Diss. ETH Nr. 22546

Entwicklung eines Bobsimulators

Abhandlung zur Erlangung des Titels

Doktor der Wissenschaften der ETH Zürich

(Dr. sc. ETH Zürich)

vorgelegt von

Georg Stefan Rempfler

MSc ETH Masch.-Ing.
geboren am 13. November 1984
von
Appenzell, Appenzell Innerrhoden
Schweiz

angenommen auf Antrag von

Prof. Dr.-Ing. Dr.-Ing. habil. Ch. Glocker, Referent
Prof. Dr. R. Riener, Korreferent

2015

Meinen Eltern

Danksagung

Während meiner Tätigkeit am Zentrum für Mechanik an der ETH Zürich entstand der in dieser Arbeit beschriebene Bobsimulator. Zum Gelingen dieses Projektes haben ganz viele Personen beigetragen, denen ich zu Dank verpflichtet bin.

An erster Stelle bedanke ich mich bei Herrn Prof. Glocker für die gebotene Möglichkeit mir in diesem Projekt Programmierkenntnisse anzueignen und die nichtglatte Dynamik an einem nützlichen Beispiel anzuwenden. Ich habe die Freiheit, die mir in der Ausgestaltung des Projektes belassen wurde, und das mir entgegengebrachte Vertrauen geschätzt. Prof. Glocker ist einer meiner grössten Lehrer. Seine sorgfältig konzipierten und fundiert ausgearbeiteten Vorlesungen und sein Überblick in der Mechanik vermögen einen zu begeistern und hinterlassen anwendbares Wissen, auf das man gerne und selbstbewusst zurückgreift. Die Passion mit der Prof. Glocker Mechanik betreibt wir mir Vorbild sein für mein weiteres Streben. Seine Lehre und die Einstellung zur Ausbildung werden mir als Ideale dienen. Ich schätze mich glücklich während meiner Zeit als Assistent eigene Leidenschaften entdeckt zu haben und bin dankbar gelernt zu haben, eigene Überzeugungen zu erarbeiten, eigener Meinungen bewusst zu werden, und den Mut aufzubringen zu solchen zu stehen.

Ein weiterer Dank geht an Herrn Prof. Riener für die Übernahme des Korreferats. Die spontane Zusage hat mich gefreut. Seine unkomplizierte Art machen den Umgang mit ihm zu einem Vergnügen.

Ein spezieller Dank gebührt meinen ehemaligen Kollegen Michael Möller und Pascal Arnold. Ihre Beiträge haben entscheidend zum Erfolg des Bobsimulator beigetragen. Von Michael Möller stammt der gesamte Quellcode für die Kollisionsdetektion und das Rendern der Szenerie. Auch die Programmierung einer Mehrkörpersimulation habe ich von ihm gelernt. Pascal Arnold seinerseits brachte die *Citius* Bobschlitten hervor. Im Bobsimulator steckt viel Know-How des *Citius* Projektes und Pascal Arnold hat sich speziell beim Erarbeiten des Mehrkörpermodells beteiligt. Ein grosses Dankeschön für eure Unterstützung und die gute Arbeit.

Ganz speziell bedanke ich mich weiter bei Luca Somaini und der Hocoma AG. Luca hat mich bei der Konzeption und Umsetzung des Force Feedback Systems spontan und tatkräftig unterstützt. Darüberhinaus hat er mir auch funktionierende Hardware der Hocoma AG vermittelt. Für die grosszügig zur Verfügung gestellte Hardware sei der Hocoma AG gedankt. An Luca geht ein warmer Dank für die treue Hilfe und die vielen aufgewendeten Stunden. Es ist beglückend ihn als Freund zu wissen.

Auch den diversen Bobpiloten, die mitgewirkt haben, sei herzlichst gedankt: Gregor Baumann, Reto Götschi, Sabina Hafner, Beat Hefti, Rico Peter, Christian Reich, Marcel Rohner, Ivo Rüegg, Caroline Spahni und Fabienne Meyer. Ihre Bereitschaft den Simulator zu nutzen und mitzuwirken hat mich stets motiviert und mir Freude an meiner Arbeit verschafft. Ebenso herzlich bedanke ich mich bei Martin Casal für seinen unermüdlichen und selbstlosen Einsatz; nicht nur, aber speziell auch für die diversen Transporte des Simulators und die Woche an der Sportech Messe im Tessin. Ich freue mich über ihn bei Hans Kohler ein neues Zuhause und damit eine Zukunft für den Bobsimulator gefunden zu haben; schon vorab vielen Dank an Hans Kohler für diese grosszügige Möglichkeit.

Ueli Marti, Jean-Claude Tomasina, Beni Cadonau, Stephan Kaufmann und Gabriela Squindo halten das Zentrum für Mechanik in Schwung und erfüllen es mit einer Seele. Meine Anliegen an die Werkstätten, die Informatik und den administrativen Bereich fanden bei ihnen stets Gehör und wurden ausnahmslos speditiv und zur vollsten Zufriedenheit erfüllt. Dafür gebührt ihnen Lob und sei ihnen herzlich gedankt.

Viele weitere Personen des Zentrums für Mechanik haben mich in verschiedensten Rollen unterstützt. Auch ihnen sage ich innigst Dank: Prof. Remco Leine, Adrian Schweizer, Marc Farine, Thomas Heimsch, Simon Eugster, Michael Baumann, Gabriel Nützi, Christian Maier, Ueli Aeberhard, Stefan Blunier, Manfred Maurer, Johannes Weickenmeier, Arabella Mauri, Johannes Hengstler, Dirk Möller, Philipp Hahn und Raoul Hopf. Bei allen Professoren des Zentrums bedanke ich mich für die offene Atmosphäre und speziell für die grosszügigen Skiwochen.

Meine Eltern haben mir das Studium des Maschineningenieurwesens ermöglicht, mich grosszügig unterstützt und mir immer die Freiheit geschenkt zu tun und lassen was mir beliebt. Ich vermag meine Dankbarkeit für diese Möglichkeiten, die stete Unterstützung meines Seins und Schaffens, und die geliebte Geborgenheit nicht in Worten auszudrücken. Das Alles macht mich jedenfalls zu einem privilegierten, glücklichen und zufriedenen Menschen. Dies möge ihnen einst vergolten sein. Bei meiner ganzen Familie mit Janine Fuchs bedanke ich mich für den unmissverständlichen Rückhalt, der mir die Kraft gibt auch schwierige Zeiten mit Zuversicht zu durchlaufen. Janine Fuchs hat mich darüber hinaus viele freudlose Tage ausgehalten, wofür ich ihr Respekt zollen und von Herzen danken möchte. Ich wünsche meiner ganzen Familie und meinen Freunden, dass uns noch viele heitere Stunden bevorstehen und der Humor immer wieder zu uns zurückkehrt.

Georg Rempfler

Zürich, im Februar 2015

Zusammenfassung

Die vorliegende Dissertation beschreibt die zentralen Elemente eines Bobsimulators, der im Rahmen dieses Projektes entwickelt, aufgebaut und getestet wurde. Er dient Profis und Amateuren in erster Linie als Trockentrainingsgelegenheit, eignet sich aber auch für die Analyse von Bahndesigns oder Konstruktionsvarianten von Schlitten. Die Simulation einer gesteuerten Bobfahrt ist eine prädestinierte Anwendung für die Methoden der nicht glatten Dynamik. Ihre Formulierung von mengenwertigen Kraftgesetzen in Form von Normalkegelinklusionen eignet sich insbesondere für die Modellierung des einseitigen, reibungsbehafteten Kontaktes zwischen Kufe und Eis. Mit der Formulierung von Reibstössen bei denen impulsive Kräfte zu Sprüngen in den Geschwindigkeiten führen, ist das temporäre Abheben der Kufen vom Eis im Modell enthalten und sind auch Schläge gegen die seitliche Berandung der Bahn mitberücksichtigt. Mit der iterativen Lösung des Inklusionsproblems, das sich in einem Zeitschritt für die Kontaktkräfte stellt, und den Time-Stepping Verfahren zur numerischen Integration sind robuste Instrumente für die Simulation nicht glatter Modelle vorhanden. Aufgrund der im Bobsport üblichen hohen Geschwindigkeiten von bis zu 150km/h stellt die Simulation einer Fahrt einen Prüfstein für die Effizienz der Methoden dar. Denn die Simulation hat in Echtzeit zu erfolgen und muss dabei stets flüssig auf der Leinwand dargestellt werden können.

Für den Bobschlitten mitsamt Mannschaft wird ein Mehrkörpermodell aus Starrkörpern entwickelt, das die wesentlichen Freiheitsgrade beinhaltet. Hier kann vom Know-How des Projektes *Citius* profitiert werden. Einzelne Kraftelemente im Schlitten sind der effektiven Bauart oder Messungen der modernen Schlitten nachempfunden. Für die anisotrope Reibung zwischen den Kufen und dem Eis wird ein eigenes Modell erarbeitet, das verschiedene Aspekte der Interaktion berücksichtigt. Insbesondere ist die mangelnde Lenkbarkeit eines Schlittens unter fehlender Last nachempfunden. Weiter kann der Schlitten von der Eisoberfläche abheben und damit auch stürzen. Er stösst aber nicht nur mit den Kufen gegen das Eis, sondern kann mit den seitlichen Bumpern auf Geraden an die Eiskanalwand oder in Kurven an die Bretterverschläge stossen. Solche Schläge werden häufig als Fehler taxiert, können aber auch einen driftenden Schlitten wieder ausrichten und so trotz Geschwindigkeitverlust nützlich sein. Das Mehrkörpermodell interagiert mit einem Modell der Bahnoberfläche. Anhand von Konstruktionsdaten wird die Betonoberfläche von Kunsteisbahnen rekonstruiert. Aus Bildmaterial und Videos werden zusätzliche Daten aufgenommen für das Hinzufügen von Wänden, Bretterverschlägen und Absperrungen von weiteren Starteinläufen in die Bahn. Das Bahnmodell wird in der dreidimensionalen Szenerie mit weiteren umgebenden Details ausgeschmückt. So können Schattenwürfe, Lampen

und Sonnenblenden verschiedene, reale Fahrsituationen nachahmen. Mit Texturen oder der Platzierung von Objekten können auch Orientierungspunkte der Athleten eingebaut werden.

Der Simulationscode nutzt die Baumstruktur des Mehrkörpermodells aus und stellt die Bewegungsgleichungen für einen Zeitschritt numerisch auf. Als Einführung in die Programmierung von Mehrkörpersimulationen werden in dieser Arbeit die wesentlichen Objekte der Softwarearchitektur beschrieben und deren Interaktion anhand von Python-Code gezeigt. Die vorgestellte Methode eignet sich speziell für den stufenweisen Auf- oder Ausbau eines Mehrkörpermodells, in dem wiederholt neue Körper, Freiheitsgrade oder Kraftelemente eingebaut werden. Grundsätzlich werden im Code die Bewegungsgleichungen in Minimalgeschwindigkeiten formuliert, weshalb neben den einseitigen Kontakten keine weiteren Bindungen berücksichtigt werden müssen. Gleichzeitig geschieht die Modellierung des Mehrkörpersystems im dreidimensionalen Anschauungsraum, was den Prozess sehr intuitiv macht.

Das User Interface des Bobsimulators ist bewusst minimal und intuitiv gehalten. Es erlaubt den Bobpiloten den Simulator ohne Anleitung zu nutzen und selbstständig zu trainieren. Über eine Eingabemaske lassen sich verschiedene Bobsetups individuell zusammenstellen, speichern und laden. Dabei werden einige Einstellungen zur Auswahl vorgegeben, die verschiedene real existierende Federsätze widerspiegeln. Der Simulator bietet neben einer (foto-)realistischen Darstellung der Szenerie auch Audio- und haptisches Feedback.

Abstract

This thesis describes the core elements of the bobsleigh simulator, that has been developed, built and tested within this project. It serves as an all year training opportunity for professional pilots and amateurs. Moreover, it leverages the analysis of track and bobsleigh designs. The simulation of a bobrun is a predestined application for the methods of non-smooth dynamics. Its formulation of set-valued force laws as normal cone inclusions is especially well suited for the modelling of the unilateral, frictional contact between a slider and the ice. With the formulation of frictional impacts, at which impulsive forces lead to a discontinuity in the velocities, the sliders may take-off from the ice and bumper hits against the side walls of the track are incorporated in the model too. With the iterative solver for the inclusion problem, that is stated for the contact forces in each time step, and the time-stepping schemes for the numerical integration of the motion, robust tools are given for the simulation of non-smooth models. Due to the high velocities of up to 150km/h occurring in the sport of bobsleighing, the simulation of a bobrun marks a benchmark problem for the efficiency of these methods. The simulation has to be performed in real-time and must be smoothly displayed on a screen.

A multi-body model with the essential degrees of freedom consisting of rigid bodies is developed for the bobsleigh including the two-men team. Here, it was made use of the knowledge from the former project *Citius*. Some force elements in the model mimic the construction of the modern *Citius* bobsleighs, or measurements taken of them. For the anisotropic friction between the sliders and the ice a new model is introduced, which reflects different aspects of the interaction. Especially, the reduced controllability of a bobsleigh under low load is replicated. Moreover, the bobsleigh can lift off the ice and therefore may turn-over. However, it does not only impact against the environment over the sliders. The side bumpers can collide with the walls of the track on straight sections or hit the wooden planking in curves. Such bumps are often considered a driving mistake, but they may also bring a drifting bobsleigh back into straight direction and can therefore be helpful too, despite the accompanying loss in velocity. The multi-body model interacts with a model of the track surface, which is reconstructed from the construction data of artificial ice tracks. Additional data is gathered from images and videos for to add side walls, plankings in curves and barriers at alternative starts further down the track. The model of the track is accompanied with further details in the surrounding three-dimensional scenery. Different realistic driving situations can be rendered. Shadows and therewith transitions to enlightened areas may disturb a pilot. Artificial light from lamps creates an entirely different situation and closed sun shades tunnel the view on the track. With textures or

by placing additional objects reference points of the athletes can be included in the scene.

The simulation code makes use of the tree structure of the multi-body model and numerically assembles the equations of motion in each time step. As an introduction to the programming of multi-body simulations the core objects of the software architecture are described and their interaction is presented in Python code. The method used herein is well suited for a step by step build-up or extension of a multi-body model, where new bodies, degrees of freedom or force elements are continuously added. Basically, the code formulates the equations of motion in minimal velocities. Therefore, beside the unilateral contacts, no additional constraints have to be taken into account. On the other side, the modelling of the multi-body system happens in the three-dimensional, euclidean space, which makes this process very intuitive.

The user interface of the bobsleigh simulator is intentionally kept minimal and intuitive. It enables the pilots to use the simulator without instruction, and to self-dependently exercise on it. Over an input mask the bobsetup can individually be adjusted, saved and loaded. Some options can alternatively be chosen from predefined lists, that represent the physically existing spring sets given to *Citius* pilots. Aside the (photo-)realistic rendering of the scenery, the bobsleigh simulator provides audio and haptic feedback.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Der Bobsport hat eine lange Tradition in der Schweiz und ist aufgrund wiederholter Erfolge von einheimischen Athleten immer noch populär. Dazu trägt sicherlich auch die Ausstrahlung der letzten weltweit verbliebenen Natureisbahn in St. Moritz bei. In der rund 125-jährigen Geschichte hat sich der Bobsport enorm entwickelt und professionalisiert. Heute können einzelne Zehntels- oder gar Hundertstelsekunden ein Rennen entscheiden, das über vier Läufe insgesamt um die vier Minuten dauert. Man vergleiche dazu [19, 20, 23, 22]. Gleichzeitig hat sich ein sehr striktes Reglement entwickelt, das nur noch minimalen Raum für technische Weiterentwicklungen des Materials zulässt. Daneben bleiben die Athletik aller Insassen und die Fahrkünste des Piloten als Ansatzpunkte für weitere Leistungssteigerungen. Die Fitness und Athletik eines Bobteams ist und bleibt natürlich ein entscheidendes Element für den Erfolg. Wer beim Start nicht mit den Besten mithalten kann, wird ein Rennen auf der Strecke nur noch schwerlich für sich entscheiden können. Dafür macht die Weltspitze zu wenig Fahrfehler. Vermag man beim Start mitzuhalten, sind mehrere fehlerfreie Läufe gefordert, um sich gegen die Konkurrenz durchsetzen zu können. Das fordert gute Lenkfähigkeiten und viel Erfahrung vom Piloten.

Um Lenkfähigkeiten zu verbessern und Erfahrung zu sammeln, sind möglichst viele Fahrten zu absolvieren. Da jedoch die Eisbahnen nur während einer relativ kurzen Zeit im Jahr betrieben werden können, ist das Fahrtraining eines Piloten sehr limitiert. Erschwerend kommt hinzu, dass das Fahrtraining sehr material-, zeit- und kostenintensiv ist. Eisbahnen sind häufig nicht den ganzen Tag geöffnet. Der Betrieb wird bei ansteigender Temperatur bzw. zunehmender Sonneneinstrahlung eingestellt. Ein Pilot schafft deshalb nur zwei bis vier Fahrten an einem Tag. Ein Schweizer Profipilot absolviert ungefähr 30 Läufe während der Saisonvorbereitung. Weitere 90 Läufe kommen im Laufe einer Weltcupsaison, inklusive der Weltmeisterschaften, hinzu. Das ergibt um die 120 Läufe pro Jahr. Mit einer ungefähren Laufzeit von einer Minute sind das total zwei Stunden Fahrzeit im Jahr.

Diese äusserst knappe totale Cockpitzeit motiviert die Entwicklung eines Bobsimulators, in dem in Echtzeit Läufe gefahren werden können. Dank dessen Orts- und Saisonunabhängigkeit kann ein Pilot das Training am Steuer massiv erhöhen und in kurzer Zeit wertvolle

Erfahrung sammeln. Darüber hinaus ist das Fahrtraining nicht mehr auf sehr vereinzelte Bahnen limitiert, sondern kann spezifisch auch auf saisonrelevanten Fahrbahnen absolviert werden. Das sollte nicht nur bessere Leistungen ermöglichen, sondern verringert gemäss der Sicherheitsstudie [86] auch nachweislich die Unfallgefahr. Neben dem intensiven Sammeln von allgemeiner und bahnspezifischer Lenkerfahrung, könnte der Simulator zukünftig auch zum Testen und Evaluieren verschiedener Setupeinstellungen verwendet werden. Oder eine Analyse verschiedener Lenkstrategien und Fahrlinien liefern, was insbesondere für Rennen auf neu gebauten Eiskanälen von Vorteil sein könnte.

1.2 Absicht und Umfang der Arbeit

Das Ziel der vorliegenden Arbeit ist der Aufbau eines Bobsimulators, der in erster Linie zu Trainingszwecken benutzt werden soll. Dazu sind im Wesentlichen zwei Komponenten nötig. Erstens muss eine echtzeitfähige Software programmiert werden und zweitens müssen Hardwarekomponenten zur realitätsgetreuen Interaktion mit der Software konzipiert werden. An die Software bestehen die nachfolgend beschriebenen Anforderungen. Sie muss real existierende Bobbahnen mitsamt umgebender Landschaft beinhalten. Nur damit kann dem Zweck des spezifischen Pilotentrainings gedient werden und nur Vergleiche von Piloten, die eine Bahn wirklich und in der Simulation gefahren sind, können die angestrebte Realitätsnähe beurteilen. Die Software benötigt weiter ein sorgfältig konzipiertes Mehrkörpermodell des Schlittens. Es soll die wesentlichen Freiheitsgrade und Setupmöglichkeiten umfassen und die relevante (Eigen-)Dynamik abbilden können, dabei aber nicht zu viele unbekannte oder unsichere Parameter beinhalten. Die unsicheren Parameter sind hinsichtlich der Realitätstreue der Simulation zu optimieren. Die Szenerie ist graphisch (foto-)realistisch darzustellen. Nur wenn sich ein Pilot in der dreidimensionalen Landschaft fühlt, kann es zu einer ernsthaften Akzeptanz als Fahrtraining führen. Schliesslich haben die graphische Darstellung und die einhergehende Berechnung des Bewegungsablaufs in Echtzeit zu erfolgen. Nur ein flüssiges Bild und eine unverzögerte Interaktion mit dem Mehrkörpermodell können überhaupt zu Realismus führen. Aus drei Gründen soll zuerst die Bahn von Whistler in Kanada implementiert werden. Erstens sind die Konstruktionsdaten aus dem *Citius*-Projekt verfügbar. Zweitens handelt es sich um die schnellste Bahn der Welt und ist damit bezüglich Echtzeitfähigkeit der geeignetste Prüfstein, und drittens wird diese Bahn als sehr schwierig und gefährlich eingestuft. Deshalb ist von einem grossen Nutzen für bestehende und angehende Piloten auszugehen. Zudem kann unter Umständen analysiert werden, was die Bahn so schwierig macht und zur erhöhten Sturzgefahr führt. Um potentiell eine breite Benutzerschar zu erreichen, sollte die Software robust laufen und eine intuitive Interaktion haben. D.h. sie sollte grundsätzlich unbeaufsichtigt zu benutzen sein.

Auf der Hardwareseite bietet sich ein Bobschlitten mit frei beweglicher Vorderachse an. Aufgrund des echten Lenksystems stimmen die Übersetzungen und bewegten Massen identisch überein. Der Lenkwinkel ist mit einem geeigneten Sensor zu messen und vom Simulationsrechner einzulesen. Für ein erhöhtes Realitätsgefühl sind weitere Stimulierungen vorzusehen. Neben der visuellen Darstellung und allfälligen Informationseinblendungen

können insbesondere Audiosignale und Haptikdisplays einer verbesserten Zustandswahrnehmung dienen. Die in einem Simulator fehlenden, oder zumindest (zwangsläufig) stark reduzierten, mechanischen Einwirkungen auf den Körper können teilweise über andere Sinneskanäle wettgemacht werden.

Der Einsatz eines Simulators als Trainingsgerät ist vielseitig. Grundsätzlich kann es schon hilfreich sein, die Fahrt auf einer Bahn wiederholt aus Piloten- und Aussensicht zu betrachten und so den Bahnverlauf und die Tempi auf einzelnen Abschnitten zu memorieren. Je besser ein Pilot das *Programm* einer Bahn kennt, desto eher kann er sich auf sein Steuerverhalten und die angestrebte Fahrlinie konzentrieren. Mittels solcher Videos ruft man sich auch eine länger nicht mehr gefahrene Bahn schneller in Erinnerung. Ein Simulator fördert auch das Kennenlernen der Dynamik eines Schlittens. Unerfahrene Piloten können in Trockentrainings ohne Gefahr das Verhalten des Geräts und ihre Einflussmöglichkeiten kennenlernen. Je realistischer die Simulation, desto eher profitieren auch bessere Piloten von der Fahrdynamik. Es können verschiedene Lenkstrategien getestet werden und kritische Stellen oder Fehler eruiert werden. Grundsätzlich sind in einem Simulator mehr Fahrten in kürzerer Zeit möglich. Auch die unmittelbare Besprechung einer ganzen Fahrt mit einem Trainer oder Pilotenkollegen bietet neue Chancen. Üblicherweise stehen an der echten Bahn die Beobachter in einzelnen Kurven. Unter Umständen sind zusätzlich noch Videos von gewissen Stellen verfügbar. Die Rückmeldungen für einen Piloten beschränken sich damit aber auf kurze Teilstücke der Bahn. Dabei sind erschwerend die Perspektiven von Sportler und Trainer grundsätzlich verschieden und die Diskussion erfolgt erst einige Zeit nach der Fahrt. Können Fahrten im Simulator aufgezeichnet und wiedergegeben werden, ermöglicht dies die sofortige Besprechung einer Fahrt. Sind sowohl Cockpit- wie Aussenansicht implementiert, basiert die Diskussion auf beiden bekannten Perspektiven. Der Austausch auf Grundlage der verschiedenen Blickwinkel fördert auch das Verständnis zwischen Sportler und Trainer bzw. generell unter Piloten.

Neben der Nutzung als Fahrtraining, kann der entwickelte Simulator verschiedene Analyseinstrumente bieten. Anhand eines direkten, visuellen Vergleichs mehrerer Fahrten können Fahrfehler und deren Folgen schnell demonstriert und erkannt werden. Mit quantitativen, statistischen Auswertungen von gespeicherten Laufdaten können systematische Untersuchungen, u.a. zu Fahrlinien, Bobsetups und Lenkverhalten, angestellt werden. Schliesslich kann ein Simulator auch zur Beurteilung der Sicherheit und Performance von Bahn- und Schlittendesigns hergenommen werden. Bei Bahnen müssen Fliehkraftlimiten nachweislich eingehalten werden. Ein Simulator könnte weiter auf die (erhöhte) Gefahr von Stürzen oder heftigen Schlägen aufmerksam machen. Für neuartige Bobdesigns könnte ein Simulator Anhaltspunkte zu Belastungen und (Eigen-)Fahrverhalten geben.

1.3 Überblick zu Simulationsprojekten im Bobsport, Rodeln und Skeleton

Nach Kenntnis des Autors haben sich gegen Ende der 1980er Jahre die ersten Projekte mit der Simulation einer Fahrt im Eiskanal beschäftigt. Seither gab es immer wieder Forschungsgruppen, die sich dieser Thematik widmeten. Der Grund für die anhaltende Attraktivität der Aufgabe liegt im enormen Anstieg der Computerleistung bzw. dem Aufkommen von erschwinglichen PCs selbst. Die Projekte in den 90er Jahren waren punkto Speicher, CPU Takt und Graphik noch enorm eingeschränkt. Der Leistungsanstieg und die (freie) Verfügbarkeit von Grafikbibliotheken und Modellierungssoftware eröffneten immer wieder neue Möglichkeiten.

Der vorliegende Überblick schliesst Computerspiele aus. Sie dienen der reinen Unterhaltung und stellen den Spielspass in den Vordergrund. Darunter leidet der Realismus, wodurch sie für den professionellen Einsatz ungeeignet sind. Bestehende Simulationsprojekte lassen sich grob in zwei Klassen auftrennen. Zum Einen verstehen sich einzelne Projekte als Designhilfen für den Bahn- und Bobschlittenbau. Die anderen Projekte zielen auf einen Einsatz als Trainingsgerät für die Bobpiloten ab.

Der *Internationale Bob und Skeleton Verband* (FIBT) schreibt obere Belastungsgrenzen vor, denen ein Sportler auf Wettkampfbahnen maximal ausgesetzt sein darf. Während der Konstruktion von neuen Bahnen wird die Einhaltung dieser Limiten rechnerisch abgeschätzt. Die Belastungen hängen kritisch von der Geometrie der Bahn ab. Iwashita et al. stellt in [50] aus dem Jahre 1995 ein echtzeitfähiges Simulationssystem zur Beurteilung von Bahndesigns vor. Es modelliert einen Bobschlitten als Punktmasse, an der unter anderem auch eine Steuerkraft angreifen kann. Die Punktmasse gleitet vorwärts entlang der Profillinie der Bahn. Die Position innerhalb eines Bahnquerschnitts, quer zur Fortschreitrichtung, wird (vermutlich) über ein Kräftegleichgewicht bestimmt. Das Simulationssystem beinhaltet bereits eine graphische Darstellung der momentanen Schlittenposition in der Bahn. Die Echtzeitfähigkeit darf allerdings mit Blick auf die heutigen Anforderungen an Bildrate und Simulationsgenauigkeit relativiert werden. Trotzdem können mit diesem System die Einhaltung der Belastungsgrenzen beurteilt und kritische Querschnitte eruiert werden, in denen der Bob sich überdrehen würde. Mössner et al. verwendet in [68] von 2011 ähnliche Modelle als Hilfsmittel für die Konzipierung der Vorgaben entsprechenden Rodelbahnen. Eine (ungesteuerte) Punktmasse bewegt sich auf einem vereinfachten Bahnmodell. Die Bahnquerschnitte werden als Viertelellipsen approximiert, was insbesondere für ältere Bahnen, wie z.B. Lillehammer, als zulässig erachtet wird. Die Berechnung der Dynamik der Punktmasse teilt sich in zwei gekoppelte Probleme auf. Anhand des Bewegungszustandes wird der Fahrpfad der Punktmasse berechnet. Eine eindimensionale Bewegungsgleichung beschreibt die Dynamik auf dieser Trajektorie, und deren Lösung führt wiederum zum Bewegungszustand.

Auch Braghin et al. widmet sich der Validierung von Bobbahnen. In [9] von 2010 wird ein Werkzeug für Simulationen von passiv gesteuerten Bobfahrten präsentiert. Die Oberfläche der Bahn wird aus den Konstruktionsdaten und Querschnittzeichnungen im

dreidimensionalen Raum nachgebildet. Das Bobschlittenmodell besteht aus neun starren Körpern und es interagiert mit der Bahnoberfläche über kompliante, einseitige Kontakte. Der Bobschlitten wird in der Simulation von einem Pilotenmodell gelenkt. Ein Pilotenmodell ist neben dem Bahn- und Schlittenmodell nötig, um gesteuerte Fahrten ohne Benutzerinteraktion (offline) zu simulieren. Die Modelle sind in [10] im Detail vorgestellt und werden gegen eine Messreihe, vgl. [11], validiert.

Wohl der Erste, der sich mit der Modellierung einer Schlittenfahrt im Eiskanal auseinandergesetzt hat, war 1973 Baumann [7]. Seine Arbeit verwendet ein ungesteuertes Punktmassenmodell, das sich entlang von verschiedenen Approximationen der Bahnprofillinie bewegt. Der Fokus richtet sich auf das Sportgerät und nicht auf die Evaluierung einer Bahn. Untersucht wird der Einfluss der Masse, des aerodynamischen Widerstands und des Reibungskoeffizienten auf die Laufzeit und Kurvenausgangsgeschwindigkeiten. Diese Arbeit liegt am Ursprung der Simulationsprojekte, welche sich der Optimierung von Gerät und dem Training der Sportler widmen. Unter den im Folgenden beschriebenen Trainingssimulatoren sind zwei besonders hervorzuheben. An der Universität von Kalifornien in Davis, USA entstand zwischen 1988 und 1998 [82] der erste Bobsimulator, der von Profis in die Vorbereitungen auf olympische Spiele eingebunden wurde. Nahezu parallel dazu entwickelte man an der technischen Universität Chemnitz in Deutschland (ehemals DDR) zwischen 1992 und 2004 ebenfalls einen Simulator zu Trainingszwecken [81]. Das Projekt in Chemnitz dürfte nicht zuletzt von den Entwicklungen in den Vereinigten Staaten inspiriert gewesen sein, denn es „bestand seitens der Sportler, auch unter dem Eindruck eines entsprechenden US-amerikanischen Systems, der Wunsch, ein Werkzeug zur Verfügung zu haben, mit dem es möglich ist, aktiv gesteuerte Fahrten auf kompletten Wettkampfbahnen in Echtzeit zu simulieren“ ([30], S. 240).

