FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

Lehrstuhl für Informatik 10 (Systemsimulation)

Iterative Rigid Multibody Dynamics

Tobias Preclik

Diplomarbeit

Iterative Rigid Multibody Dynamics

Tobias Preclik

Diplomarbeit

Aufgabensteller:	Prof. Dr. Ulrich Rüde
Betreuer:	Klaus Iglberger, M. Sc.
	Dr.-Ing. Harald Köstler
Bearbeitungszeitraum:	2.5.2008 – 3.11.2008

Erklärung:

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 3.11.2008 .....

Abstract

The simulation of large rigid multibody systems is a challenge. Collisions and friction constraints cause non-smooth motions of the rigid bodies. Differential complementarity problems are therefore used to formulate the dynamical system. The efficient solution of such problems is difficult and the major focus of this thesis. Several modeling choices of varying accuracy and difficulty are presented. Current solution methods are investigated and a new algebraic multigrid approach is proposed. Numerical tests on granular media problems are conducted to validate the practicability. The strengths and weaknesses of the solvers are identified and advanced methods suggested.

Zusammenfassung

Die Simulation von großen starren Mehrkörpersystemen ist eine Herausforderung. Kollisionen und Reibungsbedingungen verursachen nicht-glatte Starrkörperbewegungen. Differential complementarity problems werden deswegen für die Formulierung des dynamischen System verwendet. Die effiziente Lösung dieser Probleme ist schwierig und steht im Fokus dieser Arbeit. Mehrere Modellierungsmöglichkeiten unterschiedlicher Genauigkeit und Schwierigkeit werden vorgestellt. Gängige Lösungsmethoden werden untersucht und ein neuer algebraischer Mehrgitteransatz vorgeschlagen. Numerische Tests mit granularen Materien werden durchgeführt um die Praktikabilität zu validieren. Die Stärken und Schwächen der Löser werden identifiziert und weitergehende Methoden vorgeschlagen.

Acknowledgements

Every publication is formed not only by its writer but by the valuable direct or indirect support from many people. In my case my advisor Klaus Iglberger supported me in every possible respect. Thank you for making this thesis at all possible and for providing a superb physics framework! The discussion rounds together with Harald Köstler and Ulrich Rüde were enormously helpful. Harald especially advised me on multigrid concerns and proofread the thesis. Ulrich Rüde offered me the great opportunity to visit the University of Iowa. The stay abroad was an enrichment for the thesis and not least a personal gain. Suely Oliveira and David Stewart kindly hosted my stay. Thank you for providing me with advice and everything else needed for my thesis as well as for the happy hours in my off time. Last but not least I want to thank my family and friends who reminded me of the important things in life.

Thanks to all of you!

Contents

1 Introduction **9**
1.1 Relevance . . . 9
1.2 Overview . . . 9
1.3 Focus . . . 10

2 Non-smooth Problems **11**
2.1 Complementarity Problems . . . 11
2.2 Variational Inequalities . . . 13
2.3 Residuals . . . 14

3 Contact Modeling **19**
3.1 System Representation . . . 19
3.2 Contact Gaps . . . 20
3.3 Non-penetration Constraints . . . 20
3.4 Friction Constraints . . . 23
3.5 Friction Cones . . . 24
3.6 System Properties . . . 25
3.6.1 Contact Problem . . . 25
3.6.2 Matrix Properties . . . 26
3.6.3 Entry Properties . . . 27
3.6.4 Sparsity . . . 28
3.7 Reformulations and Simplifications . . . 29
3.7.1 Frictionless Problems . . . 29
3.7.2 Friction Boxes . . . 29
3.7.3 LCP Formulation . . . 30
3.7.4 Summary . . . 32

4 Solvers **33**
4.1 Splitting Methods . . . 33
4.1.1 Projections . . . 33
4.1.2 Total Enumeration . . . 34
4.1.3 Friction Handling . . . 35
4.1.4 Block Variants . . . 35
4.2 Conjugate Projected Gradient Method . . . 39
4.2.1 Optimization Reformulation . . . 39
4.2.2 Incorporation of Bounds . . . 40
4.2.3 Iterate Correction . . . 42
4.2.4 Gradient Projections . . . 43
4.2.5 Drawbacks . . . 43
4.3 Non-smooth Newton Method . . . 43
4.3.1 Root-finding Reformulation . . . 44
4.3.2 Jacobian Construction . . . 45
4.3.3 Solving the Newton Equation . . . 47
4.3.4 Drawbacks . . . 47
4.4 Projected Algebraic Multigrid . . . 48
4.4.1 Multigrid Benefits . . . 48
4.4.2 Coarsening Strategy . . . 48
4.4.3 Interpolation Operator . . . 49

4.4.4 Coarse Problems . . . 51
4.4.5 Multigrid Cycles . . . 51
4.4.6 Open Problems . . . 51
4.5 Alternatives . . . 52
4.6 Summary . . . 53

5 Numerical Tests **55**
5.1 Test Problems . . . 55
5.2 Residuals . . . 55
5.3 Velocity Errors . . . 56
5.4 Active Set Stabilization . . . 59
5.5 Convergence Issues . . . 62
5.6 Time Measurements . . . 63
5.7 Robustness . . . 65

6 Conclusion **68**
6.1 Future Work . . . 68
6.2 Results . . . 69

References **71**

List of Figures

1 A two dimensional constraint set of a VIP. 14
2 A two dimensional constraint set of a CCP. 15
3 Quantities involved at a contact gap. 20
4 The circular friction cone and a polyhedral approximation. 24
5 Relation between friction force and sliding. 25
6 The friction cone differences between the LCP and the box LCP formulations. . 31
7 Sparsity structure of the Jacobian. 46
8 Adjacency graph showing the standard C/F-splitting. 53
9 Spherical granular media inside a container. 55
10 Residual plots of the solvers for a granular media test case. 57
11 Residual plots of the PGS and CPG method for a granular media test case with varying friction coefficients. 58
12 Residual plots of the accurate and approximate Newton method. 59
13 Residual and error plots each solver on a granular media test case. 60
14 Active set changes per iteration of different solvers. 61
15 Oscillating residual plots of the PGS and CPG method. 62
16 Counter measures stop oscillations in residual plots. 63
17 The time spent per iteration against the number of unknowns. 64
18 Initial configuration of the castle simulation. 65
19 The destruction of the castle. 66
20 The castle relicts. 67

List of Tables

1 Summary of complementarity problem generalizations. 15
2 Summary of contact problem models. 32
3 Summary of solver capabilities. 54

List of Algorithms

1 The projected SOR method. 34
2 The Conjugate Projected Gradient algorithm. 41
3 The correction of the iterate in the CPG method. 42
4 The projection of the negative gradient to the tangent cone. 43
5 Backtracking line-search of Armijo type. 45

1 Introduction

1.1 Relevance

It was back in the 17th century when Isaac Newton published his findings on gravitation and thus brought the field of classical mechanics into being. One of its subfields is the studies of the motions of rigid bodies. Though the rigidity assumption is a rigorous idealization of real materials many phenomenons in nature and problems posed in engineering can be well approximated in this way. In fact a vast number of research communities is interested in the simulation of rigid bodies, each one focusing on different aspects of the general problem.

The robotics field is for example interested in the accurate prediction of friction forces, so that walking machines have stable contact on the floor and robot-arm grippers can reliably grasp objects. Though these problems often involve only few contacts there are other applications which involve a huge number of contacts like the simulation of granular materials. Materials usually range from fine particles like powders and sand, to rice grains and nuts. In either case a tremendous amount of particles has to be simulated. Those simulations reveal that granular materials often exhibit properties which are similar to flows of fluids. Phenomenons like the buoying upwards of larger particles can also be reproduced in simulations. This is usually referred to as the Brazil nut effect, since the larger Brazil nuts usually end up at the top when shaking a bowl filled with a mixture of nuts [2, section 5.3]. Another interesting problem is the simulation of the draining of granular material through an opening at the bottom of a silo. This situation arises for example in so called pebble-bed nuclear reactors where the grains are graphite fuel pebbles [4, section 5.2]. Consequently granular flows concern power companies, food as well as pharmaceutical industries. Even the research on pedestrian and evacuation dynamics benefits from rigid body simulations: A column a little upstream in front of an emergency exit helps to complete evacuations faster and with fewer injuries [1, section 1].

Completely different demands have to be met by the computer graphics community. In so called physics-based animations visually pleasant motions of bodies are usually satisfactory. Particularly the production of computer-generated imagry (CGI) films benefits from simulations. Though usually less accurate models are employed often real-time requirements are demanded instead. Especially the physics simulations in video games must run at interactive rates. Similar goals are indispensable in the virtual-reality community. VR techniques can be adopted for the rapid development of prototypes which in turn attracts amongst others the automotive industry.

1.2 Overview

To setup a simulation of rigid bodies one starts out with specifying the shapes of the bodies, their position and orientation in the global world frame. Bodies are often either primitve geometrical objects like spheres and boxes or compounds thereof. More complicated objects can be specified as triangle meshes. In addition to the position, the initial velocity has to be given which includes the linear (translational) as well as angular velocity. Every object is subject to external forces like gravity. Furthermore contact forces act at the points where objects touch each other. Contact forces include *normal forces* which prevent bodies from interpenetrating and *friction forces* which act in the tangential plane of every contact and try to resist slipping. Contrary to the external forces the contact forces are unknown and have to be computed. After this tedious job has been done the bodies can be advanced for a small time step. This process results in a simulation cycle which is repeated over and over again. Each

cycle starts by determining all points where the objects are in contact which is often referred to as *collision detection*. The simplest way to achieve this is to apply an exhaustive search, that is checking each object against each other for overlaps. In any case, the algorithm must also provide the normal direction at each contact. The collected collision information is passed on to the *collision response* module. Its primary task is to determine contact forces which should at least avoid interpenetrations of bodies. However already this part of the problem is of a non-smooth nature. The positions and velocities do not have to evolve smoothly in time. Consider for example the impact of a body. To prevent interpenetration at the time of impact, the relative velocity at the contact must change instantaneously which produces even a discontinuity in the velocity function. To cope with that problem smooth approximations have been suggested. Methods for solving ordinary differential equations remain applicable but the approximations are stiff. To obtain a numerically stable time integration, the time steps have to be chosen very small. To be precise the stiffer the approximation, the smaller the time steps must be chosen. Additional disadvantages are, that "the approximation may introduce nonphysical artifacts" and furthermore that "the smoothing parameters are notoriously difficult to tune" [1]. A different way to deal with the non-smoothness is to use hard constraints. These approaches are usually formulated as differential *complementarity problems*. Again different methods exist but in any case especially the incorporation of friction makes the problems even harder to solve and the modeling is always a trade off between correctness and computational burden. Section 4 will explore different approaches to solve the various model problems. Once the problem has been solved, the contact forces are applied which results in new body positions, velocities and collisions. The simulation cycle can start again.

1.3 Focus

This thesis focuses on the *non-smooth dynamics* of rigid multibody systems. The contact problem is formulated following an impulse-velocity approach. The final problem is stated as a complementarity problem. Different simplifications are presented which result in easier problems. Problem specific properties and ways to exploit them in solution methods are of special interest. Though some direct solvers will be mentioned, the majority of the solvers will be iterative in nature. Among them will be Gauss-Seidel type of methods, Newton methods and Conjugate-Gradient type of methods. The thesis will be closed by numerical tests and some suggestions for further research directions.

2 Non-smooth Problems

As noted earlier we will use non-smooth problem formulations to model the dynamics, that is we will not use smooth approximations. After the time-discretization we will thus not end up with a system of equations but a system with complementarity conditions. The purpose of this section is to introduce the various complementarity problems available.

2.1 Complementarity Problems

The link between all complementarity problems are the *complementarity conditions*. This condition states that one of two quantities must be zero. Quantities can be either simple variables or functions thereof. The most basic form of such a problem is the *Linear Complementarity Problem* (LCP):

$$\begin{aligned} &\text{find} && \vec{x} \in \mathbb{R}^n_+ \\ &\text{subject to} && \vec{x}, \vec{y} \geq 0 \quad \vec{x} \perp \vec{y} \\ &\text{where} && \vec{y} = \mathbf{A}\vec{x} + \vec{b} \\ & && \vec{y}, \vec{b} \in \mathbb{R}^n \quad \mathbf{A} \in \mathbb{R}^{n \times n} \end{aligned} \tag{1}$$

The complementarity condition is expressed by $\perp$ and must be understood componentwise. Any comparison operators have to be understood componentwise too. Since all quantities are non-negative, the complementarity condition equals to $\vec{x}^T \vec{y} = 0$. Here $\vec{y}$ is a dependent variable and can be expressed as an affine function of the independent variable $\vec{x}$. A slight generalization of the LCP is the *Mixed Linear Complementarity Problem* (mLCP). The name stems from the fact that additional equations are mixed into the LCP:

$$\begin{aligned} &\text{find} && \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}_+ \\ &\text{subject to} && \vec{y}_1 = 0 \quad \vec{x}_2, \vec{y}_2 \geq 0 \quad \vec{x}_2 \perp \vec{y}_2 \\ &\text{where} && \begin{pmatrix} \vec{y}_1 \\ \vec{y}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} + \begin{pmatrix} \vec{b}_1 \\ \vec{b}_2 \end{pmatrix} \\ & && \vec{y}_1, \vec{b}_1 \in \mathbb{R}^{n_1} \quad \mathbf{A}_{11} \in \mathbb{R}^{n_1 \times n_1} \quad \mathbf{A}_{12} \in \mathbb{R}^{n_1 \times n_2} \\ & && \vec{y}_2, \vec{b}_2 \in \mathbb{R}^{n_2} \quad \mathbf{A}_{21} \in \mathbb{R}^{n_2 \times n_1} \quad \mathbf{A}_{22} \in \mathbb{R}^{n_2 \times n_2} \end{aligned} \tag{2}$$

If the submatrix $\mathbf{A}_{11}$ is non-singular, then $\vec{x}_1$ can be easily eliminated from the system by inserting $\vec{x}_1 = -\mathbf{A}_{11}^{-1}\left(\vec{b}_1 + \mathbf{A}_{12}\vec{x}_2\right)$. The remaining system is a standard LCP problem.

Since the dependent variable in the LCP is defined by an affine function, the generalization to arbitrary functions suggests itself. The resulting problem is called the *Nonlinear Complementarity Problem* (NCP) and is defined as follows:

$$
\begin{array}{ll}
\text{find} & \vec{x} \in R^n_+ \\
\text{subject to} & \vec{x}, \vec{y} \geq \vec{0} \quad \vec{x} \perp \vec{y} \\
\text{where} & \vec{y} = F(\vec{x}) \\
 & \vec{y} \in \mathbb{R}^n \quad F(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^n
\end{array}
\tag{3}
$$

Up to now the independent variables are subject to unilateral bounds. To be precise either they have no bounds at all (equality constraints) or they are non-negative. This can be easily extended to the case where each variable has arbitrary lower ($\underline{x}_i$) as well as upper ($\overline{x}_i$) bounds. Since those constraints form a rectangular region $\prod_{i=1}^n [\underline{x}_i, \overline{x}_i]$, they are called box constraints. The extension of the NCP to box constraints is called *Mixed Complementarity Problem* (MCP) not to be confused with mLCPs. Similar to LCPs the dependent variable is zero as long as the independent variable is not bounded. If it is lower bounded, then the dependent variable is required to be non-negative as in the LCP case. However, if the independent variable is upper bounded, the dependent variable must be non-positive.

$$
\begin{array}{lll}
\text{find} & \vec{x} \in \prod_{i=1}^n [\underline{x}_i, \overline{x}_i] & \\
\text{subject to} & F_i(\vec{x}) = 0 \quad \to & \underline{x}_i \leq x_i \leq \overline{x}_i \\
 & F_i(\vec{x}) > 0 \quad \to & x_i = \underline{x}_i \\
 & F_i(\vec{x}) < 0 \quad \to & x_i = \overline{x}_i \\
\text{where} & F(\vec{x}) = (F_i(\vec{x}))_{i=1\ldots n} : \mathbb{R}^n \to \mathbb{R}^n & \\
 & \vec{\underline{x}} \in (\mathbb{R} \cup \{-\infty\})^n \quad \vec{\overline{x}} \in (\mathbb{R} \cup \{\infty\})^n &
\end{array}
\tag{4}
$$

For convenience I introduce a shorter notation which should be considered equal to (4):

$$
\begin{array}{ll}
\text{find} & \vec{x} \in \prod_{i=1}^n [\underline{x}_i, \overline{x}_i] \\
\text{subject to} & F(\vec{x}) \gtreqless \vec{0} \quad \perp \quad \vec{\underline{x}} \leq \vec{x} \leq \vec{\overline{x}} \\
\text{where} & F(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^n \\
 & \vec{\underline{x}} \in (\mathbb{R} \cup \{-\infty\})^n \quad \vec{\overline{x}} \in (\mathbb{R} \cup \{\infty\})^n
\end{array}
\tag{5}
$$

The MCP is the most powerful problem so far and includes every complementarity problem introduced up to this point. If the function F is affine the MCP reduces to the LCP family. If the lower bound is the zero vector and the upper bound is infinity, then we get a standard LCP. If some of the lower bounds are allowed to be negative infinity, then we end up with an mLCP. Another interesting special case, which was not introduced yet, occurs if the function F is affine but the lower and upper bounds are arbitrary. This case results in a *box-constrained LCP* (bLCP) or sometimes also misleadingly referred to as *Mixed Linear Complementarity Problem*. Though obvious it is nevertheless noteworthy that linear systems of equations are special cases of box and mixed LCPs.

In the context of rigid body dynamics there exists another modification of the *box LCP* where the bounds depend on the unknowns:

$$
\begin{aligned}
&\text{find} && \vec{x} \in \mathbb{R}^n \\
&\text{subject to} && \vec{y} \gtreqless \vec{0} \quad \perp \quad \underline{x}(\vec{x}) \leq \vec{x} \leq \overline{x}(\vec{x}) \\
&\text{where} && \vec{y} = \mathbf{A}\vec{x} + \vec{b} \\
& && \vec{y}, \vec{b} \in \mathbb{R}^n \quad \mathbf{A} \in \mathbb{R}^{n \times n} \\
& && \underline{x} : \mathbb{R}^n \to \mathbb{R}^n \quad \overline{x} : \mathbb{R}^n \to \mathbb{R}^n
\end{aligned}
\tag{6}
$$

Though a very important consequence of this modification is that the problem becomes *non-linear*, I will refer to it as a *modified box LCP* or only *box LCP* if it is clear from the context that the bounds are functions of the unknowns. Though (6) allows arbitrary functions for the upper and lower bounds, usually only affine functions are used.

2.2 Variational Inequalities

All constraint sets so far have a rectangular region. For example the unknowns of a standard LCP are constrained to the non-negative orthant $\mathbb{R}^n_+$ and the box-constrained LCP and the MCP are constrained to $\prod_{i=1}^n [\underline{x}_i, \overline{x}_i]$ as noted earlier. Variational Inequalities Problems (VIP) in contrast are defined on arbitrary constraint sets K, so that it reduces to the complementarity problems if the constraint sets are rectangular:

$$
\begin{aligned}
&\text{find} && \vec{x} \in K \\
&\text{subject to} && (\vec{y} - \vec{x})^T F(\vec{x}) \geq 0 \quad \forall \vec{y} \in K \\
&\text{where} && K \subset \mathbb{R}^n \\
& && F(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^n
\end{aligned}
\tag{7}
$$

Intuitively the constraint requires that the mapping of a solution ($F(\vec{x})$) must point at least partially in the same direction as any other vector pointing into the constraint set K from the solution $\vec{x}$. Therefore for each unknown there exists a set $S_K(\vec{x})$ in which the mapping of the unknown must lie in order for the constraint to be satisfied:

$$
S_K(\vec{x}) = \begin{cases} \left\{ \vec{s} \;\middle|\; (\vec{y} - \vec{x})^T \vec{s} \geq 0, \quad \forall \vec{y} \in K \right\} & \text{if } \vec{x} \in K \\ \emptyset & \text{else} \end{cases}
\tag{8}
$$

Thus we can restate the VIP in a different fashion:

$$
\begin{aligned}
&\text{find} && \vec{x} \in K \\
&\text{subject to} && F(\vec{x}) \in S_K(\vec{x}) \\
&\text{where} && K \subset \mathbb{R}^n \\
& && F : \mathbb{R}^n \to \mathbb{R}^n
\end{aligned}
\tag{9}
$$

Fig. 1 shows a two dimensional constraint set of a VIP. Small circles represent potential solutions of a VIP and the shaded region originating from each of those solutions is the set in which the mapping of the potential solution must lie in order for the constraint to be satisfied.

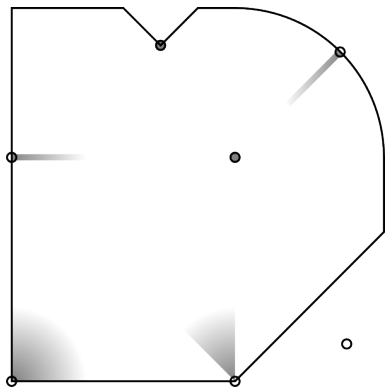


Figure 1: A two dimensional constraint set of a VIP. Some potential solutions $\vec{x}_i$ are marked by small circles. The shaded regions show a part of $S_K(\vec{x}_i)$ respectively.

For *closed convex sets* $K \subset \mathbb{R}^n$ the definition of a *normal cone* makes sense:

$$N_K(\vec{x}) = -S_K(\vec{x}) \tag{10}$$

To depict the concept of a normal cone consider a hyperplane of a constraint set. If $\vec{x}$ resides on the hyperplane, then $N_K(\vec{x})$ coincides with the normal ray of the hyperplane. This requires that K is convex, otherwise the normal cone can degenerate to the zero vector. The normal cones of unknowns in the interior of the constraint set consist of the zero vector only since it is the only vector pointing partially in any other direction. The normal cones at corners or rather at points where several hyperplanes of the constraint set intersect, can form a non-degenerate convex cone instead of only a ray or a single zero vector. A (pointed) convex cone C is a set where the linear combination with non-negative coefficients of any two elements is again part of the set:

$$C \text{ is a pointed convex cone} \quad \leftrightarrow \quad \forall \vec{x}_1, \vec{x}_2 \in C \wedge \alpha, \beta \in \mathbb{R}^+ : \quad \alpha\vec{x}_1 + \beta\vec{x}_2 \in C \tag{11}$$

The term pointed emphasizes, that the zero vector is included in the cone. Often another type of complementarity problem is introduced, where the constraint set is a convex cone. The *complementarity problem over a cone* or *cone complementarity problem* (CCP) is thus nothing else than a VIP with a convex cone C. Clearly the NCP is a special case where C is the non-negative orthant. Fig. 2 shows a two dimensional CCP including several normal cones. Note that the normal cone at the apex coincides with the *polar cone* C^* of the cone C which is defined as follows:

$$C^* = \left\{ \vec{s} \,\middle|\, \vec{y}^T \vec{s} \leq 0, \quad \forall \vec{y} \in C \right\} \tag{12}$$

Tab. 1 summarizes the generalizations of the various complementarity problems. Clearly the VIP is the swiss-army knife here. Billups and Murty give a comprehensive summary of even more complementarity problems and their applications in Billups and Murty [5].

2.3 Residuals

Later on several iterative solution methods will be presented. Iterative methods in general do not terminate in a finite number of iterations. Therefore the solution process is stopped as

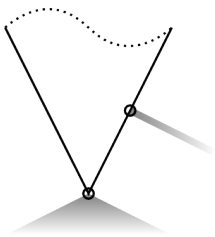


Figure 2: A two dimensional constraint set of a CCP. Some potential solutions $\vec{x}_i$ are marked by small circles. The shaded regions show a part of $N_K(\vec{x}_i)$ respectively.

