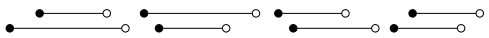


FIGURE 9.3.3. Reciprocate positions of two intersecting intervals.

to an arbitrary data, $lo\,[\cdot]$ being a table of three low corner coordinates, and $hi\,[\cdot]$ being a table of three high corner coordinates.

9.3.2. 1D interval overlap. The following two obvious facts are loosely quoted after [**216**]

FACT 9.3.5. *Two boxes intersect if and only if they intersect in every dimension independently. Hence, it is enough to consider intersection of one-dimensional intervals.*

FACT 9.3.6. *Two intervals intersect if and only if one contains the low endpoint of the other. There are four general positions of two intersecting intervals (Figure 9.3.3).*

Solving the interval overlap problem is then an essential step on the way towards the three-dimensional box intersection. Three static methods and one dynamic will be discussed for that purpose. A *static* algorithm takes as an input the complete set of intervals, and outputs the overlapping pairs. A *dynamic* algorithm bases on a data structure facilitating insertions, deletions as well as the overlap queries.

9.3.2.1. *Scanning (static).* If only we would live in a one-dimensional universe, scanning would have been the single best approach to the box overlap problem. Let a list l store as data the box tuples defined in Section 9.3.1.2. Define the following comparison for a pair u, v of box tuples

$$\begin{array}{c} u < v \,\text{iff}\, u.lo\,[d] < v.lo\,[d] \vee (u.lo\,[d] = v.lo\,[d] \wedge u.d < v.d) \\ u = v \,\text{iff}\, u.lo\,[d] = v.lo\,[d] \wedge u.d = v.d \\ u > v \,\text{otherwise.} \end{array} \tag{9.3.6}$$

where $1 \leq d \leq 3$ is a constant. The following algorithm performs scanning and reports the overlapping interval pairs.

ALGORITHM 9.3.7. *One_Way_Scan* $(l, d, tc, \delta, Report)$

```
1  if tc > 0 then List_Insertion_Sort (l)
2  else List_Merge_Sort (l)
3  while l ≠ nil
4     for u = l.n while u ≠ nil ∧ u.d.lo [d] < l.d.hi [d] do Report (δ, l.d, u.d)
5     l = l.n
6  end while
```

The first argument l of Algorithm 9.3.7 is the list of boxes. The second argument d is the dimension along which the box induced intervals should be scanned. The third argument tc is a flag indicating whether the time coherent run is to be executed ($tc > 0$). The fourth and fifth arguments are an arbitrary data δ of the overlap report callback routine *Report*, and the routine itself. In the first line, if the time coherence is on, the insertion sort is performed, using the box tuple comparison defined in (9.3.6). Otherwise, the merge sort is employed (line 2). In the next stage, a loop over the sorted elements of l is executed (lines 3-6). As intervals are sorted according to their low endpoints, it is now easy to find and report all overlapping pairs by exploiting Fact 9.3.6 (line 4). Scanning is illustrated in Figure 9.3.4.

If $tc \leq 0$, the runtime of Algorithm 9.3.7 is $O\,(n \log n + k)$, where n is the number of objects and k is the number of interval intersections. It should be noted

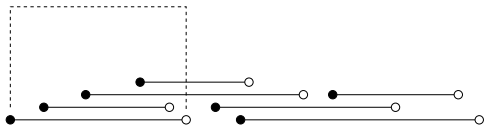


FIGURE 9.3.4. An idea of scanning. The dashed line corresponds to the overlap reports in line 4 of Algorithm 9.3.7.

that only the box intersections along the dth dimension are reported, which leaves the remaining two dimensions unverified. Hence $k \geq k_{box}$, where k_{box} is the number of actual intersections between the boxes. In practice $k \gg k_{box}$ for large n, which renders the scanning quite ineffective as the box overlap solver on large data sets. If time coherence can be enabled (the list is almost sorted), the runtime reduces to $O(n+k)$. Scanning is then ideally compliant to the speedup induced by the time coherence.

9.3.2.2. *Using segment tree (static).* Due to Fact 9.3.6, the interval overlap problem can be formulated as the stabbing query problem (Problem 9.1.25). In order to find all interval overlaps it is enough to identify the pairs *(interval, low endpoint)*, where the low endpoint stabs the interval. Thus, one can build a segment tree on a given set of intervals, and query it with the set of low endpoints. Building the tree takes $O(n \log n)$ time and space. This, together with its logarithmic query time, results in the $O(n \log n + k)$ runtime for the interval overlap problem. The advantage of the time coherence is limited in this context. Although one could argue, that the segment tree can be rebuilt in $O(n)$ time if the intervals were almost sorted, the necessity of performing n queries taking $O(\log n)$ time each is not removed.

9.3.2.3. *Spatial hashing (static).* Although it is not the best idea, also hashing can be applied to the interval intersection problem. A number of elementary features of the spatial hashing can be illustrated on a one dimensional example - this is why the technique is outlined here. Let $f(i)$ be a hashing function from the set of all integer numbers Z onto the set $\{1, 2, ..., m\}$. That is

$$f : Z \to \{1, 2, ..., m\} \tag{9.3.7}$$

An example of such function is $f(i) = i \cdot c \pmod m$, where c is typically a large prime integer [**203**]. Let a function $g : R \to Z$ surjectively map real numbers onto the integer numbers in the following way

$$g(x, s) = \left\lfloor \frac{x}{s} \right\rfloor \tag{9.3.8}$$

where s the so called *voxel* size, and $\lfloor \cdot \rfloor$ extracts the largest integer, not greater than its argument. Let $h[\cdot]$ be a hash table of size m, let $1 \leq d \leq 3$ and b be the box tuple, defined in Section 9.3.1.2. One can now define the following insertion routine

ALGORITHM 9.3.8. *Hash_ 1D_ Insert* (h, s, d, b)
1 $i = g(b.lo[d], s)$, $j = g(b.hi[d], s)$
2 ***while*** $i \leq j$
3 *List_ Insert* $(h[f(i)], b)$
4 $i = i + 1$
5 ***end while***

In the first line above the (i, j) limits of the box projection along the $d-$axis are found. Then all of the $k-$indexed cells, $i \leq k \leq j$, are hashed into the table

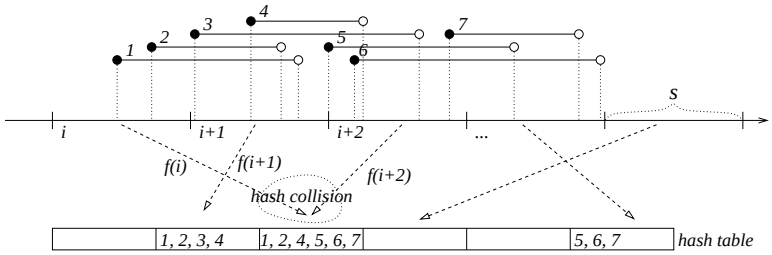


FIGURE 9.3.5. An idea of one-dimensional spatial hashing.

$h[\cdot]$ in line 3. Hence, the box pointer b is placed in the hash lists ranging from $h[i]$ to $h[j]$. Recalling the discussion from Section 9.1.4, the efficiency of the hash table depends on the average length of those lists. The shorter, the better. Neglecting the influence of the hash function, it is readily seen that the voxel size s seriously affects the length of lists stored in $h[\cdot]$. If $s \to \infty$, there will be at most two long lists, one for the negative and one for the positive coordinates. On the other hand, $s \to 0$ results in $j - i \to \infty$ and due to the finite size of $h[\cdot]$ each interval will be stored in each entry of the hash table. In between of those two extremes, there is an optimal size of the voxel. Let $l_i = b_i.hi[d] - b_i.lo[d]$ be the length of the interval associated with box b_i. It is always fair to demand that $\sum (j - i) = O(n)$ in the first line of Algorithm 9.3.8, where the sum is taken over all inserted boxes. This results from a simple observation, that the overlap search algorithm should not take more then $O(n)$ time to examine all of the inserted items. Hence $\sum l_i/s = O(n)$, which immediately leads to

$$s = O\left(\sum_{i=1}^{n} \frac{l_i}{n}\right) \tag{9.3.9}$$

being quite obviously the average interval length. This simple result was confirmed experimentally by Teschner *et al.* [**203**]. A more comprehensive analysis has been included in Section 9.3.3.3. Figure 9.3.5 illustrates an exemplary outcome of the one-dimensional hashing.

Once the intervals have been inserted into the hash table, the overlap detection can be performed. The following simple algorithm can be employed

ALGORITHM 9.3.9. *Hash_1D_Detect* $(h, m, d, \delta, Report)$
1 ***for*** $i = 1$ ***while*** $i \leq m$ ***do***
2 *One_Way_Scan* $(h[i], d, 0, \delta, Report)$
3 ***end for***

Algorithm 9.3.9 employs scanning for each of the hash lists (line 2). If the voxel size has been selected according to (9.3.9), then the total length of lists $\sum_i |h[i]| = O(n)$. Note that $tc = 0$ and the merge sort for each list takes $O(|h[i]| \log |h[i]|)$. Further, $\sum_i |h[i]| \log |h[i]| \leq n \log n$, so that the total cost of sorting the partial lists is $O(n \log n)$. The fact that overlapping intervals can be hashed into several distinct lists (Figure 9.3.5), results in the possibility of multiple intersection reports for the same pair of intervals. The repeated reports ought to be suppressed, when detecting the overlaps. This requires an additional computational effort, and hence the spatial hashing has no advantage over scanning in one dimension. A convenient way of avoiding the repeated reports will be detailed in Section 9.3.5.

Due to the action of the hashing function (9.3.7), an interval travelling over adjacent voxels can be mapped into arbitrary entries of the hash table. The coherence of hash lists is thus not preserved and the advantage of it cannot be taken.

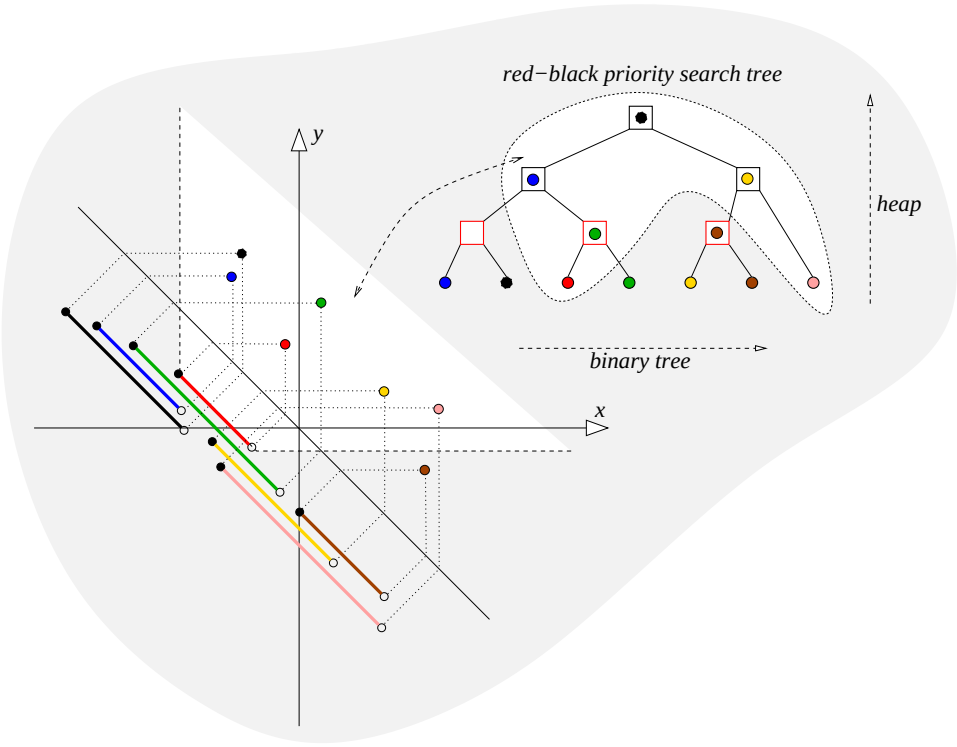


FIGURE 9.3.6. Intervals mapped into two-dimensional points and the related priority-search tree structure.

9.3.2.4. *Using priority search tree (dynamic).* There exists a curious mapping between the one-dimensional intervals and the two-dimensional points, making it possible to apply the priority-search tree as the solver to the dynamic interval overlap problem [**184**]. This is illustrated in Figure 9.3.6. Consider a set of intervals $[lo_i, hi_i]$, $i = 1, 2, ..., n$. It is easy to notice, that an interval $[lo_j, hi_j]$ intersects $[lo_i, hi_i]$ if and only if

$$hi_i > lo_j \wedge lo_i < hi_j \tag{9.3.10}$$

Note symmetry with respect to i and j. One can rewrite (9.3.10) as

$$hi_i > lo_j \wedge -lo_i > -hi_j \tag{9.3.11}$$

which implies, that the change of coordinates

$$\begin{cases} x_i = hi_i \\ y_i = -lo_i \end{cases} \quad \begin{cases} x_{0j} = lo_j \\ y_{0j} = -hi_j \end{cases} \tag{9.3.12}$$

allows to formulate the interval intersection problem as the two-sided range query Problem 9.1.13. The priority search tree, introduced in Section 9.1.7, solves Problem 9.1.13 and hence the dynamic interval intersection problem in optimal space and time. The following three routines make use of (9.3.12) and employ intervals related to $d-$projections of bounding boxes.

ALGORITHM 9.3.10. *Pst_1D_Insert* (t, d, b)
1 *Pst_Insert* $(t, b.hi[d], -b.lo[d], b)$

ALGORITHM 9.3.11. *Pst_1D_Delete* (t, d, b)
1 *Pst_Delete* $(t, b.hi[d], -b.lo[d], b)$

ALGORITHM 9.3.12. *Pst_1D_Query* $(t, d, b, \delta, Report)$
1 *Pst_Query* $(t, b.lo[d], -b.hi[d], \delta, Report)$

Having a priority search tree t, one can then insert and delete intervals in $O(\log n)$ time (Algorithms 9.3.10 and 9.3.11). At any time it possible to find all overlaps between a given interval and all intervals stored in t in $O(\log n + k)$ time (Algorithm 9.3.12).

9.3.3. 2D rectangle overlap. A static rectangle intersection algorithm is outlined first. This will not be of direct use in the three-dimensional framework. Nevertheless, it allows to visualise a general computational technique known as line sweeping. A dynamic rectangle intersection problem is solved next. Four different variants of the dynamic data structure are investigated for that purpose. They are employed later in Section 9.3.6, where the sweeping algorithm is developed in three-dimensions.

9.3.3.1. *Line-sweep algorithm.* Sweeping is one of the classical techniques in computational geometry. Some exemplary developments related to general intersection problems in the plane include [**24, 159, 63**]. As already mentioned, Six and Wood [**192**] give an $O(n \log n + k)$ time and $O(n \log n)$ space algorithm for reporting k overlaps between n planar, axis-aligned rectangles. Few years later McCreight [**148**] defined the priority search tree structure and reduced the space complexity of the overlap detection algorithm to the optimal $O(n)$.

Let an auxiliary tuple (b, t, x) store the bounding box pointer b, the type $t \in \{low, high\}$, and the coordinate x. Let $u = (b, t, x)$ be called an endpoint. Let u, v be of type (b, t, x), and the comparison of endpoints read

$$\begin{array}{c} u < v \text{ iff } u.x < v.x \vee (u.x = v.x \wedge u.b.d < v.b.d) \\ u = v \text{ iff } u.x = v.x \wedge u.b.d = v.b.d \\ u > v \text{ otherwise.} \end{array} \tag{9.3.13}$$

Let l be a list all low and high endpoints of bounding boxes, that is a list made of the compound tuples $((b, t, x), p, n)$. Consider the set of related boxes in the $i \times j$ plane, where $1 \leq i \neq j \leq 3$. McCreight's approach can now be summarised in the following

ALGORITHM 9.3.13. *Sweep_2D* $(l, i, j, tc, \delta, Report)$
1 ***for*** $u = l$ ***while*** $u \neq nil$ ***do***
2 ***if*** $u.d.t = low$ ***then*** $u.d.x = u.d.b.lo[i]$
3 ***else*** $u.d.x = u.d.b.hi[i]$
4 ***end for***
5 ***if*** $tc > 0$ ***then*** *List_Insertion_Sort* (l)
6 ***else*** *List_Merge_Sort* (l)
7 $t = nil$
8 ***while*** $l \neq nil$
9 ***if*** $l.d.t = low$ ***then***
10 $\delta.b = l.d.b$
11 *Pst_1D_Query* $(t, j, l.d.b, \delta, Report)$
12 *Pst_1D_Insert* $(t, j, l.d.b)$
13 ***else***
14 *Pst_1D_Delete* $(t, j, l.d.b)$
15 ***end if***
16 $l = l.n$

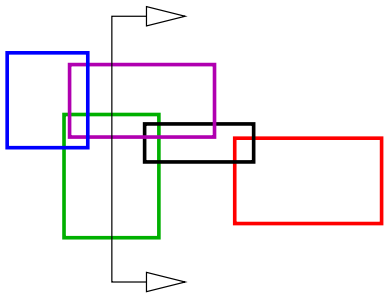


FIGURE 9.3.7. An example of the line-sweep approach. Two rectangles intersect the sweep line. The middle, black rectangle will be considered next and its overlaps with the green and purple rectangles will be detected.

17 ***end while***

In lines 1-4 above, the $i-$dimension aligned endpoints are updated to the current values of the relevant box coordinates. Then, in lines 5 and 6, either the insertion or the merge sorting is performed, where comparison (9.3.13) is in use. An empty priority search tree is initialised in line 7. Next, a loop over all endpoints is executed (lines 8-17). If the low endpoint is encountered, the priority search tree is queried with the $j-$dimension aligned extent of a box. All intersections between the box $l.d.b$ and the boxes stored in t, whose $j-$dimension extents intersect those of $l.d.b$, are reported (line 11). Then the interval is inserted into the tree (line 12). In case of the high endpoint, the interval is deleted from the tree (line 14). Note, that it is assumed that the auxiliary pointer δ has a vacant member pointer $\delta.b$, which used in line 10, so that the *Report* callback knows about the *pairs* of overlapping objects (being composed of $\delta.b$ and of the second argument of *Report*).

To bring up into the picture the actual *line* and the *sweeping* process, one should imagine a few axis aligned rectangles scattered over a plane. Sweeping a vertical line from the far left to the right allows to account for the rectangles currently being intersected by the line. Obviously, all of them must overlap along the horizontal direction. If one could now solve the interval overlap problem in the remaining, vertical direction - that would eventually reveal all pairs of overlapping rectangles. Now, it is enough to move the line from one endpoint to the other, as only at those points status change happen. In Algorithm 9.3.13, the sweep-line is symbolically represented by the current $l.d.x$ coordinate (position of the vertical line), and by the priority search tree t (storing rectangles currently intersected by the line). If a new rectangle is about to enter the line, one first looks for intersections along the vertical direction - this happens during the tree query in line 11. The rectangle is then simply inserted into the tree (line 12). As soon as its endpoint is reached (the vertical plane does not intersect it any more), it is removed from the tree t (line 14). Figure 9.3.7 gives an additional illustration.

The input list l has length $2n$ and its sorting takes $O(n \log n)$ time. The time coherence can be exploited and the list sort can eventually take $O(n)$ steps. Nevertheless, the priority search tree insertions/deletions and queries for all $2n$ endpoints can still respectively take $O(n \log n)$ and $O(n \log n + k)$ time in the worst case. One can thus only expect reduction of the constant factor in the $O(\cdot)$ notation in case of coherence. Optimistically however, if boxes are not too densely packed, only a fraction of them will be stored in t at a given moment. Then, in case of coherence, the expected runtime would be $O(n \log m + k)$, where $m \leq n$.

9.3.3.2. *Dynamic rectangle intersection.* Like in Section 9.3.2.4, the objective is to find a data structure facilitating insertions, deletions and queries corresponding to the rectangle overlap problem. Optimally, insertions and deletions should take $O(\log n)$ time, while the overlap queries should take $O(\log n + k)$ time. Unfortunately, to the best of our knowledge, such a structure has not yet been described. It seems that the closest result is due to Mortensen [**157**], where an $O(n \log n / \log\log n)$ space structure is proposed. However, it can only be applied indirectly, as it solves the dynamic orthogonal segment intersection problem. Furthermore, it is of purely theoretical interest, being too intricate for an implementation. To complement this example one should mention the paper by Samet [**185**], reviewing various rectangle indexing techniques. None of them is fully dynamic in the sense expected here. Also the so called box-trees, analysed by Agarwal *et al.* [**4**] are not dynamic and have a rather pessimistic $O(\sqrt{n} + k)$ query time. Facing the lack of a suitable structure, it remains to resort to an approximation. Four variants will be considered.

Two-dimensional hashing. The hashing function (9.3.7) from Section 9.3.2.3 needs to be redefined as

$$f : Z \times Z \to \{1, 2, ..., m\} \tag{9.3.14}$$

where a suitable example could be $f(i,j) = (i \cdot c \text{ xor } j \cdot d) \pmod m$, where c, d are large primes [**203**]. Assume also that the data pointed by the box tuple member $b.d$, has a spare *pointer* member $b.d.m$. Now the insertion/query routine can be phrased as

ALGORITHM 9.3.14. *Hash_2D_Insert* $(h, s, k_0, k_1, b, \delta, Report)$

```
1   i_0 = g(b.lo[k_0], s),  i_1 = g(b.hi[k_0], s)
2   j_0 = g(b.lo[k_1], s),  j_1 = g(b.hi[k_1], s)
3   for i = i_0 while i ≤ i_1 do
4     for j = j_0 while j ≤ j_0 do
5       flag = 0
6       for l = h[f(i,j)] while l ≠ nil do
7         if l.d.b = b then flag = 1,  l = nil
8         else if l.d.m ≠ b ∧ overlap(b, l.d, k_0, k_1) then
9           Report(δ, b, l.d)
10          l.d.m = b
11        end if
12        l = l.n
13      end for
14      if flag = 0 then
15        b.m = nil
16        List_Insert(h[f(i,j)], b)
17      end if
18      j = j + 1
19    end for
20    i = i + 1
21  end for
```

In the first two lines of Algorithm 9.3.14 the voxel index ranges (i_0, i_1) and (j_0, j_1) are determined. The $k_0 \times k_1$-rectangle of box b is covered by the voxels $(i,j) \in (i_0, i_1) \times (j_0, j_1)$. The double loop from lines 3,4 till 19, 21 iterates over all indices from that covering. The hash list corresponding to each $h[f(i,j)]$ is traversed in lines 6-13. If the box was already stored in the list, the loop is terminated and a $flag$ set up (line 7). Note, that due to the way items are inserted into the list

(Algorithm 9.1.4), b must have bean stored at the head element of the list. Hence, lines 8-10 could not be executed if $l.d.b = b$. Otherwise, the rectangles stored in the list are checked for not being *marked* ($l.d.m \neq b$), and eventually overlaps with b are reported (line 8). Before the overlap report between b and $l.d$, the box stored at l is *marked* in line 9. Marking allows to avoid repeated reports, when the same pairs of boxes occupy different hash lists. In case box b was not found in the current hash list (line 14), it is inserted into the list (line 16). Just before that, its marker pointer is set to *nil* (line 15), which ensures the correctness of marking. A much simpler deletion algorithm is given below. No comments seem necessary.

ALGORITHM 9.3.15. *Hash_2D_Delete* (h, s, k_0, k_1, b)
1 $i_0 = g(b.lo[k_0], s)$, $i_1 = g(b.hi[k_0], s)$
2 $j_0 = g(b.lo[k_1], s)$, $j_1 = g(b.hi[k_1], s)$
3 ***for*** $i = i_0$ ***while*** $i \leq i_1$ ***do***
4 ***for*** $j = j_0$ ***while*** $j \leq j_0$ ***do***
5 *List_Delete* $(h[f(i,j)], b)$
6 $j = j + 1$
7 ***end for***
8 $i = i + 1$
9 ***end for***

Two-dimensional hashing and priority search tree. This variant is similar to the previous one in that respect, that it still utilises the two-dimensional hashing. The difference is, that instead of the hash lists, the priority search trees are used at the $h[\cdot]$ entries of the hash table. This allows for a more intelligent filtering of overlaps (compared with Algorithm 9.3.14) and should improve efficiency for dense data sets. More comments will follow in Section 9.3.3.3. As the priority search tree query will be exploited, the following auxiliary callback needs to be defined.

ALGORITHM 9.3.16. *Aux_Pst_Callback* (α, b)
1 ***if*** $b.d.m = \alpha.b$ ***then return***
2 ***else if*** $b.hi[\alpha.i] \leq \alpha.b.lo[\alpha.i] \vee b.lo[\alpha.i] \geq \alpha.b.hi[\alpha.i]$ ***then return***
3 $\alpha.Report(\alpha.\delta, \alpha.b, b)$
4 $b.d.m = \alpha.b$

In the above $\alpha = (i, b, \delta, Report)$, where $1 \leq i \leq 3$, b is a box pointer, δ is a callback data pointer, and *Report* is the external callback routine. Note that the priority search tree callback used in Algorithm 9.1.23 naturally used two arguments, while for reporting overlap pairs the callback in line 3 uses three arguments. These are of course only technical details, of quite minor importance. We can now define the suitable insertion/query routine.

ALGORITHM 9.3.17. *Hash_2D_Pst_Insert* $(h, s, k_0, k_1, b, \delta, Report)$
1 $i_0 = g(b.lo[k_0], s)$, $i_1 = g(b.hi[k_0], s)$
2 $j_0 = g(b.lo[k_1], s)$, $j_1 = g(b.hi[k_1], s)$
3 ***for*** $i = i_0$ ***while*** $i \leq i_1$ ***do***
4 ***for*** $j = j_0$ ***while*** $j \leq j_0$ ***do***
5 *Pst_1D_Query* $(h[f(i,j)], k_0, b, (k_1, b, \delta, Report), Aux\_Pst\_Callback)$
6 *Pst_1D_Insert* $(h[f(i,j)], k_0, b)$
7 $j = j + 1$
8 ***end for***
9 $i = i + 1$
10 ***end for***

For each (i, j) voxel covering the $k_0 \times k_1$ rectangle of box b, the priority search tree stored at the hash table element $h[f(i,j)]$ is first queried with the

k_0-aligned interval (line 5, the choice of k_0 is arbitrary). Then the interval is inserted into the tree (line 6). Note that the tuple $(k_1, b, \delta, Report)$ and the callback $Aux\_Pst\_Callback$ are passed to the tree query routine in line 5. Then, if the k_0-dimension aligned intervals overlap, the auxiliary Algorithm 9.3.16 checks if this is not a repeated report (line 1), followed by the overlap test in k_1-dimension (line 2). If the k_1-dimensional intervals overlap and this is the first report, it is further reported in line 3, which is followed by marking the box stored in the tree (line 4), so that the repeated reports are avoided. Again, the deletion routine is simple and requires no comments.

ALGORITHM 9.3.18. *Hash_2D_Pst_Delete* (h, s, k_0, k_1, b)
1 $i_0 = g(b.lo[k_0], s)$, $i_1 = g(b.hi[k_0], s)$
2 $j_0 = g(b.lo[k_1], s)$, $j_1 = g(b.hi[k_1], s)$
3 ***for*** $i = i_0$ ***while*** $i \leq i_1$ ***do***
4 ***for*** $j = j_0$ ***while*** $j \leq j_0$ ***do***
5 $Pst\_1D\_Delete(h[f(i,j)], k_0, b)$
6 $j = j + 1$
7 ***end for***
8 $i = i + 1$
9 ***end for***

One-dimensional hashing and priority search tree. The approach from the previous paragraph might still appear somewhat exaggerated. After all, the priority search tree works optimally in one dimension and it does not seem to need the additional granularity of the two-dimensional hashing. Hence, one can hash the space along one dimension and use the tree along the other direction. The resultant code is an obvious simplification of Algorithms 9.3.17 and 9.3.18. It is given below without further comments.

ALGORITHM 9.3.19. *Hash_1D_Pst_Insert* $(h, s, j, k, b, \delta, Report)$
1 $i_0 = g(b.lo[j], s)$, $i_1 = g(b.hi[j], s)$
2 ***for*** $i = i_0$ ***while*** $i \leq i_1$ ***do***
3 $Pst\_1D\_Query(h[f(i)], j, b, (k, b, \delta, Report), Aux\_Pst\_Callback)$
4 $Pst\_1D\_Insert(h[f(i)], j, b)$
5 $i = i + 1$
6 ***end for***

ALGORITHM 9.3.20. *Hash_1D_Pst_Delete* (h, s, j, b)
1 $i_0 = g(b.lo[j], s)$, $i_1 = g(b.hi[j], s)$
2 ***for*** $i = i_0$ ***while*** $i \leq i_1$ ***do***
3 $Pst\_1D\_Delete(h[f(i)], j, b)$
4 $i = i + 1$
5 ***end for***

Priority search tree only. It remains to employ the priority search tree as the sole filtering strategy. This is obviously an abuse of its original purpose, although it will be nevertheless interesting to investigate the efficiency of this approach along with the previous ones. This however has to wait until Chapter 13. The suitable insertion/query and deletion routines are now the simplifications of Algorithms 9.3.19 and 9.3.20. They read

ALGORITHM 9.3.21. *Pst_2D_Insert* $(t, j, k, b, \delta, Report)$
1 $Pst\_1D\_Query(t, j, b, (k, b, \delta, Report), Aux\_Pst\_Callback)$
2 $Pst\_1D\_Insert(t, j, b)$

ALGORITHM 9.3.22. *Pst_2D_Delete* (t, j, b)
1 $Pst\_1D\_Delete(t, j, b)$

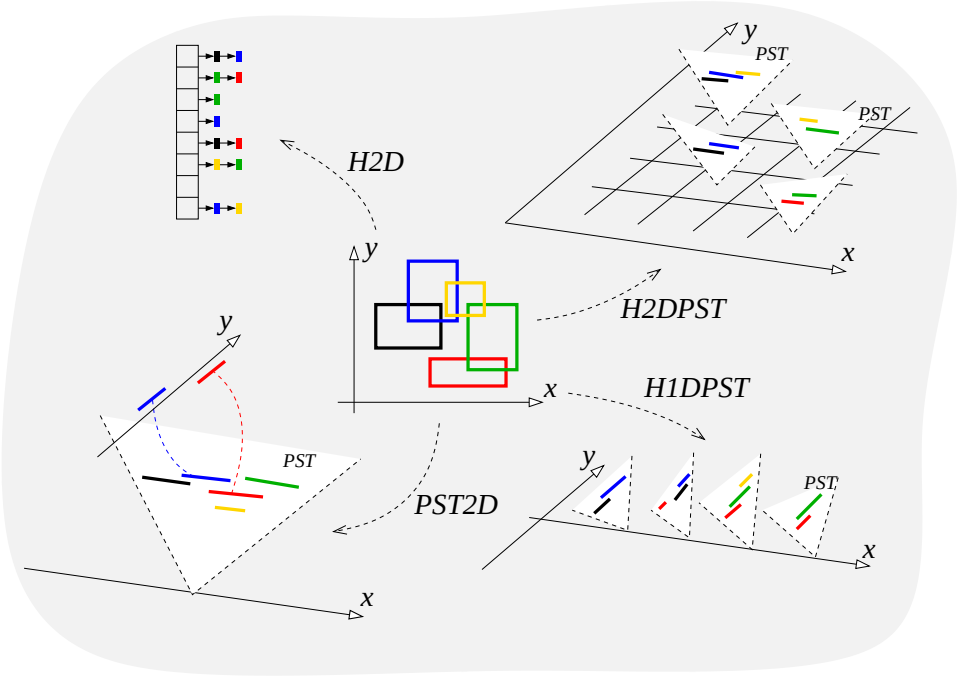


FIGURE 9.3.8. Four approximations of the dynamic rectangle intersection structure.

Common interface. Let us define a tuple (h, t), where h is a hash table, and t is a priority search tree. This will be briefly called the *dynamic rectangle* structure. Assume that $\alpha \in \{H2D, H2DPST, H1DPST, PST2D\}$ is a constant, and let $v = (h, t)$. It is convenient to define the following common interface for all four variants of the dynamic rectangle structure. The four variants of the structure have been visualised in Figure 9.3.8.

ALGORITHM 9.3.23. *Dynrect_Insert* $(\alpha, v, s, i, j, b, \delta, Report)$
1 **if** $\alpha = H2D$ **then** *Hash_2D_Insert* $(v.h, s, i, j, b, \delta, Report)$
2 **else if** $\alpha = H2DPST$ **then** *Hash_2D_Pst_Insert* $(v.h, s, i, j, b, \delta, Report)$
3 **else if** $\alpha = H1DPST$ **then** *Hash_1D_Pst_Insert* $(v.h, s, i, j, b, \delta, Report)$
4 **else if** $\alpha = PST2D$ **then** *Pst_2D_Insert* $(v.t, i, j, b, \delta, Report)$

ALGORITHM 9.3.24. *Dynrect_Delete* (α, v, s, i, j, b)
1 **if** $\alpha = H2D$ **then** *Hash_2D_Delete* $(v.h, s, i, j, b)$
2 **else if** $\alpha = H2DPST$ **then** *Hash_2D_Pst_Delete* $(v.h, s, i, j, b)$
3 **else if** $\alpha = H1DPST$ **then** *Hash_1D_Pst_Delete* $(v.h, s, i, b)$
4 **else if** $\alpha = PST2D$ **then** *Pst_2D_Delete* $(v.t, i, b)$

9.3.3.3. *Analysis of the dynamic rectangle structure.* It is not difficult to give the quite pessimistic, worst case performance estimates of the dynamic rectangle structure. Assuming that, among others, there is a hash table entry into which all of the n boxes will be mapped, one can readily obtain the bounds listed in Table 1. Nevertheless, upon a more careful study of relations between the shape of bodies, the density of their packing and the voxel size, significantly more realistic bounds can be obtained.

It should be noted, that the performance of the dynamic rectangle structure ought to be invariant with respect to rigid rotations of space. This is why a uniform voxel size s is employed along all spatial dimensions.

	$H2D$	$H2DPST$, $H1DPST$, $PST2D$
Insertion/query	$O\left(n^2+q\right)$	$O\left(\log n+q\right)$
Deletion	$O\left(n\right)$	$O\left(\log n\right)$
Space	$O\left(n\right)$	$O\left(n\right)$

TABLE 1. *Worst case complexity* of insertion/query and deletion for the dynamic rectangle structure. The number of pairs that need to be checked for intersections is $q=\Omega\left(k\right)$, which accounts for the necessity of avoiding repeated reports (k is the actual number of box intersections).

Recall the terminology introduced in Section 9.3.1. Let $\mathcal{P}=\{P_1,P_2,...,P_n\}$ be a set of objects, c_i be the core of P_i, and b_i be the enclosing box of P_i. Two ways of calculating s will be investigated

$$s=\left(\sum_{i=1}^{n}\frac{\mathrm{vol}\left(b_i\right)}{n}\right)^{1/d} \tag{9.3.15}$$

and

$$s=\left(\sum_{i=1}^{n}\frac{\mathrm{vol}\left(b_i\right)^{1/d}}{n}\right) \tag{9.3.16}$$

where d is the dimension of space. The following fact is useful to start up.

LEMMA 9.3.25. *Assume, that $\mathcal{P}$ is $\lambda-$low-density. If σ is the scale factor of $\mathcal{P}$, and each object in $\mathcal{P}$ has aspect ratio at most α, then the number of object intersections is $O\left(\sigma\lambda n\right)$, while the number of box intersections is $k=O\left(\alpha\sigma^{3/2}\lambda n\right)$.*

PROOF. There holds $\mathrm{vol}\left(b_i\right)\leq\sigma\mathrm{vol}\left(b_j\right)$. Let $j=\arg\min_i\mathrm{vol}\left(b_i\right)$. Each object can be covered by at most $O\left(\sigma\right)$ translations of b_j. Each such box can intersect at most λ objects and hence each object intersects at most $O\left(\sigma\lambda\right)$ others. Taking $K_o=O\left(\sigma\lambda n\right)$ in Theorem 9.3.2 gives $k=O\left(\alpha\sigma^{3/2}\lambda n\right)$.

□

Let $\alpha,\sigma,\lambda\ll n$ be small constants. Then Lemma 9.3.25 implies that $k=O\left(n^2\right)$ intersections cannot occur. If the hash table has size $m=O\left(n\right)$ and the hashing function has property (9.1.3), the worst case complexity corresponds to the dense cluster scenario, depicted in Figure 9.3.9.

Let us notice, that the axis aligned bounding box of object P_i is always contained within the enclosing box b_i. Thus, arguing about the enclosing boxes is more conservative than arguing about the bounding boxes.

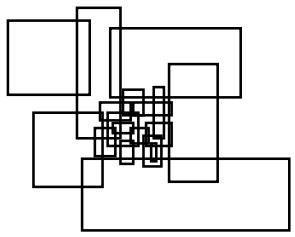


FIGURE 9.3.9. A dense cluster.

Take any $d-$dimensional $s \times s \times ... \times s$ cube u (L_∞ ball of radius s), such that $s \geq \min_i \text{radius}(b_i)$. Then $\text{vol}(b_i) \leq \sigma s^d$ and $\text{radius}(b_i) \leq \sigma^{1/d} s$. Hence, all of the boxes that can intersect u lay inside of the $u-$centred L_∞ ball v of radius $\left(1 + 2\sigma^{1/d}\right) s$. One can rewrite (9.3.15) as

$$\sum_{i \in IN(v)} \text{vol}(b_i) + \sum_{j \in OUT(v)} \text{vol}(b_j) = s^d n \tag{9.3.17}$$

where

$$IN(v) = \{i : b_i \subset v\} \tag{9.3.18}$$

$$OUT(u) = \{1, 2, ..., n\} \setminus IN(u) \tag{9.3.19}$$

The following result will be of use.

LEMMA 9.3.26. *There holds* $\sum_{i \in IN(v)} vol(b_i) = O\left(\alpha\sigma\lambda s^d\right)$.

PROOF. Consider $b_i : i \in IN(v)$. Let us split v regularly into sub-volumes v_j as long as there is no b_i, such that $\text{radius}(b_i) > \text{radius}(v_j)$. Each v_j overlaps $\lambda_j \leq \lambda$ objects P_i. Since $\text{vol}(b_i) \leq \alpha \text{vol}(c_i)$ and $\text{vol}(c_i) \leq \text{vol}(P_i)$, there follows $\sum_{i \in IN(v)} \text{vol}(b_i) \leq \alpha \sum_{i \in IN(v)} \text{vol}(P_i) \leq \alpha \sum_j \text{vol}(v_j) \lambda_j \leq \text{vol}(v) \alpha\lambda = \left(1 + 2\sigma^{1/d}\right)^d s^d \alpha\lambda = O\left(\alpha\sigma\lambda s^d\right)$. □

Obviously, the maximal number of elements of the index set $IN(v)$ corresponds to the worst case complexity. For any $i \in IN(v)$ and $j \in OUT(v)$ there holds $\text{vol}(b_j) \leq \sigma \text{vol}(b_i)$, and thus

$$\sum_{\text{any } j \in OUT(v)}^{t \text{ times}} \text{vol}(b_j) \leq \sigma \sum_{\text{any } i \in IN(v)}^{t \text{ times}} \text{vol}(b_i) \tag{9.3.20}$$

or specifically

$$l \sum_{j}^{n-l} \text{vol}(b_j) \leq \sigma (n-l) \sum_{i}^{l} \text{vol}(b_i) \tag{9.3.21}$$

hence

$$\sum_{j \in OUT(v)} \text{vol}(b_j) \leq \sigma \frac{n-l}{l} \sum_{i \in IN(v)} \text{vol}(b_i) \tag{9.3.22}$$

where $l = |IN(v)|$ is the number of elements of $IN(v)$. Due to Lemma 9.3.26, the last inequality can be summarised as

LEMMA 9.3.27. *There holds* $\sum_{j \in OUT(v)} vol(b_j) = O\left(\alpha\sigma^2\lambda s^d\right) \frac{n-l}{l}$, *where* $l = |IN(v)|$.

Equation (9.3.17), together with Lemmas 9.3.26 and 9.3.27 state $s^d n = O\left(\alpha\sigma\lambda s^d\right) + O\left(\alpha\sigma^2\lambda s^d\right) \frac{n-l}{l}$. In both Os the hidden constant is precisely 2^d, which allows to conclude that

$$l \leq \frac{\alpha\sigma^2\lambda n}{2^d n - \alpha\sigma\lambda + \alpha\sigma^2\lambda} \tag{9.3.23}$$

	H2D	*H2DPST, H1DPST*	*PST2D*
Insertion/query	$O\left(\alpha^2\sigma^5\lambda^2+q\right)$	$O\left(\sigma\log\left(\alpha\sigma^2\lambda\right)+q\right)$	$O\left(\log n+q\right)$
Deletion	$O\left(\alpha\sigma^3\lambda\right)$	$O\left(\sigma\log\left(\alpha\sigma^2\lambda\right)\right)$	$O\left(\log n\right)$

TABLE 2. *Refined complexity* of insertion/query and deletion for the dynamic rectangle structure. The number of pairs that need to be checked for intersections is $q=\Omega\left(k\right)$, which accounts for the necessity of avoiding repeated reports (k is the actual number of box intersections).

For $n\rightarrow\infty$ the above results in

$$l=O\left(\alpha\sigma^2\lambda\right) \tag{9.3.24}$$

The following overall estimates can be made.

THEOREM 9.3.28. *Let $\mathcal{P}=\{P_1,P_2,...,P_n\}$ be a set of objects, c_i be the core of P_i, and b_i be the enclosing box of P_i. Take $s=\left(\sum_{i=1}^n vol\left(b_i\right)/n\right)^{1/d}$. Assume, that $\mathcal{P}$ is $\lambda-$low-density, σ is the scale factor of $\mathcal{P}$, and each object in $\mathcal{P}$ has aspect ratio at most α. Then, the number of axis aligned bounding boxes intersecting an arbitrary $s\times s\times...\times s$ cube is $l=O\left(\alpha\sigma^2\lambda\right)$, while each box can intersect at most $r=O\left(\sigma\right)$ disjoint $s\times s\times...\times s$ cubes.*

PROOF. The $l-$estimate follows from the reasoning that led to (9.3.24), if only one can show that $\left(\sum_{i=1}^n \mathrm{vol}\left(b_i\right)/n\right)^{1/d}\geq\min_i \mathrm{radius}\left(b_i\right)$. But this implies $\sum_{i=1}^n \mathrm{vol}\left(b_i\right)/n\geq\min_i \mathrm{vol}\left(b_i\right)$ and of course the average is greater or equal to the minimum. The $r-$estimate follows from the fact that $\max_i \mathrm{vol}\left(b_i\right)/\min_i \mathrm{vol}\left(b_i\right)=\sigma$ and $s^d\geq\min_i \mathrm{vol}\left(b_i\right)$. □

In order to derive similar estimates, for the case when s is computed according to formula (9.3.16), it is convenient to assume that $\min_i \mathrm{vol}\left(b_i\right)=1$. There is no loss of generality, as it only a change of gauge is involved. Then

$$sn=\sum_{i=1}^n \mathrm{vol}\left(b_i\right)^{1/d}\leq\sum_{i=1}^n \mathrm{vol}\left(b_i\right) \tag{9.3.25}$$

and one can go along similar lines as before, in order to show that $l=O\left(\alpha\sigma^2\lambda s^{d-1}\right)$. At the same time $1\leq\mathrm{vol}\left(b_i\right)\leq\sigma$ implies that $1\leq\mathrm{radius}\left(b_i\right)\leq\sigma^{1/d}$ and thus $s\leq\sigma^{1/d}$. Hence $l=O\left(\alpha\sigma^{3-1/d}\lambda\right)$. The $r-$estimate is not affected.

Table 2 summarises the refined complexity estimates. Characteristically, due to the assumed density, all of the hashing based variants of the structure have operation times independent of the number of rectangles. Of course, these are still the worst case estimates, but this time expressed in terms of α,σ and λ. Intuitively the operations that will take that long, correspond to the largest and most distorted shapes. Due to its higher order presence, the scale factor σ plays the dominant role. In our case, of convex elements enclosed by boxes, the aspect ratio and density will usually be small constants, and the efficiency will be related to the disparity between the smallest and the largest element. In case of a uniform mesh, there follows $\alpha^2\sigma^5\lambda^2=O\left(1\right)$ and $\sigma\log\left(\alpha\sigma^2\lambda\right)=O\left(1\right)$, which indicates high efficiency.

9.3.4. The reference approach. An excellent, fast and practical algorithm for the box overlap problem was given by Zomorodian and Edelsbrunner [**216**]. In fact, it is fast enough to serve as the reference approach, against which efficiency

of the remaining developments is compared. As already mentioned, the core idea of this approach has been given by Edelsbrunner and Overmars [**67**]. It is based on solving the *batched* version of the box intersection problem, that is, querying some data structure with all boxes simultaneously. This way, at any time, only a part of the structure (being currently visited by the boxes) needs to be kept in memory. The technique is called *streaming*, and allows to reduce space demands of some otherwise unpractical structures.

Section 9.3.2.2 discusses an application of the segment tree (defined in Section 9.1.8) to the interval intersection problem. Basically a segment tree is build on a set of intervals and the queried with the low endpoints of the intervals. Assuming I to be the set of intervals and P to be the set of their low endpoints, one can apply streaming in the following way

ALGORITHM 9.3.29. *Stream_ 1D* $(I, P, lo, hi, \delta, Report)$
1 ***if*** $I = \emptyset \vee P = \emptyset$ ***then return***
2 $I_m = \{i \in I : [lo, hi) \subseteq i\}$
3 ***for*** $i \in I_m$, $p \in P$ ***do*** $Report\,(\delta, i, p)$
4 $mi =$*Approximate_ Median* $(P, h\,(P))$
5 $P_l = \{p \in P : p < mi\}$
6 $I_l = \{i \in I \setminus I_m : i \cap [lo, mi) \neq \emptyset\}$
7 *Stream_ 1D* $(I_l, P_l, lo, mi, \delta, Report)$
8 $P_r = \{p \in P : p \geq mi\}$
9 $I_r = \{i \in I \setminus I_m : i \cap [mi, hi) \neq \emptyset\}$
10 *Stream_ 1D* $(I_r, P_r, mi, hi, \delta, Report)$

In order to report all interval overlaps one calls *Stream_ 1D* $(I, P, -\infty, \infty, ...)$. In the second line, the set I_m of intervals stored at the current node of the segment tree is constructed. Note, that the current nodal interval is $[lo, hi)$, and I_m is composed of all members of I that contain it. A tree node is entered with the set of points P belonging to the nodal interval, hence points from P belong to intervals from I_m. That is, intervals from I_m and intervals corresponding to the points from P overlap. This is reported in line 3. The segment tree construction proceeds in line 4, where an approximate median of the point set is found. In [**216**] the algorithm proposed by Clarkson *et al.* [**44**] is employed. It reads

ALGORITHM 9.3.30. *Approximate_ Median* (P, h)
1 ***if*** $h = 0$ ***then return random*** $(p \in P)$
2 ***return median-of-three*** *(Approximate_ Median* $(P, h-1)$*,*
3 *Approximate_ Median* $(P, h-1)$*,*
4 *Approximate_ Median* $(P, h-1)$*)*

so that a ternary random tree of height h is build recursively, where $h\,(P) = O\,(\log |P|)$. Once the median mi has been computed, in line 4 of Algorithm 9.3.29, points P_l on the left from it and intervals I_l overlapping $[lo, mi)$ are selected (lines 5, 6). The left sub-tree is then build recursively in line 7. One can see, that the recursion continues until $I_l \neq \emptyset$ and $P_l \neq \emptyset$ (line 1). Once the left sub-tree walk is exhausted, the right sub-tree is analogously visited on the way back from the left-recursion (lines 8-9). All this takes $O\,(n \log n + k)$ time and $O\,(n)$ space.

Assume now that the sets I of intervals and P of low endpoints correspond to the $d-$projections of boxes from a set A. The one-dimensional streaming can solve the interval intersection problem along any of $d \in \{1, 2, 3\}$ dimensions. The basic insight allowing to solve the complete problem, is that the overlap reports in line 3 of Algorithm 9.3.29 can be replaced by streaming segment trees along the remaining directions. Hence the three-dimensional streaming would look like

ALGORITHM 9.3.31. *Stream_ 3D* $(I, P, lo, hi, d, \delta, Report)$
1 ***if*** $I = \emptyset \vee P = \emptyset$ ***then return***
2 $I_m = \{i \in I : [lo, hi) \subseteq i\}$
3 ***if*** $d = 1$ ***then for*** $i \in I_m$, $p \in P$ ***do*** $Report\,(\delta, i, p)$
4 ***else***
5 *Stream_ 3D* $(I_m, P, -\infty, \infty, d-1, \delta, Report)$
6 *Stream_ 3D* $(P, I_m, -\infty, \infty, d-1, \delta, Report)$
... *Rewrite lines 4-10 of Algorithm 9.3.29*
... *replacing "Stream_ 1D" with "Stream_ 3D"*
14 ***end if***

Calling *Stream_ 3D*$(A, A, -\infty, \infty, 3, ...)$ accounts for all of the box overlaps in $O\left(n \log^3 n + k\right)$ time and $O(n)$ space. It is implicitly assumed that for each call, sets I, P correspond to the $d-$projections of boxes from A. As boxes related to elements of I_m and P overlap along the direction d, it remains to check whether they overlap along the remaining directions. Hence, the $d-1$ dimensional sub-trees are traversed in lines 5, 6 (interval and point roles need to be exchanged in order to account for all possible overlaps). Only if all of the sub-trees have been checked $(d = 1)$ the box overlaps are reported (line 3).

In **[216]** the authors notice that streaming the complete segment tree is still too expensive. Although the $O\left(n \log^3 n + k\right)$ runtime seems satisfactory, the cost of recursive construction of the tree bears prohibitively high constant factors. A hybridisation technique based on one-dimensional scanning is proposed. Instead of building the complete tree, once the amount of objects drops below some *cutoff* value c, scanning is performed. Similarly, the tree construction is ceased at the lowest $d = 1$ level. Instead, again scanning is employed. The hybrid approach reads

ALGORITHM 9.3.32. *Hybrid_ 3D* $(I, P, lo, hi, d, \delta, Report)$
1 ***if*** $I = \emptyset \vee P = \emptyset$ ***then return***
2 ***if*** $d = 1$ ***then*** *OneWayScan* $(I, P, d, \delta, Report)$
3 ***if*** $|I| < c \vee |P| < c$ ***then*** *TwoWayScan* $(I, P, d, \delta, Report)$
4 ***else***
5 $I_m = \{i \in I : [lo, hi) \subseteq i\}$
6 *Hybrid_ 3D* $(I_m, P, -\infty, \infty, d-1, \delta, Report)$
7 *Hybrid_ 3D* $(P, I_m, -\infty, \infty, d-1, \delta, Report)$
... *Rewrite lines 4-10 of Algorithm 9.3.29*
... *replacing "Stream_ 1D" with "Hybrid_ 3D"*
15 ***end if***

The procedure *OneWayScan* sorts intervals from I and points from P, and scans the intervals with the points (along the dimension 1) reporting all encountered overlaps (this happens at the lowest level of the tree, so that intersections of intervals and points from I and P indicate box overlaps). The procedure *TwoWayScan* also sorts intervals from I and points from P along the dimension 1. It then performs a scan concurrently interchanging the roles of points and intervals so that all possible overlaps of intervals are discovered. For each such overlap, the remaining $2, ..., d$ overlap checks need to be performed before a conclusive box overlap report can be made.

9.3.5. Spatial hashing. Spatial hashing has been already discussed in detail in Sections 9.3.2.3, 9.3.3.2 and 9.3.3.3. Hence, without repeating the basic characteristics of this technique, it is sufficient to specify a data structure, compliant with the interface suggested in Section 9.2. Let $q = (s, d, frq, n, cur, out)$ store the size of voxel s, the dimension of scanning d (let $d = 0$ for a newly created q), the

frequency $frq \geq 1$ of updates of s, d, the number of stored boxes n, the list of currently stored boxes *cur*, and the list of boxes to be removed *out*. Let e be an element pointer. The following simple insertion routine can be implemented.

ALGORITHM 9.3.33. *Hash_3D_Insert* (q, e)
1 $adj = nil$, $lo = hi = [0, 0, 0]$
2 *List_Insert* $(q.cur, ((e, adj), lo, hi))$
3 $q.n = q.n + 1$
4 **return** $q.in$

The first line initialises some auxiliary variables. In the second line the $q.cur$ list appended by the bounding box of element e. Note, that the data field of the bounding box stores the tuple (e, adj) comprising the element and an adjacency list adj. The adjacency list stores pointers to bounding boxes overlapping the box of e. The insertion routine returns the head of the list, which contains the newly inserted data. The pointer to this list item is then backed up by the caller, and reused for fast deletion. The deletion routine follows below.