Hubbard et al. modellierte 1989 [35, 36] den Bobschlitten als Punktmasse, die bilateral auf die Bahnoberfläche gebunden ist. Die Bobbahn wird als glatte Splineoberfläche modelliert und die Position der Punktmasse in den zwei Oberflächenparametern identifiziert. Ein glattes Modell der Bahnoberfläche ist notwendige Voraussetzung, damit die Bewegungsgleichungen des Schlittens in diesen Oberflächenparametern formuliert werden können. Aufbauend auf dieser Arbeit präsentiert Huffman et al. 1993 [38] die erste Version des amerikanischen Bobsimulators, der zu dieser Zeit noch statisch aufgebaut war. An der Punktmasse greift unter anderen eine Steuerkraft an, die vom Lenkwinkel abhängig ist. Es handelt sich damit um ein steuerbares Bobmodell. Weiter kann die Punktmasse gegen die Seitenwände stossen. In Normalrichtung wird ein Newtonsches Stossgesetz verwendet. Der Tangentialstoss ist eigens modelliert worden und bremst den Schlitten. Das System rechnet die Physik (vermutlich event-driven) mit einem Takt von etwa 100Hz und rendert mit einer Bildrate von 30fps die Position des Schlittens in der Bahn. Als Analyseinstrumente stehen Diagramme der Verläufe beliebiger Zustandsgrössen zur Verfügung. Sie werden mit der besten Fahrt des Piloten bzw. dem Bahnrekord verglichen, um Schwach- oder Fehlerstellen zu identifizieren.

Die zweite Version des Simulators [33, 37] wurde um diverse Wahrnehmungskanäle erweitert. Der Simulator interagiert nicht mehr nur über das visuelle Display mit dem Piloten. Dieser sitzt in einem Aufbau, der die Rollbewegungen des Schlittens (mit *washout*)

nachahmt. An den Steuerseilen ist ein Force-Feedback Mechanismus angebracht, der den Widerstand gegen Lenkbewegungen imitiert. Akustische Signale vermitteln ein Gefühl der Vibrationen im Schlitten. Die zweite Version des Simulators fand in den Vorbereitungen der olympischen Spiele 1994 Verwendung. Die dritte Generation [52] von 2000 überarbeitet die Subsysteme für die Rollbewegung, das Force-Feedback in der Lenkung und verbessert die generelle Handhabung. Der Simulator genügt schliesslich den Sicherheits-, Ergonomie- und Robustheitsansprüchen, die der öffentliche, unbeaufsichtigte Gebrauch stellt. Er wird heute auch als Touristenattraktion verwertet. Es bleibt zu erwähnen, dass sich Zhang et al. 1995 [87] mit der optimalen Steuerung der Punktmasse auseinandergesetzt hat. Der Lenkprozess wird dabei hinsichtlich der Durchfahrzeit und Ausgangsgeschwindigkeit für einzelne Kurven optimiert. Aus Konvergenzgründen können ganze Bahnen nur stück-, also kurvenweise, analysiert werden. Es ist nicht bekannt, ob die Erkenntnisse ins Training mit dem Simulator bzw. die Analyse von Fahrten eingeflossen sind.

Die Entwicklung des Simulators in Chemnitz verläuft in ganz ähnlichen Schritten. Der Simulator bleibt allerdings vollständig nicht-aktuiert. D.h. der Schlitten, in dem der Benutzer sitzt, ist statisch und es gibt kein Force-Feedback auf der Lenkung. Dafür wird in einem Folgeprojekt ein Programm zur Evaluation von Schlittenkonstruktionsparametern aufgebaut. Günther et al. zeigt 1994 in [29] ein eigenes, gesteuertes Punktmassenmodell für den Bobschlitten. Auch hier wird die Bahn als glatte Splineoberfläche approximiert und die Punktmasse bilateral darauf gebunden. Der Lenkwinkel geht in diesem Modell nicht explizit in eine Steuerkraft, sondern in die zeitliche Änderung der Geschwindigkeitsrichtung ein. Dieses sogenannte 2-Kufenmodell unterliegt der strengen Annahme, dass sich die Hinter- und Vorderkufe auf ein und derselben Trajektorie bewegen. Es erlaubt damit kein Driften, vgl. auch [30], S. 241. Der Simulator berücksichtigt zu dieser Zeit noch keine Stösse gegen die seitlichen Banden, erlaubt aber bereits eine echtzeitnahe graphische Animation der Fahrspur (von aussen betrachtet) bei der Simulation einzelner Kurven, vgl. [29]. Auch hier ist die Echtzeitfähigkeit aus heutiger Sicht zu relativieren, wurden doch Bildraten bereits ab zehn Bildern pro Sekunde für flüssig befunden ([30], S. 244).

Gemäss [17] können im Simulator ab 1995 ganze Bahnen gefahren werden und nach einem Hardwarewechsel 1996 gelingt die Simulation inklusive grafischer Darstellung der Pilotensicht in Echtzeit. Nach diversen Überarbeitungen des Steueralgorithmus steht der Simulator ab 1999 offiziell als Trainingsgerät zur Verfügung, vgl. auch [30]. Für die Analyse von Fahrten stehen Diagramme und die Möglichkeit zur Verfügung, Fahrten nachträglich zu visualisieren und von aussen zu betrachten. In der Folge entwickelt man in Chemnitz ab 2001 ein separates Analyseprogramm für Parameterstudien. Mit Hilfe von CAD Daten wird ein parametriertes Mehrkörpermodell des Schlittens in Angriff genommen, das vorerst noch bilateral mit der Bahnoberfläche verbunden ist [60]. 2002 werden unilaterale Kontakte, als geschaltete bilaterale Bindungen, eingeführt, vgl. [61], S. 10 und 2003 das Reibmodell um verschiedene Effekte erweitert, wobei nach [53] ein Quergleiten der Kufen immer noch ausgeschlossen ist. In [59] erwähnt Maisser, dass mit diesem Mehrkörpermodell schliesslich Evaluationen verschiedener Schlittendesigns durchgeführt werden. Zwecks Reproduzierbarkeit der Simulationsresultate wird der Schlitten dabei entlang „einer bestimmten Sollfahrspur geregelt“ (S. 404). Ob in diesem Analyseprogramm die Simulationen

noch in Echtzeit gerechnet werden bzw. gerechnet werden können, ist nirgends ersichtlich. Gemäss [24] werden die Simulationsresultate ständig visuell aufbereitet. Dies geschieht allerdings nur mit dem Zweck, Simulationen jederzeit abbrechen zu können, wenn ein Fortfahren unsinnig erscheint. Die Echtzeitfähigkeit ist aufgrund des implementierten Pilotenmodells (Fahrspurregler) zumindest irrelevant geworden, da damit alles offline und auch verlangsamt simuliert werden kann. Die „bestimmte Sollfahrspur“, welcher der Regler folgt, wird leider nicht näher beschrieben. Die Vermutung liegt allerdings nahe, dass sie auf der Arbeit [58] aus dem Jahre 1998 aufbaut. Darin beschreibt Maisser unter vereinfachenden Annahmen analytisch wegkürzeste und zeitkürzeste Pfade für eine Punktmasse auf einer zweifach gekrümmten Fläche.

An der japanischen Chukyo Universität gibt es einen weiteren Bobsimulator. Es wird vermutet, dass bei diesem Projekt vor allem die Systemplattform im Vordergrund steht. Sie erlaubt die Bewegung des Schlittens mit den Insassen in allen sechs Freiheitsgraden und ist von vier Leinwänden umrundet, womit sich die Benutzer *inmitten* der virtuellen Welt vorfinden. Man vergleiche dazu [13], wo auch Fotos gezeigt sind. Die Veröffentlichungen von Ogino zu diesem Simulator [69, 70, 71] aus den Jahren 2004 bis 2009 sind in japanisch verfasst und damit dem Autor unzugänglich. Den englischen Übersetzungen der Zusammenfassungen wird entnommen, dass ein Bobmodell verwendet wird, das bilateral auf die Bahnoberfläche gebunden ist. Und aufgrund des beschriebenen Problems von zu „kleinen Reaktionskräften“ in der Simulation lassen sie vermuten, dass es sich auch um eine unzureichende Modellierung des Schlittens handeln könnte.

Es scheint auch an der Universität von Calgary in Kanada Bestrebungen für einen Simulator zu geben. In [57] von 2007 meint Levy, vielleicht sollten Simulatoren in Anbetracht der zunehmend schwierigen, neueren Bahnen[1] zum Standardtraining dazugehören. Wegen der fortwährenden Professionalisierung des Sports (und der Jagd nach Rekorden) könnten die wenigen Fahrten während des Winters, und speziell die wenigen Fahrten auf jeder einzelnen Bahn im Weltcup, bald nicht mehr ausreichend sein. Solche Entwicklungen sind in anderen Sportarten, oder beispielsweise auch für Flugzeugpiloten, bereits Tatsache. Mit zunehmender Erfahrung auf jeder einzelnen Bahn steigt auch die Sicherheit und mit der Routine wird in kritischen Situationen besser reagiert. Details zum Simulatorsystem selbst, wie z. B. zu den verwendeten Modellen von Bob und Bahn, sind nicht bekannt.

Arnold [4] veröffentlicht 2013 Resultate einer Punktmassenpfadsimulation. Die gerechneten Lauf- und Zwischenzeiten stimmen gut mit den Rennergebnissen überein, wie es auch bei [7] und [68], die ebenfalls mit eindimensionalen Modellen arbeiten, der Fall ist. Für die Voraussage von Laufzeiten und erreichten Geschwindigkeiten bzw. von den sich daraus ergebenden Fliehkraftbelastungen scheinen eindimensionale Punktmassenmodelle offenbar zu genügen. Dagegen können die Sturzgefahr und Gefahren, die sich aus extremen Fahrpfaden ergeben können, nicht oder nur schlecht beurteilt werden mit

[1] Sowohl 2006 an den olympischen Winterspielen in Turin, wie auch 2010 in Whistler, gab es seitens der Sportler Vorwürfe, die Eiskanäle seien zu gefährlich, vgl. [57]. Zwei von drei Schweizer Piloten starteten in Whistler aufgrund von vorausgehenden Stürzen nicht zu den olympischen Rennen. Im Rodeln kam es gar zu einem tragischen Todesfall. Man vergleiche dazu [34].

solchen Modellen. Sie berücksichtigen in der Regel nur Pfade von fehlerfreien Fahrten bzw. solche, die der Bahntrajektorie entsprechen. Eine nennenswerte Ausnahme bildet [34]. Weiter präsentiert Arnold [4] eine Mehrkörpersimulation für einzelne Kurven. Er beurteilt damit qualitativ den Einfluss von Schlittenparametern auf die Fahrlinien in einer Kurve. In der Simulation wird ein rudimentäres Pilotenmodell verwendet, welches den Lenkverlauf in Abhängigkeit der Zeit von vornherein vorgibt. Zu festgelegten Zeiten werden also, unabhängig der Position und Lage des Schlittens, konstante Lenkwinkel ein- und zurückgestellt. Als Bahnmodell wird ein Dreiecksgitter hergenommen, das mit dem in dieser Arbeit präsentierten Verfahren erstellt und vom Autor zur Verfügung gestellt wurde. Das Modell des Bobschlittens besteht aus zwei starren Körpern, die mit einem Rotationsgelenk verbunden sind. Sie repräsentieren den Vorder- und Hinterteil des in der Mitte getrennten Schlittens. Die Kufen sind durch je zwei Punkte auf dem vorderen bzw. hinteren Starrkörper abgebildet. Mit diesen sind vier einseitige und reibungsbehaftete harte Kontakte zum Bahngitter definiert. Die Aufstandspunkte des Vorderteils sind dabei um eine Lenkachse mit vorgebbarem Lenkwinkel drehbar, diejenigen des Hinterteils auf dem Starrkörper fixiert. Als Reibmodelle werden drei verschiedene, mengenwertige Reibgesetze verwendet. Die Simulation ist grundsätzlich mit denselben mathematischen Methoden aufgesetzt, wie sie auch in dieser Arbeit Anwendung finden und ist als Vorarbeit zum vorliegenden Simulatorprojekt zu bezeichnen.

1.4 Aufbau der Arbeit

Kapitel 2 führt anwendungsorientiert in die theoretischen Grundlagen für die Programmierung einer Mehrkörpersimulation ein. Zu den einzelnen Schritten der Simulation sind Implementierungen in der Programmiersprache *Python* [75] gezeigt. Grundlagenkenntnisse in einer objekt-orientierten Sprache sollen genügen, um mit den Bausteinen einen Simulationscode nachzubauen.

Kapitel 3 erklärt detailliert die Rekonstruktion der Betonoberfläche eines künstlichen Eiskanals. Die von den Konstruktionsdaten spezifizierten Geometrien werden vorgestellt. Die programmatische Generierung der Fahrbahnoberfläche ist schrittweise erläutert. Schliesslich werden Randelemente besprochen, welche den Eiskanal abschliessen. Solche Elemente stellen sicher, dass kein Gerät oder Sportler die Bahn verlassen kann.

Kapitel 4 präsentiert das Mehrkörpermodell des Schlittens. Neben den Freiheitsgraden werden die im Modell wirkenden Kraftelemente besprochen. Insbesondere wird das Reibmodell der Kufenkontakte vorgestellt. Das Kapitel schliesst mit einer Parameterevaluierung. Einige unsichere und schwierig zu messende Parameter werden darin mit Hilfe von professionellen Bobpiloten optimiert.

Kapitel 5 zeigt den physischen Aufbau des Bobsimulators. Es erwähnt Einstellmöglichkeiten der Software, nennt einige Performanzeckwerte der Simulation und präsentiert das haptische Feedbacksystem, das im Vorderboot des Schlittens verbaut ist. Das von den (virtuellen) Kontaktkräften induzierte Moment auf der Lenkachse wird von einem Motor auf die Steuerung im Schlitten gegeben.

Der Simulator wird in Kapitel 6 anhand eines Satzes von Fahrten gegen Rennresultate

validiert. Dazu werden die Zwischen- und Laufzeiten und deren statistische Verteilung verglichen und analysiert.

In Kapitel 7 werden die Fahrlinien von sechs Fahrten verglichen. Es wird analysiert, wie verschiedene charakteristische Fahrlinien gegeneinander abschneiden bezüglich Geschwindigkeit und Zeitverlust. Dabei zeigt sich, dass jede Kurve einer Bahn ihre Eigenheiten haben kann.

Kapitel 8 fasst die Ergebnisse des Bobsimulatorprojektes zusammen.

Kapitel 2

Programmierung einer Mehrkörpersimulation

Dieses Kapitel fasst theoretische Aspekte der nicht glatten Mehrkörperdynamik mit Hinblick auf eine objekt-orientiert implementierte Simulation der Bewegung eines Mehrkörpersystems zusammen. Als *Mehrkörpersystem* wird im Rahmen dieser Arbeit ein mechanisches Modell verstanden, das aus mehreren starren Körpern besteht, die über ideale Gelenke miteinander gekoppelt sind. Auf den Starrkörpern sind einzelne Kontaktpunkte definiert, die gegen eine statische Kollisionsgeometrie reibungsbehaftet stossen und gleiten können. Das Mehrkörpersystem ist in den dreidimensionalen euklidischen Raum eingebettet und insbesondere über ein Ursprungsgelenk mit dessen inertialem Ursprung O verbunden. Die Kollisionsgeometrie wird ebenfalls in diesem Raum platziert.

In der Mehrkörperdynamik werden bilaterale Bindungen auf zwei unterschiedliche Weisen interpretiert, was im Folgenden in einem kurzen Abriss erläutert wird und für das Verständnis des Programmaufbaus nützlich ist. In der klassischen Mechanik wird Materie mit Hilfe von kontinuierlichen Massenverteilungen oder Mengen von sogenannten Partikeln modelliert. Die massenbehafteten Punkte werden in einer Referenzkonfiguration im euklidischen Raum bezeichnet. Mit Hilfe von gewählten (und damit bereits modellierten) Verschiebungsabbildungen wird die Referenzkonfiguration auf eine verschobene, aktuelle Konfiguration abgebildet [64]. Ein Ziel der klassischen Mechanik ist die Beschreibung der fortlaufenden Veränderung der aktuellen Konfiguration unter dem Einfluss von Kräften. Der Ablauf der Konfigurationen wird mit einer einzelnen Grösse parametriert und dieser Parameter wird die *Zeit* genannt. Auch die Verschiebungsabbildungen zu einzelnen Zeitpunkten werden durch eine endliche Anzahl Grössen parametriert, z.B. [32]. Diese Parameter, die die Verschiebungsabbildung zu einem gegebenen Zeitpunkt eindeutig beschreiben, bilden den Lagezustand eines Systems zu einem gegebenen Zeitpunkt und heissen im ungebundenen System Freiheitsgrade. Nachträglich können geometrische bilaterale Bindungen auf das System gegeben werden. Geometrische bilaterale Bindungen sind einschränkende Bedingungen an die Verschiebungsabbildungen und damit an die Parameter, die diese beschreiben. Folglich schränken diese Bindungen die Freiheitsgrade eines Systems ein, z.B. [12]. Dieses Einschränken wird erzwungen durch zu den Bindungen gehörende Bindungskräfte, die den Verlauf der Konfigurationen gerade so beeinflussen, dass die Bindungen erfüllt bleiben [2]. In diesem Sinne sperren geometrische bilaterale

Bindungen manche Freiheitsgrade und lassen andere frei.

Nach dem Einführen von geometrischen bilateralen Bindungen ist man häufig nur an der Entwicklung der freien Freiheitsgrade interessiert, da man sich die gesperrten Freiheitsgrade mit den selber modellierten und hinzugefügten Bindungen ja vorgibt. Für die Beschreibung der Entwicklung der freien Freiheitsgrade wählt man einen neuen, minimalen Parametersatz für die Verschiebungsabbildungen, der nur noch mit den Bindungen verträgliche Abbildungen zulässt. Diese neuen Parameter werden Minimalkoordinaten genannt und bilden die Freiheitsgrade des gebundenen Systems. Man beachte, dass durch die wiederholte Verwendung des Begriffes Freiheitsgrad Verwirrung entstehen kann und immer angegeben oder klar sein sollte, ob man ihn gerade bezüglich dem gebundenen oder ungebundenen System verwendet. Die Entwicklung aller verbliebenen Freiheitsgrade unterliegt dem Einfluss von Kräften. Da mit den Bindungen auch Bindungskräfte eingeführt werden, interessiert speziell der Einfluss der Bindungskräfte auf die Entwicklung der Minimalkoordinaten. Das Prinzip von d'Alembert/Lagrange [25] definiert eine *ideale Bindung* als eine Bindung, deren Bindungskräfte unter mit den Bindungen verträglichen virtuellen Verschiebungen keine virtuelle Arbeit leisten. Daraus folgt das die Bindungskräfte von idealen Bindungen die Entwicklung der Minimalkoordinaten überhaupt nicht beeinflussen und damit in den Bewegungsgleichungen in Minimalkoordinaten gar nicht erscheinen bzw. nicht berücksichtigt werden müssen.[2]

Bilaterale Bindungen werden einerseits mit Fokus auf die damit eingeführten Bindungskräfte interpretiert. In dieser Sichtweise wird vor allem der sperrende Charakter von Bindungen hervorgehoben. Zielt man andererseits direkt auf eine Beschreibung in Minimalkoordinaten, also ausschliesslich in freien Freiheitsgraden ab und tut dies im Bewusstsein, dass Bindungskräfte in diesen freien Richtungen keinen Einfluss haben bzw. nicht in die Bewegungsgleichungen in Minimalkoordinaten eingehen, dann generieren Bindungen eher die (freien) Freiheitsgrade, als dass sie welche sperren. In dieser zweiten Sichtweise werden Bindungen häufig Gelenke genannt und sie ist gerade für Mehrkörpersysteme, wie sie in diesem Kapitel behandelt werden, nützlich.

In dieser Betrachtungsweise werden die Bindungen häufig Gelenke genannt, die die einzelnen Körper koppeln, und jedes Gelenk trägt Freiheitsgrade zum System bei. Die Bindungen generieren also die Freiheitsgrade. Diese Interpretation rückt die freien Richtungen von Bindungen in den Vordergrund. Für die objekt-orientierte Implementierung einer Mehrkörpersimulation, wie sie in diesem Kapitel skizziert wird, zeigt sich die zweite Interpretation als nützlich.

Es folgt ein Abschnitt der die elementaren Objekte der Simulationsimplementierung mit ihren Datenstrukturen und Methoden zusammenfasst. Es soll darin veranschaulicht werden,

[2]Man beachte, dass deshalb auch die anfängliche Wahl von Verschiebungsabbildungen unproblematisch ist. Denn diese Wahl von Abbildungen, die über eine endliche Anzahl Parameter beschrieben sind, induziert auch Bindungen zwischen den unendlich vielen Partikeln der Referenzkonfiguration. Im Allgemeinen bräuchte es nämlich unendlich viele Abbildungen, also für jeden Partikel eine, von denen jede über einen oder mehrere Parameter beschrieben wäre. Die mit der Wahl von endlich dimensional beschreibbaren Verschiebungsabbildungen eingeführten Bindungskräfte kommen aber in den Bewegungsgleichungen der (freien) Freiheitsgrade nicht vor und sind deshalb irrelevant.

welche Informationen von den einzelnen Objekten verwaltet und gespeichert werden. Im Abschnitt 2.2 wird die Baumstruktur von Mehrkörpersystemen erläutert. Sie ist elementar für die programmatische Umsetzung der Simulation. In den Abschnitten 2.3 bis 2.5 werden die Bewegungsgleichungen des glatten Mehrkörpersystems, d.h. noch ohne Berücksichtigung der Kontakte, schrittweise numerisch aufgestellt. Dabei wird die Massenmatrix $\mathbf{M}$ und der generalisierte Kraftvektor $\boldsymbol{h}$ assembliert, vgl. Gleichung (2.19). Die Kontakte werden nachträglich in Abschnitt 2.6 in die Bewegungsgleichung eingebunden. Daraus folgt die Assemblierung der generalisierten Kraftrichtungen $\boldsymbol{w}$ und der Relativgeschwindigkeitsanteile χ, vgl. Gleichung (2.26). Der Abschnitt 2.7 geht schliesslich auf die Lösung der diskretisierten Bewegungsgleichung ein. Es resultiert ein Inklusionsproblem, das die Kontaktkräfte zusammen mit der Bewegungsgleichung beschreibt. Dieses muss iterativ für die Kontaktkräfte und den Bewegungszustand gelöst werden. Damit ist schliesslich ein Zeitschritt der Simulation vollständig abgeschlossen ist und es kann wieder mit dem Aufstellen der Bewegungsgleichung begonnen werden kann.

2.1 Grundelemente der Implementierung

In diesem Abschnitt geht es darum einen Überblick über die elementaren Objekte einer Simulation zu schaffen und Notationen einzuführen. Die Ausführungen sind bewusst knapp gehalten. Die detaillierte Handhabung der Objekte und die einzelnen Schritte einer Mehrkörpersimulation sind in den restlichen Abschnitten dieses Kapitels beschrieben.

2.1.1 Systemzustände

Um ein Mehrkörpersystem vollständig beschreiben zu können sind mehrere Teilzustände notwendig. Die Dynamik, und damit deren Simulation beschreiben die zeitliche Entwicklung der Bewegung von Körpern unter dem Einfluss von Kräften. Die Zeit t ist deshalb der erste Teilzustand und parametrisiert alle weiteren Teilzustände, die für die Beschreibung der Bewegung der Körper nötig sind.

Die Gelenke zwischen den einzelnen Starrkörpern tragen Lagezustände zum Mehrkörpersystem bei, wobei jedes Gelenk genau zwei Körper koppelt. Jeder Gelenktyp beschreibt anhand eines lokalen Parametersatzes die Lage, Geschwindigkeit und Beschleunigung des zweiten Körpers relativ zur Lage, Geschwindigkeit und Beschleunigung des ersten. Es wird unterschieden zwischen kinematisch aktuierten und freien, d.h. nicht aktuierten Gelenken. Die Parametersätze der kinematisch aktuierten Gelenke sind als Systemeingangsgrössen zu betrachten. Sie sind zu jedem Zeitpunkt t bekannt[3] und unterliegen nicht dem Einfluss von Kräften und Massenträgheiten. Zum Beispiel kann der Winkel eines aktuierten Drehgelenks von einem Sensor gelesen werden und der Simulation als bekannte Grösse zur Verfügung gestellt werden. Die Gesamtheit der Parameter aller kinematisch aktuierten Gelenke in einem Mehrkörpersystem bilden generalisierte Lagen $\boldsymbol{k}$ und Geschwindigkeiten $\boldsymbol{m}$. Sie

[3]Man beachte, dass dies im Allgemeinen nicht heisst, dass die Eingangsgrössen schon a priori für alle Zeiten t bekannt sind, sondern, dass sie zu jedem einzelnen Zeitpunkt t eingelesen beziehungsweise ausgewertet werden können.

Tabelle 2.1: Die Teilzustände eines Mehrkörpersystems.

Gelenke	Koordinaten	Geschwindigkeiten	Reaktionskräfte
frei	$\boldsymbol{q}(t)$	$\boldsymbol{u}(t)$	$\boldsymbol{j}(t)$
kinematisch aktuiert	$\boldsymbol{k}(t)$	$\boldsymbol{m}(t)$	$\boldsymbol{i}(t)$

werden im Tupel $(\boldsymbol{k}, \boldsymbol{m})$ als *Systemeingangsgrössen* zusammengefasst und bilden den zweiten Teilzustand des Mehrkörpersystems. Die Parameter aller freien, nicht aktuierten Gelenke werden in den generalisierten Koordinaten $\boldsymbol{q}$ und Geschwindigkeiten $\boldsymbol{u}$ zusammengefasst. Das Tupel $(\boldsymbol{q}, \boldsymbol{u})$ bezeichnet den *dynamischen Zustand* des Systems und bildet den dritten Teilzustand des Systems. Die zeitliche Entwicklung von $(\boldsymbol{q}, \boldsymbol{u})$ wird von den Kräften und Massenträgheiten beeinflusst und wird, zusammen mit einer Kinematikgleichung, beispielsweise durch die projizierten Newton-Euler Gleichungen beschrieben, welche im Abschnitt 2.5 behandelt werden.

Der Bewegungszustand eines Mehrkörpersystems ist eindeutig parametrisiert über die Eingangsgrössen $(\boldsymbol{k}, \boldsymbol{m})$ und die generalisierten Lagen und Geschwindigkeiten $(\boldsymbol{q}, \boldsymbol{u})$. Man beachte, dass in der Literatur die Abhängigkeit einer Funktion f von den Eingangsgrössen $f(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})$ üblicherweise als explizite Zeitabhängigkeit $f(\boldsymbol{q}, \boldsymbol{u}, t)$ notiert ist. Für die Implementierung einer Mehrkörpersimulation ist es allerdings nützlich die explizit zeitabhängigen Eingangsgrössen $(\boldsymbol{k}, \boldsymbol{m})$ als Teilzustand zu führen und den davon abhängigen Methoden als Argument zu übergeben, wie es die Notation $f(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})$ impliziert.

Die Tupel $(\boldsymbol{q}, \boldsymbol{k})$ und $(\boldsymbol{u}, \boldsymbol{m})$ werden auch als *Lage-* bzw. *Geschwindigkeitszustand* des Systems bezeichnet. Ist die Änderung $(\dot{\boldsymbol{q}}, \dot{\boldsymbol{k}}, \dot{\boldsymbol{u}}, \dot{\boldsymbol{m}})$ eines *Bewegungszustands* $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})$ für eine Zeit t bestimmt, sind damit auch die Kontaktkräfte bekannt. Diese wirken in den jeweiligen Kontaktpunkten und es kann von Interesse sein, welche generalisierten Kräfte von diesen in die Gelenke induziert werden. Die in aktuierte Gelenke induzierten Kräfte werden mit $\boldsymbol{i}$, jene in nicht aktuierte Gelenke induzierten mit $\boldsymbol{j}$ bezeichnet. Tabelle 2.1 fasst die Notation der Teilzustände eines Mehrkörpersystems zusammen.

2.1.2 Starrkörper

Ein Starrkörperobjekt besitzt einen Schwerpunkt S, sowie ein körperfestes Koordinatensystem $(\boldsymbol{e}_x^{\mathrm{K}}, \boldsymbol{e}_y^{\mathrm{K}}, \boldsymbol{e}_z^{\mathrm{K}})$ mit Ursprung G. Die Lage des Schwerpunkts relativ zum Koordinatenursprung ist in körperfesten Koordinaten konstant und mit ${}_{\mathrm{K}}\boldsymbol{r}_{GS}$ gegeben. Der Starrkörper hat eine konstante Masse m und einen Trägheitstensor ${}_{\mathrm{K}}\boldsymbol{\Theta}_S$ gegenüber seinem Schwerpunkt S, welcher dargestellt in körperfesten Koordinaten ebenfalls konstant ist. Weiter führt jeder Starrkörper eine Liste $[\mathtt{j}_0, \mathtt{j}_1, \dots]$ der wegführenden Gelenke, d.h. eine Liste von Objektreferenzen zu all jenen Gelenken für die der Starrkörper erster Bezugskörper ist. Man vergleiche dazu den folgenden Unterabschnitt 2.1.3, der die Gelenkobjekte einführt.

Zu einem momentanen Zeitpunkt t speichert ein Starrkörperobjekt die Position seines Schwerpunkts ${}_{\mathrm{I}}\boldsymbol{r}_{OS}$ in inertialen Koordinaten, sowie die aktuelle Orientierung des Starrkörpers gegenüber dem Inertialsystem $(\boldsymbol{e}_x^{\mathrm{I}}, \boldsymbol{e}_y^{\mathrm{I}}, \boldsymbol{e}_z^{\mathrm{I}})$ mittels der Koordinatentransformationsmatrix $\mathbf{A}_{\mathrm{IK}} = \left({}_{\mathrm{I}}\boldsymbol{e}_x^{\mathrm{K}}, {}_{\mathrm{I}}\boldsymbol{e}_y^{\mathrm{K}}, {}_{\mathrm{I}}\boldsymbol{e}_z^{\mathrm{K}}\right)$. Weiter ist die momentane Translationsgeschwindigkeit

des Schwerpunkts ${}_{\mathrm{I}}\boldsymbol{v}_S$ und die momentane Rotationsgeschwindigkeit ${}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}$ gespeichert. Die Bestimmung dieser Grössen aus einem Systemzustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})(t)$ ist in Abschnitt 2.3 gezeigt. Weiter werden zu einem momentanen Zeitpunkt t die Pseudogeschwindigkeiten ${}_{\mathrm{I}}\boldsymbol{\nu}_S$ und ${}_{\mathrm{K}}\boldsymbol{\nu}_R$, die sich für einen hypothetischen Zustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u} = \mathbf{0}, \boldsymbol{m})(t)$ mit in den freien Gelenken verschwindenden Geschwindigkeiten einstellen, sowie die Pseudobeschleunigungen ${}_{\mathrm{I}}\boldsymbol{\kappa}_S$ und ${}_{\mathrm{K}}\boldsymbol{\kappa}_R$, die sich für den aktuellen Zustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})(t)$ und in den freien Gelenken hypothetisch verschwindenden generalisierten Beschleunigungen $\dot{\boldsymbol{u}} = \mathbf{0}$, sowie den aktuellen Beschleunigungen $\dot{\boldsymbol{m}}(t)$ in den aktuierten Gelenken ergeben. Schliesslich werden die Jacobimatrizen ${}_{\mathrm{I}}\mathbf{J}_S, {}_{\mathrm{K}}\mathbf{J}_R$ bezüglich der freien Gelenke und die Jacobimatrizen ${}_{\mathrm{I}}\mathbf{H}_S, {}_{\mathrm{K}}\mathbf{H}_R$ bezüglich der aktuierten Gelenke für einen momentanen Zeitpunkt t abgelegt. Für eine formelle Einführung dieser Grössen sei auf den Abschnitt 2.3 dieses Kapitels verwiesen.