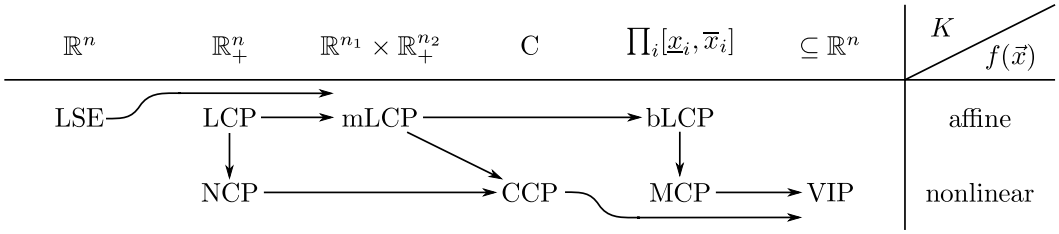


Table 1: Summary of complementarity problem generalizations.

soon as the current iterate solves the problem approximately. To employ a stopping criterion a measure for the quality of the solution is needed. For linear systems of equations the residual is usually defined as $\|\vec{b} - \mathbf{A}\vec{x}\|$. In principle the residual is computed for each equation separately as $b_i - (\mathbf{A}\vec{x})_i$. All local residuals are combined to a scalar quantity using a norm. Obviously the residual is 0 if $\vec{x}$ solves the problem. For complementarity problems the same applies, however, the residual computation is not as easy anymore. A good iterate must meet three conditions for a standard LCP: The independent variables (unknowns) $\vec{x}$ must be within the constraint set, i.e. greater equal zero. Furthermore the dependent variables $\vec{y} = \mathbf{A}\vec{x} + \vec{b}$ must also meet inequality constraints. In the case of an LCP $\vec{y} \geq \vec{0}$. Last the complementarity condition must hold - either the independent or the dependent variable must be zero $\vec{x}^T\vec{y} = 0$. The first two conditions require that the solution is *feasible* or in other words that $\vec{x}$ is in the set of feasible solutions:

$$\left\{ \vec{x} \;\middle|\; \vec{x} \geq \vec{0}, \mathbf{A}\vec{x} + \vec{b} \geq \vec{0} \right\} \tag{13}$$

Even if the unknowns are feasible they do not have to meet the complementarity requirements. A possible but computationally expensive residual can be constructed straight forward:

$$r = \left\| \Big(\min(\mathbf{A}\vec{x} + \vec{b}, \vec{0}), \min(\vec{x}, \vec{0}), \vec{x}^T(\mathbf{A}\vec{x} + \vec{b}) \Big) \right\| \tag{14}$$

A computationally cheaper alternative is the following:

$$r = \|\min(\vec{x}, \mathbf{A}\vec{x} + \vec{b})\| \tag{15}$$

Both variants are well known and examined thoroughly. The actual problem is to show that the residual is related to the error. Here we define the error as the minimum distance to a possible solution. Say the solution set is denoted by S, then the error can be defined as:

$$d(\vec{x}, S) = \inf \left\{ \|\vec{x} - \vec{x^*}\| \;\middle|\; \vec{x^*} \in S \right\} \tag{16}$$

Ideally there should exist two problem specific (positive) constants defining the range in which the error can be depending on the residual:

$$c_1 r \leq d(\vec{x}, S) \leq c_2 r \tag{17}$$

The question of existence of such constants is thoroughly studied in Cottle et al. [8], chapter 5.10. In essence those constants exist if the LCP matrix is a P-matrix. P-matrices are matrices whose principal[1] minors[2] are positive. Those matrices play an important role in LCP theory. A very important result is that an LCP has a unique solution for all vectors $\vec{b}$ if and only if $\mathbf{A}$ is a P-matrix [8, Theorem 3.3.7]. Note that positive-definite (PD) matrices are P-matrices but positive-semi-definite (PSD) matrices do not belong to that class. In Cottle et al. [8] two PSD examples are given where no constants exist (Example 5.10.1 and Exercise 5.11.22) and thus the residual (15) can break down for PSD matrices. Though the matrices arising from the contact problems will be PSD, empirical evidence exists, that the residual still suits our needs. A key difference to the counterexamples above is that the contact problem matrices do not exhibit any zeros on the diagonal.

Further on we will deal with complementarity problems with box constraints. Therefore an extension of the LCP residual (15) to box constraints is needed. In Ortiz-Rosado [12], section 3.2 the following function is derived:

$$r = \max\left(\vec{x} - \overline{x}(\vec{x}), \min\left(\vec{x} - \underline{x}(\vec{x}), F(\vec{x})\right)\right) \tag{18}$$

It is applicable to (modified) MCPs as well as all special cases thereof and reduces to (15) for a standard LCP. Since we will use it excessively later on, a closer look at the residual (18) is fruitful. Especially some special cases where $\underline{x}(\vec{x})$ is greater or equal to $\overline{x}(\vec{x})$ are of interest. Consider the mixed complementarity condition for one component and its local residual:

$$F_i(\vec{x}) \gtreqless 0 \perp \underline{x}_i \leq x_i \leq \overline{x}_i \tag{19a}$$

$$r_i = \max\left(x_i - \overline{x}_i, \min\left(x_i - \underline{x}_i, F_i(\vec{x})\right)\right) \tag{19b}$$

Assume the regular case where $\underline{x}_i < \overline{x}_i$. The residual should be zero if and only if x_i solves the MCP:

[1] Property of a submatrix, which requires that row i is removed if and only if column i is removed.

[2] The determinant of some square submatrix.

$$
\begin{aligned}
&\text{case } x_i < \underline{x}_i\text{:}\\
&\qquad r_i = \max(\underbrace{x_i - \overline{x}_i}_{<0}, \underbrace{\min\left(x_i - \underline{x}_i, F_i(\vec{x})\right)}_{<0}) < 0\\
&\text{case } x_i = \underline{x}_i\text{:}\\
&\qquad \text{case } F_i(\vec{x}) \geq 0\text{:}\\
&\qquad\qquad r_i = \max\left(\underline{x}_i - \overline{x}_i, \min\left(0, F_i(\vec{x})\right)\right) = \max\left(\underline{x}_i - \overline{x}_i, 0\right) = 0\\
&\qquad \text{case } F_i(\vec{x}) < 0\text{:}\\
&\qquad\qquad r_i = \max\left(\underline{x}_i - \overline{x}_i, \min\left(0, F_i(\vec{x})\right)\right) = \max\left(\underline{x}_i - \overline{x}_i, F_i(\vec{x})\right) \in [\underline{x}_i - \overline{x}_i, 0)\\
&\text{case } \underline{x}_i < x_i < \overline{x}_i\text{:}\\
&\qquad \text{case } F_i(\vec{x}) = 0\text{:}\\
&\qquad\qquad r_i = \max\left(x_i - \overline{x}_i, \min\left(x_i - \underline{x}_i, 0\right)\right) = \max\left(x_i - \overline{x}_i, 0\right) = 0\\
&\qquad \text{case } F_i(\vec{x}) > 0\text{:}\\
&\qquad\qquad r_i = \max\left(x_i - \overline{x}_i, \min\left(x_i - \underline{x}_i, F_i(\vec{x})\right)\right) = \min\left(x_i - \underline{x}_i, F_i(\vec{x})\right) \in (0, \overline{x}_i - \underline{x}_i)\\
&\qquad \text{case } F_i(\vec{x}) < 0\text{:}\\
&\qquad\qquad r_i = \max\left(x_i - \overline{x}_i, \min\left(x_i - \underline{x}_i, F_i(\vec{x})\right)\right) = \max\left(x_i - \overline{x}_i, F_i(\vec{x})\right) \in (\underline{x}_i - \overline{x}_i, 0)\\
&\text{case } x_i = \overline{x}_i\text{:}\\
&\qquad \text{case } F_i(\vec{x}) > 0\text{:}\\
&\qquad\qquad r_i = \max\left(0, \min\left(\overline{x}_i - \underline{x}_i, F_i(\vec{x})\right)\right) = \min\left(\overline{x}_i - \underline{x}_i, F_i(\vec{x})\right) \in (0, \overline{x}_i - \underline{x}_i]\\
&\qquad \text{case } F_i(\vec{x}) \leq 0\text{:}\\
&\qquad\qquad r_i = \max\left(0, \min\left(\overline{x}_i - \underline{x}_i, F_i(\vec{x})\right)\right) = \max\left(0, F_i(\vec{x})\right) = 0\\
&\text{case } x_i > \overline{x}_i\text{:}\\
&\qquad r_i = \max(\underbrace{x_i - \overline{x}_i}_{>0}, \underbrace{\min\left(x_i - \underline{x}_i, F_i(\vec{x})\right)}_{\in\mathbb{R}}) > 0
\end{aligned}
\tag{20}
$$

Though the residual equation was not derived, we can conclude that the residual vanishes if and only if the complementarity conditions hold as expected. Another interesting observation is that the residual is bounded if the unknowns stay within the bounds:

$$
\underline{x}_i \leq x_i \leq \overline{x}_i \quad \rightarrow \quad |r_i| \leq \overline{x}_i - \underline{x}_i \tag{21}
$$

A notable special case occurs if the lower and upper bound coincide, then the affected unknown is fixed but which requirements do we pose for the dependent variable? Assume $\underline{x}_i = \overline{x}_i$ in the following:

$$
\begin{aligned}
&\text{case } x_i = \underline{x}_i = \overline{x}_i\text{:}\\
&\qquad \text{case } F_i(\vec{x}) \geq 0\text{:}\\
&\qquad\qquad r_i = \max\left(0, \min\left(0, F_i(\vec{x})\right)\right) = 0\\
&\qquad \text{case } F_i(\vec{x}) < 0\text{:}\\
&\qquad\qquad r_i = \max\left(0, \min\left(0, F_i(\vec{x})\right)\right) = \max\left(0, F_i(\vec{x})\right) = 0\\
&\qquad \text{The cases where } x_i < \underline{x}_i \text{ and } x_i > \overline{x}_i \text{ are the same as in (20).}
\end{aligned}
\tag{22}
$$

We observe that $F_i(\vec{x})$ is unconstrained since no matter which value the function takes, the residual is 0, which matches the definition of the MCP. Note that it is also compliant with the definition of a VIP with a rectangular constraint set K. If the rectangular region is degenerate, i.e. $\underline{x}_i = \overline{x}_i$ for some i, then the set $S_K(\vec{x})$ (which specifies the vectors which $F(\vec{x})$ can take in order to satisfy the complementarity condition) does not restrict the i-th component of $F(\vec{x})$. This can be easily seen by considering a two dimensional example. The last special case where $\underline{x}_i > \overline{x}_i$ seems to be somewhat artificial or even invalid but commonly occurs for instance in the solution process of modified box LCPs where the bounds depend on the unknowns. Especially if a variable becomes negative momentarily and a box bound depends on that variable (e.g. $\underline{x}_i(\vec{x}) = -\mu x_j$, $\overline{x}_i(\vec{x}) = \mu x_j$), this special case can commence. Therefore we need to ensure that the residual definition makes sense:

$$
\begin{aligned}
&\text{case } x_i < \overline{x}_i\text{:}\\
&\qquad r_i = \max(\underbrace{x_i - \overline{x}_i}_{<0}, \underbrace{\min\left(x_i - \underline{x}_i, F_i(\vec{x})\right)}_{<0}) < 0\\
&\text{case } x_i = \overline{x}_i\text{:}\\
&\qquad r_i = \max(0, \underbrace{\min\left(x_i - \underline{x}_i, F_i(\vec{x})\right)}_{<0}) = 0\\
&\text{case } x_i > \overline{x}_i\text{:}\\
&\qquad r_i = \max(\underbrace{x_i - \overline{x}_i}_{>0}, \underbrace{\min\left(x_i - \underline{x}_i, F_i(\vec{x})\right)}_{\in\mathbb{R}}) > 0
\end{aligned}
\tag{23}
$$

Obviously in the case where upper and lower bounds are interchanged, unknowns which equal the upper bound (without any restrictions on $F_i(\vec{x})$) will be accepted as a solution by the residual. This is probably not what we want and we should make sure no invalid solutions are possible due to this special case. Finally please notice that the claim (21), that the absolute values of the components' residuals are bounded by $\overline{x}_i - \underline{x}_i$ if x_i is within the bounds, extends to the special cases (22) and (23).

3 Contact Modeling

3.1 System Representation

We consider a system of n rigid bodies. All bodies are described by positions and orientations, linear velocities and angular velocities, masses and moment of inertia tensors ("rotational mass"). The positions and orientations are collected in the *generalized coordinates* $q(t)$, whereas the linear and angular velocities are in the *generalized velocities* aggregate $v(t)$. The function $q(t)$ is not smooth since the trajectory of an impacting object has a kink but it is continuous. $v(t)$ in contrast is not even continuous, compare e.g. the pre- and post-impact velocities. Last the mass and inertia tensors are lined up on the diagonal of the *generalized mass matrix* $M(q)$. The translational and rotational quantities introduced so far need three components per body in the three dimensional case. The ordering is so that the translational and rotational quantities alternate and thus all quantities which belong to the same body are ordered consecutively.

The n rigid bodies are potentially in contact at N distinct points. In each contact exactly two bodies are involved and each of them has a point on its surface where the contact force will act on the body. The bodies are denoted $b_{a,(j)}$ and $b_{b,(j)}$. The vectors pointing from the center of mass of body b to contact j will be denoted by $\vec{r}_{b,(j)}$. Any force $\vec{f}_{(j)}$ acting at a contact j will influence the (translational) acceleration and the torque ("rotational acceleration") of both bodies. Contact forces are always given with respect to a body, say $b_{b,(j)}$. In this case the force acts positively on body $b_{b,(j)}$ and with the same magnitude negatively on body $b_{a,(j)}$. Thus the torque of body $b_{b,(j)}$ is $\tau_{b_{b,(j)}} = \vec{r}_{b,(j)} \times \vec{f}_{(j)}$. But the force acting at the center of gravity is directly transfered from the force acting at the contact. This results in the following equations:

$$\begin{aligned} \begin{pmatrix} \vec{f}_{b,(j)} \\ \vec{\tau}_{b,(j)} \end{pmatrix} &= \begin{pmatrix} \mathbf{E} \\ \mathbf{r}^{\times}_{b,(j)} \end{pmatrix} \vec{f}_{(j)} \\ \begin{pmatrix} \vec{f}_{a,(j)} \\ \vec{\tau}_{a,(j)} \end{pmatrix} &= - \begin{pmatrix} \mathbf{E} \\ \mathbf{r}^{\times}_{a,(j)} \end{pmatrix} \vec{f}_{(j)} \end{aligned} \tag{24}$$

Note that the cross product can be written in form of a matrix $\mathbf{r}^{\times}$ since it is a linear operator. Eq. (24) can be used to construct a matrix of all contact conditions depending on the geometric configuration (cf. Sauer and Schömer [19]).

$$\begin{aligned} J(q) &= (\mathbf{J}_{ij})_{i=1\ldots n, j=1\ldots N} \in \mathbb{R}^{6n \times 3N} \\ \mathbf{J}_{ij} &= \begin{cases} -\begin{pmatrix} \mathbf{E} \\ \mathbf{r}^{\times}_{a,(j)} \end{pmatrix} & \text{if } i = b_{a,(j)} \\ \begin{pmatrix} \mathbf{E} \\ \mathbf{r}^{\times}_{b,(j)} \end{pmatrix} & \text{if } i = b_{b,(j)} \\ \mathbf{0} & \text{else} \end{cases} \end{aligned} \tag{25}$$

The transpose of the mapping $J(q)$ can be used to translate from body quantities back to relative contact quantities. Therefore we can use $J(q)^T M(q)^{-1} J(q)$ to translate from contact

forces to forces and torques acting at the center of gravity of the bodies, which in turn are translated into translational and rotational accelerations of the bodies by the inverse mass matrix and last the body accelerations are translated back to relative accelerations at the contacts.

3.2 Contact Gaps

In each potential contact j two points $\vec{p}_{a,(j)}$ and $\vec{p}_{b,(j)}$ are involved, one on each of the surfaces of the contacting bodies. If the two points are closest points (in the case of differentiable surfaces), then the contact normals are precisely in the opposite direction and we can set up a local coordinate frame at each of the contacts attached to one of the two points involved. By convention we attach the coordinate frame to the first body ($b_{a,(j)}$) of each collision and the collision normal therefore points from the first to the second body and is denoted by $\vec{n}_{(j)}$. The tangential plane is spanned by the vectors $\vec{t}_{x,(j)}$ and $\vec{t}_{y,(j)}$, which are chosen orthogonally to each other and the normal vector. Fig. 3 depicts the local coordinate frame for a two dimensional example with a smooth surface. In practice neither the surfaces are smooth, nor are the points closest and the collision detection just provides some seemingly suitable contact normal.

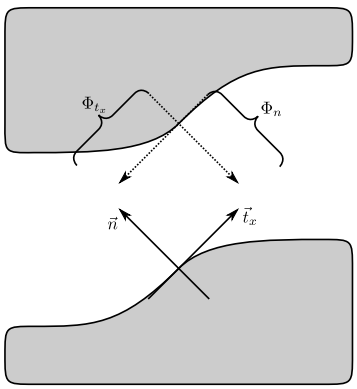


Figure 3: Quantities involved at a contact gap.

We can express the distance of the two points at the contact j in the local coordinate frame in terms of the normal and tangential components using an orthogonal matrix.

$$\Phi(q)_{(j)} = \underbrace{\begin{pmatrix}\vec{n}_{(j)} & \vec{t}_{x,(j)} & \vec{t}_{y,(j)}\end{pmatrix}^T}_{\mathbf{Q}_{(j)}{}^T} \left(\vec{p}_{b,(j)} - \vec{p}_{a,(j)}\right) = \mathbf{Q}_{(j)}^T \left(\mathbf{J}^T q\right)_{(j)} \tag{26}$$

Let us denote the aggregate of all those distances by $\Phi(q)$ and refer to it as the *gap function* $\Phi(q) = \mathbf{Q}^T \mathbf{J}^T q : \mathbb{R}^{6n} \to \mathbb{R}^{3N}$.

3.3 Non-penetration Constraints

As soon as bodies get into contact, forces must act to prevent the bodies from interpenetrating. Those forces act in normal direction. They may only act if the two bodies are actually in contact and must be repulsive but not adhesive in nature. For that reason we need to restrict the normal component of the *gap function* $\Phi_n(q)$ and the normal component of the contact

force to be non-negative and complementary to each other. That is either one must be zero. Note that if $\Phi_n(q)$ is positive, then the bodies are separated, if it is zero, then the bodies touch each other and if it is negative, then the bodies interpenetrate. The first and most important complementarity condition can be written down straight forward for any point in time t, by denoting the function providing the aggregate of all contacts' normal forces by $f_n : \mathbb{R} \to \mathbb{R}^N$:

$$f_n(t) \geq \vec{0} \perp \Phi_n(q(t)) \geq \vec{0} \tag{27}$$

The function Φ_n is usually assumed to be differentiable, though this is seldom the case in slightly more complex scenarios [1, section 2.2]. Then we can use a Taylor-expansion of Φ_n around the current geometric configuration, in order to approximate the distance function at the geometric configuration in the next time step:

$$\begin{aligned} &\Phi_n(q(t+\Delta t)) = \Phi_n(q(t)) + D\Phi_n(q(t))\Delta q + O(\Delta q^2) \\ &\text{where } \Delta q = q(t+\Delta t) - q(t) \end{aligned} \tag{28}$$

The non-penetration constraint can now be expressed for the next time step:

$$\int_t^{t+\Delta t} f_n(\tau)\,\mathrm{d}\tau \geq \vec{0} \perp \Phi_n(q(t+\Delta t)) \approx \Phi_n(q(t)) + D\Phi_n(q(t))\Delta q \geq \vec{0} \tag{29}$$

On the left-hand-side of the complementarity condition, we have a force integral over the current time step which will later become part of the final complementarity problem unknowns. However, the right-hand-side must still be expressed in terms of the unknowns. First we will express Δq in terms of the average velocity $\bar{v}$ or respectively the average acceleration $\bar{a}$ within the time interval $[t, t+\Delta t]$:

$$\begin{aligned} \Delta q = \bar{v}\Delta t &= \int_t^{t+\Delta t} v(\tau)\,\mathrm{d}\tau = \int_t^{t+\Delta t} v(t) + \bar{a}(\tau - t)\,\mathrm{d}\tau \\ &= v(t)\Delta t + \bar{a}\int_t^{t+\Delta t} \tau - t\,\mathrm{d}\tau = v(t)\Delta t + \frac{1}{2}\bar{a}\Delta t^2 \\ &= v(t)\Delta t + \frac{1}{2}\bar{a}\Delta t^2 \end{aligned} \tag{30}$$

The last step is to derive the link between the force integral and the average acceleration: In classical mechanics the momentum is directly proportional to the velocity. The time derivative leads to the expression for the force:

$$f(t) = \frac{\mathrm{d}}{\mathrm{d}t} M(q(t))v(t) = M(q(t))\dot{v}(t) + \dot{M}(q(t))v(t) \tag{31}$$

Assuming the mass to be constant and integrating over the force for a single time step leads to:

$$\int_t^{t+\Delta t} f(\tau)\,\mathrm{d}\tau = \int_t^{t+\Delta t} M(q(\tau))\dot{v}(\tau) + \dot{M}(q(\tau))v(\tau)\,\mathrm{d}\tau \overset{M(q)\text{ const}}{=} \mathbf{M}\int_t^{t+\Delta t} \dot{v}(\tau)\,\mathrm{d}\tau = \mathbf{M}\bar{a}\Delta t \tag{32}$$

Note that the function f denotes body forces. However, the complementarity conditions restrict the forces acting at the contacts (contact forces) and thus the contact force integrals will be part of the unknowns. Consequently we break the integral apart into contact forces $f_c : \mathbb{R} \to \mathbb{R}^{3N}$, external forces $f_{ext} : \mathbb{R} \to \mathbb{R}^{6n}$ and inertial forces $f_{in} : \mathbb{R} \to \mathbb{R}^{6n}$:

$$\begin{aligned}
\int_t^{t+\Delta t} f(\tau)\,\mathrm{d}\tau &= \int_t^{t+\Delta t} J(q(\tau))f_c(\tau) + f_{ext}(\tau) + f_{in}(\tau)\,\mathrm{d}\tau \\
&= (\bar{f}_{ext} + \bar{f}_{in})\Delta t + \int_t^{t+\Delta t} J(q(\tau))f_c(\tau)\,\mathrm{d}\tau \\
&\overset{J(q)=\mathbf{J}}{=} (\bar{f}_{ext} + \bar{f}_{in})\Delta t + \mathbf{J}\int_t^{t+\Delta t} f_c(\tau)\,\mathrm{d}\tau
\end{aligned} \tag{33}$$

As you can see the operator J is assumed to be constant over the time step in the first argument, which is again a simplification. The contact forces f_c are expressed in the global coordinate frame, however, our non-penetration constraint poses a condition on the normal component, i.e. the contact force pointing in the normal direction. Later the friction constraints pose conditions on the tangential components. Consequently the problem variables will be the normal and tangential force integrals which will be mapped to the total contact force integral using the orthogonal rotation matrix $\mathbf{Q}$ which changes from the local coordinate frame to the global coordinate frame. Again this is a simplification since actually $\mathbf{Q}$ depends on the geometric configuration q just as $\mathbf{J}$ does.

$$\int_t^{t+\Delta t} f_{c,(j)}(\tau)\,\mathrm{d}\tau = \int_t^{t+\Delta t} \mathbf{Q}_{(j)} \left(f_{n,(j)}(\tau), f_{t_x,(j)}(\tau), f_{t_y,(j)}(\tau)\right)^T \mathrm{d}\tau = \mathbf{Q}_{(j)}\vec{x}_{(j)} \tag{34}$$

Though (34) is formulated for a single contact j this translates straight forward to the multi-contact case where $\mathbf{Q}$ is 3×3 block diagonal matrix and $\vec{x}$ is the aggregate of all contacts' components where the components of a single contact are always grouped together. Now we are ready to substitute. We insert (34) into (33), (33) into (32), (32) into (30) and finally (30) into (29). Dividing by Δt leads to:

$$\vec{x}_n \geq \vec{0} \perp \frac{\Phi_n(q(t))}{\Delta t} + D\Phi_n(q(t)) \left(v(t) + \frac{1}{2}\mathbf{M}^{-1}\left((\bar{f}_{ext} + \bar{f}_{in})\Delta t + \mathbf{J}\mathbf{Q}\vec{x}\right)\right) \geq \vec{0} \tag{35}$$

Note that $(\cdot)_n$ selects the normal components from the operand which is actually a projection matrix applied to the argument. Analogously $(\cdot)_{t_x}$, $(\cdot)_{t_y}$ and $(\cdot)_t$ extract the first tangential, the second tangential and both tangential components respectively. We still need to discuss the role of $\Phi_n(q(t))$ and $D\Phi_n(q(t))$. The first is the current contact distance. Intuitively if a contact is generated before the impact occurs, the contact points can still approach a small distance until the impact occurs and the complementarity condition holds. Since we assume contacts to be generated only if two bodies are almost in contact or already in contact we will ignore this term. We now turn our attention to $D\Phi_n(q(t))$. $D\Phi_n(q(t))$ is the Jacobian of Φ_n at the configuration $q(t)$ and intuitively the directions in which the geometric configuration must change in order for the distance at the contact gap to increase:

$$D\Phi_n(q(t)) = \left(\mathbf{Q}^T\mathbf{J}^T\right)_n \tag{36}$$

Generalized coordinates are translated into relative contact coordinates by $\mathbf{J}^T$ and these in turn are translated to local coordinates by $\mathbf{Q}^T$. Finally the normal components give the direction where the distance increases strongest. As before $\mathbf{J}$ and $\mathbf{Q}$ actually depend on the geometric configuration but are assumed to be constant over the time step. The non-penetration constraint can be further simplified and reorganized to:

$$(\vec{x})_n \geq \vec{0} \perp \left(\mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q}\vec{x} + \mathbf{Q}^T\mathbf{J}^T\left(2v(t) + \Delta t\mathbf{M}^{-1}(\bar{f}_{ext} + \bar{f}_{in})\right)\right)_n \geq \vec{0} \tag{37}$$