ALGORITHM 9.3.34. *Hash_3D_Delete* (q, i)
1 *List_Delete* $(q.cur, i)$
2 *List_Insert* $(q.out, i.d)$
3 $q.n = q.n - 1$

The pointer i above has been returned by the insertion Algorithm 9.3.33, and hence it can be directly employed in the list deletion call (line 1). In the next line, the bounding box pointer corresponding to the deleted data $(i.d)$ is being inserted into the $q.out$ list. This will be further exploited during an update, where all of the adjacent overlaps need to signalised as released. The update routine reads

ALGORITHM 9.3.35. *Hash_3D_Update* $(q, \delta, Created, Released)$
1 **for** $v \in q.out$ **do**
2 **for** $w \in v.d.d.adj$ **do**
3 $Released\,(\delta, v.d.d.e, w.d.d.e)$
4 *List_Delete_Data* $(w.d.d.adj, v.d)$
5 **end for**
6 **end for**
7 $q.out = nil$
8 **for** $v \in q.cur$ **do**
9 **for** $w \in v.d.d.adj$ **do**
10 **if no-overlap** $(v.d, w.d)$ **then**
11 $Released\,(\delta, v.d.d.e, w.d.d.e)$
12 *List_Delete_Data* $(w.d.d.adj, v.d)$
13 *List_Delete_Data* $(v.d.d.adj, w.d)$
14 **end if**
15 **end for**
16 **update-box** $(v.d)$
17 **end for**
18 **if** $q.d = 0 \vee$ **random**$(q.frq) = 0$ **then**
19 $q.s = \left(\sum_{v \in q.cur} vol\,(b\,(v.d.d.e))\,/q.n\right)^{1/3}$
20 $q.d = \arg\min_{d \in \{1,2,3\}} \left[\max_{v,w \in q.cur} \frac{v.d.hi[d] - v.d.lo[d]}{w.d.hi[d] - w.d.lo[d]}\right]$
21 **end if**
22 $i = \{1, 2, 3\} \setminus q.d$, $j = \{1, 2, 3\} \setminus \{q.d, i\}$, $\alpha = (i, j, \delta, Created)$
23 $h =$**hash-table** $(q.n)$
24 **for** $v \in q.cur$ **do**

```
25    lo_{i∈{1,2,3}} = g(v.d.lo[i], q.s),  hi_{i∈{1,2,3}} = g(v.d.hi[i], q.s)
26    for (i,j,k) ∈ [lo_1,..,hi_1] × [lo_2,..,hi_2] × [lo_3,..,hi_3] do
27       List_Insert (h[f(i,j,k)], v.d)
28    end for
29 end for
30 for i = 1 while i ≤ q.n do
31    One_Way_Scan (h[i], q.d, 0, α, Aux_Hash_Callback)
32    i = i + 1
33 end for
```

The first seventeen lines of Algorithm 9.3.35 correspond to the released overlaps search. In the first place (lines 1-6), all of the adjacent boxes pairs involving deleted elements are reported as released. The search is continued through the remaining adjacent pairs (lines 7-17), and if an overlap release is found between previously intersecting boxes (line 10), it is reported and the adjacency lists are updated accordingly (lines 11-13). By this occasion boxes extents are updated in order to bound the moving elements (line 16). In line 18, it is checked whether the spatial dimension $q.d$ was initialised (by definition $q.d = 0$ initially), or if a random number between 1 and $q.frq$ has been drawn (the randomisation serves the purpose of minimising constant factors of the algorithm, as frequent updates of $q.s$ and $q.d$ are not necessary in practise). In any of those cases, the voxel size $q.s$ is calculated according to formula (9.3.15) and the spatial dimension $q.d$ is selected. The choice of $q.d$ is such, that the $q.d$-dimensional scale factor of box related intervals is minimal. According to the analysis given in Section 9.3.3.3, along this dimension the number of voxels spanned by a single interval is minimal. In other words maximal $hi_d - lo_d$, computed in line 25, is minimised. This in turn is expected to increase the efficiency of scanning (line 31). In the meantime a tuple $\alpha = (i, j, \delta, Created)$ is prepared in line 22. Note that i, j are the remaining dimensions (different than $q.d$). An empty hash table of size $q.n$ is created in line 23. In the loop between lines 24 and 29, functions (9.3.8) and

$$f : Z \times Z \times Z \to \{1, 2, ..., q.n\} \tag{9.3.26}$$

are used in order to map the boxes into the hash table in the usual manner (Z is the set of integers). The hashing function employed here reads $f(i,j,k) = (i \cdot a \text{ xor } j \cdot b \text{ xor } k \cdot c) \pmod{q.n}$, where a, b, c are large primes [**203**]. Scanning along the dth dimension is performed next (line 31) for all hash lists. The temporal coherence is switched off (note, that the hash lists are created anew for each update).

Pairs of boxes overlapping along the dimension $q.d$ are reported to the auxiliary Algorithm 9.3.36. There, the first two lines execute simple rejection tests corresponding to the intersection along the remaining two dimensions. In line 3, it is checked whether the two boxes have not been already adjacent (the update routine reports only newly created overlaps). If this is not the case, the newly created box intersection is reported (line 4). If the report callback returns a positive code, the overlap is recorder in the adjacency lists (lines 5, 6). This leaves some flexibility to the user, who supplies the report callbacks (if the return value is semi-negative, the box overlap will be rediscovered the next time). For example, one might like to wait until a pair of elements whose boxes overlap becomes close enough, and leave the job of suggesting this pair to the overlap update algorithm.

ALGORITHM 9.3.36. *Aux_Hash_Callback* (α, a, b)

```
1  if a.hi[α.i] ≤ b.lo[α.i] ∨ a.lo[α.i] ≥ b.hi[α.i] then return
2  else if a.hi[α.j] ≤ b.lo[α.j] ∨ a.lo[α.j] ≥ b.hi[α.j] then return
```

	Hash_3D
Insertion	$O(1)$
Deletion	$O(1)$
Update	$O(n \log n + q)$
Space	$O(n)$

TABLE 3. Complexity of insertion, deletion and update for the three-dimensional hashing. The number of attempted overlap reports is $q = \Omega(k)$, where k is the actual number of box intersections.

```
3  else if List_Find_Data (a.d.adj, b) ≠ nil then return
4  if α.Created (α.δ, a.d.e, b.d.e) > 0 then
5     List_Insert (a.d.adj, b)
6     List_Insert (b.d.adj, a)
7  end if
```

The analysis of the above approach is quite straightforward. Space complexity is $O(n)$, as the assumed voxel size (Algorithm 9.3.35, line 19) guarantees that all of the elements can be covered by $O(n)$ voxels. Insertions and deletions obviously take $O(1)$ time. As to the update, the first seventeen lines of Algorithm 9.3.35 take $O(n+k)$ time, where k is the current number of box overlaps (there is $O(k)$ items in the adjacency lists). The lines 18-29 take $O(n)$ time. Sorting hash lists inside of the scan routine (line 31) takes

$$\sum_{i}^{n} m_i \log(m_i) \leq \sum_{i}^{n} m_i \log(n) = O(n \log n) \tag{9.3.27}$$

time, where m_i is the length of ith hash list. The q overlap reports correspond to the q calls of the auxiliary Algorithm 9.3.36, which takes constant time, provided the density of the element set is bounded (line 3, adjacency search). Repeated calls to the auxiliary routine are possible, so that $q = \Omega(k)$. In total the update takes $O(n \log n + q)$ time, where $q = \Omega(k)$. All this is summarised in Table 3.

9.3.6. Plane-sweep approach. Three-dimensional sweeping is simply an extension of the two-dimensional approach outlined in Section 9.3.3.1. In the current

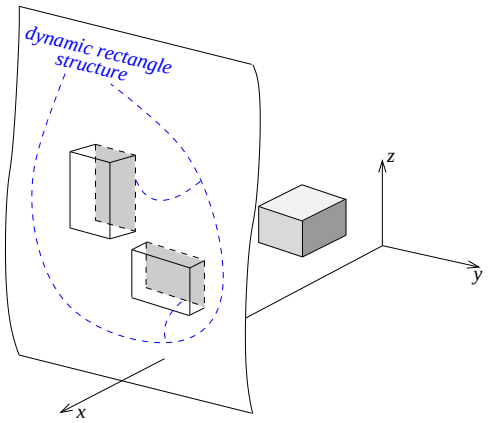


FIGURE 9.3.10. General idea of the three-dimensional plane sweeping.

case, instead of using the sweep-line and the dynamic interval intersection structure, a sweep-plane and a dynamic rectangle intersection structure are employed (Figure 9.3.10). This is why the dynamic rectangle structure was developed in Section 9.3.3.2. As the general idea should be already clear, it remains to specify a data structure, compliant with the interface suggested in Section 9.2. Let an auxiliary tuple (b, t, x) store the bounding box pointer b, the type $t \in \{low, high\}$, and the coordinate x. Let $u = (b, t, x)$ be called an endpoint. Let u, v by of type (b, t, x), and the comparison of endpoints read

$$\begin{array}{c} u < v \text{ iff } u.x < v.x \vee (u.x = v.x \wedge u.b.d < v.b.d) \\ u = v \text{ iff } u.x = v.x \wedge u.b.d = v.b.d \\ u > v \text{ otherwise.} \end{array} \tag{9.3.28}$$

Let $q = (pts, alg, s, d, frq, n, cur, in, out)$ store respectively the list of endpoints pts, the dynamic rectangle algorithm type $alg \in \{H2D, H2DPST, H1DPST, PST2D\}$, the size s of the two-dimensional voxel, the dimension d of the axis orthogonal to the sweep-plane (let $d = 0$ for a newly created q), the integer frequency $frq \geq 1$ of internal updates of s and d, the number of all stored boxes n, the list cur of currently stored boxes, the list of newly inserted boxes in, and the list of recently deleted boxes out. The element insertion routine reads

ALGORITHM 9.3.37. *Sweep_ 3D_ Insert* (q, e)
1 $b = (nil, [0, 0, 0], [0, 0, 0])$, $adj = nil$
2 $l = (b, low, 0)$, $h = (b, high, 0)$
3 $b.d = (e, adj, l, h)$
4 *List_ Insert* $(q.in, b)$
5 $q.n = q.n + 1$
6 ***return*** $q.in$

A bounding box placeholder and an empty adjacency list are prepared in the first line above. Two new endpoints are the created and linked with the box (line 2). Finally box data is pointed at the tuple (e, adj, l, h), composed of the element pointer e, the list of boxes overlapping the bounding box of e, the low endpoint pointer l, and the high endpoint pointer h. The box is inserted into the list in, and the newly created list item is returned to the caller (in order to facilitate fast deletion). The deletion routine follows below.

ALGORITHM 9.3.38. *Sweep_ 3D_ Delete* (q, i)
1 *List_ Delete* $(q.cur, i)$
2 *List_ Delete* $(q.pts, i.d.d.l)$
3 *List_ Delete* $(q.pts, i.d.d.h)$
4 *List_ Insert* $(q.out, i.d)$
5 $q.n = q.n - 1$

The input pointer i corresponds to an element of the list $q.cur$ (returned by Algorithm 9.3.37). It is deleted from the list in the first line above. Next, in lines 2 and 3, the endpoint pointers stored at the data tuple of the bounding box $i.d$ are used to remove the endpoints from the list $q.pts$. The list item is scheduled for final deletion by placing in the $q.out$ list (line 4). This will be further used in the update routine, where the overlaps released due to the deletion will be reported. The update routine can be summarised as follows

ALGORITHM 9.3.39. *Sweep_ 3D_ Update* $(q, \delta, Created, Released)$
... Repeat lines 1-17 of Algorithm 9.3.35
18 $t = nil$
19 ***for*** $v \in q.in$ ***do***

```
20    update-box (v.d)
21    List_Insert (t, v.d.d.l)
22    List_Insert (t, v.d.d.h)
23    List_Insert (q.cur, v.d)
24  end for
25  q.in = nil
26  d = q.d
27  if d = 0 ∨ random(q.frq) = 0 then
28    d =Aux_Sweep_Direction (q.cur, q.n)
29    q.s = ( Σ_{v∈q.cur} vol (b_{⊥d−dim} (v.d.d.e)) /q.n )^{1/2}
30  end if
31  for v ∈ q.cur do
32    v.d.d.l.x = v.d.lo [q.d], v.d.d.h.x = v.d.hi [q.d]
33  end for
34  List_Merge_Sort (t)
35  if q.d = d then List_Insertion_Sort (q.pts)
36  else q.d = d, List_Merge_Sort (q.pts)
37  u = q.pts, w = nil
38  while t ≠ nil do
39    while u ≠ nil ∧ u.d < t.d do
40      if w ≠ u.p then w.n = u, u.p = w
41      w = u, u = u.n
42    end while
43    if w = nil then q.pts = t, t.p = nil
44    else if w ≠ t.p then w.n = t, t.p = w
45    w = t, t = t.n
46  end while
44  i = {1, 2, 3} \ d, j = {1, 2, 3} \ {d, i}, α = (δ, Created)
45  h =hash-table (q.n), dr = (h, nil)
46  for u = q.pts while u ≠ nil do
47    if u.d.t = low then
48      Dynrect_Insert (q.alg, dr, q.s, i, j, u.d.b, α, Aux_Sweep_Callback)
49    else
50      Dynrect_Delete (q.alg, dr, q.s, i, j, u.d.b)
51    end if
52    u = u.n
53  end for
```

The first seventeen lines of Algorithm 9.3.39 are the same as in the case of three-dimensional hashing. They correspond to the released overlap detection and has been already commented on. In lines 18-24 the newly inserted boxes are updated and the list t of corresponding endpoints is created. The new boxes are transferred to the list of current boxes (line 23), and their original list is emptied (line 25). The sweep direction $q.d$ and the size of a two-dimensional voxel $q.s$ are updated in lines 26-30. This happens with a user specified probability of $1/q.frq$ as frequent updates are not practical (configuration of boxes does not change much from one update to another). It should be noted that $b_{\perp d-dim}(\cdot)$ denotes the two-dimensional enclosing box in the plane orthogonal to the dth dimension (line 29). The coordinates in the endpoints list $q.pts$ are updated in lines 31-33. Next, the list of newly created endpoints is sorted (without coherence, line 34). The list of old endpoints $q.pts$ is sorted using coherence, if only the sweep dimension $q.d$ has not just changed (line 35). Otherwise, the merge sort is employed (line 36). Comparison (9.3.28)

	Sweep_ 3D
Insertion	$O(1)$
Deletion	$O(1)$
Worst case update	$O(n(\log n + \beta) + m \log m + q)$
Typical update	$O(\beta n + q)$
Space	$O(n)$

TABLE 4. Complexity of insertion, deletion and update for the three-dimensional sweeping. The number of attempted overlap reports is $q = \Omega(k)$, where k is the actual number of box intersections. The coefficient $\beta \in \{\alpha^2\sigma^5\lambda^2, \sigma \log(\alpha\sigma^2\lambda), \log n\}$ depends on the variant of the employed dynamic rectangle structure. Integer m corresponds to the number of new element insertions, preceding the update.

is employed in sorting routines from lines 34-36. In lines 37-46 the two lists t and $q.pts$ are merged in linear time (the ordering is preserved). In lines 44-53 the final plane-sweep is performed. The dynamic rectangle structure is used in order to solve the two-dimensional sub-problem (lines 48, 50). The user specified variant of the algorithm is employed ($q.alg$). The auxiliary sweep callback is necessary in order to filter out already adjacent box pairs. In the first line of Algorithm 9.3.40 the adjacency based rejection test is performed. The callback routine follows exactly the already discussed Algorithm 9.3.36.

ALGORITHM 9.3.40. *Aux_ Sweep_ Callback* (α, a, b)
1 ***if*** *List_ Find_ Data* $(a.d.adj, b) \neq nil$ ***then return***
2 ***if*** $\alpha.Created(\alpha.\delta, a.d.e, b.d.e) > 0$ ***then***
3 *List_ Insert* $(a.d.adj, b)$
4 *List_ Insert* $(b.d.adj, a)$
5 ***end if***

Some comments about the selection of the sweep dimension are in place here. One could argue that the sweep should take place along the direction of the maximal one-dimensional scale factor. This would minimise the scale factor in the remaining two dimensions and hence improve the efficiency of the dynamic rectangle structure. Nevertheless, it is easy to see that for a case as simple as a set of uniform cubes this criterion is not conclusive. The one-dimensional scale factors are equal, although one would preferably sweep along the most elongated dimension of the box set. In result, a smaller number of objects would be stored in the dynamic rectangle structure at any time. Hence, the number of unnecessary overlap checks would decrease (the constant in the $q = \Omega(k)$ notation would be smaller). Algorithm 9.3.41 selects a dimension along which, on average, the largest number of boxes can be packed. If the density of packing is bounded, this dimension is likely to be orthogonal to the planes cutting through relatively small amount of boxes. Thus, storing as few boxes as possible in the dynamic rectangle structure is encouraged.

ALGORITHM 9.3.41. *Aux_ Sweep_ Dimension* (cur, n)
1 $\gamma_{i\in\{1,2,3\}} = \max_{v,w\in cur}[v.d.hi[i] - w.d.lo[i]]$
2 $\alpha_{i\in\{1,2,3\}} = \sum_{v\in cur}[v.d.hi[i] - v.d.lo[i]]/n$
3 $d = \arg\max_{i\in\{1,2,3\}}[\gamma_i/\alpha_i]$
4 ***return*** d

Complexity of three-dimensional sweeping is summarised in Table 4. The worst case update scenario happens when the dimension of sweeping is changed (e.g.

after the first run). This eventually happens with a user specified frequency, and does not correspond to a typical run. Even though the m newly inserted boxes always enforce the $O(m \log m)$ sort of the endpoints, typically $m \ll n$ and this term can be neglected. Hence a typical update time is $O(\beta n + q)$, where $\beta \in \left\{\alpha^2\sigma^5\lambda^2, \sigma \log\left(\alpha\sigma^2\lambda\right), \log n\right\}$ and $q = \Omega(k)$. For a set of elements with α, σ, λ being small constants this runtime quite tightly approximates the optimal $O(n + k)$ one.

9.4. Finding points and normals

Once a pair of elements likely to intersect has been identified, it remains to extract the contact point and the normal direction. It has been quite arbitrarily decided here, that *a single oriented contact point results from an overlap of two surface elements* (Definition 9.2.1). This is motivated by two factors:

(1) The point and the normal direction derived from an overlap of two elements are well defined for nonsmooth geometry.
(2) We wish to use as few contact points as possible, but still be able to control the accuracy of contact resolution by mesh refinement.

Two elements are in contact if their intersection is not empty. The intersection is d-dimensional, where $d \in \{0, 1, 2, 3\}$. Only the 3-dimensional, volumetric intersection is considered. The remaining cases are cast into the volumetric one through a simple regularisation. Assume, that two elements e_1 and e_2 overlap like in Figure 9.4.1. Their intersection $o = e_1 \cap e_2$ is a convex polyhedron, with the surface containing two parts $\partial o_1 \cup \partial o_2 \subset \partial o$, where $\partial o_k \subset \partial e_k$ and $\partial o_k = \partial o \cap \partial \mathcal{B}_k$. For each part, one can compute the resultant normal

$$\bar{\mathbf{n}}_k = \int_{\partial o_k} \mathbf{n} da \tag{9.4.1}$$

and the variation of normal

$$\tilde{\mathbf{n}}_k = \int_{\partial o_k} (\mathbf{n} - \bar{\mathbf{n}}_k)^2 da \tag{9.4.2}$$

The final outward normal is the one with a smaller variation

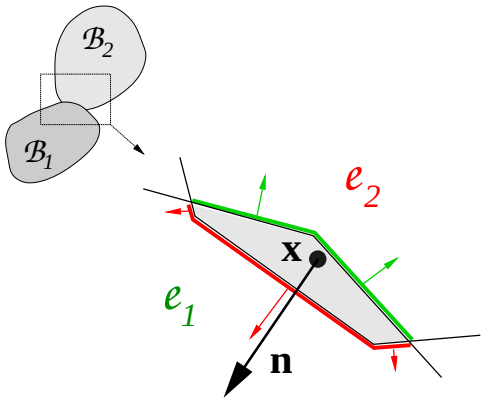


FIGURE 9.4.1. A contact point $\mathbf{x}$ and a normal $\mathbf{n}$ extracted from the intersection of two convex elements.

$$\mathbf{n}(e_1 \cap e_2) = \frac{\bar{\mathbf{n}}_k}{\|\bar{\mathbf{n}}_k\|}, \quad k = \arg\min_i (\|\tilde{\mathbf{n}}_i\|) \tag{9.4.3}$$

The contact point can be computed as the mass centre of the surface part with the larger variation of normal (hence, it is a deeper submerged point of the two possibly computed this way)

$$\mathbf{x}(e_1 \cap e_2) = \frac{\int_{\partial o_k} \mathbf{x} da}{\int_{\partial o_k} da}, \quad k = \arg\max_i (\|\tilde{\mathbf{n}}_i\|) \tag{9.4.4}$$

If elements e_1 and e_2 touch without an overlap, they are extended by a small margin: $e_k = e_k + B(0, \epsilon)$, where $B(0, \epsilon)$ is the zero centred ball of radius ϵ. The epsilon should be several orders of magnitude smaller than the shortest edge in e_k. Note, that ∂o_k are piecewise flat, and hence the above evaluations are trivial.

It remains to discuss the computation of $e_1 \cap e_2$. A simple, brute-force method could be described as follows

ALGORITHM 9.4.1. *Simple_Element_Intersection* (e_1, e_2)
1 $i = 1$, $j = 2$
2 *copy surface faces of* e_k *into* s_k *for* $k \in \{1, 2\}$
3 ***for each*** *face* $f \in s_i$ ***do***
4 ***for each*** *half-plane* h *bounding* e_j ***do***
5 *trim* f *with* h *so that* $f = f \cap h$
6 ***end do***
7 ***end do***
8 ***if*** $i = 1$ ***then*** $i = 2$, $j = 1$, ***goto*** 3
9 ***return*** $s_1 \cup s_2$

Scraps of the faces in $s_1 \cup s_2$ form the boundary of the intersection (note, that it might be empty). The method takes $O(nm)$ time where n is the number of faces in e_1 and m is the number of faces in e_2. For shapes as simple as the elements this might seem acceptable. However, this can only be verified by comparison with a more elaborate method. In this respect, the method by Müller and Preparata [**158**] has been implemented. The basic idea relies on the polarity of convex sets (cf. Rockafellar and Wets [**183**, p. 490]). For a convex set C such that $0 \in C$, the *polar* of C is the set

$$C^o = \{\mathbf{v} : \langle \mathbf{v}, \mathbf{x} \rangle \leq 1 \text{ for all } \mathbf{x} \in C\} \tag{9.4.5}$$

which is a convex and closet set. The *bipolar* of C is the set

$$C^{oo} = (C^o)^o = \{\mathbf{x} : \langle \mathbf{v}, \mathbf{x} \rangle \leq 1 \text{ for all } \mathbf{v} \in C^o\} \tag{9.4.6}$$

and $C^{oo} = C$, when C is closed (which is assumed here). For two sets C and D, their intersection and sum respectively read

$$C \cap D = \{\mathbf{x} : \mathbf{x} \in C \text{ and } \mathbf{x} \in D\} \tag{9.4.7}$$

$$C \cup D = \{\mathbf{x} : \mathbf{x} \in C \text{ or } \mathbf{x} \in D\} \tag{9.4.8}$$

If both C and D are convex, so is their intersection. Assume now, that $\mathbf{0} \in C \cap D$ and let

$$E = \mathrm{co}\,(C^o \cup D^o) \tag{9.4.9}$$

where

$$\mathrm{co}\ A = \left\{ \sum_{i=0}^{p} \lambda_i \mathbf{x}_i : \mathbf{x}_i \in A, \lambda_i \geq 0, \sum_{i=0}^{p} \lambda_i = 1, p \geq 0 \right\} \tag{9.4.10}$$

is the *convex hull* of a set. In particular, for any $\lambda \in [0,1]$ there holds

$$\lambda \mathbf{v}_C + (1-\lambda)\,\mathbf{v}_D \in E \text{ for all } \mathbf{v}_C \in C^o \text{ and } \mathbf{v}_D \in D^o \tag{9.4.11}$$

The polar set of E can now be defined as

$$E^o = \{\mathbf{x} : \langle \lambda \mathbf{v}_C + (1-\lambda)\,\mathbf{v}_D, \mathbf{x} \rangle \leq 1 \text{ for all } \mathbf{v}_C \in C^o, \mathbf{v}_D \in D^o, \lambda \in [0,1]\} \tag{9.4.12}$$

Summarising

$$\begin{aligned} \lambda \langle \mathbf{v}_C, \mathbf{x} \rangle + (1-\lambda) \langle \mathbf{v}_D, \mathbf{x} \rangle &\leq 1 \text{ for all } \mathbf{v}_C \in C^o \text{ and } \mathbf{v}_D \in D^o \\ \langle \mathbf{v}_C, \mathbf{x} \rangle &\leq 1 \text{ for all } \mathbf{x} \in C \\ \langle \mathbf{v}_D, \mathbf{x} \rangle &\leq 1 \text{ for all } \mathbf{x} \in D \end{aligned} \tag{9.4.13}$$

The first inequality in (9.4.13) can hold only, if the two remaining ones do as well. Otherwise, one can always pick $\mathbf{x} \in C$, $\mathbf{x} \notin D$ and for $\lambda = 0$ obtain $\langle \mathbf{v}_D, \mathbf{x} \rangle > 1$. Hence, E^o is composed of points $\mathbf{x} \in C \cap D$, or in other words

$$C \cap D = (\mathrm{co}\,(C^o \cup D^o))^o \tag{9.4.14}$$

The last formula is the departure point for the algorithm given by Müller and Preparata [**158**]. There are however, two stumbling blocks on the way towards its realisation. First of all, we have assumed that $\mathbf{0} \in C \cap D$. In practise, this means that one has to find a point belonging to the intersection (then it is easy to change coordinates, so that it is $\mathbf{0}$). A technique for that had been discussed in [**158**], although ten years later Gilbert *et al.* [**73**, 1988] proposed a more elegant and simpler method (Section 9.4.1). The second obstacle is related to the computation of the convex hull in (9.4.14). Müller and Preparata reference an algorithm given in [**173**]. Again, in our implementation a newer and simpler method by Barber *et al.* [**17**] is employed (Section 9.4.2).

It might seem, like the actual polarisation of a set $C \to C^o$ is also computationally nontrivial. Fortunately, for polyhedral convex sets this is not so. For any particular representation of a convex polyhedron C with n faces (an element in our case), it is easy to compute a set of planes such that

$$\mathbf{x} \in C \Leftrightarrow \langle \mathbf{n}_i, \mathbf{x} \rangle \leq 1 \text{ for all } i \in \{1, 2, ..., n\} \tag{9.4.15}$$

where $\mathbf{n}_i$ are the face normals (not necessarily unit). From the analogy between (9.4.5) and (9.4.15) it is clear, that normals $\mathbf{n}_i$ correspond to the vertices of C^o (at most, convex combinations of normals fulfil $\langle \sum_i \lambda_i \mathbf{n}_i, \mathbf{x} \rangle \leq 1$), so that

$$C^o = \mathrm{co}\,\{\mathbf{n}_1, \mathbf{n}_1, ..., \mathbf{n}_n\} \tag{9.4.16}$$

We can now to bring up a data structure, convenient for both storing and polarising convex polyhedrons. It was given in [**158**] under the name of the *doubly*

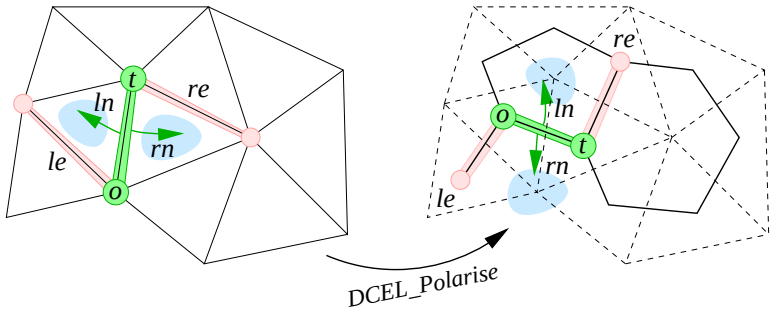


FIGURE 9.4.2. A planar graph of some convex polyhedron C and an edge in the DCEL structure. On the right, the corresponding edge in C^o is given.

connected edge list (DCEL). A tuple describing the list element can be expressed as (o, t, ln, rn, le, re), where o is a pointer to the origin vertex of the edge, t is a pointer to the terminus vertex of the edge, ln points to the normal of the face on the left from the edge, rn points to the similar normal on the right, le points to the next counter-clockwise edge around the origin, and re points to the next counter-clockwise edge around the terminus. It is implicitly assumed, that the normals stored in the data structure are such, that the condition (9.4.15) holds true. The data structure is illustrated in Figure 9.4.2. Apart from its concise format, the utility of the structure stems from the fact, that the polarisation procedure takes the following simple form

ALGORITHM 9.4.2. *DCEL_ Polarise* (d, k)
1 ***for*** $i = 1$ ***while*** $i \leq k$ ***do***
2 $o = d[i].o,\ t = d[i].t$
3 $d[i].o = d[i].ln,\ d[i].t = d[i].rn$
4 $d[i].ln = t,\ d[i].rn = o$
5 $i = i + 1$
6 ***end for***
7 ***return*** d

where $d[\cdot]$ is a table of k edges of a polyhedron. In the above procedure the edge pointers le and re in DCEL need not to be altered, although one needs to keep in mind that le and re pointers correspond now to the next ***clockwise*** edges around respectively the origin and the terminus (Figure 9.4.2). The algorithm for computing an intersection between two elements can now be given as

ALGORITHM 9.4.3. *Fast_ Element_ Intersection* (e_1, e_2)
1 $(\mathbf{p}, \mathbf{q}) =$*GJK* (e_1, e_2)
2 ***if*** $\|\mathbf{p} - \mathbf{q}\| > 0$ ***then return*** $\emptyset$
3 *assuming* $\mathbf{0} \equiv \mathbf{p}$, *compute* $\mathbf{n}_i$, $i \in \{1, 2, ..., n + m\}$, *representing* e_1 *for* $i \leq n$ *and* e_2 *for* $n < i \leq n + m$ *by* $\langle \mathbf{n}_i, \mathbf{x} \rangle \leq (1 + \epsilon)$
4 $(d, k) =$*Quickhull* $(\{\mathbf{n}_i\})$
5 ***return*** *DCEL_ Polarise* (d, k)

In the first line of Algorithm 9.4.3 the Gilbert-Johnson-Keerthi procedure is used in order to compute a pair of closest points $p \in e_1$, $q \in e_2$ (Section 9.4.1). If the distance between the elements is nonzero, an empty set is returned in the next line. Otherwise, $\mathbf{p} = \mathbf{q}$ and the coordinates are suitably changed, so that the zero point $\mathbf{0} \equiv \mathbf{p}$ (line 3). The representation (9.4.15) is computed for both elements, where $(1 + \epsilon)$ is used on the right hand side, in order to account for the

regularisation mentioned at the beginning of this section (GJK usually returns a point on the boundary of the element intersection). The *Quickhull* routine takes as an argument a set of vertices and returns the table of DCEL edges of their convex hull (Section 9.4.2). All of the normals are passed as the argument, which accounts for the union of the polar sets $C^o \cup D^o$ in (9.4.14). Finally, the returned hull is polarised in line 5, which corresponds to the outer-most operation of (9.4.14). In practice, the runtime of Algorithm 9.4.3 is close to $O(n \log n)$, where n is the maximum of the sums of node and face counts in e_1 and e_2. The exact theoretical bound however needs yet to be done, as the complexity of the GJK algorithm has not been thoroughly investigated in the literature (to our knowledge).

It ought to be stressed, that the fragments ∂o_k of the surface of $o = e_1 \cap e_2$ used in the evaluation of (9.4.1-9.4.4) should correspond only to the *surface faces* of the elements. More precisely $\partial o_k = \partial o \cap \partial \mathcal{B}_k$, where $\mathcal{B}_k$ is the body whose discretisation comprises e_k. This way the inner faces of elements, that is those that separate elements within a mesh, are not accounted for in the computations. The filtering is easily implemented, although the details have been omitted so to avoid an unnecessary clutter.

9.4.1. Finding a common point. Gilbert, Johnson and Keerthi [**73**] gave a very elegant and efficient method for finding a pair of points $\mathbf{p} \in C$ and $\mathbf{q} \in D$, such that $\|\mathbf{p} - \mathbf{q}\|$ is minimal, where C and D are convex. The algorithm is only outlined here, and it is noted that in the implementation the papers by Cameron [**35**] and Van den Bergen [**207**] were also helpful. The basic insight here is, that instead of looking for $\mathbf{p} \in C$ and $\mathbf{q} \in D$ minimising $\|\mathbf{p} - \mathbf{q}\|$, it might be more convenient to look for $\mathbf{v} \in C - D$ minimising $\|\mathbf{v}\|$. The set $C - D$ is not explicitly computed, but rather it is approximated by a series of simplices contained in it, and located successively closer to the origin. The GJK algorithm can be specified along the lines of [**207**] as follows

ALGORITHM 9.4.4. *GJK* (e_1, e_2)

1 $C =$ *vertices-of* (e_1), $D =$ *vertices-of* (e_2)
2 $W = \emptyset$, $\mu = 0$, $\mathbf{v} =$ *any-point-from* $(C - D)$
3 $toofar = true$
4 ***while*** $toofar \wedge \|\mathbf{v}\| \neq \mathbf{0}$
5 $\quad \mathbf{w} = \arg\max \{\langle -\mathbf{v}, \mathbf{x} \rangle : \mathbf{x} \in C - D\}$
6 $\quad \delta = \langle \mathbf{v}, \mathbf{w} \rangle / \|\mathbf{v}\|$
7 $\quad \mu = \max(\mu, \delta)$
8 $\quad toofar = \|\mathbf{v}\| - \mu > \epsilon$
9 $\quad$ ***if*** $toofar$ ***then***
10 $\quad\quad \mathbf{v} = \arg\min \{\|\mathbf{x}\| : \mathbf{x} \in co(W \cup \{\mathbf{w}\})\}$
11 $\quad\quad W =$ *smallest* $X \subseteq W \cup \{\mathbf{w}\}$ *such that* $\mathbf{v} \in co(X)$
12 $\quad$ ***end if***
13 ***end while***

In the first line the sets of vertices C and D are initialised. The set W is initialised as empty in the second line. It will store the simplex giving the conservative (inner) approximation of $C - D$. The parameter $\mu = 0$ will be used as a lower bound for $\|\mathbf{v}\|$ in the termination condition. Vector $\mathbf{v}$ is initially chosen as arbitrary $\mathbf{x} - \mathbf{y}$, where $\mathbf{x} \in C$ and $\mathbf{y} \in D$. The loop in lines 4-13 iterates over the successive approximations of the set W, which comprises at most four vertices (corresponding to a point, a line, a triangle and a tetrahedron). Note that, $C - D$ could be computed as a convex hull of all possible point differences $\mathbf{x} - \mathbf{y}$, where $\mathbf{x} \in C$ and $\mathbf{y} \in D$. This however, would be rather inefficient. W stores few points of $C - D$ and hence its convex hull is always an inner approximation of the set difference. At each stage

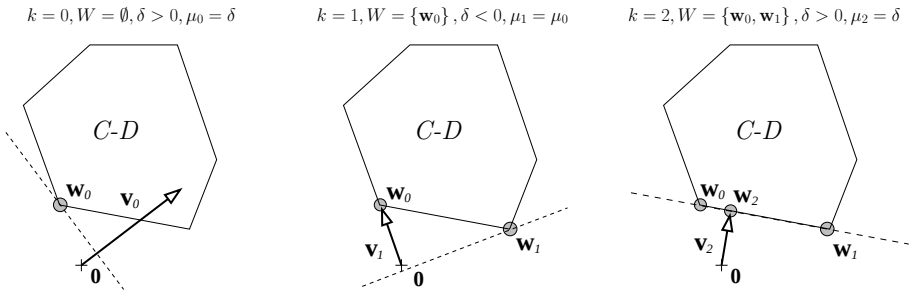


FIGURE 9.4.3. Three iterations of the GJK algorithm. The dashed line passes through a vertex $\mathbf{x}$ maximising $\langle -\mathbf{v}, \mathbf{x}\rangle$ for a current step.

of the algorithm, we can find a point $\mathbf{v} \in \mathrm{con}\,(W)$ such that $\|\mathbf{v}\|$ is minimal. If the length $\|\mathbf{v}\| = 0$ then $\mathbf{0} \in C - D$ and the two convex objects overlap. Otherwise, we can ask how much our approximation of $C - D$ can be extended in the direction of $-\mathbf{v}$, taking us closer to the origin. The point $\mathbf{w}$, extending the current set W along $-\mathbf{v}$, is computed in line 5. The next three lines deal with the termination condition. It is clear, that the sequence of produced lengths $\|\mathbf{v}_k\|$ is monotonically decreasing. After computing the new point $\mathbf{w} \in C - D$, we can check whether it actually improves upon $\mathbf{v}$ in terms of its proximity to the origin. The length of the projection of $\mathbf{w}$ along $\mathbf{v}$ can only be smaller or equal to $\|\mathbf{v}\|$, hence $\|\mathbf{v}\| - \langle \mathbf{v}, \mathbf{w}\rangle / \|\mathbf{v}\| > 0$. The parameter $\mu = \max(\mu, \langle \mathbf{v}, \mathbf{w}\rangle / \|\mathbf{v}\|)$ provides then a monotonically increasing lower bound for $\|\mathbf{v}\|$. As soon as the difference $\|\mathbf{v}\| - \mu$ becomes small enough, the algorithm is terminated (line 8). It should be noted, that application of μ in the termination condition is not really necessary for the polytope geometry. It was used in [**207**] in order to facilitate termination for smooth convex sets. It is retained here for the sake of generality. If the termination condition is not satisfied (line 9), it remains to compute new $\mathbf{v} \in W \cup \{\mathbf{w}\}$ minimising $\|\mathbf{v}\|$ (line 10). The set W is then reduced to the smallest simplex (point, line, triangle, or tetrahedron) containing $\mathbf{v}$ (line 11). Three iterations of the algorithm has been summarised in Figure 9.4.3.

GJK wouldn't probably be that successful, if not the recursive formula given in [**73**], allowing to execute the last two steps in an efficient manner. Assuming that $W = \{\mathbf{w}_0, \mathbf{w}_1, ..., \mathbf{w}_n\}$, there of course holds

$$\mathbf{v} = \sum_{i=0}^{n} \lambda_i \mathbf{w}_i \text{ and } \lambda_i \geq 0, \sum_{i=0}^{n} \lambda_i = 1 \tag{9.4.17}$$

Because $\|\mathbf{v}\|$ is minimised, $\mathbf{v}$ is orthogonal to the affine hull of the smallest subset $X \subseteq W$, such that $\mathbf{v} \in \mathrm{con}\,(X)$ (the affine hull of X is the set generated by some $\sum_{j\in I_X} \lambda_j \mathbf{w}_j$, where $\sum_{j\in I_X} \lambda_j = 1$ and $I_X \subseteq \{0, 1, ..., n\}$, hence it is the natural extension of $\mathrm{con}\,(X)$ to the whole space). Let then $X = \{\mathbf{w}_i : i \in I_X\}$ with $|I_X| \leq n$, and let equivalently $X = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_m\}$ with $m = |I_X|$. At most, there are eleven affine hulls of X (the complete space for X being the tetrahedron, four planes for the triangular faces, and six lines for the edges). One needs to select the largest subset X, for which the solution of $\langle \mathbf{v}, \mathbf{x}_0 - \mathbf{x}_j\rangle = 0$ for $j \in \{1, ..., m\}$, results in positive λs. The recursive formula for computing λs reads

$$\lambda_i = \triangle_i(X) / \triangle(X) \tag{9.4.18}$$

where

$$\triangle(X) = \sum_{i \in I_X} \triangle_i(X) \tag{9.4.19}$$

$$\triangle_i(\{\mathbf{w}_i\}) = 1 \tag{9.4.20}$$

$$\triangle_j(X \cup \{\mathbf{w}_j\}) = \sum_{i \in I_X} \triangle_i(X)\left(\langle \mathbf{w}_i, \mathbf{w}_k \rangle - \langle \mathbf{w}_i, \mathbf{w}_j \rangle\right)$$

and $j \notin I_X$, while k is a fixed index in I_X. The set X in line 11 of Algorithm 9.4.4 is selected in such a way, that $\triangle_i(X) > 0$ for all $i \in I_X$ and $\triangle_i(X \cup \{\mathbf{w}_j\}) \leq 0$ for all $j \notin I_X$. At the same time, $\mathbf{v}$ is computed as $\mathbf{v} = \sum_{i \in I_X} \lambda_i \mathbf{w}_i$.

Upon termination, it remains to obtain the pair of closest points $\mathbf{p} \in e_1$ and $\mathbf{q} \in e_2$. When computing $\mathbf{w}$s in the fifth line of Algorithm 9.4.4, one in fact stores as well the points $\mathbf{p}_i \in e_1$ and $\mathbf{q}_i \in e_2$, such that $\mathbf{w}_i = \mathbf{p}_i - \mathbf{q}_i$. Once the algorithm has terminated, the resulting pair of closest points is computed as

$$\mathbf{p} = \sum_{i \in I_X} \lambda_i \mathbf{p}_i, \quad \mathbf{q} = \sum_{i \in I_X} \lambda_i \mathbf{q}_i \tag{9.4.21}$$

9.4.2. Computing the convex hull. Barber, Dobkin and Huhdanpaa [**17**] described a fast multi-dimensional convex hull algorithm, extending the classical *Quickhull* method [**62, 34**]. The algorithm starts with a single $d-$dimensional simplex, constructed from the $d+1$ arbitrary points. The current convex polytope, describing the convex hull, is composed of faces, edges and vertices. A vertex is above a face, if it belongs to the positive half-space defined by the face plane equation. Otherwise it is below the face. Reciprocal statement can be made about a face being above or below a vertex. Each face $f = (v, e)$ is composed of a list v of unprocessed vertices placed above of the face, and a list e of edges bounding the face. Each edge $e = (o, t, f)$ comprises a pointer o to its origin vertex, a pointer t to its terminus vertex, and a pointer f to the face, being the neighbour of a face g storing e in its $g.e$ list. The incident faces of an edge e are the face g and f, such that $e \in g.e$ and $f = e.f$. Let a set of edges connected through common endpoints be called a *ridge*. The following theorem by Grünbaum [**79**] hints a basic principle behind incremental construction of convex hulls [**17**]

THEOREM 9.4.5. *(Simplified beneath-beyond) Let H be a convex hull in R^d and let $\mathbf{p}$ be a point in $R^d \setminus H$. Then f is a face of $con(H \cap \mathbf{p})$ if and only if*

1. f is a face of H and $\mathbf{p}$ is below f, or

2. f is not a face of H and its vertices are $\mathbf{p}$ and the vertices of an edge in H with one incident face below $\mathbf{p}$ and the other incident face above $\mathbf{p}$.

It is easy to see, that a convex hull can be constructed incrementally by taking the initial simplex to be H, followed by inserting one point at a time and applying the rules of the above theorem. Essentially, at each step, one detects the ridge of edges satisfying property *2* of the theorem. If the ridge is an empty set, point $\mathbf{p}$ is discarded. Otherwise, a cone of new faces is created, connecting the edges of the ridge with $\mathbf{p}$. All of the old faces located below $\mathbf{p}$ are then deleted from H. This is summarised below.

ALGORITHM 9.4.6. *Quickhull* $(\{\mathbf{p}_i\})$

```
1  H = arbitrary-tetrahedron ({p_i})
2  {p_i} = {p_i} \vertices-of (H)
3  for each p ∈ {p_i} do
4     for each face f ∈ H
```

```
5        if p is above f then List_Insert (f.v, p)
6      end for also if p is above f
7    end for
8    for each face f ∈ H ∧ f.v ≠ nil do
9      p = furthest-point-from-face (f, f.v)
10     R = ridge-of-edges-with-property-2 (H, p)
11     V = faces-below-point (p)
12     G = ∅
13     for each edge e ∈ R do
14       g = new-face (e, p)
15       G = G ∪ g
16     end for
17     for each t ∈ V do
18       for each q ∈ t.v do
19         for each g ∈ G do
20           if p is above g then List_Insert (g.v, p)
21         end for also if p is above g
22       end for
23     end for
24   H = H \ V, H = H ∪ G
25   end for
```

The novelty in the above algorithm, introduced by the authors of [**17**], is in storing in each face f the list of vertices $f.v$ located above it. The initial assignment of the input points into the face lists $f.v$ is done in lines 3-7. The loop between lines 8-24 continues, until there are faces with nonempty vertex lists $f.v \neq nil$. For each such face, an extreme vertex is chosen (line 9), maximising among all $\mathbf{p} \in f.v$ the distance from the face plane. Then the ridge R of edges having the property *2* of Theorem 9.4.5 is created (line 10). The so called *visible set* V, of faces located below the point $\mathbf{p}$ is created next. For each edge e of the ridge R, a new triangular face is created between $\mathbf{p}$ and e (lines 12-16). It should be noted, that one needs to properly maintain the adjacency information at this stage, so that faces in G and $H \setminus V$ are correctly connected. In lines 17-23 the vertices stored in the face lists $t.v$ of $t \in V$ are reassigned to the face lists $g.v$ of $g \in G$. The visible V set is deleted from H and the newly created cone of faces G is added to H in line 24. The authors of [**17**] show, that under some balance conditions, the runtime of the above algorithm is $O(n \log n)$ in three dimensions, where n is the number of input points.

9.4.3. No gaps? Traditionally, in computational contact analysis one often resorts to the notion of a *gap* between two objects. The gap can be defined as a signed scalar function, positive when two objects are apart, and semi-negative when they are in contact. The contact point and the normal direction computed from an intersection of two elements preclude an application of the gap function. This is motivated by two major factors:

(1) No direct use of gaps would be made of in the current dynamic velocity-based framework.
(2) Robust implementation of gaps is troublesome for assemblies of geometrically rough bodies.

Nevertheless, the notion of gap will be necessary in order to derive unilateral constraints in the next chapter. Also, the quasi-static contact algorithm presented therein will incorporate gaps. For these purposes, the gap function is defined as follows

$$g(t) = \begin{cases} \min_{\mathbf{x},\mathbf{y}} \|\mathbf{x}-\mathbf{y}\| : \mathbf{x} \in \bar{e}_1, \mathbf{y} \in \bar{e}_2 & \text{when } \bar{e}_1 \cap \bar{e}_2 = \emptyset \\ \min_{\mathbf{y}} \langle \mathbf{n}, \mathbf{x}-\mathbf{y} \rangle : \mathbf{y} \in \partial e_k, \mathbf{x} \in \bar{e}_1 \cap \bar{e}_2 & \text{otherwise} \end{cases} \tag{9.4.22}$$

where in the second line, the normal $\mathbf{n}$ and the point $\mathbf{x}$ are given by (9.4.3) and (9.4.4), while the k-index corresponds to the one defined in (9.4.3). The first line describes the proximity of the two elements. The second one defines a negative distance along $\mathbf{n}$, from $\mathbf{x}$ towards the surface of the intersection $\bar{e}_1 \cap \bar{e}_2$. This simple strategy is sufficient for our purposes.

9.5. Literature

Contact detection[1] is among the basic problems of computational geometry. A comprehensive introduction can be found in the survey work [**104**]. Some selected papers will be discussed here in order to put the current development in context. Section 9.5.1 enumerates papers describing general interface detection methods and related techniques. Section 9.5.2 summarises several papers dealing with computing distances and intersections between polytopes.

9.5.1. Collision detection. In an article on interface detection, Boyse [**31**] only briefly mentions the object circumscribed spheres and boxes utilised to accelerate the contact search. The focus is placed on pairwise intersection between polyhedra, with an emphasis on interface detection between a moving object and static obstacle. Detecting contact between a large number of objects is not crucial, thus no special attention is payed to the bounding volumes. Nevertheless, this is one of the earliest papers where the two-phase approach is suggested as an obvious heuristic. Culley and Kempf [**51**] propose a collision detection algorithm based on the velocity and distance bounds. Hayward [**84**] an algorithm for robotics based on the recursive octree decomposition of manipulator workspace. In the similar context of motion planning, Herman [**89**] describes another three-dimensional octree based technique. Moore and Wilhelms [**152**] build an octree structure on surface points and query it with bounding boxes of swept surface triangles. Pairs of moving points and triangles, resulting from point in box containment test, are further checked for collisions. Wu and Lee [**212**] use two-dimensional projections of three-dimensional objects in order to solve collision detection between moving robot arms. Baraff [**16**] hints bounding volumes as an enhancement of an initial search for contact candidates. He comments however in greater detail on the role of coherence in dynamic simulations. Typically geometric configuration of bodies does not change considerably between consecutive time steps. The advantage of that can be taken to accelerate both phases of interface detection. As discussed by Baraff, surface entities involved in a contact can be cached and reused. In the technical report [**88**] Heinstein *et al.* discuss a contact detection algorithm for structural dynamics, based on the node to face projection method. Garcia-Alonso *et al.* [**72**] discuss a voxel based method utilising additionally bounding boxes and an $O\left(n^2\right)$ space "collision interest matrix" used for body-pairwise events, where n is the number of bodies. A classical combination of broad and narrow phase algorithms was proposed by Cohen *et al.* [**48**]. For the pairwise collision test between convex polytopes the Lin-Canny [**139**] algorithm is employed. Closest feature of two polytopes is cached and reused as an initial guess at the next time step (this result in an expected constant runtime). Axis aligned bounding boxes are exploited to enclose convex objects. The broad phase is based on scanning along the three coordinate axes. The algorithm maintains three sorted lists of projected interval endpoints. Assuming coherence, application of insertion sort for almost ordered

[1] *contact/collision/interface detection*

lists results in an expected linear runtime. Swaps of endpoints occurring during the sort process are related to changes in overlap states. The amount of overlap status changes is of quadratic order with respect to the number of boxes n. Original implementation of the approach presented in [**48**] utilised an auxiliary $O\left(n^2\right)$ storage for status change caching. Attaway *et al.* [**15**] present a parallel collision detection framework for structural dynamics, based on the Recursive Coordinate Bisection method by Berger and Bokhari [**28**]. Gottschalk *et al.* [**75**] describe the object oriented binary tree structure, facilitating pairwise collision tests between arbitrary bodies. Hubbard [**95**] describes a technique for approximating polyhedra with spheres, and the related sphere-tree structure. Li and Chen [**138**] shown how to use hierarchical data structures in an incremental way (exploiting time coherence). Kim *et al.* [**117**] give an event-driven algorithm for collisions between moving spheres. Kitamura *et al.* [**118**] and Joukhadar [**108**] discuss collision detection between deformable polyhedra. Diekmann *et al.* [**58**] used space filling curves technique to detect contacts in planar large deformation finite element simulations. Perkins and Williams [**169**] discuss a sorting based interface detection scheme for planar objects. Feng and Owen [**68**] presented a spatial tree structure for contact detection, based on the $kd-$tree by Bentley [**26**]. Li *et al.* [**137**] presented a meshfree method based contact detection algorithm. Bruneel and De Rycke [**33**] give another spatial tree based technique for a tool-obstacle contact problem. Zomorodian and Edelsbrunner [**216**] present their fast algorithm for box intersection based on streaming the segment trees, cutoffs and scanning. Luque *et al.* [**143**] use binary space partition trees and scanning, combined with automated tree corrections improving the work balance. Teschner *et al.* [**203**] discusses the spatial hashing based approach for deformable animations. Again in the field of animation, Govindaraju *et al.* [**76**] employ graphics hardware to speed up collision detection. James and Pai [**101**] present an output-sensitive sphere tree for deformable objects. Wu *et al.* [**213**] discuss a simple vertex to face contact resolution method. Chakraborty *et al.* [**38**] present an interior point method based technique for computing distance between convex implicit surfaces. Coming and Staadt [**49**] present an event-driven sweep and prune approach for box overlap, improving upon the previous result by Cohen *et al.* [**48**]. Han *et al.* [**81**] present a method for a planar collision detection between superquadrics. Li *et al.* [**136**] present a box intersection scheme based on coherent spatial sorting, similar to the scanning used by Cohen *et al.* [**48**], although demanding only $O\left(n\right)$ space due to the employed space subdivision. Fünfzig *et al.* [**71**] presented a hierarchical spherical distance field technique for pairwise collision detection.