Jeder Starrkörper führt weiter eine Liste $[\mathtt{f}_0, \mathtt{f}_1, \dots]$ mit Objektreferenzen auf äussere Kräfte und Momente, die am Starrkörper angreifen und keine Kontakt- bzw. Reibkräfte sind. Diese hängen im Allgemeinen von Eigenschaften des Starrkörpers und seinem momentanen Zustand, wie z.B. der momentanen Lage und Geschwindigkeit ab. Als Beispiele seien die Gravitationskraft oder ein aerodynamischer Widerstand genannt. Die Gesamtheit dieser äusseren Kräfte und äusseren freien Momente wird in einer resultierenden Kraft ${}_{\mathrm{I}}\boldsymbol{F}_S$, die im Schwerpunkt S angreift, und einem bezüglich dem Schwerpunkt resultierenden Moment ${}_{\mathrm{K}}\boldsymbol{M}_S$ zusammengefasst. Für einen momentanen Zeitpunkt t werden diese Resultierenden im Starrkörperobjekt abgelegt. Man vergleiche dazu auch den Unterabschnitt 2.1.5 in diesem Kapitel.

Schliesslich führt jeder Starrkörper eine Liste $[\mathtt{c}_0, \mathtt{c}_1, \dots]$ von Kontaktobjektreferenzen, die in Unterabschnitt 2.1.4 beschrieben sind. Für jeden geschlossenen Kontakt ergeben sich Kontaktkräfte λ_n in Normal- und $\boldsymbol{\lambda}_f$ in Tangentialrichtung, die im Kontaktpunkt am Starrkörper angreifen. Das Total aller Kontaktkräfte aller Kontakte an einem Körper zu einem momentanen Zeitpunkt t wird in einer resultierenden Kraft ${}_{\mathrm{I}}\boldsymbol{F}_{\boldsymbol{\lambda},S}$, die im Schwerpunkt S angreift, und dem bezüglich dem Schwerpunkt S resultierenden Moment ${}_{\mathrm{K}}\boldsymbol{M}_{\boldsymbol{\lambda},S}$ ebenfalls im Starrkörperobjekt gespeichert.

Tabelle 2.2 fasst die Datenstruktur von Starrkörpern zusammen. Man beachte, dass der inertiale Ursprung O in jedem Mehrkörpersystem einen speziellen starren Körper repräsentiert. Auf ihm sind keine Kontakte definiert, es greifen keine äusseren Kräfte und Momente an und die Masse, Trägheit, Geschwindigkeit und Beschleunigung verschwinden. Sein Ortsvektor ${}_{\mathrm{I}}\boldsymbol{r}_{OS} = \mathbf{0}$ und die Orientierung $\mathbf{A}_{\mathrm{IK}} = \mathbf{I}$ sind für alle Zeiten konstant.

2.1.3 Gelenke

Die Tabelle 2.3 listet die Eigenschaften von Gelenkobjekten zusammenfassend auf. Jedes Gelenk des Mehrkörpersystems besitzt eine geordnete Liste $[\mathtt{b}_0, \mathtt{b}_1]$ mit Objektreferenzen zu denjenigen zwei Starrkörpern, die das Gelenk verbindet. Der Anknüpfungspunkt M auf dem ersten Körper $\mathtt{b}_0$ wird mittels den Koordinaten ${}_{\mathrm{K}_0}\boldsymbol{r}_{G_0M}$ bezeichnet. Analog befindet sich der Anknüpfungspunkt N auf dem zweiten Körper $\mathtt{b}_1$ an der Stelle ${}_{\mathrm{K}_1}\boldsymbol{r}_{G_1N}$. Die Indizes 0 und 1 referenzieren dabei immer den ersten bzw. den zweiten Starrkörper der Liste. Im Gelenkobjekt werden anhand dieser Angaben die Anknüpfungspunkte des Gelenks relativ

Tabelle 2.2: Die Datenstruktur von Starrkörperobjekten.

Beschreibung	Bezeichnung	Codename
Lage des Ursprungs G	${}_{\mathrm{K}}\boldsymbol{r}_{GS}$	`centreOfMass`
Trägheiten	m, ${}_{\mathrm{K}}\boldsymbol{\Theta}_S$	`mass`, `inertia`
Liste der wegführenden Gelenke	$[\mathtt{j}_0, \mathtt{j}_1, \dots]$	`joints`
Lage und Orientierung	${}_{\mathrm{I}}\boldsymbol{r}_{OS}$, $\mathbf{A}_{\mathrm{IK}}$	`position`, `orientation`
Geschwindigkeiten	${}_{\mathrm{I}}\boldsymbol{v}_S$, ${}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}$	`linearVelocity`, `angularVelocity`
Pseudogeschwindigkeiten	${}_{\mathrm{I}}\boldsymbol{\nu}_S$, ${}_{\mathrm{K}}\boldsymbol{\nu}_R$	`linearPseudoVelocity`, `angularPseudoVelocity`
Pseudobeschleunigungen	${}_{\mathrm{I}}\boldsymbol{\kappa}_S$, ${}_{\mathrm{K}}\boldsymbol{\kappa}_R$	`linearPseudoAcceleration`, `angularPseudoAcceleration`
Jacobimatrizen	${}_{\mathrm{I}}\mathbf{J}_S$, ${}_{\mathrm{K}}\mathbf{J}_R$	`translationJacobi`, `rotationJacobi`
Jacobimatrizen der Aktuierungen	${}_{\mathrm{I}}\mathbf{H}_S$, ${}_{\mathrm{K}}\mathbf{H}_R$	`translationControlJacobi`, `rotationControlJacobi`
Liste mit äusseren Kräften und Momenten	$[\mathtt{f}_0, \mathtt{f}_1, \dots]$	`forces`
resultierende äussere Kräfte und Momente (ohne Kontaktkräfte)	${}_{\mathrm{I}}\boldsymbol{F}_S$, ${}_{\mathrm{K}}\boldsymbol{M}_S$	`externalForce`, `externalMoment`
Liste mit Kontakten	$[\mathtt{c}_0, \mathtt{c}_1, \dots]$	`contacts`
durch Kontakte induzierte Kräfte und Momente	${}_{\mathrm{I}}\boldsymbol{F}_{\boldsymbol{\lambda},S}$, ${}_{\mathrm{K}}\boldsymbol{M}_{\boldsymbol{\lambda},S}$	`contactForce`, `contactMoment`

Tabelle 2.3: Die Datenstruktur von Gelenkobjekten.

Beschreibung	Bezeichnung	Codename
Liste der verbundenen Starrkörper	$[\mathtt{b}_0, \mathtt{b}_1]$	`parent`, `child`
Anknüpfungspunkte auf den Starrkörpern	${}_{\mathrm{K}_0}\boldsymbol{r}_{S_0M}$, ${}_{\mathrm{K}_1}\boldsymbol{r}_{S_1N}$	`parentLocation`, `childLocation`
Anzahl generalisierter Zustandslagen und -geschwindigkeiten	n_r, n_v	`coordinatesCount`, `velocitiesCount`
Indizes der Gelenkzustände im globalen Zustand	i_r, i_v	`coordinatesOffset`, `velocitiesOffset`
lokaler Zustand	$\boldsymbol{l}$, $\boldsymbol{c}$, $\dot{\boldsymbol{c}}$	`internalCoordinates`, `internalVelocities`, `internalAccelerations`
kinematische Aktuierung	`false` oder `true`	`isKinematicallyActuated`
lokaler Zustand aus globalem Zustand	$\boldsymbol{l}(\boldsymbol{q}, \boldsymbol{k})$, $\boldsymbol{c}(\boldsymbol{u}, \boldsymbol{m})$, $\dot{\boldsymbol{c}}(\dot{\boldsymbol{u}}, \dot{\boldsymbol{m}})$	`updateLocalState`
lokaler Anfangszustand	$\boldsymbol{l}_{\mathrm{init}}$, $\boldsymbol{c}_{\mathrm{init}}$	`initialCoordinates`, `initialVelocities`
lokales Kraftelement	${}_{\mathrm{I}}\boldsymbol{F}_M(\boldsymbol{l}, \boldsymbol{c})$, ${}_{\mathrm{K}_0}\boldsymbol{M}_0(\boldsymbol{l}, \boldsymbol{c})$, ${}_{\mathrm{I}}\boldsymbol{F}_N(\boldsymbol{l}, \boldsymbol{c})$, ${}_{\mathrm{K}_1}\boldsymbol{M}_1(\boldsymbol{l}, \boldsymbol{c})$	`JointForceAndMoment`
Kinematikübertragungen	$\mathbf{A}_{\mathrm{IK}_1}(\mathbf{A}_{\mathrm{IK}_0}, \boldsymbol{l})$, ${}_{\mathrm{I}}\boldsymbol{r}_{OS_1}({}_{\mathrm{I}}\boldsymbol{r}_{OS_0}, \mathbf{A}_{\mathrm{IK}_0}, \boldsymbol{l})$	`propagateDisplacement`
	${}_{\mathrm{K}_1}\boldsymbol{\omega}_{\mathrm{IK}_1}({}_{\mathrm{K}_0}\boldsymbol{\omega}_{\mathrm{IK}_0}, \boldsymbol{l}, \boldsymbol{c})$, ${}_{\mathrm{I}}\boldsymbol{v}_{S_1}({}_{\mathrm{I}}\boldsymbol{v}_{S_0}, {}_{\mathrm{K}_0}\boldsymbol{\omega}_{\mathrm{IK}_0}, \boldsymbol{l}, \boldsymbol{c})$	`propagateVelocity`
	${}_{\mathrm{K}_1}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_1}({}_{\mathrm{K}_0}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_0}, \boldsymbol{l}, \boldsymbol{c}, \dot{\boldsymbol{c}})$, ${}_{\mathrm{I}}\dot{\boldsymbol{v}}_{S_1}({}_{\mathrm{I}}\dot{\boldsymbol{v}}_{S_0}, {}_{\mathrm{K}_0}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_0}, \boldsymbol{l}, \boldsymbol{c}, \dot{\boldsymbol{c}})$	`propagateAcceleration`
Kinematikgleichung	$\dot{\boldsymbol{l}} = \mathbf{F}(\boldsymbol{l})\boldsymbol{c}$	`getCoordinatesDerivative`

zu den Schwerpunkten der entsprechenden Körper, also

$$ {}_{\mathrm{K}_0}\boldsymbol{r}_{S_0M} = {}_{\mathrm{K}_0}\boldsymbol{r}_{G_0M} - {}_{\mathrm{K}}\boldsymbol{r}_{GS}[\mathtt{b}_0] \qquad \text{und} \qquad {}_{\mathrm{K}_1}\boldsymbol{r}_{S_1N} = {}_{\mathrm{K}_1}\boldsymbol{r}_{G_1N} - {}_{\mathrm{K}}\boldsymbol{r}_{GS}[\mathtt{b}_1] $$

gespeichert. Hierin bezeichnet die Notation ${}_{\mathrm{K}}\boldsymbol{r}_{GS}[\mathtt{b}_0]$, dass der Vektor ${}_{\mathrm{K}}\boldsymbol{r}_{GS}$ vom Starrkörperobjekt $\mathtt{b}_0$ ausgelesen wird, vgl. Unterabschnitt 2.1.2. Man beachte, dass sich damit die Punkte G und S auf Starrkörperpunkte des Körpers $\mathtt{b}_0$ und sich das Koordinatensystem K auf die körperfesten Koordinaten des Körpers $\mathtt{b}_0$ beziehen, was mit Indizes ausgeschrieben dem Vektor ${}_{\mathrm{K}_0}\boldsymbol{r}_{G_0S_0}$ entspricht.

Jeder Gelenktyp definiert die Anzahl n_r generalisierter Lagen und die Anzahl n_v generalisierter Geschwindigkeiten, die ein Gelenk von diesem Typ zum Mehrkörpersystemzustand beiträgt. Mit den generalisierten Lagen $\boldsymbol{l} \in \mathbb{R}^{n_r}$ und Geschwindigkeiten $\boldsymbol{c} \in \mathbb{R}^{n_v}$ wird der interne Zustand des Gelenks bezeichnet. Dabei kann dieser, und damit das Gelenk, aktuiert oder frei sein, was in der Variable `isKinematicallyActuated` gekennzeichnet wird. Die internen Zustände aller aktuierten Gelenke werden vom Treiber, siehe Unterabschnitt 2.1.6, als (globale) Systemeingangsgrössen $(\boldsymbol{k}, \boldsymbol{m})$ und diejenigen aller freien Gelenke als (globaler) dynamischer Systemzustand $(\boldsymbol{q}, \boldsymbol{u})$ assembliert. Die Indizes i_r und i_v, die in den einzelnen Gelenken gespeichert werden, bezeichnen jeweils den ersten Eintrag der n_r bzw. n_v Gelenkzustände in dem dazugehörigen assemblierten (globalen) Zustand $(\boldsymbol{k}, \boldsymbol{m})$ bzw. $(\boldsymbol{q}, \boldsymbol{u})$. Der lokale Zustand kann also mit

$$ \boldsymbol{l}(\boldsymbol{q}, \boldsymbol{k}) = \begin{cases} (k_{i_r}, \ldots, k_{i_r+n_r-1})^\mathsf{T} \\ (q_{i_r}, \ldots, q_{i_r+n_r-1})^\mathsf{T} \end{cases}, \quad \boldsymbol{c}(\boldsymbol{u}, \boldsymbol{m}) = \begin{cases} (m_{i_v}, \ldots, m_{i_v+n_v-1})^\mathsf{T} & : \text{aktuiert} \\ (u_{i_v}, \ldots, u_{i_v+n_v-1})^\mathsf{T} & : \text{frei} \end{cases} $$

aus dem globalen Zustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})$ gelesen werden, was die Methode `updateLocalState` implementiert. Jedes Gelenk speichert weiter den initialen lokalen Zustand $(\boldsymbol{l}_{\text{init}}, \boldsymbol{c}_{\text{init}})$, woraus sich ein Anfangszustand $(\boldsymbol{q}_{\text{init}}, \boldsymbol{u}_{\text{init}})$ assemblieren lässt. Man beachte, dass der Initialzustand $(\boldsymbol{l}_{\text{init}}, \boldsymbol{c}_{\text{init}})$ prinzipiell nur als Anfangsbedingung für den dynamischen Zustand $(\boldsymbol{q}, \boldsymbol{u})$ gebraucht wird. Die Systemeingangsgrössen $(\boldsymbol{k}, \boldsymbol{m})$ werden bei Bedarf immer direkt von den Eingängen gelesen und brauchen damit nicht initialisiert zu werden.

Ein nicht aktuiertes Gelenk kann ein Kraftelement enthalten, das abhängig vom lokalen Gelenkzustand $(\boldsymbol{l}, \boldsymbol{c})$ auf die beiden mit dem Gelenk verbundenen Starrkörpern die gegengleichen Kräfte ${}_{\mathrm{I}}\boldsymbol{F}_M$ bzw. ${}_{\mathrm{I}}\boldsymbol{F}_N$ in den Anknüpfungspunkten M bzw. N und die gegengleichen Momente ${}_{\mathrm{K}_0}\boldsymbol{M}_0$ bzw. ${}_{\mathrm{K}_1}\boldsymbol{M}_1$ ausübt. Diese Kräfte und Momente tragen zu den äusseren Kräften ${}_{\mathrm{I}}\boldsymbol{F}_S[\mathtt{b}_i]$ und Momenten ${}_{\mathrm{K}}\boldsymbol{M}_S[\mathtt{b}_i]$ der verbundenen Starrkörper $\mathtt{b}_i$ bei.

Jeder Gelenktyp spezifiziert weiter drei kinematische Übertragungsfunktionen, welche die Orientierung $\mathbf{A}_{\mathrm{IK}_1}$ und die Schwerpunktsposition ${}_{\mathrm{I}}\boldsymbol{r}_{OS_1}$, die Geschwindigkeiten ${}_{\mathrm{K}_1}\boldsymbol{\omega}_{\mathrm{IK}_1}$, ${}_{\mathrm{I}}\boldsymbol{v}_{S_1}$ und Beschleunigungen ${}_{\mathrm{K}_1}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_1}$, ${}_{\mathrm{I}}\dot{\boldsymbol{v}}_{S_1}$ des zweiten Körpers $\mathtt{b}_1$ in Abhängigkeit des lokalen Zustandes und derselben Grössen des vorhergehenden ersten Körpers $\mathtt{b}_0$ bestimmen. Es folgt eine allgemeine Herleitung dieser Übertragungen, wobei darin die Teilübertragungen $\mathbf{A}_{\mathrm{K}_0\mathrm{K}_1}$, ${}_{\mathrm{K}_0}\boldsymbol{r}_{MN}$, ${}_{\mathrm{K}_0}\boldsymbol{\omega}_{\mathrm{K}_0\mathrm{K}_1}$, ${}_{\mathrm{K}_0}\dot{\boldsymbol{r}}_{MN}$, ${}_{\mathrm{K}_0}\dot{\boldsymbol{\omega}}_{\mathrm{K}_0\mathrm{K}_1}$, ${}_{\mathrm{K}_0}\ddot{\boldsymbol{r}}_{MN}$ zwischen den Anknüpfungspunkten M und N bzw. den beiden körperfesten Koordinatensystemen K_0 und K_1 für jeden Gelenktyp spezifisch sind und vom lokalen Zustand $(\boldsymbol{l}, \boldsymbol{c})$ und dessen Änderung $\dot{\boldsymbol{c}}$ abhängen. Die Teilübertragungen müssen für jeden Gelenktyp hergeleitet und implementiert werden.

Für die Beschreibung der Orientierung $\mathbf{A}_{\mathrm{IK}_1}$ eines Körpers $\mathtt{b}_1$ nutzt man die Eigenschaft, dass sich die Koordinatentransformation $\mathbf{A}_{\mathrm{IK}_1}$ von einem Koordinatensystem K_1 in das Inertialsystem I kaskadieren lässt in zwei Transformationen $\mathbf{A}_{\mathrm{K}_0\mathrm{K}_1}$ und $\mathbf{A}_{\mathrm{IK}_0}$, deren Hintereinanderschaltung über ein Zwischensystem K_0 geht. Die Lage des Schwerpunkts $\boldsymbol{r}_{OS_1}$ des auf das Gelenk folgenden Körpers relativ zur Lage $\boldsymbol{r}_{OS_0}$ lässt sich über eine Vektorkette von S_0 über M und N nach S_1 konstruieren. Die Orientierung und Lage ergeben sich deswegen nach

$$\begin{aligned} \mathbf{A}_{\mathrm{IK}_1} &= \mathbf{A}_{\mathrm{IK}_0}\mathbf{A}_{\mathrm{K}_0\mathrm{K}_1}(\boldsymbol{l}) \\ {}_{\mathrm{I}}\boldsymbol{r}_{OS_1} &= {}_{\mathrm{I}}\boldsymbol{r}_{OS_0} + \mathbf{A}_{\mathrm{IK}_0}\left({}_{\mathrm{K}_0}\boldsymbol{r}_{S_0M} + {}_{\mathrm{K}_0}\boldsymbol{r}_{MN}(\boldsymbol{l})\right) \\ &\quad + \mathbf{A}_{\mathrm{IK}_1}\left(-{}_{\mathrm{K}_1}\boldsymbol{r}_{S_1N}\right). \end{aligned} \tag{2.1}$$

Für die Herleitung der anderen Kinematikübertragungen benötigt man die Additivität von Relativwinkelgeschwindigkeiten

$${}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IC}} = {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} + {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{BC}}, \tag{2.2}$$

wobei B und C beliebige Koordinatensysteme bezeichnen, sowie die Euler'sche Differentationsregel

$${}_{\mathrm{B}}(\dot{\boldsymbol{c}}) = {}_{\mathrm{B}}\dot{\boldsymbol{c}} + {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times {}_{\mathrm{B}}\boldsymbol{c}, \tag{2.3}$$

nach der ein zeitlich abzuleitender Vektor $\boldsymbol{c}$ in einem bewegten Koordinatensystem B dargestellt wird. Insbesondere gilt

$${}_{\mathrm{K}}(\dot{\boldsymbol{\omega}}_{\mathrm{IK}}) = {}_{\mathrm{K}}\dot{\boldsymbol{\omega}}_{\mathrm{IK}}, \qquad {}_{\mathrm{I}}(\dot{\boldsymbol{c}}) = {}_{\mathrm{I}}\dot{\boldsymbol{c}} \tag{2.4}$$

weil zum Ersten ${}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}} \times {}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}} = \mathbf{0}$ und zum Zweiten ${}_{\mathrm{I}}\boldsymbol{\omega}_{\mathrm{II}} = \mathbf{0}$. Die Relativgeschwindigkeit zwischen zwei Punkten P und Q berechnet sich mit (2.3) nach

$$\begin{aligned} \boldsymbol{v}_Q - \boldsymbol{v}_P &:= \dot{\boldsymbol{r}}_{OQ} - \dot{\boldsymbol{r}}_{OP} \\ &= (\boldsymbol{r}_{OQ} - \boldsymbol{r}_{OP})^{\cdot} \\ &= \dot{\boldsymbol{r}}_{PQ} \\ \Leftrightarrow \quad {}_{\mathrm{B}}\boldsymbol{v}_Q - {}_{\mathrm{B}}\boldsymbol{v}_P &= {}_{\mathrm{B}}(\dot{\boldsymbol{r}}_{PQ}) \\ &= {}_{\mathrm{B}}\dot{\boldsymbol{r}}_{PQ} + {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times {}_{\mathrm{B}}\boldsymbol{r}_{PQ} \\ \Leftrightarrow \quad {}_{\mathrm{B}}\boldsymbol{v}_Q &= {}_{\mathrm{B}}\boldsymbol{v}_P + {}_{\mathrm{B}}\dot{\boldsymbol{r}}_{PQ} + {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times {}_{\mathrm{B}}\boldsymbol{r}_{PQ} \end{aligned} \tag{2.5}$$

und die Relativbeschleunigung zwischen zwei Punkten P und Q folglich nach

$$\begin{aligned} {}_{\mathrm{B}}((\boldsymbol{v}_Q - \boldsymbol{v}_P)^{\cdot}) = {}_{\mathrm{B}}\left((\dot{\boldsymbol{r}}_{PQ})^{\cdot}\right) &= {}_{\mathrm{B}}(\dot{\boldsymbol{v}}_Q - \dot{\boldsymbol{v}}_P) \\ &= ({}_{\mathrm{B}}(\dot{\boldsymbol{r}}_{PQ}))^{\cdot} + {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times {}_{\mathrm{B}}(\dot{\boldsymbol{r}}_{PQ}) \\ &= ({}_{\mathrm{B}}\dot{\boldsymbol{r}}_{PQ} + {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times {}_{\mathrm{B}}\boldsymbol{r}_{PQ})^{\cdot} \\ &\quad + {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times ({}_{\mathrm{B}}\dot{\boldsymbol{r}}_{PQ} + {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times {}_{\mathrm{B}}\boldsymbol{r}_{PQ}) \\ &= {}_{\mathrm{B}}\ddot{\boldsymbol{r}}_{PQ} + {}_{\mathrm{B}}\dot{\boldsymbol{\omega}}_{\mathrm{IB}} \times {}_{\mathrm{B}}\boldsymbol{r}_{PQ} + 2\,{}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times {}_{\mathrm{B}}\dot{\boldsymbol{r}}_{PQ} \\ &\quad + {}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times ({}_{\mathrm{B}}\boldsymbol{\omega}_{\mathrm{IB}} \times {}_{\mathrm{B}}\boldsymbol{r}_{PQ}). \end{aligned} \tag{2.6}$$

Die Kinematikübertragungen der Geschwindigkeiten lauten unter Verwendung von (2.2) und (2.5) mit (2.4)

$$\begin{aligned}
{}_{\mathrm{K}_1}\boldsymbol{\omega}_{\mathrm{IK}_1} &= \mathbf{A}^{\mathsf{T}}_{\mathrm{K}_0\mathrm{K}_1}(\boldsymbol{l})\,({}_{\mathrm{K}_0}\boldsymbol{\omega}_{\mathrm{IK}_0} + {}_{\mathrm{K}_0}\boldsymbol{\omega}_{\mathrm{K}_0\mathrm{K}_1}(\boldsymbol{l},\boldsymbol{c}))\,,\\
{}_{\mathrm{I}}\boldsymbol{v}_{S_1} &= {}_{\mathrm{I}}\boldsymbol{v}_{S_0} + \mathbf{A}_{\mathrm{IK}}[\mathsf{b}_0]\,({}_{\mathrm{K}_0}\boldsymbol{\omega}_{\mathrm{IK}_0} \times {}_{\mathrm{K}_0}\boldsymbol{r}_{S_0M_0}\\
&\qquad\qquad +{}_{\mathrm{K}_0}\dot{\boldsymbol{r}}_{MN}(\boldsymbol{l},\boldsymbol{c}) + {}_{\mathrm{K}_0}\boldsymbol{\omega}_{\mathrm{IK}_0} \times {}_{\mathrm{K}_0}\boldsymbol{r}_{MN}(\boldsymbol{l}))\\
&\qquad + \mathbf{A}_{\mathrm{IK}}[\mathsf{b}_1]\,({}_{\mathrm{K}_1}\boldsymbol{\omega}_{\mathrm{IK}_1} \times (-{}_{\mathrm{K}_1}\boldsymbol{r}_{S_1N_1}))\,.
\end{aligned} \tag{2.7}$$

Man beachte hierbei, dass die Transposition der Koordinatentransformationsmatrix gleichbedeutend ist mit deren Inversion. Es ist also $\mathbf{A}^{\mathsf{T}}_{\mathrm{K}_0\mathrm{K}_1} = \mathbf{A}_{\mathrm{K}_1\mathrm{K}_0}$. Die Kinematikübertragungen für die Beschleunigungen lassen sich unter Zuhilfenahme von (2.6) analog berechnen mit

$$\begin{aligned}
{}_{\mathrm{K}_1}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_1} &= \mathbf{A}^{\mathsf{T}}_{\mathrm{K}_0\mathrm{K}_1}(\boldsymbol{l})\ {}_{\mathrm{K}_0}(\dot{\boldsymbol{\omega}}_{\mathrm{IK}_1})\\
&= \mathbf{A}^{\mathsf{T}}_{\mathrm{K}_0\mathrm{K}_1}(\boldsymbol{l})\ {}_{\mathrm{K}_0}((\boldsymbol{\omega}_{\mathrm{IK}_0} + \boldsymbol{\omega}_{\mathrm{K}_0\mathrm{K}_1})^{\cdot}) = \mathbf{A}^{\mathsf{T}}_{\mathrm{K}_0\mathrm{K}_1}(\boldsymbol{l})\ {}_{\mathrm{K}_0}(\dot{\boldsymbol{\omega}}_{\mathrm{IK}_0} + \dot{\boldsymbol{\omega}}_{\mathrm{K}_0\mathrm{K}_1})\\
&= \mathbf{A}^{\mathsf{T}}_{\mathrm{K}_0\mathrm{K}_1}(\boldsymbol{l})\,({}_{\mathrm{K}_0}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_0} + {}_{\mathrm{K}_0}\dot{\boldsymbol{\omega}}_{\mathrm{K}_0\mathrm{K}_1}(\boldsymbol{l},\boldsymbol{c},\dot{\boldsymbol{c}})\\
&\qquad\qquad + {}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}[\mathsf{b}_0] \times {}_{\mathrm{K}_0}\boldsymbol{\omega}_{\mathrm{K}_0\mathrm{K}_1}(\boldsymbol{l},\boldsymbol{c}))\\
{}_{\mathrm{I}}\dot{\boldsymbol{v}}_{S_1} &= {}_{\mathrm{I}}\dot{\boldsymbol{v}}_{S_0} + \mathbf{A}_{\mathrm{IK}}[\mathsf{b}_0]\,({}_{\mathrm{K}_0}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_0} \times {}_{\mathrm{K}_0}\boldsymbol{r}_{S_0M}\\
&\qquad\qquad + {}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}[\mathsf{b}_0] \times ({}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}[\mathsf{b}_0] \times {}_{\mathrm{K}_0}\boldsymbol{r}_{S_0M})\\
&\qquad\qquad + {}_{\mathrm{K}_0}\ddot{\boldsymbol{r}}_{MN}(\boldsymbol{l},\boldsymbol{c},\dot{\boldsymbol{c}}) + {}_{\mathrm{K}_0}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_0} \times {}_{\mathrm{K}_0}\boldsymbol{r}_{MN}(\boldsymbol{l})\\
&\qquad\qquad + 2\ {}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}[\mathsf{b}_0] \times {}_{\mathrm{K}_0}\dot{\boldsymbol{r}}_{MN}(\boldsymbol{l},\boldsymbol{c})\\
&\qquad\qquad + {}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}[\mathsf{b}_0] \times ({}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}[\mathsf{b}_0] \times {}_{\mathrm{K}_0}\boldsymbol{r}_{MN}(\boldsymbol{l})))\\
&\qquad + \mathbf{A}_{\mathrm{IK}}[\mathsf{b}_1]\,({}_{\mathrm{K}_1}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_1} \times (-{}_{\mathrm{K}_1}\boldsymbol{r}_{S_1N})\\
&\qquad\qquad + {}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}[\mathsf{b}_1] \times ({}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{IK}}[\mathsf{b}_1] \times (-{}_{\mathrm{K}_1}\boldsymbol{r}_{S_1N})))\,.
\end{aligned} \tag{2.8}$$

Es bleibt zu bemerken, dass die Gleichungen (2.1), (2.7) und (2.8) die drei Kinematikübertragungen allgemein beschreiben und damit immer gültig sind. Für einzelne Gelenktypen ergeben sich aber fast immer erhebliche Vereinfachungen, mit denen eine Implementierung optimiert werden kann.

Schliesslich implementiert jeder Gelenktyp eine Funktion `getCoordinatesDerivative`, welche die (lokale) Kinematikgleichung implementiert, also die zeitliche Änderung der generalisierten Lagen

$$\dot{\boldsymbol{l}} = \mathbf{F}(\boldsymbol{l})\,\boldsymbol{c} \tag{2.9}$$

in Abhängigkeit der Lagen und Geschwindigkeiten zurückgibt.