3.4 Friction Constraints

The most prominent model for dry (unlubricated) friction used in engineering is the Coulomb model [1]. It states that the magnitude of the friction force is bounded by the coefficient of friction μ times the normal force. The friction force acts in the tangential plane of each contact. Thus the friction vector at a contact is bounded by a circular disc with a radius proportional to the normal force at the contact:

$$\sqrt{f_{t_x,(j)}(t)^2 + f_{t_y,(j)}(t)^2} \leq \mu_{(j)} f_{n,(j)}(t) \tag{38}$$

Another condition is that the friction force must oppose sliding or prevent it if sliding is imminent. Sliding is to be understood as the relative contact velocity. Analogously to the non-penetration case we can use a Taylor-expansion to get an approximation for the distance (this time tangential) of the points involved in a contact.

$$\begin{aligned} &\Phi_t(q(t+\Delta t)) = \Phi_t(q(t)) + D\Phi_t(q(t))\Delta q + O(\Delta q^2) \\ &\text{where } \Delta q = q(t+\Delta t) - q(t) \end{aligned} \tag{39}$$

The derivation is the same as before and leads to a term describing the tangential distance in the next time step:

$$\Phi_t(q(t+\Delta t)) \approx \left(\mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q}\vec{x} + \mathbf{Q}^T\mathbf{J}^T\left(2v(t) + \Delta t\mathbf{M}^{-1}(\bar{f}_{ext} + \bar{f}_{in})\right)\right)_t \tag{40}$$

As noted earlier the tangential contact forces should do their utmost to oppose sliding. This is known as the *maximum dissipation principle* and can be expressed as an optimization condition or as direct constraints:

$$f_t(t) = \underset{\substack{\|f_{t,(j)}(t)\|_2 \leq \mu_{(j)} f_{n,(j)}(t) \\ f_{t,(j)}(t)^T \vec{n}_{(j)} = 0 \\ \forall j \in \{1,\ldots,N\}}}{\operatorname{argmin}} \Phi_t(q(t))^T f_t(t) \tag{41a}$$

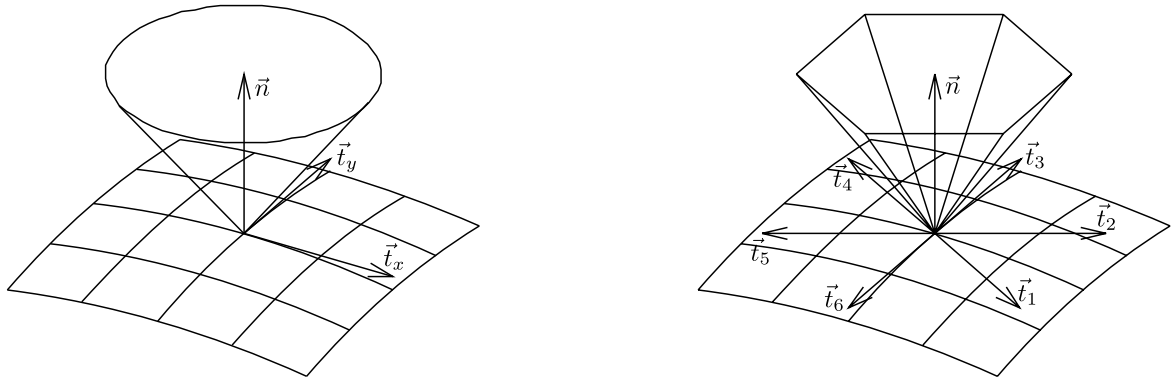
$$\begin{aligned} &\Phi_{t,(j)}(q(t))^T f_{t,(j)}(t) = -\|\Phi_{t,(j)}(q(t))\|_2 \|f_{t,(j)}(t)\|_2 \\ &\|\Phi_{t,(j)}(q(t))\|_2 \perp \mu_{(j)} f_{n,(j)}(t) - \|f_{t,(j)}(t)\|_2 \geq 0 \qquad \forall j \in \{1,\ldots,N\} \end{aligned} \tag{41b}$$

The constraints (41) are used seldom in practice since the underlying problem is too hard to solve. Instead we will introduce a simplified Coulomb law before stating the final complementarity problem.

3.5 Friction Cones

The Coulomb law as stated above demands that each contact force must be within a *circular friction cone* and additionally if the friction force is bounded by the boundary of the friction cone, then the friction force must be directly opposed to the sliding. A popular approximation to this scheme is a *polyhedral friction cone* [20] [3]. There the tangential plane is spanned by several vectors. Usually the number of vectors is even and each vector has also an opposing vector. In the case of sliding, the friction force must point in the direction of the vector which opposes the sliding best. Sometimes degenerate cases exists where the friction force is allowed to be distributed to two neighboring vectors spanning the friction cone. Fig. 4 shows a circular friction cone and its polyhedral approximation with six facets. The vectors usually span the tangential plane uniformly:

$$\vec{t}_{i,(j)} = \cos\left(\frac{i}{\eta}2\pi\right)\vec{t}_{x,(j)} + \sin\left(\frac{i}{\eta}2\pi\right)\vec{t}_{y,(j)} \qquad i \in \{1,\ldots,\eta\} \tag{42}$$



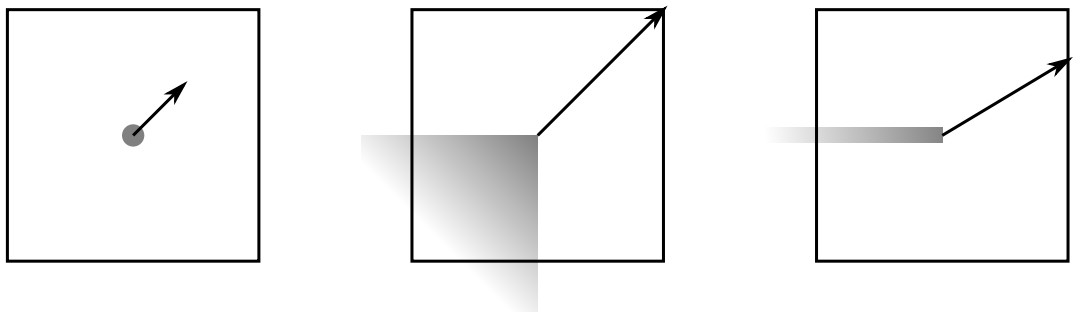
(a) The circular friction cone.

(b) The polyhedral friction cone.

Figure 4: The circular friction cone and a polyhedral approximation.

In this thesis we mostly work with a *pyramidal approximation* where the number of facets η is four. The reasoning behind it is that the matrix of the resulting complementarity problem will have desirable properties like positive-semi-definiteness and symmetry. In the general case of a polyhedral friction cone, additional constraints and variables need to be introduced to restrict the total friction force to the friction cone and to align the friction force if sliding is present. Both properties will come for free in the pyramidal case as we will see. Fig. 5 shows pyramidal friction cones. If the friction force is in the interior of the friction cone (cf. Fig. 5(a)), then no sliding is present. If the sliding is strictly inside one of the quadrants (cf. Fig. 5(b)), then the friction force is constrained to be at the opposite corner. However, if the sliding is on one of the axis (cf. Fig. 5(c)), then the friction force is a weighted average of the two opposite corners. This observation is important because it defines the *dissipativity* of our scheme. Unfortunately the friction force will not oppose the sliding directly, but it will oppose the sliding at least partially ($(\Phi_{t,(j)}(q(t+\Delta t))^T\vec{x}_{t,(j)} \leq 0$). As an example of the consequences consider a box sliding down a ramp. If the tangential vectors are not aligned explicitly, the boxes will start to rotate. The corresponding complementarity constraints are straightforward:

$$
\begin{aligned}
\Phi_{t_x}(q(t+\Delta t)) \gtreqless \vec{0} \perp \; -\mu\,(\vec{x})_n \leq (\vec{x})_{t_x} \leq \mu\,(\vec{x})_n \\
\Phi_{t_y}(q(t+\Delta t)) \gtreqless \vec{0} \perp \; -\mu\,(\vec{x})_n \leq (\vec{x})_{t_y} \leq \mu\,(\vec{x})_n
\end{aligned}
\tag{43}
$$



(a) If the friction force stays inside the friction cone, then sliding will be prevented.

(b) If the friction force resides on an edge of the friction cone, then the sliding will be in the opposite quadrant.

(c) If the friction force resides on a face of the friction cone, then the sliding will be on the opposite axis.

Figure 5: Relation between friction force and sliding.

Eq. (43) forms a complementarity problem with upper and lower bounds on the unknowns and where the function applied to the unknowns is affine (cf. Eq. (40)). It is not a box LCP since the limits depend on the unknowns, i.e. on the normal components in this case. As introduced before I will call it a modified box LCP. Though the two friction components are treated independently, it is still assured that the total friction force is within the pyramidal friction cone. In the general case of a polyhedral friction cone treating the components separately would be not sufficient since the sum of all components would not be constrained to the friction cone. Another observation is that this pyramidal friction cone circumscribes the circular friction cone.

As an example consider the case where the tangential x-component is at the upper bound. Consequently sliding in the x-direction cannot be prevented anymore and $\Phi_{t_x}(q(t+\Delta t))$ is constrained to be non-positive by the complementarity condition. The y-component is completely independent. Say the y-component is determined to stay within the bounds, so that the slip in the y-direction is prevented, then the total friction force is on the face in the positive x-direction of the friction cone and the slip is along the negative x-axis.

3.6 System Properties

3.6.1 Contact Problem

Let us put all the pieces together. We take the non-penetration complementarity constraint (37), the friction complementarity constraints (43) and combine them to a single modified box-constrained linear complementarity problem:

$$
\begin{aligned}
&\mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q}\vec{x} + \mathbf{Q}^T\mathbf{J}^T\left(2v(t) + \Delta t\mathbf{M}^{-1}(\bar{f}_{ext} + \bar{f}_{in})\right) \gtreqless \vec{0} \perp \underline{x}(\vec{x}) \leq \vec{x} \leq \overline{x}(\vec{x}) \\
&\text{where} \quad \underline{x}(\vec{x}) = \left(0,\ -\mu_{(1)}x_1,\ -\mu_{(1)}x_1,\ \ldots,\ 0,\ -\mu_{(N)}x_{3N-2},\ -\mu_{(N)}x_{3N-2}\right)^T \\
&\qquad\qquad \overline{x}(\vec{x}) = \left(\infty,\ +\mu_{(1)}x_1,\ +\mu_{(1)}x_1,\ \ldots,\ \infty,\ +\mu_{(N)}x_{3N-2},\ +\mu_{(N)}x_{3N-2}\right)^T
\end{aligned} \tag{44}
$$

3.6.2 Matrix Properties

It is important to investigate system (44) thoroughly because only by exploiting specific problem properties we will be able to solve the complementarity problem efficiently, since the general LCP is NP-complete. The easiest and most important observations are that the matrix $\mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q}$ is symmetric and positive-semi-definite (PSD). The essential prerequisites are that the mass matrix $\mathbf{M}$ is symmetric and positive-definite (PD). Since the mass matrix is a 3×3 block diagonal matrix with 3×3 identity matrices multiplied by the appropriate body mass respectively and inertia tensors, we can conclude that the mass matrix is symmetric and PD because its building blocks share these properties. Both properties transfer to the inverse:

$$
\left(\mathbf{M}^{-1}\right)^T = \left(\mathbf{M}^T\right)^{-1} \overset{\mathbf{M}=\mathbf{M}^T}{=} \mathbf{M}^{-1} \tag{45a}
$$

$$
\vec{x}^T\mathbf{M}^{-1}\vec{x} \overset{\vec{x}=\mathbf{M}\vec{y}}{=} (\mathbf{M}\vec{y})^T\mathbf{M}^{-1}(\mathbf{M}\vec{y}) = \vec{y}^T\mathbf{M}^T\vec{y} \overset{\mathbf{M}=\mathbf{M}^T}{=} \vec{y}^T\mathbf{M}\vec{y} \overset{\mathbf{M}\text{ is PD}}{>} 0 \tag{45b}
$$

The symmetry and PSD-ness of the system matrix follow easily:

$$
\left(\mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q}\right)^T = \mathbf{Q}^T\mathbf{J}^T\left(\mathbf{M}^{-1}\right)^T\mathbf{J}\mathbf{Q} = \mathbf{Q}^T\mathbf{J}^T\left(\mathbf{M}^T\right)^{-1}\mathbf{J}\mathbf{Q} \overset{\mathbf{M}=\mathbf{M}^T}{=} \mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q} \tag{46a}
$$

$$
\vec{x}^T\mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q}\vec{x} = (\mathbf{J}\mathbf{Q}\vec{x})^T\mathbf{M}^{-1}(\mathbf{J}\mathbf{Q}\vec{x}) \overset{\mathbf{M}^{-1}\text{ is PD}}{\geq} 0 \tag{46b}
$$

Please note that the system matrix is PSD instead of PD. This is because the matrix $\mathbf{JQ}$ can potentially map non-zero vectors to zero vectors which in consequence will lead to an overall zero result though the input vector is non-zero which in turn contradicts PD-ness.

Though symmetry and PD-ness implicate non-singularity, symmetry and PSD-ness are not sufficient. A simple example shows that redundant constraints cause the problem. Consider a box resting on a plane: At each of the corners which touch the ground, a contact point has to be generated because it is not clear in advance, which of them can be ignored. We will only consider the frictionless case. The normal components must add up to the weight of the box to prevent it from falling through the ground. However, the distribution is variable, merely the normal components of opposite corners must be equal, so that no rotation sets in. The existence of infinite solutions shows, that the matrix is possibly singular. However, the matrix is *underdetermined*, i.e. we pose too few independent constraints in order for the solution to be unique.

As with any singular matrix the question of solution existence arises. For the sake of clarity I substitute $\mathbf{P} = \mathbf{JQ}$ into the system. Let us have a look at the linear system of equations corresponding to (44).

$$
\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}\vec{x} = \mathbf{P}^T\vec{b} \tag{47}
$$

The system is solvable if the right-hand-side is in the range space of the matrix on the left-hand-side. However, apparently the right-hand-side is in the range space of $\mathbf{P}^T$. Consequently the system is only solvable if the range space of $\mathbf{P}^T$ is a subset of the range space of the system matrix. In fact the range spaces are equal.

Proof. $\operatorname{range}\mathbf{P}^T \supseteq \operatorname{range}\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}$. This is obvious because the additional mapping $\mathbf{M}^{-1}\mathbf{P}$ potentially restricts the range space of $\mathbf{P}^T$.

$\operatorname{null}\mathbf{P} \subseteq \operatorname{null}\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}$. This is also obvious because the additional mapping $\mathbf{P}^T\mathbf{M}^{-1}$ potentially extends the null space of $\mathbf{P}$.

$\operatorname{range}\mathbf{P} = \left(\operatorname{null}\mathbf{P}^T\right)^{\perp}$. This is a well known relationship.

$\operatorname{range}\mathbf{P}^T \subseteq \operatorname{range}\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}$. To see this we need the subset relationship of the null spaces above and the relationship of the range space and the orthogonal complement of the null space above.

$\operatorname{range}\mathbf{P}^T = (\operatorname{null}\mathbf{P})^{\perp} \subseteq \operatorname{null}\left(\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}\right)^{\perp} = \operatorname{range}\left(\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}\right)^T = \operatorname{range}\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}$

$\Rightarrow \operatorname{range}\mathbf{P}^T = \operatorname{range}\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}$ □

Since the range spaces are equal, the solvability follows as mentioned above. Thus the system is *consistent* though it is singular.

$$\mathbf{P}^T\vec{b} \in \operatorname{range}\mathbf{P}^T = \operatorname{range}\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P} \quad \Rightarrow \quad \text{Eq. (47) solvable} \tag{48}$$

Though we do know now, that the LSE (47) is solvable, we cannot reason that the complementarity problem is solvable. However, we will see in section 3.7.3, that the complementarity problem can be reformulated into an LCP, which is of the same type as the one used in Anitescu and Potra [3]. It was proofed in Anitescu and Potra [3], that the LCP is always solvable. Furthermore we know that the matrix $\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}$ is symmetric and PSD. It is elaborated in Cottle et al. [8], section 3.4, that $\mathbf{A}\vec{x} + \vec{b}$ is unique instead of the solution. In the context of the contact problems, this means that the relative contact velocities are unique though the contact forces aren't.

3.6.3 Entry Properties

Since the system matrix is PSD, unlike with PD matrices, the diagonal entries could be possibly zero. However, a closer look at the diagonal blocks reveals, that they are PD, and thus every diagonal entry is indeed positive and not only non-negative. $(\cdot)_{(j)(j)}$ denotes the 3×3 block corresponding to the influence of contact j on itself. The subscripts a and b identify the bodies involved at the contact in question. The diagonal block at contact j can be calculated to be the following:

$$\begin{aligned}\left(\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\right)_{(j)(j)} &= -\vec{r}_a \times \mathbf{I}_a^{-1} \times \vec{r}_a + m_a^{-1}\mathbf{E}_3 - \vec{r}_b \times \mathbf{I}_b^{-1} \times \vec{r}_b + m_b^{-1}\mathbf{E}_3 = \\ &\underbrace{\left(m_a^{-1} + m_b^{-1}\right)\mathbf{E}}_{PD} - \underbrace{\mathbf{r}_a^{\times}\mathbf{I}_a^{-1}\mathbf{r}_a^{\times}}_{ND} - \underbrace{\mathbf{r}_b^{\times}\mathbf{I}_b^{-1}\mathbf{r}_b^{\times}}_{ND}\end{aligned} \tag{49}$$

Note that the ND-ness is due to the antisymmetry ($(\mathbf{r}^{\times})^T = -(\mathbf{r}^{\times})$) of the crossproduct:

$$\vec{x}^T \mathbf{r}^\times \mathbf{I}^{-1} \mathbf{r}^\times \vec{x} = \left((\mathbf{r}^\times)^T \vec{x}\right)^T \mathbf{I}^{-1} \left(\mathbf{r}^\times \vec{x}\right) = -\left(\mathbf{r}^\times \vec{x}\right)^T \mathbf{I}^{-1} \left(\mathbf{r}^\times \vec{x}\right) \overset{\mathbf{I}^{-1} \text{ is PD}}{<} 0 \tag{50}$$

Note that the multiplication of the additional orthogonal matrix $\mathbf{Q}$ does not influence the result.

So the diagonal blocks are symmetric and PD, but what about the off-diagonal elements? Consider a simplified example where the center of mass of an unspecified rigid body is located at the origin. Two point masses are in contact at contact points $\vec{r}_a$ and $\vec{r}_b$ on the x-axis. The bodies are at rest and no external or inertial forces act on them. The mass of the central body is m_t and the inertia tensor is a scaled identity matrix $\mathbf{I} = m_r \mathbf{E}_3$. Then the relative contact velocity at contact b is the following:

$$\begin{aligned}
\Delta\vec{v}_b &= \left(-\mathbf{r}_b^\times \mathbf{I}^{-1} \mathbf{r}_a^\times + m_t^{-1}\mathbf{E}_3\right) \vec{f}_a + \left(-\mathbf{r}_b^\times \mathbf{I}^{-1} \mathbf{r}_b^\times + m_t^{-1}\mathbf{E}_3\right) \vec{f}_b \\
&= \left(-m_r^{-1} \mathbf{r}_b^\times \mathbf{r}_a^\times + m_t^{-1}\mathbf{E}_3\right) \vec{f}_a + \left(-m_r^{-1} \mathbf{r}_b^\times \mathbf{r}_b^\times + m_t^{-1}\mathbf{E}_3\right) \\
&= \begin{pmatrix} m_t^{-1} & & \\ & r_a r_b m_r^{-1} + m_t^{-1} & \\ & & r_a r_b m_r^{-1} + m_t^{-1} \end{pmatrix} \vec{f}_a + \\
&\quad \begin{pmatrix} m_t^{-1} & & \\ & r_b^2 m_r^{-1} + m_t^{-1} & \\ & & r_b^2 m_r^{-1} + m_t^{-1} \end{pmatrix} \vec{f}_b
\end{aligned} \tag{51}$$

To keep things simple let us choose $m_t = \infty$ and $m_r = 1$, then we get the diagonal entry r_b^2 for the y- and z-component and $r_a r_b$ as an off-diagonal entry. Depending on where we put the contact points on the x-axis, we can generate arbitrary large or small off-diagonal elements. For example the parameters $r_a = -10$ and $r_b = 1$ give a diagonal entry of 1 vs. an off-diagonal element of -10.

3.6.4 Sparsity

The matrix $\mathbf{Q}^T \mathbf{J}^T \mathbf{M}^{-1} \mathbf{J} \mathbf{Q}$ is sparse. In the following we will describe the structure of the sparsity and a suitable datastructure. First of all a very useful helper structure is the *contact graph*. This graph contains the bodies as nodes. If a contact is generated between two bodies, then an (undirected) edge is inserted inbetween the corresponding nodes. The graph can be constructed efficiently while performing the collision detection. One of the advantages of such a graph is that it is easy to split the contact problem into smaller independent problems. Every connected component of the graph is an independent problem because obviously the components do not have a single contact in common. However, we will use it to give an upper bound on the number of non-zero entries in a row of our matrix. Let us consider contacts instead of matrix entries first. A contact is influenced by all neighboring contacts. The number of neighboring contacts is bounded above by the sum of the degrees of the nodes connected by the edge corresponding to the contact. All edges between the two nodes involved in the collision are counted doubly this way. We can safely subtract one from the upper bound because we have at least one contact between the two bodies involved. Since for each contact we have three quantities, we will have at most three times the upper bound non-zero entries for each of the three rows corresponding to the contact. Note that we probably overestimated the number of contacts and the number of non-zero entries per contact. In particular if there

are several edges between the same bodies those will lead to overestimation and furthermore if the inertia tensors are only diagonal (like for spheres), then we will have only one non-zero entry per contact in each row instead of three. In any case the upper bound is fairly tight and very cheap to compute. It can be used to allocate enough entries for each row in a compressed-row-storage (CRS) matrix before the construction begins.

CRS matrices keep all non-zero entries in an array, so that the non-zero entries of each row are stored consecutively. For each entry the original column index is remembered. An additional array is used to store the array index of the first entry in each row. The implementation details vary. An interessting possibility is to store blocks instead of the matrix entries directly. If the quantities of each contact are properly grouped, then the non-zero entries occur in 3×3 blocks. However, a disadvantage of this is, that we will store the zero entries in each block too and this in turn will have an impact on the performance later, especially if we intend to simulate a lot of spheres where the blocks have diagonal structure, because a lot of unnecessary multiplications by zero will be performed. That's the main reason why we sticked to a CRS matrix with single entries.

The construction of the matrix can be done in such a way, that symmetric blocks are inserted the first time they are computed and so that each row of the CRS matrix fills from left to right. The last property is important because otherwise costly insertion operations are needed. The construction is done contactwise similar as in Preclik [14]. Thus always three rows are filled together. The set of neighboring contacts (without duplicates) has to be determined from the contact graph first. Only the upper-right triangle of the matrix is actually computed, the other half is filled as the data accrues. Thus we use a binary search first to find the current contact in its own set of neighboring contacts and compute all blocks corresponding to contacts with a higher (or equal) index in ascending order. The three rows are filled block by block from left to right this way. Each block is transposed and stored at the symmetric position. All rows fill from left to right and data is always appended to the rows of the CRS matrix, in particular no costly insertion operations are needed.