9.5.2. Polyhedra. Muller and Preparata [**158**] presented an algorithm for a pairwise intersection of convex polyhedra, and adopted it further [**174**] to compute intersection of half-spaces. A plane-sweep approach was employed by Hertel *et al.* [**90**] to solve the convex intersection problem and other set-theoretic operations. Meyer [**149**] discusses a technique for calculating distance between arbitrarily rotated boxes. Gilbert *et al.* [**73**] specify the GJK algorithm for calculating distance between convex polytopes. Sancheti and Keerthi [**186**] discuss some aspects of complexity of convex proximity algorithms. An algorithm for computing an intersection between an arbitrary and a convex polyhedron was given by Dobrindt *et al.* [**59**]. Quinlan [**176**] employs a sphere three structure and the GJK algorithm in order to compute the distance between nonconvex polyhedrons. Barber *et al.* [**17**] specify a fast algorithm for computing multi-dimensional convex hulls. Bhattacharya and Sen [**29**] give a randomised planar convex hull algorithm. Cameron [**35**] describes an enhanced version of the GJK algorithm with hill-climbing technique for speeding up restarts. Mirtich [**151**] has presented a fast Voronoi region clipping based

algorithm for finding distances between convex polyhedra. Levey *et al.* [**134**] compared some convex distance computing algorithms and designed improved metrics for an evaluation of their relative efficiency. Van den Bergen [**207**] presented another optimised implementation of the GJK algorithm, and applied it to distance computation between smooth convex sets. Kawachi and Suzuki [**116**] presented a voxel-based distance computation scheme for nonconvex polyhedra. Vlack and Tachi [**208**] presented a spatio-temporal implementation of the GJK scheme. Llanas *et al.* [**141**] give a convex distance algorithm based on face representation. Dyllong and Luther [**61**] implemented the interval arithmetic based version of GJK. Kavan *et al.* [**115**] discuss fast approximation of planar convex hulls.

CHAPTER 10

The frictional contact problem

It is standard to discuss at similar occasions, firstly and separately, the *contact problem* and the *friction problem*. The contact problem formulates motion of bodies touching without penetrations, but also without resistance to their relative slip. The friction problem introduces a simple slip resistance law. Both can be formulated in the language of *convex optimisation*, which is why their exposition is often pursued in the first place. As soon as the *frictional contact problem* is introduced, an interaction between the slip and the interpenetration precludes direct analogy with optimisation. This happens, because convexity in the problem structure is lost. Nevertheless, the foregoing methods and vocabulary are still of use in the analysis of this more realistic scenario. In the following sections, the three problems are formulated within the adopted framework of local dynamics.

10.1. The contact problem

The gap function between a pair of elements e_1 and e_2 was defined in the following way

$$g(t) = \begin{cases} \min_{\mathbf{x},\mathbf{y}} \|\mathbf{x}-\mathbf{y}\| : \mathbf{x} \in \bar{e}_1, \mathbf{y} \in \bar{e}_2 & \text{when } \bar{e}_1 \cap \bar{e}_2 = \emptyset \\ \min_{\mathbf{y}} \langle \mathbf{n}, \mathbf{x}-\mathbf{y} \rangle : \mathbf{y} \in \partial e_k, \mathbf{x} \in \bar{e}_1 \cap \bar{e}_2 & \text{otherwise} \end{cases} \tag{10.1.1}$$

where in the second line, $\mathbf{x}$ and $\mathbf{n}$ are given by (9.4.4) and (9.4.3). The latter formula defines also the k-index. The first line describes the proximity of the two elements. The second one defines a negative distance along $\mathbf{n}$, from $\mathbf{x}$ towards the surface of the intersection $\bar{e}_1 \cap \bar{e}_2$ (Figure 10.1.1).

By using the methods specified in the previous chapter, for all bodies we can identify pairs of potentially overlapping elements. Hence, at all times it is possible to maintain a vector of gaps

$$\mathbf{g}(t) = \begin{bmatrix} \dots \\ g_\alpha(t) \\ \dots \end{bmatrix} \tag{10.1.2}$$

between all of the identified pairs. Bodies do not penetrate each other, if only

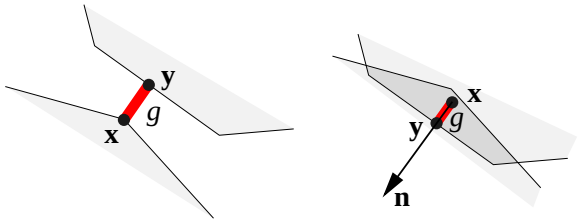


FIGURE 10.1.1. Gap according to definition (10.1.1).

$$\mathbf{g}(t) \geq 0 \text{ for all } t \tag{10.1.3}$$

where the inequality is understood in a component-wise manner. The time dependence of the gap function resolves more directly as

$$\mathbf{g}(t) = \mathbf{g}(\mathbf{q}(t)) \tag{10.1.4}$$

where $\mathbf{q}$ is the configuration of the multi-body system. In the current context we do not consider "time dependent boundaries", as these can always be realised by prescribing some time dependent joints to selected bodies. From the gaps point of view, only moving bodies are seen. This is why $\mathbf{g} = \mathbf{g}(\mathbf{q})$, rather than $\mathbf{g} = \mathbf{g}(\mathbf{q}, t)$.

10.1.1. From gaps to velocity constraints. Gradient of the gap function reads

$$\nabla g = \begin{cases} (\mathbf{x} - \mathbf{y}) / \|\mathbf{x} - \mathbf{y}\| & \text{when } \bar{e}_1 \cap \bar{e}_2 = \emptyset \\ \mathbf{n} & \text{otherwise} \end{cases} \tag{10.1.5}$$

where $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{n}$ were defined in (10.1.1). One can define a local base

$$\{\mathbf{a}_i\} = \{\mathbf{a}_{T1} \perp \nabla g, \mathbf{a}_{T2} \perp \nabla g, \mathbf{a}_N = \nabla g\} \tag{10.1.6}$$

where $\{T1, T2, N\}$ indexing replaced $\{1, 2, 3\}$, and $\mathbf{a}_{T1} \times \mathbf{a}_{T2} \neq \mathbf{0}$ was assumed. The gap velocity reads

$$\dot{g} \quad = \quad \langle \nabla g, \dot{\mathbf{x}} - \dot{\mathbf{y}} \rangle \tag{10.1.7}$$

and the local velocity with respect to the base (10.1.6) follows as

$$\mathbf{U} = \left\{\mathbf{a}^i\right\}^T (\dot{\mathbf{x}} - \dot{\mathbf{y}}) \tag{10.1.8}$$

More generally, for a multi-body system, the vector of local velocities $\mathbf{U}$ for all contact related local frames can be expressed in a familiar form

$$\mathbf{U} = \mathbf{H}\mathbf{u} \tag{10.1.9}$$

where $\mathbf{H}$ is evaluated according to (7.1.9), and $\mathbf{u}$ is the generalised velocity of the system. In the evaluation of $\mathbf{H}$, one employs the referential images of $\mathbf{x}$ and $\mathbf{y}$ together with $\left\{\mathbf{a}^i\right\}^T = \{\mathbf{a}_i\}^{-1}$. For the moment, relation (10.1.9) is understood in the time continuous, rather than discrete sense. From (10.1.9) one can extract the gap velocity function in its vector form

$$\dot{\mathbf{g}} = \mathbf{H}_{N*}\mathbf{u} \tag{10.1.10}$$

where $\mathbf{H}_{N*}$ denotes selection of the normal component rows of $\mathbf{H}$. Hence, $\dot{\mathbf{g}}$ is the vector of the local normal velocities

$$\dot{\mathbf{g}} = \begin{bmatrix} \cdots \\ U_{\alpha N} \\ \cdots \end{bmatrix} \tag{10.1.11}$$

Let us now define a set

$$\mathbf{\Gamma}(\mathbf{q}, t) = \begin{cases} \mathbf{u} \in T_{\mathbf{q}}\mathcal{Q} : U_{\alpha N} \geq 0 & \text{for } g_\alpha(t) \leq 0 \\ T_{\mathbf{q}}\mathcal{Q} & \text{otherwise} \end{cases} \tag{10.1.12}$$

In his *integration lemma*, Moreau [**153**] shows that

LEMMA 10.1.1. *If the inclusion*

$$\mathbf{u} \in \mathbf{\Gamma}(\mathbf{q}, t) \tag{10.1.13}$$

holds for almost every[1] $t \in [0, T)$ *and the inequality (10.1.3) is verified for* $t = 0$, *then the same inequality is verified for* $t \in [0, T)$.

The proof relies on an assumption, that the configuration $\mathbf{q}$ is obtained from the velocity $\mathbf{u}$ as a result of integration

$$\mathbf{q}(t) = \mathbf{q}(0) + \int_0^t \mathbf{u}(s)\, ds \tag{10.1.14}$$

which for rigid rotations needs to be understood in a suitably generalised manner. The rest of the proof can be summarised as follows. One assumes $g(\tau) < 0$, $\tau < T$ and then looks for a contradiction. As $g(0) \geq 0$ and $g(t)$ is continuous, it has to pass by 0, say at time σ, on its way towards $g(\tau) < 0$. As $\dot{g} \geq 0$ holds almost everywhere in $[0, T)$, there follows $g(\tau) = \int_\sigma^\tau \dot{g} dt \geq 0$ which gives the desired contradiction. We have omitted technical assumptions related to the regularity of the involved functions.

Taking the local velocity $\mathbf{U}$ point of view on the above lemma, the non-penetration constraint can be summarised as follows

$$\begin{aligned} \mathbf{U} &\in TE^3 \quad \text{if} \quad g > 0 \\ [\mathbf{U}_T, U_N] &\in TE^2 \times R_+ \quad \text{if} \quad g \leq 0 \end{aligned} \tag{10.1.15}$$

where $R_+ = [0, \infty)$ is the semi-positive real half-line and the second line holds almost everywhere in $[0, T)$. The local velocity is allowed to take arbitrary values, when the gap between an element pair is positive. Otherwise, while the tangent component $\mathbf{U}_T$ remains arbitrary, the normal component U_N needs to be semi-positive.

10.1.2. Moreau's sweeping. A specific instance of a solution to the problem posed by the *differential inclusion* (10.1.13) is the Moreau's *sweeping process*. One can define a set of all interpenetration free body positions as

$$\mathbf{\Phi}(t) = \{\mathbf{q} \in \mathcal{Q}(t) : \mathbf{g}(\mathbf{q}) \geq 0\} \tag{10.1.16}$$

where $\mathcal{Q}(t)$ is used to emphasise a possible presence of time dependent joints. Imagine for example someone slowly sweeping a pool table top with a hand brush. A pack of cigarettes left on the table is being pushed around slowly enough, so that it freezes right after losing contact with the brush. For each position of the brush, the pack of cigarettes could be placed anywhere within the table borders and away from the brush. The set of those placements is the interior of $\mathbf{\Phi}$, the position of the pack is $\mathbf{q}$ and its velocity is $\mathbf{u}$. The sort of behaviour just described, can be achieved by selecting for each time t an element $\mathbf{u} \in \mathbf{\Gamma}(\mathbf{q}, t)$, such that the norm $\|\mathbf{u}\|$ is minimised. To describe it more consistently, it is temporarily convenient to assume $\mathbf{g} = \mathbf{g}(\mathbf{q}, t)$ (i.e. account for the motion of the brush as a moving boundary), rather than $\mathbf{g} = \mathbf{g}(\mathbf{q})$ (i.e. consider the two-body system, where the non-penetration and the imposed motion constraints are handled simultaneously). If all $\mathbf{g} > 0$ then $\mathbf{u} = \mathbf{0}$ (the brush is away from the pack and hence the pack is left at rest). If some of the gaps $g_\alpha \leq 0$, then the point $\mathbf{q} \in \partial\mathbf{\Phi}$ touches the boundary

[1] by which one means $t \in [0, T) \setminus \mathcal{Z}$ for sets $\mathcal{Z}$ such that $\int_{[0,T)\setminus\mathcal{Z}} \mathbf{u} = \int_{[0,T)} \mathbf{u}$, where $\mathcal{Z}$ can be understood as an arbitrary sequence $\{t_n\} \subset [0, T)$

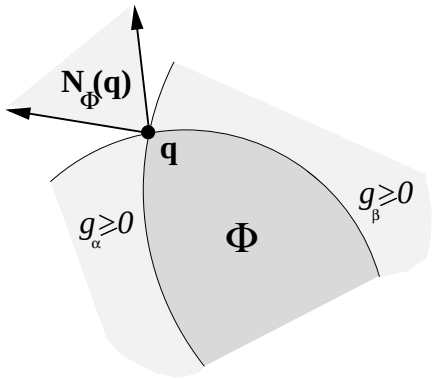


FIGURE 10.1.2. Normal cone $N_{\mathbf{\Phi}}$ at $\mathbf{q} \in \partial\mathbf{\Phi}$. Note that $\mathbf{q} + N_{\mathbf{\Phi}}$ was plotted rather than $N_{\mathbf{\Phi}}$ (which should be rooted at $\mathbf{0}$).

of $\mathbf{\Phi}$ (e.g. the brush is pushing the pack). According to (10.1.15), the condition is now $U_{\alpha N} = \nabla_{\mathbf{q}} g_\alpha \cdot \mathbf{u} + \partial g_\alpha / \partial t \geq 0$, where $\nabla_{\mathbf{q}} g_\alpha = \mathbf{H}^T_{\alpha N *}$. For $g_\alpha \leq 0$ there holds then

$$\mathbf{H}_{\alpha N *}\mathbf{u} + \partial g_\alpha / \partial t \geq 0 \tag{10.1.17}$$

We have $\partial g_\alpha / \partial t = 0$ if this is the gap between the pack and a table border, and possibly $\partial g_\alpha / \partial t \neq 0$ if this is the gap between the pack and the brush. According to the assumed selection rule, one now wishes to find the smallest velocity satisfying the above system of inequalities

$$\mathbf{u} = \arg\min \|\mathbf{u}\| : \mathbf{H}_{\alpha N *}\mathbf{u} + \partial g_\alpha / \partial t \geq 0 \tag{10.1.18}$$

If all of the $\partial g_\alpha / \partial t \geq 0$ then the brush, although touching the pack, is about to move away. Hence, the minimal $\mathbf{u} = \mathbf{0}$, as the origin belongs to the set bounded by (10.1.17). If there are some $\partial g_\alpha / \partial t < 0$, then the brush is actively pushing the pack. From the geometrical point of view, the system of inequalities (10.1.17) describes a convex polyhedral set P, not containing the origin. A point $\mathbf{u} \in P$, closest to the origin, can then be expressed as a linear combination of normals to the hyperplanes not containing the origin. This is precisely

$$\mathbf{u} = \sum_\alpha \lambda_\alpha \mathbf{H}^T_{\alpha N *} \tag{10.1.19}$$

where

$$\lambda_\alpha = -\min(0, \partial g_\alpha / \partial t) \,/\, \left\|\mathbf{H}^T_{\alpha N *}\right\|^2 \geq 0 \tag{10.1.20}$$

From the above it follows, that for $g_\alpha = 0$ the condition $\nabla_{\mathbf{q}} g_\alpha \cdot \mathbf{u} + \partial g_\alpha / \partial t = 0$ holds as equality, so that the point $\mathbf{q}$ "catches up" with the boundary of $\mathbf{\Phi}$. For $\mathbf{q} \in \partial\mathbf{\Phi}$ formula (10.1.19) describes $\mathbf{u}$ as a semi-positive linear combination of vectors $\nabla_{\mathbf{q}} g_\alpha$. Any such combination forms a *cone*. It is convenient to generalise this notion for all $\mathbf{q}$. The *normal cone* of the set $\mathbf{\Phi}$ is defined as follows (Figure 10.1.2)

$$N_{\mathbf{\Phi}} = \begin{cases} -\sum \lambda_\alpha \nabla_{\mathbf{q}} g_\alpha, \lambda_\alpha \geq 0, g_\alpha = 0 & \text{when } \mathbf{q} \in \partial\mathbf{\Phi} \\ \{\mathbf{0}\} & \text{when } \mathbf{q} \in \text{interior}\,(\mathbf{\Phi}) \\ \emptyset & \text{otherwise} \end{cases} \tag{10.1.21}$$

In [**153**] Moreau shows, that the velocity selection rule in the sweeping process can be equivalently described as

$$-\mathbf{u} \in N_{\mathbf{\Phi}(t)}(\mathbf{q}(t)) \text{ almost everywhere in } [0,T) \tag{10.1.22}$$

This results from the fact, that $\mathbf{u} \notin \emptyset$ for *almost every* t, which implies that $\nabla_{\mathbf{q}} g_\alpha \cdot \mathbf{u} + \partial g_\alpha/\partial t < 0$ is not allowed to happen (i.e. λ_α are appropriately chosen in (10.1.21), otherwise one could always pick $\mathbf{u} = \mathbf{0}$ and soon end up with $\mathbf{q} \notin \mathbf{\Phi}$ and hence $\mathbf{u} \in \emptyset$). We have then $\mathbf{q} \in \mathbf{\Phi}$, which also follows from Lemma 10.1.1 as $-N_{\mathbf{\Phi}(t)}(\mathbf{q}) \subseteq \mathbf{\Gamma}(\mathbf{q}, t)$. For $\mathbf{q} \in \text{interior}(\mathbf{\Phi})$ there follows $\mathbf{u} = \mathbf{0}$ and the point stays at rest. In the remaining case, $\mathbf{q} \in \partial\mathbf{\Phi}$ can hold for a sequence $\{t_n\} \in [0,T)$, but whenever $\mathbf{q} \in \partial\mathbf{\Phi}$ over $[a,b] \subseteq [0,T)$, there must almost everywhere in $[a,b]$ hold $\nabla_{\mathbf{q}} g_\alpha \cdot \mathbf{u} + \partial g_\alpha/\partial t = 0$, which leads to (10.1.19). There cannot be $\mathbf{q} \in \partial\mathbf{\Phi}$ over $[a,b]$ together with $\nabla_{\mathbf{q}} g_\alpha \cdot \mathbf{u} + \partial g_\alpha/\partial t > 0$, because $g_\alpha(\mathbf{q}, b) = g_\alpha(\mathbf{q}, a) + \int_a^b \dot{g}_\alpha(\mathbf{q}, s)\, ds > 0$ and $\mathbf{q}$ departs from the boundary.

Let us recapitulate. Assume that $\mathbf{q} \in \mathbf{\Phi}$ at $t = 0$. We can sweep $\mathbf{q}$ with set $\mathbf{\Phi}$ as soon as $\mathbf{u} = \mathbf{0}$ if $\mathbf{g} > 0$, or (10.1.19) is used when $g_\alpha \leq 0$. Formula (10.1.14) is utilised to advance $\mathbf{q}$. The sweeping process can be understood as quasi-static pushing of $\mathbf{q}$ by the boundary of $\mathbf{\Phi}$. In [**155**] Moreau detailed this idea in the infinite-dimensional context. An introduction to Moreau's sweeping with applications to unilateral mechanics can be found in Kunze and Monteiro Marques [**129**].

10.1.3. Velocity jumps. In order to preserve non-penetration, it is necessary to admit jumps in the graph of the gap velocity $\dot{g}(t)$. This allows bodies to rebound, while the graph of the gap can have sharp minima touching the horizontal axis. For a given local frame, at each time t one can define the left and the right velocity

$$U_N^-(t) = \lim_{s \downarrow 0} U_N(t-s), \quad U_N^+(t) = \lim_{s \downarrow 0} U_N(t+s) \tag{10.1.23}$$

An impact corresponds to $g(t) = 0$ and $U_N^- < 0$. To secure non-penetration, for the right velocity there needs to hold $U_N^+ \geq 0$. The change of sign in the relative velocity cannot output more energy than it consumes, and hence

$$U_N^+ = -\eta U_N^- \tag{10.1.24}$$

where $\eta \in [0,1]$ is called Newton's *coefficient of restitution*. The extrema of the above relation

$$U_N^+ = 0 \text{ and } U_N = -U_N^- \tag{10.1.25}$$

correspond respectively to the *ideally plastic* and *ideally elastic* impacts. In the former case, after an impact the material points move within the tangent plane spanned by $\mathbf{a}_{T1}$ and $\mathbf{a}_{T2}$. In the latter one, they rebound without loss of the kinetic energy $E_k = \frac{1}{2}\|\mathbf{U}\|^2$.

It is convenient to rephrase condition (10.1.15) as

$$\begin{aligned} \mathbf{U}^+ &\in TE^3 \quad \text{if} \quad g > 0 \\ \left[\mathbf{U}_T^+, U_N^+\right] &\in TE^2 \times R_+ \quad \text{if} \quad g \leq 0 \end{aligned} \tag{10.1.26}$$

which must hold *everywhere* in $[0,T)$. While the above assures non-penetration, no particular value is assigned to U_N^+. For computational purposes it is convenient to define the following auxiliary velocity

$$\bar{U}_N = U_N^+ + \eta \min\left(0, U_N^-\right) \tag{10.1.27}$$

The unilateral contact constraint can be spelt out again as

$$\begin{aligned} \mathbf{U}^+ &\in TE^3 \quad \text{if} \quad g > 0 \\ \left[\mathbf{U}_T^+, \bar{U}_N\right] &\in TE^2 \times R_+ \quad \text{if} \quad g \leq 0 \end{aligned} \tag{10.1.28}$$

where for $\bar{U}_N = 0$ the Newton's restitution law (10.1.24) is recovered if $U_N^- < 0$.

10.1.4. Back to the discrete case. In the time discretised context, for each t we shall identify

$$\mathbf{U}^+ = \mathbf{U}^{t+h} \text{ and } \mathbf{U}^- = \mathbf{U}^t \tag{10.1.29}$$

Because

$$\mathbf{U}^{t+h} = \mathbf{W}\mathbf{R} + \mathbf{B} \tag{10.1.30}$$

conditions put on $\mathbf{U}^{t+h}$ can be realised by an appropriate choice of $\mathbf{R}$. We assume lack of resistance with respect to the tangential motion

$$\mathbf{R}_T = \mathbf{0} \tag{10.1.31}$$

In the absence of the free velocity $\mathbf{B} = \mathbf{0}$, one can see that the above condition and semi-positive definiteness of $\mathbf{W}$ imply that a positive normal reaction $R_N > 0$ causes a semi-positive normal velocity U_N^{t+h}. In other words, a positive normal reaction implies separation, while the negative one can pull a pair of material points together. We do not consider adhesion and hence

$$R_N \geq 0 \tag{10.1.32}$$

Consequently, a semi-positive reaction is needed in order to assure $\bar{U}_N \geq 0$. This allows to state conditions for the contact reaction $\mathbf{R}$, analogous to (10.1.28)

$$\begin{aligned} \mathbf{R} &= \mathbf{0} \quad \text{if} \quad g > 0 \\ \left[\mathbf{R}_T, R_N\right] &\in \mathbf{0} \times R_+ \quad \text{if} \quad g \leq 0 \end{aligned} \tag{10.1.33}$$

If an impact happens between t and $t+h$, or an established contact persists over $[t, t+h]$, the normal reaction is used so to assure that $\bar{U}_N = 0$. Note that when $U_N^- < 0$ this results in Newton's restitution, while when $U_N^- = 0$ then $U_N^+ = 0$ follows and the contact persists. Defining $\bar{U}_N = U_N^+ + \eta \min\left(0, U_N^-\right)$ is meant to be adjusted to our way of detection and resolution of contact. An element overlap can persist over a sequence of adjacent time moments $t, t+h, ..., t+nh$, although an impact corresponds only to the reversal of the velocity sign. It would be inappropriate to use $\bar{U}_N = U_N^+ + \eta U_N^-$ when $U_N^- > 0$, as then the condition $\bar{U}_N = 0$ could imply $U_N^+ < 0$. In the next step that would lead to $U_N^- < 0$ and the velocity sign would continue reversing as long as the overlap between the elements would hold. Using (10.1.27) naturally prevents this scenario.

Conditions (10.1.28), (10.1.33) and the above discussion lead to the following *complementarity* between the auxiliary normal velocity and the normal reaction

$$\bar{U}_N \geq 0, \;\; R_N \geq 0, \;\; \bar{U}_N R_N = 0 \tag{10.1.34}$$

The above is sometimes referred to as the *velocity Signorini condition* (cf. Jean [**102**]). Conditions (10.1.34) can be combined with the normal part of relation

(10.1.30) and together form a *linear complementary problem* (in short an *LCP*) as follows

$$\left\{ \begin{array}{l} \mathbf{U}_N^{t+h} = \mathbf{W}_{NN}\mathbf{R}_N + \mathbf{B}_N \\ \bar{U}_{\alpha N} \geq 0, \;\; R_{\alpha N} \geq 0, \;\; \bar{U}_{\alpha N} R_{\alpha N} = 0 \end{array} \right. \tag{10.1.35}$$

where $\mathbf{U}_N^{t+h}$ and other vectors with index N comprise only normal components, and $\mathbf{W}_{NN}$ is obtained from $\mathbf{W}$ by removing all tangential terms. Any pair $\mathbf{U}_N^{t+h}$, $\mathbf{R}_N$ verifying the above system together with $\mathbf{R}_T = \mathbf{0}$, solve the discrete dynamic contact problem. Whether the continuous contact problem is solved when $h \to 0$ is a separate question. Signorini conditions (10.1.34) imply that the right velocity $U_N^{t+h} \geq 0$ for all $t + h$. As $U_N^+ = \lim_{h \to 0} U_N^{t+h}$, in the limit condition (10.1.26) is verified. If the left velocity U_N^- is negative only on a sequence of points $\{t_n\} \subset [0, T)$, then condition (10.1.13) in the integration Lemma 10.1.1 does hold almost everywhere. If $U_N^+ > 0$, then the contact is released and due to the continuity of the gap function, some time is needed before $U_N^- < 0$ can happen again. This separates two impact events. If $U_N^+ = 0$, then the gap function can remain zero or grows as $U_N^+ \geq 0$ for $g = 0$. Again, this separates two consecutive $U_N^- < 0$ events. The above discussion is rather rough, and does not mention regularity assumptions. Intuitively, "it should work" provided that (10.1.35) can be always solved and the free velocity $\mathbf{B}_N(t)$ is not everywhere discontinuous. Fore a rigorous treatment we refer the reader to the already mentioned references [**155, 129**].

10.1.5. From inequalities to equalities. We would like to use the uniform notation $\mathbf{C}(\mathbf{U}, \mathbf{R}) = \mathbf{0}$ for all constraints. This is not quite the case for the complementary conditions (10.1.34), but it is not difficult to cast them into the form of equality. The following variational inequality is equivalent to the complementarity conditions (10.1.34)

$$R_N \in R_+ \;\; \forall S \in R_+ \;\; \bar{U}_N (S - R_N) \geq 0 \tag{10.1.36}$$

where R_+ stands for the semi-positive real half-space. This can be checked by inspection. Take any $R_N > 0$, then $\bar{U}_N (S - R_N) \geq 0$ implies that $\bar{U}_N = 0$. Take $\bar{U}_N > 0$, then there must hold $R_N = 0$. Finally, for $R_N = 0$ we have $\bar{U}_N \geq 0$. The inequality $\bar{U}_N (S - R_N) \geq 0$ can be rewritten as

$$\left(R_N - \left(R_N - \rho \bar{U}_N\right)\right)(S - R_N) \geq 0 \tag{10.1.37}$$

for any $\rho > 0$. As $R_N, S \in R_+$ the above can be viewed as a definition of projection

$$R_N = \mathrm{proj}_{R_+}\left(R_N - \rho \bar{U}_N\right) \tag{10.1.38}$$

of the vector $R_N - \rho \bar{U}_N$ onto the convex set R_+ (Figure 10.1.3). The act of subtraction $R_N - \rho \bar{U}_N$ requires a comment. Note, that components of the reaction $\mathbf{R}$ are expressed with respect to the dual base $\mathbf{a}^i$, while the components of the velocity $\mathbf{U}$ are expressed with respect to the base $\mathbf{a}_i$. Thus operation $\mathbf{R} \pm \mathbf{U}$ does not make sense, unless one of the objects is brought to the base of the other one. For example, the metric tensor $\mathcal{A} = \{\mathbf{a}_i\}^T \{\mathbf{a}_i\}$ can be employed in order to compute $U_i = \mathcal{A}_{ij} U^j$. This follows from $\{\mathbf{a}^i\} \mathbf{U}_{\{i\}} = \{\mathbf{a}_i\} \mathbf{U}^{\{i\}}$ and $\{\mathbf{a}_i\}^T \{\mathbf{a}^i\} = \mathbf{I}$. The correction reads $\mathbf{R} \pm \mathcal{A}\mathbf{U}$. Nevertheless, due to the definition of the base (10.1.6), the metric tensor looks like

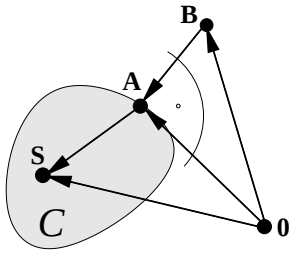


FIGURE 10.1.3. Interpretation of formula (10.1.37) as projection. Let C be a convex set. For all $\mathbf{S} \in C$ we have $\langle \mathbf{A} - \mathbf{B}, \mathbf{S} - \mathbf{A} \rangle \geq 0$. This implies that $\mathbf{A} = \mathrm{proj}_C(\mathbf{B})$.

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}_{11} & \mathcal{A}_{12} & 0 \\ \mathcal{A}_{21} & \mathcal{A}_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10.1.39}$$

and hence the normal component $U_N = U^N$. The projection formula (10.1.38) is then consistent. We can now state the contact law as

$$\mathbf{C}(\mathbf{U}, \mathbf{R}) = \mathbf{0} \tag{10.1.40}$$

where

$$\mathbf{C}(\mathbf{U}, \mathbf{R}) = \begin{cases} \mathbf{R} & \text{if} \quad g > 0 \\ \begin{bmatrix} \mathbf{R}_T \\ R_N - \mathrm{proj}_{R_+}\left(R_N - \rho \bar{U}_N\right) \end{bmatrix} & \text{if} \quad g \leq 0 \end{cases} \tag{10.1.41}$$

Additional comments about the above derivation can be found for example in Wosle and Pfeiffer [**210**].

10.1.6. Non-smoothness. Projection in (10.1.41) is a *nonsmooth* function. To picture that, let us consider a one dimensional simplification of the contact problem

$$\begin{cases} u = wr + b \\ r = \mathrm{proj}_{R_+}(r - u) \end{cases} \tag{10.1.42}$$

where $\rho = 1$ and $\eta = 0$ was assumed. The above system can be rewritten as

$$c(r) = r - \max(0, r(1 - w) - b) = 0 \tag{10.1.43}$$

which is a nonlinear equation the root of which is sought. One can see that for $r < b/(1 - w)$ the root is $r = 0$ while in the remaining case $r = -b/w$. In the former case $u = b \geq 0$, as $b < 0$ suppresses the root $r = 0$. In the latter case $u = 0$ and $r \geq 0$, which recovers the Signorini condition. The multi-dimensional version of (10.1.43) reads

$$c_\alpha(\mathbf{r}) = r_\alpha - \max\left(0, r_\alpha(1 - w_{\alpha\alpha}) - b_\alpha - \sum_{\beta \neq \alpha} w_{\alpha\beta} r_\beta\right) = 0 \tag{10.1.44}$$

A series of plots of $c(r)$ for various values of b corresponds to a series of sections of $c_\alpha(\mathbf{r})$ for some fixed $r_{\beta \neq \alpha}$. This can be observed for the two-dimensional case in Figure 10.1.4. What is also visible is the non-smoothness of the constraint graphs.

Each surface plot splits into the part where $c_\alpha(\mathbf{r}) = r_\alpha$, and into another one where $c_\alpha(\mathbf{r})$ is an arbitrarily inclined half-plane. Both parts are connected in a continuous manner along the line $r_\alpha - \rho u_\alpha = 0$. This is where the non-smoothness occurs. In situations when derivatives of $\mathbf{C}(\mathbf{U}, \mathbf{R})$ need to be computed, one has to sort out differentiation along $r_\alpha - \rho u_\alpha = 0$. This will be further commented on in the next chapter.

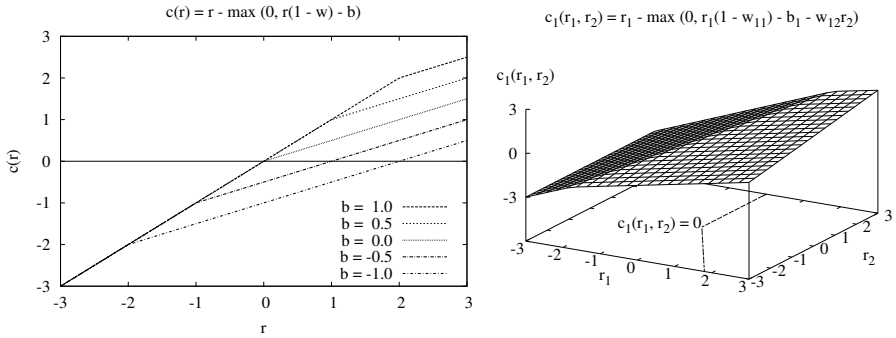


FIGURE 10.1.4. Plots of the Signorini constraints (10.1.43) and (10.1.44) for $w = w_{11} = 0.5$, $w_{12} = 0.3$, $b_1 = 0$.

10.1.7. Existence of solutions. It is relevant to ask whether the solution for (10.1.44) actually exists. As suggested by the contour line $c_1(r_1, r_2) = 0$ in Figure 10.1.4, each constraint contributes a curve composed of two straight half-lines. If all such curves have a nonempty intersection, the solution exists. If the intersection happens to be a point, the solution is unique. The half-line components are either $r_\alpha = 0$ or $b_\alpha + w_{\alpha\beta} r_\beta = 0$, where the *summation over β is assumed.*

Let $\mathbf{W}$ be positive-definite. The matrix $\mathbf{w}$ is the principal sub-matrix of $\mathbf{W}$, obtained by removing all rows and columns involved in the tangential response $\mathbf{w} = \mathbf{W}_{NN}$. Constructed this way, $\mathbf{w}$ remains symmetric and positive definite [**200**, p. 339]. Such $\mathbf{W}$ and $\mathbf{w}$ are sometimes called P-matrices or said to have P-property. It is readily seen that this property assures the existence of a solution. For the two extreme cases one has either all $r_\alpha = 0$ (no contact) or the system of equations $b_\alpha + w_{\alpha\beta} r_\beta = 0$ is uniquely solved (all points in contact). In the remaining cases some $r_{\alpha \in I} = 0$ and some emerge as a solution of $b_\alpha + w_{\alpha\beta} r_\beta = 0$, where $\alpha, \beta \notin I$. The latter system can always be solved due to the P-property of $\mathbf{w}$. In order to show uniqueness, let us select the index set I of minimal size $|I|$, such that $b_\alpha + w_{\alpha\beta} r_\beta = 0$ results in $r_\beta > 0$ for all $\alpha, \beta \notin I$. Such I is unique and can be empty. Then by definition, for each $\alpha \in I$ and $\beta \notin I$

$$r_\alpha = -\frac{b_\alpha + w_{\alpha\beta} r_\beta}{w_{\alpha\alpha}} \leq 0 \tag{10.1.45}$$

and since $w_{\alpha\alpha} > 0$ (positive-definiteness), there follows $u_\alpha = b_\alpha + w_{\alpha\beta} r_\beta \geq 0$. If a larger I with property $b_\alpha + w_{\alpha\beta} r_\beta = 0 \Rightarrow r_\beta > 0$ for $\alpha, \beta \notin I$ would be considered, then by similar argument $u_\alpha \leq 0$ and complementarity wouldn't hold. Hence the uniqueness. This is in fact a classical result related to convex optimisation, or solution of linear complementary problems (cf. Hintermüller *et al.* [**93**]).

When $\mathbf{W}$ is only semi-positive definite, existence of a solution cannot be assured for arbitrary b_α. Luckily, as $b_\alpha = B_{\alpha N}$ and $\mathbf{B} = \mathbf{H}\mathbf{M}^{-1}\mathbf{b}$ (cf. Chapter 7), a linear dependency in $\mathbf{H}$ affects $\mathbf{b}$ in the same way as it does affect $\mathbf{w}$. In other words the

range of $\mathbf{H}$ is the same as the range of $\mathbf{W}$ and hence $b_\alpha + w_{\alpha\beta} r_\beta = 0$ is likely to be solvable, because there cannot be any b_α from outside of the range of $\mathbf{w}$. This will be also visible from the structure of the minimisation problem in Section 10.1.9. Of course, a solution may fail to exist for a specific instance of time dependent constraints. One can imagine a rigid block squeezed from two opposite directions, so that the two constraints cannot be simultaneously fulfilled. This is a situation when contradictory pair exists, $b + w_{1\beta} r_\beta = 0$ and $w_{2\beta} r_\beta - b = 0$, where $w_{1\beta} = \lambda w_{2\beta}$ and $\lambda > 0$. However, this case can be perceived as a "modelling error".

10.1.8. Contact problem as root finding. Let us denote

$$d_N \left(U_N, R_N \right) = R_N - \rho \bar{U}_N \tag{10.1.46}$$

and call d_N a *normal predictor*. We can rewrite the non-penetration constraint (10.1.38) as

$$C_N \left(U_N, R_N \right) = R_N - \max \left(0, d_N \right) \tag{10.1.47}$$

Gathering all the constraints into a vector operator and using local dynamics, we can then state the following root finding problem

$$\mathbf{C}_N \left(\mathbf{U}_N, \mathbf{R}_N \right) = \mathbf{0} |_{\mathbf{U}_N = \mathbf{W}_{NN} \mathbf{R}_N + \mathbf{B}_N} \tag{10.1.48}$$

or in short

$$\mathbf{C}_N \left(\mathbf{R}_N \right) = \mathbf{0} \tag{10.1.49}$$

An important feature of the operator $\mathbf{C}_N$ is its *monotonicity*. This means that for all pairs $\mathbf{A}$, $\mathbf{B}$ there holds

$$\left\langle \mathbf{C}_N \left(\mathbf{A} \right) - \mathbf{C}_N \left(\mathbf{B} \right), \mathbf{A} - \mathbf{B} \right\rangle \geq 0 \tag{10.1.50}$$

The above can be shown to hold as follows. Let Newton's coefficient of restitution be $\eta = 0$. This does not obscure generality, while the predictor can now be expressed as

$$\mathbf{d}_N \left(\mathbf{R}_N \right) = \mathbf{R}_N - \rho \mathbf{U}_N \left(\mathbf{R}_N \right) \tag{10.1.51}$$

Operator $\mathbf{C}_N$ can be rewritten as

$$\mathbf{C}_N \left(\mathbf{R}_N \right) = \mathbf{R}_N - \mathrm{proj}_X \left(\mathbf{d}_N \left(\mathbf{R}_N \right) \right) \tag{10.1.52}$$

where X is the positive orthant $R_+ \times R_+ \times ... \times R_+$. It will be helpful to notice, that

$$\left\| \mathbf{W}_{NN} \mathbf{A} \right\| \leq \lambda_{max} \left\| \mathbf{A} \right\| \tag{10.1.53}$$

$$\left\langle \mathbf{W}_{NN}^{-1} \mathbf{A}, \mathbf{A} \right\rangle \geq \frac{1}{\lambda_{max}} \left\| \mathbf{A} \right\|^2 \tag{10.1.54}$$

where λ_{max} is the maximal eigenvalue of $\mathbf{W}_{NN}$. Estimate (10.1.53) holds, because the l_2 norm of a symmetric matrix is equal to its spectral radius. Estimate (10.1.54) can also be derived from the spectral picture of the scalar product, and the fact that $\lambda_{min} \left(\mathbf{W}_{NN}^{-1} \right) = 1 / \lambda_{max} \left(\mathbf{W}_{NN} \right)$. In the following derivation it will be convenient to use $\delta \mathbf{R}_N = \mathbf{R}_{1N} - \mathbf{R}_{2N}$ and $\delta \mathbf{U}_N = \mathbf{U}_{1N} - \mathbf{U}_{2N}$. It will be also useful to note

$$\delta \mathbf{U}_N = \mathbf{W}_{NN} \delta \mathbf{R}_N \tag{10.1.55}$$

and to introduce the ratio $\beta = \|\delta \mathbf{U}_N\| / \|\delta \mathbf{R}_N\|$. From the semi-positive definiteness of $\mathbf{W}_{NN}$, the Schwarz inequality $|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| \|\mathbf{y}\|$ and (10.1.55), one can obtain $\|\delta \mathbf{U}_N\| \|\delta \mathbf{R}_N\| \geq \langle \delta \mathbf{U}_N, \delta \mathbf{R}_N \rangle \geq \lambda_{min} \|\delta \mathbf{R}_N\|^2$, and by using (10.1.53) conclude, $\lambda_{min} \leq \beta \leq \lambda_{max}$. The two extreme eigenvalues are both of $\mathbf{W}_{NN}$. The projection onto a convex set is a contraction (cf. [**183**, p. 545]) and hence

$$\begin{aligned}
\|\mathrm{proj}_X (\mathbf{d}_N (\mathbf{R}_{1N})) - \mathrm{proj}_X (\mathbf{d}_N (\mathbf{R}_{2N}))\|^2 &\leq \\
\|\mathbf{d}_N (\mathbf{R}_{1N}) - \mathbf{d}_N (\mathbf{R}_{2N})\|^2 &\leq \\
\|\delta \mathbf{R}_N - \rho \delta \mathbf{U}_N\|^2 &\leq \\
\|\delta \mathbf{R}_N\|^2 - 2\rho \left\langle \mathbf{W}_{NN}^{-1} \delta \mathbf{U}_N, \delta \mathbf{U}_N \right\rangle + \rho^2 \|\delta \mathbf{U}_N\|^2 &\leq \\
\left(1 - \frac{2}{\lambda_{max}} \rho \beta^2 + \rho^2 \beta^2\right) \|\delta \mathbf{R}_N\|^2 &\leq \\
\|\delta \mathbf{R}_N\|^2 \quad \text{for} \quad \rho < \frac{2}{\lambda_{max}} &
\end{aligned} \tag{10.1.56}$$

and

$$\min_{\rho} \left(1 - \frac{2}{\lambda_{max}} \rho \beta^2 + \rho^2 \beta^2\right) = 1 - \frac{\beta^2}{\lambda_{max}^2} \leq 1 - \frac{\lambda_{min}^2}{\lambda_{max}^2} \tag{10.1.57}$$

A more general derivation of (10.1.56) can be found in the paper by Laborde and Renard [**168**]. Finally we can write

$$\begin{aligned}
\langle \mathbf{C}_N (\mathbf{A}) - \mathbf{C}_N (\mathbf{B}), \mathbf{A} - \mathbf{B} \rangle &= \\
\langle \mathbf{A} - \mathrm{proj}_X (\mathbf{d}_N (\mathbf{A})) - \mathbf{B} + \mathrm{proj}_X (\mathbf{d}_N (\mathbf{B})), \mathbf{A} - \mathbf{B} \rangle &= \\
\|\mathbf{A} - \mathbf{B}\|^2 - \langle \mathrm{proj}_X (\mathbf{d}_N (\mathbf{A})) - \mathrm{proj}_X (\mathbf{d}_N (\mathbf{B})), \mathbf{A} - \mathbf{B} \rangle &\geq 0
\end{aligned} \tag{10.1.58}$$

This proves monotonicity of $\mathbf{C}_N$ for $\rho < \frac{2}{\lambda_{max}}$. Because of property (10.1.50), a simple recursive scheme $\mathbf{R}_N^{n+1} = \mathrm{proj}_X (\mathbf{d}_N (\mathbf{R}_N^n))$ converges to a fixed point, being the root of $\mathbf{C}_N$. Also, semi-positive definiteness of the Jacobian $\partial \mathbf{C}_N / \partial \mathbf{R}_N$ represents a useful fact, when Newton scheme is applied to the root finding problem (10.1.49). In a more general sense, monotone mappings can sometime be identified with generalised gradients of convex functions (cf. [**183**, p. 547] or [**180**]). In this context, the root of (10.1.49) corresponds to a minimum of such function. This argument can be sometimes used to argue about existence and uniqueness of solutions.

10.1.9. Contact problem as minimisation. The discrete contact problem can also be stated as the following *convex program*

$$\begin{array}{l} \min_{\mathbf{u}} L (\mathbf{u}) \\ \bar{U}_{\alpha N} (\mathbf{u}) \geq 0 \end{array} \tag{10.1.59}$$

where L was defined as (7.0.3), and $\bar{U}_N$ is given in (10.1.27). The convexity of $\mathcal{U} = \left\{ \mathbf{u} : \bar{U}_{\alpha N} (\mathbf{u}) \geq 0 \right\}$ results from the affine structure of the constraints. L is strictly convex and it attains a unique minimum at the velocity of an unconstrained motion (cf. Chapter 7). Hence, the existence of a solution for the constrained problem relies solely on the constraints. An obvious necessary condition is that $\mathcal{U} \neq \emptyset$. For $\mathcal{U}$ to become empty, it is enough to have a pair of contradictory

constraints, similarly as in the example given in the last paragraph of Section 10.1.7. Assume, that this is not the case. A vector $\mathbf{R}_N = [..., R_{\alpha N}, ...]$, where $R_{\alpha N} \geq 0$ will be called a *Kuhn-Tucker vector*, if the infimum of the function

$$f(\mathbf{u}) = L(\mathbf{u}) - \sum_{\alpha} R_{\alpha N} \bar{U}_{\alpha N}(\mathbf{u}) \tag{10.1.60}$$

is equal to the optimal value of the convex program (10.1.59). If such vector exists and is known, one can easily compute the solution. Since $f(\mathbf{u})$ is strictly convex (as a sum of L and some linear functions), its unconstrained minimisation will lead to a unique point, being the solution of (10.1.59).

The Kuhn-Tucker vector, if it exists, naturally leads to the complementarity conditions (10.1.34). This follows from the fact that by definition $R_{\alpha N} \geq 0$, $\bar{U}_{\alpha} \geq 0$ and the minimum of f is equal to $L(\bar{\mathbf{u}})$ for some $\bar{\mathbf{u}} \in \mathcal{U}$. As generally there holds $f \leq L$, the only situation for which $f = L$ is possible requires $R_{\alpha N}\bar{U}_{\alpha N} = 0$. The linear complementary problem (10.1.35) can then be viewed as a summary of the optimality conditions of the convex program (10.1.59). This is readily seen, as $\mathbf{U}_N = \mathbf{W}_{NN}\mathbf{R}_N + \mathbf{B}_N$ is merely an algebraic transformation of $\nabla f = 0$.

By the definition of the normal cone (10.1.21) and because of the complementarity (10.1.34) shown in the previous paragraph, the optimality condition $\nabla f = 0$ can be expressed as

$$-\nabla L(\bar{\mathbf{u}}) \in N_{\mathcal{U}}(\bar{\mathbf{u}}) \tag{10.1.61}$$

where the normal cone reads

$$N_{\mathcal{U}} = \begin{cases} -\sum R_{\alpha N} \nabla \bar{U}_{\alpha N} & \text{when } \mathbf{u} \in \partial\mathcal{U} \\ \{\mathbf{0}\} & \text{when } \mathbf{u} \in \text{interior}(\mathcal{U}) \\ \emptyset & \text{otherwise} \end{cases} \tag{10.1.62}$$

Existence of the Kuhn-Tucker vector corresponds then to $\mathcal{U}$ having a boundary, which seems to be trivially true for all kinds of $\mathcal{U}$. However, not all kinds of the boundaries are equally "good". Definition of $N_{\mathcal{U}}$ is not precise in this respect. A degeneracy corresponds to $\mathcal{U}$ being a single point, and hence $\partial\mathcal{U} = \mathcal{U}$. Of course then $\mathcal{U} = \{\bar{\mathbf{u}}\}$, comprises only the solution. At the same time, there is no restriction on the direction of the gradient $\nabla L(\bar{\mathbf{u}})$. If we admit for an instant $\bar{U}_{\alpha N}(\mathbf{u})$ to be a general smooth functions, then the singleton $\{\bar{\mathbf{u}}\}$ can be obtained in a variety of ways. For example two curves can touch just at this single point. For such a case the corresponding gradients $\nabla \bar{U}_{\alpha N}$ span only a single line and hence $\nabla L(\bar{\mathbf{u}})$ cannot be, in general, expressed as their linear combination. The Kuhn-Tucker vector is not guaranteed to exist, if the constraints are nonlinear and $\partial\mathcal{U} = \mathcal{U}$. In our case however, the constraints are linear. An intersection of half-planes can be a point. But then, their normals need to span the complete space. The Kuhn-Tucker vector is then guaranteed to exist, if only $\mathcal{U} \neq \emptyset$.

Let us summarise. The strict convexity of L, the fact that it admits a finite unconstrained minimum, and the linearity of constraints ensure existence of a unique solution to (10.1.59) whenever $\mathcal{U} \neq \emptyset$. Presence of redundant constraints does not alter this conclusion, as the "shape" of $\mathcal{U}$ is not changed. The only consequence of redundancy is the non-uniqueness of the corresponding Kuhn-Tucker vector. A through exposition of the related issues can be found in Rockafellar [**182**, pp. 273-290]. The result corresponding to the linear constraints is given there in Corollary 28.2.2.

10.1.10. Quasi-statics. In case of quasi-statics, we would like to exploit to some extent complementarity between the gap function and the contact reaction. That is

$$g(t) \geq 0, \;\; R_N(t) \geq 0, \;\; g(t) R_N(t) = 0 \tag{10.1.63}$$

where the above is assumed to hold almost everywhere in a considered time interval (so we do not need to worry, what $R_N(t)$ means during impacts). In the view of the implicit Euler scheme adopted in Section 5.3, the gap function discretisation reads

$$g^{t+h} = g^t + hU_N^{t+h} \tag{10.1.64}$$

The discretised gap-force complementarity can be rewritten as

$$g^{t+h} \geq 0, \;\; R_N \geq 0, \;\; g^{t+h} R_N = 0 \tag{10.1.65}$$

The above relation can be divided by h resulting in

$$\frac{g^t}{h} + U_N^{t+h} \geq 0, \;\; R_N \geq 0, \;\; \left(\frac{g^t}{h} + U_N^{t+h}\right) R_N = 0 \tag{10.1.66}$$

The following substitution

$$\bar{U}_N = \frac{\max(0, g^t)}{h} + U_N^{t+h} \tag{10.1.67}$$

allows for (10.1.66) to be rewritten in a modified form

$$\bar{U}_N \geq 0, \;\; R_N \geq 0, \;\; \bar{U}_N R_N = 0 \tag{10.1.68}$$

which bears the name of the *quasi-inelastic shock law* [**102**]. It is seen that (10.1.68) corresponds to the gap complementarity (10.1.65), if the contact at t is not established. It is related to the velocity complementarity (10.1.34) with zero restitution $\eta = 0$, in case of an established contact. The contact law (10.1.68) is adopted here as it is numerically better behaved compared with (10.1.65). This is related to the low deformability of the utilised kinematic models, for which cancellation of negative gaps might result in excessively high contact reactions. The discussion of the previous sections applies without changes, once $\bar{U}_N$ defined according to (10.1.67) is employed instead of the one defined in (10.1.27).

10.2. The friction problem

While the contact problem was derived from a purely kinematic idea of non-penetration, the friction problem deals with the *resistance* with respect to the tangential motion. As such, it needs to be stated in terms of forces, and eventually linked with a kinematic effect of their action. A simple visualisation could comprise a coin resting on a table top. A sufficiently small horizontal force applied to the coin is not able to alter its position. Only after some threshold value is reached, the coin will start moving. The motion will be opposed by the friction force. This can be summarised as follows

$$\begin{cases} \|\mathbf{R}_T\| \leq F & \\ \|\mathbf{R}_T\| < F & \Rightarrow \mathbf{U}_T = \mathbf{0} \\ \|\mathbf{R}_T\| = F & \Rightarrow \exists_{\lambda \geq 0} \mathbf{U}_T = -\lambda \mathbf{R}_T \end{cases} \tag{10.2.1}$$

where F the threshold value. The above relation is sometimes called *Tresca's friction law*. A distinctive feature and the core of simplification is in the lack of coupling with the contact problem, as F is assumed fixed and arbitrary. The above relation is assumed to hold almost everywhere in a considered time interval. From now on the time discretised case is considered only.