2.1.4 Kontakte

Auf einzelnen Starrkörpern werden Kontaktpunkte definiert, die mit der Kollisionsgeometrie interagieren können. Zur Vereinfachung der Kollisionserkennung, wird ein vorgesehener Kontaktpunkte D mit einem ins Körperinnere zeigenden Liniensegment $\boldsymbol{d}$ versehen, auf dem der aktuelle Kontaktpunkt D' bei (numerischem) Eindringen zu liegen kommt. Abbildung 2.1 illustriert einen solchen *Segmentkontakt*. Das Kontaktobjekt speichert eine Referenz `b` auf

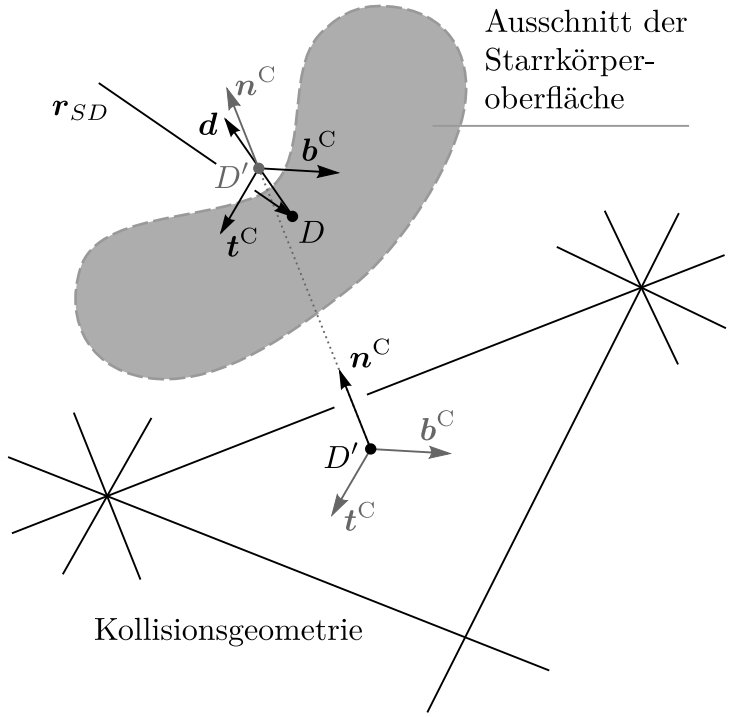

Abbildung 2.1: Segmentkontakt mit vordefiniertem Kontaktpunkt D und Kontaktsegment $\boldsymbol{d}$. Der aktuelle Kontaktpunkt D' kommt auf dem Segment zu liegen. Anhand der Kontaktnormalen $\boldsymbol{n}^{\mathrm{C}}$ werden die beiden Tangentialrichtungen $\boldsymbol{t}^{\mathrm{C}}$ und $\boldsymbol{b}^{\mathrm{C}}$ konstruiert.

den dazugehörigen Starrkörper, die Position ${}_{\mathrm{K}}\boldsymbol{r}_{SD}$ des Kontaktpunktes D relativ zum Schwerpunkt S des Starrkörpers in dessen körperfesten Koordinaten K, sowie die Richtung und Länge des Liniensegmentes ${}_{\mathrm{K}}\boldsymbol{d}$. Das Liniensegment muss kein Einheitsvektor sein. Der Treiber, siehe Unterabschnitt 2.1.6, führt im Zuge der Lösung der Bewegungsgleichung eine Kollisionsdetektion durch und bestimmt die aktuellen Kontaktpunkte ${}_{\mathrm{I}}\boldsymbol{r}_{OD'}$ aller Kontakte zusammen mit den dazugehörigen Kontaktnormalen ${}_{\mathrm{I}}\boldsymbol{n}$, d.h. den Normalvektoren der geschnittenen Dreiecke der Kollisionsgeometrie. Der aktuelle Kontaktpunkt und die zugehörige Normale werden im Kontaktobjekt gespeichert. Anhand der Normalen konstruiert ein Kontaktobjekt einen Tangentialvektor ${}_{\mathrm{K}}\boldsymbol{t}$ und mit ${}_{\mathrm{K}}\boldsymbol{b} = {}_{\mathrm{K}}\boldsymbol{n} \times {}_{\mathrm{K}}\boldsymbol{t}$ einen Bitangentialvektor. Im Falle von isotroper Reibung sind die Richtungen dieser beiden Vektoren in der Tangentialebene beliebig. Bei anisotroper Reibung wird der Tangentialvektor ${}_{\mathrm{K}}\boldsymbol{t}$ in einer ausgeprägten Richtung des dazugehörigen Starrkörpers konstruiert. Kontakte sind einseitige Bindungen und mit ihnen gehen, wie auch bei bilateralen Bindungen, Bindungskräfte einher, die die Bindung erzwingen. In Normalrichtung heisst die Bindungskraft (Kontakt-)Normalkraft λ_n, in tangentialer Richtung Reibkraft $\boldsymbol{\lambda}_f$ und beide werden als Kontaktkraft $\boldsymbol{\lambda}_c = (\lambda_n, \boldsymbol{\lambda}_f)^{\mathsf{T}}$ zusammengefasst. Die Kontaktkräfte aller Kontakte c werden aber vom Treiber, Unterabschnitt 2.1.6, verwaltet. Allerdings definiert jeder Kontakttyp eine `prox`-Funktion, die bei der Bestimmung der einzelnen Kontaktkräfte $\boldsymbol{\lambda}_c$ benötigt wird. Darauf wird in Abschnitt 2.7 detailliert eingegangen. Tabelle 2.4 zeigt einen Überblick über die Daten von Kontaktobjekten.

Tabelle 2.4: Die Datenstruktur eines Kontakobjektes.

Beschreibung	Bezeichnung	Codename
zugehöriger Starrkörper	`b`	`body`
Ursprung des Kontaktsegments	${}_{\mathrm{K}}\boldsymbol{r}_{SD}$	`segmentOrigin`
Richtung des Kontaktsegments	${}_{\mathrm{K}}\boldsymbol{d}$	`segmentDirection`
aktueller Kontaktpunkt	${}_{\mathrm{I}}\boldsymbol{r}_{OD'}$, ${}_{\mathrm{I}}\boldsymbol{n}$	`collisionPoint`
Tangentialvektoren	${}_{\mathrm{K}}\boldsymbol{t}({}_{\mathrm{K}}\boldsymbol{n})$, ${}_{\mathrm{K}}\boldsymbol{b}({}_{\mathrm{K}}\boldsymbol{n})$	`getTangents`
prox-Funktion des Kontakttyps	$\mathrm{prox}(\boldsymbol{\lambda}_c, r\mathbf{G}_c\boldsymbol{\lambda} + \boldsymbol{c}_c, \Delta t)$	`prox`

2.1.5 Explizite äussere Kräfte

Neben den Kraftelementen, die allfällig in nicht aktuierten Gelenken spezifiziert sind, und den Kontaktkräften, können weitere äussere Kräfte und Momente an einem Starrkörper angreifen. Ein Kraftobjekt implementiert eine Funktion `ForceAndMoment`, welche in Abhängigkeit der Starrkörperträgheiten, -lagen und -geschwindigkeiten, ihre Beiträge zu den Resultierenden ${}_{\mathrm{I}}\boldsymbol{F}_S$ und ${}_{\mathrm{K}}\boldsymbol{M}_S$ des Starrkörpers liefert. Die auf einen Körper wirkenden Kraftobjekte werden zur Liste $[\mathtt{f}_0, \mathtt{f}_1, \dots]$ der äusseren Kräfte des betreffenden Starrkörpers hinzugefügt.

Im Rahmen der Simulation einer Bobfahrt sind die Gravitationskraft und ein aerodynamischer Widerstand als solche Kraftobjekte implementiert. Die Gravitationskraft leistet einen Beitrag

$$ {}_{\mathrm{I}}\boldsymbol{F}_S[\mathtt{b}] \mathrel{+}= m[\mathtt{b}]{}_{\mathrm{I}}\boldsymbol{g}, \qquad {}_{\mathrm{K}}\boldsymbol{M}_S[\mathtt{b}] \mathrel{+}= \mathbf{0} $$

zur Resultierenden äusseren Kraft des betroffenen Körpers `b` und kein Moment bezüglich dessen Schwerpunkt S. Hierbei bezeichnet ${}_{\mathrm{I}}\boldsymbol{g} = (0, -9.81\frac{\mathrm{m}}{\mathrm{s}}, 0)^{\mathsf{T}}$ die Erdbeschleunigung. Der aerodynamische Widerstand wird als im Schwerpunkt angreifend modelliert und trägt deshalb ebenfalls nichts zum äusseren Moment ${}_{\mathrm{K}}\boldsymbol{M}_S[\mathtt{b}]$ bei. Die Kraft

$$ {}_{\mathrm{I}}\boldsymbol{F}_S[\mathtt{b}] \mathrel{+}= -\frac{1}{2}\rho c_d A \|{}_{\mathrm{I}}\boldsymbol{v}_S[\mathtt{b}]\| {}_{\mathrm{I}}\boldsymbol{v}_S[\mathtt{b}], \qquad {}_{\mathrm{K}}\boldsymbol{M}_S[\mathtt{b}] \mathrel{+}= \mathbf{0} $$

ist proportional zum Quadrat der Geschwindigkeit und ist dieser entgegengerichtet. Der Luftdruck wird hierin $\rho = 1.12\frac{\mathrm{kg}}{\mathrm{m}^3}$ gesetzt und der Widerstandsbeiwert c_d wurde zusammen mit der effektiven Fläche A im Rahmen des *Citius* Projektes gemessen und mit $c_d A = 0.2052\mathrm{m}^2$ implementiert [3].

2.1.6 Treiber (Solver)

Der Treiber ist zuständig für Aufstellen der Bewegungsgleichung, für die Kontaktdetektion und für das Lösen des Inklusionsproblems. Er treibt den Bewegungszustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})$

Tabelle 2.5: Übersicht der vom Treiber verwalteten Daten.

Beschreibung	Bezeichnung	Codename
Inertialreferenz	$\mathtt{b}_O$	`inertialReference`
Zeit	t	`time`
Eingangszustand	$\boldsymbol{k}$, $\boldsymbol{m}$, $\dot{\boldsymbol{m}}$	`controls`, `controlVelocities`, `controlAccelerations`
dynamischer Zustand	$\boldsymbol{q}$, $\boldsymbol{u}$	`coordinates`, `velocities`
kontaktinduzierte Kräfte	$\boldsymbol{i}$, $\boldsymbol{j}$	`backpropagatedControlForces`, `backpropagatedJointForces`
Dimension des Eingangszustands,	$n_{\boldsymbol{k}}$	`countControls`
der Minimalkoordinaten,	$n_{\boldsymbol{q}}$	`countCoordinates`
der Minimalgeschwindigkeiten	$n_{\boldsymbol{u}}$	`countVelocities`
Liste aller Gelenke	$[\mathtt{j}_0, \mathtt{j}_1, \dots]$	`joints`
Liste der Starrkörper ohne Ursprung	$[\mathtt{b}_0, \mathtt{b}_1, \dots]$	`bodies`
Liste aller Kontakte	$[\mathtt{c}_0, \mathtt{c}_1, \dots]$	`contacts`
Kollisionsgeometrie		`collisionGeometry`

in Zeitschritten Δt voran und berechnet die von den Kontaktkräften induzierten Kräfte $(\boldsymbol{i}, \boldsymbol{j})$, die später mit haptischem Rendern für den Piloten erlebbar gemacht werden. Das haptische Rendern ist Inhalt von Kapitel 5. Die einzelnen Schritte für das Voranbringen des Bewegungszustandes sind in den restlichen Abschnitten 2.2 bis 2.7 dieses Kapitels behandelt.

Tabelle 2.5 listet einige, der vom Treiber verwalteten Daten auf. Er hält eine Referenz $\mathtt{b}_O$ auf den inertialen Ursprung O, der die Wurzel des Kinematikgraphen, vgl. Abschnitt 2.2 darstellt. Der Treiber verwaltet die Systemzustände aus Tabelle 2.1, zusammen mit der Zeit t und speichert die Dimensionen $n_{\boldsymbol{q}}$, $n_{\boldsymbol{u}}$ und $n_{\boldsymbol{k}}$ der Koordinaten $\boldsymbol{q}$, Geschwindigkeiten $\boldsymbol{u}$ bzw. Lagen $\boldsymbol{k}$ der Eingangsgrössen. Weiter hält der Treiber Listen aller Gelenke, Körper und Kontakte im Mehrkörpermodell und eine Referenz auf die Kollisionsgeometrie, über die die Kontaktdetektion stattfindet.

Daneben besitzt der Treiber, im Englischen *Solver* genannt, viele Methoden, die in den folgenden Abschnitten 2.2 bis 2.7 erläutert sind und hier der Kürze wegen nicht einzeln aufgelistet werden.

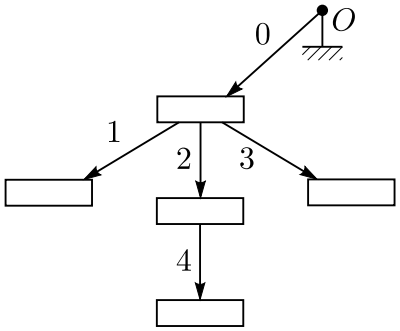

Abbildung 2.2: Allgemeiner kreisfreier Kinematikgraph. Die Knoten stellen Starrkörper, die Kanten Gelenke dar.

2.2 Topologie des Mehrkörpersystems

Zur Kinematik eines Mehrkörpersystems lässt sich ein Graph zeichnen, dessen Knoten die starren Körper und Kanten die dazwischenliegenden Gelenke darstellen. Diese besondere Struktur lässt sich wiederholt ausnutzen und ermöglicht es die Bewegungsgleichungen numerisch aufzustellen. Für die Simulation eines Mehrkörpermodells muss deshalb einzig diese Struktur initialisiert werden. Danach kann der Treiber das Mehrkörpermodell simulieren. Für das Aufbauen des Graphen müssen die einzelnen Starrkörper angelegt und Gelenke initialisiert werden, die jeweils zwei der Starrkörperobjekte verbinden.

Dieses Kapitel beschränkt sich auf Systeme, deren Kinematikgraph eine *offene Baumstruktur* aufweist mit jeweils einer einzelnen und gerichteten Kante zwischen zwei Knoten. Ein solcher Graph ist gerichtet, zusammenhängend und enthält keine Mehrfachkanten, Schlingen oder Kreise [84]. Abbildung 2.2 zeigt beispielhaft einen solchen Graphen. Man beachte, dass der Ursprung O auch als Knoten und damit (spezieller) Starrkörper in den Graphen aufgenommen wird.

Der inertiale Ursprung O ist immer die Wurzel des Graphen und die Liste seiner weggehenden Gelenke $[\mathtt{j}_0, \mathtt{j}_1, \dots]$ liefert die ersten Kanten des Graphen. Über den verbundenen Starrkörper $\mathtt{b}_1$ eines jeden Gelenkes und dessen Liste von wegführenden Gelenken ist der Graph programmatisch realisiert und durchlaufbar. In einer Traversierung des Kinematikgraphen nach Algorithmus 2.1 wird der Speicher für den Systemzustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})$ alloziert und mit den Anfangswerten $(\boldsymbol{l}_{\text{init}}, \boldsymbol{c}_{\text{init}})$ der Gelenke initialisiert. Beim Algorithmus 2.1 handelt es sich um einen rekursiv formulierten Tiefendurchlauf[4] des Graphen [14, 54], womit sich für das Beispiel aus Abbildung 2.2 die einzelnen Tupel $\boldsymbol{q}$, $\boldsymbol{k}$, $\boldsymbol{u}$ und $\boldsymbol{m}$ aus den entsprechenden Gelenkzuständen in der Reihenfolge $0, 1, 2, 4, 3$ aufbauen. In einem weiteren Durchlauf durch den Kinematikgraphen gemäss Algorithmus 2.2 werden die Listen $[\mathtt{j}_0, \mathtt{j}_1, \dots]$ der Gelenke, $[\mathtt{b}_0, \mathtt{b}_1, \dots]$ Starrkörper und $[\mathtt{c}_0, \mathtt{c}_1, \dots]$ Kontakte des Treibers assembliert, welche später für den Aufbau der Bewegungsgleichungen nützlich sind.

[4] Häufig wird dafür auch das Wort Tiefensuche verwendet. Es wird hier vermieden, weil es sich eher um ein Besuchen als eine Suche handelt. Noch geläufiger sind die englischen Ausdrücke *depth-first search* oder *depth-first traversal*.

Algorithmus 2.1

```
# call to assemble initial system state (q,k,u,m).
# 'solver' is supposed to be a Solver object.
solver.constructSystemState(solver.inertialReference,
                            solver.coordinates, solver.controls,
                            solver.velocities, solver.controlVelocities)

# this function is a member method of the Solver class.
# q,k,u,m are extendible vectors and passed by reference.
# They have a property '.length' that returns the number
# of items currently stored in the instance and a method
# '.add(int n)', that allocates memory for another n entries.
def constructSystemState(self,body,q,k,u,m):
  for joint in body.joints:
    if not joint.isKinematicallyActuated:
      joint.coordinatesOffset = q.length
      q.add(joint.coordinatesCount)
      for i in range(joint.coordinatesCount):
        q[joint.coordinatesOffset+i] = joint.initialCoordinates[i]
      joint.velocitiesOffset = u.length
      u.add(joint.velocitiesCount)
      for i in range(joint.velocitiesCount):
        u[joint.velocitiesOffset+i] = joint.initialVelocities[i]
    else: # joint is kinematically actuated
      joint.coordinatesOffset = k.length
      k.add(joint.coordinatesCount)
      joint.velocitiesOffset = m.length
      m.add(joint.velocitiesCount)
      # note that in case of an actuated joint the state (k,m)
      # is not actively initialised. These values are usually
      # read from sensors and therefore are processed later.
      # However, depending on the programming language, these
      # arrays may be initialised with zero by default.
    self.constructSystemState(joint.child,q,k,u,m)
```

Algorithmus 2.2

```
# call to assemble 'joints', 'bodies' and 'contacts' lists of
# a Solver object. 'solver' is supposed to be a Solver object.
solver.assembleLists()

# this function is a member method of the Solver class.
# 'self' therefore refers to the solver instance.
def assembleLists(self):
  self.joints = list(self.inertialReference.joints)
  self.bodies = [j.child for j in self.joints]
  index = 0
  while index < len(bodies):
    body = self.bodies[index]
    self.joints.extend(body.joints)
    for joint in body.joints:
      self.bodies.append(joint.child)
    self.contacts.extend(body.contacts)
    index += 1
```

2.3 Kinematik des Mehrkörpersystems

In einem ersten Schritt werden mit Hilfe der Graphstruktur der Kinematik des Mehrkörpersystem anhand eines aktuellen Zustands $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})$ die kinematischen Zustände aller Starrkörper aktualisiert. Dazu gehören nach Tabelle 2.2 die Orientierung und Position, die Geschwindigkeiten, Pseudogeschwindigkeiten und Pseudobeschleunigungen, sowie die Jacobimatrizen. Ist ein aktueller Zustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})$ gegeben, lassen sich mit den Algorithmen 2.3 und 2.4 in zwei Traversierungen des Kinematikgraphen die Lagen $\mathbf{A}_{\mathrm{IK}_i}$, ${}_{\mathrm{I}}\boldsymbol{r}_{OS_i}$ und Geschwindigkeiten ${}_{\mathrm{K}_i}\boldsymbol{\omega}_{\mathrm{IK}_i}$, ${}_{\mathrm{I}}\boldsymbol{v}_{S_i}$ aller Starrkörper $\{i\}$ aktualisieren. Darin implementieren die Funktkionen `propagateDisplacement` und `propagateVelocities` gemäss Tabelle 2.3 die Gleichungen (2.1) bzw. (2.7).

Die zeitlichen Ableitungen $(\dot{\boldsymbol{q}}, \dot{\boldsymbol{k}})$ der generalisierten Koordinaten werden in Abhängigkeit der generalisierten Geschwindigkeiten dargestellt. Diese Arbeit beschränkt sich auf Systeme in denen die kinematischen Gleichungen von der Form

$$\dot{\boldsymbol{k}} = \boldsymbol{m}, \qquad \dot{\boldsymbol{q}} = \mathbf{B}(\boldsymbol{q})\boldsymbol{u} \tag{2.10}$$

sind. Man beachte, dass dies auch den Spezialfall $\mathbf{B}(\boldsymbol{q}) = \mathbf{I}$ enthält. Aufgrund der vollständigen Parametrisierung der Lagen und Orientierungen aller Körper im System durch die generalisierten Koordinaten $(\boldsymbol{q}, \boldsymbol{k})$ kann die Translationsgeschwindigkeit $\boldsymbol{v}_{S_i}$ des Schwer-

Algorithmus 2.3

```
# call to to update 'orientation' and 'position' of
# all rigid bodies in the model.
# 'solver' is supposed to be a Solver object.
solver.updateDisplacement(solver.inertialReference,
                          solver.coordinates,solver.controls)

# this function is a member method of the Solver class.
# 'self' therefore refers to the solver instance.
def updateDisplacement(self,body,q,k):
  for joint in body.joints:
    joint.updateLocalState(q,k)
    [joint.child.orientation, joint.child.position] = \
    joint.propagateDisplacement(joint.parent.orientation,
                                joint.parent.position)
    self.updateDisplacement(joint.child,q,k)
```

Algorithmus 2.4

```
# call to update 'angularVelocity' and 'linearVelocity'
# of all rigid bodies in the model.
# 'solver' is supposed to be a Solver object.
solver.updateVelocities(solver.inertialReference,
                        solver.coordinates, solver.controls,
                        solver.velocities, solver.controlVelocities)

# this function is a member method of the Solver class.
# 'self' therefore refers to the solver instance.
def updateVelocities(self,body,q,k,u,m):
  for joint in body.joints:
    joint.updateLocalState(q,k,u,m)
    [joint.child.angularVelocity,
     joint.child.linearVelocity] = \
    joint.propagateVelocities(joint.parent.angularVelocity,
                              joint.parent.linearVelocity)
    self.updateVelocities(joint.child,q,k,u,m)
```

Algorithmus 2.5

```
# 'solver' is supposed to be a Solver object.
# 'zeroVelocities' is supposed to be a vector of the same
# length as 'solver.velocities' with all entries being zero.
zeroVelocities = [0 for i in range(solver.velocities.length)]
# call to update 'angularPseudoVelocity' and 'linearPseudoVelocity'
# of all rigid bodies in the model.
solver.updatePseudoVelocities(solver.inertialReference,
                              solver.coordinates, solver.controls,
                              zeroVelocities, solver.controlVelocities)

# this function is a member method of the Solver class.
# 'self' therefore refers to the solver instance.
def updatePseudoVelocities(self,body,q,k,u,m):
  for joint in body.joints:
    joint.updateLocalState(q,k,u,m)
    [joint.child.angularPseudoVelocity,
     joint.child.linearPseudoVelocity] = \
    joint.propagateVelocities(joint.parent.angularPseudoVelocity,
                              joint.parent.linearPseudoVelocity)
    self.updatePseudoVelocities(joint.child,q,k,u,m)
```

punkts eines Starrkörpers i nach

$$\begin{aligned} {}_{\mathrm{I}}\boldsymbol{v}_{S_i} = {}_{\mathrm{I}}\dot{\boldsymbol{r}}_{OS_i} &= \frac{\partial_{\mathrm{I}}\boldsymbol{r}_{OS_i}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial_{\mathrm{I}}\boldsymbol{r}_{OS_i}}{\partial \boldsymbol{k}}\dot{\boldsymbol{k}} \\ &= {}_{\mathrm{I}}\mathbf{J}_{S_i}\boldsymbol{u} + {}_{\mathrm{I}}\mathbf{H}_{S_i}\boldsymbol{m} \\ &= {}_{\mathrm{I}}\mathbf{J}_{S_i}\boldsymbol{u} + {}_{\mathrm{I}}\boldsymbol{\nu}_{S_i} \end{aligned} \tag{2.11}$$

geschrieben werden und analog seine Winkelgeschwindigkeit ${}_{\mathrm{K}_i}\boldsymbol{\omega}_{\mathrm{IK}_i}$ mit

$$\begin{aligned} {}_{\mathrm{K}_i}\boldsymbol{\omega}_{\mathrm{IK}_i} &= {}_{\mathrm{K}_i}\mathbf{J}_{R_i}\boldsymbol{u} + {}_{\mathrm{K}_i}\mathbf{H}_{R_i}\boldsymbol{m} \\ &= {}_{\mathrm{K}_i}\mathbf{J}_{R_i}\boldsymbol{u} + {}_{\mathrm{K}_i}\boldsymbol{\nu}_{R_i}. \end{aligned} \tag{2.12}$$

Darin bezeichnen die Matrizen $\mathbf{J}_{S_i}$ und $\mathbf{H}_{S_i}$ die Translationsjacobimatrizen des Körpers i bezüglich seinem Schwerpunkt S_i, und $\mathbf{J}_{R_i}$ und $\mathbf{H}_{R_i}$ die Rotationsjacobimatrizen des Körpers i. Die Pseudogeschwindigkeiten ${}_{\mathrm{I}}\boldsymbol{\nu}_{S_i}$ und ${}_{\mathrm{K}}\boldsymbol{\nu}_{R_i}$ aller Körper lassen sich in einem weiteren Durchgang durch den Kinematikgraphen mit einem Pseudozustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u} = \mathbf{0}, \boldsymbol{m})$ bestimmen. Denn für verschwindende generalisierte Geschwindigkeiten $\boldsymbol{u} = \mathbf{0}$ bleiben gemäss den Gleichungen (2.11) und (2.12) in den Kinematikübertragungen der Gelenke genau die Pseudogeschwindigkeiten übrig. Algorithmus 2.5 präsentiert das Aktualisieren der Pseudogeschwindigkeiten anhand des Pseudozustandes $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u} = \mathbf{0}, \boldsymbol{m})$. Die

Algorithmus 2.6

```
# 'solver' is supposed to be a Solver object.
# 'zeroAccelerations' is supposed to be a vector of the same
# length as 'solver.velocities' with all entries being zero.
zeroAccelerations = [0 for i in range(solver.velocities.length)]
# call to update 'angularPseudoAcceleration' and
# 'linearPseudoAcceleration' of all rigid bodies in the model.
solver.updatePseudoVelocities(
        solver.inertialReference,
        solver.coordinates, solver.controls,
        solver.velocities, solver.controlVelocities,
        zeroAccelerations, solver.controlAccelerations)

# this function is a member method of the Solver class.
# 'self' therefore refers to the solver instance.
def updatePseudoAccelerations(self,body,q,k,u,m,du,dm):
  for joint in body.joints:
    joint.updateLocalState(q,k,u,m,du,dm)
    [joint.child.angularPseudoAcceleration,
     joint.child.linearPseudoAcceleration] = \
    joint.propagateVelocities(
            joint.parent.angularPseudoAcceleration,
            joint.parent.linearPseudoAcceleration)
    self.updatePseudoVelocities(joint.child,q,k,u,m,du,dm)
```

Beschleunigungen der Starrkörper $\{i\}$ lassen sich mit den Jacobimatrizen in der Form

$$\begin{aligned} {}_{\mathrm{K}_i}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_i} &= {}_{\mathrm{K}_i}\mathbf{J}_{R_i}\dot{\boldsymbol{u}} + {}_{\mathrm{K}_i}\boldsymbol{\kappa}_{R_i}, \\ {}_{\mathrm{I}}\dot{\boldsymbol{v}}_{S_i} &= {}_{\mathrm{I}}\mathbf{J}_{S_i}\dot{\boldsymbol{u}} + {}_{\mathrm{I}}\boldsymbol{\kappa}_{S_i} \end{aligned} \tag{2.13}$$

schreiben. Die Pseudobeschleunigungen ${}_{\mathrm{I}}\boldsymbol{\kappa}_{S_i}$ und ${}_{\mathrm{K}}\boldsymbol{\kappa}_{R_i}$ lassen sich analog den Pseudogeschwindigkeiten mit dem aktuellen Zustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}, \boldsymbol{m})$ und Pseudoänderungen $(\dot{\boldsymbol{u}} = \mathbf{0}, \dot{\boldsymbol{m}})$ in einer Traversierung des Kinematikgraphen nach Algorithmus 2.6 aktualisieren.

Schliesslich müssen die Jacobimatrizen aller Körper aktualisiert werden. Dabei nützt man aus, dass aufgrund von den Gleichungen (2.11) und (2.12)

$$\begin{aligned} {}_{\mathrm{I}}\mathbf{J}_{S_i}\boldsymbol{u} + {}_{\mathrm{I}}\mathbf{H}_{S_i}\boldsymbol{m} &= {}_{\mathrm{I}}\boldsymbol{v}_{S_i}, \\ {}_{\mathrm{K}_i}\mathbf{J}_{R_i}\boldsymbol{u} + {}_{\mathrm{K}_i}\mathbf{H}_{R_i}\boldsymbol{m} &= {}_{\mathrm{K}_i}\boldsymbol{\omega}_{\mathrm{IK}_i} \end{aligned}$$

gilt. Propagiert man einen Pseudozustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u}_n, \boldsymbol{m} = \mathbf{0})$ durch den Kinematikgraphen mit $\boldsymbol{u}_n = (0, \ldots, 0, 1, 0, \ldots, 0)^{\mathsf{T}}$, wobei die Eins an der Stelle n steht, ergeben sich als Geschwindigkeiten der einzelnen Starrkörper gerade die n-ten Spalten der Jacobimatrizen ${}_{\mathrm{I}}\mathbf{J}_{S_i}$ bzw. ${}_{\mathrm{K}_i}\mathbf{J}_{R_i}$. Mit wiederholten Traversierungen für alle $\boldsymbol{u}_n$ werden die Jacobimatrizen

Algorithmus 2.7: (1. Teil)

```
# 'solver' is supposed to be a Solver object.
# call to update 'rotationJacobi', 'translationJacobi',
# 'rotationControlJacobi' and 'translationControlJacobi'
# of all rigid bodies in the model.
solver.updateJacobis(solver.coordinates, solver.controls)

# these functions are a member methods of the Solver class.
# 'self' therefore refers to the solver instance.
def updateJacobis(self,q,k):
  # update classical Jacobis J
  pseudoControlVelocities = \
  [0 for i in range(controlVelocities.length)]
  for n in range(velocities.length):
    pseudoVelocities = [0 for i in range(velocities.length)]
    pseudoVelocities[n] = 1
    self.updateJacobiColumn(true, n, self.inertialReference,
                            self.coordinates, self.controls,
                            pseudoVelocities, pseudoControlVelocities)
  # update input Jacobis H
  pseudoVelocities = [0 for i in range(velocities.length)]
  for n in range(controlVelocities.length):
    pseudoControlVelocities = \
    [0 for i in range(controlVelocities.length)]
    pseudoControlVelocities[n] = 1
    self.updateJacobiColumn(false, n, self.inertialReference,
                            self.coordinates, self.controls,
                            pseudoVelocities, pseudoControlVelocities)
```

spaltenweise berechnet. Analog ergeben sich die n-ten Spalten der Jacobimatrizen ${}_{\mathrm{I}}\mathbf{H}_{S_i}$ und ${}_{\mathrm{K}_i}\mathbf{H}_{R_i}$ mit einem Pseudozustand $(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{u} = \mathbf{0}, \boldsymbol{m}_n)$. Unter der Annahme, dass die Lagen der Starrkörper bereits mit den aktuellen generalisierten Lagen $(\boldsymbol{q}, \boldsymbol{k})$ aktualisiert sind, assembliert Algorithmus 2.7 die Jacobimatrizen aller Starrkörper spaltenweise mit Hilfe der Geschwindigkeitsübertragung `propagateVelocity` der Gelenke nach Gleichung (2.7). Man beachte, dass sämtliche Jacobimatrizen des inertialen Ursprungs O Nullmatrizen sind, da dieser konstant verschwindende Geschwindigkeiten hat, d.h. inertial ruht.