3.7 Reformulations and Simplifications

3.7.1 Frictionless Problems

In section 3.6.1 the contact problem for pyramidal friction cones was established. The system can be easily reformulated into a standard LCP if there is no friction at all. In this case all coefficients of friction are zero and thus all tangential components are constrained to be zero as well. Consequently all corresponding unknowns can be removed. Because the tangential complementarity conditions are always fulfilled they can also be removed. The reduced problem results in the following standard LCP:

$$\mathbf{N}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{N}\vec{x} + \mathbf{N}^T\mathbf{J}^T\left(2v(t) + \Delta t\mathbf{M}^{-1}(\bar{f}_{ext} + \bar{f}_{in})\right) \geq \vec{0} \perp \vec{x} \geq \vec{0} \tag{52}$$

3.7.2 Friction Boxes

Normally the contact forces are restricted to reside within a circular or polyhedral *cone*. The problem with that is, that the friction forces are coupled to the normal forces. If the normal forces are larger, than also the friction forces may be larger. One can address this problem by estimating the normal forces à priori. This allows us to keep the bounds on the friction forces

fixed, i.e. not coupled to the normal forces. What we actually do is replace the friction cone by a friction cylinder or in the case of a pyramidal friction cone by a (axis-aligned) box. The guessing of the normal forces can be done for example by exploiting the time coherence, which means that we can often expect the contact forces to be of similar magnitude in successive time steps. By caching the contact problem results from the last time step and using the previous normal forces as an initial guess, box friction might give accurate results. Alternatively we could approximately solve a frictionless problem and use the normal forces thus obtained as an initial guess. In any case the contact problem (44) reduces to an (unmodified) box-constrained LCP.

$$
\begin{aligned}
&\mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q}\vec{x} + \mathbf{Q}^T\mathbf{J}^T\left(2v(t) + \Delta t\mathbf{M}^{-1}(\bar{f}_{ext} + \bar{f}_{in})\right) \gtreqless \vec{0} \perp \underline{\vec{x}} \leq \vec{x} \leq \overline{\vec{x}} \\
&\text{where} \quad \underline{\vec{x}} = \left(0,\ -\mu_{(1)}\tilde{x}_1,\ -\mu_{(1)}\tilde{x}_1,\ \ldots,\ 0,\ -\mu_{(N)}\tilde{x}_{3N-2},\ -\mu_{(N)}\tilde{x}_{3N-2}\right)^T \\
&\qquad\quad \overline{\vec{x}} = \left(\infty,\ +\mu_{(1)}\tilde{x}_1,\ +\mu_{(1)}\tilde{x}_1,\ \ldots,\ \infty,\ +\mu_{(N)}\tilde{x}_{3N-2},\ +\mu_{(N)}\tilde{x}_{3N-2}\right)^T \\
&\qquad\quad \tilde{x} \in \mathbb{R}^{3N} \text{ is a guess of the contact forces}
\end{aligned}
\tag{53}
$$

3.7.3 LCP Formulation

The modified box-constrained formulation (44) can be reformulated into a standard LCP. In fact this is the more popular formulation and was introduced in Stewart and Trinkle [20], taken up and enhanced by many authors [3] [19]. The LCP is provably solvable by the Lemke algorithm. Unfortunately the Lemke algorithm, as well as other direct solvers, suffers from two major drawbacks [8, chapter 5]. First of all they require $\mathcal{O}(n^2)$ data storage and thus cannot exploit the sparsity property of the contact problem. Indeed the sparsity is destroyed after only a few pivot steps. The second problem with those methods is, that round-off errors "can cause severe numerical problems" [8]. For instance round-off errors can evoke an incorrect pivot choice. In particular the Lemke algorithm can enter infinite loops because the degeneracy resolution breaks down and the algorithm cycles through the same pivot steps over and over again. Iterative solvers in contrast can use sparse datastructures like the CRS matrix format to avoid quadratic storage costs. Furthermore they are "much less sensitive to round-off errors" [8]. The problem with the LCP formulation is that it is asymmetric and exhibits zeros on the diagonal. This is caused by additional variables and constraints to align the friction force in a slip condition and to restrict the total friction force to the cone. Let us have a closer look at the arising LCP matrix. To avoid the introduction of more identifiers, we stick to the matrix $\mathbf{Q}$. However, the matrix now contains not only the normal and two tangential vectors per contact but usually an even number of tangential vectors uniformly spanning the tangential plane as for example defined in (42). Thus the number of variables for the friction forces is η. Furthermore an additional helper variable is needed for each contact (corresponding to the last column in the following blocks):

$$
\begin{aligned}
\mathbf{A}_{(i)(i)} &= \begin{pmatrix} & & & & 0 \\ & & & & 1 \\ & \left(\mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q}\right)_{(i)(i)} & & & \vdots \\ & & & & 1 \\ \mu_{(i)} & -1 & \dots & -1 & 0 \end{pmatrix} \qquad \forall i \in \{1,\dots,N\} \\
\mathbf{A}_{(i)(j)} &= \begin{pmatrix} & & & & 0 \\ & & & & 0 \\ & \left(\mathbf{Q}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{Q}\right)_{(i)(j)} & & & \vdots \\ & & & & 0 \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix} \qquad \forall i,j \in \{1,\dots,N\}, i \neq j \\
\vec{b}_{(i)} &= \begin{pmatrix} \left(\mathbf{Q}^T\mathbf{J}^T\left(2v(t) + \Delta t\mathbf{M}^{-1}(\bar{f}_{ext} + \bar{f}_{in})\right)\right)_{(i)} \\ 0 \end{pmatrix} \qquad \forall i \in \{1,\dots,N\}
\end{aligned}
\tag{54}
$$

Note that the resulting matrix $\mathbf{A}$ has asymmetric diagonal blocks but the rest of the matrix is symmetric. The contact problem is now a standard LCP which has less favorable properties and considerably more variables:

$$
\begin{aligned}
&\mathbf{A}\vec{x} + \vec{b} \geq \vec{0} \perp \vec{x} \geq \vec{0} \\
&\text{where } \vec{x}, \vec{b} \in \mathbb{R}^{(\eta+2)N}, \mathbf{A} \in \mathbb{R}^{(\eta+2)N \times (\eta+2)N}
\end{aligned}
\tag{55}
$$

Please note that if you want an exact reformulation of the box LCP into a standard LCP, then you have to be careful with the tangential vectors. The solution set of the LCP is not the same if you just use $\vec{t}_x, \vec{t}_y, -\vec{t}_x, -\vec{t}_y$ as the tangential vectors. Please study Fig. 6. The box LCP forms an axis-aligned box restricting the friction forces, however, the standard LCP formulation requires that the linear-combination of the friction components must reside within the cone (i.e. for example $f_{t_1} + f_{t_2} \leq \mu f_n$). A consequence of that is, that the polyhedral friction cone is inscribed instead of circumscribed as in the box LCP case. So first of all the tangential vectors need to be rotated by 45° to get the same alignment of the cone and secondly we need to enlarge the cone to get a circumscribed one by choosing $\sqrt{2}\mu$ instead of μ as the coefficient of friction.

(a) The friction constraints in the standard LCP formulation.

(b) The friction constraints in the box LCP formulation.

Figure 6: The friction cone differences between the LCP and the box LCP formulations.

friction model	math. model	properties
no friction	LCP	✗ rare practical use ✓ symmetric system matrix ✓ robust solvers
box friction	box LCP	✗ significant simplification (normal force decoupling) ✗ requires estimation of normal forces ✗ anisotropic friction ✓ symmetric system matrix ✓ robust solvers
pyramidal cone friction	(mod.) box LCP	✗ non-standard bounds ✗ anisotropic friction ✓ symmetric system matrix
polyhedral cone friction	LCP	✗ anisotropic friction ✗ asymmetric system matrix ✗ hard to solve ✓ adjustable anisotropy
circular cone friction	NCP	✗ very hard to solve ✓ isotropic friction

Table 2: Summary of contact problem models.

3.7.4 Summary

All of the models introduced have benefits and drawbacks. Tab. 2 tries to summarize them. The frictionless model is the easiest to solve but lacks interesting applications. The simplest friction model investigated is the box friction. The formulation as an LCP with box constraints conserves the symmetry property of the system matrix, but obviously does not include a coupling between the friction and the normal forces. Though it is a very simplified model robust solvers are available. The normal force coupling can be introduced into the box friction model by replacing the constant bounds by adaptive bounds, which leads to the pyramidal cone friction model. Special techniques will have to be employed to cope with the non-standard bounds. The box shape of the cone entails anisotropies in the friction force. Polyhedral cone friction models counter the problem by adding more facets to the friction pyramid. The tangential vectors spanning the tangential plane of the polyhedral friction cone are no more linearly independent and additional constraints need to be included, which introduce asymmetries and zeros on the diagonal of the system matrix. The most involved model deals with circular friction cones and does not exhibit anisotropies.

4 Solvers

This chapter introduces several iterative methods for the solution of the complementarity problems arising from the various contact models discussed above. At the end a review of the advantages and drawbacks is given.

4.1 Splitting Methods

For solving linear systems of equations, splitting methods are well known. The system matrix $\mathbf{A}$ is split into two parts, which add up to $\mathbf{A}$ again.

$$\mathbf{A}\vec{x} + \vec{b} = \hat{\mathbf{A}}\vec{x} + (\mathbf{A} - \hat{\mathbf{A}})\vec{x} + \vec{b} = \vec{0} \tag{56}$$

The splitting can be used to construct a simple fixed-point iteration.

$$\hat{\mathbf{A}}\vec{x}^{k+1} + (\mathbf{A} - \hat{\mathbf{A}})\vec{x}^{k} + \vec{b} = \vec{0} \tag{57}$$

In every iteration of the method, equation (57) is solved for $\vec{x}^{k+1}$ and (hopefully) the iterate contracts to the solution of the system. The most prominent methods use the diagonal part $\mathbf{D}$ or the lower-left triangular part $\mathbf{L} + \mathbf{D}$ as $\hat{\mathbf{A}}$ where $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{R}$. Those are the Jacobi method ($\hat{\mathbf{A}} = \mathbf{D}$), the Gauss-Seidel method ($\hat{\mathbf{A}} = \mathbf{L} + \mathbf{D}$) and the successive-overrelaxation (SOR) method ($\hat{\mathbf{A}} = \omega^{-1}\mathbf{D} + \mathbf{L},\ \omega \in (0, 2)$). Note that a fixed-point (where $\vec{x}^{k+1} = \vec{x}^{k}$) solves the LSE $\mathbf{A}\vec{x} + \vec{b} = \vec{0}$.

The very same idea can be applied to complementarity problems. Consider the following box LCP where the matrix is already split up. A fixed-point will again solve the original complementarity problem.

$$\hat{\mathbf{A}}\vec{x}^{k+1} + (\mathbf{A} - \hat{\mathbf{A}})\vec{x}^{k} + \vec{b} \gtreqless \vec{0} \perp \underline{\vec{x}} \leq \vec{x}^{k+1} \leq \overline{\vec{x}} \tag{58}$$

Analogously to LSEs, $\hat{\mathbf{A}}$ should be easily invertible, otherwise nothing is gained. In the case where $\hat{\mathbf{A}}$ has a triangular structure, we can use a forward (or backward) substitution to solve it componentwise:

$$\hat{\mathbf{A}}_{ii}\vec{x}_i^{k+1} + \left(\sum_{j<i} \hat{\mathbf{A}}_{ij}\vec{x}_j^{k+1} + \sum_{j>i} \mathbf{A}_{ij}\vec{x}_j^{k} + \vec{b}_i \right) \gtreqless \vec{0} \perp \underline{\vec{x}}_i \leq \vec{x}_i^{k+1} \leq \overline{\vec{x}}_i \tag{59}$$

Nevertheless 1×1 complementarity problems have to be solved. Or if we intend to use a block variant where $\mathbf{D}$ is block diagonal, then we need to solve complementarity problems whose order corresponds to the block size. Thus it is important to know how to solve small LCPs efficiently. We will explore this question in the following.

4.1.1 Projections

In essence we have to solve 1×1 complementarity problems like this one:

$$ax + b \gtreqless 0 \perp \underline{x} \leq x \leq \overline{x} \tag{60}$$

We approach the solution by checking first if the dependent variable $ax + b$ can equal zero without violating the bounds on the unknowns. Therefore we solve $ax + b = 0$ and check if x exceeds its bounds. If it doesn't we have a solution at hand, otherwise we can use projections to project the solution back into the constraint set. As an example consider the case where $x = -a^{-1}b > \overline{x}$. The unknown is projected back by setting it to $\overline{x}$. So can we be sure that it still solves the problem?

$$\begin{aligned} &x = \overline{x} + \delta \quad \delta > 0 \\ &a\overline{x} + b = a(x - \delta) + b = ax - a\delta + b = -a\delta \end{aligned} \tag{61}$$

It turns out that if we project back on the upper bound, the complementarity condition is still satisfied, as long as $a \geq 0$. In this case x is bounded above and $ax + b \leq 0$. The complementarity condition also holds for the case where we project back on the lower bound. This observation matches with the statement made in Cottle et al. [8], section 5.2 in the context of LCPs, that the diagonal elements are assumed to be positive. This gives reason to introduce the splitting methods *projected Jacobi* (PJ), *projected Gauss-Seidel* (PGS) and *projected SOR* (PSOR), where the complementarity subproblems are solved by projections and where the diagonal elements are assumed to be positive. Though we deal with positive-semi-definite matrices, it was established in section 3.6.3 that the diagonal blocks are PD and thus the diagonal elements are ensured to be positive.

Alg. 1 lists pseudo-code for a projected SOR method. Note that the initial guess $\vec{x}^0$ should stay within the bounds. This can be achieved by either choosing one of the bounds as the initial iterate or by projecting an arbitrary initial guess to the constraint set. For the termination criterion a lot of different possibilities exist. The easiest is a fixed number of iterations. Another one is to check if the iterate stagnates, i.e. $\|\vec{x}^{k+1} - \vec{x}^k\| \leq \epsilon$. The advantage of this one is that even if the solver cannot contract completely to the solution, we do not get stuck in an infinite loop (or until we reach the maximum number of iterations), but still get a reasonable approximation. Last but not least we can check a residual of the complementarity problem. For example (18) can be used.

Algorithm 1: The projected SOR method.

```
k := 0
x^k := x̲
while x^k is not satisfactory do
    for i ∈ {0, ..., n − 1} do
        x_i^{k+1} := min( x̄_i, max( x̲_i, x_i^k − ω A_ii^{−1} ( b_i + Σ_{j<i} A_ij x_j^{k+1} + Σ_{j≥i} A_ij x_j^k ) ) )
    end
    k := k + 1
end
```

4.1.2 Total Enumeration

Though we can use projections to solve (most of the) 1×1 complementarity problems, the projection approach breaks down for larger complementarity problems. Thus if we want to apply

a block-variant of the splitting scheme introduced above, we need fast direct complementarity problem solvers. As mentioned earlier, complementarity problems are NP-complete. Though only algorithms with exponential complexity are known, the effort is absolutely justifiable if the systems are very small. So a total enumeration solver, which tries every possible state of the unknown, is still feasible for solving the subproblems occurring in a PGS block-variant. For box LCPs, we have to try three possibilities for each unknown: unbounded, bounded above and bounded below. Assuming we have n unknowns in the subproblem this results in 3^n choices. In contrast standard LCPs are somewhat easier because "only" 2^n possibilities have to be checked. For each choice we have to check if it is a feasible solution, which satisfies the complementarity conditions. First of all, the dependent variable corresponding to the respective unknown must match the complementarity condition, e.g. if the unknown is bounded above, the dependent variable must be non-positive. Furthermore we need to solve the linear system of equations which contains all equations of the presumably unbounded unknowns, where the bounded unknowns are already eliminated. If the solution stays within the given bounds, then we found a solution to the complementarity problem. Note that for symmetric and PD blocks all of the linear subsystems which occur are solvable because any principal submatrix is still symmetric and PD and thus non-singular.

4.1.3 Friction Handling

The frictional contact problem introduced in section 3.6.1 is formulated as a modified box LCP, where the bounds on the unknowns depend on the unknowns. This is because the friction forces are bounded by the normal component times the friction coefficient. So how to deal with this coupling? One way is to treat normal and tangential forces decoupled. That is we restrict the contact forces to a friction box or cylinder instead of a cone. Clearly the bounds for the friction forces do not depend on the normal component anymore. But what about the size of the box? We can adapt the size of the friction box each time the normal component changes and pretend that we have box constraints when computing the tangential components. This requires virtually no changes to Alg. 1 (only $\vec{\underline{x}}$ and $\vec{\overline{x}}$ are functions instead of constants). However, the solution approach is no more a standard PGS working on a symmetric PD matrix. Instead the bound adaptations act like matrix asymmetries which in turn can lead to convergence failures.

4.1.4 Block Variants

The approach described in section 4.1.3 is a Gauss-Seidel type of method which uses projections after each relaxation and adaptations of the bounds afterwards. Though the previous approach can also be extended to a block-variant, other block-variants with different, more interesting properties can be constructed.

4.1.4.1 Friction Cylinder Projections

Instead of successively adapting the size of the friction box, we can use a friction cylinder. It seems to be a good idea to compute the friction force necessary to prevent any slip and afterwards project it back to the cylinder. The friction force is composed of two tangential components and thus we compute both components together, i.e. we treat the tangential components as a 2×2 block in the PGS scheme. As mentioned before, the diagonal blocks are symmetric and PD and so are the principal submatrices. Therefore the i-th 2×2 block

corresponding to the tangential components of the i-th contact can always be decomposed into $\mathbf{G}_i$ and $\mathbf{G}_i^T$ using a Cholesky factorization. These factorizations can be precomputed prior to the first iteration. Each time we solve for a friction force we can then use a forward and backward substitution to efficiently compute it.

$$\begin{aligned}
\hat{\vec{x}}_t^{(i),k+1} =& \vec{x}_t^{(i),k} - \mathbf{A}_{tt}^{(i)(i)^{-1}} \left(\vec{b}_t^{(i)} + \sum_{j<i} \mathbf{A}_{t\cdot}^{(i)(j)} \vec{x}^{(j),k+1} + \sum_{j\geq i} \mathbf{A}_{t\cdot}^{(i)(j)} \vec{x}^{(j),k} \right) \\
=& \vec{x}_t^{(i),k} - \Delta x_t^{(i),k+1} \\
\mathbf{A}_{tt}^{(i)(i)} \Delta x_t^{(i),k+1} =& \mathbf{G}_i \mathbf{G}_i^T \Delta x_t^{(i),k+1} = \vec{b}_t^{(i)} + \sum_{j<i} \mathbf{A}_{t\cdot}^{(i)(j)} \vec{x}^{(j),k+1} + \sum_{j\geq i} \mathbf{A}_{t\cdot}^{(i)(j)} \vec{x}^{(j),k}
\end{aligned} \tag{62}$$

If the friction force $\hat{\vec{f}}_t = \hat{\vec{x}}_t^{(i),k+1}$ is outside the cylinder, then we project it back to the cylinder. Let us denote the current radius of the cylinder with μf_n. The projection of the friction force works as follows:

$$\vec{f}_t = \begin{cases} \hat{\vec{f}}_t & \text{if } \|\hat{\vec{f}}_t\|_2 \leq \mu f_n \\ \frac{\mu f_n}{\|\hat{\vec{f}}_t\|_2} \hat{\vec{f}}_t & \text{else} \end{cases} \tag{63}$$

If we apply this scheme, then we will get a friction force inside the true friction cone but what about the dissipativity of the friction force? What happens when we project the friction force back to the cylinder? To investigate that, we split up the not yet projected friction force $\hat{\vec{f}}_t$ into the projected result and the part that is lost when projecting.

$$\begin{aligned}
\hat{\vec{f}}_t =& \vec{f}_t + \delta \vec{f}_{t,0} \\
=& \mu f_n \vec{f}_{t,0} + \delta \vec{f}_{t,0} \\
\vec{f}_{t,0} =& \frac{\vec{f}_t}{\|\vec{f}_t\|_2}, \ \delta \geq 0
\end{aligned} \tag{64}$$

For simplicity we will refer to $\mathbf{A}_{tt}^{(i)(i)}$ as $\mathbf{A}$ and to the rest of the LSE as $\vec{b}$. The dissipativity can be checked by checking the direction of the projected friction force in reference to the slip.

$$\begin{aligned}
\vec{f}_t^T \left(\mathbf{A} \vec{f}_t + \vec{b} \right) =& \left(\mu f_n \vec{f}_{t,0} \right)^T \left(\mathbf{A} \left(\hat{\vec{f}}_t - \delta \vec{f}_{t,0} \right) + \vec{b} \right) \\
=& \mu f_n \vec{f}_{t,0}^T \big(\underbrace{\mathbf{A} \hat{\vec{f}}_t + \vec{b}}_{\vec{0}} - \delta \mathbf{A} \vec{f}_{t,0} \big) \\
=& - \underbrace{\delta \mu f_n}_{\geq 0} \underbrace{\vec{f}_{t,0}^T \mathbf{A} \vec{f}_{t,0}}_{\geq 0 \text{ since } \mathbf{A} \text{ is PD}} \leq 0
\end{aligned} \tag{65}$$

Eq. (65) suggests that the friction force is partially dissipative since it opposes at least partially the slip direction. However, it is not maximally dissipative in general unless $\mathbf{A}$ is a multiple of the identity matrix. Thus this solver is particularly interesting for solving spherical granular media problems, where the *tangential* diagonal blocks are usually diagonal and both entries are of the same magnitude. The solution will then be maximally dissipative.

4.1.4.2 Friction Cone Projections

We explored the idea of projecting the friction force onto the cross section of the friction cone. So the enhanced idea of projecting the contact force to the friction cone stands to reason. Instead of having 1×1 or 2×2 blocks, we have 3×3 blocks now. Again these blocks are ensured to be invertible. So how would we project? Up to this point we used orthogonal projections. Extending the method to use orthogonal projections onto the friction cone would be straightforward. Analogously to the previous methods we *could* hope that the resulting relative velocity opposes the part of the force which was removed by the projection. This part is inside the normal cone $N_K(\vec{x})$ where K is a friction cone here. Thus the relative contact velocity would be in the negative normal cone $S_K(\vec{x})$ (cf. section 2.2). Indeed we *would* solve a variational inequality or equivalently a cone complementarity problem where the constraint set K is a friction cone. Let us assume for a moment that this actually works, then the VIP still does not represent the actual contact problem. Consider a potential contact force which resides on one of the facets of the friction pyramid and satisfies the VIP constraints. In this case the dependent variables must be in $S_K(\vec{x})$, that is orthogonally to the corresponding friction pyramid face. Consequently the dependent variables, which represent the relative contact velocity, will have a non-zero tangential and a non-zero normal component. Or in other words, as soon as friction is present, there will also be a normal velocity which will separate the contact at once, which is obviously not what we want. Therefore formulating the contact problem as a VIP with the cartesian product of all friction cones as a constraint set is not exactly what we are looking for. Nevertheless it is also a modelling option and considerable research is going in this direction [1] [4]. In Anitescu [1], section 5.1 it is explained, that "[the undesirable] effect disappears for many interesting examples as the time step goes to zero and that the solution of the relaxed time-stepping scheme satisfies the measure differential inclusion". Though it would be worthwhile to explore this model because of the simpler solvers, it is not what we are looking for right now.

Back to the original idea of projecting onto the friction cone. So orthogonal projections do not the proper thing. Instead we could come up with other projections, e.g. projecting in parallel to the tangential plane onto the friction cone. No matter which projection we choose we still encounter the problem, that the dependent variable does not perfectly oppose the part removed by the projection. The reasoning is similar as with the projection of the friction force onto the cross section of the friction cone in section 4.1.4.1. The relative velocity will oppose the part of the contact force which is removed by the projection *only partially*. Consequently there exists no projection which will enable us to solve the contact problem for a single contact, because the resulting direction of the relative velocity is perturbed depending on the diagonal block.

4.1.4.3 One Contact Problems

Though we cannot use projections to solve the contact problem for a single contact as described in the last section, we can use other methods to solve them. There are numerous options. All of them solve for a contact force treating all other contacts as fixed. It is no longer important to follow the box LCP formulation but instead we can treat the problem per se. That is we construct a subproblem in whatever model seems appropriate. In particular we could employ a circular friction cone model and use a powerful solver like the non-smooth Newton method as it will be explained in section 4.3 to solve a root-finding reformulation of the one contact subproblem. This is what M. Jean suggested in Jean [10], section 5.3 and in reference to the

paper the method should be called the *non-smooth contact dynamics* (NSCD) method. For frictionless models it reduces to the PGS and for box friction models it reduces to a 3×3 PGS block-variant. The major advantage over the PGS variants is, that we can actually solve the polyhedral and circular cone (Coulomb) friction models. Furthermore the block splitting should also lead to better convergence in most cases.