10.2.1. Retrieving the projection formula. As in Section 10.1.5, we shall derive an equality form of relation (10.2.1). The Tresca friction law can be expressed in form of the *maximal dissipation principle*

$$\mathbf{R}_T(t) \in D(F), \quad \forall \mathbf{S} \in D(F), \quad \langle \mathbf{U}_T, \mathbf{S} - \mathbf{R}_T \rangle \geq 0 \tag{10.2.2}$$

where $D(F)$ is a two-dimensional $\mathbf{0}$-centred disc of radius F, $\mathbf{R}_T$ is the tangential reaction and $\mathbf{U}_T$ is the tangential relative velocity. In the above $\langle \cdot, \cdot \rangle$ stands for the scalar product with respect to the local tangent coordinates. The norm in (10.2.1) is related to the inner product through

$$\|\mathbf{R}_T\|^2 = \left\langle \mathcal{A}_{TT}^{-1} \mathbf{R}_T, \mathbf{R}_T \right\rangle \tag{10.2.3}$$

where contravariant components of $\mathbf{R}_T$ were obtained by inverting $R_i = \mathcal{A}_{ij} R^j$ and using the structure of (10.1.39). The disk $D(F)$ can then be deformed into a skewed ellipse, which allows to account for an anisotropy. The friction force smaller than F implies *sticking*, while *sliding* occurs for the tangential force of value F, and with the direction opposite to the slip velocity. Equivalence of (10.2.2) and (10.2.1) can be again verified by inspection. If $\|\mathbf{R}_T\| < F$, then $\mathbf{S} - \mathbf{R}_T$ is allowed to have all possible direction in E^2. Hence, $\mathbf{U}_T = \mathbf{0}$. On the other hand if $\|\mathbf{R}_T\| = F$, then for $\langle \mathbf{U}_T, \mathbf{S} - \mathbf{R}_T \rangle \geq 0$ to hold, $\mathbf{U}_T$ must be normal to the disk $D(F)$ at point $-\mathbf{R}_T$. Hence, $\mathbf{U}_T = -\lambda \mathbf{R}_T$ and $\lambda \geq 0$. The inequality in (10.2.2) can be rewritten as

$$\left\langle \mathcal{A}_{TT}^{-1} \mathbf{R}_T - \left(\mathcal{A}_{TT}^{-1} \mathbf{R}_T - \rho \mathbf{U}_T \right), \mathbf{S} - \mathbf{R}_T \right\rangle \geq 0 \tag{10.2.4}$$

where $\rho > 0$. In analogy with (10.1.37-10.1.38) and Figure 10.1.3 one can write

$$\mathbf{R}_T = \mathcal{A}_{TT} \mathrm{proj}_{D(F)} \left(\mathcal{A}_{TT}^{-1} \mathbf{R}_T - \rho \mathbf{U}_T \right) \tag{10.2.5}$$

Having acknowledged the above possibility, we shall assume in the following, that *the local frame* $\mathbf{a}_i$ *is always orthonormal*, and hence $\mathcal{A} \equiv \mathbf{I}$. The following, simplified form of the projection formula will be further employed

$$\mathbf{R}_T = \mathrm{proj}_{D(F)} \left(\mathbf{R}_T - \rho \mathbf{U}_T \right) \tag{10.2.6}$$

In dynamic applications it might be of use sometimes to account for a "tangential shock", resulting in the velocity restitution rather then sticking. For such case Moreau [**154**] proposed to replace $\mathbf{U}_T$ in (10.2.1-10.2.6) with a convex combination

$$\bar{\mathbf{U}}_T = \frac{1}{1+\tau} \mathbf{U}_T^+ + \frac{\tau}{1+\tau} \mathbf{U}_T^- \tag{10.2.7}$$

The sticking condition $\bar{\mathbf{U}}_T = 0$ implies then $\mathbf{U}_T^+ = -\tau \mathbf{U}_T^-$, where $\tau \in [0, 1]$ is the tangential coefficient of restitution.

10.2.2. Friction problem as root finding. Let us denote

$$\mathbf{d}_T(\mathbf{U}_T, \mathbf{R}_T) = \mathbf{R}_T - \rho \mathbf{U}_T \tag{10.2.8}$$

and call $\mathbf{d}_T$ a *tangential predictor*. Like in the work of Hüeber *et al.* [**96**] we can rewrite the single point Tresca constraint (10.2.1) as

$$\mathbf{C}_T(\mathbf{U}_T, \mathbf{R}_T) = \max(F, \|\mathbf{d}_T\|)\mathbf{R}_T - F\mathbf{d}_T = \mathbf{0} \tag{10.2.9}$$

Gathering all the constraints into a vector operator and using local dynamics, we can then state the following root finding problem

$$\mathbf{C}_T(\mathbf{U}_T, \mathbf{R}_T) = \mathbf{0}|_{\mathbf{U}_T = \mathbf{W}_{TT}\mathbf{R}_T + \mathbf{B}_T} \tag{10.2.10}$$

or in short

$$\mathbf{C}_T(\mathbf{R}_T) = \mathbf{0} \tag{10.2.11}$$

Operator $\mathbf{C}_T$ can be rewritten as

$$\mathbf{C}_T(\mathbf{R}_T) = \mathbf{R}_T - \mathrm{proj}_Y(\mathbf{d}_T(\mathbf{R}_T)) \tag{10.2.12}$$

where $Y = D(F_1) \times D(F_2) \times ... \times D(F_n)$ is convex. By exactly the same argument as for $\mathbf{C}_N$ in Section 10.1.8, $\mathbf{C}_T$ is monotone for $\rho < \frac{2}{\lambda_{max}}$. Similarly as before, this hints, that the root finding problem (10.2.11) is well behaved.

10.2.3. Friction problem as minimisation. The discrete friction problem can also be stated as the following convex program

$$\begin{array}{c} \min_{\mathbf{R}} L^*_{\mathbf{H}}(\mathbf{R}) \\ \|\mathbf{R}_{\alpha T}\| \leq F_\alpha, \;\; R_{\alpha N} = 0 \end{array} \tag{10.2.13}$$

where $L^*_{\mathbf{H}}$ was defined in (7.0.16) and the normal reaction $\mathbf{R}_N = \mathbf{0}$ was cancelled, in order to preserve the decoupled character of friction and contact problems. As the constraints are put on the forces, it is most convenient to utilise the dual formulation with the local conjugate of L as the merit function. $L^*_{\mathbf{H}}$ is convex, although not strictly so, when $\mathbf{H}$ is not of full rank. Hence the minima, if they exist, do not need to be unique. If we quite reasonably assume that all $F_\alpha > 0$, then there is $\mathbf{R}_T \in \mathrm{interior}(Y)$, where $Y = D(F_1) \times D(F_2) \times ... \times D(F_n)$. The reasoning given in Section 10.1.59, and more rigorously Theorem 28.2 in [**182**, p. 277], ensure then the existence of the Kuhn-Tucker vector for the problem (10.2.13). Note, that the elements of the Kuhn-Tucker vectors λ_α have been used in the definition of the Tresca condition, when $\|\mathbf{R}_{\alpha T}\| = F_\alpha$ then $\exists_{\lambda_\alpha \geq 0} \mathbf{U}_{\alpha T} = -\lambda_\alpha \mathbf{R}_{\alpha T}$. Because $\mathbf{U} = \nabla L^*_{\mathbf{H}}(\mathbf{R})$, Tresca law (10.2.1) corresponds in fact to the optimality conditions of the convex program (10.2.13). Assume that $\mathbf{R}_T$ is a solution. When $\|\mathbf{R}_{\alpha T}\| < F_\alpha$ then $\mathbf{U}_T = \mathbf{0}$, because the minimum is unconstrained. Otherwise $\mathbf{U}_{\alpha T} = -\lambda_\alpha \mathbf{R}_{\alpha T} / \|\mathbf{R}_{\alpha T}\|$ belongs to the normal cone $N_Y(\mathbf{R}_T)$.

In summary, the solution is guaranteed to exist if all $F_\alpha > 0$. It is not unique, when $\mathbf{H}$ is not of full rank. Of course here, as well as in Section 10.1.59, convexity of the optimisation problems remains in direct relation with the monotonicity of the corresponding root finding problems.

10.3. The frictional contact problem

This is the case where the analogies related to convexity break down. Unfortunately, this is also the most realistic case. Constraints on the friction forces are now described by the Coulomb law

$$\left\{\begin{array}{ll} \|\mathbf{R}_T\| \leq \mu R_N & \\ \|\mathbf{R}_T\| < \mu R_N & \Rightarrow \mathbf{U}_T = \mathbf{0} \\ \|\mathbf{R}_T\| = \mu R_N & \Rightarrow \exists_{\lambda \geq 0} \mathbf{U}_T = -\lambda \mathbf{R}_T \end{array}\right. \tag{10.3.1}$$

where $\mu \geq 0$ is the *Coulomb's coefficient of friction*. As the normal reaction is employed, the above conditions need to be stated together with the Signorini law

$$\bar{U}_N \geq 0, \;\; R_N \geq 0, \;\; \bar{U}_N R_N = 0 \tag{10.3.2}$$

The frictional contact problem will be also called the Signorini-Coulomb problem. As it was shown, the contact problem alone can be most naturally stated as constrained minimisation with respect to velocities. On the other hand, the friction problem can be most naturally phrased as constrained minimisation with respect to forces. In an attempt of merging these two, one fails to identify a single field, be it primal or dual, optimisation problem for the Signorini-Coulomb law. This is quite a shortcoming, both in theory and practice. This feature of the frictional contact law is often referred to as *lack of normality* or as being *non-associated*.

10.3.1. Projection formulae. In the view of (10.1.38) and (10.2.6) the projection formulae for the frictional contact problem read

$$\left\{\begin{array}{l} R_N = \mathrm{proj}_{R_+}\left(R_N - \rho \bar{U}_N\right) \\ \mathbf{R}_T = \mathrm{proj}_{D(\mu R_N)}\left(\mathbf{R}_T - \rho \mathbf{U}_T\right) \end{array}\right. \tag{10.3.3}$$

A single projection formulation is also possible. This might be beneficial in numerical applications. The formula is due to De Saxcé and Feng [**55**] and is given here for the sake of completeness. There follows

$$\mathbf{R} = \mathrm{proj}_{C(\mu)}\left(\mathbf{R} - \rho \left[\begin{array}{c} \mathbf{U}_T \\ \bar{U}_N + \mu \|\mathbf{U}_T\| \end{array}\right]\right) \tag{10.3.4}$$

where $C(\mu)$ is the *friction cone*

$$C(\mu) = \{\mathbf{R} : \|\mathbf{R}_T\| \leq \mu R_N, \;\; R_N \geq 0\} \tag{10.3.5}$$

More will be said about (10.3.4) in Section 10.3.4.

10.3.2. Potentials, normality, monotonicity and association. It is common in mechanics to prescribe a relation between primal and dual variables, which accounts for an observable physical phenomenon. Such relation is customarily called a *constitutive equation*. An example was given in Section 4.1.2, where the Saint Venant - Kirchhoff material was specified. There, a function Ψ was assumed to exist such that

$$\bar{\mathbf{P}} = \partial \Psi(\mathbf{F}) / \partial \mathbf{F} \tag{10.3.6}$$

where $\bar{\mathbf{P}}$ was a stress, and $\mathbf{F}$ was a deformation gradient. Whenever Ψ is a convex function, its sum over a domain can be minimised (also in the presence of constraints on $\mathbf{F}$), which corresponds to the solvability of a static boundary value problem. Convexity of Ψ allows also to derive a conjugate relation

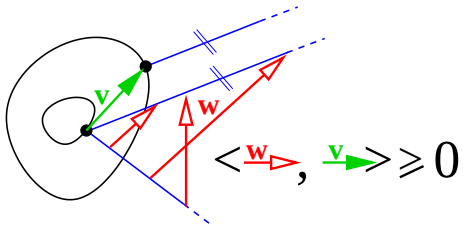


FIGURE 10.3.1. Monotonicity of convex functions. Closed contours correspond to the level curves of a convex function.

$$\mathbf{F} = \partial\Psi^* \left(\bar{\mathbf{P}}\right) / \partial \bar{\mathbf{P}} \tag{10.3.7}$$

where Ψ^* is the Legendre-Fenchel transform of Ψ (cf. Chapter 7). Any of the above relations can be at first discovered in a form of equality $\bar{\mathbf{P}}{=}\psi\left(\mathbf{F}\right)$ or $\mathbf{F}{=}\psi^{-1}\left(\bar{\mathbf{P}}\right)$. At a later point one can realise, that there exists a suitable *potential* Ψ (or Ψ^*), with which ψ (or ψ^{-1}) is *associated* by being its gradient. This usually corresponds to some symmetries in the structure of ψ. Whenever however such an identification cannot be pursued, the constitutive relation ψ needs to be registered as *non-associated*. In other words, there does not exist a potential, whose gradient ψ might be. Looking at the same thing from the geometrical point of view, one can see that $\nabla f|_{x=x_0}$ corresponds to a normal to the level curve $f\left(x\right) = f\left(x_0\right)$ taken at the point x_0. This is why, the above relations are said to comply with *normality*. Hence in general, one does not need for Ψ to be convex in order to speak about normality. In fact, convexity is also not necessary in order to speak about association. However then, Ψ and Ψ^* are not any more conjugate in the sense of being the Legendre-Fenchel transforms of each other. When convexity is present, each level curve $f\left(x\right) = a$ bounds a convex set $S_a = \{x : f\left(x\right) \leq a\}$. For any two arguments x and y, there holds either $S_{f(x)} \subseteq S_{f(y)}$ or $S_{f(y)} \subseteq S_{f(x)}$. Convexity assures, that $\langle \nabla f_x - \nabla f_y, x - y\rangle \geq 0$ because

$$\langle \nabla f_y, x - y\rangle \leq f\left(x\right) - f\left(y\right) \leq \langle \nabla f_x, x - y\rangle \tag{10.3.8}$$

This is equivalent to the *monotonicity* of the gradient mapping ∇f. A graphical interpretation is given in Figure 10.3.1. Whenever the constitutive function ψ can be identified with a gradient of a convex function, the monotonicity holds. This condition is not necessary though, as ψ can be monotone without corresponding to a gradient (e.g. a non-symmetric positive semi-definite matrix).

In the absence of smoothness the above discussion remains valid, although some technical amendments are necessary. For convex f, the notion of gradient ∇f at x needs to be replaced by the *subgradient* x^*, defined at x by

$$f\left(y\right) \geq f\left(x\right) + \langle x^*, y - x\rangle \text{ for all } y \tag{10.3.9}$$

The subgradient x^* corresponds then to the normal of a supporting plane of the graph of f at x. The set of all subgradients bears the name of *subdifferential*. It is defined at x as

$$\partial f = \{x^* : f\left(y\right) \geq f\left(x\right) + \langle x^*, y - x\rangle \text{ for all } y\} \tag{10.3.10}$$

It is an example of a *set valued* mapping. When $y \in S_{f(x)}$, then $f\left(y\right) \leq f\left(x\right)$ and hence $\langle x^*, y - x\rangle \leq 0$. This shows that the subdifferential is equivalent to the

normal cone at x to the set $S_{f(x)}$, quite like it was said in the vicinity of formulae (10.1.21) or (10.1.62). One more tool is necessary, in order to define potentials encompassing the contact and friction laws. The *indicator function* of a set S is defined as follows

$$\delta\left(x|S\right)=\left\{\begin{array}{ccc} 0 & \text{if} & x\in S \\ \infty & \text{if} & x\notin S \end{array}\right. \tag{10.3.11}$$

Signorini law can now be then expressed as

$$-R_N\in\partial\delta\left(\bar{U}_N|R_+\right) \tag{10.3.12}$$

$$\bar{U}_N\in\partial\delta^*\left(-R_N|R_+\right) \tag{10.3.13}$$

and similar relations can be obtained for the Tresca law

$$-\mathbf{R}_T\in\partial\delta^*\left(\mathbf{U}_T,D\left(F\right)\right) \tag{10.3.14}$$

$$\mathbf{U}_T\in\partial\delta\left(-\mathbf{R}_T,D\left(F\right)\right) \tag{10.3.15}$$

Function $\delta^*\left(\cdot,S\right)$ is the Legendre-Fenchel transform of $\delta\left(\cdot,S\right)$ and it is also called the *support function* of the convex set S. We have

$$\delta^*\left(x^*,S\right)=\sup_x\left\{\left\langle x,x^*\right\rangle-\delta\left(x,S\right)\right\}=\sup_{x\in S}\left\langle x,x^*\right\rangle \tag{10.3.16}$$

One might like to note, that the support function was already employed in the fifth line of Algorithm 9.4.4. Existence of nonsmooth, yet convex potentials for the contact and friction problems additionally confirms their well-behavedness.

10.3.3. Lack of potential, normality, monotonicity and association. One can show the lack of monotonicity of the frictional contact law. This implies, that there does not exist a convex potential related to it. Let us first note, that monotonicity of the pure friction law is related to the inequality

$$\left\langle\mathbf{R}_T^1-\mathbf{R}_T^2,\mathbf{U}_T^1-\mathbf{U}_T^2\right\rangle\leq 0 \tag{10.3.17}$$

which holds true for all pairs of $\mathbf{U}_T$ and $\mathbf{R}_T$ verifying Tresca's relation. In the infinitesimal sense, this implies dissipation of energy for all increments of the variables. In order to obtain the $\geq$ inequality, one should use $-\mathbf{R}_T$ instead, similarly like in (10.3.14) and (10.3.15). This is merely a matter of convention. The important bit is in preserving the particular kind of inequality for *all* pairs of variables verifying an interface law. It is then enough to show, that one kind of inequality cannot hold, in order to prove nonmonotonicity. This is simply done for the frictional contact law. Let us take

$$\mathbf{R}_T^1=\alpha\mathbf{R}_T^2,\ \ \left\|\mathbf{R}_T^i\right\|=\mu R_N^i,\ \ \alpha>0 \tag{10.3.18}$$

$$\mathbf{U}_T^1=\beta\mathbf{U}_T^2,\ \ \beta>0 \tag{10.3.19}$$

Then, there holds

$$\left\langle\mathbf{R}_T^2-\mathbf{R}_T^1,\mathbf{U}_T^2-\mathbf{U}_T^1\right\rangle=\left(1-\alpha\right)\left(1-\beta\right)\left\langle\mathbf{R}_T^2,\mathbf{U}_T^2\right\rangle \tag{10.3.20}$$

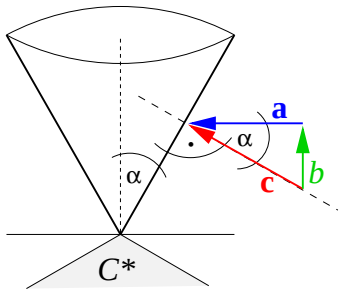


FIGURE 10.3.2. Construction of a vector $\mathbf{c} = [\mathbf{a}, b]^T$, such that $-\mathbf{c} \in N_{C(\mu)}$ from a horizontal$\mathbf{a}$ and $b = \tan(\alpha)\,\|\mathbf{a}\|$.

Since α and β are not related, any sign in the above scalar product is possible. Hence, the frictional contact law is not monotone. In connection with optimisation, analogous observation can be made with respect to the normality. Assume, that we would like to extend problem (10.2.13) to the following one

$$\begin{array}{c} \min_{\mathbf{R}} L^*_{\mathbf{H}}(\mathbf{R}) \\ \|\mathbf{R}_{\alpha T}\| \leq \mu R_{\alpha N}, \;\; R_{\alpha N} \geq 0 \end{array} \tag{10.3.21}$$

The constraints state, that the contact reactions belong to the friction cone (10.3.5). When considering optimality conditions for the above problem, one can notice that the velocity

$$-\left[\begin{array}{c} \mathbf{U}_T \\ \bar{U}_N \end{array}\right] \notin N_{C(\mu)}(\mathbf{R}) \tag{10.3.22}$$

does not belong to the normal cone of the friction cone $C(\mu)$. This is because, the normal velocity restitution law "knows nothing" about the μ-slope of the friction cone. Fore example when Newton's restitution coefficient is $\eta = 0$, then $\mathbf{U}$ is parallel to the plane spanned by $\mathbf{a}_{1T}$ and $\mathbf{a}_{2T}$. In consequence, the gradient of $-L^*_{\mathbf{H}}$ cannot be expressed as a linear combination of gradients of the active constraints. The Kuhn-Tucker vector does not exist and one cannot establish the first order optimality conditions. Hence, the frictional contact problem cannot be perceived as minimisation.

10.3.4. The Bipotential Method. A formal workaround for the lack of normality of the frictional contact law was proposed by De Saxcé and Feng [**55**]. Although (10.3.22) cannot be helped, one can modify the left hand side, so that a vector belonging to the normal cone (of the friction cone) is obtained. That is

$$-\left[\begin{array}{c} \mathbf{U}_T \\ \bar{U}_N + \mu\,\|\mathbf{U}_T\| \end{array}\right] \in N_{C(\mu)}(\mathbf{R}) \tag{10.3.23}$$

for which a simple geometrical explanation is given in Figure 10.3.2. It turns out, that the above inclusion implies the frictional contact law. For $R_N > 0$ there must hold $\bar{U}_N = 0$ and then $-\mathbf{U}_T \in N_{D(\mu R_N)}(\mathbf{R}_T)$. For $R_N = 0$ the normal cone $N_{C(\mu)}(\mathbf{R})$ is the *polar cone* C^* of the friction cone $C(\mu)$, marked in Figure 10.3.2. In this case the geometrical construction allows any $\bar{U}_N \geq 0$ which retrieves the Signorini condition. For $R_N = 0$ no restriction is put on the magnitude of the slip velocity $\|\mathbf{U}_T\|$. Of course, by construction, the frictional contact law implies inclusion (10.3.23), which establishes their equivalence.

Inclusion (10.3.23), although useful on its own, can be further shown to fit into the framework of *Implicit Standard Materials*, proposed by the authors in [**55**]. As discussed in the previous section, there does not exist a single-field convex potential, whose gradient expresses the frictional contact law. The authors consider instead *bipotentials*, that is functions implicitly handling a relation between dual variables. By definition, a bipotential is

$$X \times X^* \to [-\infty, +\infty] : (x, x^*) \to b(x, x^*) \tag{10.3.24}$$

where $b(\cdot, x^*)$ and $b(x, \cdot)$ are *separately* convex, lower semi-continuous[2], and such that for all x and x^* there holds

$$b(x, x^*) \geq \langle x, x^* \rangle \tag{10.3.25}$$

The above inequality allows to write

$$x \in \partial_{x^*} b(x, x^*) \tag{10.3.26}$$

$$x^* \in \partial_x b(x, x^*) \tag{10.3.27}$$

which follows from (10.3.25) as

$$b(x, y^*) \geq b(x, x^*) + \langle x, y^* - x^* \rangle \text{ for all } y^* \tag{10.3.28}$$

$$b(y, x^*) \geq b(x, x^*) + \langle y - x, x^* \rangle \text{ for all } y \tag{10.3.29}$$

Condition (10.3.25) is a generalisation of Fenchel's inequality

$$f(x) + f^*(x) \geq \langle x, x^* \rangle \tag{10.3.30}$$

which in turn is a consequence of $f^*(x^*) = \sup_x \{\langle x, x^* \rangle - f(x)\}$. Let $\bar{\mathbf{U}} = \left[\mathbf{U}_T, \bar{U}_N\right]^T$. In [**55**] a bipotential for the frictional contact law is defined as

$$b\left(\bar{\mathbf{U}}, \mathbf{R}\right) = \delta\left(-\bar{U}_N | R_-\right) + \delta\left(\mathbf{R}, C(\mu)\right) + \mu R_N \left\| -\mathbf{U}_T \right\| \tag{10.3.31}$$

and inclusion (10.3.23) is shown to be equivalent to

$$-\bar{\mathbf{U}} \in \partial_{\mathbf{R}} b\left(\bar{\mathbf{U}}, \mathbf{R}\right) \tag{10.3.32}$$

This partially brings back the idea of normality. Nevertheless, it does not remove difficulties related to the solution of the frictional contact problem.

10.3.5. Frictional contact as root finding. Similarly to the work of Hüeber *et al.* [**96**] we can state the single point frictional contact operator as

$$\mathbf{C}(\mathbf{U}, \mathbf{R}) = \begin{bmatrix} \max(\mu d_N, \|\mathbf{d}_T\|)\, \mathbf{R}_T - \mu \max(0, d_N)\, \mathbf{d}_T \\ R_N - \max(0, d_N) \end{bmatrix} \tag{10.3.33}$$

where the components of the predictor

$$\mathbf{d}(\mathbf{U}, \mathbf{R}) = \begin{bmatrix} \mathbf{d}_T(\mathbf{U}_T, \mathbf{R}_T) \\ d_N(U_N, R_N) \end{bmatrix} \tag{10.3.34}$$

[2] A function f is *lower semi-continuous* if for all α sets $\{x : f(x) \leq \alpha\}$ are closed.

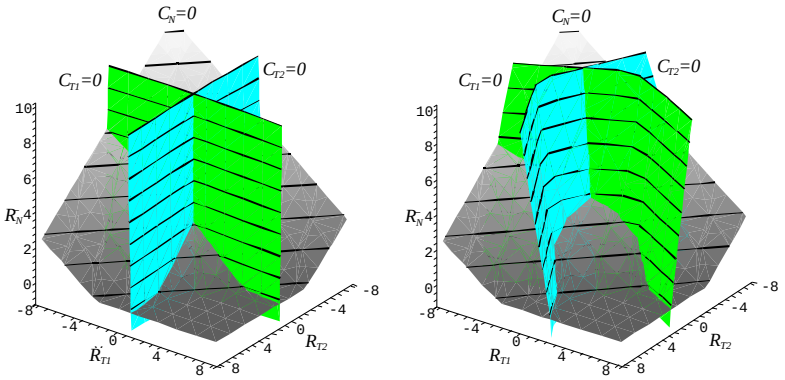


FIGURE 10.3.3. Plots of the Signorini-Coulomb constraints for $\rho = 1$, $\eta = 0$, $W_{ii} = 0.8$, $W_{i\neq j} = 0.4$, $B_{T1} = B_{T2} = 0$, $B_N = -2$. The coefficient of friction was $\mu = 0.1$ in the left picture, and $\mu = 0.9$ in the right one.

were defined in (10.1.46) and (10.2.8). As usual, the root finding problem $\mathbf{C}(\mathbf{R}) = \mathbf{0}$ can be stated by eliminating $\mathbf{U} = \mathbf{W}\mathbf{R} + \mathbf{B}$. This time however, not much can be said about its structure. Some visualisation is possible, at most, for a single point problem. The three scalar equations in $\mathbf{C}(\mathbf{R}) = \mathbf{0}$ describe some implicit surfaces in the $\mathbf{R}$-space. This is depicted in Figure 10.3.3. The inclined surface describes the normal constraint $C_N(\mathbf{R}) = 0$. The two nearly vertical (aligned with the R_N axis) surfaces correspond to the components of the tangential constraint $\mathbf{C}_T(\mathbf{R}) = \mathbf{0}$. The solution rests at the intersection point of all three surfaces. Small friction coefficient in the left picture results in frictional slipping. Larger coefficient in the right picture allows to pronounce the transition from the slip to the stick state. The constraint surfaces are visibly curved, as the non-smooth transition bends them along vertical lines. The solution point in the right picture is in the state of sticking. Essentially, for problems with many contact points, one is interested in finding intersection points like those in Figure 10.3.3.

Surely, operator $\mathbf{C}(\mathbf{R})$ is not monotone. This implies that there exist $\mathbf{R}_1$, $\mathbf{R}_2$ such that

$$\langle \mathbf{C}(\mathbf{R}_1) - \mathbf{C}(\mathbf{R}_2), \mathbf{R}_1 - \mathbf{R}_2 \rangle < 0 \tag{10.3.35}$$

Thus[3], a linear expansion of $\mathbf{C}(\mathbf{R})$ can experience a negative definite Jacobian. This is not a desirable feature from the point of view of a Newton scheme applied to the solution of the root finding problem $\mathbf{C}(\mathbf{R}) = \mathbf{0}$. When $\mathbf{C}(\mathbf{R})$ is perceived as a gradient of a general nonconvex function, finding $\mathbf{C}(\mathbf{R}) = \mathbf{0}$ can be regarded as looking for a local extremum. A local minimum is preferred as a stable solution, but the negative definiteness of the Jacobian can spoil the convergence. In such case the scheme requires globalisation (line search) in order to avoid divergence or convergence to a local maximum. A technique of this sort will be examined in the next chapter.

[3]**ERRATA:** In our paper [**123**], Section 4.1.3, there is an erroneous statement: "*A simple numerical experiment shows that for data from Figure 10.3.3, and force pairs generated randomly on a unit ball the above inequality holds true in* 30% *of cases* ($\mu = 0.4$)." This was wrongly concluded due to a flawed code. In fact, for randomly generated force pairs and the values of ρ largely exceeding $2/\lambda_{max}$, a small percentage (up to 3%) of negative results (10.3.35) is obtained.

10.3.6. The well-behaved juxtaposed simplification. What happens if we juxtapose the contact problem and the friction problem operators? We have

$$\mathbf{C_F}(\mathbf{R}) = \begin{bmatrix} \mathbf{C}_T(\mathbf{R}) \\ \mathbf{C}_N(\mathbf{R}) \end{bmatrix} = \begin{bmatrix} \mathbf{R}_T - \mathrm{proj}_Y(\mathbf{d}_T(\mathbf{R})) \\ \mathbf{R}_N - \mathrm{proj}_X(\mathbf{d}_N(\mathbf{R})) \end{bmatrix} \tag{10.3.36}$$

where X is the positive orthant $R_+ \times R_+ \times ... \times R_+$ and $Y = D(F_1) \times D(F_2) \times ... \times D(F_n)$. The $\mathbf{F}$ index stands for the vector of all friction thresholds $[..., F_\alpha, ...]$. By exactly the same argument as in (10.1.56), one can show that the above operator is monotone for $\rho < 2/\lambda_{max}$, where this time λ_{max} is the maximal eigenvalue of the complete operator $\mathbf{W}$. Of course, finding $\mathbf{R}$ such that $\mathbf{C_F}(\mathbf{R}) = \mathbf{0}$, although it might be easy, does not solve the frictional contact problem. Nevertheless, one might try to do it repeatedly, while updating $\mathbf{F}$ to the most recent value of $\mu\mathbf{R}_N$. Convergence of such procedure will depend on the fact, whether from the global point of view it can be perceived as a contraction. If so, it shall converge to a fixed point $\mathbf{F} = \mathbf{R}_N$. The issue of convergence was studied by Stadler [**195**], where the uniqueness of solution was shown for sufficiently small coefficient of friction (see also [**168**]). All of the root finding schemes discussed in the next chapter can be regarded as exploiting a contraction property of some operators. Nevertheless, it will be customary to refer to the specified here procedure as the *fixed point method*.

10.3.7. Can the frictional contact operator be monotone? Let us rewrite (10.3.33) as

$$\mathbf{C}(\mathbf{R}) = \begin{bmatrix} \mathbf{C}_T(\mathbf{R}) \\ \mathbf{C}_N(\mathbf{R}) \end{bmatrix} = \begin{bmatrix} \mathbf{R}_T - \mathrm{proj}_Z(\mathbf{d}_T(\mathbf{R})) \\ \mathbf{R}_N - \mathrm{proj}_X(\mathbf{d}_N(\mathbf{R})) \end{bmatrix} \tag{10.3.37}$$

where X is the positive orthant $R_+ \times R_+ \times ... \times R_+$ and $Z = D\left(\mu_1 (d_{1N})_+\right) \times D\left(\mu_2 (d_{2N})_+\right) \times ... \times D\left(\mu_n (d_{nN})_+\right)$. By $(x)_+$ we mean $\max(0, x)$. In analogy with (10.1.58), $\mathbf{C}$ becomes monotone, if only the projection

$$\mathbf{R} = \mathrm{proj}_{Z\times X}(\mathbf{d}(\mathbf{R})) \tag{10.3.38}$$

is a contraction. By Lemma 2 from [**168**], there holds

$$\begin{aligned} &\left\| \mathrm{proj}_{Z(\mathbf{A})}(\mathbf{d}_T(\mathbf{R}_1)) - \mathrm{proj}_{Z(\mathbf{B})}(\mathbf{d}_T(\mathbf{R}_2)) \right\|^2 \leq \\ &\leq \|\mathbf{d}_T(\mathbf{R}_1) - \mathbf{d}_T(\mathbf{R}_2)\|^2 + \mu_{max}^2 \|\mathbf{d}_N(\mathbf{R}_1) - \mathbf{d}_N(\mathbf{R}_2)\|^2 \end{aligned} \tag{10.3.39}$$

where $\mu_{max} = \max_\alpha \mu_\alpha$. One can now write

$$\begin{aligned} &\left\| \mathrm{proj}_{Z\times X}(\mathbf{d}(\mathbf{R}_1)) - \mathrm{proj}_{Z\times X}(\mathbf{d}(\mathbf{R}_2)) \right\|^2 \leq \\ &\leq \|\mathbf{d}(\mathbf{R}_1) - \mathbf{d}(\mathbf{R}_2)\|^2 + \mu_{max}^2 \|\mathbf{d}(\mathbf{R}_1) - \mathbf{d}(\mathbf{R}_2)\|^2 \end{aligned} \tag{10.3.40}$$

and further, by the same argument as in (10.1.56), obtain

$$\begin{aligned} &\left\| \mathrm{proj}_{Z\times X}(\mathbf{d}(\mathbf{R}_1)) - \mathrm{proj}_{Z\times X}(\mathbf{d}(\mathbf{R}_2)) \right\|^2 \leq \\ &\leq \left(1 + \mu_{max}^2\right) \left(1 - \tfrac{2}{\lambda_{max}} \rho\beta^2 + \rho^2\beta^2\right) \|\mathbf{R}_1 - \mathbf{R}_2\|^2 \end{aligned} \tag{10.3.41}$$

According to (10.1.57), the minimum over ρ of the second bracket above is bounded by $1 - \lambda_{min}^2/\lambda_{max}^2$ and hence the condition $\left(1 + \mu_{max}^2\right)\left(1 - \lambda_{min}^2/\lambda_{max}^2\right) \leq 1$ resolves as

$$\mu_{max} \leq \sqrt{\frac{\lambda_{min}^2}{\lambda_{max}^2 - \lambda_{min}^2}} \tag{10.3.42}$$

The projection $\mathrm{proj}_{Z\times X}(\cdot)$ is a contraction and $\mathbf{C}(\mathbf{R})$ becomes monotone, only for a sufficiently small coefficient of friction, where λ_{min} and λ_{max} are the minimal and the maximal eigenvalues of $\mathbf{W}$. The above discussion shows, that the frictional contact problem enjoys a unique solution, when condition (10.3.42) is satisfied.

10.4. Cohesion

In our framework cohesion can bond points of distinct bodies, similarly to a glue. It can be used with respect to selected contact points, provided that they were present at $t = 0$. The situation is rather simple. For all contact points we can write

$$\bar{U}_{\alpha N} \geq 0, \;\; R_{\alpha N} + hc_\alpha \geq 0, \;\; \bar{U}_{\alpha N}\left(R_{\alpha N} + hc_\alpha\right) = 0 \tag{10.4.1}$$

where $hc_\alpha \geq 0$ is an integral of the cohesion threshold over $[t, t+h]$. This means that the average normal contact force $R_{\alpha N}/h$ can be negative up to the absolute level of c_α. Only when this value is surpassed, the normal velocity is allowed to become positive, resulting in decohesion. This needs to be followed by setting $c_\alpha = 0$ to indicate a brittle failure. Hence, the cohesion law should be completed by a condition, executed after the solution for contact reactions is preformed. It reads

$$\text{if } c_\alpha > 0 \wedge \left(R_{\alpha N} + hc_\alpha\right) = 0 \text{ then } c_\alpha = 0 \tag{10.4.2}$$

which brings back the adhesion-less form of the Signorini formula. When $c_\alpha > 0$ the remaining formulae in the formulation of the root finding problem need to be updated accordingly. Let us first denote

$$\bar{R}_{\alpha N} = R_{\alpha N} + hc_\alpha \tag{10.4.3}$$

so that

$$\bar{U}_{\alpha N} \geq 0, \;\; \bar{R}_{\alpha N} \geq 0, \;\; \bar{U}_{\alpha N}\bar{R}_{\alpha N} = 0 \tag{10.4.4}$$

holds. If in all necessary formulae we could replace R_Ns with $\bar{R}_N$s, then global picture would not be altered. This can be done by consequently applying the change of coordinates (10.4.3). As $R_{\alpha N} = \bar{R}_{\alpha N} - hc_\alpha$, there holds

$$\mathbf{U} = \mathbf{W}_{*T}\mathbf{R}_T + \mathbf{W}_{*N}\left(\bar{\mathbf{R}}_N - h\mathbf{c}\right) + \mathbf{B} \tag{10.4.5}$$

which can be rewritten as

$$\mathbf{U} = \mathbf{W}\bar{\mathbf{R}} + \bar{\mathbf{B}} \tag{10.4.6}$$

$$\bar{\mathbf{R}} = \left[..., \mathbf{R}_{\alpha T}, \bar{R}_{\alpha N}, ...\right]^T \tag{10.4.7}$$

$$\bar{\mathbf{B}} = \mathbf{B} - \mathbf{W}_{*N} h\mathbf{c} \tag{10.4.8}$$

where the start $*$ stands for all relevant indices, so that $\mathbf{W}_{*N}$ comprises the 1×3 normal column blocks $\mathbf{W}_{\alpha\beta *N} = [\mathbf{W}_{\alpha\beta TN}, W_{\alpha\beta NN}]^T$. Owing to (10.4.4) and (10.4.6), one can now solve the usual root finding problem, involving the projection formulae having R_N replaced by $\bar{R}_N$.

In order to avoid clutter in the notation, *it is from now on assumed*, that the following sequence of steps is executed whenever solution for the constraints is discussed

$$\mathbf{R}_N = \mathbf{R}_N + h\mathbf{c} \tag{10.4.9}$$

$$\mathbf{B} = \mathbf{B} - \mathbf{W}_{*N} h\mathbf{c} \tag{10.4.10}$$

$$\mathbf{C}\left(\mathbf{W}\mathbf{R} + \mathbf{B}, \mathbf{R}\right) = \mathbf{0} \tag{10.4.11}$$

$$\mathbf{R}_N = \mathbf{R}_N - h\mathbf{c} \tag{10.4.12}$$

$$\text{for all } \alpha, \text{ if } c_\alpha > 0 \wedge (R_{\alpha N} + hc_\alpha) = 0 \text{ then } c_\alpha = 0 \tag{10.4.13}$$

This will be recalled, when the complete time stepping schemes are assembled in Chapter 12. For the moment, let us forget about that. A comprehensive discussion about incorporation of more sophisticated interface laws can be found in Jean *et al.* [**103**].

10.5. Energetic consistency

Total energy of a multi-body system should not grow due to the incorporation of contact and friction constraints. In particular, considering a dynamical system without unbalanced forces (a rigid multi-body system with some initial velocity), this statement needs to hold with respect to the kinetic energy. Similarly as in [**42**] one can then write

$$2\left(E_k^+ - E_k^-\right) = \left(\mathbf{u}^+ + \mathbf{u}^-\right)^T \mathbf{M}\left(\mathbf{u}^+ - \mathbf{u}^-\right) \tag{10.5.1}$$

and use the momentum balance over an impact episode

$$\mathbf{M}\left(\mathbf{u}^+ - \mathbf{u}^-\right) = \mathbf{H}^T\mathbf{R} \tag{10.5.2}$$

in order to arrive at

$$\begin{aligned} 2\left(E_k^+ - E_k^-\right) &= \left(\mathbf{u}^+ + \mathbf{u}^-\right)^T \mathbf{H}^T\mathbf{R} \\ &= \left\langle \mathbf{H}\left(\mathbf{u}^+ + \mathbf{u}^-\right), \mathbf{R}\right\rangle \\ &= \left\langle \mathbf{U}^+ + \mathbf{U}^-, \mathbf{R}\right\rangle \end{aligned} \tag{10.5.3}$$

The last formula suggests, that if only $\bar{\mathbf{U}}_T = a\left(\mathbf{U}_T^+ + \mathbf{U}_T^-\right)$ and $\bar{U}_N = b\left(U_N^+ + U_N^-\right)$ were employed in the contact and friction constraints (10.1.34) and (10.3.1), dissipativity could be assured ($a, b > 0$). This would correspond to the fully elastic tangential shock for sticking contacts, and to the fully elastic normal impact for $U_N^- \leq 0$. Such choice, with $a = b = \frac{1}{2}$, is in fact quite natural in the view of the configuration update formula $\mathbf{q}^{t+h} = \mathbf{q}^t + \frac{1}{2}\left(\mathbf{u}^t + \mathbf{u}^{t+h}\right)$. This seems to be the basis of the energetically consistent developments by Laursen and Chawla [**202, 206**]. Nevertheless, in the context of kinematic models with limited deformability, the

fully elastic restitution is rather constraining. Here, one would like to use a variety of restitution coefficients at different contact points, distinct for the normal and tangential components. Unfortunately, for the simple Newton's restitution model energetic consistency can only be assured in few special situations. This, combined with the inconsistencies related to the frictional effects, renders the adopted contact-impact-friction framework only an illustrative tool. Having said that, let us discuss some particular sources of the (in)consistency.

10.5.1. Contacts, ideally plastic impacts, and friction. By a *contact* we mean a situation, where the gap function $g \leq 0$ and $U_N^- \geq 0$. In this case, the contact point should not be excluded from the formulation of the constraints. It is well possible, that due to the kinematic interactions with other contact points, the right velocity becomes negative $U_N^+ < 0$. This should not be allowed. The contact point is then preserved, and $\bar{U}_N = U_N^+ + \eta \min\left(0, U_N^-\right) = U_N^+$. On the other hand, by an *impact* we mean, that $g \leq 0$ and $U_N^- < 0$, and hence $\bar{U}_N = U_N^+ + \eta U_N^-$. Only ideally plastic impacts are considered here, where $\eta = 0$. Thus again, $\bar{U}_N = U_N^+$. In this situation it is easy to show, that dissipativity always holds. We have

$$\mathbf{U}^+ = \mathbf{W}\mathbf{R} + \mathbf{U}^- \tag{10.5.4}$$

and hence

$$\begin{gathered} \langle \mathbf{U}^+ + \mathbf{U}^-, \mathbf{R} \rangle = \langle 2\mathbf{U}^+ - \mathbf{W}\mathbf{R}, \mathbf{R} \rangle = \\ 2\left\langle \mathbf{U}_T^+, \mathbf{R}_T \right\rangle - \langle \mathbf{W}\mathbf{R}, \mathbf{R} \rangle + 2\left\langle \mathbf{U}_N^+, \mathbf{R}_N \right\rangle \end{gathered} \tag{10.5.5}$$

The first scalar product $\left\langle \mathbf{U}_T^+, \mathbf{R}_T \right\rangle \leq 0$, due to the friction law (10.3.1). The quadratic form $\langle \mathbf{W}\mathbf{R}, \mathbf{R} \rangle \geq 0$, because $\mathbf{W}$ is semi-positive definite. Finally $\left\langle \mathbf{U}_N^+, \mathbf{R}_N \right\rangle = 0$, due to the Signorini condition (10.1.34). Thus, $E_k^+ - E_k^- \leq 0$.

10.5.2. Frictionless impacts and contacts. We are assuming now $\mathbf{R}_T = \mathbf{0}$. Let $\mathcal{S}$ be the index set of α, where $U_{\alpha N}^- < 0$ and thus $U_{\alpha N}^+ = -\eta_\alpha U_{\alpha N}^-$ (impacts). Let $\mathcal{P}$ be the index set of α, where $U_{\alpha N}^- \geq 0$ and thus $U_{\alpha N}^+ \geq 0$ (contacts). The question is about the sign of $\left\langle \mathbf{U}_N^+ + \mathbf{U}_N^-, \mathbf{R}_N \right\rangle$. For $\alpha \in \mathcal{P}$ and $U_{\alpha N}^+ > 0$ there follows $R_{\alpha N} = 0$, and hence the corresponding term in the scalar product is zero. It is fair to assume $U_{\alpha N}^+ = 0$ for all $\alpha \in \mathcal{P}$. Then

$$\begin{cases} U_{\alpha N}^+ = -\eta_\alpha U_{\alpha N}^- \text{ for } \alpha \in \mathcal{S} \\ U_{\beta N}^+ = 0 \text{ for } \beta \in \mathcal{P} \end{cases} \tag{10.5.6}$$

For better illustration let $\eta_\alpha = \eta$ for all α. One can now write

$$\mathbf{U}_N^+ + \mathbf{U}_N^- = \begin{bmatrix} (1-\eta)\,\mathbf{U}_{\mathcal{S}N}^- \\ \mathbf{U}_{\mathcal{P}N}^- \end{bmatrix} \tag{10.5.7}$$

$$\mathbf{R}_N = -\mathbf{W}_{NN}^{-1} \begin{bmatrix} (1+\eta)\,\mathbf{U}_{\mathcal{S}N}^- \\ \mathbf{U}_{\mathcal{P}N}^- \end{bmatrix} \tag{10.5.8}$$

Hence

$$\begin{gathered} -\left\langle \mathbf{U}_N^+ + \mathbf{U}_N^-, \mathbf{R}_N \right\rangle = \\ = \left\langle \mathbf{W}_{NN}^{-1} \mathbf{U}_N^-, \mathbf{U}_N^- \right\rangle - \eta^2 \left\langle \mathbf{W}_{NN}^{-1} \begin{bmatrix} \mathbf{U}_{\mathcal{S}N}^- \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{U}_{\mathcal{S}N}^- \\ \mathbf{0} \end{bmatrix} \right\rangle \end{gathered} \tag{10.5.9}$$

In the above, the right hand side difference can become negative. This is seen either from the spectral picture of the scalar products[4], or simply by noticing that subtraction from a quadratic form of some of its diagonal squares turns it into a saddle surface. Hence, the energy growth is possible. On the other hand, for $\mathcal{P}=\emptyset$, dissipativity is restored. This follows also from $\left\langle \mathbf{U}_N^+ + \mathbf{U}_N^-, \mathbf{R}_N \right\rangle = \left\langle (1-\eta)\,\mathbf{U}_N^-, \mathbf{R}_N \right\rangle \leq 0$, as it was shown by Glocker [**42**]. One could then consider only impacts, defined by $U_N^- \leq 0$. This however is not always suitable. For example, for flat surfaces in a statically overdetermined contact, such assumption would lead to a noisy, spurious rocking after the decay of bouncing, for some $\eta < 1$. This would result from successive switching off and on of some contact points, for which the value of U_N^- would oscillate around zero, on the level of numerical tolerances. Processing a nonempty set $\mathcal{P}$ stabilises this sort of behaviour. In the example from Section 13.4.3, the energy growth does not happen, because there indeed $\mathcal{P}=\emptyset$. An additional factor easing off a possible energy growth is the square η^2 in (10.5.9), where $\eta \leq 1$.

10.5.3. Impacts, contacts and friction. Considering now also the frictional effects $\mathbf{R}_T \neq \mathbf{0}$, let us additionally assume that some unbalanced forces do exist. In the context of the dynamic time stepping, we can write

$$\frac{1}{2}\left(\mathbf{u}^+ + \mathbf{u}^-\right)\mathbf{M}\left(\mathbf{u}^+ - \mathbf{u}^-\right) = \frac{h}{2}\left(\mathbf{u}^+ + \mathbf{u}^-\right)\mathbf{f} + \frac{1}{2}\left\langle \mathbf{U}^+ + \mathbf{U}^-, \mathbf{R}\right\rangle \tag{10.5.10}$$

which corresponds to

$$\triangle E_k + \triangle E_p = \frac{1}{2}\left\langle \mathbf{U}^+ + \mathbf{U}^-, \mathbf{R}\right\rangle \tag{10.5.11}$$

where the conservative force $\mathbf{f} = -\partial E_p/\partial \mathbf{q}$, and the fact that $\triangle \mathbf{q} = \frac{h}{2}\left(\mathbf{u}^+ + \mathbf{u}^-\right)$ were used. Taking the total energy $E = E_k + E_p$, one obtains $\triangle E_k + \triangle E_p = E^+ - E^-$, and hence similarly as in the previous case

$$2\left(E^+ - E^-\right) = \left\langle \mathbf{U}^+ + \mathbf{U}^-, \mathbf{R}\right\rangle \tag{10.5.12}$$

Now

$$\mathbf{U}^+ = \mathbf{W}\mathbf{R} + \mathbf{B} \tag{10.5.13}$$

rather than (10.5.4). Thus, derivation of kind (10.5.5) is no longer possible. One can merely write

$$\begin{gathered}\left\langle \mathbf{U}^+ + \mathbf{U}^-, \mathbf{R}\right\rangle = \\ = \left\langle \mathbf{U}_T^+, \mathbf{R}_T\right\rangle + \left\langle \mathbf{U}_{\mathcal{P}N}^+, \mathbf{R}_{\mathcal{P}N}\right\rangle + \left\langle \mathbf{U}_T^-, \mathbf{R}_T\right\rangle + \left\langle -\mathbf{Q}\mathbf{U}_{\mathcal{S}N}^-, \mathbf{R}_{\mathcal{S}N}\right\rangle\end{gathered} \tag{10.5.14}$$

where $\mathbf{Q} = \operatorname{diag}(\eta_\alpha)$, and index sets $\mathcal{S}$ and $\mathcal{P}$ were already defined in the previous section. On the right hand side above, the sum of first two scalar products is seminegative, due to the friction and contact conditions. The last scalar product is positive. The one but the last can be positive, when $\left\langle \mathbf{U}_T^+, \mathbf{U}_T^-\right\rangle \leq 0$ (as $\mathbf{U}_T^+ = -\lambda \mathbf{R}_T$ with $\lambda \geq 0$). This corresponds to *slip reversal*. The amount of positive and negative values in (10.5.14) is the matter of a particular setup. Nothing general can be said about it, maybe with the exception of stating, that the inconsistency results from our model and it should be corrected by developing a better one. It is relevant to mention, that the quasi-static scheme from Section 5.3 remains consistent, as there $\mathbf{U}^- = \mathbf{0}$ by definition.

[4] An unpractical criterion for stability would be $\eta^2 \leq \min_i a_i / \max_j b_j$, where $\mathbf{a}$ is the projection of $\mathbf{U}_N^-$ on the eigenbasis of $\mathbf{W}_{NN}^{-1}$, and $\mathbf{b}$ is a similar projection of $\left[\mathbf{U}_{\mathcal{S}T}^{-1}, \mathbf{0}\right]^T$.

10.6. Literature

A review paper on dynamics of rigid bodies with friction and impacts was given by Stewart [**198**]. Another general review, oriented more towards the event driven strategies is due to Brogliato *et al.* [**32**]. A review paper on the formulation of elastostatic frictional contact problems was also given by Mijar and Arora [**150**]. Monograph by Wriggers [**211**] gives a comprehensive summary of computational techniques related to deformable contact problems.

The Contact Dynamics method developed by Moreau [**156**] and Jean [**102**] is particularly convenient for kinematically modest finite deformation formulations, where the dual form of the contact problem can be inexpensively utilised (e.g. assembly of rigid bodies, for which an explicit inversion of the inertia operator is computationally feasible). The main features of the method comprise the use of a velocity level time stepping, the non-regularised treatment of frictional contact law, and the block Gauss-Seidel relaxation utilised stepwise in order to resolve unilateral constraints. The method was developed within the context of rigid and finite element discretised continua. Applications range from granular flow [**178**], through statics of masonry [**40**], to deep drawing simulations [**109**]. Contact Dynamics belongs to a broader category of schemes dealing with the non-smooth dynamical systems. For example, developments by Wösle and Pfeiffer [**210**] and Pfeiffer *et al.* [**170**] utilise the same theoretical apparatus, although they differ in details (accelerations are involved, Newton scheme is used rather than the Gauss-Seidel).

In the context of rigid multi-body simulations, Stewart and Trinkle [**199**] developed a time-stepping method based on an inelastic impact law and polyhedral linearisation of the Coulomb friction cone. Their formulation does not allow violation of interpenetration constraints, and the resulting linear complementary problem (LCP) is guaranteed to posses solutions at all times. Pang *et al.* [**165**] developed a linear programing technique to solve an uncoupled complementary problem resulting from the planar formulation of a quasi-static evolution of rigid multi-body systems. This work was later extended to three dimensions by Trinkle *et al.* [**204**], where polyhedral discretisation of the friction cone (like in [**199**]) allowed to preserve the original algebraic structure. Anitescu *et al.* [**10**] review several aspects of time-stepping methods for rigid bodies. Anitescu and Potra [**9**] design a time-stepping method for rigid multi-body systems with stiff forces. A linearly implicit time integrator is used in combination with the LCP formulation of constraints. The method is shown to be stable in the stiff limit, where a stiff force joining two points acts as a joint constraint. Potra *et al.* [**70**] describe a second order, stiffly stable linearly implicit time-stepping for rigid multi-body frictional contact problems. They employ an event prediction method, a Poisson restitution model and an LCP formulation of the constraints. The second order convergence rate of the method is exemplified on few examples (cf. Sections 13.4.4 and 13.4.5). Song *et al.* [**194**] describe a linear complementarity based framework for compliant contacts, and prove solvability of the so posed problem. Leine and Glocker [**132**] develop a Coulomb-Contensou frictional law for rigid bodies, where the contact surface is approximated by a disk, allowing to extract a torque-spin relation. The constitutive law is applied in the contact of time-stepping methods, for an example of the tippe-top toy. This work can be related to the one by Goyal *et al.* [**77**], where a similar holistic approach to the motion of rigid, sliding bodies was undertaken. Glocker [**42**] discusses differences between Newton's and Poisson's impact models. Leine *et al.* [**131**] design a simple mechanical system named the Frictional Impact Oscillator, and examine the occurence of Painlevé paradox for their setup. A number of interesting conclusions is drawn, regarding the conditions under which frictional hopping can happen. Stewart [**197**] gives a proof of convergence of a time-stepping

algorithm similar to the one given in [**199**]. He also resolves a particular instance of the Painlevé paradox.