Damit sind jetzt alle kinematischen Grössen der Starrkörper aktualisiert. Es folgt die Aktualisierung der an den Körpern angreifenden äusseren Kräfte, bevor mit diesen Grössen die Bewegungsgleichungen des glatten Mehrkörpersystems aufgestellt werden.

Algorithmus 2.7: (Fortsetzung)

```
def updateJacobiColumn(self, updateClassicalJacobis, columnIndex,
                       body, q, k, pseudo_u, pseudo_m):
  for joint in body.joints:
    joint.updateLocalState(q,k,pseudo_u,pseudo_m)
    if updateClassicalJacobis:
      pJR = joint.parent.rotationJacobi
      parentAngularVelocity = pJR[:,columnIndex]
      pJS = joint.parent.translationJacobi
      parentLinearVelocity = pJS[:,columnIndex]
    else:
      pHR = joint.parent.rotationControlJacobi
      parentAngularVelocity = pHR[:,columnIndex]
      pHS = joint.parent.translationControlJacobi
      parentLinearVelocity = pHS[:,columnIndex]
    [columnRotationJacobi, columnTranslationJacobi] = \
    joint.propagateVelocity(parentAngularVelocity,
                            parentLinearVelocity)
    if updateClassicalJacobis:
      [child.rotationJacobi[:,columnIndex],
       child.translationJacobi[:,columnIndex]] = \
      [columnRotationJacobi, columnTranslationJacobi]
    else: # update control jacobis
      [child.rotationControlJacobi[:,columnIndex],
       child.translationControlJacobi[:,columnIndex]] = \
      [columnRotationJacobi, columnTranslationJacobi]
    self.updateJacobiColumn(updateClassicalJacobis, columnIndex,
                            joint.child, q, k, pseudo_u, pseudo_m)
```

2.4 Kraftelemente im Mehrkörpersystem

Wie in Abschnitt 2.1 erläutert, treten in Mehrkörpersystemen, abgesehen von den Kontaktkräften, zwei Klassen von Kraftelementen als äussere Kräfte der Starrkörper auf. Zum Einen sind das *freie Kraftelemente*, die an einzelnen Starrkörpern angreifen und deren Betrag und Richtung von den kinematischen Starrkörpergrössen abhängen. Man vergleiche dazu Unterabschnitt 2.1.5. Zum Anderen können *Kraftelemente in Gelenken* definiert sein, die auf die zwei verbundenen Körper mit sich entgegengesetzten Kräften und Momenten wirken, siehe Unterabschnitt 2.1.3. Die Beträge hängen dabei vom lokalen Zustand $(\boldsymbol{l}, \boldsymbol{c})$ des Gelenks und die Richtung von gelenksspezifischen Richtungen wie der (momentanen) Drehachse oder dem Verbindungsvektor ${}_{\mathrm{K}_0}\boldsymbol{r}_{NM}$ ab.

Die Resultierenden äusseren Kräfte und Momente eines jeden Starrkörpers werden aktualisiert in Schleifen über alle Körper $[\mathtt{b}_0, \mathtt{b}_1, \dots]$ und deren freie Kraftelemente $[\mathtt{f}_0, \mathtt{f}_1, \dots]$ nach Algorithmus 2.8 und über alle Gelenke $[\mathtt{j}_0, \mathtt{j}_1, \dots]$ gemäss Algorithmus 2.9. Man beachte, dass die Funktion `JointForceAndMoment` in Algorithmus 2.9 für Gelenke ohne Kraftelement Nullvektoren zurück liefern soll. Man kann dies so implementieren oder im Treiber eine separate Liste `forcedJoints` erstellen, die ausschliesslich Gelenke mit Kraftelementen enthält, und nur über diese iterieren.

Algorithmus 2.8

```
# this function is a member method of the Solver class.
# 'self' therefore refers to the solver instance.
def updateExternalForces(self):
  for b in self.rigidBodies:
    for f in b.forces:
      [I_F_S, K_M_S] = \
        f.ForceAndMoment(b.mass, b.position,
                         b.orientation,b.linearVelocity)
      b.externalForce += I_F_S
      b.externalMoment += K_M_S
```

Algorithmus 2.9

```
# this function is a member method of the Solver class.
# 'self' therefore refers to the solver instance.
def updateJointForces(self,q,k,u,m):
  for j in self.joints:
    [l,c] = j.updateLocalState(q,k,u,m)
    [K0_F_M, K0_M_0, K1_F_N, K1_M_1] = \
      j.JointForceAndMoment(l,c)
    parent.externalForce += parent.A_IK*K0_F_M
    parent.externalMoment += (K0_M_0 +
      cross(parentLocation, K0_F_M))
    child.externalForce += child.A_IK*K1_F_N
    child.externalMoment += (K1_M_1 +
      cross(childLocation, K1_F_N))

def cross(v1,v2):
  # vector product of two three dimensional vectors:
  return Vector3d(v1.y*v2.z-v1.z*v2.y,
    v1.z*v2.x-v1.x*v2.z, v1.x*v2.y-v1.y*v2.x)
```

2.5 Bewegungsgleichungen des glatten Systems

Nach der Aktualisierung der kinematischen Grössen und der äusseren Kräfte aller Starrkörper, werden in diesem Abschnitt die Bewegungsgleichungen für das Mehrkörpersystem ohne Berücksichtigung der Kontakte numerisch aufgestellt. Ohne die Kontakte handelt es sich um eine glatte Bewegung und alle Zustände sind damit differenzierbar. Im nächsten Abschnitt werden die Kontakte nachträglich als einseitige Bindungen hinzugefügt und auf eine allgemeinere Massdifferentialinklusion übergegangen, die auch nicht glatte (Stoss-)Bewegungen beschreibt.

Jeder Starrkörper i eines Mehrkörpersystems setzt sich ursprünglich aus einer Menge $\mathscr{S}_i$ von materiellen Punkten zusammen. Die Menge $\mathscr{S}_i$ jedes Körpers ist in einem (eigenen) dreidimensionalen Vektorraum mit Ursprung G_i und Basis $(\boldsymbol{e}_x^{\mathrm{K}_i}, \boldsymbol{e}_y^{\mathrm{K}_i}, \boldsymbol{e}_z^{\mathrm{K}_i})$ eingebettet, was in Abbildung 2.3 am Beispiel von zwei Körpern gezeigt ist. Diese Einbettung dient als Referenz für jeden Körper. Werden für eine Simulation einzelne Körper in CAD- oder 3D-Modellierungsprogrammen gezeichnet, können praktischerweise die Koordinatensysteme dieser Programme als die eben beschriebenen Referenzräume gewählt werden. In diesem Referenzsystem werden die Punkte definiert an denen später Gelenke und/oder Kraftelemente angreifen sollen. Jeder materielle Punkt x ist mit einem Masseninkrement $\mathrm{d}m$ behaftet. Mit der Integration über alle materiellen Punkte eines Starrkörpers ergeben sich die folgenden Eigenschaften für einen Starrkörper, vgl. [25, 64]. Die *Masse* eines Starrkörpers i ist definiert als die Summe über die Masseninkremente aller seiner materiellen

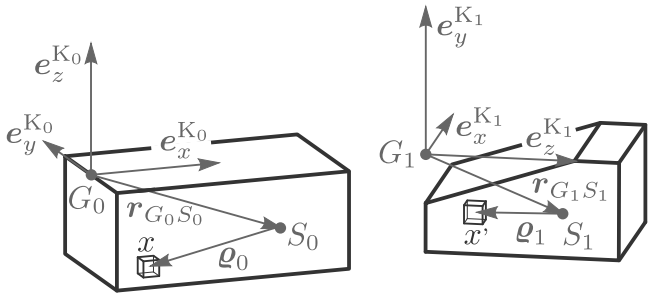

Abbildung 2.3: Referenz eines Mehrkörpersystems bestehend aus mehreren Körpern i mit Schwerpunkten S_i. Jeder materielle Punkt $x \in \mathscr{S}_i$ wird über einen Vektor $\boldsymbol{\varrho}_i(x)$ adressiert ausgehend vom Schwerpunkt S_i desjenigen Körpers in dem x liegt.

Punkte, also

$$m_i = \int_{\mathscr{S}_i} \mathrm{d}m.$$

Im eingeführten Raum kann jeder materielle Punkt $x \in \mathscr{S}_i$ über einen Vektor $\boldsymbol{r}_{G_iS_i} + \boldsymbol{\varrho}_i(x)$ adressiert werden, wobei S_i den *Massenschwerpunkt* bezeichnet für den

$$\int_{\mathscr{S}_i} \boldsymbol{r}_{G_iS_i} + \boldsymbol{\varrho}_i(x)\,\mathrm{d}m = m_i \boldsymbol{r}_{G_iS_i}$$

gilt. Weiter ist mit

$$\boldsymbol{\Theta}_{S_i} = \int_{\mathscr{S}_i} \tilde{\boldsymbol{\varrho}}_i^{\mathsf{T}} \tilde{\boldsymbol{\varrho}}_i \,\mathrm{d}m$$

der *Trägheitstensor* eines Starrkörpers bezüglich seinem Schwerpunkt definiert. Hierbei bezeichnet $\tilde{\boldsymbol{a}}$ die zum Kreuzprodukt gehörende schiefsymmetrische Matrix mit der $\tilde{\boldsymbol{a}}\boldsymbol{x} = \boldsymbol{a} \times \boldsymbol{x}$. Durch Auswerten dieser Integrale oder modellierendes Festlegen sind die Schwerpunktsposition $\boldsymbol{r}_{G_iS_i}$, die Masse m_i und der Trägheitstensor $\boldsymbol{\Theta}_{S_i}$ bezüglich dem Schwerpunkt für jeden Körper des Systems bekannt.

Ein Inertialsystem $O, (\boldsymbol{e}_x^{\mathrm{I}}, \boldsymbol{e}_y^{\mathrm{I}}, \boldsymbol{e}_z^{\mathrm{I}})$ bildet den euklidischen Anschauungsraum und jeder Körper wird aus seinem Referenzraum zuerst in eine Referenzlage im Inertialsystem abgebildet. Diese Abbildung soll den Körper *unverändert* lassen und deshalb die Abstände zwischen allen Punkten erhalten. Eine solche Abbildung wird auch Isometrie genannt. Diese Anforderung entspricht der *Starrkörperbedingung* und aus dieser folgt nach [77], dass einzig Translationen und Rotationen als Abbildungen erlaubt sind. Verschiebungen und Verdrehungen bilden deshalb die sogenannten *Starrkörperabbildungen*. Zusammen mit der Einbettung der materiellen Punkte in den Referenzraum des Körpers, wird jeder materielle Punkt $x \in \mathscr{S}_i$ über einen Vektor $\boldsymbol{x}_i$ im Inertialsystem addressiert und für jeden Starrkörper lässt sich diese Abbildung schreiben als

$$\begin{aligned} \boldsymbol{x}_i \colon \mathscr{S}_i &\longrightarrow (O, (\boldsymbol{e}_x^{\mathrm{I}}, \boldsymbol{e}_y^{\mathrm{I}}, \boldsymbol{e}_z^{\mathrm{I}})) \\ x &\longmapsto \boldsymbol{x}_i(x) = \boldsymbol{r}_{OG_i}^{\circ} + \mathbf{R}_{\mathrm{IK}_i}^{\circ} \left(\boldsymbol{r}_{G_iS_i} + \boldsymbol{\varrho}_i(x) \right). \end{aligned} \tag{2.14}$$

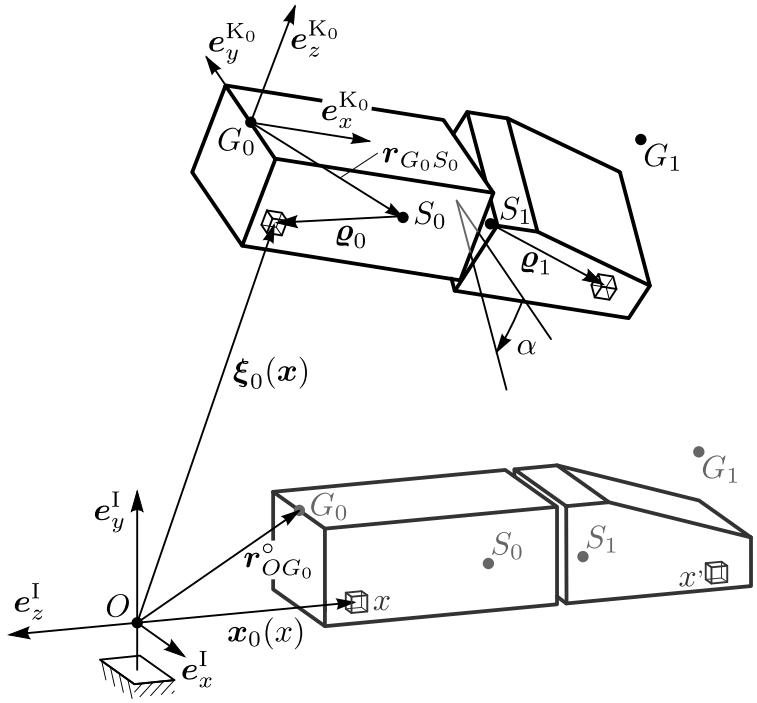

Abbildung 2.4: Ins Inertialsystem eingebettete Referenzlage eines Mehrkörpersystems (unten) und deren Abbildung in eine verschobene Lage (oben).

Mit diesen Abbildungen ist die Referenzlage des Mehrkörpersystems im Inertialsystem bestimmt und diese Referenzlage ist in Abbildung 2.4 unten gezeichnet.

Aus der Referenzlage wird das Mehrkörpersystem in Abhängigkeit der generalisierten Lagen $(\boldsymbol{q}, \boldsymbol{m})$ in eine verschobene Konfigurationen ausgelenkt, vgl. Abbildung 2.4 oben. Dazu wird jeder materielle Punkt eines Körpers i mit einer Abbildung $\boldsymbol{\xi}_i$ verschoben. Im Falle von Starrkörpern kommt dafür ausschliesslich eine Starrkörperabbildung in Frage, weshalb sich die Verschiebungsabbildung für jeden Körper mit

$$\begin{aligned} \boldsymbol{\xi}_i \colon (O, (\boldsymbol{e}_x^{\mathrm{I}}, \boldsymbol{e}_y^{\mathrm{I}}, \boldsymbol{e}_z^{\mathrm{I}})) &\longrightarrow (O, (\boldsymbol{e}_x^{\mathrm{I}}, \boldsymbol{e}_y^{\mathrm{I}}, \boldsymbol{e}_z^{\mathrm{I}})) \\ \boldsymbol{x}_i &\longmapsto \boldsymbol{\xi}_i(\boldsymbol{x}_i) = \boldsymbol{r}_{OG_i} + \mathbf{R}_{\mathrm{K}_i\mathrm{I}} \left(\boldsymbol{x}_i - \boldsymbol{r}^\circ_{OG_i} \right) \end{aligned} \tag{2.15}$$

schreiben lässt.

Die Geschwindigkeit $\dot{\boldsymbol{\xi}}_i$ eines Starrkörperpartikels schreibt sich nach zeitlicher Ableitung mit

$$\begin{aligned} \dot{\boldsymbol{\xi}}_i &= \dot{\boldsymbol{r}}_{OG_i} + \dot{\mathbf{R}}_{\mathrm{K}_i\mathrm{I}} \left(\boldsymbol{x}_i - \boldsymbol{r}^\circ_{OG_i} \right) \\ &= \dot{\boldsymbol{r}}_{OG_i} + \dot{\mathbf{R}}_{\mathrm{K}_i\mathrm{I}} \mathbf{R}_{\mathrm{K}_i\mathrm{I}}^{\mathsf{T}} \mathbf{R}_{\mathrm{K}_i\mathrm{I}} \left(\boldsymbol{x}_i - \boldsymbol{r}^\circ_{OG_i} \right) \\ &= \dot{\boldsymbol{r}}_{OG_i} + \tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i} \mathbf{R}_{\mathrm{K}_i\mathrm{I}} \left(\boldsymbol{x}_i - \boldsymbol{r}^\circ_{OG_i} \right) \\ &= \dot{\boldsymbol{r}}_{OG_i} + \tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i} \left(\boldsymbol{r}_{G_iS_i} + \boldsymbol{\varrho}_i \right) \\ &= \boldsymbol{v}_{S_i} + \tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i} \boldsymbol{\varrho}_i. \end{aligned} \tag{2.16}$$

In der ersten Zeile ist bereits berücksichtigt, dass es sich bei $\boldsymbol{x}_i - \boldsymbol{r}^{\circ}_{OG_i}$ um einen ein konstanten Vektor (in der Referenzlage) handelt. In der zweiten Zeile wird eine Einheitsmatrix $\mathbf{I} = \mathbf{R}_{\mathrm{K}_i\mathrm{I}}^{\mathsf{T}}\mathbf{R}_{\mathrm{K}_i\mathrm{I}}$ zwischengeschaltet und in der dritten Zeile die Definition der Winkelgeschwindigkeit $\dot{\mathbf{R}}_{\mathrm{K}_i\mathrm{I}}\mathbf{R}_{\mathrm{K}_i\mathrm{I}}^{\mathsf{T}} = \tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i}$ eingeführt. In der vierten Zeile wird (2.14) verwendet und die Tatsache, dass die Inversion einer Rotation deren Transposition entspricht, also $\mathbf{R}_{\mathrm{IK}_i} = \mathbf{R}_{\mathrm{K}_i\mathrm{I}}^{-1} = \mathbf{R}_{\mathrm{K}_i\mathrm{I}}^{\mathsf{T}}$. Schliesslich ist

$$\begin{aligned}
\tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i}\,\boldsymbol{r}_{G_iS_i} &= \dot{\mathbf{R}}_{\mathrm{K}_i\mathrm{I}}\,\mathbf{R}_{\mathrm{K}_i\mathrm{I}}^{\mathsf{T}}\,\boldsymbol{r}_{G_iS_i} \\
&= \dot{\mathbf{R}}_{\mathrm{K}_i\mathrm{I}}\,\mathbf{R}_{\mathrm{IK}_i}\,\boldsymbol{r}_{G_iS_i} \\
&= (\mathbf{R}_{\mathrm{K}_i\mathrm{I}}\,\mathbf{R}_{\mathrm{IK}_i}\,\boldsymbol{r}_{G_iS_i})^{\cdot},\ \text{da } \mathbf{R}_{\mathrm{IK}_i}\boldsymbol{r}_{G_iS_i} \text{ konstant (Referenzlage)} \\
&= \dot{\boldsymbol{r}}_{G_iS_i}
\end{aligned} \tag{2.17}$$

und mit (2.5) deshalb $\dot{\boldsymbol{r}}_{OG_i} + \tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i}\boldsymbol{r}_{G_iS_i} = \dot{\boldsymbol{r}}_{OS_i} = \boldsymbol{v}_{S_i}$. Man beachte, dass mit (2.14) die Verschiebungsabbildung (2.15) zu $\boldsymbol{\xi}(\boldsymbol{x}) = \boldsymbol{r}_{OG_i} + \boldsymbol{r}_{G_iS_i} + \boldsymbol{\varrho}_i = \boldsymbol{r}_{OS_i} + \boldsymbol{\varrho}_i$ wird, was man leicht auch aus Abbildung 2.4 entnehmen kann. Die Geschwindigkeit $\dot{\boldsymbol{\xi}} = \boldsymbol{v}_{S_i} + \tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i}\,\boldsymbol{\varrho}_i$ folgte dann mit (2.17) in entgegengesetzter Argumentationsrichtung. Mit abermaliger Differenzierung nach der Zeit erhält man die Partikelbeschleunigung

$$\begin{aligned}
\ddot{\boldsymbol{\xi}}_i &= \dot{\boldsymbol{v}}_{S_i} + \dot{\boldsymbol{\omega}}_{\mathrm{IK}_i} \times \boldsymbol{\varrho}_i + \boldsymbol{\omega}_{\mathrm{IK}_i} \times \dot{\boldsymbol{\varrho}}_i \\
&= \dot{\boldsymbol{v}}_{S_i} - \boldsymbol{\varrho}_i \times \dot{\boldsymbol{\omega}}_{\mathrm{IK}_i} + \boldsymbol{\omega}_{\mathrm{IK}_i} \times (\boldsymbol{\omega}_{\mathrm{IK}_i} \times \boldsymbol{\varrho}_i).
\end{aligned}$$

Und analog zur Geschwindigkeit (2.16) schreibt sich die virtuelle Geschwindigkeit eines Starrkörperpartikels mit

$$\underline{\delta}\dot{\boldsymbol{\xi}}_i = \underline{\delta}\boldsymbol{v}_{S_i} + \underline{\delta}\tilde{\boldsymbol{\omega}}_i\boldsymbol{\varrho}_i,$$

wobei $\underline{\delta}\boldsymbol{v}_{S_i}$ einer virtuellen Verschiebungsgeschwindigkeit entspricht und $\underline{\delta}\tilde{\boldsymbol{\omega}}_i$ eine virtuelle Rotationsgeschwindigkeit bezeichnet, vgl. [85].

Damit kann die virtuelle Leistung, z.B. [12], des Mehrkörpersystems

$$\begin{aligned}
\underline{\delta}P &= \sum_i \int_{\mathscr{S}_i} \underline{\delta}\dot{\boldsymbol{\xi}}_i^{\mathsf{T}}(\ddot{\boldsymbol{\xi}}_i\,\mathrm{d}m - \mathrm{d}\boldsymbol{F}) \\
&= \sum_i \begin{pmatrix} \underline{\delta}\boldsymbol{v}_{S_i} \\ \underline{\delta}\boldsymbol{\omega}_i \end{pmatrix}^{\mathsf{T}} \begin{bmatrix} m_i\dot{\boldsymbol{v}}_{S_i} - \boldsymbol{F}_{S_i} \\ \boldsymbol{\Theta}_{S_i}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_i} + \tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i}\boldsymbol{\Theta}_{S_i}\boldsymbol{\omega}_{\mathrm{IK}_i} - \boldsymbol{M}_{S_i} \end{bmatrix}
\end{aligned}$$

ausgewertet werden, was in den Kapiteln 2 und 3 von [25] detailliert ausgeführt ist. Das Prinzip der virtuellen Leistung, z.B. [85], besagt schliesslich, dass die virtuelle Leistung unter mit den Bindungen verträglichen virtuellen Geschwindigkeiten $\underline{\delta}\dot{\boldsymbol{\xi}}_{i,vertr.}$ verschwindet. Die *verträglichen* virtuellen Geschwindigkeiten eines Starrkörpers sind genau jene welche durch die generalisierten virtuellen Geschwindigkeiten induziert werden. Es ist also mit den Gleichungen (2.11) und (2.12) gemäss [12]

$$\underline{\delta}\dot{\boldsymbol{\xi}}_{i,vertr.} = \underline{\delta}\boldsymbol{v}_{S_i,vertr.} - \tilde{\boldsymbol{\varrho}}_i\,\underline{\delta}\boldsymbol{\omega}_{i,vertr.} = \mathbf{J}_{S_i}\,\underline{\delta}\boldsymbol{u} - \tilde{\boldsymbol{\varrho}}_i\mathbf{J}_{R_i}\,\underline{\delta}\boldsymbol{u},$$

womit das Prinzip der virtuellen Leistung zu

$$\begin{aligned} 0 &= \underline{\delta} P \quad \forall\, \underline{\delta}\dot{\boldsymbol{\xi}}_{i,vertr.} \\ &= \underline{\delta}\boldsymbol{u}^{\mathsf{T}} \sum_i \begin{pmatrix} \mathbf{J}_{S_i} \\ \mathbf{J}_{R_i} \end{pmatrix}^{\mathsf{T}} \begin{bmatrix} m_i \dot{\boldsymbol{v}}_{S_i} - \boldsymbol{F}_{S_i} \\ \boldsymbol{\Theta}_{S_i}\dot{\boldsymbol{\omega}}_{\mathrm{IK}_i} + \tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i}\boldsymbol{\Theta}_{S_i}\boldsymbol{\omega}_{\mathrm{IK}_i} - \boldsymbol{M}_{S_i} \end{bmatrix} \quad \forall\, \underline{\delta}\boldsymbol{u} \end{aligned} \tag{2.18}$$

wird. Geht man nun auf eine Koordinatendarstellung der Vektoren über und setzt Gleichung (2.13) ein, ergibt sich die Bewegungsgleichung für das glatte System, d.h. das System ohne Berücksichtigung der Kontakte

$$\begin{aligned} \mathbf{0} = \sum_i \Big[&({}_{\mathrm{I}}\mathbf{J}_{S_i}^{\mathsf{T}} m_{i\,\mathrm{I}}\mathbf{J}_{S_i} + {}_{\mathrm{K}_i}\mathbf{J}_{R_i\mathrm{K}_i}^{\mathsf{T}} \boldsymbol{\Theta}_{S_i\mathrm{K}_i}\mathbf{J}_{R_i})\dot{\boldsymbol{u}} \\ &+ \Big({}_{\mathrm{I}}\mathbf{J}_{S_i}^{\mathsf{T}} m_{i\,\mathrm{I}}\boldsymbol{\kappa}_{S_i} + {}_{\mathrm{K}_i}\mathbf{J}_{R_i}^{\mathsf{T}}({}_{\mathrm{K}_i}\boldsymbol{\Theta}_{S_i\mathrm{K}_i}\boldsymbol{\kappa}_{R_i} + {}_{\mathrm{K}_i}\tilde{\boldsymbol{\omega}}_{\mathrm{IK}_i\mathrm{K}_i}\boldsymbol{\Theta}_{S_i\mathrm{K}_i}\boldsymbol{\omega}_{\mathrm{IK}_i})\Big) \\ &-({}_{\mathrm{I}}\mathbf{J}_{S_i\,\mathrm{I}}^{\mathsf{T}}\boldsymbol{F}_{S_i} + {}_{\mathrm{K}_i}\mathbf{J}_{R_i\mathrm{K}_i}^{\mathsf{T}}\boldsymbol{M}_{S_i})\Big] \\ = \mathbf{M}(\boldsymbol{q},\boldsymbol{k})\dot{\boldsymbol{u}} &- \boldsymbol{h}(\boldsymbol{q},\boldsymbol{k},\boldsymbol{u},\boldsymbol{m}) \end{aligned} \tag{2.19}$$

in Minimalgeschwindigkeiten bzw. deren zeitlicher Änderung $\dot{\boldsymbol{u}}$. Zusammen mit der Kinematikgleichung (2.10) ist damit eine vollständige Beschreibung des Bewegungsablaufs gegeben. Man beachte, dass die einzelnen Beiträge (Summanden) zur virtuellen Leistung in Gleichung (2.18) in verschiedenen Koordinatensystemen ausgewertet werden dürfen, da es sich um eine invariante und damit koordinatenunabhängige, skalare Grösse handelt. Der Treiber assembliert die Massenmatrix $\mathbf{M}$ und den Vektor $\boldsymbol{h}$ nach Algorithmus 2.10 anhand der zuvor aktualisierten kinematischen Zustände der Körper.

Algorithmus 2.10

```
# call to assemble 'massMatrix' and 'forceVector' in the
# solver's process of solving a time step.
# 'massMatrix' is supposed to be a mutable matrix of size
# countVelocities x countVelocities. It is a temporary object
# hold by the solver instance.
# 'forceVector' is supposed to be a mutable vector of size
# countVelocities. It is a temporary object hold by the
# solver instance.
# 'solver' is supposed to be a Solver object.
solver.assembleMassMatrixAndForceVector(massMatrix, forceVector)

# these functions are member methods of the Solver class.
# 'self' therefore refers to the solver instance.
def assembleMassMatrixAndForceVector(self, massMatrix, forceVector):
  for body in self.bodies:
    self.addToMassMatrix(body,massMatrix)
    self.addToForceVector(body,forceVector)

def addToMassMatrix(self,body,M):
  for n in range(self.velocitiesCount):
    JSnMass = body.translationJacobi[:,n] * body.mass
    JRnInertia = body.rotationJacobi[:,n] * body.inertia
    for nn in range(self.velocitiesCount):
      M[n,nn] += (JSnMass * body.translationJacobi[:,nn] +
                  JRnInertia * body.rotationJacobi[:,nn])

def addToForceVector(self,body,h):
  linPart = (- body.mass * body.linearPseudoAcceleration +
             body.externalForce)
  rotPart = (-(body.inertia*body.angularPseudoAcceleration +
               cross(body.angularVelocity,
                     body.inertia * body.angularVelocity)) +
             body.externalMoment)
  for n in range(velocitiesCount):
    h_n = (body.translationJacobi[:,n] * linPart +
           body.rotationJacobi[:,n] * rotPart)
    h[n] += h_n
```

2.6 Einführung von Kontakten und Reibung

Nach der Berechnung der Massenmatrix $\mathbf{M}$ und dem Kraftvektor $\boldsymbol{h}$ werden die Bewegungsgleichungen (2.19) in diesem Abschnitt erweitert, so dass auch reibungsbehaftete Kontakte berücksichtigt werden können. Bei Kontakten handelt es sich um einseitige Bindungen mit denen Bindungskräfte einhergehen. Ein Kontakt wird als Kraftelement betrachtet, das zwischen den zwei Kontaktpunkten als Kraftangriffspunkte liegt [25]. Das Kraftgesetz ist auf Lageebene in Abhängigkeit der Kontaktöffnung, also dem Abstand der zwei Kontaktpunkte formuliert. Auch bei der Reibung handelt es sich um ein Kraftelement, das zwischen den Kontaktpunkten wirkt. Das zugehörige Kraftgesetz formuliert sich in Abhängigkeit der Relativgeschwindigkeit. Sowohl für Reibung, wie für Kontakte lassen sich die Kraftgesetze (auf Geschwindigkeitsebene) als Normalkegelinklusionen

$$\boldsymbol{\gamma} \in \mathscr{N}_{\mathscr{C}}(-\boldsymbol{\lambda}) := \{\boldsymbol{y} \mid \boldsymbol{y}^{\mathsf{T}}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^*) \leq 0, \quad \forall\, -\boldsymbol{\lambda}^* \in \mathscr{C}\} \tag{2.20}$$

formulieren, nach denen die jeweilige Kraft $\boldsymbol{\lambda}$ in Relation zur entsprechenden Relativgeschwindigkeit $\boldsymbol{\gamma}$ liegt. Die konvexe Menge $\mathscr{C}$ kann hierbei als Kraftreservoir, d.h. die Menge aller möglichen Kräfte interpretiert werden. Für eine eingehende Behandlung von Normalkegelinklusionen und deren Anwendung in der nichtglatten Mechanik sind [25, 55, 79] empfohlen.