To be able to compare the solver with the other solvers an NSCD approach was devised, which solves pyramidal cone friction models as subproblems. In principle this subproblem could be formulated as a standard LCP (cf. Eq. (54)) and solved by the Lemke algorithm. But this formulation is of the order $\eta + 2 = 6$ and the Lemke might need $2^6 = 64$ steps to solve it. As an alternative we can use a total enumeration scheme as suggested in section 4.1.2 to solve the modified box linear complementarity subproblem which is only of order 3. Unfortunately the bounds are non-constant and thus the enumeration scheme gets more involved. Assume a subproblem of the following type where the other constant contact forces are included in $\vec{b}$:

$$\begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_n \\ x_{t_x} \\ x_{t_y} \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \gtreqless \vec{0} \perp \begin{pmatrix} 0 \\ -\mu x_n \\ -\mu x_n \end{pmatrix} \leq \begin{pmatrix} x_n \\ x_{t_x} \\ x_{t_y} \end{pmatrix} \leq \begin{pmatrix} \infty \\ \mu x_n \\ \mu x_n \end{pmatrix} \tag{66}$$

To begin with we need to check if the contact is separating or not. A separating contact will always possess a zero contact force. It is enough to check if b_1 is non-negative, which indicates that the contact is separating. If this is not the case, it is clear that the contact persists and that the relative normal velocity can be solved to be zero. Thus the first complementarity constraint turns into the following equation:

$$a_{11}x_n + a_{21}x_{t_x} + a_{31}x_{t_y} + b_1 = 0 \tag{67}$$

From this equation we can easily check all $2 \cdot 2 = 4$ cases where slipping occurs in both tangential directions, which means that x_{t_x} and x_{t_y} are both bounded either by μx_n or $-\mu x_n$. Substituting x_{t_x} and x_{t_y} by their respective bounds in Eq. (67) and solving for x_n enables us to check if the resulting relative contact velocities $\mathbf{A}\vec{x} + \vec{b}$ fulfill the complementarity conditions. If neither of these cases apply, the contact must be static in either of the tangential directions. The procedure is still the same. The tangential variable which is not static (i.e. bounded) is eliminated from (66) and the remaining 2×2 LSE is solved. The resulting contact forces are checked if they comply with the complementarity constraints. As an example consider the case where x_{t_x} is bounded by μx_n and the contact is static in tangential y-direction:

$$\begin{pmatrix} a_{11} + \mu a_{21} & a_{31} \\ a_{31} + \mu a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_n \\ x_{t_y} \end{pmatrix} + \begin{pmatrix} b_1 \\ b_3 \end{pmatrix} = \vec{0}, \quad x_n \geq 0, \quad x_{t_x} = \mu x_n, \quad -\mu x_n \leq x_{t_y} \leq \mu x_n \tag{68}$$

After solving the LSE, the inequalities in (68) need to be checked and additionally $(a_{21} + \mu a_{22})x_n + a_{32}x_{t_y} + b_2$ must be non-positive to fulfill the complementarity constraint. Lastly the contact can be completely static in which case we must solve the 3×3 LSE $\mathbf{A}\vec{x} + \vec{b} = \vec{0}$. This can be sped up by decomposing each of the diagonal blocks by a Cholesky factorization ahead of the first iteration. Thus in total there are only 10 cases to test for each subproblem.

4.2 Conjugate Projected Gradient Method

The Conjugate Gradient (CG) method has good reputation for solving symmetric and PSD linear systems of equations. Essentially it works with an (unconstrained) optimization reformulation, where the minimum corresponds to the solution of the LSE. Compared to the splitting methods it has quadratic instead of linear convergence and thus an equivalent for LCPs is desirable.

4.2.1 Optimization Reformulation

Any LCP can be restated as a quadratic program (QP). A QP is a type of optimization problem, where a quadratic objective function is minimized subject to a (polyhedral) constraint set on the variables. The reformulation is straightforward:

$$\begin{aligned} \text{minimize} \quad & f(\vec{x}) = \vec{x}^T \left(\mathbf{A}\vec{x} + \vec{b}\right) \\ \text{subject to} \quad & \mathbf{A}\vec{x} + \vec{b} \geq \vec{0} \\ & \vec{x} \geq \vec{0} \end{aligned} \tag{69}$$

The non-negativity constraints are directly adopted from the LCP. Since the objective function can only take non-negative values, clearly there exists a vector $\vec{x}$ for which the objective function is zero, if the LCP is solvable. For symmetric matrices, a simpler formulation is possible, which only contains the constraints to keep the unknowns in the non-negative orthant ($\vec{x} \geq 0$) as detailed in Cottle et al. [8], section 1.4. Our problem is symmetric, however, only if we use box constraints in the complementarity formulation. Therefore we need to extend the results to box-constrained LCPs. We will see, that for any symmetric matrix $\mathbf{A}$ and lower and upper bounds $\underline{\vec{x}}$, $\overline{\vec{x}}$, a solution to the following QP will also solve the underlying box-constrained LCP.

$$\begin{aligned} & \text{minimize} && \tfrac{1}{2}\vec{x}^T\mathbf{A}\vec{x} + \vec{x}^T\vec{b} \\ & \text{subject to} && \underline{\vec{x}} \leq \vec{x} \leq \overline{\vec{x}} \\ \overset{\breve{\vec{x}}=\vec{x}-\underline{\vec{x}}}{\Leftrightarrow} & \text{minimize} && \left(\breve{\vec{x}} + \underline{\vec{x}}\right)^T \left(\tfrac{1}{2}\mathbf{A}\left(\breve{\vec{x}} + \underline{\vec{x}}\right) + \vec{b}\right) \\ & \text{subject to} && \breve{\vec{x}} \geq \vec{0} \\ & && -\breve{\vec{x}} - \underline{\vec{x}} + \overline{\vec{x}} \geq \vec{0} \\ \Leftrightarrow & \text{minimize} && \tfrac{1}{2}\breve{\vec{x}}^T\mathbf{A}\breve{\vec{x}} + \tfrac{1}{2}\breve{\vec{x}}^T\mathbf{A}\underline{\vec{x}} + \tfrac{1}{2}\underline{\vec{x}}^T\mathbf{A}\breve{\vec{x}} + \breve{\vec{x}}^T\vec{b} + \overbrace{\frac{1}{2}\underline{\vec{x}}^T\mathbf{A}\underline{\vec{x}} + \underline{\vec{x}}^T\vec{b}}^{\text{const}} \\ & \text{subject to} && \breve{\vec{x}} \geq \vec{0} \\ & && -\mathbf{I}\breve{\vec{x}} \geq \underline{\vec{x}} - \overline{\vec{x}} \\ \overset{\mathbf{A}=\mathbf{A}^T}{\Leftrightarrow} & \text{minimize} && \tfrac{1}{2}\breve{\vec{x}}^T\mathbf{A}\breve{\vec{x}} + \breve{\vec{x}}^T\left(\mathbf{A}\underline{\vec{x}} + \vec{b}\right) \\ & \text{subject to} && \breve{\vec{x}} \geq \vec{0} \\ & && -\mathbf{I}\breve{\vec{x}} \geq \underline{\vec{x}} - \overline{\vec{x}} \end{aligned} \tag{70}$$

This QP can be easily transformed into an LCP using the Karush-Kuhn-Tucker (KKT) conditions as it is done in [8, section 1.2]. The KKT conditions are necessary conditions for a point to be a global optimum of the QP, however, they are not sufficient in general. In other words a local optimum of the QP is a solution of the LCP and vice versa. The QP transforms into the following LCP:

$$
\begin{aligned}
& \begin{pmatrix} \mathbf{A} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{\tilde{x}} \\ \vec{y} \end{pmatrix} + \begin{pmatrix} \mathbf{A}\vec{\underline{x}} + \vec{b} \\ \vec{\overline{x}} - \vec{\underline{x}} \end{pmatrix} \geq \vec{0} \perp \begin{pmatrix} \vec{\tilde{x}} \\ \vec{y} \end{pmatrix} \geq \vec{0} \\
\overset{\vec{x}=\vec{\tilde{x}}+\vec{\underline{x}}}{\Leftrightarrow} \quad & \begin{pmatrix} \mathbf{A} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} + \begin{pmatrix} -\mathbf{A}\vec{\underline{x}} \\ \vec{\underline{x}} \end{pmatrix} + \begin{pmatrix} \mathbf{A}\vec{\underline{x}} + \vec{b} \\ \vec{\overline{x}} - \vec{\underline{x}} \end{pmatrix} \geq \vec{0} \perp \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} \geq \begin{pmatrix} \vec{\underline{x}} \\ \vec{0} \end{pmatrix} \\
\Leftrightarrow \quad & \begin{pmatrix} \mathbf{A} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} + \begin{pmatrix} \vec{b} \\ \vec{\overline{x}} \end{pmatrix} \geq \vec{0} \perp \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} \geq \begin{pmatrix} \vec{\underline{x}} \\ \vec{0} \end{pmatrix} \\
\Leftrightarrow \quad & \mathbf{A}\vec{x} + \vec{b} \gtreqless \vec{0} \perp \vec{\underline{x}} \leq \vec{x} \leq \vec{\overline{x}}
\end{aligned}
\tag{71}
$$

Since we are dealing with a PSD matrix, the objective function is convex and thus a solution to the underlying LCP constitutes a globally optimal solution of the QP [8, section 1.2]. Note that the derivation above assumes, that the bounds are constant. If we would allow them to depend on the unknowns, then the objective function of the QP could not be simplified as above and the KKT conditions would presumably not lead to the underlying complementarity problem. Naturally the question arises, how we can use the QP to solve cone friction models, where the bounds are variable instead of constant. As with the splitting methods, we can again successively adapt the size of the friction box. Either by means of an outer fixed-point iteration of QPs or by adapting the bounds in the course of the solution method. In Renouf and Alart [17] a Conjugate Gradient (CG) type of method was proposed, where the adherence to the bounds is achieved by gradient projections and the algorithm is consistently called Conjugate Projected Gradient (CPG) method. The method allows to adapt the size of the friction box without an additional outer loop but instead adapts the bounds as the algorithm progresses. The details are explained in the following sections.

4.2.2 Incorporation of Bounds

The CPG method is in principle a CG method: The iteration starts by computing the steepest descent direction, which equals the negative gradient. Subsequently the real descent direction is computed, in a way that it is A-conjugate to the previous descent direction. Then a line search along the descent direction is performed, which is easy because the objective function is assumed to be quadratic. Then the iterate minimizes the objective function on the subspace which is spanned by all past descent directions which are consecutively conjugated. We can compute the residual to check if the iterate suits our needs.

To handle the bounds we need to add two more ingredients. First of all, if the iterate moved out of the admissible region when the line search was performed, it must be projected back or otherwise corrected (cf. section 4.2.3). This defines our current *active set* of constraints. In the next iteration we must not move in the direction of the active constraints again. To prevent that, the descent direction must be projected (cf. section 4.2.4). Pseudo-code is given in Alg. 2.

Algorithm 2: The Conjugate Projected Gradient algorithm.

```
k := 0
p⃗ := 0⃗
x⃗^k := x̲⃗
while x⃗^k is not satisfactory do
    // Compute steepest descent direction
    r⃗ := −(A x⃗^k + b⃗)
    // Projection of descent directions
    r⃗̂ := project r⃗
    p⃗̂ := project p⃗
    // Compute next descent direction (conjugation process)
    β := (p⃗^T A p⃗ = 0 ? 0 : −(r⃗̂^T A p⃗)/(p⃗̂^T A p⃗))
    p⃗ := r⃗̂ + β p⃗̂
    // Line search along descent direction
    α := (p⃗^T A p⃗ = 0 ? 0 : (r⃗^T p⃗)/(p⃗̂^T A p⃗))
    if α = 0 and β = 0 then
        break
    end
    x⃗^{k+1} := x⃗^k + α p⃗
    // Correction of iterate
    x⃗^{k+1} := correct x⃗^{k+1}
    k := k + 1
end
```

In theory we can prevent the iterate from exceeding the bounds when performing the line search. One possibility is to find the smallest parameter α, for which the new iterate would reside on one of the friction pyramid boundaries. The main problem with that is, that in each iteration of the QP solver only one additional constraint would be added to the active set. In theory we could improve upon that but in practice it is easier to just allow the line search to exceed the bounds and thus allow multiple constraints to enter the active set. It is very desirable to find the final active set quickly because afterwards successive descent directions are indeed conjugated.

The computation of the step length α and the conjugation coefficient β can break down in Alg. 2 if $\vec{p}^T \mathbf{A} \vec{p}$ vanishes. In the case of β this means we cannot conjugate $\vec{r}$ and $\vec{p}$ and we just fall back to the (projected) steepest descent direction $\vec{r}$. If the denominator becomes zero when computing the step length α, then the objective function cannot be reduced along the line search $\vec{p}$. If we do not make any progress in the line search ($\alpha = 0$) and already search along the steepest descent direction ($\beta = 0$), we stop the algorithm.

4.2.3 Iterate Correction

After each line search a correction of the iterate must be done. For box constraints this is easy. Each component of the iterate can be checked separately. We need to consider several cases. If the constraint for a particular variable is active and we do not want to move away from it, that is the component of the gradient is zero (has been projected), then the constraint will stay active and actually nothing has to be done. However, to be able to deal with modified box-constrained LCPs, we set the variable to the value of the bound. In the other case we will just check if the variable exceeds its bounds. If it did, a constraint becomes active, if it didn't it becomes or stays inactive. Alg. 3 presents some pseudo-code.

Algorithm 3: The correction of the iterate $\vec{x} \in \mathbb{R}^n$.

```
for i ∈ {0, ..., n − 1} do
    if any bound i active ∧ p⃗_i = 0 then
        if upper bound i active then
            x⃗_i := x̄⃗_i
        else
            x⃗_i := x̲⃗_i
        end
    else
        if x⃗_i ≤ x̲⃗_i then
            x⃗_i := x̲⃗_i
            mark lower bound i active
        else if x⃗_i ≥ x̄⃗_i then
            x⃗_i := x̄⃗_i
            mark upper bound i active
        else
            mark constraint i unbounded
        end
    end
end
```

I also encourage the explicit tracking of the active set as it is suggested in Renouf and Alart

[17], but I prefer to use a more generic approach which makes it also applicable for other problems. Instead of remembering the status of a contact, like a slipping or sticking contact, we can keep track of the *activity* of each bound. To be precise, I store for each variable if it is bounded above, below or if it is unbounded. This makes gradient projections easy as we will see.

4.2.4 Gradient Projections

Unless the active set is empty, we need to adapt the computation of the descent direction, because active constraints dictate in which direction we may minimize the objective function. Thus if the iterate is bounded by the friction box and the negative gradient points outside the admissible region, then it must be projected to the *tangent cone*, because we may not search for a more optimal solution in this direction. Note that this projection of the gradient must survive the conjugation process with the previous descent direction. The easiest way to achieve this is to project the components of the previous descent direction each time we project the components of the gradient. In that case the component of the resulting descent direction will be zero as it is a linear combination of the gradient and the previous descent direction and both are zero due to the projections. Alg. 4 lists pseudo-code for the projection to the tangent cone of a box-constrained set.

Algorithm 4: The projection of the negative gradient $\vec{r} \in \mathbb{R}^n$ to the tangent cone of a box-constrained set.

```
for i ∈ {0, ..., n − 1} do
    if lower bound i active then
        r_i := max(r_i, 0)
    else if upper bound i active then
        r_i := min(r_i, 0)
    end
end
```

4.2.5 Drawbacks

CG algorithms are good at minimizing purely quadratic objective functions. Unfortunately the contact problems with cone friction aren't perfectly quadratic. The non-quadratic terms are hidden in the adaptation of the bounds in the algorithm above. Each adaptation will degrade the conjugate subspace even if the active set stays fixed. So high friction coefficients will cause large adaptations, which in turn will destroy the conjugation of successive descent directions. Consequently the reduction of the objective function will be worse with high friction coefficients. A possible fix could be to keep the bounds fixed for several iterations. This fix is particularly easy to implement in the framework above and affects only the correction of the iterate. If we never adapt the bounds, we end up with the box friction model, which is solved very efficiently by the CPG method as is the frictionless model.

4.3 Non-smooth Newton Method

Unlike CG methods, Newton's method is well suited for solving general nonlinear systems and still features quadratic convergence. Of course the benefits come at a price: The method is

not guaranteed to converge globally. Nevertheless it is the most flexible approach available.

4.3.1 Root-finding Reformulation

Newton's method is usually used to solve nonlinear systems of equations and as such it can also be used to find a root of a nonlinear function. Recall that we derived a residual function for our complementarity system in section 2.3. By construction it evaluates to zero for any solution of the complementarity problem and we can use Newton's method to find one of these roots. However, if we have a look at the residual equation (18), then we notice that it is not differentiable in the classical sense due to the min and max functions. In fact the function is non-smooth but continuous. This raises a problem, because the classical Newton method relies on derivatives. Fortunately Jong-Shi Pang extended the Newton method for B-differentiable equations in Pang [13]. B-differentiability refers to Bouligand differentiability which requires that the function is Lipschitz continuous and directionally differentiable. The B-differential of our residual function, which we denote by G in the following, is defined as in Pang [13]:

$$\partial_B G\left(\vec{x}^*\right) := \left\{ \lim_{k\to\infty} \partial G(\vec{x}^k) \;\middle|\; \lim_{k\to\infty} \vec{x}^k = \vec{x}^* \right\} \tag{72}$$

Note that if G is differentiable in the classical sense (Fréchet- respectively F-differentiable) then $\partial_B G\left(\vec{x}^*\right)$ only contains the Jacobian $DG\left(\vec{x}^*\right)$. Newton's method solves a linear system of equations in each iteration. This system of equations is called the *Newton equation* and is obtained by linearizing the equation at the current iterate:

$$\begin{aligned} G\left(\vec{x}+\Delta\vec{x}\right) &\approx G\left(\vec{x}\right) + \partial G(\vec{x})\Delta\vec{x} \overset{!}{=} \vec{0} \\ \partial G(\vec{x})\Delta\vec{x} &= -G\left(\vec{x}\right) \end{aligned} \tag{73}$$

If G is an affine function then $\vec{x}+\Delta\vec{x}$ would solve the root-finding problem. In general this is not the case and the process is iterated until a satisfying solution is found. Unfortunately the Newton method has only local convergence, which means it will only converge in a region nearby the actual solution. We can enlarge this region by applying a line search in the computed Newton direction $\Delta\vec{x}$ instead of taking the whole Newton step in each iteration. To be precise the line search considers the vectors $\vec{x}+\lambda\Delta\vec{x}$, where $\lambda > 0$. Since we want to find a multi-dimensional root of the residual function $G(\vec{x})$, we can use $g(\vec{x}) = \frac{1}{2}G(\vec{x})^T G(\vec{x})$ as an objective function which we would like to minimize in the line search. In other words we want to solve a one-dimensional constrained minimization problem. However, the mere reduction of the objective function is not enough to obtain convergence. Instead often a backtracking step selection of Armijo type is adopted. The details are presented e.g. in Dennis and Schnabel [9], section 6.3. Alg. 5 presents pseudo-code for the line search. The process is characterized as backtracking because it starts out with the full Newton step and successively shortens the step length until it satisfies the Armijo conditions. Note that Alg. 5 is only guaranteed to terminate, if the Newton direction is indeed a descent direction. As derived in Dennis and Schnabel [9], section 6.5 this is the case as long as the Jacobian is non-singular. The parameters of the line search are subject to tuning, however, "α is set quite small, so that hardly more than a decrease in function value is required" [9, section 6.3.2]. They suggest a value of $\alpha = 10^{-4}$. β determines the factor by which the step length is reduced if the point does not satisfy the Armijo conditions. $\beta = \frac{1}{2}$ seems to be a good starting point.

Algorithm 5: Backtracking line-search of Armijo type.

```
α ∈ (0, 1/2)
β ∈ (0, 1)
λ := 1
while g(x⃗ + λΔx⃗) > (1 − 2αλ) g(x⃗) do
    λ := βλ
end
```

4.3.2 Jacobian Construction

So far we assumed that the system of equations is differentiable and that we have a Jacobian readily available. However, as noted before the function we deal with is non-smooth and thus we need to look at the extension of Newton's method to B-differentiable functions. Essentially Pang reduces the original (potentially nonlinear) complementarity problem to a mixed LCP in each Newton iteration, where the number of inequalities is (usually) substantially less. Nevertheless obviously we still need to solve an mLCP in each iteration. Christensen et al. suggested in Christensen et al. [7] also in the context of contact problems: "[...] numerical tests show that for the frictional contact problems treated in this paper, [the Newton mLCP] can be substituted for a related linear system of equations without effecting the convergence of the method.". We will stick to Christensen's approach in the following. The derivative of the function $G(\vec{x})$ is straightforward:

$$(\partial G(\vec{x}))_i = \begin{cases} \nabla\left(\vec{x}-\underline{\vec{x}}\right)_i & \text{if } \left(\vec{x}-\underline{\vec{x}}\right)_i \leq \left(\mathbf{A}\vec{x}+\vec{b}\right)_i \wedge \left(\vec{x}-\underline{\vec{x}}\right)_i \geq \left(\vec{x}-\overline{\vec{x}}\right)_i \\ \nabla\left(\mathbf{A}\vec{x}+\vec{b}\right)_i = (\mathbf{A})_{i\cdot} & \text{if } \left(\mathbf{A}\vec{x}+\vec{b}\right)_i \leq \left(\vec{x}-\underline{\vec{x}}\right)_i \wedge \left(\mathbf{A}\vec{x}+\vec{b}\right)_i \geq \left(\vec{x}-\overline{\vec{x}}\right)_i \\ \nabla\left(\vec{x}-\overline{\vec{x}}\right)_i & \text{else} \end{cases} \tag{74}$$

Please note that there are some ambiguous cases. These cases are exactly those, where the function is not differentiable in the classical sense. However, since we want to end up with an LSE, we will just pick any derivative from $\partial_B G\left(\vec{x}\right)$. And since we have the choice we should prefer case one and three since they will lead to a simpler (more structured) LSE, which we will exploit.

Let us denote the set of indices i of all variables which belong to the first case in (74) by α, the second by β and the third by γ. The union of α and γ shall be δ and subscripts n and t select all normal and tangential components respectively. By reordering the variables, so that components in β come first, followed by α_n and ended by δ_t, we arrive at a permuted matrix whose structure is sketched in Fig. 7. The permutation matrix is denoted by $\mathbf{P}$.

The upper blocks contain permuted rows of the matrix $\mathbf{A}$. For normal components in α_n, the row of the Jacobian reduces to $\nabla\left(\vec{x}-\underline{\vec{x}}\right)_i = \nabla\vec{x}_i = \vec{e}_i$. Note that normal components do not have an upper bound and thus γ_n is empty. Tangential components are different. The gradient of $\left(\vec{x}-\underline{\vec{x}}\right)_i$ is $\vec{e}_i + \mu\vec{e}_j$, where j is the index of the normal component associated to the same contact as the tangential component. The normal component can be either in α_n or β_n and the coefficient of friction is placed in the lower-left or the lower-right shaded block in Fig. 7. The same holds for γ_t, except that we insert $-\mu$. Therefore each row of δ_t contains exactly two entries. One for the coefficient of friction and the other is a one located on the diagonal of the lower-right block. Furthermore the number of non-zero entries per column in

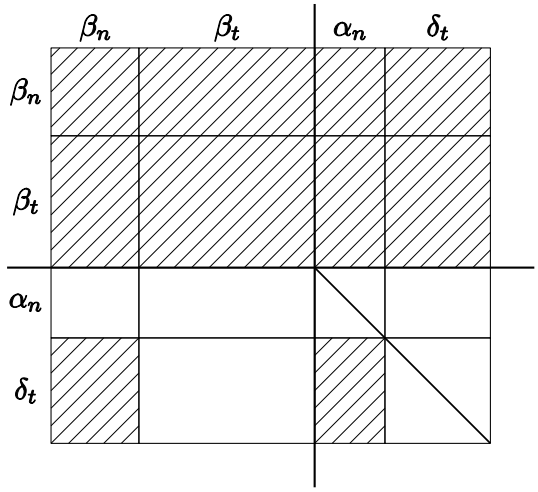


Figure 7: Sparsity structure of $\mathbf{P}^T \partial G(\vec{x})\mathbf{P}$, where shaded regions are sparse, unshaded regions are empty and the diagonal in the lower-right part consists of only ones.

the lower shaded regions is at most two because at most two tangential components depend on the same normal component.

The linear system of equations can be simplified by constructing the Schur complement. The Schur complement can be viewed as an elimination of a matrix block instead of a single matrix entry as in a Gaussian elimination procedure. Let us decompose the permuted Jacobian in four blocks: $\begin{pmatrix} \mathbf{J}_{\beta\beta} & \mathbf{J}_{\beta\delta} \\ \mathbf{J}_{\delta\beta} & \mathbf{J}_{\delta\delta} \end{pmatrix}$. The upper-right block can be eliminated by subtracting $\mathbf{J}_{\beta\delta}\mathbf{J}_{\delta\delta}^{-1}$ times the lower blocks from the upper ones. In the upper-left part the Schur complement is produced. Solving the modified system is computationally better, because the lower-right block can be easily inverted due to the triangular structure and the Schur complement system is typically much smaller. After the Schur complement system is solved, we can easily solve for the components in δ. Remember that the right-hand-side is transformed by the elimination procedure as well and has to be permuted in the same way as the matrix.