In the context of deformable continua, an augmented Lagrangian formulation of the frictional contact problem was developed by Alart and Curnier [**7**]. The authors derive a linearisation of their formulation and apply Newton method as a solution strategy. This formulations serves as a basis for many subsequent developments. For example, Heegaard and Curnier [**85**] discuss a suitable extension to large slip problems, and Heege and Alart [**86**] develop a finite element for metal forming applications. Jones and Papadopoulos [**106**] develop a heuristic method of imposing stick and slip conditions based on a relaxed inflation of friction cone and a control of slip reversal. Newton method is used as a solution strategy for some two-dimensional examples. An anisotropic friction model is developed in the following work by Jones and Papadopoulos [**107**]. Kane *et al.* [**111**] develop a formulation of the frictionless contact problem by applying tools of nonsmooth analysis [**43**]. Their non-penetration condition based on volumetric overlap of finite elements is similar to the one undertaken in the current work. The contact problem is formulated as a generalised, non-smooth minimisation (indicator functions are used) and the sequential quadratic programming scheme is employed as a solution strategy. Pandolfi *et al.* [**164**] extend this framework into the frictional case and, in the algorithmic sense, they maintain the variational (minimisation based) structure of the formulation. The Bipotential formulation summarised in Section 10.3.4 was initially introduced by De Saxe and Feng in [**56**].

CHAPTER 11

Solvers

The objective is to solve

$$\mathbf{C}(\mathbf{WR}+\mathbf{B},\mathbf{R})=\mathbf{0} \tag{11.0.1}$$

where all kinds of constraints are included. In the following, it will be at times convenient to write $\mathbf{C}(\mathbf{R})=\mathbf{0}$ instead of the above. Calculation of the constraint reactions allows to advance the time step and step up a consecutive system of constraint equations. Properties of $\mathbf{C}$ and several numerical techniques for solving (11.0.1) will be discussed in the following sections.

11.1. Properties of C

Operator $\mathbf{C}$ inherits its properties after both, the individual constraints and $\mathbf{W}$. In the presence of the frictional contact constraints it is unavoidably nonmonotone. This suggests the possibility of non-uniqueness of roots of $\mathbf{C}(\mathbf{R})=\mathbf{0}$. A process of looking for those can be further undermined by the lack of invertibility of $\mathbf{C}$. This remains in a direct relation with the invertibility of $\mathbf{W}$. Finally, non-smoothness of $\mathbf{C}$ plays a role whenever derivatives are to be computed. The nonmonotonicity has already been discussed at some length. The invertibility and non-smoothness require few additional comments.

11.1.1. Invertibility. In general, $\mathbf{C}(\mathbf{R})$ needs not to be invertible (more precisely *locally invertible*, that is, invertible for a sufficiently small neighbourhood of each $\mathbf{R}$). This is more of an issue for poorly deformable kinematics (like here), although for FEM discretised models the problem practically disappears. The discussion on the invertibility of $\mathbf{W}$ (Section 7.1) remains valid, nevertheless one needs to realise that $\mathbf{C}(\mathbf{R})\simeq\mathbf{WR}+\mathbf{B}$ only for some specific situations (e.g. when all contacts are sticky), so that invertibility of $\mathbf{W}$ does not directly translate into that of $\mathbf{C}$. In the further exposition we shall make a pragmatic assumption:

ASSUMPTION. *In the context of Newton methods presented in Section 11.2, it will be implicitly assumed that* $\mathbf{C}(\mathbf{R})$ *is locally invertible.*

Hence, the Jacobian $\partial\mathbf{C}(\mathbf{R})/\partial\mathbf{R}$ (or its generalisation) will be by the assumption non-singular. The linearisation based methods will be tested in combination with pseudo-rigid kinematics. This renders our simplification easier to achieve. Extension of the methods from Section 11.2 to the case of non-invertible $\mathbf{C}(\mathbf{R})$ needs to be registered as a matter of future research.

11.1.2. Semi-smoothness. It can be shown that $\mathbf{C}(\mathbf{R})$ is *Lipschitz continuous*, that is for all $\mathbf{R}_1$ and $\mathbf{R}_2$

$$\|\mathbf{C}(\mathbf{R}_1)-\mathbf{C}(\mathbf{R}_2)\|\leq K\|\mathbf{R}_1-\mathbf{R}_2\| \tag{11.1.1}$$

where K is a constant. One can use (10.3.41) and the triangle inequality in order to show that. K depends on the maximal friction coefficient μ, on the scaling

parameter ρ, and on the maximal eigenvalue of $\mathbf{W}$. As it was for example shown in Section 10.1.6, $\mathbf{C}(\mathbf{R})$ is not smooth. The source of non-smoothness is due to the max function employed in the projection formulae describing contact constraints. Because of this feature, $\mathbf{C}(\mathbf{R})$ is not differentiable in the usual sense for all $\mathbf{R}$. This means that the Fréchet derivative $D\mathbf{C}$, defined as

$$\lim_{\|\mathbf{h}\|\to 0} \frac{\|\mathbf{C}(\mathbf{R}+\mathbf{h}) - \mathbf{C}(\mathbf{R}) - D\mathbf{C}(\mathbf{R})\cdot\mathbf{h}\|}{\|\mathbf{h}\|} = 0 \tag{11.1.2}$$

does not exist for all $\mathbf{R}$. In our setting, $\max(x,y)$ is not differentiable when $x=y$. This corresponds to a surface $\mathcal{S}$ in $\mathbf{R}$-space, implicitly defined by $\mu d_{\alpha N} = \|\mathbf{d}_{\alpha T}\|$ and $d_{\alpha N} = 0$, cf. (10.3.33). Intuitively, across $\mathcal{S}$ the contact and friction states change (e.g. from stick to slip). Hence, one cannot describe $D\mathbf{C}$ by a single linear operator, when $\mathbf{R}\in\mathcal{S}$. Several generalisations of differentiability have been proposed in order to work around similar difficulties [**43, 39**]. For convex functions, the subdifferential defined in (10.3.10) is an example. $\mathbf{C}(\mathbf{R})$ however does not pertain to convexity, as it was shown to be nonmonotone. For Lipschitz functions, Clarke [**43**, p. 70] defines a generalised Jacobian

$$\partial\mathbf{C}(\mathbf{R}) = \mathrm{co}\left\{ \lim_{\substack{\mathbf{R}_i\to\mathbf{R}\\ \mathbf{R}_i\in D_{\mathbf{C}}}} D\mathbf{C}(\mathbf{R}_i) \right\} \tag{11.1.3}$$

as the convex hull of all limits of Fréchet derivatives, where $D_{\mathbf{C}}$ denotes the set of points where $\mathbf{C}$ is differentiable (see also [**39**]). Qi and Sun [**175**] use the notion of *semi-smoothness* in reference to (locally) Lipschitzian functions, for which the limit

$$\lim_{\substack{V\in\partial\mathbf{C}(\mathbf{R}+t\mathbf{g})\\ \mathbf{g}\to\mathbf{h},\, t\downarrow 0}} \{V\mathbf{g}\} \tag{11.1.4}$$

exists for any $\mathbf{h}$. The authors show, that for semismooth functions there holds

$$V\mathbf{h} - \mathbf{C}'(\mathbf{R};\mathbf{h}) = o(\|\mathbf{h}\|),\ \ V\in\partial\mathbf{C}(\mathbf{R}),\ \ \mathbf{h}\to\mathbf{0} \tag{11.1.5}$$

$$\mathbf{C}(\mathbf{R}+\mathbf{h}) - \mathbf{C}(\mathbf{R}) - \mathbf{C}'(\mathbf{R};\mathbf{h}) = o(\|\mathbf{h}\|),\ \ \mathbf{h}\to\mathbf{0} \tag{11.1.6}$$

where $\mathbf{C}'(\mathbf{R},\mathbf{h})$ is the directional derivative

$$\mathbf{C}'(\mathbf{R};\mathbf{h}) = \lim_{t\downarrow 0} \frac{\mathbf{C}(\mathbf{R}+t\mathbf{h}) - \mathbf{C}(\mathbf{R})}{t} \tag{11.1.7}$$

and $f(x) = o(g(x))$, when $\lim f(x)/g(x) = 0$ for $x\to 0$. Assuming invertibility of $V\in\partial\mathbf{C}(\mathbf{R})$ and uniform boundedness[1] of V^{-1} in the neighbourhood of $\mathbf{R}$, formulae (11.1.5) and (11.1.6) allow to show local super-linear convergence of the following *semi-smooth Newton scheme*

$$\mathbf{R}^{k+1} = \mathbf{R}^k - V_k^{-1}\mathbf{C}\left(\mathbf{R}^k\right),\ \ V_k\in\partial\mathbf{C}\left(\mathbf{R}^k\right) \tag{11.1.8}$$

[1] there exist a neighbourhood $N(\mathbf{R})$ and a constant C, such that $\|V^{-1}\| < C$ for all $V\in\partial\mathbf{C}(\mathbf{S})$, where $\mathbf{S}\in N(\mathbf{R})$

One might like to note, that in the above scheme some freedom is left as to the choice of V_k. In particular for $\mathbf{R}_k \in \mathcal{S}$, one can choose the semi-smooth tangent operator to be a limit of just one sequence (11.1.3), ranging through points of a smooth patch adjacent to $\mathcal{S}$ in the neighbourhood of $\mathbf{R}_k$. This freedom will be used in Section 11.2, when defining *active sets*. Now, taking $\mathbf{R}^*$ to be the solution of $\mathbf{C}(\mathbf{R}) = \mathbf{0}$, we can write

$$\begin{gathered}\left\|\mathbf{R}^{k+1} - \mathbf{R}^*\right\| = \left\|\mathbf{R}^k - \mathbf{R}^* - V_k^{-1}\mathbf{C}\left(\mathbf{R}^k\right)\right\| = \\ \left\|-V_k^{-1}\left[\mathbf{C}\left(\mathbf{R}^k\right) - \mathbf{C}\left(\mathbf{R}^*\right) - \mathbf{C}'\left(\mathbf{R}^*; \mathbf{R}^k - \mathbf{R}^*\right) + \mathbf{C}'\left(\mathbf{R}^*; \mathbf{R}^k - \mathbf{R}^*\right) - V_k\left(\mathbf{R}^k - \mathbf{R}^*\right)\right]\right\| \\ \leq \left\|V_k^{-1}\right\| \left\|\left[\mathbf{C}\left(\mathbf{R}^k\right) - \mathbf{C}\left(\mathbf{R}^*\right) - \mathbf{C}'\left(\mathbf{R}^*; \mathbf{R}^k - \mathbf{R}^*\right)\right]\right\| + \\ + \left\|V_k^{-1}\right\| \left\|\left[V_k\left(\mathbf{R}^k - \mathbf{R}^*\right) - \mathbf{C}'\left(\mathbf{R}^*; \mathbf{R}^k - \mathbf{R}^*\right)\right]\right\| = \\ = o\left(\left\|\mathbf{R}^k - \mathbf{R}^*\right\|\right)\end{gathered}$$

Hence, by picking a starting point $\mathbf{R}^0$ for sufficiently close to $\mathbf{R}^*$, a super-linear convergence can be achieved, as $\left\|\mathbf{R}^{k+1} - \mathbf{R}^*\right\| / \left\|\mathbf{R}^k - \mathbf{R}^*\right\|$ becomes arbitrarily small. In practise, it is the major difficulty to find an appropriate starting point. The Newton method presented Section 11.2, can be regarded as an instance of the semi-smooth technique sketched above. A formal proof would have to show, that $\mathbf{C}(\mathbf{R})$ is semi-smooth for $\mathbf{R} \in \mathcal{S}$. It is smooth for the remaining part of the domain. It seems clear, that similarly as for the augmented Lagrangian corresponding to a convex program, shown to be semismooth in [**175**], one can pursue such exercise in our case. On the other hand, the issue of a particular choice of the local convergence theory, remains in a sense the matter of taste. For example, for similar class of problems, Pang [**166**] applied the idea of *B-differentiability*. Although in that development, existence of Fréchet derivative was assumed at $\mathbf{R}^*$, it did not prevent a successful application of the method to frictional contact problems [**41**]. Also, a generalisation of the local convergence theory was proposed by Chen *et al.* [**39**], where the notion of *slant differentiability* was introduced. Among the useful features of this approach, there is applicability in the infinite dimensional context, as well as no need for the uniform boundedness of a linear operator generalising the Jacobian in the vicinity of a solution point.

11.2. Newton method

We present a linearisation of the frictional contact problem, as it is the most involving part of (11.0.1). Inclusion of other kinds of constraints corresponds merely to a simple extension of the linear systems presented in the following. This will be discussed at a later point. Operator (11.0.1) for the frictional contact problem can be rewritten as

$$\left\{\begin{array}{l}\mathbf{U} = \mathbf{B} + \mathbf{W}\mathbf{R} \\ \mathbf{C}(\mathbf{U}, \mathbf{R}) = \mathbf{0}\end{array}\right. \tag{11.2.1}$$

where

$$\mathbf{C}(\mathbf{U}, \mathbf{R}) = \left[\begin{array}{c}\ldots \\ \mathbf{C}_{\alpha T}(\mathbf{U}_\alpha, \mathbf{R}_\alpha) \\ C_{\alpha N}(\mathbf{U}_\alpha, \mathbf{R}_\alpha) \\ \ldots\end{array}\right] \tag{11.2.2}$$

$$\mathbf{C}_{\alpha T}(\mathbf{U}_\alpha, \mathbf{R}_\alpha) = \max\left(\mu d_{\alpha N}, \|\mathbf{d}_{\alpha T}\|\right)\mathbf{R}_{\alpha T} - \mu \max\left(0, d_{\alpha N}\right)\mathbf{d}_{\alpha T} \tag{11.2.3}$$

$$C_{\alpha N}\left(\mathbf{U}_\alpha, \mathbf{R}_\alpha\right) = R_{\alpha N} - \max\left(0, d_{\alpha N}\right) \tag{11.2.4}$$

$$\mathbf{d}_{\alpha T}\left(\mathbf{U}_\alpha, \mathbf{R}_\alpha\right) = \mathbf{R}_{\alpha T} - \rho \mathbf{U}_{\alpha T} \tag{11.2.5}$$

$$d_{\alpha N}\left(\mathbf{U}_\alpha, \mathbf{R}_\alpha\right) = R_{\alpha N} - \rho \bar{U}_{\alpha N} \tag{11.2.6}$$

Similar formulation is a starting point of the development by Hüeber *et al.* [**96**]. There however, the finite element mortar discretisation provides the first relation in (11.2.1). Contrary to the above, the formulation in [**96**] is stated in the standard primal form, with displacements acting on the global tangent operator. The current formulation is usually more suitable for kinematic models with a moderate amount of freedom, as $\mathbf{W}$ and $\mathbf{B}$ can be inexpensively computed.

In order to approximately solve (11.2.1), the active set strategy and the frictional Newton step proposed in [**96**] will be adopted. For the class of problems like the unilateral contact alone, the primal-dual active set technique was shown to be equivalent to the semismooth Newton method by Hintermüller *et al.* [**93**], so that the overall development can be regarded as a variant of the Newton algorithm.

11.2.1. Unilateral contact. The frictionless case is briefly examined. Finding normal reactions reduces to a well behaved problem, the structure of which was already discussed in Section 10.1.7. According to the reasoning presented therein, once the index sets of zero and nonzero reactions are identified, the solution can be obtained in one step. The two possible index sets will be denoted as active $\mathcal{A}_N$ and inactive $\mathcal{I}_N$. Although their immediate identification is usually not possible, the predictive formula (11.2.6) and the normal constraint (11.2.4) suggest the following approximation

$$\mathcal{A}_N = \{\alpha : d_{\alpha N} \geq 0\} \qquad \mathcal{I}_N = \{\alpha\} \setminus \mathcal{A}_N \tag{11.2.7}$$

The primal-dual active set algorithm solves a series of reduced linear systems for successive approximations of the above sets. This can be summarised as follows

ALGORITHM 11.2.1. *UNIL*

1 $k = 0$
2 $\mathbf{T}^k = \mathbf{W}\mathbf{R}^k + \mathbf{B} - \mathbf{U}^k$
3 $\mathcal{A}_N^k = \left\{\alpha : d_{\alpha N}^k \geq 0\right\} \quad \mathcal{I}_N^k = \{\alpha\} \setminus \mathcal{A}_N^k$
4 ***if*** $k > 0 \wedge \mathcal{A}_N^k = \mathcal{A}_N^{k-1}$ ***then stop***
5 $\mathcal{X} = \mathcal{A}_N^k N \quad \mathcal{Y} = \left\{\mathcal{A}_N^k T\right\} \cup \mathcal{I}_N^k$
6 $\begin{bmatrix} \mathbf{W}_{\mathcal{X}\mathcal{X}} & \mathbf{W}_{\mathcal{X}\mathcal{Y}} \\ \mathbf{0} & \mathbf{I}_{\mathcal{Y}} \end{bmatrix} \begin{bmatrix} \delta\mathbf{R}_{\mathcal{X}} \\ \delta\mathbf{R}_{\mathcal{Y}} \end{bmatrix}^k = \begin{bmatrix} -\bar{\mathbf{U}}_{\mathcal{X}} - \mathbf{T}_{\mathcal{X}} \\ -\mathbf{R}_{\mathcal{Y}} \end{bmatrix}^k$
7 $\mathbf{R}^{k+1} = \mathbf{R}^k + \delta\mathbf{R}^k$
8 $\mathbf{U}^{k+1} = \mathbf{U}^k + \mathbf{W}\delta\mathbf{R}^k + \mathbf{T}^k$
9 $k = k + 1$
10 ***goto*** *2*

For the sake of consistency with the forthcoming frictional linearisation, the incremental formulation is utilised above. An update of the residual $\mathbf{T}^k$ in line 2 is followed by the prediction of the active and inactive sets in line 3. From the complementarity considerations, it is seen that once the correct sets were predicted, they are not changed in line 3. Thus, the termination criterion takes quite specific form (line 4). In line 5 two index sets are created: $\mathcal{X}$ enumerating normal components in

the active set, and $\mathcal{Y}$ enumerating all of the tangential components together with the inactive normal ones. The linear system in line 6 follows from

$$\mathbf{U}^k + \delta\mathbf{U}^k = \mathbf{W}\left(\mathbf{R}^k + \delta\mathbf{R}^k\right) + \mathbf{B} \tag{11.2.8}$$

when considered with $\bar{U}^k_{\alpha N} + \delta U^k_{\alpha N} = 0$ for $\alpha \in \mathcal{A}_N$ and $R^k_{\alpha N} + \delta R^k_N = 0$ for $\alpha \in \mathcal{I}_N$, as well as $\mathbf{R}^k_{\alpha T} + \delta\mathbf{R}^k_{\alpha T} = \mathbf{0}$ for all α. The last four lines conclude the algorithm in an obvious way. The above recipe can be optimised by eliminating all tangential components.

11.2.2. Frictional tangents. As explained by Hüeber *et al.* [**96**], a numerically robust linearisation of the frictional constraint (11.2.3) requires some heuristic modifications. The authors examine a number of such modifications, one of which proves to be the most effective. Here, a brief derivation of the relevant formulae is provided.

The basic technical step relies on the differentiation of the max function, as the non-smoothness of the Euclidean norm in (11.2.3) will not play any role (the term vanishes for sticking points and is nonzero otherwise). The generalised partial derivative of the function $f(x,y) = \max(x,y)$ can be written as $Gf_x = 1$ if $x \geq y$ and $Gf_x = 0$ if $x < y$. Gf_y is calculated analogously. As adopted in [**175**], at any point the generalised derivative belongs to the set-valued gradient defined by Clarke [**43**]. As a consequence, the partial derivatives at $x = y$ can be equal to any number in the range $[0,1]$. Thus when comparing x and y, the equality can be adopted on either side. From the algorithmic point of view this corresponds to a nuance in the definition of the active and inactive sets, utility of which will be commented on at a later point (Section 11.2.3). The active and inactive tangential sets are defined as follows

$$\mathcal{A}_T = \{\alpha \in \mathcal{A}_N : \|\mathbf{d}_{\alpha T}\| - \mu d_{\alpha N} \geq 0\} \qquad \mathcal{I}_T = \mathcal{A}_N \setminus \mathcal{A}_T \tag{11.2.9}$$

Let us focus on a contact point with index α, and temporarily neglect the α-indexing. The characteristic function $\chi_{\mathcal{S}} = 1$ if $\alpha \in \mathcal{S}$ and $\chi_{\mathcal{S}} = 0$ otherwise. According to the above definitions the differential of the tangential constraint reads

$$\begin{array}{c} G_{\mathbf{C}_T}(\delta\mathbf{R}, \delta\mathbf{U}) = \chi_{\mathcal{A}_T} \frac{\mathbf{d}_T(\delta\mathbf{R}_T - \rho\delta\mathbf{U}_T)}{\|\mathbf{d}_T\|}\mathbf{R}_T \\ +\chi_{\mathcal{I}_T}\mu\left(\delta R_N - \rho\delta U_N\right)\mathbf{R}_T + \max\left(\mu d_N, \|\mathbf{d}_T\|\right)\delta\mathbf{R}_T \\ -\chi_{\mathcal{A}_N}\mu\left(\delta R_N - \rho\delta U_N\right)\mathbf{d}_T - \mu\max\left(0, d_N\right)\left(\delta\mathbf{R}_T - \rho\delta\mathbf{U}_T\right) \end{array} \tag{11.2.10}$$

and the tangential Newton step takes the form

$$G_{\mathbf{C}_T}\left(\delta\mathbf{R}^k, \delta\mathbf{U}^k\right) = -\mathbf{C}_T\left(\mathbf{R}^k, \mathbf{U}^k\right) \tag{11.2.11}$$

$$\left(\mathbf{R}^{k+1}, \mathbf{U}^{k+1}\right) = \left(\mathbf{R}^k, \mathbf{U}^k\right) + \left(\delta\mathbf{R}^k, \delta\mathbf{U}^k\right) \tag{11.2.12}$$

In case of frictional sticking ($\|\mathbf{d}_T\| < \mu d_N$), equation (11.2.11) simplifies to

$$\delta\mathbf{U}^k_T = -\frac{\mathbf{U}^k_T}{d^k_N}\delta R_N - \frac{\mathbf{U}^k_T}{d^k_N}\rho\bar{U}^k_N - \mathbf{U}^k_T \tag{11.2.13}$$

Condition $\bar{U}^k_N + \delta U^k_N = 0$ was utilised to derive the above (frictional linearisation is considered on the active normal set). Using (11.2.12), formula (11.2.13) can be rewritten as $\mathbf{U}^{k+1}_T = \left(1 - R^{k+1}_N/d^k_N\right)\mathbf{U}^k_T$, where it is seen that for a convergent

sequence of iterates $\mathbf{U}_T^{k+1} \to \mathbf{0}$, as $\left|R_N^{k+1} - d_N^k\right| \to 0$. In the remaining case of frictional slipping ($\|\mathbf{d}_T\| \geq \mu d_N$), equation (11.2.11) takes the following form

$$\mathbf{R}_T^k + \left(\mathbf{I} - \mathbf{M}^k\right)\delta\mathbf{R}_T^k + \rho\mathbf{M}^k\delta\mathbf{U}_T^k = \mathbf{v}_T^k\mu\left(R_N^k + \delta R_N\right) \tag{11.2.14}$$

where $\mathbf{I}$ stands for the two-dimensional identity matrix, and

$$\mathbf{M}^k = e^k\left(\mathbf{I} - \mathbf{F}^k\right) \tag{11.2.15}$$

$$\mathbf{F}^k = \frac{\mathbf{R}_T^k \otimes \mathbf{d}_T^k}{\mu d_N^k \left\|\mathbf{d}_T^k\right\|} \tag{11.2.16}$$

$$e^k = \frac{\mu d_N^k}{\left\|\mathbf{d}_T^k\right\|} \tag{11.2.17}$$

$$\mathbf{v}_T^k = \frac{\mathbf{d}_T^k}{\left\|\mathbf{d}_T^k\right\|} \tag{11.2.18}$$

Equation (11.2.14) expresses a ray-wise Coulomb constraint along the predictor direction $\mathbf{v}_T^k$. Evidently, variations of the tangential reaction and velocity together contribute to the fulfilment of the linearised constraint. Thus, the iterates of the reaction $\mathbf{R}_T^{k+1}$ do not necessarily belong to the friction cone before the convergence tightens. The following modification

$$\tilde{\mathbf{F}}^k = \frac{\mathbf{R}_T^k \otimes \mathbf{d}_T^k}{\max\left(\mu d_N^k, \left\|\mathbf{R}_T^k\right\|\right)\left\|\mathbf{d}_T^k\right\|} \tag{11.2.19}$$

results in an approximate projection of $\mathbf{R}^{k+1}$ onto the tangent to the current section of the friction cone [**96**]. This results from the fact, that whenever $\mathbf{R}_T^k$ and $\mathbf{d}_T^k$ are nearly aligned, together with $\left\|\mathbf{R}_T^k\right\| \geq \mu d_N^k$, the matrix $\mathbf{I} - \tilde{\mathbf{F}}^k$ acts roughly as a projection on the direction perpendicular to $\mathbf{v}_T^k$. Therefore, the modified $\mathbf{M}^k$ filters out components parallel to $\mathbf{v}_T^k$. One can see, that when (11.2.19) is in power, $\mathbf{R}_T^k + \delta\mathbf{R}_T^k$ will approximately lay on the line perpendicular to $\mathbf{v}_T^k$ and tangent to the μR_N^{k+1} section of the friction cone. This can be best observed in Figure 11.2.1. In practice then, the coefficients in relation (11.2.14) are computed with (11.2.16) replaced by (11.2.19). The modification results in faster and more robust convergence behaviour. This is because the iterates of $\mathbf{R}_T^{k+1}$ remain closer to the friction cone, thus less significantly interact through the kinematic coupling in $\mathbf{W}$. This seems particularly helpful in the formulation admitting large rotations and therefore stronger normal-tangential coupling.

It is appropriate to mention another modification investigated in [**96**]. The authors regularise the operator $\mathbf{I} - \mathbf{M}^k$ in (11.2.14), so that it is always invertible and positive definite. This is not pursued here, as it proved not to be consistently beneficial in the numerical realisation. This might be due to the different way of eliminating variables in the current development.

11.2.3. Complete algorithm. The normal active set strategy from Section 11.2.1 can now be combined with the tangential linearisation, in order to deliver a complete Newton scheme for the frictional contact problem. As it was mentioned in Section 10.3.5, the nonmonotone character of the adopted contact law results in the need to globalise the Newton scheme. This is provided by means of the nonmonotone line search technique by Grippo *et al.* [**78**]. The choice seems to be more relevant to the nature of problem at hand. Nevertheless, the simple Armijo's

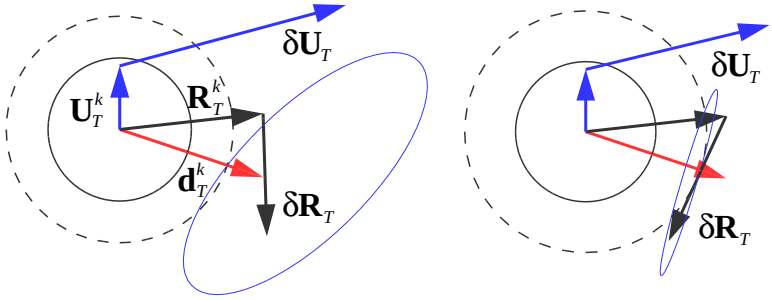


FIGURE 11.2.1. The Effect of the modification (11.2.19). The circle radii are μR_N^k (solid), and μR_N^{k+1} (dashed). On the left, the unmodified iteration of the Newton step (11.2.11), (11.2.12) is presented. The ellipsoid corresponds to the points of $\mathbf{R}_T^k + \left(\mathbf{I} - \mathbf{M}^k\right) \delta\mathbf{R}_T^k + \rho\mathbf{M}^k\mathbf{S}$, where $\|\mathbf{S}\| = \left\|\delta\mathbf{U}_T^k\right\|$. On the right, the same iteration with (11.2.19) enabled results in the narrowed ellipsoid, with one of its eigenvectors nearly orthogonal to $\mathbf{d}_T^k$. Thus, $\mathbf{R}_T^{i+1}$ is approximately placed on the tangent to the current section of the friction cone.

[**11**] line search will also be investigated. This type of monotone globalisation was applied by Christensen *et al.* [**41**] in the context of two dimensional linearly elastic problems.

The sequence of iterates of contact reactions is generated according to

$$\mathbf{R}^{k+1} = \mathbf{R}^k + \alpha^k \delta\mathbf{R}^k \tag{11.2.20}$$

where $\delta\mathbf{R}^k$ is the search direction, and $\alpha^k \in (0, 1]$ is the step size. The search direction results from the semismooth Newton step applied to the system (11.2.1). Three ways of calculating $\delta\mathbf{R}^k$ will be discussed. The first one results from the consistent linearisation of (11.2.1). The normal active set strategy and the tangential linearisation are combined as follows

ALGORITHM 11.2.2. *NEWT*

1 $\mathbf{T}^k = \mathbf{W}\mathbf{R}^k + \mathbf{B} - \mathbf{U}^k$

2 $\mathcal{A}_N^k = \left\{\alpha : d_{\alpha N}^k \geq 0\right\}$ $\mathcal{I}_N^k = \{\alpha\} \setminus \mathcal{A}_N^k$

3 $\mathcal{A}_T^k = \left\{\alpha : \left\|\mathbf{d}_{\alpha T}^k\right\| - \mu d_{\alpha N}^k \geq 0 \wedge \alpha \in \mathcal{A}_N^k\right\}$ $\mathcal{I}_T^k = \mathcal{A}_N^k \setminus \mathcal{A}_T^k$

4 ***solve*** $\Omega^k \delta\mathbf{R}^k = \Pi^k$ ***where***

5 ***for*** $\alpha \in \mathcal{I}_N^k$
$\Omega_{\alpha\alpha}^k = \mathbf{I}$ $\Omega_{\alpha\beta}^k = \mathbf{0}$ $\Pi^k = -\mathbf{R}^k$

6 ***for*** $\alpha \in \mathcal{A}_N^k$
$\Omega_{\alpha\beta N*}^k = \mathbf{W}_{\alpha\beta N*}$ $\Pi_{\alpha N}^k = -\bar{U}_{\alpha N}^k - T_{\alpha N}^k$

7 ***for*** $\alpha \in \mathcal{I}_T^k$
$\Omega_{\alpha\alpha T*}^k = \left[\begin{array}{cc} \mathbf{W}_{\alpha\alpha TT} & \mathbf{W}_{\alpha\alpha NT} + \mathbf{U}_{\alpha T}^k / d_{\alpha N}^k \end{array}\right]$
$\Omega_{\alpha\beta T*}^k = \mathbf{W}_{\alpha\beta T*}$ $\Pi_{\alpha T}^k = -\left(1 + \rho\bar{U}_{\alpha N}^k / d_{\alpha N}^k\right) \mathbf{U}_{\alpha T}^k - \mathbf{T}_{\alpha T}^k$

8 ***for*** $\alpha \in \mathcal{A}_T^k \wedge d_{\alpha N} = 0$
$\Omega_{\alpha\alpha T*}^k = \left[\begin{array}{cc} \mathbf{I} & \mathbf{0} \end{array}\right]$ $\Omega_{\alpha\beta T*}^k = \mathbf{0}$ $\Pi_{\alpha T}^k = -\mathbf{R}_{\alpha T}^k$

9 ***for*** $\alpha \in \mathcal{A}_T^k \wedge d_{\alpha N} > 0$
$\Omega_{\alpha\alpha T*}^k = \left[\begin{array}{cc} \left(\mathbf{I} - \mathbf{M}_\alpha^k\right) + \rho\mathbf{M}_\alpha^k\mathbf{W}_{\alpha\alpha TT} & \rho\mathbf{M}_\alpha^k\mathbf{W}_{\alpha\alpha NT} - \mu\mathbf{v}_{\alpha T}^k \end{array}\right]$
$\Omega_{\alpha\beta T*}^k = \rho\mathbf{M}_\alpha^k\mathbf{W}_{\alpha\beta T*}$ $\Pi_{\alpha T}^k = \mu\mathbf{v}_{\alpha T}^k R_{\alpha N}^k - \mathbf{R}_{\alpha T}^k - \rho\mathbf{M}_\alpha^k\mathbf{T}_{\alpha T}^k$

The residual update in the first line above is followed by the selection of active and inactive sets (lines 2, 3). The tangential sets are subsets of the normal active one, $\mathcal{A}_N^k = \mathcal{A}_T^k \cap \mathcal{I}_T^k$. The increment $\delta\mathbf{R}^k$ results from the solution of a linear system in line 4. The system matrix Ω is composed of dense 3×3 blocks $\Omega_{\alpha\beta}$, and has the same block-sparsity pattern as $\mathbf{W}$. This is of use in the numerical realisation, as the symbolic factorisation of Ω can be computed only once. The structure of rows in line 5 results from the fact that $\mathbf{R}_\alpha^{k+1} = \mathbf{0}$ is assumed on $\mathcal{I}_N^k$ for the full Newton scheme ($\alpha^k = 1$). Thus $\delta\mathbf{R}_\alpha^k = -\mathbf{R}_\alpha^k$ on the inactive normal set. The normal row structure in line 6[2] results from the reasoning already presented in Section 11.2.1. In short, it corresponds to the rows of

$$\mathbf{U}^k + \delta\mathbf{U}^k = \mathbf{W}\left(\mathbf{R}^k + \delta\mathbf{R}^k\right) + \mathbf{B} \tag{11.2.21}$$

written with the account of $\bar{U}_{\alpha N}^k + \delta U_{\alpha N} = 0$ for $\alpha \in \mathcal{A}_N$. In line 7, the coupling (11.2.13) is utilised in order to eliminate the velocity increment $\delta\mathbf{U}_{\alpha T}^k$ from the tangential rows of (11.2.21), with $\alpha \in \mathcal{I}_T^k$. As the active sets were defined with the equality inclusion $\geq$, one needs to deal with the active tangential case, corresponding to the zero friction bound (line 8). This is a pragmatic choice motivated by a faster communication during the solution process. In particular, considering a structure composed of contacting bodies with a force applied to only one of them, the above algorithm will assume the frictionless contact for all bodies not directly adjacent to the one with the nonzero force. The next iteration will then start from some nonzero reactions guess for all bodies connected in the contact graph (the nonzero block pattern graph of $\mathbf{W}$). If the sharp inequality $>$ was utilised to define the active sets, the nonzero contact forcing would have to gradually propagate according to the immediate adjacency in the contact graph. Coming back to the line 8, it is seen that the zero tangential response is enforced for $\alpha \in \mathcal{A}_T^k$ and $d_{\alpha N} = 0$. The remaining non-degenerate tangential case ($d_{\alpha N} > 0$) is considered in line 9. Here, the tangential rows of Ω are obtained by substituting $\delta\mathbf{U}_T^k$, calculated from (11.2.21) into the tangent relation (11.2.14). This way of eliminating variables is motivated by the intention of preserving the impact of modification (11.2.19).

In fact, the descent directions provided by the algorithm `NEWT` are not among the most effective, when calculated far from the solution. As a result of numerical experiments aimed at improvement of the global convergence properties, the following hybrid scheme has arisen

ALGORITHM 11.2.3. *HYB*

1 $\mathbf{T}^k = \mathbf{W}\mathbf{R}^k + \mathbf{B} - \mathbf{U}^k$

2 $\mathcal{A}_N^k = \{\alpha : d_{\alpha N}^k \geq 0\}$ $\mathcal{I}_N^k = \{\alpha\} \setminus \mathcal{A}_N^k$

3 $\mathcal{A}_T^k = \{\alpha : \|\mathbf{d}_{\alpha T}^k\| - \mu R_{\alpha N}^k \geq 0\}$ $\mathcal{I}_T^k = \{\alpha\} \setminus \mathcal{A}_T^k$

4 ***solve*** $\Omega^k \delta\mathbf{R}^k = \Pi^k$ ***where***

5 ***for*** $\alpha \in \mathcal{I}_N^k$

$\Omega_{\alpha\alpha}^k = \mathbf{I}$ $\Omega_{\alpha\beta}^k = \mathbf{0}$ $\Pi^k = -\mathbf{R}^k$

6 ***for*** $\alpha \in \mathcal{A}_N^k$

$\Omega_{\alpha\beta N*}^k = \mathbf{W}_{\alpha\beta N*}$ $\Pi_{\alpha N}^k = -\bar{U}_{\alpha N}^k - T_{\alpha N}^k$

7 ***for*** $\alpha \in \mathcal{I}_T^k$

$\Omega_{\alpha\beta T*}^k = \mathbf{W}_{\alpha\beta T*}$ $\Pi_{\alpha T}^k = -\mathbf{U}_{\alpha T}^k - \mathbf{T}_{\alpha T}^k$

8 ***for*** $\alpha \in \mathcal{A}_T^k \wedge d_{\alpha N} = 0$

$\Omega_{\alpha\alpha T*}^k = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}$ $\Omega_{\alpha\beta T*}^k = \mathbf{0}$ $\Pi_{\alpha T}^k = -\mathbf{R}_{\alpha T}^k$

9 ***for*** $\alpha \in \mathcal{A}_T^k \wedge d_{\alpha N} > 0$

[2] In the algorithms presented in this section the asterisk subscript "$*$" replaces "all relevant indices", e.g. $\Omega_{N*} = \begin{bmatrix} \Omega_{NT} & \Omega_{NN} \end{bmatrix}$.

$$\Omega^k_{\alpha\alpha T*} = \left[\ \left(\mathbf{I} - \mathbf{M}^k_\alpha\right) + \rho\mathbf{M}^k_\alpha\mathbf{W}_{\alpha\alpha TT} \quad \rho\mathbf{M}^k_\alpha\mathbf{W}_{\alpha\alpha NT} - \mu\mathbf{v}^k_{\alpha T}\ \right]$$
$$\Omega^k_{\alpha\beta T*} = \rho\mathbf{M}^k_\alpha\mathbf{W}_{\alpha\beta T*} \quad \Pi^k_{\alpha T} = \mu\mathbf{v}^k_{\alpha T}R^k_{\alpha N} - \mathbf{R}^k_{\alpha T} - \rho\mathbf{M}^k_\alpha\mathbf{T}^k_{\alpha T}$$

Contrary to the previous case, the normal and tangential sets are independently set in lines 2 and 3. Also, the bound employed in line 3 is not any more based on the predictor $d^k_{\alpha N}$, but owes to the currently computed normal reaction $R^k_{\alpha N}$. The last difference with regard to the algorithm NEWT is the assumption of $\mathbf{U}^{k+1}_{\alpha T} = \mathbf{0}$ for $\alpha \in \mathcal{I}^k_T$, expressed in line 7. This means, that the strict linearisation of the inactive tangential case is not pursued. The above scheme can be linked to the fixed point technique, presented in the reference work [**96**]. The major inconsistency is in using the full Newton linearisation for the slip contact, in line 9. In the fixed point approach, the tangential slip relations are linearised according to the Tresca friction model. The linearisation can be obtained from (11.2.14), by discarding the term involving δR^k_N and employing a fixed normal bound b_N instead of d^k_N

$$\mathbf{R}^k_T + \left(\mathbf{I} - \mathbf{M}^k\right)\delta\mathbf{R}^k_T + \rho\mathbf{M}^k\delta\mathbf{U}^k_T = \mathbf{v}^k_T\mu b_N \tag{11.2.22}$$

where in the intermediate formulae (11.2.16), (11.2.17), and (11.2.19) the normal predictor d^k_N is replaced by the fixed bound b_N. The fixed point approach is summarised below

ALGORITHM 11.2.4. *FIX*

1 $\mathbf{T}^k = \mathbf{W}\mathbf{R}^k + \mathbf{B} - \mathbf{U}^k$
2 $\mathcal{A}^k_N = \left\{\alpha : d^k_{\alpha N} \geq 0\right\}\ \ \mathcal{I}^k_N = \{\alpha\} \setminus \mathcal{A}^k_N$
3 $\mathcal{A}^k_T = \left\{\alpha : \left\|\mathbf{d}^k_{\alpha T}\right\| - \mu b_{\alpha N} \geq 0\right\}\ \ \mathcal{I}^k_T = \{\alpha\} \setminus \mathcal{A}^k_T$
4 ***solve*** $\Omega^k\delta\mathbf{R}^k = \Pi^k$ ***where***
5 ***for*** $\alpha \in \mathcal{I}^k_N$
$\Omega^k_{\alpha\alpha} = \mathbf{I}\ \ \Omega^k_{\alpha\beta} = \mathbf{0}\ \ \Pi^k = -\mathbf{R}^k$
6 ***for*** $\alpha \in \mathcal{A}^k_N$
$\Omega^k_{\alpha\beta N*} = \mathbf{W}_{\alpha\beta N*}\ \ \Pi^k_{\alpha N} = -\bar{U}^k_{\alpha N} - T^k_{\alpha N}$
7 ***for*** $\alpha \in \mathcal{I}^k_T$
$\Omega^k_{\alpha\beta T*} = \mathbf{W}_{\alpha\beta T*}\ \ \Pi^k_{\alpha T} = -\mathbf{U}^k_{\alpha T} - \mathbf{T}^k_{\alpha T}$
8 ***for*** $\alpha \in \mathcal{A}^k_T \wedge d_{\alpha N} = 0$
$\Omega^k_{\alpha\alpha T*} = \left[\ \mathbf{I} \quad \mathbf{0}\ \right]\ \ \Omega^k_{\alpha\beta T*} = \mathbf{0}\ \ \Pi^k_{\alpha T} = -\mathbf{R}^k_{\alpha T}$
9 ***for*** $\alpha \in \mathcal{A}^k_T \wedge d_{\alpha N} > 0$
$\Omega^k_{\alpha\alpha T*} = \left[\ \left(\mathbf{I} - \mathbf{M}^k_\alpha\right) + \rho\mathbf{M}^k_\alpha\mathbf{W}_{\alpha\alpha TT} \quad \rho\mathbf{M}^k_\alpha\mathbf{W}_{\alpha\alpha NT}\ \right]$
$\Omega^k_{\alpha\beta T*} = \rho\mathbf{M}^k_\alpha\mathbf{W}_{\alpha\beta T*}\ \ \Pi^k_{\alpha T} = \mu\mathbf{v}^k_{\alpha T}b_{\alpha N} - \mathbf{R}^k_{\alpha T} - \rho\mathbf{M}^k_\alpha\mathbf{T}^k_{\alpha T}$

It is seen that the third line of the algorithm HYB is a special case of the corresponding line of FIX, with $b_{\alpha N} = R^k_{\alpha N}$. This corresponds to an update of the friction bound in every iteration of the fixed point scheme. In practice, at least in case of the current kinematic formulation, that frequent update of the friction bound prevents convergence of the fixed point approach. This happens because the tangential-normal coupling in line 9 results only from the problem kinematics. Nevertheless, the Tresca regularisation usually results in a good global convergence behaviour, provided the updates of $b_{\alpha N}$ are sparse enough. The hybrid approach provides then an intermediate case between the full Newton and the fixed point methods. The poor convergence of the fixed point approach with the most frequent update of normal bounds is remedied by the full linearisation of the tangential slip. This heuristic attempt of synergy between the Newton and fixed point approaches is to be numerically investigated in the next section.

One thing to be noted about all three algorithms is that they result in an unsymmetric systems to be solved for $\delta\mathbf{R}^k$. This can be to some extent remedied

by multiplying the two tangential rows given in lines 9 by the operator $\left[\rho\mathbf{M}_{\alpha}^{k}\right]^{-1}$. This introduces more symmetry into Ω and seems to be particularly advantageous for the fixed point scheme, where the system matrix becomes gradually symmetric with progressing convergence. On the other hand, the modification (11.2.19) does not act any more on the entire row of Ω. The experience shows that (at least within the current formulation) this way of eliminating variables is not effective. Thus, it will not be further investigated.

The above three schemes need to be embraced by some global criteria of advancing the iterations. In case of `NEWT` and `HYB` this will be provided by the mentioned line search technique. The fixed point scheme, being generally better behaved in terms of the global convergence, will only be wrapped into a suitable external loop updating the normal bounds $b_{\alpha N}$. All together, this can be stated as follows

ALGORITHM 11.2.5. *SCSOL* $(\mathbf{ALG}, \sigma, \gamma, \beta, J, \epsilon, K, \phi, L)$

1 $k = 0$
2 ***do***
3 $\quad \delta\mathbf{R}^k = \mathbf{ALG}\,()$
4 $\quad \alpha^k = 1$
5 $\quad$ ***while*** $\mathcal{M}\left(\mathbf{R}^k + \alpha^k\delta\mathbf{R}^k\right) > \max\limits_{0\le j\le\min(k,J)} \mathcal{M}^{k-j} - 2\gamma\alpha^k\mathcal{M}^k$
$\qquad \wedge\, k > 0 \wedge \alpha^k > \beta \wedge \mathbf{ALG} \neq \boldsymbol{FIX}$ ***do*** $\alpha^k = \sigma\alpha^k$
6 $\quad \mathbf{R}^{k+1} = \mathbf{R}^k + \alpha^k\delta\mathbf{R}^k$
7 $\quad \mathbf{U}^{k+1} = \mathbf{U}^k + \alpha^k\mathbf{W}\delta\mathbf{R}^k + \mathbf{T}^k$
8 $\quad err = \left\|\delta\mathbf{R}^k\right\| / \left\|\mathbf{R}^{k+1}\right\|$
9 $\quad k = k + 1$
10 $\quad$ ***if*** $\mathbf{ALG} = \boldsymbol{FIX} \wedge err \le \epsilon$ ***then***
11 $\quad\quad err = \left\|\mathbf{R}_N^k - \mathbf{b}_N\right\| / \left\|\mathbf{R}_N^k\right\|$
12 $\quad\quad \mathbf{b}_N = \mathbf{R}_N^k$
13 $\quad$ ***end if***
14 $\quad$ ***if*** $\mathbf{ALG} \neq \boldsymbol{FIX} \wedge k > L$ ***then*** $SCALE\,(\{\rho_\alpha\}, \phi, L)$
15 ***while*** $err > \epsilon \wedge k \le K$

The argument `ALG` can be `NEWT`, `HYB` or `FIX`. The next three arguments $\sigma, \gamma, \beta \in (0, 1)$ correspond to the line search step. The ϵ describes numerical accuracy, K bounds the maximal number of iterations, and J is the length of memory buffer used by the line search. The remaining arguments $\phi, L \ge 1$ are used for the purpose of the penalty scaling and will be commented on later in this section. In the third line above, the current increment of reactions is computed by `ALG`. The initial scaling parameter α^k is set equal to 1 in the following line. The loop in line 5 corresponds to the nonmonotone line search [**78**]. Note, that for $J = 0$ it is equivalent to the line search of Armijo's type [**11**]. Both approaches were originally designed for smooth problems. The analysis suitable for the nonsmooth setting was provided by Ferris and Lucidi [**69**]. The auxiliary merit function is defined as

$$\mathcal{M}\left(\mathbf{R}\right) = \frac{1}{2}\mathbf{C}^T\left(\mathbf{R}, \mathbf{U}\right)\mathbf{C}\left(\mathbf{R}, \mathbf{U}\right)\Big|_{\mathbf{U}=\mathbf{W}\mathbf{R}+\mathbf{B}} \tag{11.2.23}$$

where (11.2.1) was utilised. The symbol $\mathcal{M}^k$ refers to $\mathcal{M}\left(\mathbf{R}^k\right)$. If the minimisation in line 5 is successful (iterations end before $\alpha^k \le \beta$), it is seen that a monotonic decrease of the merit function is enforced for $J = 0$, while this is not necessarily the case for $J > 0$. The acceptability criterion

$$\mathcal{M}\left(\mathbf{R}^k + \alpha^k\delta\mathbf{R}^k\right) \le \max_{0\le j\le\min(k,J)} \mathcal{M}^{k-j} - 2\gamma\alpha^k\mathcal{M}^k \tag{11.2.24}$$

allows for the temporary growth of the merit function if only $J > 0$. At the same time the solution point remains inside of the nested level sets $\mathbf{R}^k \in \Lambda^k \subseteq \Lambda^{k-1}$

$$\Lambda^k = \left\{ \mathbf{R} : \mathcal{M}(\mathbf{R}) \leq \max_{0 \leq j \leq \min(k,J)} \mathcal{M}\left(\mathbf{R}^{k-j}\right) \right\} \tag{11.2.25}$$

The parameter J is then proportional to the allowed extent of the temporary growth of the merit function. Grippo *et al.* [**78**] prove that this relaxation does not hinder the global convergence, if only some conditions hold (roughly, the merit function must be bounded below, and $\delta\mathbf{R}$ must be a descend direction). At the same time, for the nonmonotone problems this may lead to a faster convergence, as a convergent sequence of iterates does not have to correspond to a monotonically decreasing sequence of function values. The threshold value β is used due to the finite precision of numerical computations (the line search loop exits after a finite number of steps). For this reason the line search cannot be fully robust in practice. In the above algorithm, the line search technique is applied for $k > 0$, which results in $\alpha^0 = 1$. This is a heuristic dictated by an observation, that usually it is more effective to start globalisation from the iterate obtained by the pure Newton step corresponding to an initial residual. In other words, it often happens that the subsequent alphas are "large", while if the line search was performed for $k = 0$, initial alphas often happen to be "small". Finally, it is seen that the line search is omitted for the fixed point scheme. The update of reactions and velocities follows in lines 6, 7. Line 7 corresponds to the Newton step

$$\delta\mathbf{U}^k = \mathbf{W}\delta\mathbf{R}^k + \mathbf{T}^k \tag{11.2.26}$$

$$\mathbf{T}^k = \mathbf{W}\mathbf{R}^k + \mathbf{B} - \mathbf{U}^k \tag{11.2.27}$$

thus the residual $\mathbf{T}^{k+1}$ is always zero

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \alpha^k \mathbf{W}\delta\mathbf{R}^k + \mathbf{W}\mathbf{R}^k + \mathbf{B} - \mathbf{U}^k = \mathbf{W}\mathbf{R}^{k+1} + \mathbf{B} \tag{11.2.28}$$

It is possible to modify `UNIL`, `NEWT`, `HYB` and `FIX` by replacing the computation of $\mathbf{T}^k$ by the update of velocity $\mathbf{U}^k = \mathbf{W}\mathbf{R}^k + \mathbf{B}$. This, combined with the removal of line 7 from `SCSOL` still provides a feasible framework. Nevertheless, experience shows that processing the residual is advantageous and results in smaller numbers of iterations. From the lines 8-13 it is seen that in case of the fixed point method, after each convergent run with a fixed normal bound, the bound is updated and the relative error of this update replaces the error controlling the termination of the algorithm. Note also, that scaling is not applied in case of the fixed point scheme (line 14), as we are interested in testing the plainest possible version of this approach.

Algorithms like the one above, where the tangent operator results from a non-smooth, and to some extent combinatorial structure, are prone to cycling. By this it is meant that for some parameter sets, the algorithm may get caught into a cycle (here corresponding to a sequence of contact states) preventing further convergence. In the context of frictional contact problems, this was mentioned by Alart and Curnier [**7**], or DeSaxcé and Feng [**55**]. In general, cycling is more frequent for stiff problems and its occurrence is related to the regularisation parameter ρ, used in predictive formulae (11.2.5) and (11.2.6). This issue is more thoroughly commented in [**7**]. In this work, initial values of ρ_α are independently set for each contact $\rho_\alpha = 1/\lambda_{max}(\mathbf{W}_{\alpha\alpha})$, where λ_{max} is the maximal eigenvalue of the diagonal block matrix $\mathbf{W}_{\alpha\alpha}$. This corresponds to the monotonicity criteria of the diagonal sub-problems of the simplified problem (10.3.36). In the course of solution, each

change of the tangential contact state (from stick to slip or vice versa) is further penalised by increasing ρ_α. As the maximal value of ρ_α ought to be bounded by $2/\lambda_{max}(\mathbf{W})$ [7], only a finite number (bounded by L) of such increases is performed, although an explicit estimate of $\lambda_{max}(\mathbf{W})$ is not accounted for. In order to avoid excessive number of heuristic parameters, L provides also the lower iterations bound after which the scaling is applied. This strategy seems to be sufficient in practice. The `SCALE` routine is summarised below (initial l_α are assumed equal zero)

ALGORITHM 11.2.6. *SCALE* $(\{\rho_\alpha\}, \phi, L)$

```
1  for each α do
2    if l_α < L ∧ (stick → slip ∨ slip → stick)_α then
3      ρ_α = φρ_α
4      l_α = l_α + 1
5    end if
6  end for
```

It is relevant to ask why does not the line search procedure suffice to avoid cycling. In Section 10.3.5 it was shown that the single contact point problem behaves in a nonmonotone way. Using the unconstrained minimisation analogy, one could say that the merit function corresponding to this simplest case possesses a region of concavity. For problems with many contact points the corresponding merit functions possess regions of concavity not only juxtaposed from the single point problems, but also created through their kinematic interactions. In other words the problem becomes highly nonlinear. Theoretically, enforcing a monotone decrease of the merit function ($J = 0$) should guarantee a descent towards the local minimum. In practise though, the line search loop is forced to end after a relatively small number of steps (one is not interested in updating the solution with α^k close to the numerical zero). After an "unfinished" search the solution point may jump to a neighbouring hill. This process may continue in a cyclic manner. The nonmonotone line search ($J > 0$) only increases the probability of such scenario (nevertheless, it is potentially beneficial otherwise).