Für die Berücksichtigung von Kontakten und Reibung in den Bewegungsgleichungen sind der Reihe nach folgende Schritte notwendig. Zuerst werden in einer Kollisionsdetektion alle geschlossenen Kontakte und deren Kontaktpunkte auf den Körpern bestimmt. Dann werden die Kontaktkräfte als äussere Kräfte der Körper behandelt, womit sie gleich wie die Resultierenden der äusseren Kräfte $\boldsymbol{F}_{S_i}$ und äusseren Momente $\boldsymbol{M}_{S_i}$ in die Bewegungsgleichungen (2.19) eingehen. Schliesslich werden die zugehörigen Kraftgesetze auf Geschwindigkeitsebene in der Form (2.20) angegeben und die dafür notwendige Relativkinematik hergeleitet. Vorerst wird weiter eine stossfreie Bewegung betrachtet, d.h. zu den beschriebenen Zeitpunkten sind die Kontakte entweder geschlossen, offen oder im Begriff zu öffnen. Abschnitt 2.6.5 behandelt dann Stösse, d.h. Zeitpunkte zu denen sich Kontakte schliessen. Abschnitt 2.6.6 führt schliesslich die Notation der Massdifferentialinklusion ein, welche für alle Zeiten gültig ist.

2.6.1 Kollisionsdetektion

In der Methode `findCollisionPoints` sucht der Treiber nach geschlossenen Kontakten. Dies geschieht mit einem Bounding-Volume-Hierarchy Suchalgorithmus [63]. Darin wird zuerst um die vordefinierten Kontaktpunkte $\boldsymbol{r}_{SD_c}$ und die dazugehörigen Segmente $\boldsymbol{d}_c$ aller Segmentkontakte $\{c\}$ eine umhüllende Kugel konstruiert. Alle innerhalb der Umhüllenden liegenden Dreiecke der Kollisionsgeometrie, in einer Vordetektion bestimmt, werden auf Schnittpunkte mit allen Kontaktsegmenten getestet. Man vergleiche dazu Abbildung 2.1. Bei allen geschlossenen Kontakten wird der `collisionPoint`, bestehend aus der aktuellen Position ${}_I\boldsymbol{r}_{OD'}$ des Kontaktpunktes auf dem Körper und der Normalen des beteiligten Dreiecks der Kollisionsgeometrie, aktualisiert. Schliesslich retourniert die

Methode `findCollisionPoints` die Menge $\{k\}$ aller geschlossenen Kontakte als Liste mit Referenzen auf Kontaktobjekte, siehe Tabelle 2.4.

Mit der Position und der Orientierung des Starrkörpers `b`, auf dem ein Segmentkontakt definiert ist, berechnet sich der Kontaktpunkt relativ zum Körperschwerpunkt in körperfesten Koordinaten mit

$$ {}_{\mathrm{K}}\boldsymbol{r}_{SD'} = \mathbf{A}_{\mathrm{IK}}^{\mathsf{T}}[\mathsf{b}]({}_{\mathrm{I}}\boldsymbol{r}_{OD'} - {}_{\mathrm{I}}\boldsymbol{r}_{OS}[\mathsf{b}]). \tag{2.21} $$

Über die Funktion `getTangents` des Kontaktobjektes erhält man in Abhängigkeit der Normalen ${}_{\mathrm{K}}\boldsymbol{n}^{\mathrm{C}} = \mathbf{A}_{\mathrm{IK}}^{\mathsf{T}}[\mathsf{b}]{}_{\mathrm{I}}\boldsymbol{n}^{\mathrm{C}}$ die Tangential- bzw. Bitangentialvektoren ${}_{\mathrm{K}}\boldsymbol{t}^{\mathrm{C}}$ und ${}_{\mathrm{K}}\boldsymbol{b}^{\mathrm{C}}$, welche dem Treiber mit ${}_{\mathrm{I}}\boldsymbol{t}^{\mathrm{C}} = \mathbf{A}_{\mathrm{IK}}[\mathsf{b}]_{\mathrm{K}}\boldsymbol{t}^{\mathrm{C}}$ und ${}_{\mathrm{I}}\boldsymbol{b}^{\mathrm{C}} = \mathbf{A}_{\mathrm{IK}}[\mathsf{b}]_{\mathrm{K}}\boldsymbol{b}^{\mathrm{C}}$ auch in Inertialkoordinaten zur Verfügung stehen. Damit ist die Kollisionsdetektion abgeschlossen und anhand der berechneten Grössen werden im nächsten Abschnitt die Relativgeschwindigkeiten ausgerechnet.

Es bleibt zu sagen, dass die Kollisionsdetektion für die in dieser Arbeit verwendeten Segmentkontakte bereits umfangreich, aber vergleichsweise einfach ist. Mit anderen Kontaktmodellen, z.B. wenn beliebige Dreiecksgitter gegeneinander kollidieren können sollen, gestaltet sich das Finden der Kontaktpunkte komplexer. Kollisionsdetektion ist ein oft, u.a. in der Computergrafik, vorkommendes Problem und die effiziente Lösung solcher Aufgaben bildet einen eigenen Forschungszweig. Durch diverse Optimierungen kann viel Rechenzeit gewonnen werden. Der Kürze halber und weil der Autor an der Implementierung der Kollisionsdetektion nicht beteiligt war, sondern Bibliotheken von Möller [63] verwendet werden, wird auf eine detailliertere Darstellung der Kontaktpunktfindung verzichtet. Es sei stattdessen auf [16] als Referenz für auf Echtzeitfähigkeit hin optimierte Algorithmen verwiesen.

2.6.2 Kontaktkraft und Reibung als äussere Kraft am Körper

Für die folgenden Erläuterungen betrachte man nochmals Abbildung 2.1. Die Kontaktkraft

$$ {}_{\mathrm{I}}\boldsymbol{F}_{\lambda_n} = {}_{\mathrm{I}}\boldsymbol{n}^{\mathrm{C}}\lambda_n $$

im dreidimensionalen Raum zeigt in Richtung der Kontaktnormalen ${}_{\mathrm{I}}\boldsymbol{n}^{\mathrm{C}}$ und ist vom Betrag λ_n. Die Reibkraft

$$ {}_{\mathrm{I}}\boldsymbol{F}_{\boldsymbol{\lambda}_f} = {}_{\mathrm{I}}\boldsymbol{t}^{\mathrm{C}}\lambda_t + {}_{\mathrm{I}}\boldsymbol{b}^{\mathrm{C}}\lambda_b = ({}_{\mathrm{I}}\boldsymbol{t}^{\mathrm{C}}, {}_{\mathrm{I}}\boldsymbol{b}^{\mathrm{C}})\boldsymbol{\lambda}_f $$

liegt in der Tangentialebene und setzt sich zusammen aus zwei Komponenten. Um den Betrag λ_n bzw. die Komponenten $(\lambda_t, \lambda_b)^{\mathsf{T}} = \boldsymbol{\lambda}_f$ kümmert sich Unterabschnitt 2.6.3. Hier interessieren vorab die Richtungen.

Die Kräfte ${}_{\mathrm{I}}\boldsymbol{F}_{\lambda_n}$ und ${}_{\mathrm{I}}\boldsymbol{F}_{\boldsymbol{\lambda}_f}$ sind nachträglich hinzugefügte Bestandteile der äusseren Kraft ${}_{\mathrm{I}}\boldsymbol{F}_{S_i}$ eines Körpers und greifen im Punkt ${}_{\mathrm{K}}\boldsymbol{r}_{SD'}$ am Körper an. Der Angriffspunkt resultiert aus der Kollisionsdetektion nach Gleichung (2.21). Diese Kräfte induzieren bezüglich dem Schwerpunkt des Körpers ein Moment

$$ \begin{aligned} {}_{\mathrm{K}}\boldsymbol{M}_{\boldsymbol{\lambda},S} \mathrel{+}&= {}_{\mathrm{K}}\tilde{\boldsymbol{r}}_{SD'}\, \mathbf{A}_{\mathrm{KI}}({}_{\mathrm{I}}\boldsymbol{F}_{\lambda_n} + {}_{\mathrm{I}}\boldsymbol{F}_{\boldsymbol{\lambda}_f}) \\ &= {}_{\mathrm{K}}\tilde{\boldsymbol{r}}_{SD'}\,({}_{\mathrm{K}}\boldsymbol{n}^{\mathrm{C}}\lambda_n + {}_{\mathrm{K}}\boldsymbol{t}^{\mathrm{C}}\lambda_t + {}_{\mathrm{K}}\boldsymbol{b}^{\mathrm{C}}\lambda_b), \end{aligned} $$

das ein nachträglich hinzugefügter Bestandteil des äusseren Moments ${}_{\mathrm{K}}\boldsymbol{M}_S$ des Körpers darstellt. Es sei hier bemerkt, dass allfällige Reibmomente, wie sie bei Coulomb-Contensou oder Rollreibung [55] vorkommen, zu diesen induzierten Momenten addiert werden müssten. In dieser Arbeit kommen aber keine solchen freien Reibmomente vor.

Wie man Gleichung (2.19) entnimmt, gehen die Kontakt- und Reibkräfte als Bestandteile der äusseren Kraft und des äusseren Moments eines Körpers mit

$$\begin{aligned} &- \left({}_{\mathrm{I}}\mathbf{J}_{S_i}^{\mathsf{T}}({}_{\mathrm{I}}\boldsymbol{n}^{\mathrm{C}}\lambda_n + ({}_{\mathrm{I}}\boldsymbol{t}^{\mathrm{C}}, {}_{\mathrm{I}}\boldsymbol{b}^{\mathrm{C}})\boldsymbol{\lambda}_f) + {}_{\mathrm{K}_i}\mathbf{J}_{R_i}^{\mathsf{T}}({}_{\mathrm{K}}\tilde{\boldsymbol{r}}_{SD'}\,({}_{\mathrm{K}}\boldsymbol{n}^{\mathrm{C}}\lambda_n + ({}_{\mathrm{K}}\boldsymbol{t}^{\mathrm{C}}, {}_{\mathrm{K}}\boldsymbol{b}^{\mathrm{C}})\boldsymbol{\lambda}_f))\right) \\ &= - ({}_{\mathrm{I}}\mathbf{J}_{S_i}^{\mathsf{T}}{}_{\mathrm{I}}\boldsymbol{n}^{\mathrm{C}} + {}_{\mathrm{K}_i}\mathbf{J}_{R_i}^{\mathsf{T}}{}_{\mathrm{K}}\tilde{\boldsymbol{r}}_{SD'\mathrm{K}}\boldsymbol{n}^{\mathrm{C}})\lambda_n + \left({}_{\mathrm{I}}\mathbf{J}_{S_i}^{\mathsf{T}}({}_{\mathrm{I}}\boldsymbol{t}^{\mathrm{C}}, {}_{\mathrm{I}}\boldsymbol{b}^{\mathrm{C}}) + {}_{\mathrm{K}_i}\mathbf{J}_{R_i}^{\mathsf{T}}({}_{\mathrm{K}}\tilde{\boldsymbol{r}}_{SD'}({}_{\mathrm{K}}\boldsymbol{t}^{\mathrm{C}}, {}_{\mathrm{K}}\boldsymbol{b}^{\mathrm{C}})\right)\boldsymbol{\lambda}_f \\ &= - (\boldsymbol{w}_n\lambda_n + \mathbf{W}_f\boldsymbol{\lambda}_f) \end{aligned} \tag{2.22}$$

in die Bewegungsgleichungen ein. Hierbei werden die Grössen $\boldsymbol{w}_n$ und $\mathbf{W}_f$ generalisierte Kraftrichtungen genannt. Mit der Menge der geschlossenen Kontakte $\{k\}$ haben die Bewegungsgleichungen die Form

$$\mathbf{0} = \mathbf{M}\dot{\boldsymbol{u}} - \boldsymbol{h} - \sum_k (\boldsymbol{w}_n\lambda_n + \mathbf{W}_f\boldsymbol{\lambda}_f)_k, \quad \text{f.ü.}, \tag{2.23}$$

wobei f.ü. für *fast überall* steht. Ausgenommen sind die einzelnen Zeitpunkte zu denn sich Kontakte schliessen (Stoss) und zu denen Beschleunigungssprünge (Gleit-Haftübergang) stattfinden. An solchen Stellen existiert $\dot{\boldsymbol{u}}$ nicht und man muss dort die Gleichung im Sinne von links- und rechtsseitigen Grenzwerten $\dot{\boldsymbol{u}}^-$ bzw. $\dot{\boldsymbol{u}}^+$ interpretieren [25, 55].

2.6.3 Normalkontakt- und Reibkraftgesetz

Wie bereits einleitend erwähnt, folgen sowohl die Kontaktkraft λ_n in Normalrichtung, als auch die Reibkraft $\boldsymbol{\lambda}_f$ in Tangentialrichtung Normalkegelinklusionen. Der Normalkontakt ist auf Geschwindigkeitsebene mit

$$\begin{aligned} g_n > 0: &\quad \lambda_n = 0 \\ g_n = 0: &\quad \gamma_n \in \mathscr{N}_{\mathbb{R}_0^-}(-\lambda_n) \end{aligned} \tag{2.24}$$

formuliert. Abbildung 2.5 illustriert dieses mengenwertige Kraftgesetz. Für einen offenen Kontakt ($g_n > 0$) ist keine Kraft vorhanden. Bei geschlossenem Kontakt entscheidet sich anhand der Öffnungsgeschwindigkeit $\gamma_n = \dot{g}_n$, ob der Kontakt geschlossen bleibt ($\gamma_n = 0$) und noch eine Kontaktkraft wirkt oder, ob sich der Kontakt am Öffnen ist ($\gamma_n > 0$) und die Kontaktkraft verschwindet. Dass man Kraftgesetze von der Lageebene auf Geschwindigkeitsebene überführen kann, ist in [25] und [79] ausgeführt. Die Reibung unterliegt

$$\boldsymbol{\gamma}_f \in \mathscr{N}_{\mathscr{C}_f}(-\boldsymbol{\lambda}_f) \tag{2.25}$$

und ist auf Geschwindigkeitsebene formuliert. Ein hypothetisches Beispiel dafür ist in Abbildung 2.6 gezeigt. Bei der Modellierung von Reibung muss man sich eine konvexe

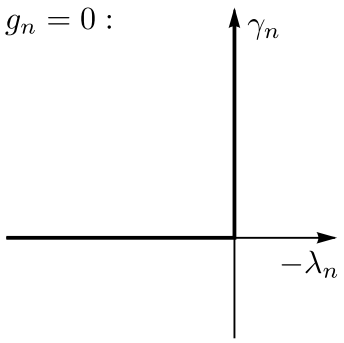

Abbildung 2.5: Kraftgesetz für einen geschlossenen Normalkontakt. Die Kontaktöffnung wird mit g_n und die Öffnungsgeschwindigkeit mit γ_n bezeichnet. Die Kontaktkraft folgt der Normalkegelinklusion $\gamma_n \in \mathscr{N}_{\mathbb{R}_0^-}(-\lambda_n)$.

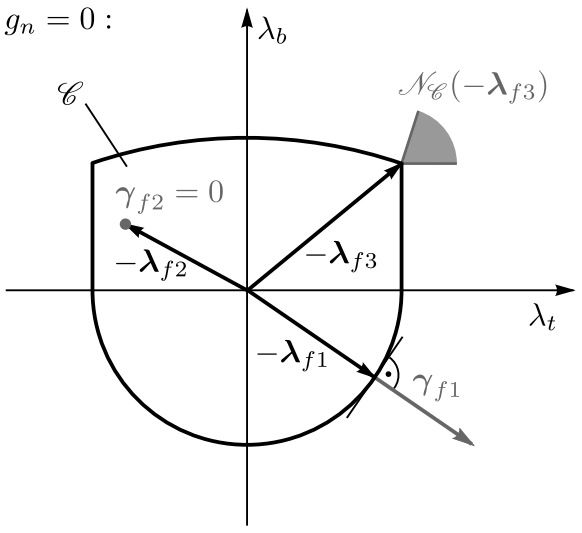

Abbildung 2.6: Kraftgesetz für die Reibung in einem Normalkontakt. Ist der Kontakt geschlossen ($g_n = 0$) liegt die Reibkraft $\boldsymbol{\lambda}_f$ in der Tangentialebene des Kontaktes und unterliegt der Inklusion $\boldsymbol{\gamma}_f \in \mathscr{N}_{\mathscr{C}}(-\boldsymbol{\lambda}_f)$. Die Menge $\mathscr{C}$ ist konvex und repräsentiert das Kraftreservoir der Reibung. Verschwindet die Relativgeschwindigkeit $\boldsymbol{\gamma}_{f2} = 0$ ist $-\boldsymbol{\lambda}_{f2} \in \mathscr{C}$. Ansonsten zeigt $-\boldsymbol{\lambda}_{f1}$ auf den Rand $\partial\mathscr{C}$ der Menge $\mathscr{C}$, wo $\boldsymbol{\gamma}_{f1}$ senkrecht darauf steht.

Menge $\mathscr{C}$ der erlaubten Reibkräfte wählen oder kreiieren und typischerweise ist diese Menge $\mathscr{C}$ proportional zur Normalkraft λ_n im Kontakt. Damit verschwindet bei offenem oder sich öffnendem Kontakt wegen $\lambda_n = 0$ auch die Reibkraft. Deshalb hängt die Reibung de facto wie der Normalkontakt von der Kollisionsdetektion auf Lageebene ab.

2.6.4 Relativkinematik

Für die Evaluierung der Kraftgesetze werden die Relativgeschwindigkeiten in Normal- und Tangentialrichtung benötigt. Man betrachte dazu nochmals Abbildung 2.1. Die Relativgeschwindigkeit zwischen dem Kontaktpunkt auf einem Körper und der statischen Kollisionsgeometrie ist $\boldsymbol{v}_{D'} - \boldsymbol{0}$, womit die Projektion der Relativgeschwindigkeit in Normalrichtung

$$\begin{aligned}
\gamma_n &= \boldsymbol{n}^\mathsf{T}(\boldsymbol{v}_{D'} - \boldsymbol{0}) \\
&= \boldsymbol{n}^\mathsf{T}(\boldsymbol{v}_S + \boldsymbol{\omega}_{\mathrm{IK}} \times \boldsymbol{r}_{SD'}) \\
&= {}_{\mathrm{I}}\boldsymbol{n}^\mathsf{T}({}_{\mathrm{I}}\mathbf{J}_S\boldsymbol{u} + {}_{\mathrm{I}}\boldsymbol{\nu}_S) + {}_{\mathrm{K}}\boldsymbol{n}^\mathsf{T}\left(({}_{\mathrm{K}}\mathbf{J}_R\boldsymbol{u} + {}_{\mathrm{K}}\boldsymbol{\nu}_R) \times {}_{\mathrm{K}}\boldsymbol{r}_{SD'}\right) \\
&= ({}_{\mathrm{I}}\boldsymbol{n}^\mathsf{T}{}_{\mathrm{I}}\mathbf{J}_S + {}_{\mathrm{K}}\boldsymbol{n}^\mathsf{T}{}_{\mathrm{K}}\tilde{\boldsymbol{r}}^\mathsf{T}_{SD'\mathrm{K}}\mathbf{J}_R)\boldsymbol{u} + {}_{\mathrm{I}}\boldsymbol{n}^\mathsf{T}{}_{\mathrm{I}}\boldsymbol{\nu}_S + {}_{\mathrm{K}}\boldsymbol{n}^\mathsf{T}{}_{\mathrm{K}}\tilde{\boldsymbol{r}}^\mathsf{T}_{SD'\mathrm{K}}\boldsymbol{\nu}_R \\
&= ({}_{\mathrm{I}}\boldsymbol{n}^\mathsf{T}{}_{\mathrm{I}}\mathbf{J}_S[\mathsf{b}] + ({}_{\mathrm{K}}\tilde{\boldsymbol{r}}_{SD'\mathrm{K}}\boldsymbol{n})^\mathsf{T}{}_{\mathrm{K}}\mathbf{J}_R[\mathsf{b}])\boldsymbol{u} + {}_{\mathrm{I}}\boldsymbol{n}^\mathsf{T}{}_{\mathrm{I}}\boldsymbol{\nu}_S[\mathsf{b}] + ({}_{\mathrm{K}}\tilde{\boldsymbol{r}}_{SD'\mathrm{K}}\boldsymbol{n})^\mathsf{T}{}_{\mathrm{K}}\boldsymbol{\nu}_R[\mathsf{b}] \\
&= \boldsymbol{w}_n^\mathsf{T}\boldsymbol{u} + \chi_n
\end{aligned} \tag{2.26}$$

wird. In der zweiten Zeile wird die Herleitung der Geschwindigkeit eines Starrkörperpunktes nach Gleichung (2.17) verwendet und danach die Tatsache, dass bei einer skalaren Grösse die einzelnen Summanden (Skalarprodukte) in verschiedenen Koordinatensystemen evaluiert werden dürfen. Dies lässt sich mit $\boldsymbol{b}^\mathsf{T}\mathbf{A}^\mathsf{T} = (\mathbf{A}\boldsymbol{b})^\mathsf{T}$ leicht zeigen: ${}_{\mathrm{I}}\boldsymbol{a}^\mathsf{T}{}_{\mathrm{I}}\boldsymbol{b} = (\mathbf{A}_{\mathrm{IKK}}\boldsymbol{a})^\mathsf{T}\mathbf{A}_{\mathrm{IKK}}\boldsymbol{b} = {}_{\mathrm{K}}\boldsymbol{a}^\mathsf{T}\mathbf{A}^\mathsf{T}_{\mathrm{IK}}\mathbf{A}_{\mathrm{IKK}}\boldsymbol{b} = {}_{\mathrm{K}}\boldsymbol{a}^\mathsf{T}{}_{\mathrm{K}}\boldsymbol{b}$. Weiter werden die Distributivität des Kreuzproduktes und dessen Schiefsymmetrie $\boldsymbol{a} \times \boldsymbol{b} = \tilde{\boldsymbol{a}}\boldsymbol{b} = -\tilde{\boldsymbol{b}}\boldsymbol{a} = \tilde{\boldsymbol{b}}^\mathsf{T}\boldsymbol{a}$ verwendet. Analog lässt sich die Relativgeschwindigkeit in den Tangentialrichtungen aufstellen

$$\begin{aligned}
\boldsymbol{\gamma}_f &= \begin{pmatrix} \boldsymbol{t}^\mathsf{T} \\ \boldsymbol{k}^\mathsf{T} \end{pmatrix} (\boldsymbol{v}_{D'} - \boldsymbol{0}) \\
&= \begin{pmatrix} \boldsymbol{w}_t^\mathsf{T} \\ \boldsymbol{w}_b^\mathsf{T} \end{pmatrix} \boldsymbol{u} + \begin{pmatrix} \chi_t \\ \chi_b \end{pmatrix} = \mathbf{W}_f^\mathsf{T}\boldsymbol{u} + \boldsymbol{\chi}_f.
\end{aligned} \tag{2.27}$$

Im Vergleich mit Gleichung (2.22) zeigt sich, dass die generalisierten Kraftrichtungen $\boldsymbol{w}_n$, $\boldsymbol{w}_t$ und $\boldsymbol{w}_b$ übereinstimmen, vgl. auch [27]. Sie werden vom Treiber zusammen mit den χ_n und $\boldsymbol{\chi}_f$ nach Algorithmus 2.11 berechnet.

2.6.5 Stösse

Die Bewegungsgleichungen (2.23) beschreiben zusammen mit den Kraftgesetzen (2.24) und (2.25) die stossfreie Bewegung des Mehrkörpersystems unter Berücksichtigung der unilateralen Bindungen. Die Formulierung von harten Kontakten impliziert (instantane) Geschwindigkeitssprünge bei stossendem Schliessen, also wenn ein Körper gegen die Kollisionsgeometrie stösst. Dabei treten impulsive Kräfte in (zur Kontaktoberfläche) normaler

Algorithmus 2.11

```
# call to get a list 'activeContacts' of the closed contacts and to
# assemble 'forceDirections' and 'velocityOffsets' in the solver's
# process of solving a time step. 'forceDirections' is the concatenation
# of the generalised force directions of all closed contacts, i.e.
# 'forceDirections' = [w_n1,w_t1,w_b1, w_n2,w_t2,w_b2, ...].
# 'velocityOffsets' is the corresponding vector of chi's, i.e.
# 'velocityOffsets' = [chi_n1,chi_t1,chi_b1, chi_n2,chi_t2,...]^T
# 'solver' is supposed to be a Solver object.
activeContacts = solver.getForceDirectionsAndVelocityOffsets(
                         forceDirections, velocityOffsets)

# this function is a member method of the Solver class.
# 'self' therefore refers to the solver instance.
def getForceDirectionsAndVelocityOffsets(self, forceDirections,
                                         velocityOffsets):
  closedContacts = self.findCollisionPoints() # closedContacts is
  # a list of references to all closed contact objects. Their
  # collisionPoints D' are updated within findCollisionPoints().
  l = 0
  for contact in closedContacts:
    body = contact.body
    location = (body.orientation.T *        # .T denotes transpose
                (contact.collisionPoint.position - body.position))
    normal = contact.collisionPoint.normal
    bodyNormal = body.orientation.T * normal
    [bodyTangent,bodyBitangent] = contact.getTangents(bodyNormal)
    tangent = body.orientation * bodyTangent
    bitangent = body.orientation * bodyBitangent
    bodyNormalMoment = cross(location, bodyNormal)
    bodyTangentMoment = cross(location, bodyTangent)
    bodyBitangnetMoment = cross(location, bodyBitangent)
    for n in self.velocitiesCount:
      JS = body.translationJacobi[:,n] # n-th column
      JR = body.rotationJacobi[:,n]    # n-th column
      forceDirections[n,l] = normal * JS + bodyNormalMoment * JR
      forceDirections[n,l+1] = (tangent*JS + bodyTangentMoment * JR)
      forceDirections[n,l+2] = (bitangent*JS + bodyBitangentMoment * JR)
    velocityOffsets[l] = (normal * body.linearPseudoVelocity +
                bodyNormalMoment * body.angularPseudoVelocity)
    velocityOffsets[l+1] = (tangent*body.linearPseudoVelocity +
                bodyTangentMoment * body.angularPseudoVelocity)
    velocityOffsets[l+2] = (bitangent*body.linearPseudoVelocity +
                bodyBitangentMoment * body.angularPseudoVelocity)
    l += 3;
  return closedContacts
```

und tangentialer Richtung auf. Der Geschwindigkeitssprung zu einem Stosszeitpunkt ist mit der Stossgleichung

$$\mathbf{M}(\boldsymbol{q},\boldsymbol{k})(\boldsymbol{u}^+ - \boldsymbol{u}^-) - \sum_k (\boldsymbol{w}_n \Lambda_n + \mathbf{W}_f \boldsymbol{\Lambda}_f)_k = \mathbf{0} \tag{2.28}$$

beschrieben. Die links- bzw. rechtsseitigen Grenzwerte $\boldsymbol{u}^-$ und $\boldsymbol{u}^+$ bezeichnen dabei die Vor- bzw. Nachstossgeschwindigkeit. Denn die „Geschwindigkeit zum Stosszeitpunkt“ ist nicht mehr definiert, wie das auch bei Beschleunigungssprüngen für die Beschleunigung gilt. Der Geschwindigkeitssprung wird durch die Impulse $\boldsymbol{w}_n\Lambda_n$ und $\mathbf{W}_f\boldsymbol{\Lambda}_f$ erzwungen, die in denselben Unterräumen wie die Kontaktkräfte liegen, d.h. der zum Betrag Λ_n gehörende Impuls zeigt im Anschauungsraum normal zur Kontaktfläche, also in Richtung $\boldsymbol{n}^{\mathrm{C}}$ und der zu $\boldsymbol{\Lambda}_f$ gehörende Impuls liegt im Tangentialraum dazu, aufgespannt durch $\boldsymbol{t}^{\mathrm{C}}$ und $\boldsymbol{b}^{\mathrm{C}}$.

Die Impulse selbst müssen über (modellierte) Stossgesetze bestimmt werden. Diese Stossgesetze sind das Äquivalent zu den Kraftgesetzen. Es wird ein erweitertes Newton Stossgesetz

$$\begin{aligned} \xi_n &= \gamma_n^+ + \varepsilon_n \gamma_n^- \in \mathscr{N}_{\mathbb{R}_0^-}(-\Lambda_n), \\ \boldsymbol{\xi}_f &= \boldsymbol{\gamma}_f^+ + \varepsilon_f \boldsymbol{\gamma}_f^- \in \mathscr{N}_{\mathscr{C}_f}(-\boldsymbol{\Lambda}_f) \end{aligned} \tag{2.29}$$

nach [26, 67] verwendet. Hierbei bilden sich die Vor- und Nachstossrelativgeschwindigkeiten nach den Gleichungen (2.26) und (2.27) mit der entsprechenden Vor- bzw. Nachstossgeschwindigkeit $\boldsymbol{u}^-$ oder $\boldsymbol{u}^+$. Die Faktoren ε_n und ε_f sind newtonsche Stosskoeffizienten. Bei einem Stoss wird in Normalrichtung die Nachstossrelativgeschwindigkeit im betroffenen Kontakt $\gamma_n^+ = -\varepsilon_n\gamma_n^-$ proportional zum Koeffizienten invertiert. Der Impuls $\Lambda_n > 0$ ist in diesem Fall beliebig gross und $\xi_n = 0$ verschwindet.

2.6.6 Massdifferentialinklusion

Mit den Gleichungen (2.10) und (2.23) ist die stossfreie Bewegung beschrieben und die Stossgleichung (2.28) komplettiert die Beschreibung auf alle Zeiten. Mit der Einführung von Differentialmassen können die Gleichungen (2.23) und (2.28) in einer Gleichung

$$\mathbf{M}\,\mathrm{d}\boldsymbol{u} - \boldsymbol{h}\,\mathrm{d}t - \sum_k (\boldsymbol{w}_n\,\mathrm{d}P_n + \boldsymbol{W}_f\,\mathrm{d}\boldsymbol{P}_f)_k = \mathbf{0} \tag{2.30}$$

vereint werden. Man kann sich diese Gleichung grob als Kombination (2.23)$\cdot\,\mathrm{d}t$+(2.28)$\cdot\,\mathrm{d}\eta$ der vorherigen zwei Bestandteile des Bewegungsbeschriebs vorstellen. Die Differentialmasse ergeben sich nach

$$\begin{aligned} \mathrm{d}\boldsymbol{u} &= \dot{\boldsymbol{u}}\,\mathrm{d}t + (\boldsymbol{u}^+ - \boldsymbol{u}^-)\,\mathrm{d}\eta, \\ (\mathrm{d}P_n)_k &= (\lambda_n)_k\,\mathrm{d}t + (\Lambda_n)_k\,\mathrm{d}\eta, \quad (\mathrm{d}\boldsymbol{P}_f)_k = (\boldsymbol{\lambda}_f)_k\,\mathrm{d}t + (\boldsymbol{\Lambda}_f)_k\,\mathrm{d}\eta. \end{aligned} \tag{2.31}$$

Der Vollständigkeit halber seien nochmals die Kraft- bzw. Impulsgesetze wiederholt

$$\begin{aligned} g_n = 0: \quad \xi_n &= (1+\varepsilon_n)\gamma_n \in \mathscr{N}_{\mathbb{R}_0^-}(-\lambda_n), \quad \xi_n = \gamma_n^+ + \varepsilon_n\gamma_n^- \in \mathscr{N}_{\mathbb{R}_0^-}(-\Lambda_n), \\ \boldsymbol{\xi}_f &= (1+\varepsilon_f)\boldsymbol{\gamma}_f \in \mathscr{N}_{\mathscr{C}_f}(-\boldsymbol{\lambda}_f), \quad \boldsymbol{\xi}_f = \boldsymbol{\gamma}_f^+ + \varepsilon_f\boldsymbol{\gamma}_f^- \in \mathscr{N}_{\mathscr{C}_f}(-\boldsymbol{\Lambda}_f). \end{aligned} \tag{2.32}$$

Man beachte, dass im stossfreien Fall $\gamma_l^- = \gamma_l^+ = \gamma_l,\ l \in \{n, f\}$ ist und die Kraftgesetze in diesem Fall ebenfalls in ξ_l formuliert werden können. Dabei wird verwendet, dass ein positives Vielfaches eines Vektors, im Normalkegel bleibt, wenn der Vektor selbst schon drin liegt. Das lässt sich mit der Definition (2.20) des Normalkegels leicht zeigen, denn ein positives Vielfaches eines Skalarprodukts ändert nie dessen Vorzeichen. Damit ist die Bewegung des Mehrkörpersystems in einem einzelnen Gleichungssatz beschrieben.