However, we can further simplify the system. The inverse of $\mathbf{J}_{\delta\delta}$ is nearly the same as $\mathbf{J}_{\delta\delta}$ itself, merely the coefficients in the shaded region switch the signs.

$$\begin{aligned} \mathbf{J}_{\delta\delta}\mathbf{J}_{\delta\delta}^{-1} &= (\mathbf{E} + \mathbf{J}_{\delta_t\alpha_n})(\mathbf{E} - \mathbf{J}_{\delta_t\alpha_n}) \\ &= \mathbf{E} + \mathbf{J}_{\delta_t\alpha_n} - \mathbf{J}_{\delta_t\alpha_n} - \underbrace{\mathbf{J}_{\delta_t\alpha_n}\mathbf{J}_{\delta_t\alpha_n}}_{\mathbf{0}} \\ &= \mathbf{E} \end{aligned} \tag{75}$$

Consequently the Schur complement system simplifies considerably:

$$\begin{aligned} \left(\mathbf{J}_{\beta\beta} - \mathbf{J}_{\beta\delta}\mathbf{J}_{\delta\delta}^{-1}\mathbf{J}_{\delta\beta}\right)\Delta\vec{x} &= \vec{b}_\beta - \mathbf{J}_{\beta\delta}\mathbf{J}_{\delta\delta}^{-1}\vec{b}_\delta \\ (\mathbf{J}_{\beta\beta} - \mathbf{J}_{\beta\delta}\mathbf{J}_{\delta\beta})\,\Delta\vec{x} &= \vec{b}_\beta - \mathbf{J}_{\beta\delta}\left(\vec{b}_\delta - \mathbf{J}_{\delta_t\alpha_n}\vec{b}_\delta\right) \\ \text{where } \begin{pmatrix} \vec{b}_\beta \\ \vec{b}_\delta \end{pmatrix} &= -\mathbf{P}^T G(\vec{x}) \end{aligned} \tag{76}$$

4.3.3 Solving the Newton Equation

Eq. (76) has to be solved in each iteration of the Newton method. Therefore the solution process should be as efficient as possible and we should thoroughly analyze the system. $\mathbf{J}_{\beta\beta}$ is a principal submatrix of the system matrix $\mathbf{A}$ (ignoring the permutation for a moment). $\mathbf{J}_{\beta\delta}\mathbf{J}_{\delta\beta}$ is subtracted from the submatrix and thus introduces asymmetric changes which destroy the consistency of the system (cf. section 3.6.2). In other words the system of equations can cease to be solvable. This is not just a theoretical issue but occurs in practice. One way to deal with it, is to try to find the least-squares solution. But still, we do not have any guarantees anymore, that the resulting Newton direction Δx is a descent direction in the case of a least-squares solution. Apart from that we can use for example the *generalized minimal residual method* (GmRes) to solve the Schur complement system. R. Ortiz proposed this approach in his Ph.D. thesis [12] and improved the convergence using an algebraic multigrid preconditioner. However, GmRes is a rather resource intense method and a more economic solver would be preferable. So let us return to the Schur complement system (76). This system is much simpler if we again ignore the dependence of the friction force on the normal force. The shaded regions in the lower part of the matrix disappear in Fig. 7, likewise the terms involving $\mathbf{J}_{\delta\beta}$ and $\mathbf{J}_{\delta_t\alpha_n}$ can be removed from the Schur complement system:

$$\begin{aligned} &\mathbf{J}_{\beta\beta}\Delta\vec{x} = \vec{b}_\beta - \mathbf{J}_{\beta\delta}\vec{b}_\delta \\ &\text{where } \begin{pmatrix} \vec{b}_\beta \\ \vec{b}_\delta \end{pmatrix} = -\mathbf{P}^T G(\vec{x}) \end{aligned} \tag{77}$$

I want to stress that Eq. (77) is always solvable since the consistency is preserved. The reason is that it is of the same structure as Eq. (47) and we already established the consistency of that type of system. So the first advantage is, that we cannot encounter any singular Newton equations and second, the matrix is symmetric now, which allows us to employ a CG method. In theory we introduced a drawback, because in fact we again use friction box size adaptations which can cause stagnations. However, in practice the adaptation rather prevents the solver from being caught in local minima.

4.3.4 Drawbacks

The Newton method is the most flexible solver so far and can even be extended to solve contact problems with circular friction cones [7]. However, the line search is not enough to make it globally convergent. Stagnations were permanently observed while experimenting with the unsimplified approach, where the iterate got caught in a local minimum of the objective function. Christensen et al. noticed the same and proposed a minimum step length when performing the line search: "Otherwise, it is very likely that the algorithm will stall." [7, section 6.1]. Unfortunately it is tedious to tune the parameter, because if it is chosen too large, then the objective function starts to cycle, since it is reduced until the minimum step size causes it to increase again and if we are unlucky we again converge to the local minimum. In contrast if the minimum step size is chosen too low, then we will not leave the local minimum. Frictionless and pure box friction problems *do not* suffer from these global convergence issues, and the simplified pyramidal friction approach, where we adapt the size of the friction box, does not suffer that severe therefrom.

Though the Newton methods are potentially only locally convergent, we can use them to reduce the residual of already fairly good solutions to machine precision. In particular, if the

approximate initial solution already has the correct active set, then the exact solution is at hand after a single iteration - provided the solver for the Newton equation is exact.

4.4 Projected Algebraic Multigrid

So far the Newton method and the NSCD method are the most flexible solvers. Unfortunately the Newton method is only locally convergent. The NSCD and PGS methods in contrast are normally globally convergent but only linear instead of quadratic. The convergence of GS type of methods can often be improved by employing multigrid approaches. A proper multigrid algorithm consequently provides a fast and globally convergent solver.

4.4.1 Multigrid Benefits

In section 4.1 several splitting-methods were introduced including the NSCD method, which can compute the solution to contact problems with circular friction cones. All those Gauss-Seidel type of methods suffer from bad convergence. A popular approach to improve the convergence of Gauss-Seidel methods are multigrid ideas. In addition to the original system also coarser versions of the system are constructed. Gauss-Seidel methods efficiently reduce local errors and therefore high frequency errors in the solution. However, error components which are of a more global nature, that is low frequency errors are not efficiently reduced and thus the Gauss-Seidel iterations only slowly improve the solution after local errors have been eliminated. The idea behind multigrid is, that the coarser system representation or *coarse grid* is used to remove lower frequency error components. To solve the coarse system again the Gauss-Seidel method or in a general context a *smoother* is applied to reduce local errors on the coarse grid, that is lower frequency errors than on the fine grid. The results are transfered from the coarser to the finer grid (interpolation) and vice versa (restriction). Typically a hierarchy of grids is established and the smallest grid is usually solved using a direct solver.

Unfortunately, the contact problem does not have an underlying structured grid and it is not obvious how a coarser system looks like. Instead the unknowns (contact forces) are distributed irregularly in space. Furthermore, the unknowns are subject to bounds and complementarity constraints. Even worse, the bounds depend on the unknowns. Consequently it is not possible to define a coarser representation of the contact problem in a geometric sense. Instead we have to apply algebraic multigrid ideas. For the moment we will consider frictionless and box frictional models only, where the bounds are fixed and do not depend on the unknowns. In the cone friction case, the problem becomes even more non-standard, since we need to make sure that, as soon as a tangential component is added to the coarse grid set of points, the normal component is also added to the coarse-grid set because it is needed for the correct bounds. Thus, by sticking to the frictionless and box friction models we can ignore the coupling and coarsen the normal and tangential components independently of each other.

4.4.2 Coarsening Strategy

The coarsening strategy employed here is a standard AMG coarsening strategy as for example presented in Briggs et al. [6]. The decision which unknowns are chosen as coarse-grid quantities is based on strong influences and dependencies. An unknown i *strongly depends* on another unknown j if the magnitude of the corresponding matrix entry a_{ij} is about the same as the largest off-diagonal matrix entry according to the magnitude:

$$|a_{ij}| \geq \theta \max_{k \neq i} \{|a_{ik}|\} \qquad \theta \in (0,1]\,,\ j \neq i \tag{78}$$

The threshold value θ controls the number of strong dependencies here. Furthermore, if unknown i strongly depends on unknown j, then on the other hand unknown j *strongly influences* unknown i. We can use the property of strong dependency for the interpolation of fine-grid quantities from coarse-grid quantities, since a strongly influencing quantity is a good candidate for accurate interpolation. This observation drives the partitioning of the unknowns into coarse-grid quantities and quantities that exist only on the fine grid. However, we must be picky about which strongly influencing quantities move into the coarse-grid set C, otherwise C will grow too large and the coarse-grid problem will be of similar size as the original problem which renders the multigrid approach useless. Thus in order to control the number of coarse-grid quantities, we aim for an assortment where no quantity in C strongly depends on another coarse-grid quantity. In a first pass we preliminarily partition the quantities into coarse- and fine-grid quantities, where we try to respect the restriction on the quantities in C and still have good approximation properties. The usual way of doing this is to assign a quality λ_i to each quantity i. This number measures the suitability as a coarse-grid quantity. It is defined as the number of quantities which are strongly influenced by the unknown i. The quantity with the maximum quality is chosen as a coarse-grid quantity first. After moving it into C, all other undecided quantities which strongly depend on it are moved into F, because they would clash the constraint, that no coarse-grid quantities should strongly depend on each other. For each quantity which was assigned to F, all undecided quantities, which strongly influence them, become even more valuable as an interpolatory coarse-grid quantity. Thus we increment their qualities accordingly. The coarse-grid quantity selection process can start from the beginning and choose the undecided quantity with the highest quality as the next coarse-grid quantity until no more undecided quantities are available.

4.4.3 Interpolation Operator

So far we have a basic partitioning of the unknowns. Next we need to clarify how we intend to interpolate the fine-grid values from the coarse unknowns. All fine-grid quantities which are also coarse-grid quantities are merely injected. All others have to be interpolated. Again we stick to standard interpolations. Those depend on the fact, that at least on average an *algebraically smooth error* $\vec{e}$ has small residuals.

$$\vec{r} = \mathbf{A}\vec{e} \approx \vec{0} \tag{79}$$

By algebraically smooth we mean that the error components, which can be removed efficiently by the smoother, have been removed from the iterate. Gauss-Seidel smoothers are good at removing local errors, that is they will lower the local residuals efficiently. For a more mathematical derivation refer to Briggs et al. [6], chapter 8 for example.

The quantities influencing a particular quantity i can be divided into strongly influencing coarse-grid quantities C_i^s, strongly influencing fine-grid quantities F_i^s and weakly influencing (coarse- or fine-grid) quantities $D_i^w = C_i^w \cup F_i^w$. Consequently we can write (79) componentwise as sums over these sets:

$$a_{ii}e_i \approx -\sum_{j \in C_i^s} a_{ij}e_j - \sum_{j \in F_i^s} a_{ij}e_j - \sum_{j \in D_i^w} a_{ij}e_j \tag{80}$$

To be able to interpolate quantity i from all strongly influencing coarse-grid quantities C_i^s, we need to express all e_j where $j \in F_i^s \cup D_i^w$ as coarse-grid errors. The quantities in D_i^w are less critical and we just distribute them to the diagonal, that is we assume $e_j \approx e_i$ for all $j \in D_i^w$. This approach can be justified since we either make a small error due to the weak dependency or if we underestimated the dependency then we can expect the error to be of similar magnitude as e_i, because "smooth error varies slowly in the direction of strong dependence"[6, chapter 8]. We need to be more careful with all quantities in F_i^s due to the strong dependency. One way to deal with those, is to express them in terms of a weighted average of the errors in C_i^s. As weights we use the matrix entries in the j-th row corresponding to the indices in C_i^s. Note that we intentionally do not use C_j^s or even C since that would introduce more non-zeros as necessary in the i-th row of the interpolation operator. Thus the interpolation formula for $j \in F_i^s$ is the following:

$$e_j \approx \frac{\sum_{k \in C_i^s} a_{jk} e_k}{\sum_{k \in C_i^s} a_{jk}} \tag{81}$$

Thus we can insert (81) into (80) and distribute the weakly influencing quantities to the diagonal. Only the quantities in C_i^s remain and we can determine their coefficients ω_{ij}. Please refer to Briggs et al. [6] for details. We now established the i-th row of the interpolation operator $\mathbf{I}_H^h$.

$$\left(\mathbf{I}_H^h \vec{e}\right)_i = \begin{cases} e_i & \text{if } i \in C \\ \sum_{j \in C_i^s} \omega_{ij} e_j & \text{if } i \in F \end{cases} \tag{82}$$

The way we interpolate strongly influencing fine-grid quantities (cf. Eq. (81)), requires that the denominator is not zero. For this to be true, the strongly influencing quantity must have at least one non-zero entry in the j-th row among C_i^s. That is the i-th and j-th fine-grid quantity absolutely *must* share a common coarse-grid quantity. Unfortunately our partitioning into coarse- and fine-grid quantities does not fulfill this requirement. A second pass of our partitioning algorithm is necessary before constructing the interpolation operator to enforce this property.

The second pass iterates over all quantities i and basically checks if all strongly influencing fine-grid quantities j share a common coarse-grid quantity. If this is not the case for a quantity j, then this one is declared as a *tentative* coarse-grid quantity until the rest of the quantities which strongly influence i are checked. However, if we find another fine-grid quantity strongly influencing i, with no common coarse-grid quantity, then it is advantageous to declare quantity i as a coarse-grid quantity and revert the present tentative coarse-grid quantity to a fine-grid quantity. This pass aims to enforce the necessary properties for the interpolation while keeping the number of new coarse-grid quantities low.

It is mandatory to note, that the above construction of the interpolation operator is purely based on algebraic multigrid ideas for linear systems of equations. To achieve better results a more sophisticated approach must carry over the underlying principles to complementarity systems. In particular Eq. (79) actually should be the residual for complementarity problems.

4.4.4 Coarse Problems

As soon as the interpolation operator $\mathbf{I}_H^h$ is constructed, the setup of the coarse-grid matrix is fairly easy. It is the product of the restriction operator, the fine-grid matrix and the interpolation operator, where the restriction operator is the transpose of the interpolation operator due to the variational property.

$$\mathbf{A}^H = {\mathbf{I}_H^h}^T \mathbf{A}^h \mathbf{I}_H^h \tag{83}$$

Usually the *residual equation* is solved on the coarser grid. However, the complementarity constraints act on the contact forces instead of the errors. Therefore we will follow a *Full Approximation Scheme* (FAS), which is normally used to solve nonlinear problems. Instead of solving for the error on the coarse grid, we solve for the coarse unknowns. Consequently we can use the same complementarity constraints as on the fine grid. And the coarse-grid complementarity problem is the following:

$$\begin{aligned} &\mathbf{A}^H \vec{x}^H - \mathbf{A}^H \left(\vec{x}^h\right)_C - \vec{r}^H \gtreqless \vec{0} \perp \underline{\vec{x}}^H \leq \vec{x}^H \leq \overline{\vec{x}}^H \\ &\text{where } \vec{r}^H = \mathbf{I}_h^H \vec{r}^h,\ \vec{r}^h = \mathbf{A}^h \vec{x}^h + \vec{b}^h,\ \underline{\vec{x}}^H = \left(\underline{\vec{x}}^h\right)_C,\ \overline{\vec{x}}^H = \left(\overline{\vec{x}}^h\right)_C \end{aligned} \tag{84}$$

This complementarity problem can again be coarsened in a recursive fashion until the problem size is small enough for a direct solver to be efficient.

4.4.5 Multigrid Cycles

The overall solution process advances in recursive correction cycles. It starts on the finest grid with an arbitrary initial guess. First the smoother is applied ν_1 times. Then the residual is computed and restricted. The iterate is restricted too but not by using the restriction operator but by selecting the coarse-grid quantities. The restricted iterate together with the residual form the right-hand-side of the coarser problem. The restricted iterate is also used as an initial guess. We will now recursively solve the problem. On return from the recursion, a coarse solution is at hand and we subtract the restricted iterate from it which gives the coarse error. The error is then interpolated and added to the current iterate on the fine grid. If a component now exceeds the bounds of the complementarity condition, we can project it back or alternatively we just ignore this particular error correction. The multigrid cycle is ended by another ν_2 smoother iterations. Optionally the recursion can be applied μ times.

4.4.6 Open Problems

The problems are underdetermined and the smoothers converge to any of the possible solutions. Therefore the inter-grid error transfers do not work together since the solutions on the different grid levels are not equivalent. This can be fixed by forcing the problem to possess a unique solution. The PSD property (cf. section 3.6.2) of the system matrix accounts for the non-uniqueness. By adding any positive diagonal matrix ϵ, we can force the matrix to become PD.

$$\vec{x}^T (\mathbf{A} + \epsilon) \vec{x} = \vec{x}^T \mathbf{A} \vec{x} + \vec{x}^T \epsilon \vec{x} \overset{\mathbf{A} \text{ is PSD}}{\geq} \vec{x}^T \epsilon \vec{x} = \sum_{i=0}^{n} \epsilon_i \vec{x}_i^2 \overset{\forall i: \epsilon_i > 0}{>} 0 \quad \forall \vec{x} \neq \vec{0} \tag{85}$$

To avoid too strong or too weak perturbations, we can scale the system matrix diagonal instead of adding a constant matrix, which ensures that ϵ is appropriate for the actual problem scaling. The diagonal scaling makes the multigrid approach actually applicable but the convergence improvements do not justify the costly algebraic multigrid setup phase.

Furthermore the coarsening described above works best for M-matrices but can break down for large positive off-diagonal elements. As explained in section 3.6.3 the contact problem matrices can have arbitrary large positive as well as negative off-diagonal elements. In Trottenberg et al. [22], appendix A the *standard interpolation* is presented, which is prepared for positive as well as negative off-diagonal elements. The standard interpolation was implemented and as expected this interpolation gives slightly better results.

All hitherto problems aside, the approach still has procedural drawbacks. The interpolation operator construction is suited for linear systems of equations but not for complementarity problems. Naturally the interpolation will work best for LCPs which have many unbounded variables and thus are very similar to LSEs (which have *exclusively* unbounded variables). A possible resort for problems with many dynamic contacts could be to warm-start the multigrid solver by executing several PGS iterations beforehand. As we will see in section 5.4 a large fraction of the correct active set will be identified by then. The coarsening process can then take advantage of this information by exchanging the matrix rows of all bounded variables by their respective unit vectors for the coarsening process - e.g. if variable i is bounded above, then $\vec{x}_i = \vec{\overline{x}}_i$ and $\mathbf{A}_{i\cdot} = \vec{e}_i$.

To extend the multigrid approach to cone frictional problems, the coarsening needs to be adapted, since tangential complementarity constraints always need the respective normal component to determine the bound for the friction force. Thus as soon as a tangential component is determined to be a coarse-grid quantity, the corresponding normal component must become a coarse-grid quantity too. To keep it simple the components of each contact should be treated as a group. Once the type (coarse- or fine-grid quantity) of one of them is determined, the others immediately adopt the same type. Fig. 8 shows a detail enlargement of an adjacency graph of the system matrix constructed from a granular media test case, where C-points are identified by boxes. The coarsening was produced by the standard AMG coarsening described in section 4.4.2. The unmodified C/F-splitting most often does coarsen all variables of a contact together anyway, thus the modification above will most likely perform comparably.

4.5 Alternatives

There are some other solvers which can be used to solve the complementarity problem. To begin with I did not dwell on direct solvers for LCPs. We can use the Lemke algorithm to solve the LCP (54). This possibility was explored in Preclik [14] and as already noted in section 3.7.3, it can suffer fatally from rounding errors and storage requirements. There is a whole bunch of other pivoting methods which are applicable to solve the linear complementarity problem and various of them are presented in Cottle et al. [8], chapter 4 in the context of LCPs, e.g. Zoutendijk's procedure and Dantzig's algorithm. At least Dantzig's algorithm is extendible to box-constrained LCPs, but none of them except for the Lemke algorithm can process the LCP (54) but only the complementarity problem arising from the box friction model. Furthermore none of them can be modified in an obvious way, so that we can at least adapt

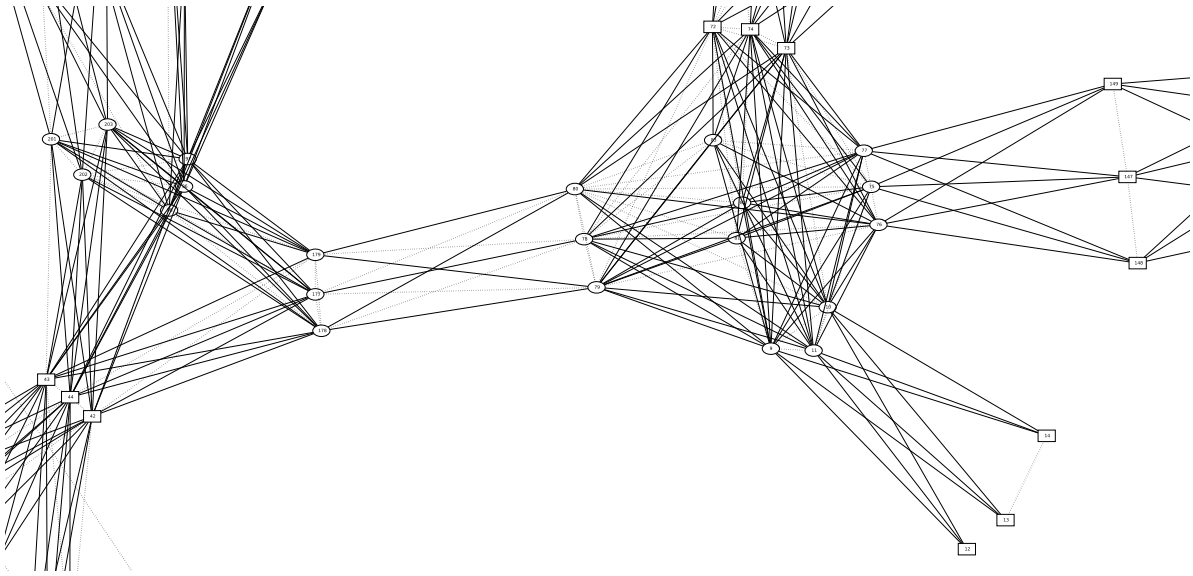


Figure 8: Adjacency graph showing the C/F-splitting produced by a standard coarsening. Boxed nodes denote coarse-grid quantities. Solid edges have a strong dependency in one direction, bold edges in both directions, dotted edges have weak dependencies in both directions. Node numbers denote component indices.

the size of the friction box as we did with the iterative solvers. Nevertheless, direct methods will easily outperform any iterative method for small problem sizes.

Last but not least interior point (IP) methods have "a well documented effectiveness for solving linear programs and monotone complementarity problems" [7]. Christensen et al. investigated an IP approach for contact problems in [7], but came to the conclusion, that for three dimensional problems "this approach is not promising.". Furthermore Trinkle et al. compared the Lemke algorithm to an IP method applied on rigid multibody contact problems in Trinkle et al. [21]. The contact model was a force-acceleration based model instead of an impulse-velocity model as it is presented here. Force-acceleration based problems do not always possess a solution, so it is not surprising, that "overall, the two algorithms solved just under 70% [...] of the data sets" [21]. However, the IP method sometimes was not able to find solutions though the Lemke algorithm did. In summary interior point approaches seem to be at least troublesome if at all reliably applicable to rigid multibody contact problems and therefore I refrained from including them here.

4.6 Summary

Many different solvers have been introduced in the previous sections. All of them have advantages and disadvantages and none of them is perfectly suitable in general. In the following I give a summary of the solver capabilities in Tab. 3. All solvers can reliably solve frictionless and box friction models. For applications which need visually pleasant simulations, box friction is usually satisfactory, however, the question of how to estimate the friction bounds needs to be addressed. The pyramidal cone friction model in contrast does not need any estimation of the normal forces. The PGS and CPG methods can be easily modified to continuously adapt the size of the friction boxes. The Lemke, NSCD and Newton methods can solve the problem without bound adaptations. The circular cone friction model (i.e. Coulomb friction), can only be solved by the NSCD and the Newton method. A block PGS variant can be used

	Dantzig	Lemke	PGS	NSCD	CPG	Newton
no friction	✓	✓	✓	✓	✓	✓
box friction	✓	✓	✓	✓	✓	✓
pyramidal cone friction		✓	✓[1]	✓	✓[1]	✓
polyhedral cone friction		✓		✓	✓[1]	✓
circular cone friction			∼[1]	✓		✓

[1] bound adaptation necessary

Table 3: Summary of solver capabilities. Shaded marks indicate that non-convergence can occur.

to project the friction force on the circular cross section of the friction cone. Unfortunately the method does not lead to a maximally dissipative solution in general unless we simulate only spheres.

All of the cone friction models are hard to solve and neither of the iterative solvers is guaranteed to converge. Solely the Lemke algorithm can solve the polyhedral models reliably at least in theory. The splitting methods (PGS, NSCD) can fail because the iterations might not contract to a fixed-point. The PGS and CPG method can fail due to the bound adaptations. The Newton method is only locally convergent in general and the iterate can get caught in local minima. Last but not least a multigrid approach was presented. It tries to remedy the slow convergence issues without sacrificing the solvable range of models of the PGS and NSCD methods. Unfortunately several difficulties still need to be overcome with the multigrid method.