11.2.4. Inclusion of joints. When joints are present, they correspond to additional rows in the system $\Omega^k \delta\mathbf{R}^k = \Pi^k$. Joints are expressed as linear constraints on selected components of local velocity. For example a constraint

$$aU_{\alpha N} + b = 0 \tag{11.2.29}$$

through linearisation

$$a\left(U_{\alpha N} + \delta U_{\alpha N}\right) + b = 0 \tag{11.2.30}$$

and

$$\left(U_{\alpha N} + \delta U_{\alpha N}\right) = \mathbf{W}_{\alpha N *}\left(\mathbf{R} + \delta\mathbf{R}\right) + \mathbf{B}_{\alpha N} \tag{11.2.31}$$

results in a row

$$\mathbf{W}_{\alpha N *}\delta\mathbf{R}^k + = -\frac{b}{a} - \mathbf{B}_{\alpha N} - \mathbf{W}_{\alpha *}\mathbf{R}^k \tag{11.2.32}$$

so that

$$\mathbf{\Omega}_{\alpha *} = \mathbf{W}_{\alpha N *} \tag{11.2.33}$$

$$\mathbf{\Pi}_\alpha = -\frac{b}{a} - \mathbf{B}_{\alpha N} - \mathbf{W}_{\alpha N *}\mathbf{R}^k \tag{11.2.34}$$

If there is no restriction on the motion in the tangent plane, two additional rows $\mathbf{I}\delta\mathbf{R}^k_{\alpha T} = -\mathbf{R}^k_{\alpha T}$ can be added to the system, assuring $\mathbf{R}^{k+1}_{\alpha T} = \mathbf{0}$. Alternatively, one can use $\mathbf{\Omega}_{\alpha\alpha} = [\mathbf{0}, W_{\alpha\alpha NN}]$ and this way avoid processing of $\mathbf{R}_{\alpha T}$.

11.3. Gauss-Seidel method

One of the characteristic features of the Contact Dynamics Method is a block Gauss-Seidel relaxation employed to solve (11.0.1). The method is rather robust in practise, although its local convergence can be extremely slow. The advantage is, that it converges regardless of the invertibility of $\mathbf{W}$. An incomplete proof of the convergence for the three-dimensional frictional contact problem was given by Jourdan *et al.* [**109**] (where also a complete, two-dimensional proof can be found). Let $\{\mathcal{C}_\alpha\}$ be the set of all individual constraints, prescribed as $\mathbf{C}_\alpha(\mathbf{U}_\alpha, \mathbf{R}_\alpha) = \mathbf{0}$. The Gauss-Seidel method can be summarised as follows

ALGORITHM 11.3.1. *Gauss_ Seidel* (ϵ, K)

```
1  k = 1
2  do
3     for all α do
4        B̄_α = Σ_{β≠α} W_αβ R_β + B_α
5        solve C(W_αα R_α^{k+1} + B̄_α, R_α^{k+1}) = 0
6     end for
7     err = ‖R^{k+1} − R^k‖ / ‖R^{k+1}‖
8     k = k + 1
9  while err > ε ∧ k ≤ K
```

The paradigm of a Gauss-Seidel relaxation is clearly pronounced in the above. A series of diagonal problems is solved in the internal loop from lines 3-6. As a solution method in line 5, any of the schemes described in the previous section can be used. The single diagonal problem is usually quite well behaved, and a semi-smooth Newton method requires just few iterations (without line search) in order to find a solution. A very simple convergence criterion is used in lines 7, 9. In the literature specific to the Contact Dynamics Method, some more elaborate criteria are mentioned [**102, 177**].

11.4. Literature

The Newton method under consideration in Section 11.2 stems from a broader range of schemes for non-differentiable systems. General developments of this kind were discussed by Pang [**166**] and Qi and Sun [**175**]. Global convergence of such schemes was discussed by Han *et al.* [**82**], Ferris and Lucidi [**69**], or Dai [**53**]. In the context of the mixed frictional contact formulation Alart and Curnier [**7**] discuss the generalised Newton method (GNM), which belongs to the same category. An observation made in [**7**], about the practical robustness of GNM applied to frictionless problems, was later confirmed under the umbrella of the primal-dual active set method [**94**]. The latter was shown to be equivalent to the semismooth Newton method by Hintermüller *et al.* [**93**]. In case of frictional problems, Christensen *et al.* [**41**] developed a linearisation along the lines of [**166**], and presented two-dimensional linearly elastic examples. In three dimensions, the non-smooth Newton scheme was recently applied by Jones and Papadopoulos [**107**] to solve anisotropic frictional problems. The reference development for Section 11.2, Hüeber *et al.* [**96**], discusses a multi-grid implementation of the fixed point Tresca

approach and compares it with the semi-smooth Newton step employing a direct solver. Barral *et al.* [**19**] describe a generalised Newton method applied to a planar frictionless contact problem with Maxwell-Norton material. Zavarise and Wriggers [**214**] obtain a super-linear method for the augmented Lagrangian formulation of the frictionless contact problem. The first order update of Lagrange multipliers is enhanced by a heuristic method of retrieving higher order information. The technique retains simplicity of the Uzawa-like algorithm, although its extensibility to the Coulomb friction problem is not clear. Large multi-body contact problems were not extensively studied within the context of Newton methods. Two-dimensional frictionless developments involving the primal-dual active set approach can be found in Ainsworth and Mihai [**6**], as well as in Hüeber and Wohlmuth [**97**].

Multi-body formulations, including friction and finite kinematics usually resort to methods avoiding formation of global tangents. The Gauss-Seidel technique of the Contact Dynamics method [**156, 102**] is a good example here. Jourdan *et al.* [**109**] prove the convergence of the Gauss-Seidel scheme for two-dimensional problems. The scheme is similar to other splitting-type techniques, relaying on the fixed point ideas. In an elegant paper, Laborde and Renard [**168**] discuss a number of fixed point strategies to the frictional contact problem. Their formulation facilitates fast translation of results between finite dimensional and function space settings. Bisegna *et al.* [**30**] discuss relaxation techniques for two dimensional Signorini-Coulomb problems based on the dual formulation, and hence similar to the Gauss-Seidel approach. This is a typical splitting technique, where the friction and the contact problems are solved alternately. Another splitting based algorithm is discussed by Haslinger *et al.* [**83**] and Dostál *et al.* [**60**]. As shown in [**83**], for this type of approaches a fixed point exists for a sufficiently small friction coefficient. A splitting type method was also used by Ainsworth and Mihai [**5**] in the context of large, dynamic simulations of masonry. The primal-dual active set method was applied in order to alternatively solve the friction and the contact problems. In the context of the Gauss-Seidel method [**109**], Joli and Feng [**105**] developed linearisation of the projection formula pertinent to the Bipotential Method [**55**], and utilised it in a Newton method, solving the local diagonal sub-problems.

In [**188**] Sha *et al.* developed a linear complementary formulation of a deformable explicit frictional contact problem and applied a conjugate gradient method as a solution strategy. A conjugate gradient method is also developed by Heinstein and Laursen [**87**] and applied in the context of an incremental matrix-free formulation. In the context of two-dimensional granular media simulations, Renouf and Alart [**177**] develop a preconditioned conjugate-gradient solver, which is shown to outperform the Gauss-Seidel method used in Contact Dynamics [**102**].

For rigid bodies, Stewart and Trinkle [**199**] use polyhedral approximation of the friction cone and develop a linear complementary (LCP) formulation solvable by Lemke's method [**133**]. A number of following developments in rigid multi-body dynamics involves similar LCP approach [**10, 194, 70**]. An interesting and efficient simplification of the rigid multi-body frictional contact problem was developed by Kaufman *et al.* [**114**] and applied in the field of computer animation.

CHAPTER 12

Implementation

The framework described in the foregoing chapters has been implemented as a computer program named *Solfec*. The dynamic and the quasi-static time-steppings underlying this implementation have been summarised below.

ALGORITHM 12.0.1. *Solfec_Dynamic* (h, T)

1 ***for*** $t = 0$ ***while*** $t < T$ ***do***
2 $\quad \mathbf{q}^{t+\frac{h}{2}} =$ ***half-step*** $(\mathbf{q}^t, \mathbf{u}^t)$
3 $\quad \{\mathcal{C}_\alpha\}^c =$ ***update-contacts*** $\left(\mathbf{q}^{t+\frac{h}{2}}\right)$
4 $\quad \{\mathcal{C}_\alpha\}^j =$ ***update-joints*** $\left(\mathbf{q}^{t+\frac{h}{2}}\right)$
5 $\quad (\mathbf{H}, \mathbf{W}, \mathbf{B}) =$ ***compute-operators*** $\left(\{\mathcal{C}_\alpha\}^c \cup \{\mathcal{C}_\alpha\}^j\right)$
6 $\quad$ ***solve*** $\mathbf{C}\,(\mathbf{WR} + \mathbf{B}, \mathbf{R})$
7 $\quad \mathbf{u}^{t+h} = \mathbf{u}^t + \mathbf{M}^{-1} h \mathbf{f}^{t+\frac{h}{2}} + \mathbf{M}^{-1}\mathbf{H}^T\mathbf{R}$
8 $\quad \mathbf{q}^{t+h} =$ ***half-step*** $\left(\mathbf{q}^{t+\frac{h}{2}}, \mathbf{u}^{t+h}\right)$
9 $\quad t = t + h$
10 ***end for***

The time step is h and the duration of simulation is T are the arguments of Algorithm 12.0.1. In the second line, the mid-step configuration $\mathbf{q}^{t+\frac{h}{2}}$ is obtained by performing a half-step, according to (5.1.1) for the linear and deformable motion, and according to (5.2.9) for rigid rotations. Based on the mid-step configuration, a contact detection algorithm is executed in the third line. This could be any of the methods presented in Sections 9.3.4, 9.3.5 or 9.3.6, combined with an extraction of local frames as described in Section 9.4. The contact update involves deletions of local frames for element pairs whose overlap has ceased. It involves as well an update of all local frames related to the new and to the old contact points. In the fourth line, the local frames corresponding to joints are updated. The operators describing local dynamics are computed in line 5, as described in Section 7.1. The constraint equations are solved next (line 6), where one of the methods described in Chapter 11 is employed. It is recalled, that steps (10.4.9-10.4.13) need to be executed in order to account for cohesion. The velocity update follows next, and it is accompanied by the final update of configuration in line 8. For linear and deformable motion $\mathbf{q}^{t+h}$ is obtained according to (5.1.3). For rigid rotations the final configuration is computed with (5.2.13) or (5.2.16).

ALGORITHM 12.0.2. *Solfec_Static* (h, T, K, r)

1 ***for*** $t = 0$ ***while*** $t < T$ ***do***
2 $\quad \{\mathcal{C}_\alpha\}^c =$ ***update-contacts*** $(\mathbf{q}^t)$
3 $\quad \mathbf{M} =$ ***scale-inertia*** $(h, 4.0, \{\mathcal{B}_i\})$
4 $\quad \mathbf{q}_0^{t+h} = \mathbf{q}^t$, $k = 1$, $V = \infty$
5 $\quad$ ***do***
6 $\quad\quad \{\mathcal{C}_\alpha\}^c =$ ***update-gaps*** $\left(\mathbf{q}^{t+h}\right)$
7 $\quad\quad \{\mathcal{C}_\alpha\}^j =$ ***update-joints*** $\left(\mathbf{q}^{t+h}\right)$

```
8       (H, W, B) = compute-operators ( {C_α}^c ∪ {C_α}^j )
9       solve C (WR + B, R)
10      u^{t+h} = A^{-1} h f (t + h, q^t) + A^{-1} H^T R
11      q_k^{t+h} = step ( q_{k-1}^{t+h}, u^{t+h} )
12      E^k = kinetic-energy-of-mass-centres ({B_i})
13      if k = 2 then V = max ( log ( E^1 / E^2 ), 0 )
14      k = k + 1
15    while k < K ∧ log ( E^{k-1} / E^k ) ≥ rV
16    t = t + h
17  end for
```

The quasi-static Algorithm 12.0.2 takes as its arguments the time step h, the duration T, a dynamic relaxation iterations bound K, and a kinetic energy drop rate factor r. There are two loops in the algorithm, between lines 1-17 and between lines 5-15. The external loop advances the artificial time, while the internal one attempts to find a steady state solution for each instant of time. Contacts are detected and update in the external loop, in line 2. This task is relatively costly and hence we do not wish to run it too frequently. Instead, in the inner loop (line 6), only contact gaps are updated, according to formula (10.1.1). Another motivation behind the sparser updates of contacts, is to avoid "noise" in the dynamic relaxation loop 5-15. This would be introduced due to the small changes of contact frames occurring after the configuration updates in line 11. In line 3, the inertia operators of individual bodies are scaled in order to assure a uniform damping of the implicit Euler scheme. Such scaling has been described in Section 5.3. In our routine the maximal eigenvalue of $\mathbf{M}^{-1}\mathbf{K}$ is scaled in order to assure $\lambda_{max} h = 4$ for each pseudo-rigid body. Once the gaps have been updated (line 6), the update of local frames related to the equality constraints follows in line 7. The local dynamics operators are computed in line 8 and the solution of the constraint equations follows in the next line (note, that (10.4.9-10.4.13) is executed in the presence of cohesion). The velocity update is performed next (line 10). It should stressed, that the time is fixed here to $t + h$, so that the time-dependent loadings (or constraints) do not change in the internal loop. The configuration update follows in line 11. For pseudo-rigid bodies the formula $\mathbf{q}_k^{t+h} = \mathbf{q}_{k-1}^{t+h} + h\mathbf{u}^{t+h}$ is executed, while a general step update in line 11 hints the possibility of an analogous update for rigid bodies $\mathbf{\Lambda}_k^{t+h} = \mathbf{\Lambda}_{k-1}^{t+h} \exp\left[h\mathbf{\Omega}^{t+h}\right]$. The kinetic energy of mass centres is computed next. The rate of decay of the energy is used as a termination criterion for the inner loop (together with the bound on the maximal number of iterations K). If the energy is decreasing, the initial slope of its drop is used as a reference value V. On the other hand, if the energy grows, then $V = 0$ and the inner loop terminates. The logarithmic scale is employed in order to conveniently account for the drop spanning several orders of magnitude. The rate of the energy drop is used, because the graph of the kinetic energy usually resembles Figure 12.0.1. Naturally, the local convergence of our simplified relaxation method cannot be fast, as the necessary linearisations have been skipped.

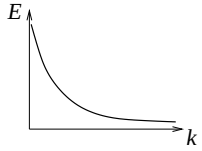


FIGURE 12.0.1. A typical decay of kinetic energy for the dynamic relaxation loop.

CHAPTER 13

Examples

13.1. Rigid rotations

Schemes (5.2.9-5.2.16) from Chapter 5 are further compared with LIEMID[EA] by Krysl [**126**] and the explicit method by Simo and Wong [**100**].

13.1.1. Unstable rotation. This example is referred to after Simo and Wong [**100**]. The example is based on the fact that rigid rotation is stable only about the axes of minimum and maximum moment of inertia (Arnold [**13**], Chapter 29.2). Small perturbation of rotation around the axis of intermediate moment of inertia leads to unstable oscillation. The initial rotation is identity, the initial angular velocity is zero, and the referential inertia tensor is $\mathbf{J} = \mathrm{diag}\,[5, 10, 1]$. The spatial torque reads

$$\mathbf{t}\,(t) = \begin{cases} [20, 0, 0] & \text{for} \quad 0 \le t < 2 \\ [0, 1/\,(5h)\,, 0] & \text{for} \quad 2 \le t \le 2 + h \\ [0, 0, 0] & \text{for} \quad 2 + h < t \end{cases}$$

so that an impulse inverse proportional to the time step is delivered at $t = 2$. Due to the dependence of torque on the time step, the convergence rate can be only

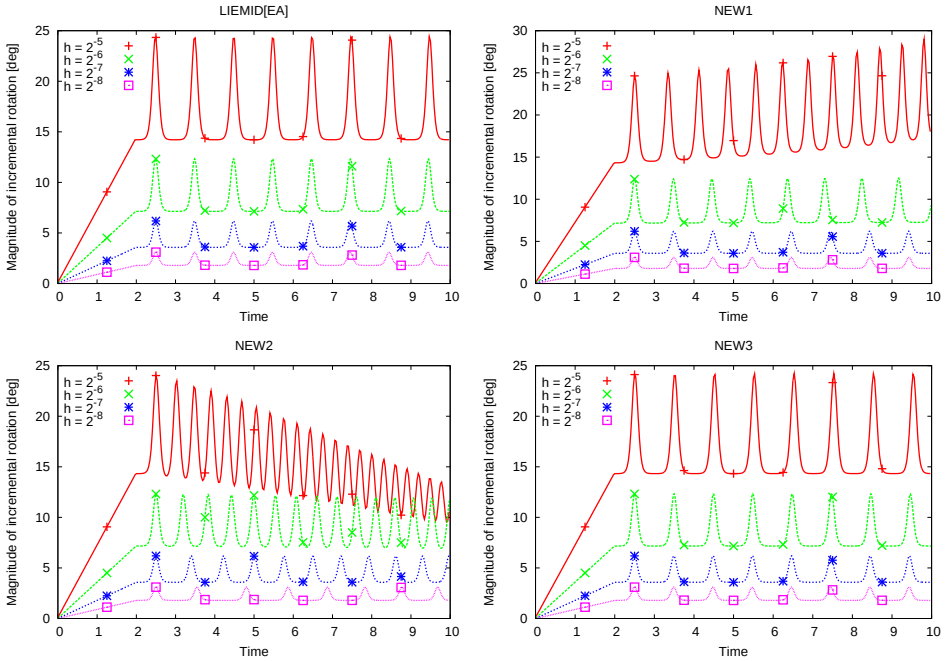


FIGURE 13.1.1. Unstable rotation. Magnitude of the incremental rotation vector for a range of time steps.

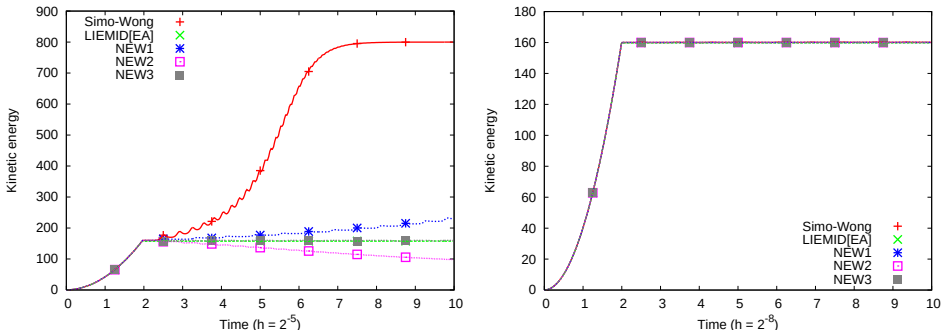


FIGURE 13.1.2. Unstable rotation. Kinetic energy for $h = 2^{-5}$ (left) and for $h = 2^{-8}$ (right).

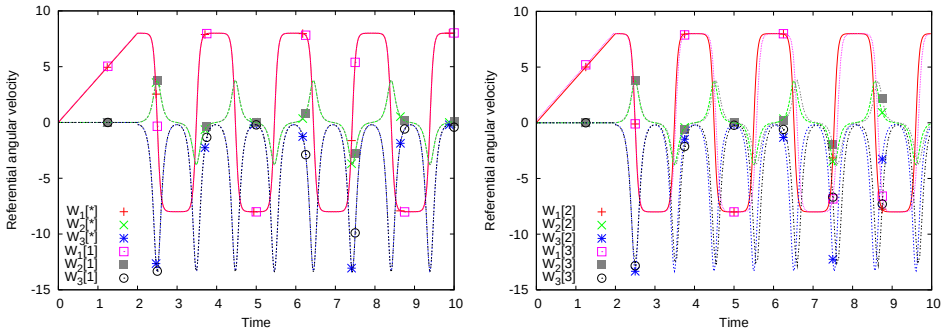


FIGURE 13.1.3. Unstable rotation. Components of the angular velocity in body frame. The $\mathbf{W}[*]$ components have been obtained with the explicit scheme by Simo and Wong and $h = 0.001$. The $\mathbf{W}[1]$ components have been obtained with NEW1 and $h = 0.01$. Components $\mathbf{W}[2]$ correspond to the largest step, for which a qualitatively correct result was obtained with NEW2 ($h = 0.0019$). Components $\mathbf{W}[3]$ correspond to the similar result obtained with NEW3 ($h = 0.05$).

linear for this example. Nevertheless the convergence analysis is included, as this example seems particularly appealing in the context of contact/impact analysis.

Figure 13.1.1 compares magnitudes of the incremental rotation vector at the range of time steps from $h = 2^{-5}$ to $h = 2^{-8}$. The characteristic drift properties of the new scheme are clearly visible here. It is seen that the positive drift of NEW1 is smaller than the negative drift of NEW2. At the same time NEW3 gives the best qualitative match with the results obtained with LIEMID[EA].

Figure 13.1.2 illustrates the characteristic energy behaviour at $h = 2^{-5}$ and $h = 2^{-8}$. The energy drift of NEW1 and NEW2 is much smaller in comparison with the one experienced by the explicit scheme by Simo and Wong [**100**] at the larger time step. For the smaller time step all algorithms deliver the solution without a visible drift.

Figure 13.1.3 shows the characteristic profile of the body-frame angular velocity. High accuracy of the body-frame variables obtained with NEW1 is confirmed, as the solution obtained with this algorithm at $h = 0.01$ coincides with the reference solution obtained with the explicit scheme by Simo and Wong at $h = 0.001$. It is also visible that the relative accuracy of NEW2 is smaller, as the first qualitatively

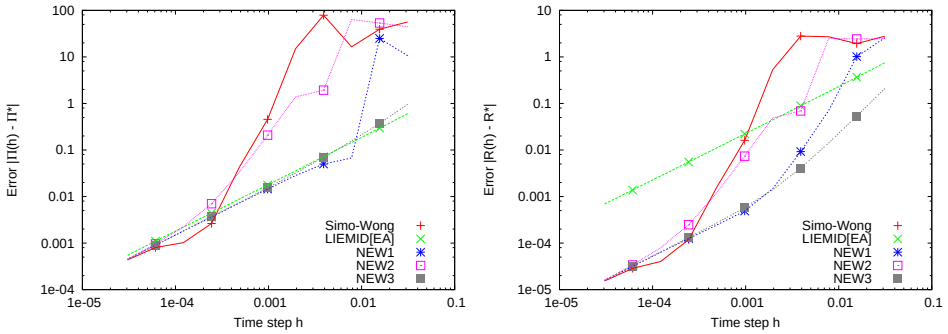


FIGURE 13.1.4. Unstable rotation. Convergence of the body-frame angular momentum $\mathbf{\Pi} = \mathbf{JW}$ (left), and the rotation operator $\mathbf{R}$ (right). The reference solutions $\mathbf{\Pi}*$ and $\mathbf{R}*$ have been computed with the explicit scheme by Simo and Wong and $h = 2^{-22}$ at time $t = 10$. The solutions $\mathbf{\Pi}(h)$ and $\mathbf{R}(h)$ were computed for time steps $h \in \left\{2^{-5}, 2^{-6}, ..., 2^{-15}\right\}$.

correct result has been obtained at $h \simeq 0.0019$. NEW3 still gives a qualitatively acceptable result at $h = 0.05$.

Figure 13.1.4 illustrates the convergence behaviour. It is noteworthy that the spatial torque formula had to be modified so that the interval $2 \leq t \leq 2 + 0.9h$ was considered for the disturbance impulse. Without this modification LIEMID[EA] consistently delivered very poor results, which is related to the fact that this scheme calculates the torque at the ends of the time interval. Again it can be seen that NEW1, NEW3 and LIEMID[EA] perform similarly in terms of the absolute error in the referential angular momentum $\mathbf{\Pi}$, although NEW1 and NEW3 seem much more accurate with respect to the computation of the rotation operator $\mathbf{R}$. The reference solution was computed in this case with the explicit scheme by Simo and Wong with $h = 2^{-22}$ at time $t = 10$.

13.1.2. Heavy top. This is the second example referred to after Simo and Wong [**100**]. The heavy symmetrical top is spinning around the fixed base point. In this example the applied torque depends on the configuration, introducing additional source of nonlinearity. The top of mass M and axis of symmetry $\mathbf{E}_3$ rotates in the uniform gravitational field $-g\mathbf{e}_3$. The spatial torque reads

$$\mathbf{t} = -Mg\mathbf{r} \times \mathbf{e}_3 \quad \mathbf{r} = l\mathbf{R}\mathbf{E}_3 = \mathbf{R}_{i3}, \;\; i \in \{1, 2, 3\}$$

where the assumed values are $M = 20$, $g = 1$, $l = 1$. As Krysl points out [**126**], the heavy top model conserves the Hamiltonian

$$H = \frac{1}{2}\pi \cdot \mathbf{j}^{-1}\pi + Mg\mathbf{e}_3 \cdot \mathbf{r}$$

where $\pi = \mathbf{jw}$ is the spatial angular momentum, $\mathbf{j} = \mathbf{RJR}^T$ is the spatial tensor of inertia, and $\mathbf{w} = \mathbf{RW}$ is the spatial angular velocity. In this example the initial rotation is $\mathbf{R}(0) = \exp[0.3, 0, 0]$, the initial angular velocity is $\mathbf{W}(0) = [0, 0, 50]$ and the spatial torque reads $\mathbf{t}(t) = 20\left[-R_{23}(t), R_{13}(t), 0\right]$.

Figure 13.1.5 illustrates the Hamiltonian history computed with the large time step $h = 2^{-5}$ (nearly $\pi/2$ of rotation increment per step) and the history computed with the smaller step $h = 2^{-8}$ (10 deg rotation increment). The characteristic drift behaviour is visible for the large step, while after the decrease of the time step

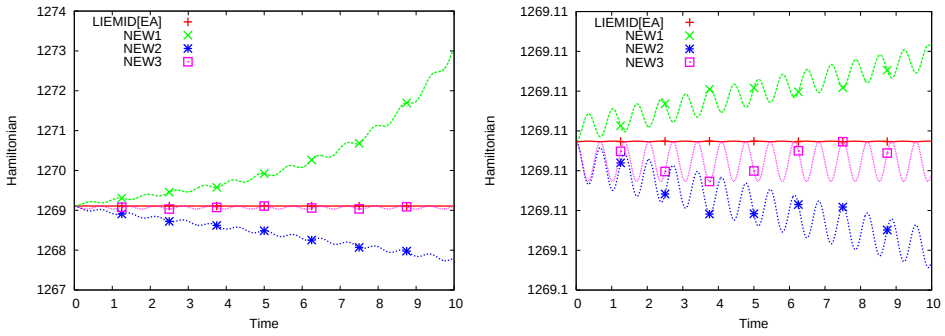


FIGURE 13.1.5. Heavy top. Plots of Hamiltonian for $h = 2^{-5}$ (left) and for $h = 2^{-8}$ (right).

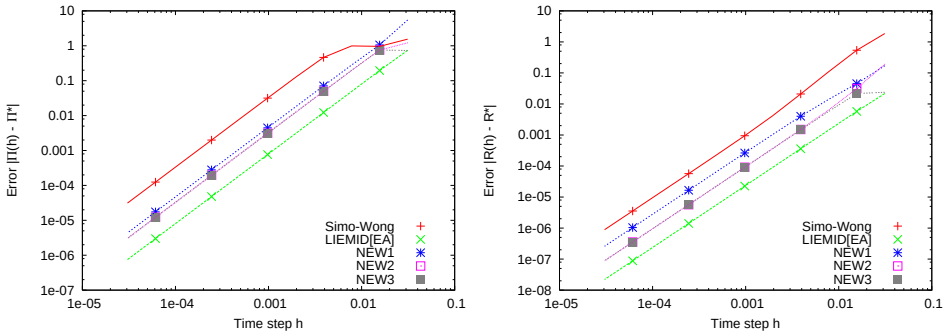


FIGURE 13.1.6. Heavy top. Convergence of the body-frame angular momentum $\Pi = \mathbf{JW}$ (left), and the rotation operator $\mathbf{R}$ (right). The reference solutions $\Pi*$ and $\mathbf{R}*$ have been computed with LIEMID[EA] and $h = 2^{-20}$ at time $t = 10$. The solutions $\Pi(h)$ and $\mathbf{R}(h)$ were computed for time steps $h \in \{2^{-5}, 2^{-6}, ..., 2^{-15}\}$.

by the factor of eight, the drift becomes negligible for NEW1 and NEW2. NEW3 behaves stably, although the negative oscillations are clearly pronounced.

Figure 13.1.6 illustrates the convergence behaviour. The reference solution was computed with LIEMID[EA] and $h = 2^{-20}$ at time $t = 10$. LIEMID[EA] also clearly outperforms other schemes. All of the proposed algorithms are positioned in between of the explicit approach by Simo and Wong and LIEMID[EA]. NEW2 and NEW3 behave alike and are more accurate in comparison with NEW1.

13.1.3. Rotating plate. In the last example the pendulum comprising a light rectangular plate and a weightless rigid rod is considered (Figure 13.1.7). The plate has dimensions $0.2 \times 0.2 \times 0.01$ and the length of the rod is $l = \sqrt{0.1}$. In the initial configuration, the rod is fixed to the mass centre of the side wall of the plate at one end. The other end rests at a spatial point placed at distance $h = 0.3$ above the mass centre of the plate. The configuration of the plate is $\mathbf{q} = [\mathbf{R}, \bar{\mathbf{x}}]$, where $\bar{\mathbf{x}}$ is the spatial placement of the mass centre. The initial configuration reads $\mathbf{q}(0) = [\mathbf{I}, \mathbf{0}]$, and the initial angular velocity is $\mathbf{W}(0) = [0, 0, 50]$. The initial linear velocity is zero. The mass density is $\rho = 1$ and the the uniform gravitational field is $-g\mathbf{e}_3$, where $g = 9.81$.

Figure 13.1.8 illustrates the history of the kinetic energy computed over the time interval $[0, 10]$ with the time step $h = 2^{-10}$ ($\|\Psi\| < 10\,\text{deg}$). It is seen that

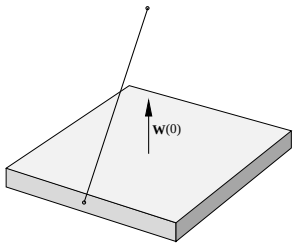


FIGURE 13.1.7. Rotating plate. Rectangular plate with the initial angular velocity $\mathbf{W}(0)$ is constrained by the rigid rod fixed to the centre of the side wall. The other end of the rod rests at a spatial point passing trough the axis collinear with $\mathbf{W}(0)$ and coincident with the mass centre of the plate.

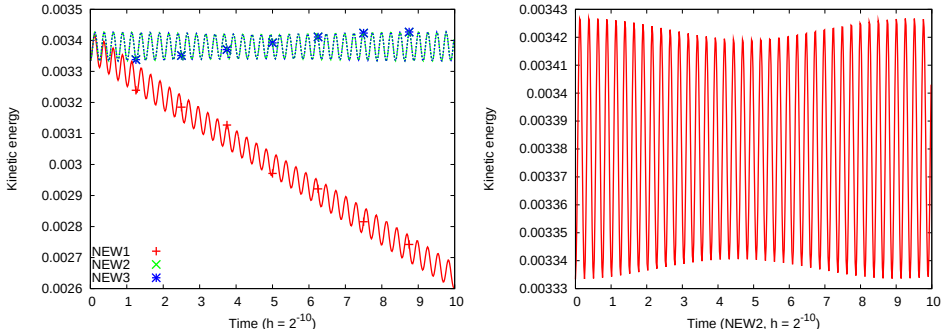


FIGURE 13.1.8. Rotating plate. Kinetic energy computed with $h = 2^{-10}$ by the three proposed algorithms (left), and a closer look at the kinetic energy computed with NEW2 (right).

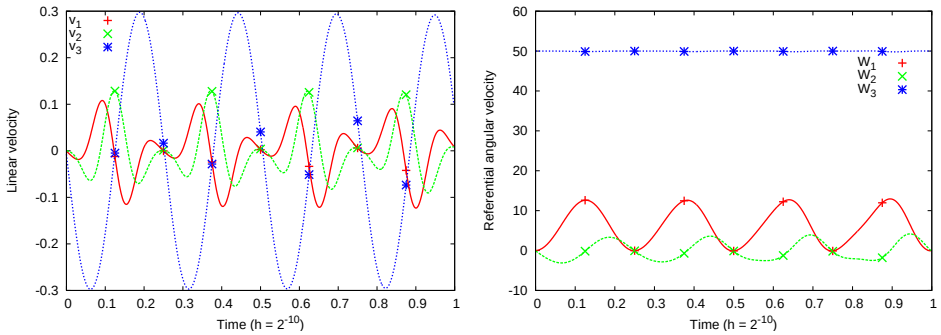


FIGURE 13.1.9. Rotating plate. The linear (left) and the angular (right) velocities over the time interval $[0, 1]$, computed with NEW2 at the time step $h = 2^{-10}$.

an interaction between the drift of the spatial angular momentum and the imposed constraint results in the considerable loss of energy for NEW1. The momentum conserving schemes NEW2 and NEW 3 pursue the analysis without a visible dissipation.

NEW2 was utilised in order to obtain the time histories of the linear and the angular velocities in Figure 13.1.9. Fast rotation around the vertical axis stabilises the plate so that the mass centre oscillates around its initial position.

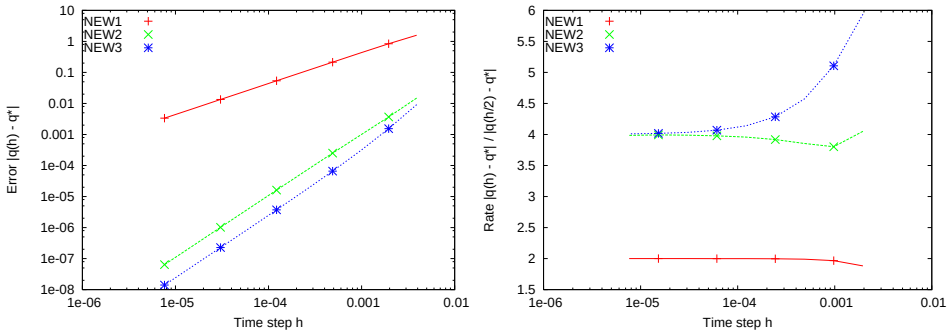


FIGURE 13.1.10. Rotating plate. Absolute error of the configuration $\mathbf{q} = [\mathbf{R}, \bar{\mathbf{x}}]$ (left), and the convergence rate (right). The reference solution $\mathbf{q}*$ has been computed with NEW3 and $h = 2^{-22}$ at time $t = 1$. The solution $\mathbf{q}(h)$ was computed for time steps $h \in \{2^{-8}, 2^{-9}, ..., 2^{-18}\}$.

Figure 13.1.10 illustrates the convergence. The reference solution $\mathbf{q}*$ was computed with NEW3 and $h = 2^{-22}$ at time $t = 1$. The solution $\mathbf{q}(h)$ was computed for time steps $h \in \{2^{-8}, 2^{-9}, ..., 2^{-18}\}$. The momentum drift of NEW1 reduces its accuracy to the first order for the considered instance of the constrained motion. NEW2 and NEW3 maintain the second order accuracy. Clearly, NEW3 is the most accurate scheme.

13.2. Contact search

We illustrate performance of the broad phase algorithms for the pairwise overlap detection between the axis aligned bounding boxes. Three kinds of box test sets are used in the evaluation. A $2 \times 2 \times 2$ cube is filled with: a randomly generated box set, a set of adjacently packed boxes, and a set of spherically distributed boxes. These are illustrated in Figure 13.2.1. All boxes are of a cubical shape. Their size is chosen, so that each box has on average 10 overlaps with other boxes in all of the test sets.

Figures 13.2.2, 13.2.3 and 13.2.4 illustrate the runtimes[1] for sizes of test sets ranging from 10^4 to 10^6. Clearly, the plane-sweep algorithm using only the priority search tree as a dynamic rectangle structure performs very poorly (SWEEP-PST2D). This is because the priority tree is essentially one dimensional. The hybrid approach by Zomorodian and Edelsbrunner [**216**] performs extremely well in most of the cases. It consistently outperforms the algorithms proposed in Chapter 9 for

[1] 1.7GHz CPU with 1GB of RAM

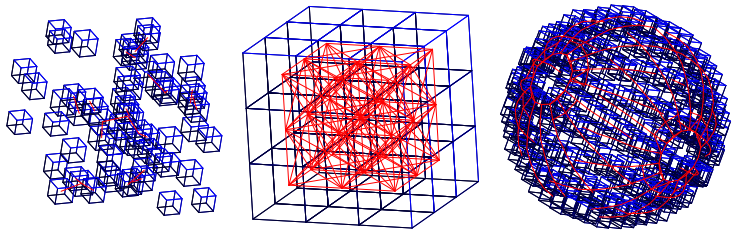


FIGURE 13.2.1. Examples of three classes of testing sets: random, adjacent, and spherical distributions of bounding boxes.

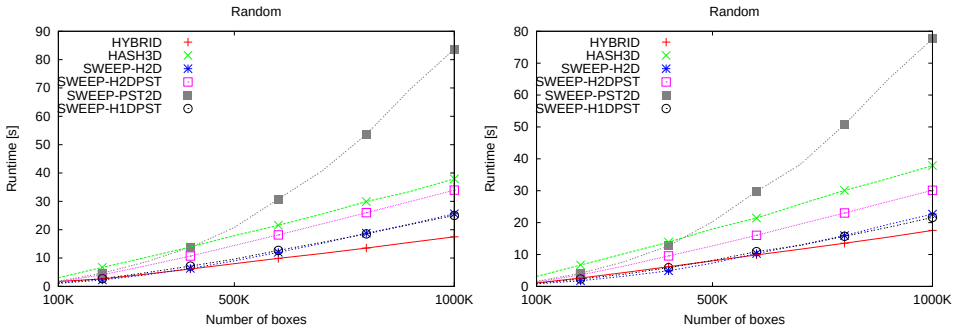


FIGURE 13.2.2. Random distribution. Without (left) and with (right) time-coherence.

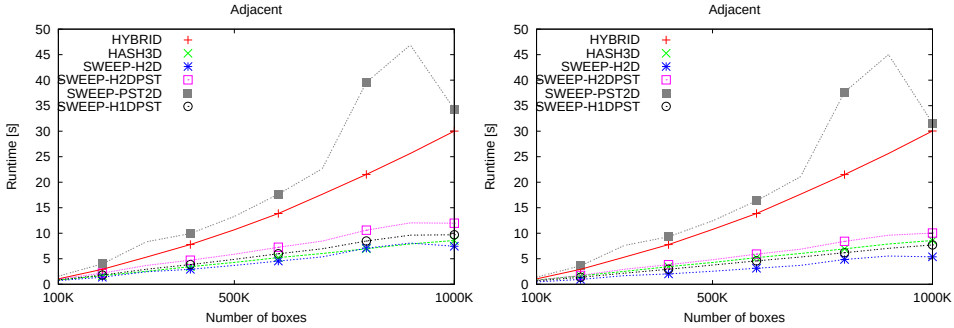


FIGURE 13.2.3. Adjacent distribution. Without (left) and with (right) time-coherence.

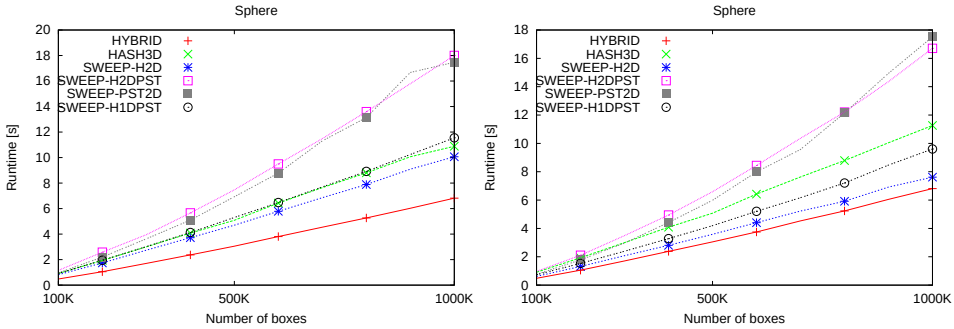


FIGURE 13.2.4. Sphere distribution. Without (left) and with (right) time-coherence.

random and spherical box distributions. Without surprise, the algorithms based on spatial hashing perform well for the uniform distribution of adjacent boxes. It should be noted, that the simple combination of sweeping and two dimensional hashing performs best among the proposed schemes (SWEEP-H2D). The second is the sweeping combined with the dynamic rectangle structure based on one-dimensional hashing and the priority search tree (SWEEP-H1DPST). This structure most logically uses strengths of hashing and the combinatorial filtering property of the binary tree. It can also be noticed, that the time-coherence (linear time sorting along the sweep dimension) has only a minor effect on the performance. This suggests that

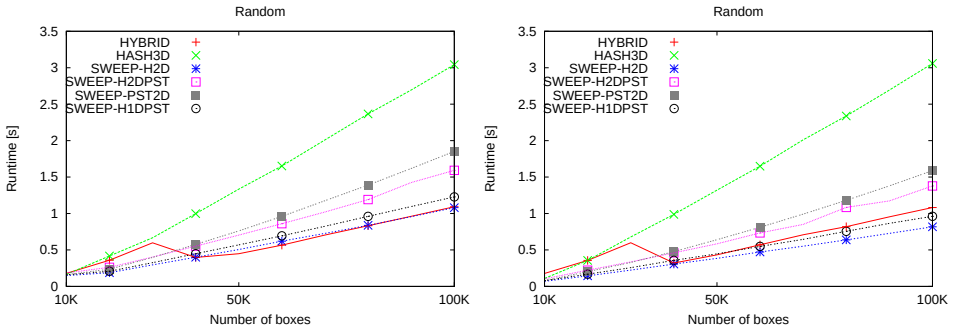


FIGURE 13.2.5. Random distribution ($\leq 10^4$). Without (left) and with (right) time-coherence.

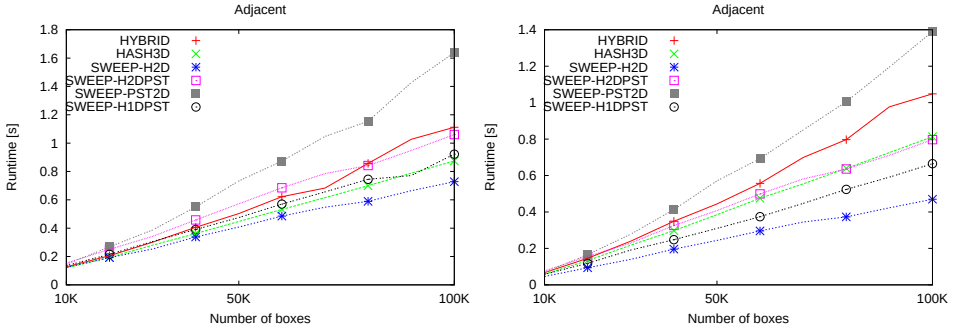


FIGURE 13.2.6. Adjacent distribution ($\leq 10^4$). Without (left) and with (right) time-coherence.

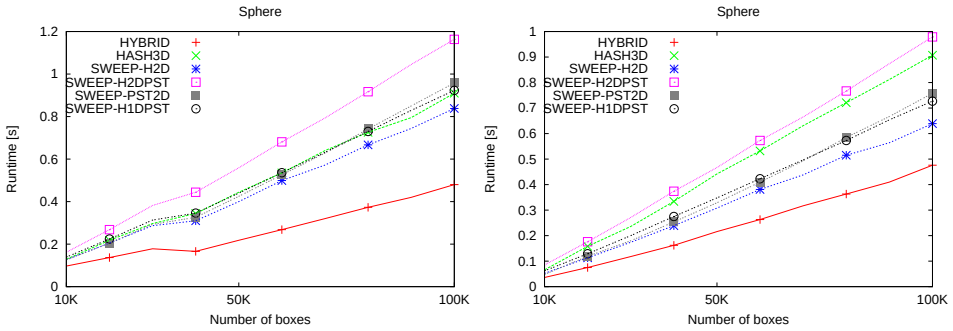


FIGURE 13.2.7. Sphere distribution ($\leq 10^5$). Without (left) and with (right) time-coherence.

the constant factors in our implementation are high. A need for optimisations is hinted.

Figures 13.2.5, 13.2.5 and 13.2.7 illustrate the runtimes for the sizes of sets below 10^4. Also here SWEEP-H2D performs best among the proposed algorithms. Nevertheless, HYBRID remains the overall winner. In general this preliminary comparison suggests that more care should be put into the implementation of SWEEP-H2D, while the other approaches can be well abandoned. Eventually, SWEEP-H1DPST might still be of interest, when disparity of box sizes and aspects ratios

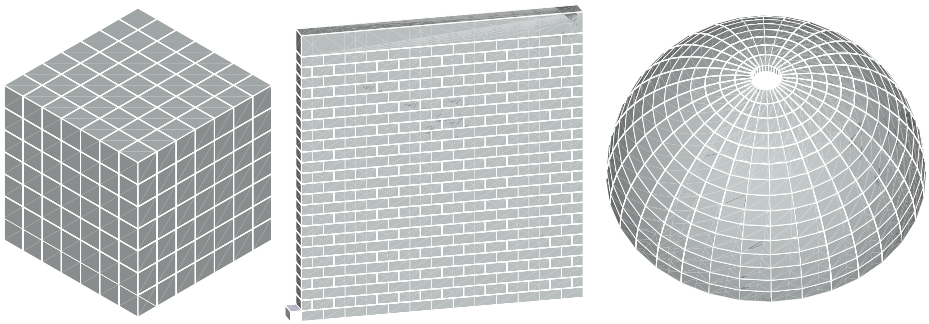


FIGURE 13.3.1. The cube, wall and dome assemblies, all resting on a rigid foundation. Cube units have dimensions $0.1 \times 0.1 \times 0.1\,m$, and are subjected to gravity of $(2, 2, -10)\,m/s^2$. Wall units are of dimensions $0.2 \times 0.2 \times 0.1\,m$, and are subjected to gravity of $(0, 0, -10)\,m/s^2$, along with the upper bar load of $(0, 0, -30)\,kN$. The inner radius of dome is $10\,m$, and the thickness is $0.6\,m$. It deforms under the gravity of $(0, 0, -10)\,m/s^2$. The lower left corner of the wall is restrained by a rigid cubic obstacle. Material properties are $15.5\,GPa$ for Young modulus, 0.2 for Poisson ratio, and $2200\,kg/m^3$ for the density.

is included. Further investigation is necessary before a definite conclusion can be reached.

13.3. Newton solvers

Three examples are studied for three sizes of assemblies and a range of friction coefficients. The two-dimensional wall example corresponds to the experimental setup by Lourenco *et al.* [**142**]. The three-dimensional cube, and the dome examples have been selected to picture convergence for various geometrical placements of contact points. The focus is on the numerical properties of `SCSOL`, rather then on the mechanical response of test examples. Assembly geometries, loading conditions and material properties are given in Figure 13.3.1. Pseudo-rigid cuboids are employed as the individual bodies, and hence a single contact point is established between each pair of adjacent bricks.

The time stepping from Section 5.3 is employed. As a quasi-static response is considered, inertia properties were scaled in order to impose uniform numerical damping. For the implicit Euler scheme, a reasonable amount of damping can be obtained for $\lambda h \geq 4$, where λ is a selected eigenvalue of $\mathbf{M}^{-1}\mathbf{K}$, h is the time step, and $\mathbf{K}$ is the current stiffness tangent [**98**]. Here $h = 1$ was assumed, and inertia tensors $\mathbf{E}_0$ were scaled, so that $\lambda_{max}\left(\mathbf{M}^{-1}\mathbf{K}\right) = 4$ for all bodies.

Table 1 summarises numbers of bodies, contact points, and condition numbers of respective $\mathbf{W}$ operators. Assemblies of variable size preserve geometrical features described in Figure 13.3.1. The condition numbers were obtained with `dgscon` routine of the sparse factorisation package SuperLU [**57**], which was also employed as the linear solver. The condition numbers are high, yet far from singular. Nevertheless, for the wall example the ill-conditioning of $\mathbf{W}$ significantly grows with the structure size. This corresponds to the discussion presented in Section 7.1. Conditioning of $\mathbf{W}$ does not directly correlate to that of Ω. In fact $\Omega = \mathbf{W}$ only if all contact points are in the frictional stick state. In most cases $\Omega \neq \mathbf{W}$ and Ω ought to be assembled with some care. As $\mathbf{W}$ corresponds to the inverse of a

TABLE 1. Numbers of bodies, contact points, and condition numbers of $\mathbf{W}$.

Example	Bodies	Contacts	$\mathbf{W}$ conditioning
CUBE1	27	63	2E+7
CUBE2	125	325	5E+7
CUBE3	343	931	9E+7
WALL1	56	147	3E+6
WALL2	162	451	4E+7
WALL3	338	963	2E+8
DOME1	60	120	7E+5
DOME2	220	440	2E+6
DOME3	480	960	8E+6

TABLE 2. Parameters of **SCSOL** used in the performance study.

σ	γ	β	J	ϵ	K	ϕ	L
0.9	0.1	0.034	0 or 10	1E-10	1000	10	6

stiffness matrix, its entries are likely to be quite small ($O\left(10^{-8}\right)$ for example). The linearised constraints though, are usually of the order $O(1)$. For this reason, to prevent ill-conditioning, system rows corresponding to those constraints are scaled by the relevant diagonal entries of $\mathbf{W}$. For example a row $...\,0\,1\,0...\,\mathbf{R} = \Pi_i$ is replaced by $...\,0\,W_{ii}\,0...\,\mathbf{R} = W_{ii}\Pi_i$. Generally, scaling is applied to system rows defined in lines 5, 8, and 9 of `NEWT`, `HYB`, and `FIX`. As a result, the condition numbers of Ω are comparable to those of $\mathbf{W}$, provided the scaling of the regularisation parameter ρ_α is not excessive (routine `SCALE`).

The input parameters of `SCSOL` are summarised in Table 2. Both the monotone ($J = 0$) and nonmonotone ($J = 10$) variants were investigated. The set of tested friction coefficients was $\mu \in \left\{0, \frac{1}{3}, \frac{2}{3}, 1\right\}$. For each discretisation (Table 1), one hundred incremental runs of the time stepping were performed. In all test cases the zero initial guess was used for $\mathbf{R}$ and $\mathbf{U}$ for the first run of `SCSOL`. The consecutive runs started from the previous solution. To report averages of entities spanning several orders of magnitude, the following procedure was applied

$$average = \exp\left(\sum_{i=1}^{n} \log\left(value_i\right)/n\right) \tag{13.3.1}$$

where n is either the total number of system solutions (when reporting the conditioning of Ω) or the total number of convergent runs (when reporting the average final value of the merit function $\mathcal{M}$). In the following, instead of referring to `SCSOL` with a particular argument `ALG`, a direct reference to `NEWT`, `HYB` or `FIX` is sometimes made. The monotone (Armijo's type, $J = 0$) line search based algorithms are denoted by `NEWT(A)` and `HYB(A)`, while the nonmonotone (Grippo's type, $J = 10$) line search based ones are denoted by `NEWT(G)`, `HYB(G)`.

For the frictionless problems `SCSOL` reduces to `UNIL`, regardless of the argument `ALG`. Results for this case are presented in Table 3. For all examples numbers of iterations are smaller than five. It is also seen that the system matrices are rather well behaved. This case can be tackled very efficiently.