2.7 Lösen der Bewegungsgleichungen

Der im letzten Abschnitt erarbeitete Gleichungssatz, der die Bewegung des Mehrkörpersystems für alle Zeiten beschreibt, wird nachfolgend über ein Zeitintervall integriert und gleichzeitig diskretisiert. Dabei resultiert ein Inklusionsproblem für die Kontaktperkussionen $\hat{\boldsymbol{P}}_l$ aller geschlossenen Kontakte, das für jeden Zeitschritt iterativ zu lösen ist. Sind die Kontaktperkussionen bestimmt, folgt daraus der neue Bewegungszustand zum Ende des Zeitschrittes.

2.7.1 Systemeingangszustand

Während zu einer Zeit $t^{[i]}$ die generalisierten Lagen und Geschwindigkeiten $(\boldsymbol{q}^{[i]}, \boldsymbol{u}^{[i]})$ als Anfangsbedingung gegeben sind, müssen die Eingangsgrössen $(\boldsymbol{k}^{[i]}, \boldsymbol{m}^{[i]})$ für eine Zeit $t^{[i]}$ bestimmt bzw. eingelesen werden. Kinematisch aktuierte Gelenke sind oft so realisiert, dass sie die lokalen Lagen $\boldsymbol{l}$ von einem *Eingang* $\boldsymbol{\Phi}$ lesen. Im Falle des Bobsimulators wird der Lenkwinkel, also die Lage eines Rotationsgelenkes, über einen Sensor in die Simulation eingelesen. Von diesem Eingang muss auch die Geschwindigkeit $\boldsymbol{c}$ und Beschleunigung $\dot{\boldsymbol{c}}$ abgeleitet werden, was beispielsweise über ein explizites Euler Verfahren

$$\begin{aligned}
t^{[i+1]} &= t^{[i]} + \Delta t, & t^{[0]} &= t_0,\\
\boldsymbol{l}^{[i+1]} &= \boldsymbol{\Phi}(t^{[i+1]}),\\
\boldsymbol{c}^{[i+1]} &= \frac{\boldsymbol{l}^{[i+1]} - \boldsymbol{l}^{[i]}}{\Delta t}, & \boldsymbol{c}^{[0]} &= \boldsymbol{c}_0,\\
\dot{\boldsymbol{c}}^{[i+1]} &= \frac{\boldsymbol{c}^{[i+1]} - \boldsymbol{c}^{[i]}}{\Delta t}, & \dot{\boldsymbol{c}}^{[0]} &= \dot{\boldsymbol{c}}_0
\end{aligned} \tag{2.33}$$

numerisch approximiert wird. Alternativ zur numerischen Approximation müsste man sonst neben den Lagen $\boldsymbol{l}$ auch die Geschwindigkeiten $\boldsymbol{c}$ und Beschleunigungen $\dot{\boldsymbol{c}}$ über Sensoren messen oder die Lagen $\boldsymbol{l}$ über eine zweifach differenzierbare Funktion $\boldsymbol{\Phi}(t)$ vorgeben und deren Ableitungen ebenfalls implementieren[5]. In dem man alle Eingänge gemäss Gleichung (2.33) abarbeitet, werden die Eingangsgrössen $(\boldsymbol{k}^{[i]}, \boldsymbol{m}^{[i]})$ für eine Zeit $t^{[i]}$ assembliert.

[5]Häufig werden in kinematisch aktuierten Gelenken auch die lokalen Geschwindigkeiten über einen Eingang $\boldsymbol{\Phi}$ gesetzt. In diesem Fall müssen die lokalen Lagen $\boldsymbol{l}$ und Beschleunigungen $\dot{\boldsymbol{c}}$ dazu bestimmt

2.7.2 Diskretisierung der Massdifferentialgleichung

Die folgenden Schritte sind Abschnitt 5.3 von [79] entnommen. Die Massdifferentialgleichung (2.30) wird über ein Zeitintervall $\Delta t = t_E - t_B$ integriert und die einzelnen Integrale mit

$$\begin{aligned}
\int_{[t_B,t_E]} \mathbf{M}(\boldsymbol{q},\boldsymbol{k})\,\mathrm{d}\boldsymbol{u} &\approx \mathbf{M}(\boldsymbol{q}^M,\boldsymbol{k}^{[i+1]})(\boldsymbol{u}^E-\boldsymbol{u}^B),\\
\int_{[t_B,t_E]} \boldsymbol{h}(\boldsymbol{q},\boldsymbol{k},\boldsymbol{u},\boldsymbol{m})\,\mathrm{d}t &\approx \boldsymbol{h}(\boldsymbol{q}^M,\boldsymbol{k}^{[i+1]},\boldsymbol{u}^B,\boldsymbol{m}^{[i+1]})\Delta t,\\
\int_{[t_B,t_E]} (\boldsymbol{w}_n(\boldsymbol{q},\boldsymbol{k})\,\mathrm{d}P_n)_k &\approx (\boldsymbol{w}_n(\boldsymbol{q}^M,\boldsymbol{k}^{[i+1]})\hat{P}_n)_k,\\
\int_{[t_B,t_E]} (\mathbf{W}_f(\boldsymbol{q},\boldsymbol{k})\,\mathrm{d}\boldsymbol{P}_f)_k &\approx (\mathbf{W}_f(\boldsymbol{q}^M,\boldsymbol{k}^{[i+1]})\hat{\boldsymbol{P}}_f)_k
\end{aligned} \tag{2.34}$$

approximiert, wobei folgende Abkürzungen

$$\begin{aligned}
&\boldsymbol{q}^B = \boldsymbol{q}(t_B),\quad \boldsymbol{q}^E = \boldsymbol{q}(t_E),\quad \boldsymbol{u}^B = \boldsymbol{u}(t_B),\quad \boldsymbol{u}^E = \boldsymbol{u}(t_E),\\
&\boldsymbol{q}^M = \boldsymbol{q}^B + \frac{\Delta t}{2}\dot{\boldsymbol{q}}(\boldsymbol{q}^B,\boldsymbol{k}^{[i+1]},\boldsymbol{u}^B) = \boldsymbol{q}^B + \frac{\Delta t}{2}\mathbf{B}(\boldsymbol{q}^B,\boldsymbol{k}^{[i+1]})\boldsymbol{u}^B
\end{aligned} \tag{2.35}$$

verwendet sind und für das Aufstellen von $\boldsymbol{q}^M$ die Gleichung (2.10) benutzt wird. Von hier weg seien mit $\mathbf{M}, \boldsymbol{h}, \boldsymbol{w}_n$ und $\mathbf{W}_f$ die approximierten Grössen bezeichnet, die an den in (2.33) angegebenen Stellen ausgewertet sind. Es sind deshalb auch für alle geschlossenen Kontakte $\{k\}$ die $\chi_n = \chi_n(\boldsymbol{q}^M,\boldsymbol{k}^{[i+1]},\boldsymbol{u}^B,\boldsymbol{m}^{[i+1]})$ und $\boldsymbol{\chi}_f = \boldsymbol{\chi}_f(\boldsymbol{q}^M,\boldsymbol{k}^{[i+1]},\boldsymbol{u}^B,\boldsymbol{m}^{[i+1]})$. Die Kontaktperkussionen folgen den Normalkegelinklusionen

$$\begin{aligned}
\hat{\xi}_n &= \gamma_n^E + \varepsilon_n\gamma_n^B \in \mathscr{N}_{\mathbb{R}_0^-}(-\hat{P}_n),\\
\hat{\boldsymbol{\xi}}_f &= \boldsymbol{\gamma}_f^E + \varepsilon_f\boldsymbol{\gamma}_f^B \in \mathscr{N}_{\hat{\mathscr{C}}_f}(-\hat{\boldsymbol{P}}_f).
\end{aligned} \tag{2.36}$$

Um diese Inklusionen numerisch lösen zu können, wird auf implizite Gleichungen übergegangen. Normalkegelinklusionen (2.20) sind äquivalent [79, 64] zu

$$-\boldsymbol{\lambda} = \mathrm{prox}_{\mathscr{C}}(r\boldsymbol{\gamma} - \boldsymbol{\lambda}), \tag{2.37}$$

wobei

$$\boldsymbol{x} = \mathrm{prox}_{\mathscr{C}}(\boldsymbol{y}) := \underset{\boldsymbol{y}^*\in\mathscr{C}}{\mathrm{argmin}}\,\|\boldsymbol{y}-\boldsymbol{y}^*\| \tag{2.38}$$

---

werden, was beispielsweise ebenfalls über ein explizites Euler Verfahren

$$\begin{aligned}
t^{[i+1]} &= t^{[i]} + \Delta t, & t^{[0]} &= t_0,\\
c^{[i+1]} &= \boldsymbol{\Phi}(t^{[i+1]}), & &\\
l^{[i+1]} &= l^{[i]} + c^{[i+1]}\Delta t, & \boldsymbol{l}^{[0]} &= \boldsymbol{l}_0,\\
\dot{\boldsymbol{c}}^{[i+1]} &= \frac{\boldsymbol{c}^{[i+1]} - \boldsymbol{c}^{[i]}}{\Delta t}, & \dot{\boldsymbol{c}}^{[0]} &= \dot{\boldsymbol{c}}_0
\end{aligned}$$

geschehen kann.

denjenigen Punkt $\boldsymbol{x} \in \mathscr{C}$ bezeichnet, der den Abstand zum Punkt $\boldsymbol{y}$ minimiert. Damit schreiben sich die Inklusionen mit

$$(\hat{P}_n)_k = -\operatorname{prox}_{\mathbb{R}_0^-}\left((r_n\hat{\xi}_n - \hat{P}_n)_k\right), \quad (\hat{\boldsymbol{P}}_f)_k = -\operatorname{prox}_{\hat{\mathscr{C}}_f}\left((r_f\hat{\boldsymbol{\xi}}_f - \hat{\boldsymbol{P}}_f)_k\right). \tag{2.39}$$

Löst man die integrierte Massdifferentialinklusion auf nach der Geschwindigkeit zum Ende des Integrationsintervalls

$$\boldsymbol{u}^E = \mathbf{M}^{-1}\left(\sum_k (\boldsymbol{w}_n\hat{P}_n + \mathbf{W}_f\hat{\boldsymbol{P}}_f)_k\right) + \mathbf{M}^{-1}\boldsymbol{h}\Delta t + \boldsymbol{u}^B, \tag{2.40}$$

und stellt mit den Gleichungen (2.26) und (2.27) die $\hat{\boldsymbol{\xi}}_l$ aus den Kontaktperkussionsgesetzen (2.36) auf, ergibt sich für jedes Kraftelement l in allen geschlossenen Kontakten $\{k\}$

$$\begin{aligned}\hat{\boldsymbol{\xi}}_l &= \mathbf{W}_l^\mathsf{T}\mathbf{M}^{-1}\left(\sum_k (\boldsymbol{w}_n\hat{P}_n + \mathbf{W}_f\hat{\boldsymbol{P}}_f)_k\right) + \mathbf{W}_l^\mathsf{T}\mathbf{M}^{-1}\boldsymbol{h}\Delta t + (1+\varepsilon_l)(\mathbf{W}_l^\mathsf{T}\boldsymbol{u}^B + \boldsymbol{\chi}_l)\\ &= \mathbf{W}_l^\mathsf{T}\mathbf{M}^{-1}\left(\sum_k (\boldsymbol{w}_n\hat{P}_n + \mathbf{W}_f\hat{\boldsymbol{P}}_f)_k\right) + \boldsymbol{c}_l\\ &= \mathbf{G}_l\hat{\boldsymbol{P}} + \boldsymbol{c}_l, \qquad \in \mathscr{N}_{\hat{\mathscr{C}}_l}(-\hat{\boldsymbol{P}}_l).\end{aligned} \tag{2.41}$$

Iterativ wird diese Gleichung für alle $\{l\}$ aller geschlossenen Kontakte $\{k\}$ gelöst, in dem für jedes l

$$\hat{\boldsymbol{P}}_l^{[j+1]} = -\operatorname{prox}_{\hat{\mathscr{C}}_l}(r_l(\mathbf{G}_l\hat{\boldsymbol{P}}^{[j]} + \boldsymbol{c}_l) - \hat{\boldsymbol{P}}_l^{[j]}) \tag{2.42}$$

die Perkussionen aktualisiert werden, bis das Abbruchkriterium

$$|\hat{\boldsymbol{P}}_l^{[j]} - \hat{\boldsymbol{P}}_l^{[j+1]}| < e_r|\hat{\boldsymbol{P}}_l^{[j+1]}| + e_a \tag{2.43}$$

komponentenweise erfüllt ist. Die relative Fehlertoleranz wird zu $e_r = 10^{-6}$, die absolute zu $e_a = 10^{-4}$ gewählt.

2.7.3 Zusammenfassung

Dieser Unterabschnitt fasst das Lösen der Bewegungsgleichungen über einen Zeitschritt zusammen. Das verwendete Time-stepping Verfahren implementiert das Mittelpunktschema von Moreau. Ausgehend von einem Zustand $(\boldsymbol{q}^B, \boldsymbol{k}^{[i]}, \boldsymbol{u}^B, \boldsymbol{m}^{[i]}, \dot{\boldsymbol{m}}^{[i]})$ werden mit (2.35) mittlere Lagen $\boldsymbol{q}^M$ eruiert. Mit diesem Zustand $(\boldsymbol{q}^M, \boldsymbol{k}^{[i]}, \boldsymbol{u}^B, \boldsymbol{m}^{[i]}, \dot{\boldsymbol{m}}^{[i]})$ werden die Massenmatrix $\mathbf{M}$, der Kraftvektor $\boldsymbol{h}$, sowie alle generalisierten Kraftrichtungen $\boldsymbol{w}_n$ und $\mathbf{W}_f$ der in diesem Zustand geschlossenen Kontakte assembliert. Das geschieht über die in den vorangegangenen Abschnitten beschriebenen Algorithmen. Schliesslich werden anhand dieser Grössen die Matrizen $\mathbf{G}_l$ und Vektoren $\boldsymbol{c}_l$ der geschlossenen Kontakte aufgestellt.

Danach ist das Inklusionsproblem iterativ zu lösen. Für den Bobsimulator geschieht dies nach einem Gauss-Seidel Schema. Es ist zu beachten, das in die Aktualisierung der

Kontaktperkussionen $\hat{\boldsymbol{P}}_l^{[j+1]}$ in Gleichung (2.42) mit $\mathbf{G}_l\hat{\boldsymbol{P}}$ die Kontaktperkussionen aller Kontakte einfliessen. Dabei werden in $\hat{\boldsymbol{P}}$ innerhalb der Iteration die bereits aktualisierten $\hat{\boldsymbol{P}}_l^{[j+1]}$ direkt verwendet für die Aktualisierung der verbleibenden. Die Ausführung der `prox`-Funktionen hat innerhalb einer Schleife über alle Kontaktelemente $\{l\}$ aller Kontakte $\{k\}$ zu erfolgen. Für eine umfassende Behandlung von Iterationsschemen zur Lösung des Inklusionsproblems wird auf [64, 80, 79] verwiesen.

Sind die Kontaktperkussionen $\hat{P}_n$ und $\hat{\boldsymbol{P}}_f$ für alle $\{k\}$ bestimmt, folgt mit Gleichung (2.40) die generalisierte Geschwindigkeit $\boldsymbol{u}^E$ zum Ende des Zeitschritts. Damit werden zum Schluss mit Gleichung (2.10) die Lagen

$$\boldsymbol{q}^E = \boldsymbol{q}^M + \frac{\Delta t}{2}\dot{\boldsymbol{q}}(\boldsymbol{q}^M, \boldsymbol{k}^{[i+1]}, \boldsymbol{u}^E) = \boldsymbol{q}^M + \frac{\Delta}{2}\mathbf{B}(\boldsymbol{q}^M, \boldsymbol{k}^{[i+1]})\boldsymbol{u}^E \tag{2.44}$$

zum Ende des Zeitschritts bestimmt.

2.7.4 In die Gelenke induzierte Kontaktkräfte

Mit der Bestimmung des dynamischen Zustandes $\boldsymbol{q}^E, \boldsymbol{u}^E$ zum Ende eines Zeitschrittes, kann grundsätzlich der nächste Zeitschritt in Angriff genommen werden. Für den Einbau eines Force-Feedbacksystems in einen Simulator kann es von Interesse sein, zuerst noch zu bestimmen, welche Kräfte und Momente von den Kontaktperkussionen in den einzelnen Gelenken induziert werden. Bei den Kontaktperkussionen $\hat{\boldsymbol{P}}_l$ handelt es sich um die über den Zeitschritt integrierten Kontaktkräfte $\boldsymbol{\lambda}_l$ zusammen mit endlich vielen Stossimpulsen $\boldsymbol{\Lambda}_l$. Als Näherung wird von den Perkussionen eines Kontaktes auf durchschnittlich während dem Zeitschritt wirkende Kräfte

$$\bar{\lambda}_n = \frac{\hat{P}_n}{\Delta t}, \qquad\qquad \bar{\boldsymbol{\lambda}}_f = \frac{\hat{\boldsymbol{P}}_f}{\Delta t} \tag{2.45}$$

geschlossen. Im euklidischen Anschauungsraum ergibt sich daraus die approximierte Kontaktkraft $\boldsymbol{F}_{\bar{\boldsymbol{\lambda}}} = \bar{\lambda}_n\boldsymbol{n}^{\mathrm{C}} + \bar{\lambda}_t\boldsymbol{t}^{\mathrm{C}} + \bar{\lambda}_b\boldsymbol{b}^{\mathrm{C}}$, welche am jeweiligen Körper im Kontaktpunkt D' angreift. Mit Hilfe der axiomatischen Invarianz der virtuellen Leistung unter Koordinatentransformation lässt sich die durch $\boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}$ in den freien Gelenken induzierte generalisierte Kraft $\boldsymbol{j}$ bestimmen. Es gilt nämlich

$$\begin{aligned}
\underline{\delta}P = \boldsymbol{j}^{\mathsf{T}}\,\underline{\delta}\boldsymbol{u} &= \boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}^{\mathsf{T}}\,\underline{\delta}\boldsymbol{v}_{D'} \\
&= \boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}^{\mathsf{T}}\,\underline{\delta}\boldsymbol{v}_S - \boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}^{\mathsf{T}}(\boldsymbol{r}_{SD'} \times \underline{\delta}\boldsymbol{\omega}) \\
&= \boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}^{\mathsf{T}}\mathbf{J}_S\,\underline{\delta}\boldsymbol{u} - \boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}^{\mathsf{T}}\tilde{\boldsymbol{r}}_{SD'}\mathbf{J}_R\,\underline{\delta}\boldsymbol{u}) \\
&= (\boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}^{\mathsf{T}}\mathbf{J}_S - \boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}^{\mathsf{T}}\tilde{\boldsymbol{r}}_{SD'}\mathbf{J}_R)\,\underline{\delta}\boldsymbol{u})
\end{aligned} \tag{2.46}$$

und damit ergibt sich der Anteil von $\boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}$ in der induzierten generalisierten Kraft

$$\boldsymbol{j} = {}_{\mathrm{I}}\mathbf{J}_{S_i\mathrm{I}}^{\mathsf{T}}\boldsymbol{F}_{\bar{\boldsymbol{\lambda}}} + {}_{\mathrm{K}_i}\mathbf{J}_{R_i\mathrm{K}_i}^{\mathsf{T}}\tilde{\boldsymbol{r}}_{SD'\mathrm{K}_i}\boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}. \tag{2.47}$$

Analog dazu lässt sich mit $\underline{\delta}P = \boldsymbol{i}^{\mathsf{T}}\,\underline{\delta}\boldsymbol{m}$ der Anteil in den aktuierten Gelenken schreiben mit

$$\boldsymbol{i} = {}_{\mathrm{I}}\mathbf{H}_{S_i\mathrm{I}}^{\mathsf{T}}\boldsymbol{F}_{\bar{\boldsymbol{\lambda}}} + {}_{\mathrm{K}_i}\mathbf{H}_{R_i\mathrm{K}_i}^{\mathsf{T}}\tilde{\boldsymbol{r}}_{SD'\mathrm{K}_i}\boldsymbol{F}_{\bar{\boldsymbol{\lambda}}}. \tag{2.48}$$

Kapitel 3

Modell eines künstlichen Eiskanals

Dieses Kapitel beschreibt ein Modell des Eiskanals, das auf den Konstruktionsdaten einer existierenden Kunsteisbahn basiert. Die Konstruktionsdaten definieren die Geometrie der Betonoberfläche, auf die im Winter die Eisschicht aufgespritzt ist. Das Modell rekonstruiert digital die spezifizierte Betonoberfläche. Aus Videoaufnahmen werden zusätzliche Daten gewonnen, anhand derer die Betonoberfläche mit Berandungselementen ergänzt wird. Üblicherweise sind in Kurven am oberen Rand Bretterverschläge als Endanschlag angebracht, damit der Bobschlitten die Bahn unter keinen Umständen verlassen kann. Zur Überhöhung der Eiskanaltiefe oder Abriegelung von Startzufahrten in den Eiskanal, sind häufig an verschiedenen Stellen auch zusätzliche Wände angebracht.

Aus den Konstruktionsdaten wird schrittweise eine Splinefläche aufgebaut. Ihre Bestandteile werden in Abschnitt 3.1 beschrieben. Danach wird die Splinefläche trianguliert, also in ein Dreiecksgitter umgewandelt, was Gegenstand von Abschnitt 3.2 ist. Als solches Gitter wird das Bahnmodell einerseits beim Rendern der Szenerie und andererseits als Kollisionsgeometrie für das Bobmodell in der Physiksimulation verwendet. Abschnitt 3.3 widmet sich schliesslich den erwähnten Randelementen, mit denen gleich verfahren wird wie mit der Betonoberfläche.

3.1 Konstruktionsdaten einer Bahn

Die Konstruktionsdaten einer Kunsteisbahn beinhalten drei Datensätze. Zum Einen ist die ebene Geometrie der Aufsicht auf den Bahnverlauf spezifiziert. Diese Geometrie wird *Profillinie* genannt und sie wird im Abschnitt 3.1.1 vorgestellt. Zum Zweiten nennen die Konstruktionsdaten die Parameter eines *Höhenprofils*, das die Höhe der Bahn entlang der Profillinie in Meter über Meer beschreibt. Das Höhenprofil wird in Abschnitt 3.1.2 diskutiert. Zusammen mit der Profillinie wird daraus das *Streckenprofil* gebildet, welches den eindimensionalen Bahnverlauf im dreidimensionalen Raum beschreibt. Das Streckenprofil ist in Abschnitt 3.1.3 behandelt. Schliesslich definieren die Konstruktionsdaten *Querschnittsgeometrien*, die in gleichmässigen Abständen entlang der Profillinie verteilt sind und auf dem Streckenprofil positioniert werden. Abschnitt 3.1.4 bespricht die Querschnitte.

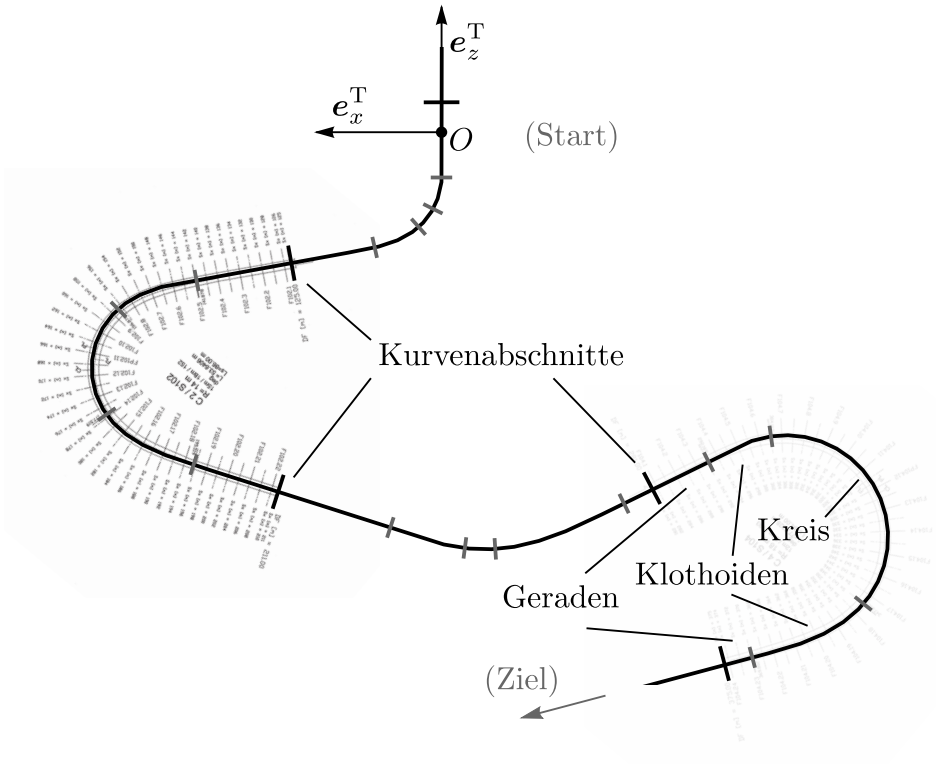

Abbildung 3.1: Profillinie der ersten vier Kurven von der Bahn in Whistler, Kanada. Jede Kurve besteht aus zwei Geraden, zwei Klothoidenübergängen und einem Kreissegment. Hinterlegt sind Ausschnitte der Konstruktionspläne.

3.1.1 Profillinie

Die Profillinie einer Bobbahn ist in Kurven unterteilt, wobei jede Kurve selbst aus fünf (Kurven-)Segmenten besteht. Abbildung 3.1 zeigt die Profillinie der ersten vier Kurven der Bobbahn in Whistler. Jeder Kurveneingang beginnt mit einem geraden Teilstück, gefolgt von einer Klothoide, auch Euler-Spirale genannt [56], die die Krümmung mit der Bogenlänge linear erhöht. Der Mittelteil einer Kurve besteht aus einem Kreisstück und ist damit von konstanter Krümmung. Es folgt wieder eine Klothoide, die die Krümmung linear zu ihrer Länge verkleinert, und eine weitere Gerade zum Ausgang der Kurve. Die Konstruktionsdaten [39, 42, 45, 49] sind üblicherweise über verschiedene Pläne und Tabellen verteilt und listen für jede Kurve Masse auf, die die Geometrien der einzelnen Kurvensegmente definieren. In diesem Rohdatensatz sind diverse Grössen redundant, weshalb daraus ein Satz von zehn Parametern extrahiert wird. Basierend auf diesen wird eine Parametrisierung ${}_{\mathrm{T}}\boldsymbol{r}_{OP}(s) = {}_{\mathrm{T}}\boldsymbol{p}(s)$ der Profillinie konstruiert. Das Argument s bezeichnet darin die Bogenlänge der Linie und identifiziert damit eindeutig eine Position auf der Bahn, weshalb später auch von der *Stelle* s gesprochen wird. Der Punkt O bezeichnet den Ursprung des inertialen Koordinatensystems $(\boldsymbol{e}_x^{\mathrm{T}}, \boldsymbol{e}_y^{\mathrm{T}}, \boldsymbol{e}_z^{\mathrm{T}})$, in dem die Bahn modelliert wird. Er liegt an der Stelle $s = 0$, wo auch die Messung der Laufzeit ausgelöst wird. Die Ziellinie befindet sich an der Stelle $s = L$. Man beachte, dass sich die Bahn über diese

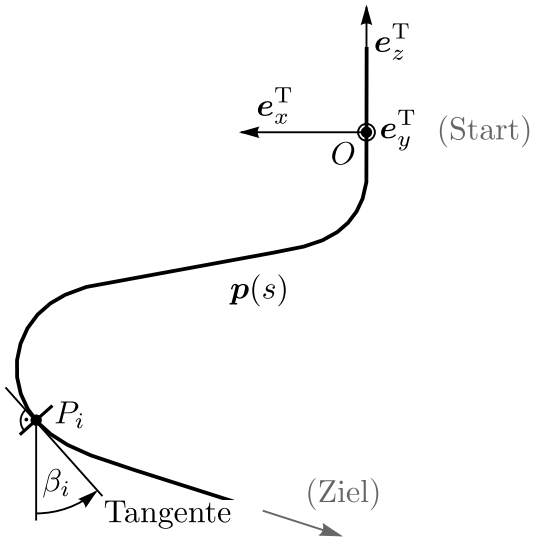

Abbildung 3.2: Richtung $\beta(s)$ der Profillinie $\boldsymbol{p}(s)$. Der Winkel $\beta_i = \beta(s_i)$ misst sich zwischen $-\boldsymbol{e}_z^{\mathrm{T}}$ und der Tangente an $\boldsymbol{p}(s)$ im Punkt $\boldsymbol{r}_{OP_i} = \boldsymbol{p}(s_i)$.

Stellen hinaus erstreckt. Die Bahn beginnt mit einem Startbereich für welchen $s < 0$ ist und sie zieht sich nach der Ziellinie $s > L$ einen Bremshang hinauf. Neben der Profillinie selbst, wird anhand des gewählten Parametersatzes auch eine Parametrisierung $\beta(s)$ ihrer Richtung konstruiert. Man vergleiche dazu Abbildung 3.2, die den Winkel $\beta_i = \beta(s_i)$ an einer Stelle s_i zeigt. Der Winkel $\beta(s)$ misst sich in $\boldsymbol{e}_y^{\mathrm{T}}$-Richtung zwischen $-\boldsymbol{e}_z^{\mathrm{T}}$ und der Tangente an die Profillinie in $\boldsymbol{p}(s)$ in Fahrtrichtung. Die Position eines Punktes P an der Stelle s auf der Profillinie und der dazugehörige Winkel $\beta(s)$ werden als Tupel $(s, {}_{\mathrm{T}}\boldsymbol{r}_{OP}, \beta)$ zusammengefasst und *Profilpunkt* genannt.