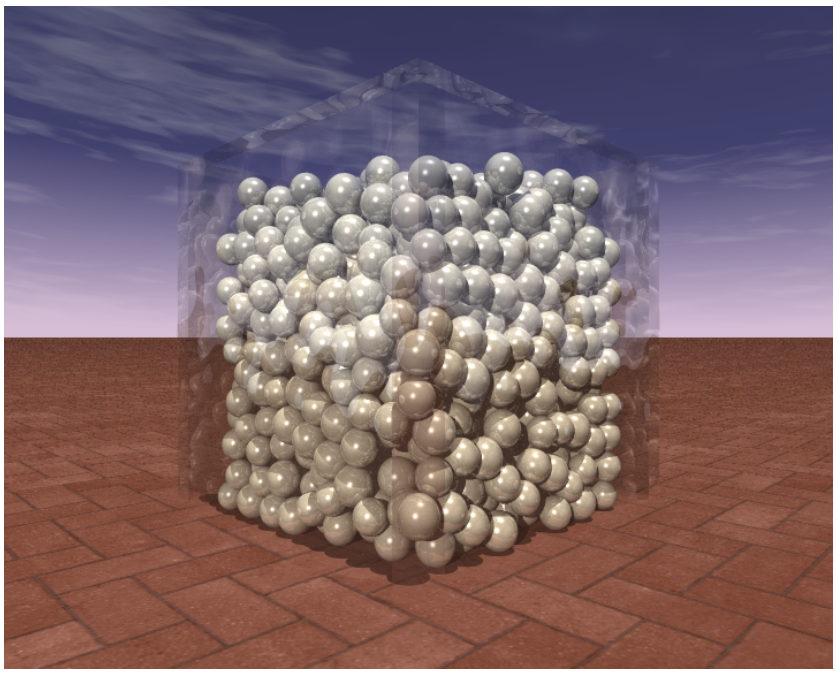


Figure 9: Spherical granular media inside a container.

5 Numerical Tests

5.1 Test Problems

It is very hard to construct representative test cases. Often test cases possess specific peculiarities, which might give certain solvers advantages over other solvers. E.g. a common test in the graphics community is to stack several boxes on top of each other. In this case each of the contacts is a static contact. To get a reasonable mixture of dynamic and static contacts I constructed several granular media test cases with only spheres. The spheres are settling in a box-shaped container but are not yet at rest. The problem sizes range up to approximately 5000 unknowns. Fig. 9 is a rendering of such a test case.

5.2 Residuals

Fig. 10 is a plot of the residuals of the PGS, the CPG and the non-smooth Newton method. The scenario is a granular media setup where the friction coefficient is 0.4. 858 spheres are in contact at 1378 contacts. The applied model is a box friction model. The PGS has better convergence in the beginning until the local errors are smoothed out and then settles on a slow convergence, which is the usual Gauss-Seidel behaviour. It has the smoothest residual graph of all solvers. The residual in the CPG method drops slightly slower in the beginning but settles not quite on quadratic convergence. That is, it takes the CPG method half of the iterations to reduce the residual by the same amount as the PGS. Furthermore the residual does exhibit some jitter commonly observed with CG solvers. The Newton method also reduces the residual fairly good in the beginning, followed by a phase of slow convergence. However, convergence accelerates in the final phase followed by a steep drop in the end. The final drop happens as soon as the correct active set is found. In this case the solution of the linear subsystem

immediately solves the LCP. The Newton method's residual always drops monotonically as enforced by the line search.

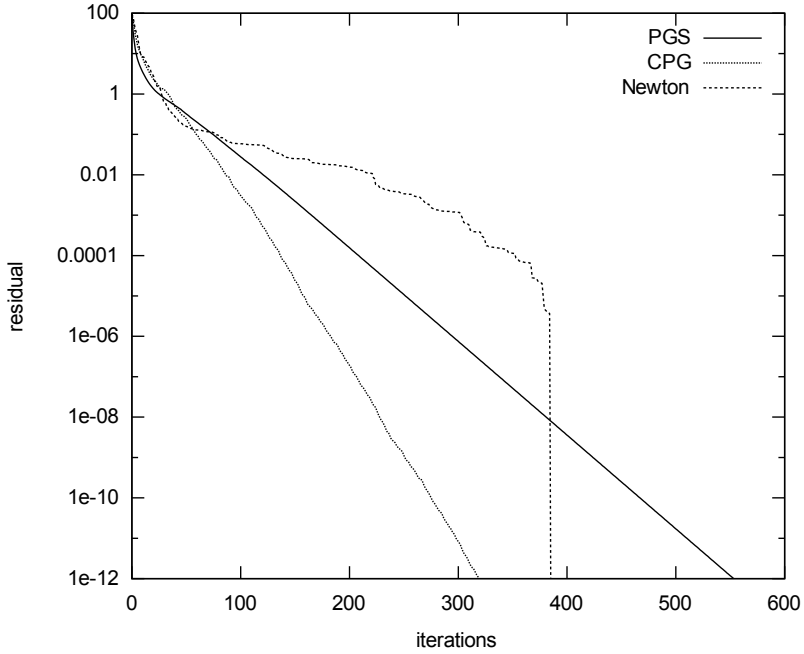
The results critically depend on the specific problem settings. Especially high friction coefficients cause slower convergence. However, the overall impression stays the same. CPG often needs significantly less iterations than the PGS but not always achieves quadratic convergence. Though the Newton method usually also needs less iterations than the PGS, the costs per iteration are enormous.

The residual plots so far are rather regular. But when switching to a (pyramidal) cone friction model, the residual graphs become more involved. Fig. 11 shows the residual plots for a smaller granular media setup where a cone friction model was used and two different coefficients of friction. For the first run a low friction coefficient of 0.1 was used for all contacts. The convergence rate of the CPG method is linear now. For the second test a high friction coefficient of 0.8 was chosen. The CPG's convergence rate is considerably slower than the PGS's rate. Furthermore the jittering increased. Both features are due to the bound adaptions which are only necessary for cone friction. Large bound adaptations cause the previous descent directions not to be perfectly conjugated. Consequently the minimization over the subspace is only suboptimal. Note that the cause for the bad convergence is not due to active set changes, because the CPG for box friction nearly has quadratic convergence but about equally as many active set changes. To support this claim we will investigate the stabilization of the active set in section 5.4.

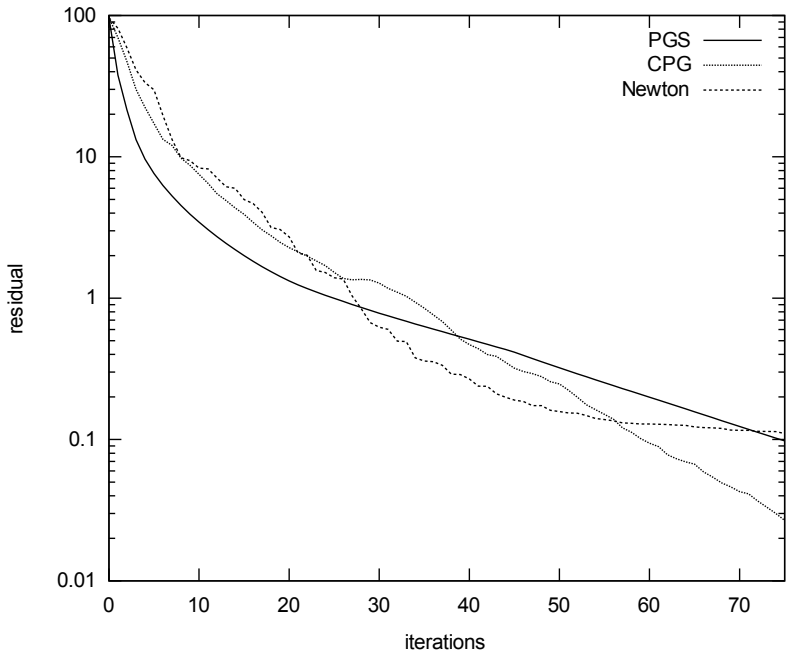
Fig. 12 shows residual plots for the Newton methods. For cone friction there are two options. Either the Jacobian is constructed exactly and an asymmetric (possibly singular) linear subsystem has to be solved in each iteration by a sophisticated solver (i.e. GmRes) or the Jacobian is constructed approximately pretending we deal with box friction. Then the system is symmetric and consistent and can be solved by a CG solver. Both residual graphs are plotted in the figure. Both approaches need about the same number of iterations. The accurate Newton has a steep drop in the end. The approximate method is not able to obtain an exact solution in the end. The reason is that the approximate Newton relies on a fixed-point iteration to find the correct friction box sizes and the accurate Newton does not. There are some more peculiarities. As noted earlier, the line search ensures that the residual decreases monotonically. But the plots exhibit several increases and especially at the end there is a small peak. Each of them occurred because the line search was capped by the minimum step length. Consequently the residual increased but the solver did not get caught in a local minimum in return. For this test a minimum step length of 10^{-3} was chosen. Another difficult to tune property is the precision up to which the linear subsystem is solved. If it is too high, the subsystem solver needs a lot of iterations to find a good solution in each iteration of the Newton method. Since especially in the beginning it is not necessary to obtain an accurate search direction, a high precision is wasteful. However, if we intend to get an accurate overall solution we also need an accurate search direction. An obvious remedy for this dilemma is to increase the required precision of the search direction in the course of the solution process depending on the current overall residual.

5.3 Velocity Errors

Though residuals give us a good hint about the quality of the solution, they are not suited very well for comparing different solvers. Instead we are interested in the evolution of the error, that is the distance of the iterate from the exact solution. Unfortunately, the complementarity problems discussed do not possess a unique solution as elaborated in section 3.6.2. Though

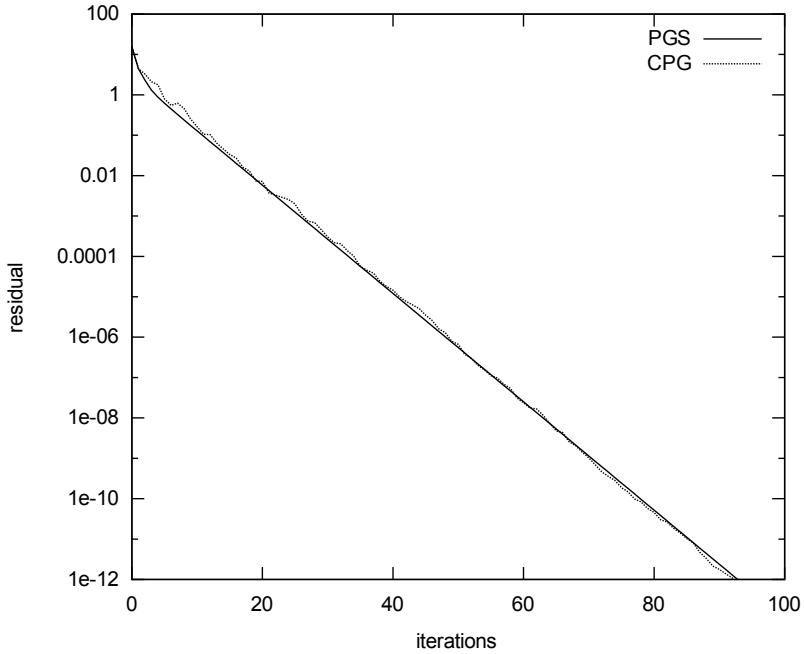


(a) The residual plots of the PGS, CPG and Newton method.

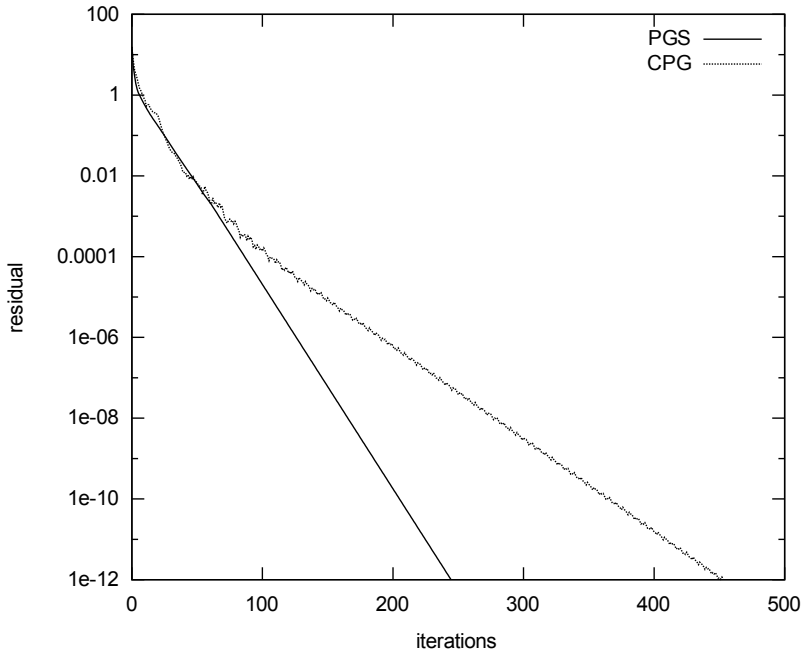


(b) Close-up of the first 75 iterations.

Figure 10: Residual plots of the solvers for a granular media test case with a friction coefficient of 0.4 applying a box friction law.



(a) The residual plots for 0.1 cone friction.



(b) The residual plots for 0.8 cone friction.

Figure 11: The residual plots of the PGS and CPG method for a granular media test case with different friction coefficients.

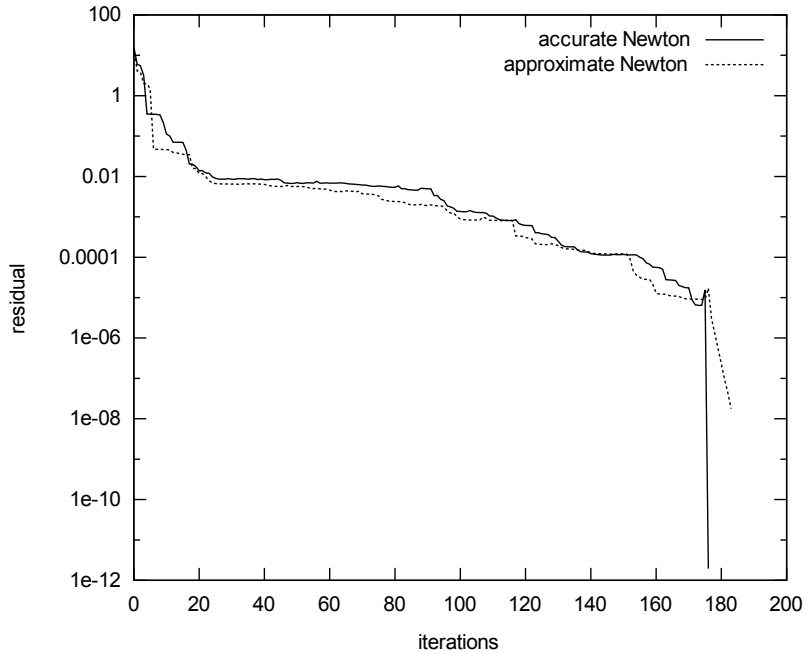


Figure 12: The residual plots of the accurate Newton method and an approximate one for a granular media test case with 0.4 cone friction.

the contact forces are ambiguous, the resulting contact velocities are not, which means that $\mathbf{A}\vec{x} + \vec{b}$ is unique instead of $\vec{x}$. So to compare the solvers we will investigate how the *velocity error* develops in relation to the residual. To compute the velocity error we need to compute a reference velocity first. In the following I used an iterative solver to reduce the residual until the iterate is likely to be inside the convergence region of the Newton method. Then few Newton steps were usually enough to retrieve one of potentially infinitely many exact solutions, from which we can easily calculate the reference velocity. Afterwards we can test each of the solvers and record the evolution of the velocity error.

Fig. 13 shows the residuals and velocity errors of each solver on the granular media problem used in Fig. 10 with a 0.8 friction coefficient. The PGS and CPG error graphs are closely coupled to their residual graphs. In particular the error in the PGS drops again faster in the beginning than the CPG error just like the residual did. However the velocity error of the Newton method is only loosely related to the residual. This makes the Newton method even more unattractive because it requires many iterations where the velocity error is barely reduced. Unless we iterate until the sharp drop-off occurs, the Newton method on its own is by far inferior. This property transfers to the cone friction model and for different coefficients of friction.

5.4 Active Set Stabilization

As noted earlier, the CPG heavily depends on a stable active set. Therefore a close look at the number of changes in the active set of bounds is worthwhile. The constraints belonging to the active set can be easily determined for the PGS and CPG method. Each time a component of the iterate exceeds its bounds and hasn't been projected before, the constraint enters the active set. If a component moves away from the bound, the constraint leaves the active set. This approach is not directly transferable to the Newton method because there the iterate

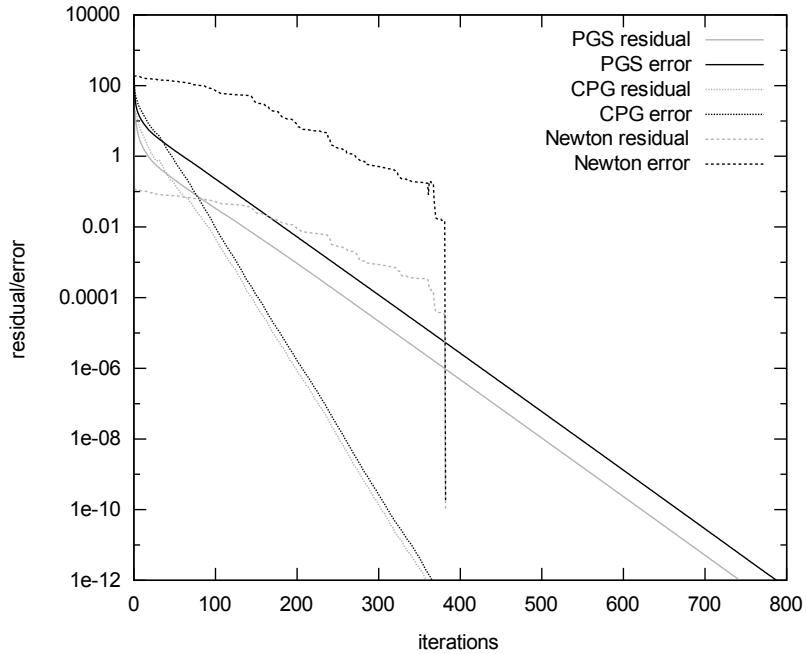
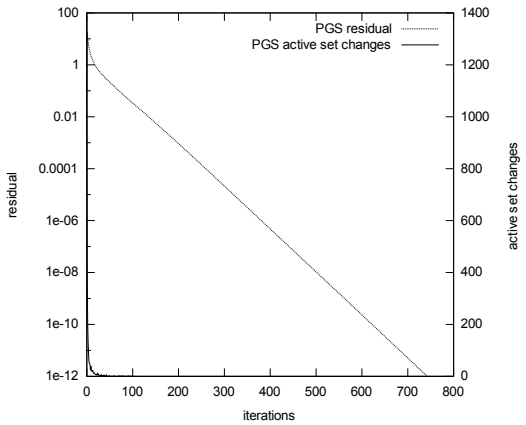


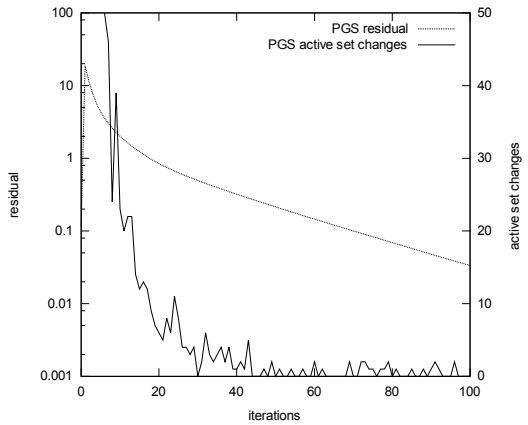
Figure 13: The residual and the corresponding error for each solver on a granular media test case and a box friction model.

is not constrained to stay within the constraint set, i.e. the components may exceed their bounds in the course of the algorithm. The consequence is that a tangential component can be constrained by several bounds simultaneously. As an alternative we will use the Jacobian (cf. Eq. (74)) to define the active bounds. If the derivative of component i belongs to the set of variables α it is lower bounded, if it belongs to γ it is upper bounded. If a variable moves from one of the sets to another, an active set change takes place. Since, in order to construct the Jacobian, we need to analyze to which set the variables belong to anyway, we get the number of active set changes practically as a byproduct.

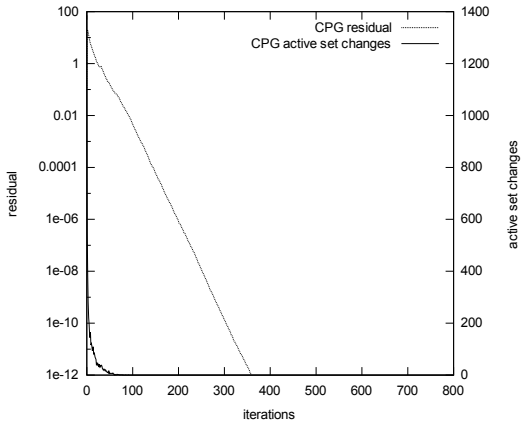
The solvers differ dramatically in their ability to stabilize the active set. Fig. 14(a), 14(c) and 14(e) plot the residual in a logarithmic scale and the number of active set changes in a non-logarithmic scale per iteration for the PGS, CPG and Newton method. The test problem is again a granular media test case though the results are qualitatively the same for other problems. The PGS finds the active set the fastest, closely followed by the CPG method. The Newton method is clearly worse. PGS and CPG reduce the active set changes *exponentially* whereas the Newton method has much larger variations and the number of changes decreases only linearly on average. Fig. 14(b), 14(d) and 14(f) are enlarged details. Note that it takes the Newton method four times the number of iterations to find a stable active set. This observation suggests to combine the PGS's strength to find the correct active set with other solvers which are better at reducing the remaining errors. One particular realization could be to stray in Newton steps as soon as the active set stabilized and constrain the Newton search direction to the constraint set. The latter is important because otherwise the PGS will immediately project back the iterate and introduce a possibly large residual. As soon as the active set is correct, the exact solution will be at hand. Promising research in this direction has been done in Morales et al. [11].



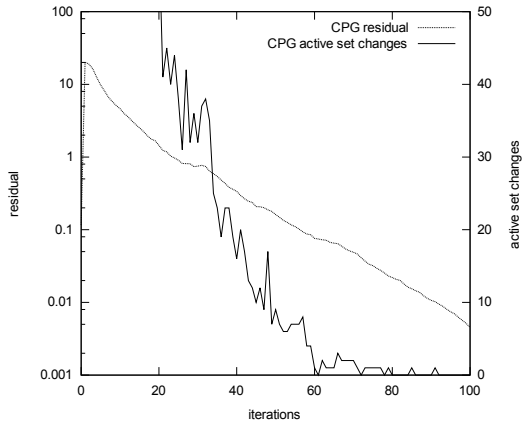
(a) Active set evolution of the PGS method.



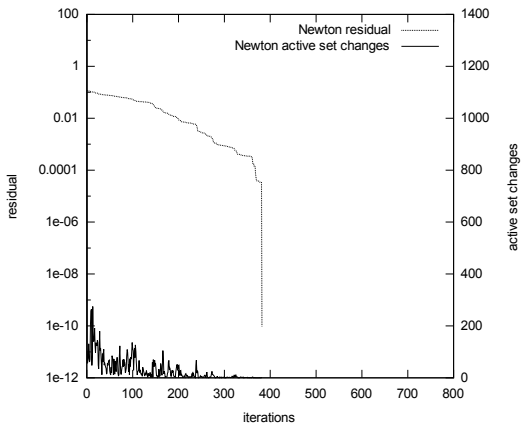
(b) Detail enlargement of the active set evolution of the PGS method.



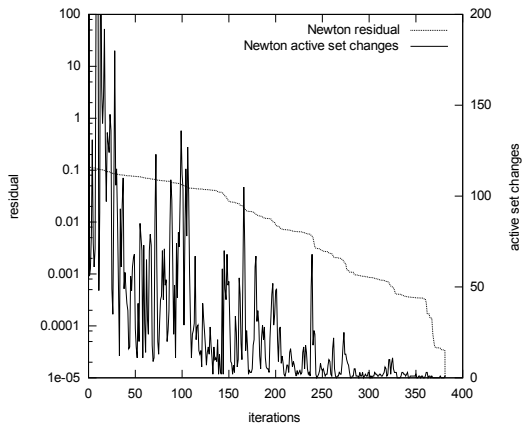
(c) Active set evolution of the CPG method.



(d) Detail enlargement of the active set evolution of the CPG method.



(e) Active set evolution of the Newton method.



(f) Detail enlargement with *different scale* of the active set evolution of the Newton method.

Figure 14: Active set changes per iteration of different solvers.