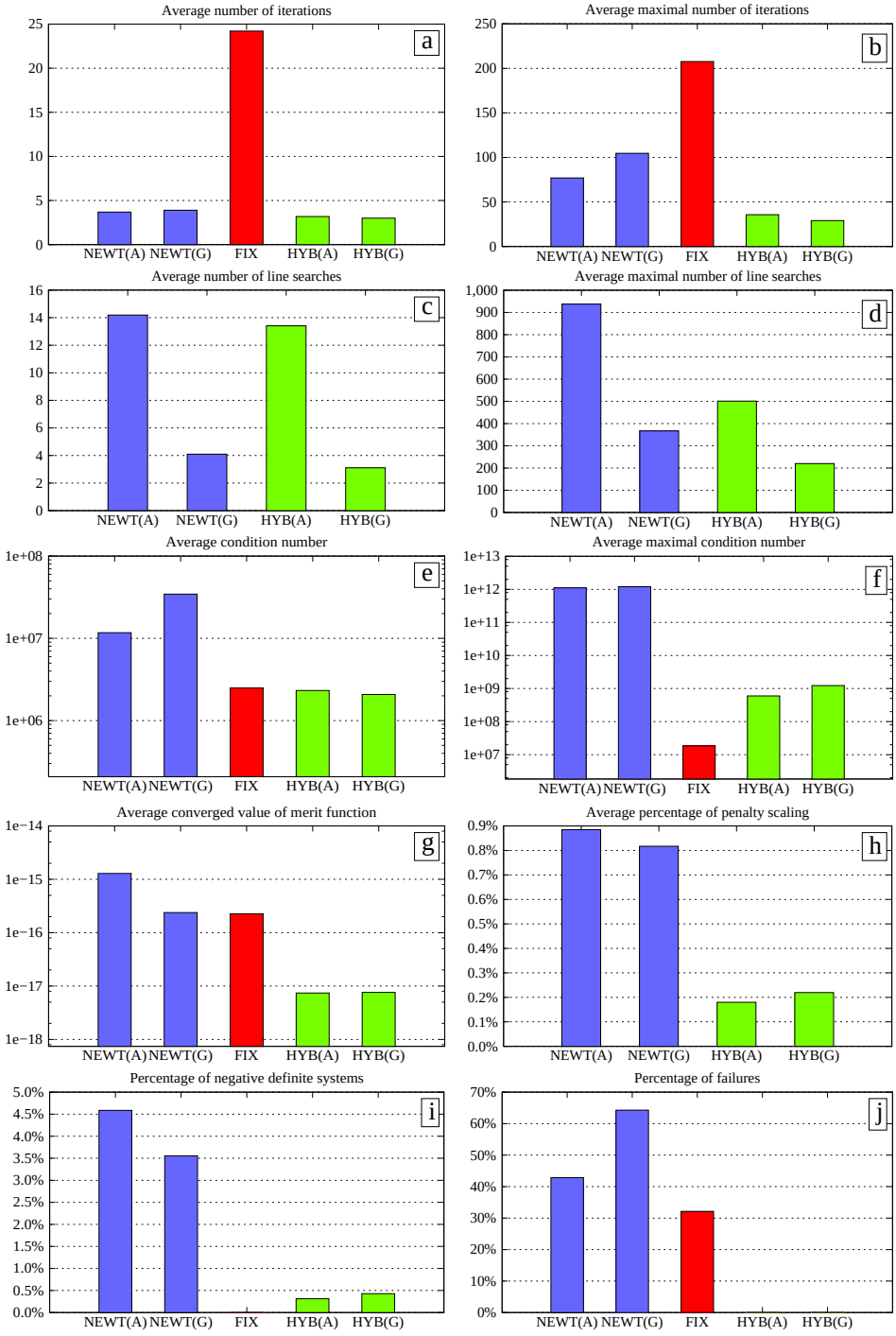


FIGURE 13.3.2. Aggregate statistics for **NEWT**, **FIX** and **HYB** (nine examples, three friction levels $\mu \in \{\frac{1}{3}, \frac{2}{3}, 1\}$, hundred increments). Note that statistics on line searches and scaling are not applicable to **FIX** and therefore omitted. **NEWT(A)** and **HYB(A)** correspond to the monotone (Armijo's) line search, while **NEWT(G)** and **HYB(G)** to the nonmonotone (Grippo's) line search.

TABLE 3. Results for the frictionless case, $\mu = 0$.

Maximal number of iterations	5
Average conditioning of Ω	2E+3
Average final value of $\mathcal{M}$	1E-23

Aggregate statistics for all frictional computations with $\mu \in \left\{\frac{1}{3}, \frac{2}{3}, 1\right\}$ have been summarised in Figure 13.3.2. The reported *average* values correspond to the 2700 runs of `SCSOL` (nine examples, three friction levels, hundred increments), while the reported *average maximal* values correspond to the averages of the 27 maxima (nine examples, three friction levels) taken over the hundred increments. As the maximal values usually correspond to the first run of `SCSOL` (starting from the zero initial guess), the average maxima give an estimate of the worst case performance.

In Figure 13.3.2 (j) it is seen, that while `NEWT` and `FIX` often failed to converge within the prescribed 1000 iterations, `HYB` is the only scheme which succeeded in all cases. It must be stated though, that while the failures of the full Newton approach correspond to the divergence (unbounded growth of the auxiliary merit function), those of the fixed point method correspond to the insufficient number of converging iterations. At the same time the full Newton method is more prone to divergence, when combined with the nonmonotone line search. This is because the minimisation along a given direction is not always successful ($\beta > 0$), and an unbounded growth of the auxiliary merit function (11.2.23) is thus possible. In the nonmonotone search case, a number of such failed minimisations can be stored and the maximal of them used as the reference value in the line search loop, resulting in a greater probability of divergence.

Comparison of the average iteration numbers in Figure 13.3.2 (a) shows that the hybrid approach inherits good local convergence properties of the full Newton scheme - the numbers of iterations are similar for both approaches (less then 5). At the same time the fixed point method needs considerably more iterations to converge (25 on average). In Figure 13.3.2 (b) it is seen that the average worst case performance of `HYB` compares favourably with the competitors. The nonmonotone version of the line search results in slightly smaller numbers of iterations for the hybrid approach, while it is quite on the contrary for the full Newton scheme (cf. comments in the previous paragraph). It should be noted, that the number of iterations for the fixed point scheme was found to be clearly related to the problem size (although it cannot be deduced from the presented figures).

In Figures 13.3.2 (c), (d) it can be seen that the nonmonotone line search consistently results in a smaller average numbers of line searches, when compared to the Armijo's type line search.

In terms of the system conditioning, it is seen in Figure 13.3.2 (e) that the hybrid linearisation inherits good properties of the fixed point scheme. The high worst case averages in Figure 13.3.2 (f) correspond to the nearly singular systems occurring towards the end of the first solver run. This issue does not represent a significant numerical difficulty, as SuperLU is capable of tackling ill-conditioned problems. The ill conditioning of systems produced by `HYB` is milder, compared to those resulting from `NEWT`.

Figure 13.3.2 (g) shows that the hybrid scheme on average results in the smallest final values of the auxiliary merit function. This is in relation with the amount of penalty scaling, presented in Figure 13.3.2 (h), which is smaller for the hybrid method (the penalty scaling percentage equals, for one solver run, to the percentage of regularisation parameters ρ_α affected by the routine `SCALE`). Similarly, the

penalty scaling is related to the average worst case system conditioning presented in Figure 13.3.2 (f).

It is seen in Figure 13.3.2 (i), that the full Newton scheme generated roughly ten times more negative definite systems, compared with the hybrid method (which produced less then 0.5% of them). Using the unconstrained minimisation analogy, one could say that the full Newton method visits the tops of the hills more frequently then the hybrid scheme. This might to some extent explain its poor robustness.

In conclusion, the full Newton scheme (`NEWT`) appears to be unreliable in our setting, although it performs pretty well, whenever convergent. The fixed point method (`FIX`) performs robustly, and usually deals with well conditioned systems. Nevertheless it does fail to converge within a thousand iterations for relatively elementary test examples. The hybrid linearisation (`HYB`) nearly consistently delivers the best performance, especially when combined with the nonmonotone line search.

13.4. Some benchmarks

Several benchmarks are presented. The purpose is to validate the implementation on few simple, documented examples.

13.4.1. Pendulum.

Reference: W. Rubinowicz, W. Królikowski, Mechanika teoretyczna (Theoretical mechanics), Państwowe Wydawnictwo Naukowe, Warszawa, 1998, pp. 91-99.
Summary: A mathematical pendulum composed of a mass point and a weightless rod swings with a large amplitude. Pendulum period, energy conservation, constraint satisfaction and convergence are examined.
Kinematics/Analysis/Solver: Rigid/Dynamic/Gauss-Seidel

The period of an oscillatory mathematical pendulum reads

$$T = 2\pi\sqrt{\frac{l}{g_3}}\left(1 + \left(\frac{1}{2}\right)^2 k^2 + \left(\frac{1\cdot 3}{2\cdot 4}\right)k^2 + \left(\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\right)k^2 + ...\right) \tag{13.4.1}$$

where

$$k = \sin\left(\frac{\theta_{max}}{2}\right) \tag{13.4.2}$$

and l is the length of the pendulum, g_3 is the vertical component of the gravity acceleration and θ_{max} is the maximal tilt angle of the pendulum. Let us assume the initial velocity of the pendulum to be zero. Thus $\theta_{max} = \theta(0)$. Taking the rest configuration position of the mass point $\bar{\mathbf{x}} = [0, 0, 0]$ and considering the swing in the $x - z$ plane, the initial position of the pendulum reads

$$\bar{\mathbf{x}}(0) = \begin{bmatrix} l\sin(\theta_{max}) \\ 0 \\ l(1 - \cos(\theta_{max})) \end{bmatrix} \tag{13.4.3}$$

Without the initial kinetic energy ($E_k(0) = 0$), the energy conservation requires that

$$E_k(t) + E_p(t) = E_p(0) \tag{13.4.4}$$

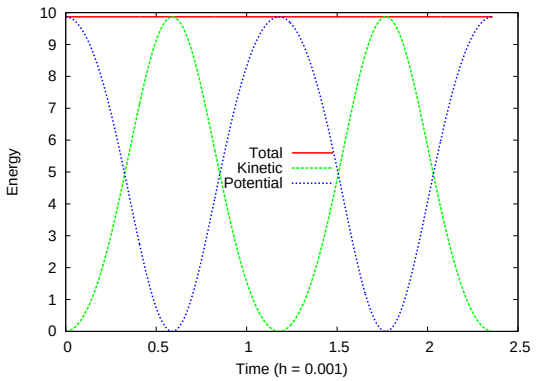


FIGURE 13.4.1. Energy balance over one period of the pendulum (red line corresponds to the total energy).

where

$$E_p(0) = mg_3\bar{x}_3(0) \tag{13.4.5}$$

and m is the scalar mass.

Input parameters

Length (m)	$l = 1$
Mass (kg)	$m = 1$
Initial angle $\theta(0) = \theta_{max}$ (rad)	$\theta_{max} = \pi/2$
Gravity acceleration $\left(m/s^2\right)$	$\mathbf{g} = \boxed{0, 0, -\pi^2}$

The gravity acceleration g_3 has been chosen so that for $\theta_{max} = 0\,\text{deg}$ there holds $T = 2s$.

Results

The table below summarises the results for the time step $h = 0.001$. It is seen that the solution is accurate and stable, regardless of the duration of the numerical simulation.

	Target	*Solfec*	Ratio
Pendulum period - 1 swing (s)	2.36068	2.63000	0.9997
Pendulum length - 1 swing (m)	1.0	1.0	1.0
Total energy - 1 swing (J)	π^2	9.86960	1.0
Pendulum period - 1000 swings (s)	2360.68	2360.68	1.0
Pendulum length - 1000 swings (m)	1.0	1.0	1.0
Total energy - 1000 swings (J)	π^2	9.86960	1.0

Figure 13.4.1 illustrates the energy balance over one period of the pendulum. The potential and kinetic energies sum up to π^2. Figure 13.4.2 shows oscillatory but stable behaviour of the equality constraint (the length of the rigid rod). Figure 13.4.3 confirms the second order convergence in the presence of equality constraints (the reference solution $\mathbf{q}*$ has been computed at time $t = 1.0$ with $h = 2^{-20}$).

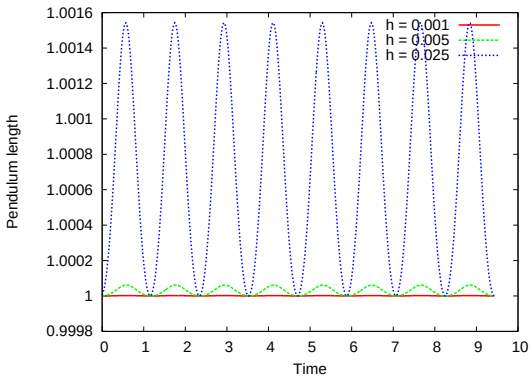


FIGURE 13.4.2. Length of the pendulum over the time of four periods, computed for several time steps $h \in \{0.001, 0.005, 0.025\}$.

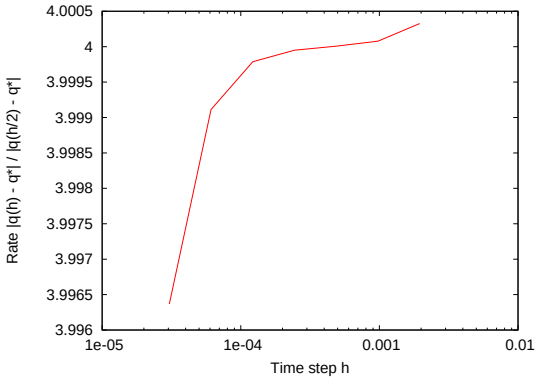


FIGURE 13.4.3. The convergence rate $\simeq 4$ confirms the second order accuracy in the presence of equality constraints.

13.4.2. Sphere impacting a plate.

Reference: The solution is self-evident.
Summary: A sphere impacts a plate. Newton impact law is validated for several values of the restitution parameter η and a single-point contact.
Kinematics/Analysis/Solver: Rigid/Dynamic/Gauss-Seidel

Sphere of radius r and with the initial velocity v_z impacts the horizontal frictionless surface (Figure 13.4.4). Single contact point is established. The pre- and post-impact velocities are related through the Newton's law

$$v_z^+ = -\eta v_z^- \tag{13.4.6}$$

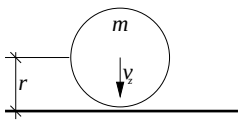


FIGURE 13.4.4. Sphere in the initial configuration.

thus for $\eta = 1$ the total energy is conserved while for $\eta < 1$ the energy is dissipated. In the initial configuration the sphere is about to hit the plate, so that $v_z(0) = v_z^-$ for the first impact.

Input parameters

Sphere radius (m)	$r = 0.1$
Sphere mass (kg)	$m = 1$
Initial velocity	$\mathbf{v}(0) = [0, 0, -4]$
Gravity acceleration (m/s^2)	$\mathbf{g} = [0, 0, -10]$
Velocity restitution	$\eta \in \{0, 0.25, 0.5, 0.75, 1\}$
Coulomb friction coefficient	$\mu = 0$

Results

Figure 13.4.5 illustrates the energy balance over the time interval $[0, 2.4]$ for the ideally elastic impact, $\eta = 1$. It is seen that the total energy is conserved, while three consecutive impacts take place. In Figure 13.4.6 the velocity component v_z is depicted for five restitution coefficients ranging from the ideally elastic to the ideally plastic one. The plots start from v_z^+ following the initial impact and thus the values $4, 3, 2, 1, 0$ correspond to the restitution coefficients $1, 0.75, 0.5, 0.25, 0$. For the consecutive impacts the post-impact velocities are appropriately decreased and eventually vanish, when the time between the two consecutive impacts becomes of the order of the time step.

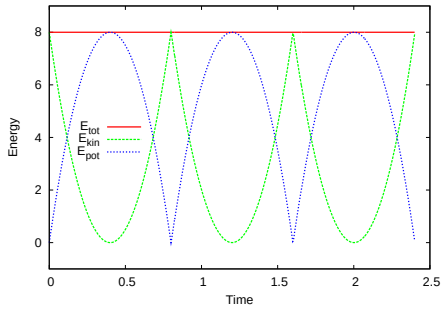


FIGURE 13.4.5. Energy balance for the ideally elastic impact $\eta = 1$, computed with the time step $h = 0.001$.

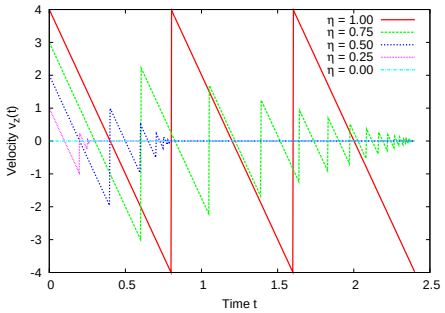


FIGURE 13.4.6. The velocity component v_z plots for restitution coefficients $\eta \in \{0, 0.25, 0.5, 0.75, 1\}$, computed with the time step $h = 0.001$.

13.4.3. Cube impacting a plate.

Reference: The solution is self-evident.
Summary: A cube impacts a plate. Newton impact law is validated for several values of the restitution parameter η and a multi-point contact.
Kinematics/Analysis/Solver: Rigid/Dynamic/Gauss-Seidel

This example mimics the previous one (Example 13.4.2), with the cube of dimensions $a \times b \times h$ replacing the sphere (Figure 13.4.7). Again, in the initial configuration the cube is about to hit the plate, so that $v_z(0) = v_z^-$ for the first impact. Due to the discretisation of the geometry four contact points are established.

Input parameters

Cube dimensions (m)	$a \times b \times h = 0.1 \times 0.1 \times 0.1$
Cube density (kg/m^3)	$\rho = 125$
Initial velocity	$\mathbf{v}(0) = [0, 0, -4]$
Gravity acceleration (m/s^2)	$\mathbf{g} = [0, 0, -10]$
Velocity restitution	$\eta \in \{0, 0.25, 0.5, 0.75, 1\}$
Coulomb friction coefficient	$\mu = 0$

Results

The mass density has been selected such that the cube example should behave exactly as Example 13.4.2. It is seen that Figures 13.4.5 and 13.4.8 are identical. The same can be said about Figures 13.4.6 and 13.4.9. All the comments from Example 13.4.2 apply here.

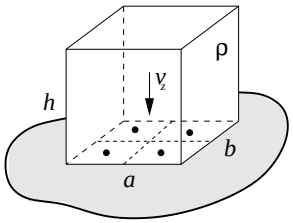


FIGURE 13.4.7. Cube in the initial configuration.

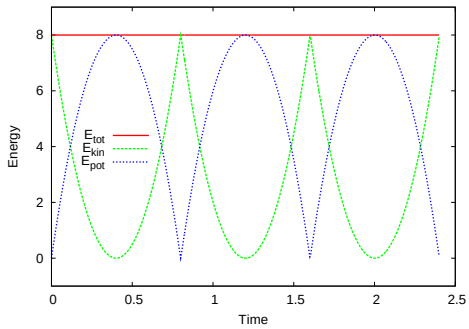


FIGURE 13.4.8. Energy balance for the ideally elastic impact $\eta = 1$, computed with the time step $h = 0.001$.

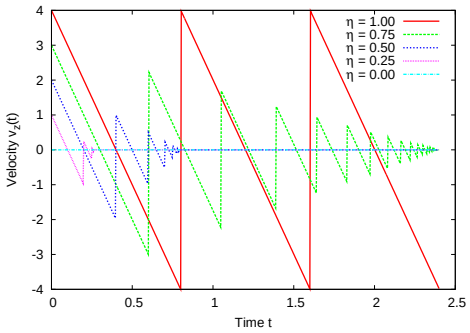


FIGURE 13.4.9. The velocity component v_z plots for restitution coefficients $\eta \in \{0, 0.25, 0.5, 0.75, 1\}$, computed with the time step $h = 0.001$.

13.4.4. Double pendulum impacting a rigid wall.

Reference: Florian A. Potra, Mihai Anitescu, Bogdan Gavrea, Jeff Trinkle. A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints, and friction. International Journal for Numerical Methods in Engineering, vol. 66, pp. 1079-1124, 2006.
Summary: A double pendulum composed of two mass points connected by weightless rods impacts a rigid wall. Position and energy plots are compared against those available in the source paper.
Kinematics/Analysis/Solver: Rigid/Dynamic/Gauss-Seidel

The reference [**70**] uses the Poisson impact model, while *Solfec* uses the Newton model. Both models are equivalent in case of frictionless impact if all restitution coefficients are identical [**42**]. This is the case in the example, thus the comparison is feasible. As *Solfec* does not handle contacts between objects with zero volume, mass points were approximated by spheres and the distance between the wall and the rest configuration of the pendulum was shifted by the sphere radius.

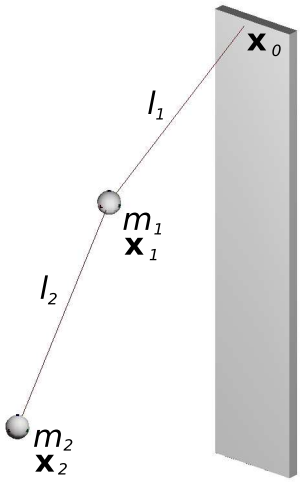


FIGURE 13.4.10. Double pendulum in the initial configuration.

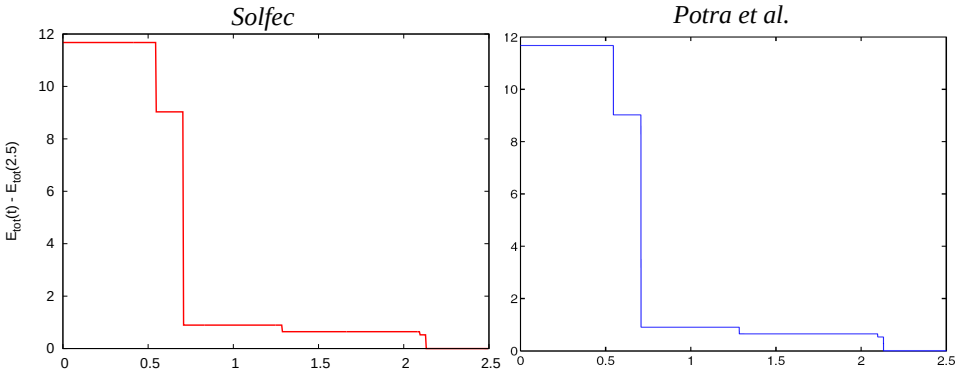


FIGURE 13.4.11. Comparison the total energy plots versus time.

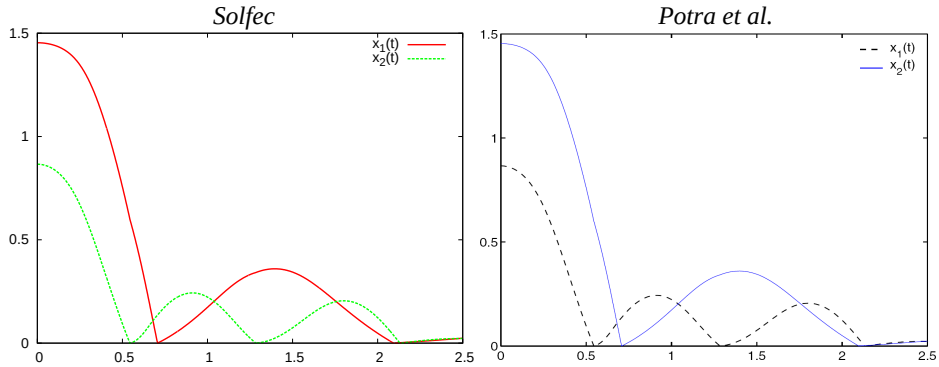


FIGURE 13.4.12. Comparison of the x-coordinate plots ($x_i\,(t)$ stands for the i-th mass point x-coordinate).

Input parameters

Mass (kg)	$m_1 = m_2 = 1$
Length (m)	$l_1 = l_2 = 1$
Point $\mathbf{x}_0$ (m)	$\mathbf{x}_0 = [0, 0, 2]$
Point $\mathbf{x}_1$ (m)	$\mathbf{x}_1 = \left[\sin\left(\frac{\pi}{3}\right), 0, 2 - \cos\left(\frac{\pi}{3}\right)\right]$
Point $\mathbf{x}_2$ (m)	$\mathbf{x}_2 = \left[\sin\left(\frac{\pi}{3}\right) + \sin\left(\frac{\pi}{5}\right), 0, 2 - \cos\left(\frac{\pi}{3}\right) - \cos\left(\frac{\pi}{5}\right)\right]$
Initial velocities (m/s)	all zero
Gravity acceleration (m/s^2)	$\mathbf{g} = [0, 0, -9.81]$
Velocity restitution	$\epsilon = 0.1$
Coulomb friction coefficient	$\mu = 0$

Results

Simulation over the time interval $[0, 2.5]$ was performed with the time step $h = 0.001$. As the reference **[70]** does not specify numerical values of the results, only a visual comparison of the total energy and the x-coordinate histories of the mass points is available. The figures are juxtaposed for clarity, although they exactly overlap when processed in a graphical software.

13.4.5. Block sliding on a frictional table.

Reference: Florian A. Potra, Mihai Anitescu, Bogdan Gavrea, Jeff Trinkle. A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints, and friction. International Journal for Numerical Methods in Engineering, vol. 66, pp. 1079-1124, 2006.
Summary: A block subjected to a sinusoidal force slips over a frictional surface. Position and velocity plots are compared against those available in the source paper.
Kinematics/Analysis/Solver: Rigid/Dynamic/Gauss-Seidel

The block has been discretised into four hexahedral elements, thus four contact points result from the element to element contact model implemented in *Solfec*. An equivalent three-dimensional model is used in *Solfec* as the reference [**70**] uses a two-dimensional set-up. The external force acting on the mass centre of the cube reads

$$\mathbf{f}(t) = [8\cos(t), 0, 0] \tag{13.4.7}$$

Input parameters

Block density (kg/m^3)	$\rho = 111.1(1)$
Block dimensions (m)	$a \times b \times h = 0.3 \times 0.3 \times 0.1$
Initial velocities (m/s)	all zero
Gravity acceleration (m/s^2)	$\mathbf{g} = [0, 0, -9.81]$
Velocity restitution	$\epsilon = 0$
Coulomb friction coefficient	$\mu = 0.8$

Results

Simulation over the time interval $[0, 10]$ was performed with the time step $h = 0.001$. As the reference [**70**] does not specify numerical values of the results, only a visual comparison of the v_x velocity component and the x-coordinate histories of the mass centre is available. The figures are juxtaposed for clarity, although they exactly overlap when processed in a graphical software.

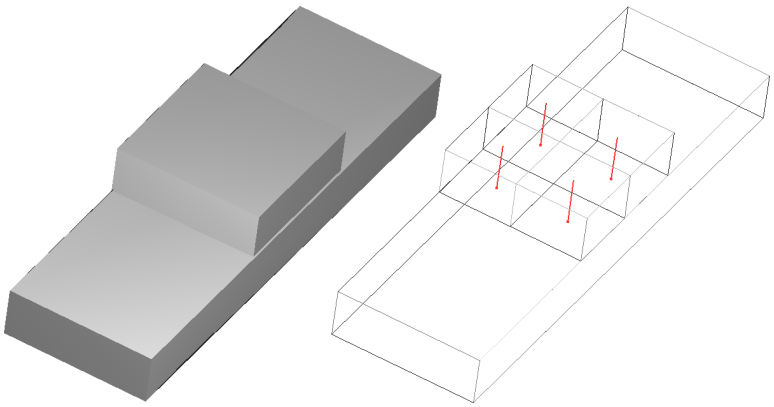


FIGURE 13.4.13. Block sliding on top of a frictional surface - initial configuration with four contact points.

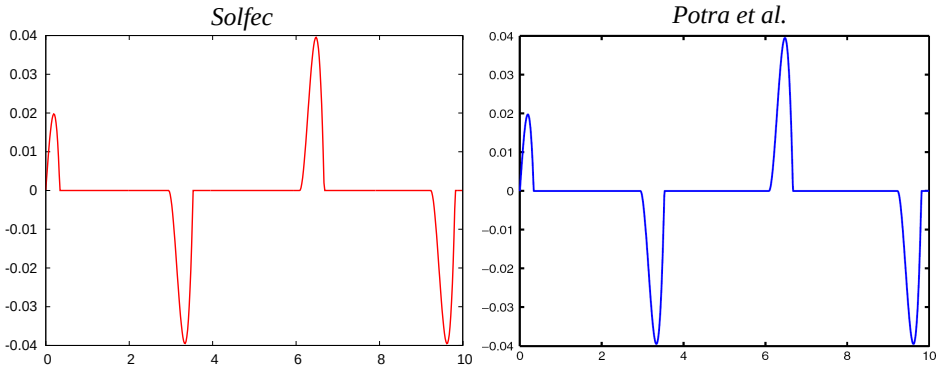


FIGURE 13.4.14. Comparison of the v_x velocity component plots of the block mass centre.

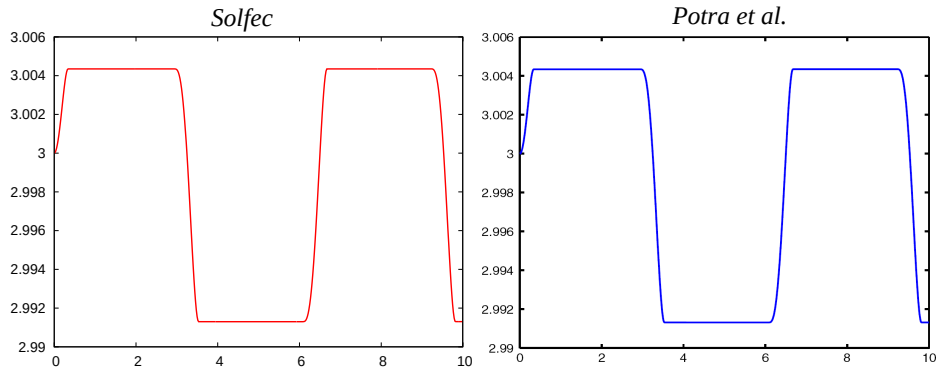


FIGURE 13.4.15. Comparison of the x-coordinate plots of the block mass centre.

13.4.6. Newton's cradle.

Reference: F. Herrmann, P. Schmälzle. A simple explanation of a well-known collision experiment, Am. J. Phys. 49, 761 (1981).
Summary: Newton's cradle is modelled by five interacting pendulums. Ideally elastic impact ($\eta = 1$) is assumed.
Kinematics/Analysis/Solver: Rigid/Dynamic/Gauss-Seidel

As shown in the reference, it is not possible to explain the behaviour of Newton's cradle solely by the principles of energy and momentum conservation. If the number of balls is larger then two, it is the dispersion-free propagation of an elastic wave which results in the characteristic behaviour of the cradle. Thus, in general, Newton's cradle is not compatible with rigid kinematics. This implies that considering all impacts at the same time results in a multiplicity of solutions. It is not guaranteed that a physically plausible solution will be selected by the numerical scheme. A simple workaround is to separate the balls by a small distance, and therefore algorithmically enforce the wave propagation effect. This approach is undertaken here.

Input parameters

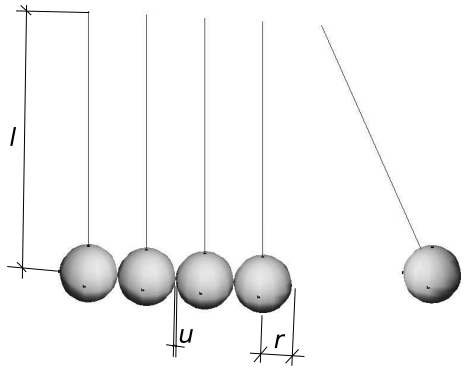


FIGURE 13.4.16. Newton's cradle in the initial configuration.

Mass density (kg/m^3)	$\rho = 1000$
Ball radius (m)	$r = 0.05$
Pendulum length (m)	$l = 0.5$
Pendulum separation (m)	$u = 10^{-10}$
Initial angle (rad)	$\theta_{max} = \pi/8$
Initial velocities (m/s)	all zero
Gravity acceleration (m/s^2)	$\mathbf{g} = [0, 0, -9.81]$
Velocity restitution	$\epsilon = 1$
Coulomb friction coefficient	$\mu = 0$

Results

Conservation of energy requires that

$$E_{tot}(t) = E_{pot}(0) = \rho \cdot \frac{4}{3}\pi r^3 \cdot |g_3| \cdot l\,(1 - \cos(\theta_{max})) = 0.195497$$

Upon full energy restitution the cradle behaves essentially as a single pendulum. Thus formula (13.4.1) can be used in order to calculate the period of the cradle. Table below summarises (among others) numerically computed periods for successively smaller time steps. It is evident that the convergence rate is linear. This is an algorithmic feature of the scheme implemented in *Solfec* in the presence of unilateral constraints (impacts, stick-slip transitions). It is also seen that the total energy is conserved exactly - regardless of the time step (note that only linear motion is present).

	Target	*Solfec*	Ratio
Cradle period T, $h = 0.01$ (s)	1.432297	1.500000	1.05
Cradle period T, $h = 0.001$ (s)	1.432297	1.438000	1.004
Cradle period T, $h = 0.0001$ (s)	1.432297	1.433000	1.0005
Total energy at $t = 10T$, $h = 0.01$ (J)	0.195497	0.195497	1.0
Total energy at $t = 10T$, $h = 0.001$ (J)	0.195497	0.195497	1.0
Total energy at $t = 10T$, $h = 0.0001$ (J)	0.195497	0.195497	1.0

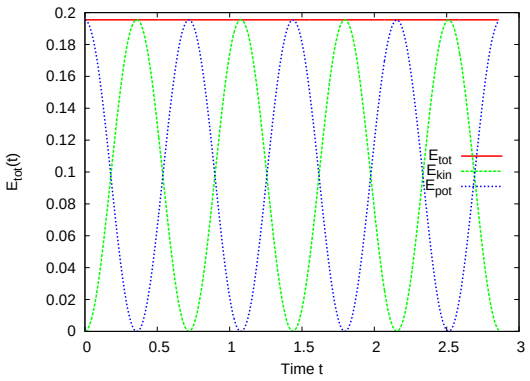


FIGURE 13.4.17. Energy balance over two periods of the cradle.

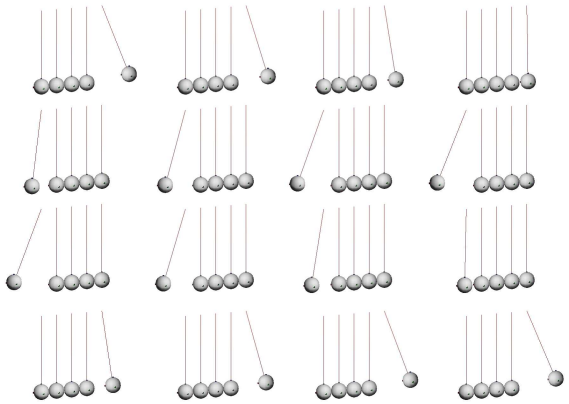


FIGURE 13.4.18. Sixteen frames of the simulation over the complete period. The sequence proceeds from left to right, top to bottom.

13.4.7. Masonry arch.

Reference: Gilbert, M. and Casapulla, C. and Ahmed, H. M., Limit analysis of masonry block structures with non-associative frictional joints using linear programming, Computers and Structures, vol. 84, pp. 873-887, 2006.
Summary: A semicircular arch is subjected to the uniform gravitational field. The dynamic stability of the arch is investigated for varying ratios of the thickness to centreline radius h/r. The results are compared against the available findings based on the limit-state analysis.
Kinematics/Analysis/Solver: Rigid/Dynamic/Gauss-Seidel

Gilbert *et al.* [**74**] present a numerical solution to the classical problem of the stability of a semicircular arch under gravity load. The analysis provided in [**74**] spans friction coefficients from the interval $[0.2, 0.8]$ and identifies three geometrical failure modes (Figure 13.4.22). The classical analysis provided by Heyman [**92**] assumes no frictional slip, and therefore covers only one case of mechanism formation (mode I - typical for large friction). Several factors need to be taken into account when considering reproduction of the results presented in Figure 13.4.22:

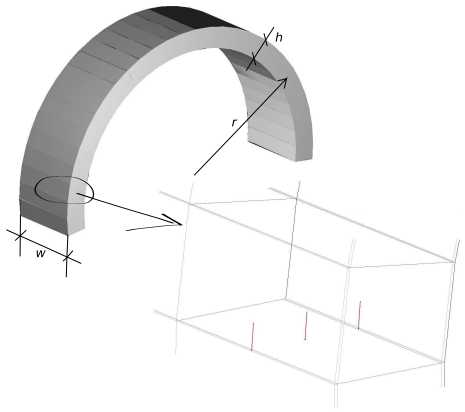


FIGURE 13.4.19. The three-dimensional arch model in *Solfec*. Each of the 27 blocks is composed of 6 elements: two along the width w, and three along the thickness h. Thus six contact points are established initially between a pair of blocks.

(1) A linear programming based limit-state formulation is employed in [**74**], whereas the dynamic contact algorithm is used in *Solfec*.
(2) The analysis provided in [**74**] is two-dimensional, whereas *Solfec* deals with a three-dimensional model.
(3) A node to face contact model is employed in [**74**], whereas the face to face (or more generally element to element) contact model is employed in *Solfec*.

Due to the modelling differences (inertial effects, contact resolution) it is reasonable to accept a margin of discrepancy between the results obtained by both methods. The dynamic stability analysis will be based on the observation of the kinetic energy histories, calculated for arches with thicknesses varying around the documented in [**74**] stability limits. Figure 13.4.19 summarises the geometry and discretisation adopted in the *Solfec* model. In order to geometrically capture the hinging effect from the first moments of simulation, the subdivision along the block thickness comprises two narrow elements at the extrados and intrados of the arch.

Input parameters

Under the assumptions discussed by Heyman [**92**], formation of a failure mechanism is of purely geometrical nature. Therefore the material parameters can be chosen arbitrary (none have been reported in [**74**]). The table below summarises the assumed parameters.

Mass density (kg/m^3)	$\rho = 1$
Centreline radius (m)	$r = 10$
Arch width (m)	$w = 5$
Number of blocks	$\{27, 15\}$
Initial velocities (m/s)	all zero
Gravity acceleration (m/s^2)	$\mathbf{g} = [0, 0, -9.81]$
Velocity restitution	$\eta = 0$
Time step	0.001

Results

The critical thickness to radius ratios h/r, computed for the expected mode-I and mode-II failures have been summarised in the table below. The number of blocks was 27, similarly like in [**74**]. Taking the mentioned modelling differences, it can be concluded that the results obtained with *Solfec* remain within an acceptable margin of accuracy.

	Target	*Solfec*	Ratio
Critical ratio h/r, $\mu = 0.4$	0.1070	0.1082	1.011
Critical ratio h/r, $\mu = 0.311$	0.1955	0.1965	1.005

Figures 13.4.20 and 13.4.21 illustrate the kinetic energy histories corresponding to the values reported in the table. The initial growth of the energy results from the fact, the contact forces are all zero at $t = 0$. Hence, the structure undergoes a dynamic process, purposely started in the vicinity of a steady state solution. The slight overestimation of the critical thickness results in part from the inertial effects related to the dynamic process. Also, as the element to element contact model is used, the locations of contact forces are shifted away by a small distance from the external surfaces of the arch. This decreases the effective thickness, and has an additional influence on the overestimation of the critical ratio.

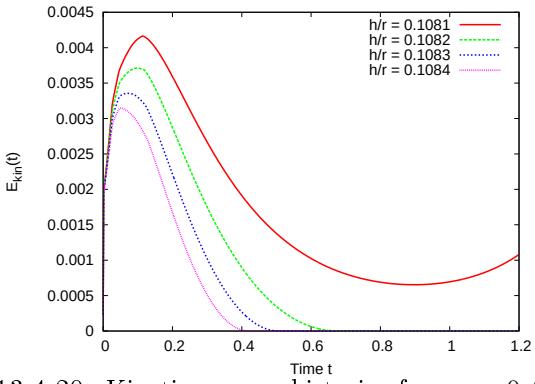


FIGURE 13.4.20. Kinetic energy histories for $\mu = 0.4$ and four different ratios h/r.

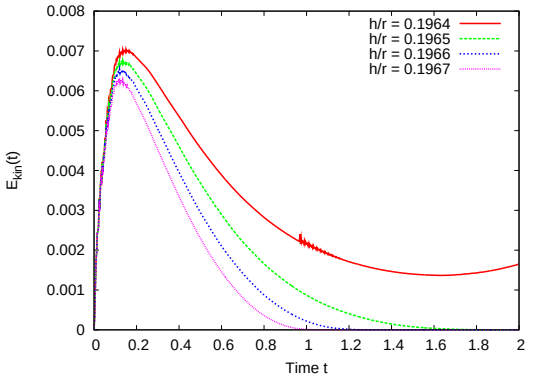


FIGURE 13.4.21. Kinetic energy histories for $\mu = 0.311$ and four different ratios h/r.

Figure 13.4.22 illustrates the results computed for an arch comprising 15 blocks. A friction-cohesion map of critical thickness values $h\,(\mu, c)$ was obtained on a 10×10 grid of $\mu \times c$, that is friction×cohesion. It is in the first place clear, that the three failure modes reported in Gilbert *et al.* [**74**] have been well reproduced for the zero cohesion case. The influence of cohesion results in a decrease of the critical arch thickness.

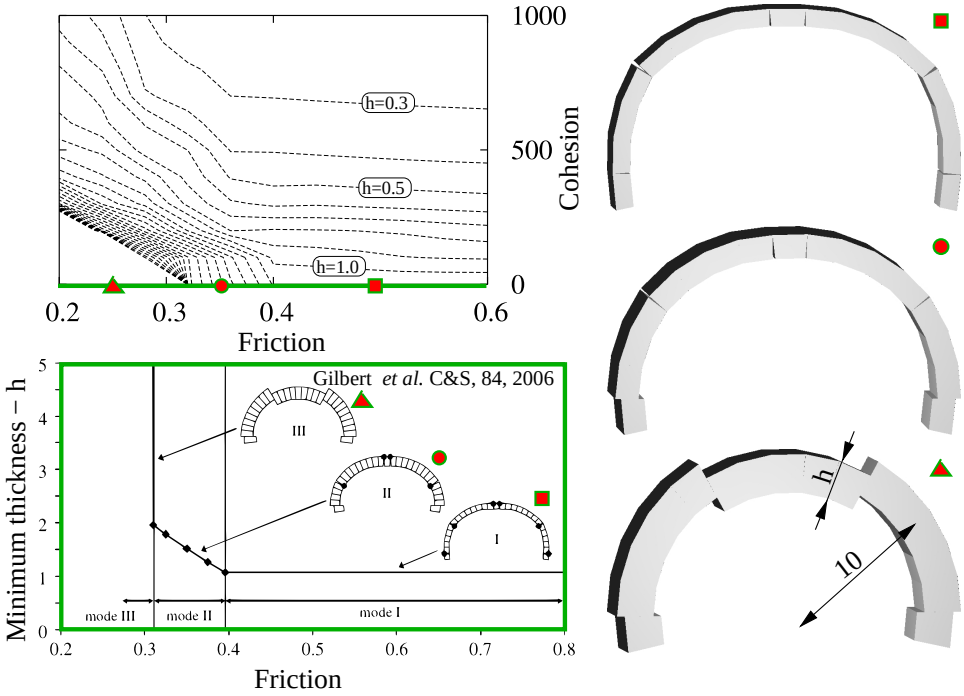


FIGURE 13.4.22. Friction-cohesion map of the critical arch thickness h and the three characteristic failure modes for the zero cohesion case.

13.4.8. Box-kite push until lockup.

Reference: Reports C6508/TR/0006 and 5014549/06/34/0 provided by Atkins.
Summary: Two layers of flat, nonconvex, acrylic bricks are fitted into a 3×3 pattern. The middle bricks are cracked and oriented at various angles. Shear and separation loads are applied to the top brick halves. The relative shear and separation displacements at lockup are reported.
Kinematics/Analysis/Solver: Rigid/Dynamic/Gauss-Seidel

Acrylic bricks were assembled into a 3×3 two-layer pattern embraced by a wooden frame (Figure 13.4.23). The middle two bricks were cracked independently at various angles (Figure 13.4.24). A hand load was applied to the two top brick halves and the maximal lockup displacements were reported. A model of the box-kite prepared in *Solfec* was used to cross-examine an FEM model used by Atkins. The mechanical model comprised:

- ideally plastic impacts (in order to approximate quasi-static conditions of the experiment)

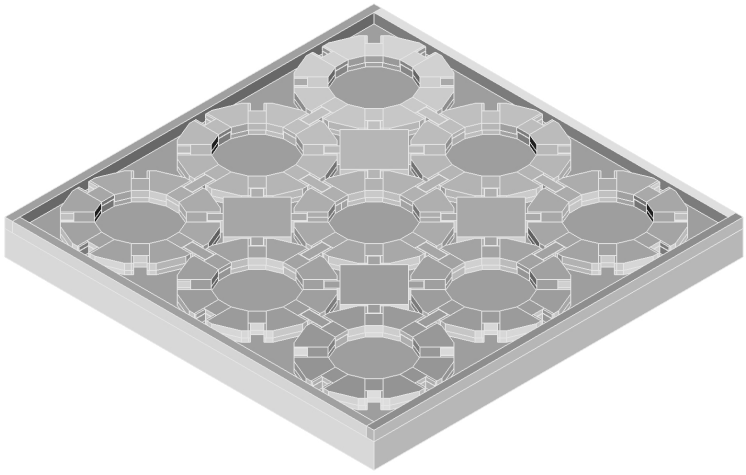


FIGURE 13.4.23. The box-kite assembly of bricks as modelled in *Solfec*.

- boundary conditions induced solely by contact (no explicit restrictions on displacements and rotations)
- shear and separation loads applied directly to the mass centres of the two top brick halves (no load-induced rotation)

The difficulty in reproducing the experimental results was twofold:

(1) The force was manually applied during the experiment, an exact manner of which was unknown.
(2) The way in which the shear and separation displacements were measured was also unknown.

The first difficulty was resolved by applying the force to the mass centres of the two top brick halves. This is equivalent to any force system whose resultant torque is zero and hence inducing only a linear motion. Any rotations happen solely due to the contact interactions. The second difficulty has been approached by measuring the relative displacement for a variety of control points. As illustrated in Figure 13.4.25, the strategy is to pick two arbitrary points A and B and allow them be convected by the motion of the respective top brick halves. The relative displacement is measured along the fixed directions of the action of the applied forces. Only one set of results, corresponding to the selection of mass centres as the control points is summarised further.

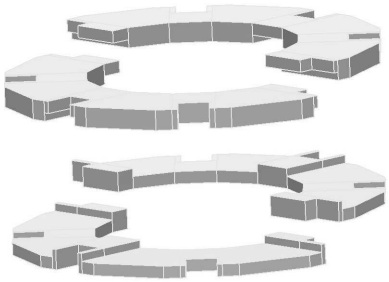


FIGURE 13.4.24. Example of cracked middle bricks from the top and bottom layers.

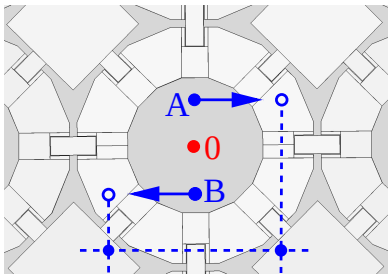


FIGURE 13.4.25. The two top brick halves and an exemplary shear displacement measurement.

Input parameters

Mass density (kg/m^3)	$\rho \in \{10, 1150\}$
Initial velocities (m/s)	all zero
Gravity acceleration (m/s^2)	$\mathbf{g} = [0, 0, -10]$
Velocity restitution	$\eta = 0$
Time step	$h = 0.001$
Friction	$\mu \in \{0, 0.3, 0.5, 0.8\}$

The smaller mass density $\rho = 10$ is used in frictionless calculations (this irrelevant from the results standpoint, but it speeds up solution for contact reactions). When the effect of friction is investigated, the density $\rho = 1150$ typical for the acrylic glass is assumed (we wish those results to be easier to imagine).

Results

Figure 13.4.26 summarises the initial set of contacts. There are no horizontal normals in the figure, because all of the bricks are separated by a small clearance. In the experiment, two clearance sizes were considered. Without getting into details, these will be further called the *large* and the *small* clearance. Various orientations of crack angles correspond to different test cases, specifically numbered in the referenced reports. As there would be not much gain from specifying the angles, without giving other detailed information, we do not attempt to do that. It is enough to say that the numbering convention is of the kind $31N$ or $31T$, where the N and T letters correspond to the separation and shear tests. The current example should then be regarded only as a qualitative demonstration of the computational framework.

Figures 13.4.27 and 13.4.28 compare the experimental, FEM (Atkins) and *Solfec* results. Both, in *Solfec* and FEM computations zero friction was assumed. Two largest discrepancies happen for cases 31 and 44. Case 31 undergoes a complete separation. Case 44 opens too wide in shear. Similarly, for the small clearance, case 48 opens too wide in separation, while case 61 opens too wide in shear. In the remaining cases we are somewhat closer to the experiment, when compared with FEM (small clearance, Figure 13.4.28).

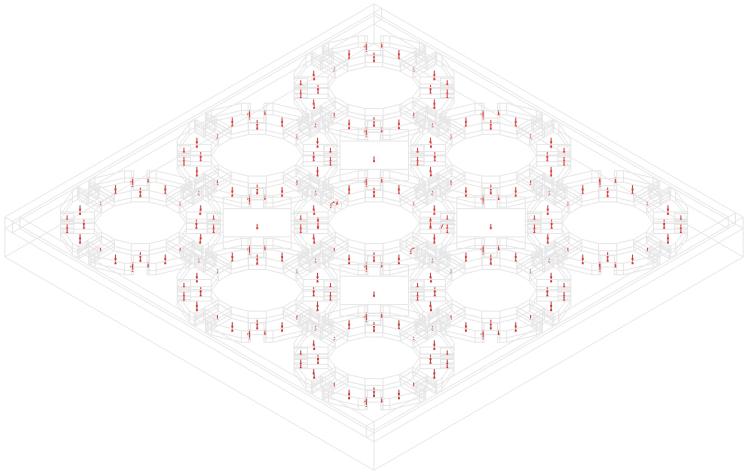


FIGURE 13.4.26. Contacts detected after the first step of the time stepping.

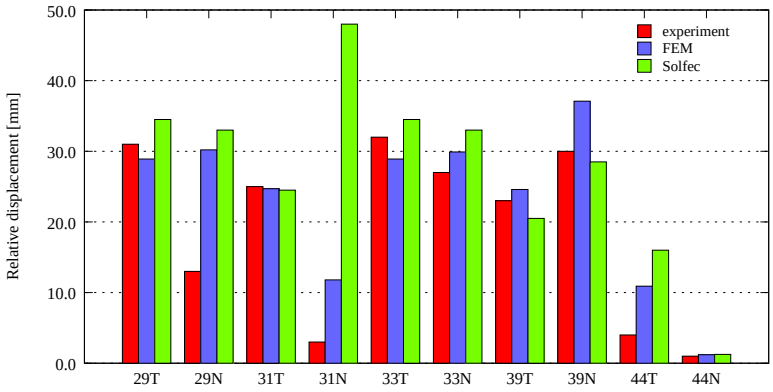


FIGURE 13.4.27. Large clearance. Experiment, FEM and *Solfec*.

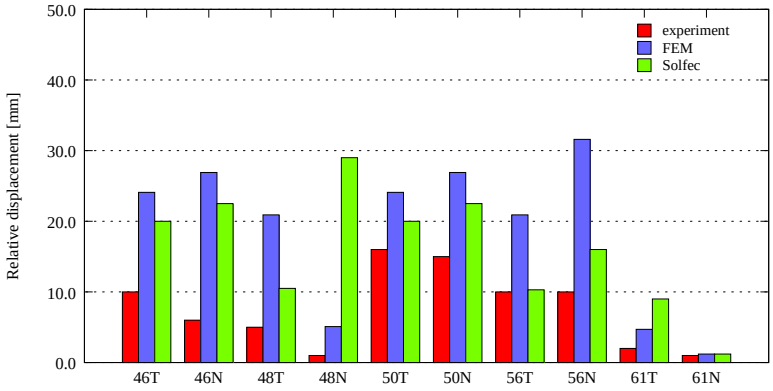


FIGURE 13.4.28. Small clearance. Experiment, FEM and *Solfec*.