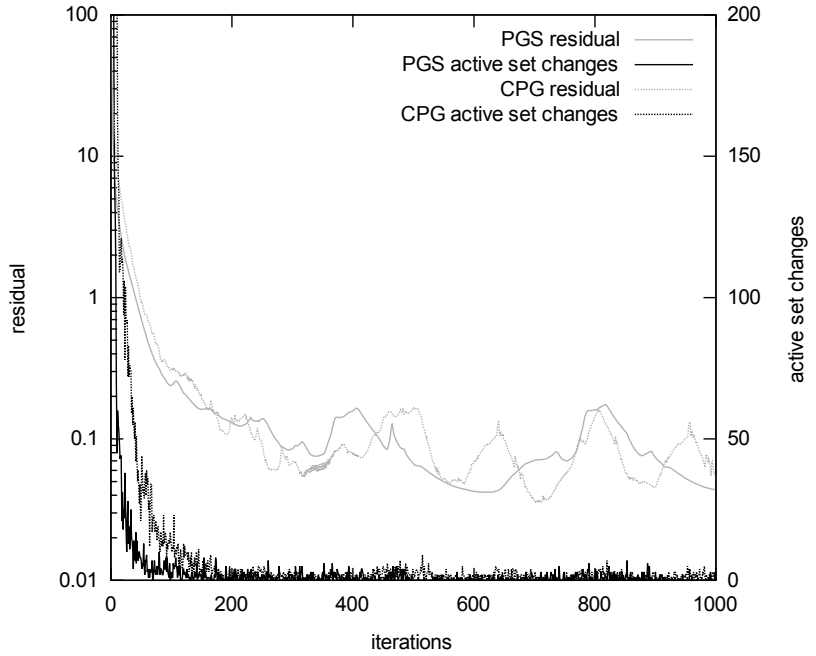
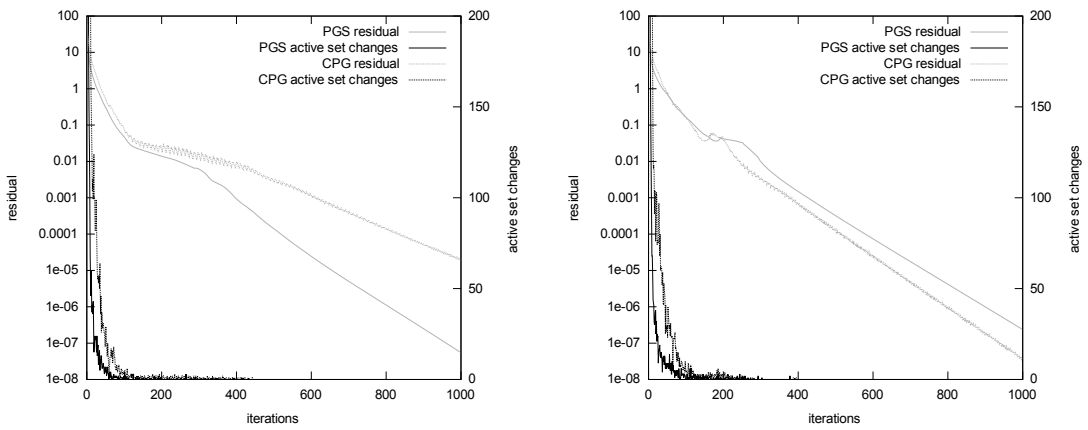


Figure 15: Oscillating residual plots of the PGS and CPG method processing a granular media test case with strong friction.

5.5 Convergence Issues

The convergence of the cone frictional contact problem solvers, as indicated in the summarizing Tab. 3, is not guaranteed. Fig. 15 shows residual plots where the PGS and CPG solvers fail to reduce the residual and instead oscillate. The test problem is a large granular media test case with a *cone friction* model and a high friction coefficient. The strong friction is responsible for the non-convergence, because the very same problem for friction coefficients $\lesssim 0.74$ converge. However, what can be done to achieve convergence in the strong friction case? First of all the NSCD method comes into mind. If the friction box adaptations cause the non-convergence, then the NSCD should converge. Recall that granular media test cases have diagonal blocks on the diagonal. This peculiarity makes the two methods equivalent. Thus the NSCD method reduces to the PGS, where we relax the normal component before relaxing the corresponding tangential components. Indeed the residual evolution of the NSCD method perfectly matches the one in Fig. 15. Thus the NSCD method can fail just as the PGS and CPG methods because the iterate *does not contract* to a fixed-point. One option to address the convergence problems is to modify the system matrix in order to improve its numerical properties. As suggested before in section 4.4.6, we can carefully apply diagonal scaling to obtain positive definiteness. Fig. 16(a) shows the residual plots for the same test case but diagonal scaling by 101% activated.

Altering the system matrix is not always desirable. Another potential improvement could be to apply a line search on the difference of the iterates of two successive splitting method iterations. The line search then ensures the reduction of a merit (objective) function like $\frac{1}{2}\vec{x}^T\mathbf{A}\vec{x} + \vec{x}^T\vec{b}$ and additionally restricts the iterate to the constraint set. In Cottle et al. [8], section 5.5 this idea is explained for standard LCPs. Though the PGS and NSCD methods could benefit from the line search, the CPG already performs line searches. So in this case the adaptations of the friction boxes cause the oscillation. Please bear in mind that the line searches are slightly different. The CPG line search is not limited to the constraint set to



(a) Diagonal scaling by 101% is enough for convergence.

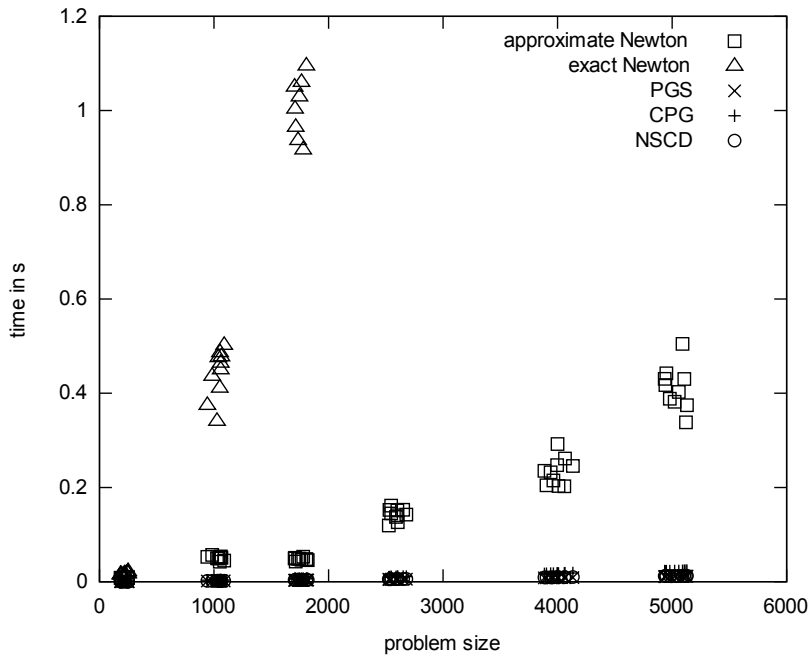
(b) The same problem with lower friction coefficient 0.64 converges too.

Figure 16: Residual plots of the PGS and CPG method processing a granular media test case with strong friction.

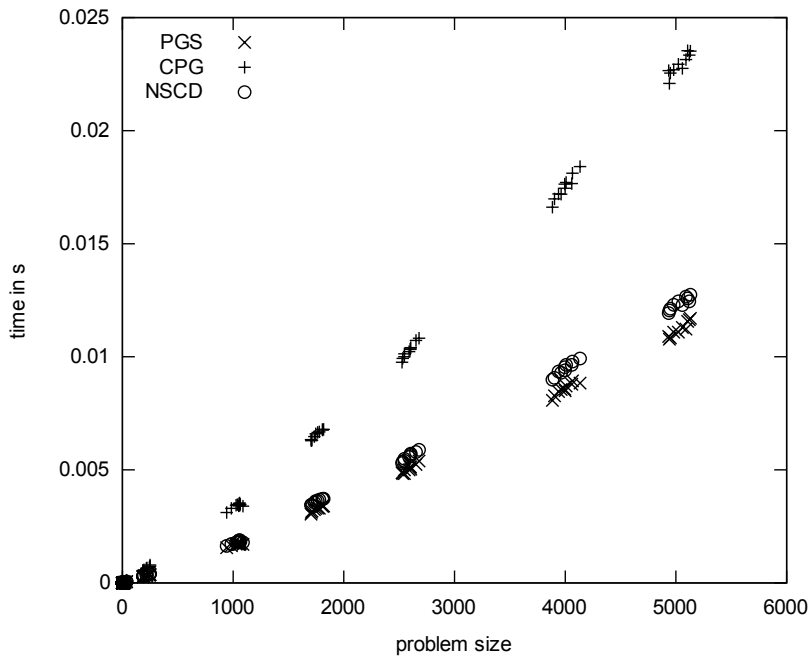
allow the method to add as many constraints as possible within one iteration. Not until after the line search the iterate is projected back to the constraint set.

5.6 Time Measurements

Up to now only the number of iterations were analyzed. However, the time spent by each solver for one iteration is largely different. All but the Newton method's iteration have linear complexity assuming the number of non-zero entries per row is bounded above by some arbitrary constant d. This is a reasonable assumption because the number of neighbors of a contact cannot grow infinitely with the number of total contacts. For example the PGS sweep is basically a matrix-vector multiplication in $\mathcal{O}(n \cdot d)$ and some additional vector operations in $\mathcal{O}(n)$. The CPG needs two matrix-vector multiplications and several vector operations and the NSCD method is basically a PGS where the subproblem solution is more complex but still in constant time. The Newton method in contrast must solve a subsystem of linear equations in each iteration. Here the time spent heavily depends on the subsystem solver applied, the fraction of bounded variables and the condition number of the matrix in case of a CG subsystem solver. Fig. 17(a) illustrates the average durations in seconds per iteration of the PGS, CPG, NSCD and Newton method with a CG and one with a GmRes subsystem solver against the problem size measured in number of unknowns. All but the Newton methods approximately show linear growth in the number of unknowns as expected. Fig. 17(b) shows only the PGS, CPG and NSCD times. The CPG iterations need roughly twice the time of the PGS or NSCD iterations. This is because the CPG needs to evaluate two matrix-vector multiplications instead of one. Thus the matrix-vector multiplication clearly is the most expensive part and the other vector operations are of nearly no consequence. The same applies to the NSCD iterations, which are only marginally more expensive than the PGS iterations since the subproblem solutions do not carry weight compared to the matrix-vector multiplication. The measurements were carried out on a 2.2 GHz processor. The time values are averages over 100 iterations on a particular problem. The problems were again granular media test cases of different sizes.



(a) PGS, CPG, NSCD and Newton iteration times.



(b) PGS, CPG, NSCD iteration times.

Figure 17: The time spent per iteration against the number of unknowns.

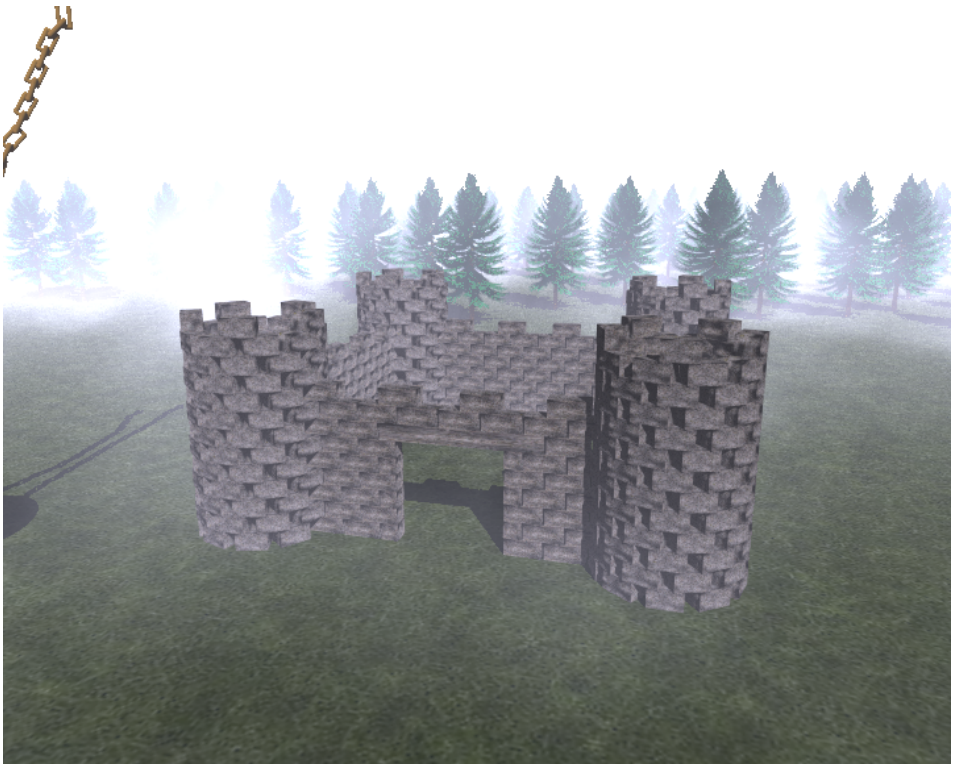


Figure 18: Initial configuration of the castle simulation.

5.7 Robustness

Though we only investigated granular media test cases, the solvers are by no means limited to spheres. In the following the stability of the models and robustness of the solvers shall be demonstrated. Fig. 18 shows the initial scene. The castle is composed of 689 bricks. Next to one of the towers a battering ram is placed, which is hung up above the scene at two chains. Each chain consists of 25 chain links. The number of contacts in the initial scene amounts to 7098 contacts and 21294 unknowns respectively. The time steps need to be very small because of the rather thin chain links. Otherwise interpenetrations would accumulate over the time and lead to the breakage of the chains. For our purposes we chose a time step of 10^{-4}. Please note that for the castle alone much larger time steps could be used without losing any stability. This is an advantage due to the implicit character of the discretization. The CPG method was chosen as a solver and a box friction model was adopted. In each time step a frictionless contact problem was solved first. The solution was used to estimate the friction bounds and additionally served as an initial solution of the box friction model. The friction coefficient was constantly 0.4. It turned out that the iterate delta is well suited as a stopping criterion for the solver. The threshold of $5 \cdot 10^{-7}$ was more than necessary to achieve a stable simulation. It offers a good trade-off between accuracy and computational burden. The scene was simulated for 170000 time steps resulting in a simulated time of 17 seconds.

(a) The battering ram touches one of the towers for the first time.

(b) The battering ram penetrates the outer wall of one of the towers.

Figure 19: The destruction of the castle.

(a) The bricks are tossed aside and the battering ram swings back.

(b) Configuration of the castle simulation after 170000 time steps.

Figure 20: The castle relicts.

6 Conclusion

6.1 Future Work

The rigid multibody contact problems proof to be hard to solve. There were numerous new publications on solution methods for multibody contact problems but still there is no convincing break through. The most reliable model is the box friction model and the CPG method decently solves them. More research is needed for sophisticated preconditioners. By repeating the whole solution process a successive approximation for the friction forces can be found. Instead of wrapping a fixed-point iteration around the CPG, it is possible to adapt the bounds in each iteration. But especially high friction coefficients deteriorate the convergence. Even convergence failures cannot be obviated. Splitting methods are no exceptions, neither the inexact PGS nor the NSCD method. Though none of them were presented, the splitting methods can fail even on frictionless problems. But still they are most often used as a solver since they are best at efficiently reducing the error in the beginning, easy to implement and the computational costs per iteration are low. Newton iterations in contrast are expensive but can compute the exact solution efficiently if the active set is accurate. Estimation of a proper active set is carried out best by the splitting methods and worst by the Newton method. Therefore a combination appears to be beneficial. The multigrid approach could benefit from a stable active set too but more fundamental problems have to be solved first.

All solvers employ or can profit from line searches. Several interesting problems arise. The CPG line search for instance does not consider bounds because as many constraints as possible should be able to become active in a single iteration. A more elegant solution would be to restrict the line search to the constraint set and update the search direction instead of advancing to the next iteration. Similar ideas can be adopted for the Newton method. Furthermore the possibility of extending the splitting methods by line searches should be explored. Also the influence of improved line searches, like line searches which exploit the nearly quadratic shape of the function, should be investigated. Unfortunately often the Newton search direction is not very good and there is nothing the line search can do about it. One possible resort could be smoothing techniques.

On the other hand, not all modelling options were explored. As briefly mentioned in section 4.1.4.2, the contact problem can be modelled approximately as a CCP with the friction cones as a constraint set. This model has the potential to become another trade-off between accuracy and efficient solvability - more accurate than box friction but less accurate than cone friction models. The CCP can be reformulated into a monotone QP with cone constraints similar to the box friction model. Though this QP is not box-constrained, it is still possible to alter the CPG projections to compute the CCP solution efficiently. Devising such a scheme and comparing it to existing splitting-methods (e.g. [4]) should definitely be part of future work.

An important extension to the models proposed are models which incorporate not only unilateral contacts but also bilateral contacts and other joints. Instead of constructing the complete system with all auxiliary variables I propose to only setup a matrix which transforms $3N$ contact forces into $3N$ relative contact velocities. The actual subproblems can be constructed as needed and the solution can be converted into the actual force acting at the associated point (contact). Each joint or contact has its own subproblem solver for example a total enumeration scheme or a potentially inexact projection solver or possibly a sophisticated nonsmooth Newton solver. The number of variables is kept small and even the contact model can be chosen for each contact separately this way. Optionally inexact subproblem solvers

can also be exchanged by exact subproblem solvers during the solution process if necessary. The extension of the CPG and Newton methods in contrast requires the construction of the complete complementarity system.

Though this thesis exclusively deals with rigid multibody dynamics, the same principles apply to soft body dynamics too. The modeling of ropes, cloths and deformable bodies would considerably enrich the range of possible simulations. Naturally the number of necessary contacts will increase, but the stronger structuredness could make (geometric) multigrid approaches beneficial. The algebraic multigrid method needs more attention too but should be tested on non-singular problems first. From the theoretical point of view convergence results for the solvers in general would be interesting. As observed, the solvers are not necessarily capable of exactly solving the problems. But even if they would always provide the precise solution, accumulation of error cannot be prevented due to modeling insufficiencies and discretization errors. Therefore a mechanism which at the expense of adding energy to the system resolves existing interpenetrations is needed.

6.2 Results

In the scope of this thesis, the non-smooth mathematical foundation for simulating rigid multibody systems was described. The derivation points out any simplification and thus is a good starting point for deriving even more advanced models. The consistent treatment of the normal and tangential components makes the formulation unique and elegant and the complementarity system becomes as simple as $\Phi(q(t+\Delta t)) \gtreqless \vec{0} \perp \underline{x}(\vec{x}) \leq \vec{x} \leq \overline{x}(\vec{x})$. The specifics of this problem were analyzed thoroughly and interesting properties were identified. Starting from the uniform formulation numerous models were presented, which range from frictionless to complex cone frictional models. Three established solution methods were subsequently implemented. The implementations follow generic guidelines which enable the easy and efficient replacement of the projection operators but still offer high-performance code. Consequently minimum effort was needed to derive contact solvers for every possible solver model combination. For the well-known projected Gauss-Seidel splitting method several block variants were suggested. One of them uses projections onto the cross-section of the circular friction cone. It was shown that if the solver converges, then the solution is at least partially dissipative and for spherical granular media even maximally dissipative. Another block variant was introduced, where a block contains all components of one contact. To solve the subproblems, a total enumeration scheme was suggested, which considerably reduces the order of the subproblem. Numerical tests show that the method performs comparably to the PGS but typically needs less iterations. Aside from splitting methods, the conjugate projected gradient method which was suggested by Renouf and Alart was investigated and details were adapted in order to design it in a more generic fashion. In this context the connection to optimization problems was presented. Last but not least a non-smooth Newton method was analyzed and implemented. The asymmetric subproblems often turned out to be singular. A computationally much more attractive variant was devised, where the idea of friction box adaptations was adopted. Notably the symmetric subproblems are provably solvable. Numerical tests were conducted and showed that the simplified Newton method compares to the exact Newton in terms of convergence but is considerably faster because of the easier to solve subproblems. In addition to the established solution methods for multibody dynamics a new purely algebraic multigrid approach was tested which builds on the splitting methods' strengths to reduce local errors efficiently. The approach was tested, problems identified and possible remedies suggested. Thus a total of sixteen solver variants were implemented and

integrated into the physics framework $\mathcal{P}e$[3]. Finally numerical tests were carried out to validate and compare the solvers. Strengths and weaknesses were identified and approaches to exploit them were suggested. The availability of a large number of solvers and their modular implementations considerably facilitates the experimentations with solver combinations and thus possibly paves the way for the construction of a superior all-in-one solver.

[3]The physics engine $\mathcal{P}e$ is developed at the computer science department for system simulation at the Friedrich-Alexander University Erlangen-Nürnberg

References

[1] M. Anitescu. Optimization-based simulation of nonsmooth rigid multibody dynamics. *Math. Program.*, 105(1):113–143, 2006. ISSN 0025-5610. doi: http://dx.doi.org/10.1007/s10107-005-0590-7.

[2] M. Anitescu and Gary D. Hart. A fixed-point iteration approach for multibody dynamics with contact and small friction. *Math. Program.*, 101(2):3–32, 2004. doi: http://dx.doi.org/10.1007/s10107-004-0536-6.

[3] M. Anitescu and F.A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *ASME Nonlinear Dynamics*, 1997.

[4] M. Anitescu and Alessandro Tasora. An iterative approach for cone complementarity problems for nonsmooth dynamics. 2007. URL http://www.mcs.anl.gov/~anitescu/PUBLICATIONS/projtheo.pdf.

[5] Stephen C. Billups and Katta G. Murty. Complementarity problems. *J. Comput. Appl. Math.*, 124(1-2):303–318, 2000. ISSN 0377-0427. doi: http://dx.doi.org/10.1016/S0377-0427(00)00432-5.

[6] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. ISBN 0-89871-462-1.

[7] P.W. Christensen, A. Klarbring, Jong-Shi Pang, and N. Strömberg. Formulation and comparison of algorithms for frictional contact problems. *International Journal for Numerical Methods in Engineering*, 42:145–173, 1998.

[8] R.W. Cottle, Jong-Shi Pang, and R.E. Stone. *The Linear Complementarity Problem*. Academic Press, Inc., 1992.

[9] J.E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983. ISBN 0-13-627216-9.

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

[11] J.L. Morales, J. Nocedal, and M. Smelyanskiy. An algorithm for the fast solution of linear complementarity problems. Technical report, Instituto Tecnológico Autónomo de México, 2007. URL http://www.optimization-online.org/DB_FILE/2007/06/1675.pdf.

[12] Ricardo Ortiz-Rosado. *Newton/Amg Algorithm for Solving Complementarity Problems Arising in Rigid Body Dynamics with Frictional Impacts*. PhD thesis, University of Iowa, July 2007.

[13] Jong-Shi Pang. Newton's method for b-differentiable equations. *Mathematics of Operations Research*, 15(2), May 1990.

[14] Tobias Preclik. *Frictional Rigid Body Dynamics*. student thesis, Friedrich-Alexander University Erlangen-Nuremberg, 2007. URL http://www10.informatik.uni-erlangen.de/Publications/Theses/2007/Preclik_SA_07.pdf.

[15] M. Renouf and V. Acary. Comparison and coupling of algorithms for collisions, con-

tact and friction in rigid multi-body simulations. In *Computational Mechanics: Solids, Structures and Coupled Problems*, Lisbon, Portugal, June 2006.

[16] M. Renouf and P. Alart. Gradient type algortihsm for 2d/3d frictionless/frictional multi-contact problems. In *ECCOMAS 2004 Proceedings*, University of Jyväskylä, July 2004.

[17] M. Renouf and P. Alart. Conjugate gradient type algorithms for frictional multi-contact problems: applications to granular materials. *Computer Methods in Applied Mechanics Engineering*, 194:2019–2041, July 2005.

[18] M. Renouf, V. Acary, and G. Dumont. 3d frictional contact and impact multibody dynamics: A comparison of algorithms suitable for real-time applications. In *Multibody Dynamics. Computational Methods and Applications*, Universidad Politecnica de Madrid, June 2005.

[19] Jörg Sauer and Elmar Schömer. A constraint-based approach to rigid body dynamics for virtual reality applications. In *Proceedings of the ACM symposium on Virtual reality software and technology*, 1998.

[20] David E. Stewart and J.C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering*, 39:2673–2691, 1996.

[21] J.C. Trinkle, Jong-Shi Pang, Sandra Sudarsky, and Grace Lo. On dynamic multi-rigid-body contact problems with coulomb friction. Technical report, Texas A & M University, College Station, TX, USA, 1995.

[22] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.