In order to verify the role of friction, the case $31N$ has been given a closer look. The assumed material parameters were $\rho = 1150$ for the mass density and $\mu \in \{0.0, 0.3, 0.5, 0.8\}$ for friction. Case 31 separates fully in the frictionless case, and the purpose here is to investigate whether frictional effects can affect this result (which might have happened during the experiment). The load of value $150N$ is ramped over the time interval $[0, 1, 2]$ (Figure 13.4.29). Separation is large, although the effect of friction is clear. The increased load of $250N$ was again applied the time interval $[0, 1, 2]$. This corresponds to lifting up $50kg$, although here the left and the right hand apply the load in opposite directions. Figure 13.4.30 shows that the separation is now much closer to the frictionless case. Nevertheless, the effect of friction is still visible.

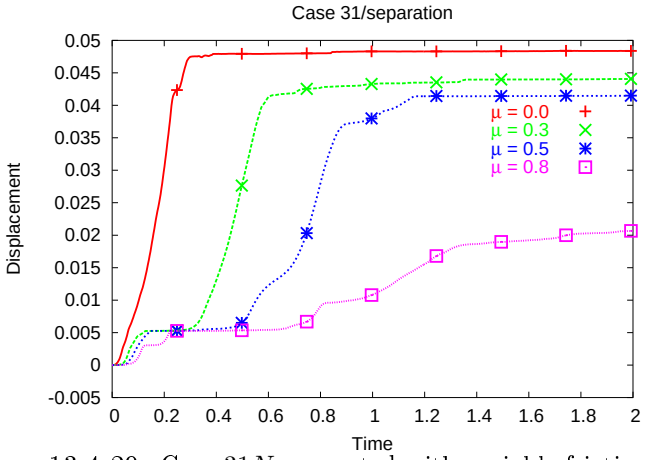


FIGURE 13.4.29. Case $31N$ computed with variable friction coefficient and ramped load of $150N$ ramped over $[0, 1, 2]$ seconds.

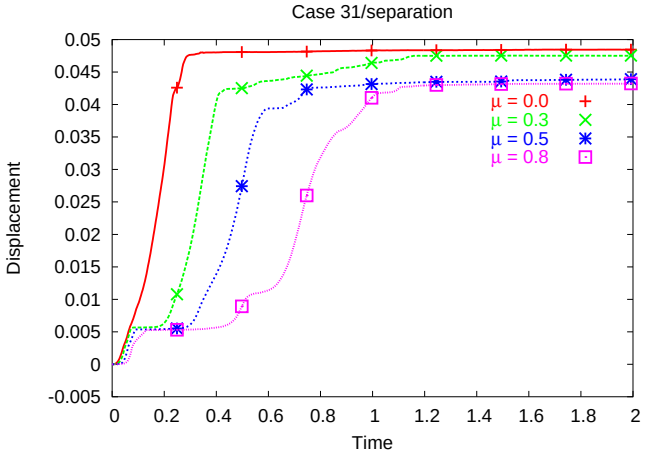


FIGURE 13.4.30. Case $31N$ computed with variable friction coefficient and ramped load of $250N$ ramped over $[0, 1, 2]$ seconds.

13.4.9. Lourenco's wall.

Reference: Lourenco, P. B. and Oliveira, D. V. and Roca, P. and Orduna, A., Dry joint stone masonry walls subjected to in-plane combined loading, Journal of Structural Engineering, vol. 131, pp. 1665-1673, 2005.
Summary: A dry masonry wall undergoes a combined loading. After an initial phase of vertical loading, a horizontal loading is applied and the load-displacement path is recorded.
Kinematics/Analysis/Solver: Pseudo-Rigid/Quasi-Static/Hybrid-Newton

The quasi-static time stepping is verified against the experimental data by Lourenco *et al.* [**142**]. A series of dry joint stone planar masonry wall tests were performed under combined loading. The scheme of the experimental setup is presented in Figure 13.4.31. The wall is first loaded with the vertical force, followed by a displacement controlled horizontal loading. Plots of the horizontal displacement versus the horizontal force were obtained under constant vertical loading of 30kN. In experiments, a high strength mortar was used on the upper and lower layers of stones in order to correct roughness of contact surfaces. Due to the existence of the rigid obstacle in the lower left corner, no cohesion at the lower layer was assumed in the numerical model. At the upper layer, a small value of cohesion of $c = 0.3$MPa was assumed.

Two cases of the load control (5N/s and 1N/s) and two cases of the displacement control were computed (0.1mm/s and 0.02mm/s). In case of the load control the reported horizontal force is the sum of frictional contact forces acting on the lower surface of the concrete slab, while the displacement is measured at the centre point of the surface. The displacement control was obtained by placing a dummy contact point where the horizontal force should be applied. At this point a prescribed velocity was applied and the resulting contact force and displacement were reported. The following scaling of the control point velocity was used $s(t) = 1e{-}5t/(1{+}1e{-}5t)$ in order to obtain smooth transition from the initial state. Thus as a result, the control velocity was growing with time according to the formula $v_{horizontal}(t) = v_i s(t)$, where $v_i \in \{0.1\text{mm/s}, 0.02\text{mm/s}\}$. This transition proved to be necessary in order to avoid abrupt changes of solution at the initial stage of displacement loading.

Input parameters

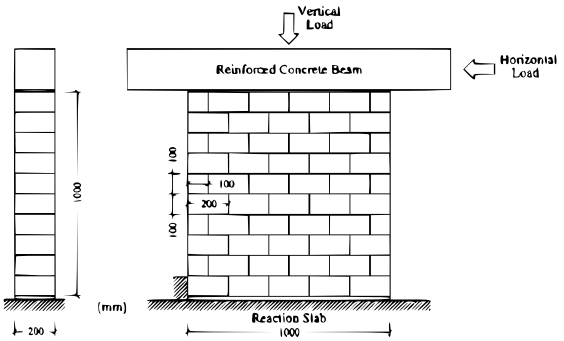


FIGURE 13.4.31. Wall geometry and loading (Lourenco *et al.* [**142**]).

Young's modulus (GPa)	$E = 15.5$
Poisson's ratio	$\nu = 0.2$
Coulomb friction	$\mu = 0.62$
Cohesion (MPa)	$c = 0.3$
Time step (s)	$h = 1$
Mass scaling	$\lambda_{max} h = 4$
Dynamic relaxation termination ratio	$r = 0.1$
Dynamic relaxation iterations bound	$K = 100$
Load control velocities (N/s)	$\{5, 1\}$
Displacement control velocities (mm/s)	$\{0.1, 0.02\}$
Vertical load (kN)	30
Maximum stepwise displacement (mm)	$\delta l_{max} = 1$

Results

Firgure 13.4.32 shows the maxium compressive components of Cauchy stress for the horizontal displacement of 15mm. At this stage a damage mechanism was fully formed in the experimental setup. It can be seen that numerical simulations are capable of reproducing the characteristic shear and rocking failure, for which the lower triangular part of the wall is unloading (subjected only to the gravitational loading). Sensitivity of the results with respect to the control mechanism is visible, as the range of compressive stresses differs for the force and the displacement controlled cases.

Poor performance of the pseudo-rigid bodies in the elastic part of the displacement-force graphs (Figures 13.4.33, 13.4.34) is no surprise. Assumption of uniform deformations results in a very stiff behaviour, and this cannot be helped without a higher order kinematics. The nonlinear part of graphs displays clearly a rate-dependence of the numerical model. While this is in some accordance with the physical reality and numerically corresponds to the inertial terms being involved in the transfer of contact forces, no rate-dependent components exist in the underlying formulation. For the displacement control case this can be explained by the

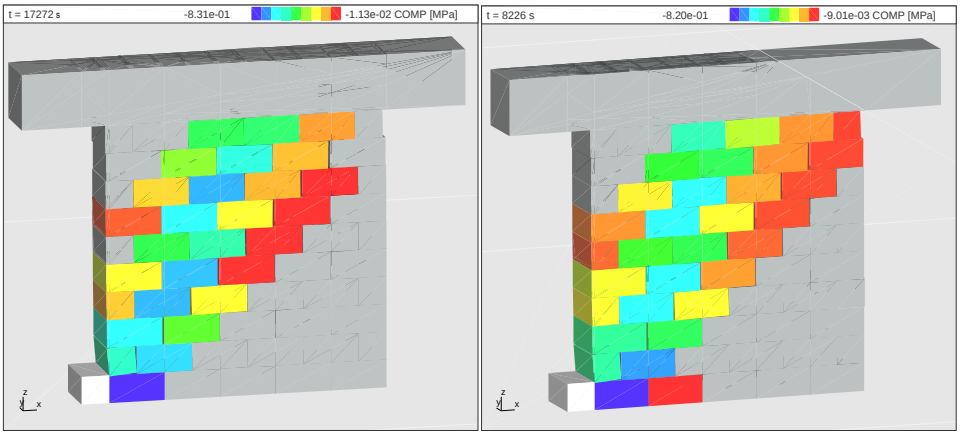


FIGURE 13.4.32. Maximum compressive Cauchy stress for horizontal displacement equal $15mm$. On the left the force control was applied at a rate $1N/s$. On the right displacement control was applied at a rate $0.02mm/s$. Bricks below the threshold of 1% of the maximum compressive stress value are not coloured.

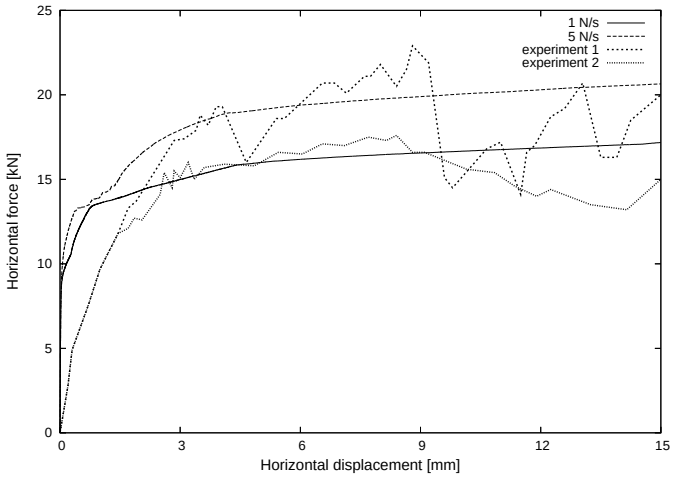


FIGURE 13.4.33. Force controlled horizontal displacement versus horizontal force load paths.

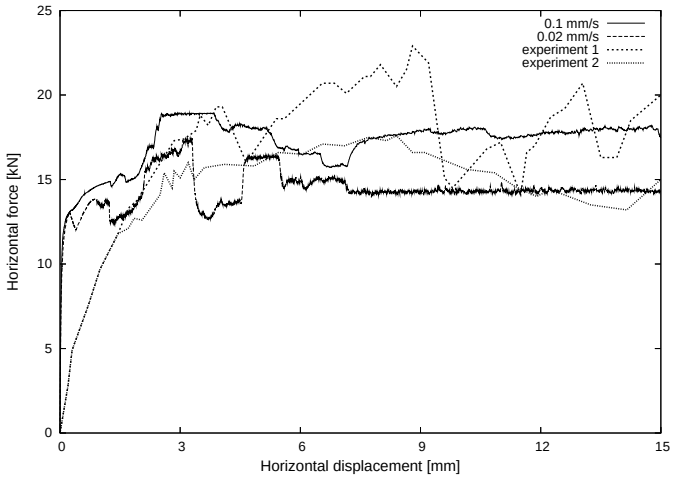


FIGURE 13.4.34. Displacement controlled horizontal displacement versus horizontal force load paths.

fact, that assuming a specific velocity at a control point enforces a specific distance which the point travels across the time step. With a fixed time step, a large enforced displacement results in a large value of the control reaction, which instantaneously propagates through the structure, rendering its answer stiffer. In the case of the load control a larger stepwise increment of the horizontal force results in higher velocity, which propagates instantaneously with the same effect. It is the question for a further research, whether and how this kind of effect can be excluded in the context of a non-regularised quasi-static multi-body formulation.

CHAPTER 14

Conclusions

There is a number of issues which need to be addressed, in order to complete the presented framework:

(1) Energetic inconsistency, pointed out in Section 10.5, needs to be resolved. Only the ideally plastic impact model can be applied with some confidence, whenever such simplification is acceptable. This was the case in Section 13.4.8. Most conveniently, for deformable kinematics, one can use the average velocity $\bar{\mathbf{U}} = \frac{1}{2}(\mathbf{U}^+ + \mathbf{U}^-)$ in the formulation of frictional contact constraints. For rigid bodies however, a more versatile solution is needed. Perhaps, it will be necessary to use a two-phase approach, combined with Poisson's impact model, as it was done in [**8, 70**].

(2) In the context of rigid kinematics, a rigorous treatment of multiple impacts has to be worked out. The lack of a rational incorporation of the shock propagation effects represents a serious drawback. This is still an active research topic. Recent development by Liu *et al.* [**140**] deals with the frictionless case and seems to be a good starting point in this respect.

(3) The hybrid Newton technique from Section 11.2 needs to be extended in order to cope with singular problems. Only then it can become useful in the context of rigid kinematics. Apart from the rigid case, of equal importance is an inclusion of the finite element discretised kinematics. It remains a matter of future research to investigate whether the proposed hybrid linearisation performs for these classical approaches as well as it does in the pseudo-rigid setting.

(4) Convergence of the complete time-stepping remains to be shown. Quite likely, on the way towards such a proof, some changes to the overall design will be necessary. However, this should not hinder the practical utility of the numerical tool already at hand.

(5) Theoretical estimates of complexity of the dynamic rectangle structure from Section 9.3.3.3 need to be experimentally verified. Also in the context of contact detection, implementation of the fast intersection Algorithm 9.4.3 needs to be completed and compared against the simpler approach from Algorithm 9.4.1. For the moment, only Algorithm 9.4.1 was employed in all of the presented examples involving contact.

(6) On the presentation side, it would be useful to draw a link between the equality form of contact and friction constraints and the augmented Lagrangian method by Hestenes, Powell and Rockafellar [**91, 172, 181**]. This would shed additional light on the origins of the predictor $\mathbf{d} = \mathbf{R} - \rho\bar{\mathbf{U}}$.

The pseudo-rigid continuum model by Cohen and Muncaster [**46**] was exemplified only in the context of quasi-statics. Integration of an unconstrained dynamic motion merely confirms conservation properties of the time stepping scheme (5.1.1-5.1.3). While the single impact behaviour was already studied in [**193, 113, 112**], it might be interesting to investigate application of the pseudo-rigid model as a simple workaround to the lack of a practical multiple-impact resolution for rigid

kinematics. Some early dynamic examples were given in [**121**]. The practical limitation is in the necessity of using an extremely small time step, for realistic values of the material parameters. This, combined with the need for the solution of an implicit nonlinear problem at every time step, renders this approach rather unfeasible for large and dense multi-body problems. On the other hand, only for such problems the simplified deformability can be eventually accepted. An interesting improvement here would be to time-homogenise contact variables, and hence solve the frictional contact problem only every n steps. In the quasi-static context, the pseudo-rigid model proved useful and allowed to test contact solvers on the prototype of a finite-kinematics, multi-body framework (cf. Section 13.3). From this point of view, the model can be regarded as a good stress post-processor, although its elastic response is too stiff (cf. Section 13.4.9). In practise, it might be more convenient to use few finite elements instead of a single pseudo-rigid body - especially in the situations, where large rotations are not essential.

The hybrid Newton solver from Section 11.2 shows promise in dealing with the frictional contact problem. Apart from the already mentioned refinement, facilitating application to over-determined systems, one can also think about a parallel implementation of this approach. A direct linear solver could be replaced by an iterative one, preconditioned with positive-definite tangents resulting from the Tresca formulation. An implementation of the framework presented here has already been partly parallelised [**122**]. Nevertheless, this effort stumbled on the difficulty with an effective, distributed memory implementation of the Gauss-Seidel solver. This motivated developments of Section 11.2.

As a more accomplished fact, one should mention the time stepping schemes from Section 5.2.2. NEW2 and NEW3 do have some good properties. For several reasons NEW2 appears to be well suited for the short to moderate term analysis of constrained systems. As it was shown, the exact conservation of the angular momentum may occur necessary in order to maintain accuracy (Example 13.1.3). At the same time, the amount of the energy loss is often acceptable for the incremental rotations of magnitudes dictated by an accurate integration of the constrained motion. Additionally, the dissipative behaviour of NEW2 seems advantageous in the context of an explicit multi-body contact analysis, where the episodes of excessively high contact reactions should not render the analysis unstable. For longer term analysis or for the cases where a higher accuracy is required, NEW3 comes quite handy, with only a moderate increase of the computational cost and still offering all of the advantages of NEW2.

Bibliography

[1] http://planetmath.org/encyclopedia/SpecialOrthogonalGroup.html.

[2] *LS-DYNA Theory Manual*. LSTC, 1998.

[3] V. Acary and M. Jean. Numerical modeling of three dimensional divided structures by the non smooth contact dynamics method. In B. H. V. Topping, editor, *The Fifth International Conference on Computational Structures Technology*, pages 211–222. Civil-Comp Press, Edimburgh, 2000.

[4] Pankaj K. Agarwal, Mark de Berg, Joachim Gudmundsson, Mikael Hammar, and Herman J. Haverkort. Box-trees and r-trees with near-optimal query time. In *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry*, pages 124–133, New York, NY, USA, 2001. ACM.

[5] M. Ainsworth and L. A. Mihai. Modeling and numerical analysis of masonry structures. *Numer. Methods Partial Differential Equations*, 23(4):798–816, 2007.

[6] Mark Ainsworth and L. Angela Mihai. A comparison of solvers for linear complementarity problems arising from large-scale masonry structures. *Appl. Math.*, 51(2):93–128, 2006.

[7] P. Alart and A. Curnier. Mixed formulation for frictional contact problems prone to Newton like solution methods. *Computer Methods in Applied Mechanics and Engineering*, 92:353–375, 1991.

[8] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14:231–247, November 1997.

[9] M. Anitescu and F.A. Potra. A time-stepping method for stiff multibody dynamics with contact and friction. *International J. Numer. Methods Engineering*, 55(7):753–784, 2002.

[10] Mihai Anitescu, Florian A. Potra, and David E. Stewart. Time-stepping for three-dimensional rigid body dynamics. *Computer Methods in Applied Mechanics and Engineering*, 177:183–197, July 1999.

[11] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

[12] V. I. Arnold. *Metody matematyczne mechaniki klasycznej (Mathematical methods of classical mechanics)*. Państwowe Wydawnictwo Naukowe, Warszawa, 1981.

[13] V. I. Arnold. *Mathematical Methods of Classical Mechanics*, volume 60 of *Graduate Texts in Mathematics*. Springer Verlag, New York, 2^{nd} edition, 1989. 508 pages.

[14] W. I. Arnold. *Równania różniczkowe zwyczajne (Ordinary differential equations)*. Państwowe Wydawnictwo Naukowe, Warszawa, 1975.

[15] S. W. Attaway, B. A. Hendrickson, S. J. Plimpton, D. R. Gardner, C. T. Vaughan, K. H. Brown, and M. W. Heinstein. A parallel contact detection algorithm for transient solid dynamics simulations using pront03d. *Computational Mechanics*, 22:143–159, 1998.

[16] David Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics (ACM)*, 24:19–28, 1990.

[17] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[18] J. P. Bardet and J. Proubet. Adaptative dynamic relaxation for statics of granular materials. *Computers & Structures*, 39:221–229, 1991.

[19] P. Barral, C. Moreno, P. Quintela, and M. T. Sánchez. A numerical algorithm for a signorini problem associated with maxwell-norton materials by using generalized newton's methods. *Computer Methods in Applied Mechanics and Engineering*, 195:880–904, February 2006.

[20] Sebastiano Battiato, Domenico Cantone, Dario Catalano, Gianluca Cincotti, and Micha Hofri. An efficient algorithm for the approximate median selection problem. *Lecture Notes in Computer Science*, 1767:226–238, 2000.

[21] Rudolf Bayer. Symmetric binary b-trees: Data structure and maintenance algorithms. *Acta Informatica*, 1:290–306, December 1972.

[22] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. J. Wiley and Sons, New York, 2000.

[23] D. J. Benson and J. O. Hallquist. A simple rigid body algorithm for structural dynamics programs. *International Journal for Numerical Methods in Engineering*, 22:723–749, 1986.

[24] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, 28(9):643–647, 1979.

[25] J. L. Bentley and D. Wood. An optimal worst case algorithm for reporting intersections of rectangles. *IEEE Trans. Comput.*, 29(7):571–577, 1980.

[26] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[27] Jon Louis Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980.

[28] M. J. Berger and S. H. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Trans. Comput.*, 36(5):570–580, 1987.

[29] Binay K. Bhattacharya and Sandeep Sen. On a simple, practical, optimal, output-sensitive randomized planar convex hull algorithm. *J. Algorithms*, 25(1):177–193, 1997.

[30] P. Bisegna, F. Lebon, and F. Maceri. Relaxation procedures for solving signorini-coulomb contact problems. *Advances in Engineering Software*, 35:595–600, 00 2004.

[31] John W. Boyse. Interference detection among solids and surfaces. *Commun. ACM*, 22(1):3–9, 1979.

[32] B Brogliato, AA ten Dam, L Paoli, F Genot, and M Abadie. Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. *Applied Mechanics Reviews*, 55(2):107–150, 2002.

[33] Herman C. J. Bruneel and Igor De Rycke. Quicktrace: a fast algorithm to detect contact. *International Journal for Numerical Methods in Engineering*, 54:299–316, 2002.

[34] A. Bykat. Convex hull of a finite set of points in two dimensions. *Information Processing Letters*, 7:296–298, 1978.

[35] Stephen Cameron. Enhancing gjk: Computing minimum and penetration distance between convex polyhedra. In *Proceedings of International Conference on Robotics and Automation*, pages 3112–3117, 1997.

[36] J. Casey. Pseudo-rigid continua: Basic theory and a geometrical derivation of lagrange's equations. *Proceedings of the Royal Society - Mathematical, Physical and Engineering Sciences (Series A)*, 460:2021–2049, 2004.

[37] J. Casey. The ideal pseudo-rigid continuum. *Proceedings of the Royal Society - Mathematical, Physical and Engineering Sciences (Series A)*, 462(2074):3185–3196, 2006.

[38] N. Chakraborty, Jufeng Peng, S. Akella, and J. Mitchell. Proximity queries between convex objects: an interior point approach for implicit surfaces. In *Robotics and Automation*, pages 1910 – 1916, Orlando, FL, 2006.

[39] Xiaojun Chen, Zuhair Nashed, and Liqun Qi. Smoothing methods and semismooth methods for nondifferentiable operator equations. *SIAM Journal on Numerical Analysis*, 38(4):1200–1216, 2000.

[40] B. Chetouane, F. Dubois, M. Vinches, and C. Bohatier. Nscd discrete element method for modelling masonry structures. *International Journal for Numerical Methods in Engineering*, 64:65–94, 2005.

[41] P. W. Christensen, A. Klarbring, J. S. Pang, and N. Strömberg. Formulation and comparison of algorithms for frictional contact problems. *International Journal for Numerical Methods in Engineering*, 42:145–173, 1998.

[42] Christoph Glocker. On Frictionless Impact Models in Rigid-Body Systems. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 359(1789):2385–2404, 2001.

[43] Frank H Clarke. *Optimization and Nonsmooth Analysis*. John Wiley and Sons, New York, 1983.

[44] Kenneth L. Clarkson, David Eppstein, Gary L. Miller, Carl Sturtivant, and Shang-Hua Teng. Approximating center points with iterative Radon points. *Int. J. Comput. Geometry Appl*, 6(3):357–377, 1996.

[45] H. Cohen and G. P. Mac Sithigh. Collisions of pseudo-rigid bodies: A brach-type treatment. *International Journal of Engineering Science*, 34:249–256, 1996.

[46] H. Cohen and R. G. Muncaster. *The Theory of Pseudo-rigid Bodies*. Springer, New York, 1988.

[47] H. Cohen and G. P. Mac Sithigh. Impulsive motions of elastic pseudo-rigid bodies. *Journal of Applied Mechanics*, 58(4):1042–1048, 1991.

[48] Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav Ponamgi. I-collide: an interactive and exact collision detection system for large-scale environments. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 189–ff., New York, NY, USA, 1995. ACM.

[49] Daniel S. Coming and Oliver G. Staadt. Kinetic sweep and prune for multi-body continuous motion. *Computers and Graphics*, 30:439–449, June 2006.

[50] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[51] R. K. Culley and K. G. Kempf. Collision detection algorithm based on velocity and distance bounds. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1064–1069, San Francisco, 1986.

[52] J. S. Dai. An historical review of the theoretical development of rigid body displacements from rodrigues parameters to the finite twist. *Mechanism and Machine Theory*, 41:41–52, 2006.

[53] Y. H. Dai. On the nonmonotone line search. *Journal of Optimization Theory and Applications*, 112:315–330, February 2002.

[54] Mark de Berg, Matthew Katz, A. Frank van der Stappen, and Jules Vleugels. Realistic input models for geometric algorithms. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 294–303, New York, NY, USA, 1997. ACM.

[55] G. De Saxcé and Z. Q. Feng. The bipotential method: a constructive approach to design the complete contact law with friction and improved numerical algorithms. *Mathematical and Computer Modelling*, 28:225–245, 1998.

[56] G. De Saxe and Z.Q. Feng. New inequality and functional for contact with friction: the implicit standard material approach. *Mech. Struct. and Mach.*, 19(3):301–325, 1991.

[57] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[58] Ralf Diekmann, Jan Hungershöfer, Michael Lux, Lars Taenzer, and Jens-Michael Wierum. Using space filling curves for efficient contact searching. In *Proc. IMACS*, 2000.

[59] Katrin Dobrindt, Kurt Mehlhorn, and Mariette Yvinec. A complete and efficient algorithm for the intersection of a general and a convex polyhedron. In *WADS '93: Proceedings of the Third Workshop on Algorithms and Data Structures*, pages 314–324, London, UK, 1993. Springer-Verlag.

[60] Zdenek Dostál, Jaroslav Haslinger, and Radek Kucera. Implementation of the fixed point method in contact problems with coulomb friction based on a dual splitting type technique. *Journal of Computational and Applied Mathematics*, 140:245–256, March 2002.

[61] Eva Dyllong and Wolfram Luther. The gjk distance algorithm: An interval version for incremental motions. *Numerical Algorithms*, 37:127–136, December 2004.

[62] William F. Eddy. A new convex hull algorithm for planar sets. *ACM Trans. Math. Softw.*, 3(4):398–403, 1977.

[63] H Edelsbrunner and L J Guibas. Topologically sweeping an arrangement. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 389–403, New York, NY, USA, 1986. ACM.

[64] H. Edelsbrunner and H. A. Maurer. On the intersection of orthogonal objects. *Information Processing Letters*, 13:177–181, 1981.

[65] Herbert Edelsbrunner. A new approach to rectangle intersections part i. *International Journal of Computer Mathematics*, 13:209–219, 1983.

[66] Herbert Edelsbrunner. A new approach to rectangle intersections part ii. *International Journal of Computer Mathematics*, 13:221–229, 1983.

[67] Herbert Edelsbrunner and Mark H. Overmars. On the equivalence of some rectangle problems. *Information Processing Letters*, 14:124–127, May 1982.

[68] Y. T. Feng and D. R. J. Owen. An augmented spatial digital tree algorithm for contact detection in computational mechanics. *International Journal for Numerical Methods in Engineering*, 55:159–176, 2002.

[69] M. Ferris and S. Lucidi. Nonmonotone stabilization methods for nonlinear equations. *Journal of Optimization Theory and Applications*, 81:53–71, 1994.

[70] Mihai Anitescu Bogdan Gavrea Jeff Trinkle Florian A. Potra. A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints, and friction. *International Journal for Numerical Methods in Engineering*, 66:1079–1124, 2006.

[71] C. Funfzig, T. Ullrich, and D. W. Fellner. Hierarchical spherical distance fields for collision detection. *IEEE Computer Graphics and Applications*, 26:64–74, 2006.

[72] Alejandro Garcia-Alonso, Nicolás Serrano, and Juan Flaquer. Solving the collision detection problem. *IEEE Comput. Graph. Appl.*, 14(3):36–43, 1994.

[73] Elmer G. Gilbert, Daniel W. Johnson, and S. Sathiya Keerthi. Fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE journal of robotics and automation*, 4:193–203, 1988.

[74] M. Gilbert, C. Casapulla, and H. M. Ahmed. Limit analysis of masonry block structures with non-associative frictional joints using linear programming. *Computers and Structures*, 84:873–887, 2006.

[75] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180, New York, NY, USA, 1996. ACM.

[76] Naga K. Govindaraju, Stephane Redon, Ming C. Lin, and Dinesh Manocha. Cullide: interactive collision detection between complex models in large environments using graphics hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 25–32, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[77] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Limit surface and moment function descriptions of planar sliding. In *Proc. of the 1989 IEEE Intl. Conf. on Robot. and Automat*, pages 794–799, Los Alamitos, CA, 1989. IEEE.

[78] L Grippo, F Lampariello, and S Lucidi. A nonmonotone line search technique for Newton's method. *SIAM J. Numer. Anal.*, 23(4):707–716, 1986.

[79] B. Grünbaum. Measures of symmetry for convex sets. In *Proc. Sympos. Pure Math.*, volume 7, pages 233–270, Amer. Math. Soc., 1963.

[80] Leonidas J. Guibas and Robert Sedgewick. A dichromatic framework for balanced trees. In *FOCS*, pages 8–21, Ann Arbor, MI, USA, 1978.

[81] K. Han, Y. T. Feng, and D. R. J. Owen. Polygon-based contact resolution for superquadrics. *International Journal for Numerical Methods in Engineering*, 66:485–501, 2006.

[82] Shih-Ping Han, Jong-Shi Pang, and Narayan Rangaraj. Globally convergent newton methods for nonsmooth equations. *Mathematics of Operations Research*, 17(3):586–607, 1992.

[83] Jaroslav Haslinger, Zdenek Dostál, and Radek Kucera. On a splitting type algorithm for the numerical realization of contact problems with coulomb friction. *Computer Methods in Applied Mechanics and Engineering*, 191:2261–2281, March 2002.

[84] V. Hayward. Fast collision detection scheme by recursive decomposition of a manipulator workspace. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1044–1049, San Francisco, CA, 1986. IEEE.

[85] J. Heegaard and A. Curnier. An augmented Lagrangian method for discrete large slip contact problems. *Int. J. for Numerical Meth. in Engng.*, 36(4):569–593, 1993.

[86] A. Heege and P. Alart. A frictional contact element for strongly curved contact problems. *Int. J. Numer. Meth. Eng.*, 39:165–184, 1996.

[87] M. W. Heinstein and T. A. Laursen. An algorithm for the matrix-free solution of quasistatic frictional contact problems. *International Journal for Numerical Methods in Engineering*, 44:1205–1226, 1999.

[88] M.W. Heinstein, S.W. Attaway, J.W. Swegle, and F.J. Mello. A general purpose contact detection algorithm for nonlinear structural analysis codes. Technical Report SAND92-2141, Sandia National Laboratories, Albuquerque, NM, 1993.

[89] Martin Herman. Fast, three-dimensional, collision-free motion planning. In *Robotics and Automation*, volume 2, pages 1056–1063. IEEE, 1986.

[90] Stefan Hertel, Martti Mäntylä, Kurt Mehlhorn, and Jurg Nievergelt. Space sweep solves intersection of convex polyhedra. *Acta Informatica*, 21:501–519, December 1984.

[91] Magnus R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320, MM 1969.

[92] Jacques Heyman. *Equilibrium of Shell Structures*. Clarendon Press, Oxford, England, 1977.

[93] M. Hintermüller, K. Ito, and K. Kunisch. The primal-dual active set strategy as a semismooth Newton method. *SIAM Journal on Optimization*, 13:865–888, 2003.

[94] M. Hintermüller, V.A. Kovtunenko, and K. Kunisch. The primal-dual active set method for a crack problem with non-penetration. *IMA J Appl Math*, 69(1):1–26, 2004.

[95] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. In *Acm transactions on graphics*, volume 15, pages 179–210, 1996.

[96] S. Hüeber, G. Stadler, and B. I. Wohlmuth. A primal-dual active set algorithm for three-dimensional contact problems with Coulomb friction. *SIAM Journal on Scientific Computing*, 30(2):572–596, 2007.

[97] S. Hüeber and B. I. Wohlmuth. A primal-dual active set strategy for non-linear multibody contact problems. *Computer Methods in Applied Mechanics and Engineering*, 194:3147–3166, 2005.

[98] T. J. R. Hughes. Analysis of Transient Algorithms with Particular Reference to Stability Behavior. In T. Belytschko and T.J.R. Hughes, editors, *Computational methods for transient analysis*, pages 67–155. Elsevier Science, 1983.

[99] Adnan Ibrahimbegović, François Frey, and Ivica Kožar. Computational aspects of vector-like parametrization of three-dimensional finite rotations. *International Journal for Numerical Methods in Engineering*, 38:3653–3673, 1995.

[100] K.K. Wong J. C. Simo. Unconditionally stable algorithms for rigid body dynamics that exactly preserve energy and momentum. *International Journal for Numerical Methods in Engineering*, 31:19–52, 1991.

[101] Doug L. James and Dinesh K. Pai. Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph.*, 23(3):393–398, 2004.

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

[103] M. Jean, V. Acary, and Y. Monerie. Non-smooth contact dynamics approach of cohesive materials. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences (Series A)*, 359:2497–2518, 2001.

[104] P. Jimenez, F. Thomas, and C. Torras. 3d collision detection: A survey. *Computers and Graphics (Pergamon)*, 25:269–285, 2001.

[105] P. Joli and Z.-Q. Feng. Uzawa and Newton algorithms to solve frictional contact problems within the bi-potential framework. *International Journal for Numerical Methods in Engineering*, 73:317–330, 2008.

[106] R.E. Jones and P. Papadopoulos. A Yield-limited Lagrange Multiplier Formulation for Frictional Contact. *Int. J. Num. Meth. Engrg.*, 48:1127–1149, 2000.

[107] Reese E. Jones and Panayiotis Papadopoulos. Simulating anisotropic frictional response using smoothly interpolated traction fields. *Computer Methods in Applied Mechanics and Engineering*, 195:588–613, January 2006.

[108] A. Joukhadar, A. Scheuer, and Ch. Laugier. Fast contact detection between moving deformable polyhedra. In *Ieee international conference on intelligent robots and systems*, volume 3, pages 1810–1815, 1999.

[109] F. Jourdan, P. Alart, and M. Jean. A Gauss-Seidel like algorithm to solve frictional contact problems. *Computer Methods in Applied Mechanics and Engineering*, 155:31–47, 1998.

[110] J. C. Chiou K. C. Park. A discrete momentum-conserving explicit algorithm for rigid body dynamics analysis. *International Journal for Numerical Methods in Engineering*, 36:1071–1083, 1993.

[111] C. Kane, E.A. Repetto, M. Ortiz, and J.E. Marsden. Finite element analysis of nonsmooth contact. *Computer Methods in Applied Mechanics and Engineering*, 180(1):1–26, 1999.

[112] E. Kanso and P. Papadopoulos. Dynamics of pseudo-rigid ball impact on rigid foundation. *International Journal of Non-Linear Mechanics*, 39:299–309, 2004.

[113] E. Kanso and P. Papadopoulos. Pseudo-rigid ball impact on an oscillating rigid foundation. *International Journal of Non-Linear Mechanics*, 39:1129–1145, 2004.

[114] D. M. Kaufman, T. Edmunds, and D. K. Pai. Fast frictional dynamics for rigid bodies. In *ACM Transactions on Graphics*, volume 24, pages 946–956, 2005.

[115] Ladislav Kavan, Ivana Kolingerova, and Jiri Zara. Fast approximation of convex hull. In *ACST'06: Proceedings of the 2nd IASTED international conference on Advances in computer science and technology*, pages 101–104, Anaheim, CA, USA, 2006. ACTA Press.

[116] Katsuaki Kawachi and Hiromasa Suzuki. Distance computation between non-convex polyhedra at short range based on discrete voronoi regions. In *GMP '00: Proceedings of the Geometric Modeling and Processing 2000*, page 123, Washington, DC, USA, 2000. IEEE Computer Society.

[117] Dong-Jin Kim, Leonidas J. Guibas, and Sung Yong Shin. Fast collision detection among multiple moving spheres. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):230–242, 1998.

[118] Y. Kitamura, A. Smith, H. Takemura, and F. Kishino. A real-time algorithm for accurate collision detection for deformable polyhedral objects. *Presence: Teleoperators and Virtual Environments*, 7:36–52, 1998.

[119] Donald E. Knuth. *Sztuka Programowania (The art of computer programming)*, volume 3. Wydawnictwa Naukowo-Techniczne, Warszawa, 2002.

[120] I. I. Kosenko. Integration of the equations of a rotational motion of a rigid body in quaternion algebra. the euler case. *Journal of Applied Mathematics and Mechanics*, 62:193–200, 1998.

[121] Tomasz Koziara and Nenad Bićanić. Non-smooth contact dynamics method for pseudo-rigid bodies. In *Proc. 14th ACME/ISSEC Conference*, pages 159–162, Belfast, 2006.

[122] Tomasz Koziara and Nenad Bićanić. On parallelisation in the nonsmooth contact dynamics method. In *Proc. II International Conference on Nonsmooth/Nonconvex Mechanics*, pages 339–346, Thessaloniki, 2006.

[123] Tomasz Koziara and Nenad Bićanić. Semismooth newton method for frictional contact between pseudo-rigid bodies. *Computer Methods in Applied Mechanics and Engineering*, 197:2763–2777, June 2008.

[124] Tomasz Koziara and Nenad Bicanic. Simple and efficient integration of rigid rotations suitable for constraint solvers. *Submited to the International Journal for Numerical Methods in Engineering*, 2008.

[125] Erwin Kreyszig. *Introductory functional analysis with applications*. John Wiley & Sons, 1989.

[126] P. Krysl. Explicit momentum-conserving integrator for dynamics of rigid bodies approximating the midpoint lie algorithm. *International Journal for Numerical Methods in Engineering*, 63:2171–2193, 2005.

[127] P. Krysl. Direct time integration of rigid body motion with discrete-impulse midpoint approximation: explicit newmark algorithms. *Communications in Numerical Methods in Engineering*, 22:441–451, 2006.

[128] P. Krysl. Dynamically equivalent implicit algorithms for the integration of rigid body rotations. *Communications in Numerical Methods in Engineering*, 24(2):141 – 156, 2006.

[129] M. Kunze and M. D. P. Monteiro Marques. An Introduction to Moreau's Sweeping Process. In B. Brogliato, editor, *Impacts in Mechanical Systems. Analysis and Modelling*, volume 551 of *Lecture Notes in Physics, Berlin Springer Verlag*, pages 1–60, 2000.

[130] T. A. Laursen. The convected description in large deformation frictional contact problems. *International Journal of Solids and Structures*, 31:669 – 681, 1994.

[131] R. I. Leine, B. Brogliato, and H. Nijmeijer. Periodic motion and bifurcations induced by the painlevé paradox. *European Journal of Mechanics - A/Solids*, 21:869–896, 2002.

[132] R. I. Leine and Ch. Glocker. A set-valued force law for spatial coulomb-contensou friction. *European Journal of Mechanics, A/Solids*, 22:193–216, 2003.

[133] C. E. Lemke. Bimatrix equilibrium points and mathematical programming. *Management Science*, 11:681–689, 1965.

[134] E. Levey, C. Peters, and O'Sullivan C. New Metrics for Evaluation of Collision Detection Techniques. Technical Report TCD-CS-1999-55, University of Dublin, 1999.

[135] D. Lewis and J. C. Simo. Nonlinear stability of rotating pseudo-rigid bodies. *Royal Society of London Proceedings Series A*, 427:281–319, 1990.

[136] C. F. Li, Y. T. Feng, and D. R. J. Owen. Smb: collision detection based on temporal coherence. *Computer Methods in Applied Mechanics and Engineering*, 195:2252–2269, 2006.

[137] S. Li, D. Qian, W.K. Liu, and T. Belytschko. A meshfree contact-detection algorithm. *Computer Methods in Applied Mechanics and Engineering*, (190):3271–3292, 2001.

[138] Tsai-Yen Li and Jin-Shin Chen. Incremental 3d collision detection with hierarchical data structures. In *VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 139–144, New York, NY, USA, 1998. ACM.

[139] Ming C Lin. *Efcient collision detection for animation and robotics*. PhD thesis, University of California, Berkeley, CA, 1993.

[140] Caishan Liu, Zhen Zhao, and Bernard Brogliato. Theoretical Analysis and Numerical Algorithm for Frictionless Multiple Impacts in Multibody Systems. Technical Report 00204018/1, Inria, 2008.

[141] B. Llanas, M. Fernandez De Sevilla, and V. Feliu. Minimum distance between the faces of two convex polyhedra: A sufficient condition. *Journal of Global Optimization*, 26:361–385, MM 2004.

[142] P. B. Lourenco, D. V. Oliveira, P. Roca, and A. Orduna. Dry joint stone masonry walls subjected to in-plane combined loading. *Journal of Structural Engineering*, 131:1665–1673, 2005.

[143] Rodrigo G. Luque, Joao L. D. Comba, and Carla M. D. S. Freitas. Broad-phase collision detection using semi-adjusting bsp-trees. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 179–186, New York, NY, USA, 2005. ACM Press.

[144] G. Jelenic M. A. Crisfield. Objectivity of strain measures in the geometrically exact three-dimensional beam theory and its finite-element implementation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 455:1125–1147, March 1999.

[145] D. Camotim M. Ritto-Corrêa. On the differentiation of the rodrigues formula and its significance for the vector-like parameterization of reissner-simo beam theory. *International Journal for Numerical Methods in Engineering*, 55:1005–1032, 2002.

[146] Jari Mäkinen. *A Formulation for Flexible Multibody Mechanics (PhD thesis)*. Applied mechanics and optimization, Tampere University of Technology, Finland, 2004.

[147] Jerrold E. Marsden and Thomas J. R. Hughes. *Mathematical Foundations of Elasticity*. Courier Dover Publications, 1994.

[148] Edward M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14:257–276, 1985.

[149] Walter Meyer. Distances between boxes: applications to collision detection and clipping. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 597–602, San Francisco, CA, 1986.

[150] A. R. Mijar and J. S. Arora. Review of formulations for elastostatic frictional contact problems. *Structural and Multidisciplinary Optimization*, 20:167–189, November 2000.

[151] Brian Mirtich. V-clip: Fast and robust polyhedral collision detection. In *ACM Transactions on Graphics*, volume 17, pages 177–208, 1998.

[152] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics (ACM)*, 22:289–298, 1988.

[153] J. J. Moreau. Some basics of unilateral dynamics. In F. Pfeiffer and C. Glocker, editors, *Unilateral Multibody Contacts*. Kluwer, Dordrecht, 1999.

[154] J. J. Moreau. The Kinematics of Unilaterality. Slides, Laboratoire de Mécanique et Génie Civil. Université Montpellier II, Siconos-Da Vinci Meeting, Grenoble, July 2005.

[155] Jean Jacques Moreau. Evolution problem associated with a moving convex set in a Hilbert space. *Journal of differential equations*, 26:347–374, 1977.

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

[157] Christian Worm Mortensen. Fully-dynamic two dimensional orthogonal range and line segment intersection reporting in logarithmic time. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 618–627, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[158] D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7:217–236, 1978.

[159] J. Nievergelt and F. P. Preparata. Plane-sweep algorithms for intersecting geometric figures. *Commun. ACM*, 25(10):739–747, 1982.

[160] T. R. Nordenholz and O. M. O'Reilly. On steady motions of isotropic, elastic cosserat points. *IMA Journal of Applied Mathematics*, 60:55–72, 1998.

[161] T. R. Nordenholz and O. M. O'Reilly. Class of motions of elastic, symmetric cosserat points: Existence, bifurcation, and stability. *International Journal of Non-Linear Mechanics*, 36:353–374, 2001.

[162] I. P. Omelyan. Algorithm for numerical integration of the rigid-body equations of motion. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 58:1169–1172, 1998.

[163] L. Endres P. Krysl. Explicit newmark/verlet algorithm for time integration of the rotational dynamics of rigid bodies. *International Journal for Numerical Methods in Engineering*, 62:2154–2177, 2005.

[164] A. Pandolfi, C. Kane, J.E. Marsden, and M. Ortiz. Time-discretized variational formulation of nonsmooth frictional contact. *International Journal for Numerical Methods in Engineering*, 53(4):1801–1829, 2002.

[165] J. S. Pang, J. C. Trinkle, and G. Lo. A complementarity approach to a quasistatic multi-rigid-body contact problem. *Computational Optimization and Applications*, 5:139–154, 1996.

[166] Jong-Shi Pang. Newton's method for β-differentiable equations. *Mathematics of Operations Research*, 15:311–341, 1990.

[167] P. Papadopoulos. On a class of higher-order pseudo-rigid bodies. *Mathematics and Mechanics of Solids*, 6:631–640, 2001.

[168] Yves Renard Patrick Laborde. Fixed point strategies for elastostatic frictional contact problems. *Mathematical Methods in the Applied Sciences*, 31:415–441, 2008.

[169] Eric Perkins and John R. Williams. A fast contact detection algorithm insensitive to object sizes. *Engineering Computations*, 18(1/2):48 – 62, 2001.

[170] Friedrich Pfeiffer, Martin Foerg, and Heinz Ulbrich. Numerical aspects of non-smooth multibody dynamics. *Computer Methods in Applied Mechanics and Engineering*, 195:6891–6908, October 2006.

[171] William Shelton Jr Phani Kumar, V. V. Nukala. Semi-implicit reversible algorithms for rigid body rotational dynamics. *International Journal for Numerical Methods in Engineering*, 69:2636–2662, 2007.

[172] M. J. D. Powell. *Optimization*, chapter A method for non-linear constraint in optimization problems, pages 283–298. Academic Press, London, 1969.

[173] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20(2):87–93, 1977.

[174] F. P. Preparata and D. E. Muller. Finding the intersection of n half-spaces in time O(n log n). *Theoretical Computer Science*, 8:45–55, 1979.

[175] Liqun Qi and Jie Sun. A nonsmooth version of Newton's method. *Mathematical Programming*, V58:353–367, 1993.

[176] Sean Quinlan. Efficient distance computation between non-convex objects. In *Proceedings - IEEE international conference on robotics and automation*, volume 4, pages 3324–3329, San Diego, CA, 1994.

[177] M. Renouf and P. Alart. Conjugate gradient type algorithms for frictional multi-contact problems: Applications to granular materials. *Computer Methods in Applied Mechanics and Engineering*, 194:2019–2041, 2005.

[178] M. Renouf, F. Dubois, and P. Alart. A parallel version of the non smooth contact dynamics algorithm applied to the simulation of granular media. *Journal of Computational and Applied Mathematics*, 168:375–382, 2004.

[179] J. M. Rico-Martinez and J. Gallardo-Alvarado. Simple method for the determination of angular velocity and acceleration of a spherical motion through quaternions. *Meccanica*, 35:111–118, 2000.

[180] R. T. Rockafellar. Characterization of the subdifferentials of convex functions. *Pacific Journal of Mathematics*, 17(3):497–510, 1966.

[181] R. T. Rockafellar. The multiplier method of Hestenes and Powell applied to convex programming. *Journal of Optimization Theory and Applications*, 12:555–562, MM 1973.

[182] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1997.

[183] R. Tyrrell Rockafellar and Roger J-B. Wets. *Variational analysis*. Springer, 1998.

[184] H. Samet. *The design and analysis of spatial data structures*. Addison-Wesley, Reading, MA, 1990.

[185] Hanan Samet. Hierarchical representations of collections of small rectangles. *ACM Comput. Surv.*, 20(4):271–309, 1988.

[186] N.K. Sancheti and S.S. Keerthi. Computation of certain measures of proximity between convex polytopes: a complexity viewpoint. In *Proceedings IEEE International Conference on Robotics and Automation*, volume 3, pages 2508 – 2513, Nice, France, 1992.

[187] John R. Williams Benjamin K. Cook Scott M. Johnson. Quaternion-based rigid body rotation integration algorithms for use in particle methods. *International Journal for Numerical Methods in Engineering*, 74(8):1303–1313, 2008.

[188] D. Sha, K.K. Tamma, and M. Li. Robust Explicit Computational Developments and Solution Strategies for Impact Problems Involving Friction. *International Journal of Numerical Methods in Engineering*, 39(5):721–739, 1996.

[189] Ravishankar Shivarama and Eric P. Fahrenthold. Hamilton's equations with Euler parameters for rigid body dynamics modeling. *Journal of Dynamic Systems, Measurement, and Control*, 126(1):124–130, 2004.

[190] J. C. Simo and N. Tarnow. The discrete energy-momentum method. Conserving algorithms for nonlinear elastodynamics. *Zeitschrift für Angewandte Mathematik und Physik (ZAMP)*, 43:757–792, September 1992.

[191] J. C. Simo and L. Vu-Quoc. On the dynamics in space of rods undergoing large motions – a geometrically exact approach. *Computer Methods in Applied Mechanics and Engineering*, 66:125–161, February 1988.

[192] H. W. Six and D. Wood. The rectangle intersection problem revisited. *BIT Numerical Mathematics*, 20:426–433, December 1980.

[193] J. M. Solberg and P. Papadopoulos. Impact of an elastic pseudo-rigid body on a rigid foundation. *International Journal of Engineering Science*, 38:589–603, 2000.

[194] Peng Song, Jong-Shi Pang, and R. Vijay Kumar. A Semi-Implicit Time-Stepping Model For Frictional Compliant Contact Problems. *International Journal for Numerical Methods in Engineering*, 60(13):2231–2261, 2004.

[195] G. Stadler. *Infinite-Dimensional Semi-Smooth Newton and Augmented Lagrangian Methods for Friction and Contact Problems in Elasticity*. PhD thesis, University of Graz, 2004.

[196] David J. Steigmann. On pseudo-rigid bodies. *Proceedings of the Royal Society - Mathematical, Physical and Engineering Sciences (Series A)*, 462:559–562, 2006.

[197] D. E. Stewart. Convergence of a Time-Stepping Scheme for Rigid-Body Dynamics and Resolution of Painlevé's Problem. *Archive for Rational Mechanics and Analysis*, 145:215–260, 1998.

[198] D. E. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42:3–39, 2000.

[199] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39:2673–2691, 1996.

[200] Gilbert Strang. *Linear algebra and its applications*. Brooks Cole, 1988.

[201] S. Suri, P. M. Hubbard, and J. F. Hughes. Analyzing bounding boxes for object intersection. In *ACM Transactions on Graphics*, volume 18, pages 257–277, 1999.

[202] V. CHAWLA T. A. LAURSEN. Design of energy conserving algorithms for frictionless dynamic contact problems. *International Journal for Numerical Methods in Engineering*, 40:863–886, 1997.

[203] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of VMV*, pages 47–54, Munich, Germany, 2003.

[204] J. C. Trinkle, S. Berard, and J. S. Pang. A time-stepping scheme for quasistatic multibody systems. In *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, pages 174–181, 2005.

[205] P. Underwood. Dynamic relaxation. In T. Belytschko and T.J.R. Hughes, editors, *Computational methods for transient analysis*, pages 245–265. Elsevier Science, 1983.

[206] T. A. Laursen V. Chawla. Energy consistent algorithms for frictional contact problems. *International Journal for Numerical Methods in Engineering*, 42:799–827, 1998.

[207] Gino Van den Bergen. A fast and robust gjk implementation for collision detection of convex objects. *J. Graph. Tools*, 4(2):7–25, 1999.

[208] K. Vlack and S. Tachi. Fast and Accurate Spacio-temporal Intersection Detection with the GJK Algorithm. In *Int. Conf. Artif. Real Telexistence*, pages 79–84, Tokyo, Japan, 2001.

[209] Herbert S. Wilf. *Algorithms and Complexity*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 1986.

[210] M. Wosle and F. Pfeiffer. Dynamics of multibody systems containing dependent unilateral constraints with friction. *Journal of Vibration and Control*, 2:161–192, 1996.

[211] Peter Wriggers. *Computational Contact Mechanics*. Springer, 2 edition, 2006.

[212] Chia-Ju Wu and Tsu-Tian Lee. Projective method for collision-detection of robot arms moving among obstacles. In *Proceedings of the Annual Southeastern Symposium on System Theory*, pages 86–90, Charlotte, NC, USA, 1988.

[213] Jian-Hong Wu, C. H. Juang, and Hung-Ming Lin. Vertex-to-face contact searching algorithm for three-dimensional frictionless contact problems. *International Journal for Numerical Methods in Engineering*, 63:876–897, 2005.

[214] G. Zavarise and P. Wriggers. A superlinear convergent augmented Lagrangian procedure for contact problems. *Engineering Computations*, 16(1):88 – 119, 1999.

[215] Y. Zhou and S. Suri. Analysis of a bounding box heuristic for object intersection. *Journal of the ACM*, 46:833–857, 1999.

[216] A. Zomorodian and H. Edelsbrunner. Fast software for box intersections. *International Journal of Computational Geometry and Applications*, 12:143–172, 2002.