Der erste Parameter $c \in \{-1, 1\}$ der zehn Konstruktionsvariablen gibt an, in welche Richtung sich die Kurve krümmt. Der Wert $c = 1$ entspricht einer Drehung um die positive $\boldsymbol{e}_y^{\mathrm{T}}$-Achse und damit einer Linkskurve. Eine Rechtskurve hat den Wert $c = -1$ und dreht um die negative $\boldsymbol{e}_y^{\mathrm{T}}$-Achse. Man beachte, dass die Fahrtrichtung in den Abbildungen 3.1 und 3.2 von oben nach unten, am Start also in $-\boldsymbol{e}_z^{\mathrm{T}}$-Richtung zeigt. Der nächste Parameter s_b bezeichnet den Anfang einer Kurve und der Parameter s_c den Beginn des gekrümmten Teils der Kurve, also den Startpunkt der Eingangsklothoide. Die Länge der Eingangsgeraden ist damit $l_{s,en} = s_c - s_b$. Abbildung 3.3 skizziert die Ausgangslage für die Parametrisierung von Profilpunkten, die innerhalb eines geraden Kurvensegments liegen. Ausgehend von einem Startprofilpunkt $(s_0, {}_{\mathrm{T}}\boldsymbol{r}_{OP_0}, \beta_0)$ und einer Gesamtlänge l_s des geraden Segments lassen sich die Profilpunkte mit

$$\begin{aligned} s \in [s_0; s_0 + l_s]: \quad {}_{\mathrm{T}}\boldsymbol{e}_z^{\mathrm{P}} &= (\sin(\beta_0 + \pi), 0, \cos(\beta_0 + \pi))^{\mathsf{T}}, \\ {}_{\mathrm{T}}\boldsymbol{p}(s) &= {}_{\mathrm{T}}\boldsymbol{p}_0 + (s - s_0)\,{}_{\mathrm{T}}\boldsymbol{e}_z^{\mathrm{P}}, \\ \beta(s) &= \beta_0 \end{aligned} \tag{3.1}$$

parametrisieren.

Auf die Gerade folgt die Eingangsklothoide. Abbildung 3.4 skizziert die Eingangsklothoi-

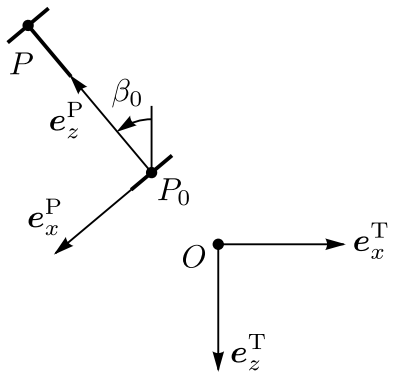

Abbildung 3.3: Skizze zur Parametrisierung eines geraden Kurvensegments.

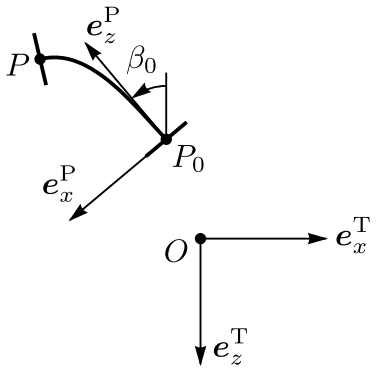

Abbildung 3.4: Skizze zur Parametrisierung der Eingangsklothoide einer Kurve.

de wiederum ausgehend von einem Profilpunkt $(s_0, {}_{\mathrm{T}}\boldsymbol{r}_{OP_0}, \beta_0)$. Eine Klothoide ist nach [62] definiert über die Gesamtbogenlänge l_{cloth}, die Anfangskrümmung k_A und die Krümmung $k_B = \frac{1}{r_B}$ im Endpunkt der Kurve, wobei r_B den Krümmungsradius am Ende der Klothoide bezeichnet. Im Falle von Bobbahnprofillinien verschwindet die Anfangskrümmung $k_A = 0$. Damit ist der Übergang von der Eingangsgeraden zum Klothoidensegment (mit dem Ausnahmefall ganz verschwindender Klothoidensegmente) im Allgemeinen krümmungsstetig. Die Konstruktionsdaten geben die Bogenlänge der Eingangsklothoide mit einem Parameter $l_{cl,en}$ an. Der Krümmungsradius r_B zum Ende der Klothoide ist in den Konstruktionsdaten implizit über einen Parameter $A^2_{en} := \frac{l_{cl,en}}{k_B - k_A} = l_{cl,en} r_B$ bestimmt. In den meisten Fällen, aber nicht immer, entspricht dieser Krümmungsradius dem Radius des anschliessenden Kreiskurvensegments. Mit der Koordinatentransformationsmatrix

$$\mathbf{A}_{\mathrm{TP}} = [{}_{\mathrm{T}}\boldsymbol{e}^{\mathrm{P}}_x, {}_{\mathrm{T}}\boldsymbol{e}^{\mathrm{P}}_y, {}_{\mathrm{T}}\boldsymbol{e}^{\mathrm{P}}_z] = \begin{bmatrix} \cos(\beta_0 + \pi) & 0 & \sin(\beta_0 + \pi) \\ 0 & 1 & 0 \\ -\sin(\beta_0 + \pi) & 0 & \cos(\beta_0 + \pi) \end{bmatrix}, \tag{3.2}$$

$l_{cloth} = l_{cl,en}$ und $A^2 = A^2_{en}$ lassen sich Profilpunkte, die auf der Kurveneingangsklothoide liegen, mit

$$\begin{aligned} s \in [s_0; s_0 + l_{cloth}]: \qquad a &= \sqrt{\pi A^2},\ t(s) = \frac{s - s_0}{a}, \\ {}_{\mathrm{T}}\boldsymbol{p}(s) &= {}_{\mathrm{T}}\boldsymbol{p}_0 + \mathbf{A}_{\mathrm{TP}}\, a \begin{pmatrix} c\, S(t(s)) \\ 0 \\ C(t(s)) \end{pmatrix}, \\ \Delta\beta(s) &= \frac{(s - s_0)^2}{2A^2}, \\ \beta(s) &= \beta_0 + c\Delta\beta(s) \end{aligned} \tag{3.3}$$

parametrisieren, wobei

$$\begin{aligned} S(t) &= \int_0^t \sin(\frac{\pi}{2}u^2)\,\mathrm{d}u \\ C(t) &= \int_0^t \cos(\frac{\pi}{2}u^2)\,\mathrm{d}u \end{aligned}$$

gemeinhin als *Fresnelsche Integrale* [1] bezeichnet werden. Man vergleiche hierzu und generell zu Klothoiden auch [51, 62]. Diese Integrale werden für das Bahnmodell mit Hilfe der *SciPy*-Bibliothek [78] numerisch ausgewertet.

Das kreisförmige Mittelsegment einer Kurve ist in Abbildung 3.5 skizziert. Profilpunkte in solchen Segmenten lassen sich ausgehend von einem Punkt $(s_0, {}_{\mathrm{T}}\boldsymbol{r}_{OP_0}, \beta_0)$ mit

$$\begin{aligned} s \in [s_0; s_0 + l_{circ}]: \quad \Delta\beta(s) &= \frac{(s - s_0)}{r_c}, \\ {}_{\mathrm{T}}\boldsymbol{p}(s) &= {}_{\mathrm{T}}\boldsymbol{p}_0 + \mathbf{A}_{\mathrm{TP}}\, r_c \begin{pmatrix} c\,(1 - \cos(\Delta\beta)) \\ 0 \\ \sin(\Delta\beta) \end{pmatrix}, \\ \beta(s) &= \beta_0 + c\,\Delta\beta(s) \end{aligned} \tag{3.4}$$

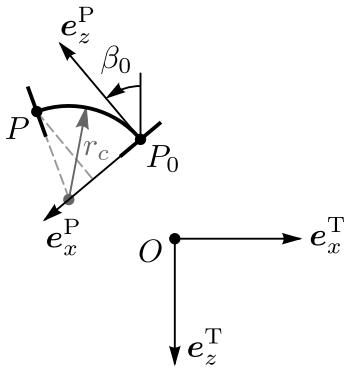

Abbildung 3.5: Skizze zur Konstruktion des kreisförmigen Mittelsegments einer Kurve.

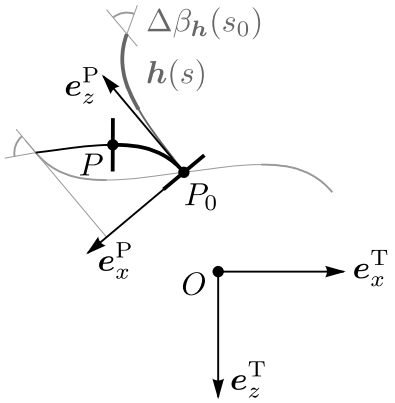

Abbildung 3.6: Skizze zur Konstruktion der Ausgangsklothoide einer Kurve.

parametrisieren, wobei die Länge l_{circ} des Kreissegments und dessen Radius r_c in den Konstruktionsdaten spezifiziert sind und die Trafomatrix $\mathbf{A}_{\mathrm{TP}}$ vorab nach Gleichung (3.2) zu evaluieren ist.

Die Ausgangsklothoide ist in den Konstruktionsdaten, wie die Eingangsklothoide, über Parameter $l_{cl,ex}$ und A_{ex}^2 spezifiziert. Im Falle der Ausgangsklothoide nimmt allerdings die Krümmung in Durchlaufrichtung ab und mündet am Ende mit Krümmung Null in die Ausgangsgerade. Für die Parametrisierung der Ausgangsklothoide kann man sich jener der Eingangklothoide bedienen und damit Programmcode wiederverwenden. Es wird eine Hilfsklothoide ${}_{\mathrm{T}}\boldsymbol{h}(s)$ nach (3.3) mit invertierter Krümmung $c \mapsto -c$ und den beiden Parametern $l_{cloth} = l_{cl,ex}$, $A^2 = A_{ex}^2$ initialisiert und in umgekehrter Richtung, also mit $(s - s_0) \mapsto l_{cloth} - (s - s_0)$, durchlaufen. Abbildung 3.6 skizziert diese Hilfsklothoide neben der angestrebten Ausgangklothoide und dies wiederum für einen Startprofilpunkt $(s_0, {}_{\mathrm{T}}\boldsymbol{p}_0, \beta_0)$. Neben einer Verschiebung des Endpunktes der Hilfsklothoide in den Urspung P_0 des Koordinatensystems $(\boldsymbol{e}_x^{\mathrm{P}}, \boldsymbol{e}_y^{\mathrm{P}}, \boldsymbol{e}_z^{\mathrm{P}})$, ist noch eine Rotation um die positive $\boldsymbol{e}_y^{\mathrm{P}}$-Achse um den Winkel $c\,\Delta\beta_{\boldsymbol{h}}(s_0) + \pi$ nötig. Die Parametrisierung der Ausgangsklothoide kann

folglich mit

$$
\begin{aligned}
s \in [s_0; s_0 + l_{cloth}]: \qquad a &= \sqrt{\pi A^2},\ t(s) = \frac{l_{cloth} - (s - s_0)}{a}, \\
{}_{\mathrm{T}}\boldsymbol{h}(s) &= {}_{\mathrm{T}}\boldsymbol{p}_0 + \mathbf{A}_{\mathrm{TP}}\, a \begin{pmatrix} -c\, S(t(s)) \\ 0 \\ C(t(s)) \end{pmatrix}, \\
\Delta\beta_{\boldsymbol{h}}(s) &= \frac{(l_{cloth} - (s - s_0))^2}{2A^2}, \\
{}_{\mathrm{T}}\boldsymbol{p}(s) &= {}_{\mathrm{T}}\boldsymbol{p}_0 + \mathbf{R}(c\Delta\beta_{\boldsymbol{h}}(s_0) + \pi)\,({}_{\mathrm{T}}\boldsymbol{h}(s) - {}_{\mathrm{T}}\boldsymbol{h}(s_0))\,, \\
\beta(s) &= \beta_0 - c\,(\Delta\beta_{\boldsymbol{h}}(s) - \Delta\beta_{\boldsymbol{h}}(s_0))\,.
\end{aligned}
\tag{3.5}
$$

beschrieben werden, wobei

$$
\mathbf{R}(\gamma) = \begin{bmatrix} \cos(\gamma) & 0 & \sin(\gamma) \\ 0 & 1 & 0 \\ -\sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix} \tag{3.6}
$$

die Rotation um die $\boldsymbol{e}_y^{\mathrm{P}}$-Achse um den Winkel γ bezeichnet.

Auf die Ausgangsklothoide folgt die Ausgangsgerade, deren Richtung β_0 mit dem Endpunkt der Ausgangsklothoide feststeht, und deren Länge $l_{s,ex}$ mit dem in den Konstruktionsdaten angegebenen Ende der Kurve s_e nach

$$
l_{s,ex} = s_e - s_b - (l_{cl,ex} + l_{circ} + l_{cl,en} + l_{s,en})
$$

festgelegt ist. Die Parametrisierung gestaltet sich identisch zu jener der Eingangsgeraden nach Gleichung (3.1).

Mit den angegebenen Parametrisierungen der einzelnen Kurvensegmente lässt sich eine Parametrisierung von Bahnkurven und damit der gesamten Profillinie zusammensetzen. Die Koordinaten ${}_{\mathrm{T}}\boldsymbol{r}_{OP}$ eines Punktes P auf der Profillinie an der Stelle s in Bezug auf das Koordinatensystem $(\boldsymbol{e}_x^{\mathrm{T}}, \boldsymbol{e}_y^{\mathrm{T}}, \boldsymbol{e}_z^{\mathrm{T}})$ werden deshalb mit

$$
{}_{\mathrm{T}}\boldsymbol{r}_{OP} = {}_{\mathrm{T}}\boldsymbol{p}(s) = \begin{pmatrix} p_x(s) \\ 0 \\ p_z(s) \end{pmatrix} \tag{3.7}
$$

bezeichnet. Man beachte, dass die Profillinie eine planare Kurve ist und in der $z-x$-Ebene konstruiert wird. Für das Zusammensetzen der Profillinie aus den einzelnen Segmentparametrisierungen wird der Referenzpunkt $(s^\circ, {}_{\mathrm{T}}\boldsymbol{p}^\circ = (0, 0, -s^\circ)^{\mathsf{T}}, \beta^\circ = 0)$ vorgegeben, welcher als Startpunkt $(s_0, {}_{\mathrm{T}}\boldsymbol{p}_0, \beta_0)$ für das erste Segment hergenommen wird. Ausgehend von diesem wird für die Auswertung eines Punktes an einer Stelle s mit den einzelnen Segmentlängen eruiert, in welchem Kurvensegment (und welcher Kurve) diese Stelle zu liegen kommt. Parallel dazu wird für jedes übersprungene Segment dessen Endpunkt an der Stelle $s_0 + l$ berechnet, welcher als Startpunkt $(s_0, {}_{\mathrm{T}}\boldsymbol{p}_0, \beta_0)$ für das nächste, zu prüfende Segment dient. Gelangt man zu demjenigen Segment $[s_0, s_0 + l]$ in dem s liegt, wertet man die korrespondierende Parametrisierung an der Stelle s aus.

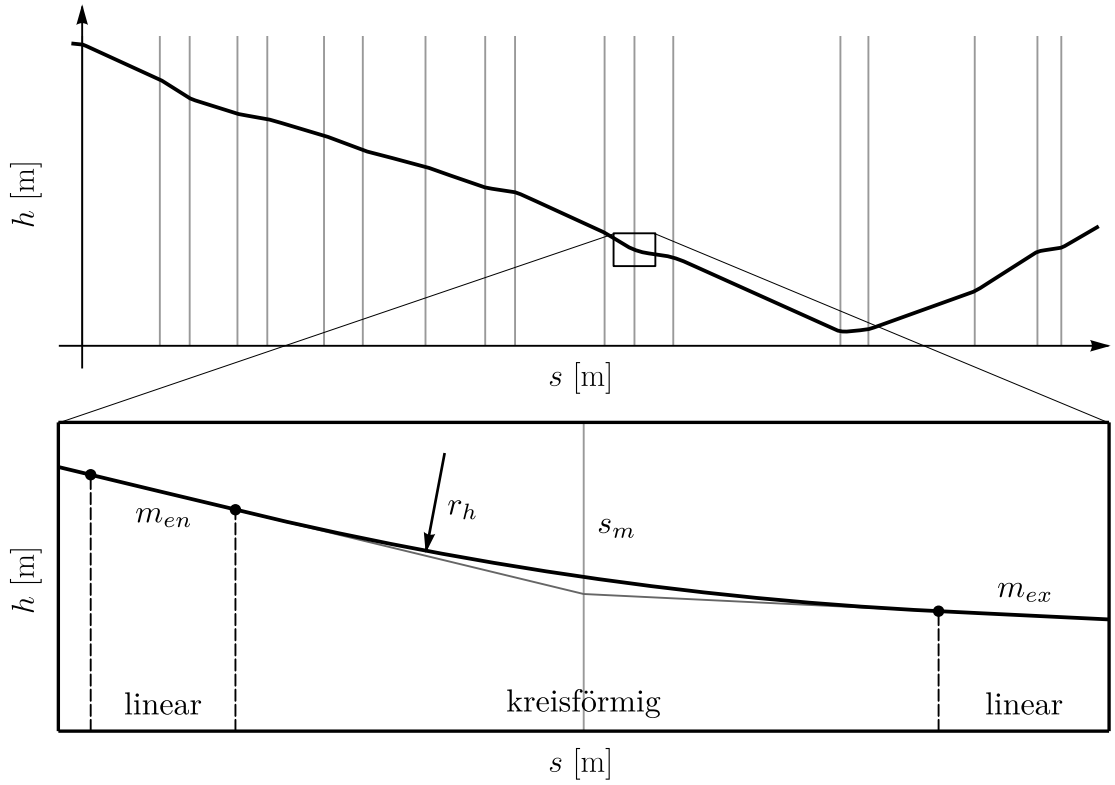

Abbildung 3.7: Höhenprofil der Bahn von Whistler, Kanada (oben) und ein einzelner Übergang zwischen zwei Abschnitten mit konstanter Steigung (unten) mit den pro Übergang genannten Konstruktionsdaten.

3.1.2 Höhenprofil

Das Höhenprofil beschreibt die Höhe der Bahn in Meter über Meer entlang der Profillinie. Es ist in Abschnitte von konstanter Steigung unterteilt, wobei die Übergänge zwischen den einzelnen Abschnitten kreisförmig sind. Man vergleiche dazu die Abbildung 3.7, die das Höhenprofil der Bobbahn von Whistler in Kanada zeigt und im Detail einen solchen Abschnittsübergang aufzeichnet. Die Konstruktionsdaten [43, 48] nennen alle Stellen $\{s_m\}$, an denen sich die extrapolierten linearen Abschnitte schneiden. Die Konstruktionsdaten spezifizieren weiter zu jeder solchen Stelle die Steigung[6] des Höhenprofils m_{en} vor dem Übergang, jene nach dem Übergang m_{ex}, und den vorzeichenbehafteten Radius r_h des kreisförmigen Teilstücks. Der Radius ist positiv, wenn die Steigung von m_{en} zu m_{ex} zunimmt und im anderen Falle negativ. Zusammen mit einer bahnspezifischen initialen Höhe h° an einer Stelle s° definieren diese Konstruktionsdaten das Höhenprofil eindeutig.

Für die Parametrisierung $h(s)$ wird das Höhenprofil in Segmente zerlegt, die mit

[6] Die Konstruktionsdaten nennen eigentlich das Gefälle in Prozent. Die Steigung m ergibt sich mit

$$m = -\frac{\text{Gefälle}}{100\%}.$$

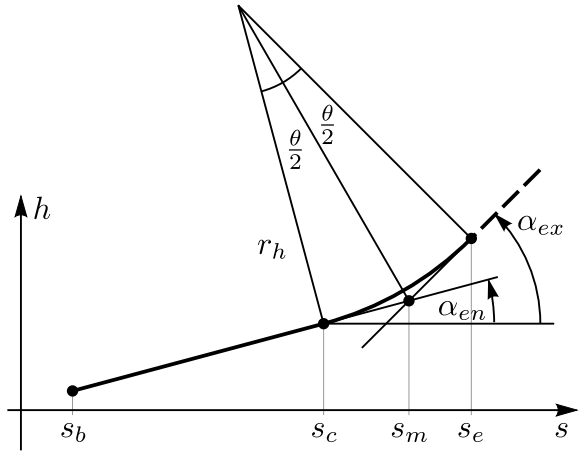

Abbildung 3.8: Skizze eines einzelnen Segments des Höhenprofils.

einem linearen Teilstück beginnen und das kreisförmige Übergangsstück beinhalten. Man vergleiche dazu die Abbildung 3.8. Die Aneinanderreihung solcher Segmente ergibt wieder das gesamte Höhenprofil mit Ausnahme des geraden Auslaufs nach dem letzten Segment. Dieser wird später separat behandelt. Das Tuple (s°, h°) dient als Startpunkt (s_b, h_b) des ersten Segments und der Segmentendpunkt (s_e, h_e) dient als Startpunkt (s_b, h_b) des jeweils folgenden Segments. Für die nachfolgende Parametrisierung werden in Abhängigkeit der Konstruktionsdaten und eines Startpunktes (s_b, h_b) die Grössen

$$\begin{aligned}
\bar{h}_m &= h_b + m_{en}(s_m - s_b), \\
\alpha_{en} &= \tan^{-1}(m_{en}, 1), & \alpha_{ex} &= \tan^{-1}(m_{ex}, 1), \\
\theta &= \alpha_{ex} - \alpha_{en}, & d &= |r_h| \tan(\theta/2), \\
s_c &= s_m - d\cos(\alpha_{en}), & h_c &= \bar{h}_m - d\sin(\alpha_{en}), \\
s_e &= s_m + d\cos(\alpha_{ex}), & h_e &= \bar{h}_m + d\sin(\alpha_{ex})
\end{aligned} \tag{3.8}$$

für jedes Segment bestimmt. Hierbei beranden die Stellen s_c und s_e den kreisförmigen Teil und $\bar{h}_m$ bezeichnet die Höhe des Schnittpunktes der extrapolierten linearen Abschnitte an der Stelle s_m. Man beachte, dass der Schnittpunkt $(s_m, \bar{h}_m)$ selbst nicht auf dem Höhenprofil liegt, sondern nur eine Konstruktionshilfe bildet. Ein einzelnes Segment lässt sich hiermit nach

$$h(s) = \begin{cases} h_b + m_{en}(s - s_b), & s \in [s_b; s_c] \\ h_c + \Delta h(s), & s \in (s_c; s_e] \end{cases} \quad \text{mit} \tag{3.9}$$

$$\Delta h(s) = r_h\cos(\alpha_{en}) - \operatorname{sign}(r_h)\sqrt{(r_h\cos(\alpha_{en}))^2 - 2r_h\sin(\alpha_{en})(s - s_c) - (s - s_c)^2}$$

parametrisieren. Durch die Aneinanderreihung der Segmente ist damit das ganze Höhenprofil parametrisiert.

Für Stellen s, die hinter dem Ende s_e des letzten Segments der Bahn, also im bereits erwähnten Auslauf der Bahn liegen, wird die Höhe nach $h(s) = h_e + m_{ex}(s - s_e)$ bestimmt, wobei sich s_e, h_e und m_{ex} auf Daten des letzten Segments beziehen.

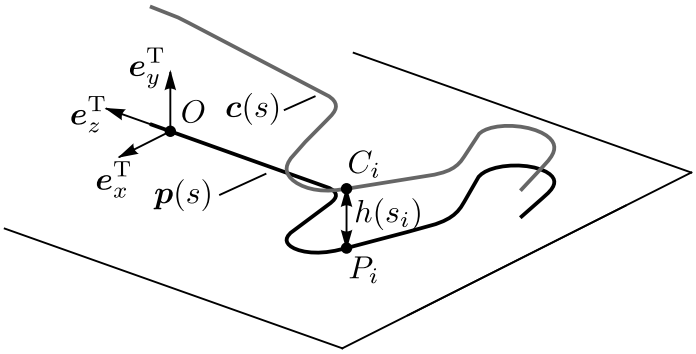

Abbildung 3.9: Profillinie $\boldsymbol{p}(s)$ und Streckenprofil $\boldsymbol{c}(s)$ der ersten vier Kurven von der Bahn in Whistler, Kanada.

3.1.3 Streckenprofil

Das Streckenprofil beschreibt den Verlauf der Bahn durch den dreidimensionalen Raum und ist in Abbildung 3.9 wiederum für die ersten vier Kurven in Whistler skizziert. Basierend auf dem Höhenprofil $h(s)$ nach Gleichung (3.9) und der Profillinie ${}_{\mathrm{T}}\boldsymbol{p}(s)$ aus Gleichung (3.7) wird das Streckenprofil ${}_{\mathrm{T}}\boldsymbol{c}(s) \in \mathbb{R}^3$ mit

$$ {}_{\mathrm{T}}\boldsymbol{c}(s) = {}_{\mathrm{T}}\boldsymbol{p}(s) + {}_{\mathrm{T}}\boldsymbol{e}_y^{\mathrm{T}} h(s) = {}_{\mathrm{T}}\boldsymbol{r}_{OC}(s) = \begin{pmatrix} p_x(s) \\ h(s) \\ p_z(s) \end{pmatrix} \tag{3.10} $$

zusammengesetzt. Diese Parametrisierung ist im Bahnmodellierungscode implementiert. Für die später folgende Positionierung der Bahnquerschnitte entlang dem Streckenprofil wird ein Satz von Streckenprofilpunkten $(s_i, {}_{\mathrm{T}}\boldsymbol{r}_{OC_i}, \beta_i)$ an vorgegebenen Positionen $\{s_i\}$ in regelmässigen Abständen $\Delta s = s_{i+1} - s_i = 0.5\mathrm{m}$ evaluiert und abgelegt. Bei einem *Streckenprofilpunkt* $(s_i, {}_{\mathrm{T}}\boldsymbol{r}_{OC_i}, \beta_i)$ handelt es sich um ein Tupel, das aus der Stelle s_i der Parametrisierungsevaluation besteht, dem Ortsvektor ${}_{\mathrm{T}}\boldsymbol{r}_{OC}(s_i)$ des entsprechenden Punktes C_i auf dem Streckenprofil und dem Winkel $\beta(s_i)$, der die Richtung der Profillinie ${}_{\mathrm{T}}\boldsymbol{p}$ im zu C_i korrespondierenden Punkt P_i auf der Profillinie bezeichnet. Man vergleiche dazu auch Abbildung 3.9. Der Satz von evaluierten Streckenprofilpunkten wird in der Folge auch *diskretes Streckenprofil* genannt.

3.1.4 Querschnittsgeometrie

Zuletzt beschreiben die Konstruktionsdaten einen Satz von Querschnittgeometrien in regelmässigen Abständen Δs entlang des Streckenprofils. Für die Bahn von Whistler ist pro halbem Meter eine Geometrie spezifiziert [41, 40], im Falle von Sochi alle zwei Meter [46, 47]. Abbildung 3.10 zeigt eine allgemeine Querschnittsgeometrie. Sie wird in sieben Segmente unterteilt, wovon drei gerade und drei kreisförmig sind. Ein Segment ist im Allgemeinen elliptisch, degeneriert aber zu einer Geraden auf den geraden Kurvensegmenten eingangs und ausgangs jeder Kurve. Die acht Punkte, die die einzelnen Segmente beranden, werden mit den Zahlen *0* bis *7* bezeichnet.

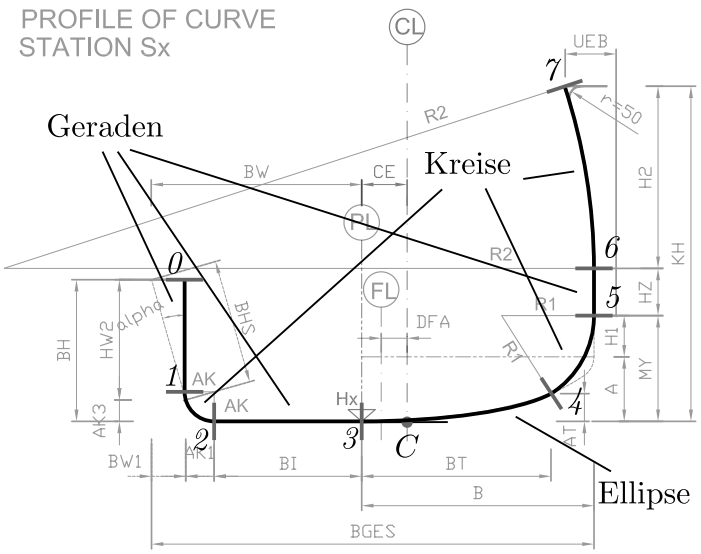

Abbildung 3.10: Die Querschnittsgeometrie einer Bobbahn. Hinterlegt sind die Bemassungen des Konstrukteurs.

Die Geometrie des Querschnitts wird anhand von 14 Bemassungen festgelegt. Sie sind in Abbildung 3.11 gezeigt. Die Eruierung dieser Masse aus den Konstruktionsdaten wird detailliert in Anhang A besprochen. Sie können nicht alle direkt aus den Konstruktionsdaten übernommen werden, weil die Konstruktionsdaten viele redundante Grössen beinhalten und gerundete Werte angeben. Die Rundungsungenauigkeiten würden zu inakzeptablen Deformationen einiger Querschnitte führen. Einige Masse werden deshalb fallspezifisch aus redundanten Grössen neu errechnet. Zu jedem Querschnitt wird weiter die Stelle s_i gelistet, anhand welcher seine Position auf dem Streckenprofil zugeordnet werden kann. In Abhängigkeit dieser Stelle s_i wird aus den Konstruktionsdaten der Profillinie, vergleiche Abschnitt 3.1.1, die Richtung c der Krümmung derjenigen Kurve bestimmt, in welcher der Querschnitt zu liegen kommt. Anhand dieses Parameters wird entschieden, ob die überhöhte Seite des Querschnitts rechts oder links liegt. Allein aus der Bemassung lässt sich nämlich nicht schliessen, ob es sich um eine Rechts- oder Linkskurve handelt.

Anhand der 14 Bemassungen aus Abbildung 3.11 und der Kurvenrichtung c wird jeder Querschnitt als nicht uniformer, rationaler B-Spline (NURBS) rekonstruiert. Für eine umfassende Behandlung von NURBS-Kurven und -Flächen sei auf [73] verwiesen. Wesentlich ist hier, dass die später entlang dem Streckenprofil aufgereihten Querschnittsplines zu einer Splinefläche verbunden werden können und damit eine Beschreibung der Bahnoberfläche vorliegen wird und, dass NURBS vom Grad Zwei Kegelschnitte (und Geraden), wie es die einzelnen Querschnittsegmente sind, exakt darstellen können. Man vergleiche dazu Kapitel 7 in [73]. Nach Abschnitt 4.2 in [73] ist eine NURBS-Kurve definiert über den Grad der zugrunde liegenden B-Spline Basisfunktionen, einen Satz von Kontrollpunkten mit zugeordneten Gewichten und einem sogenannten Knotenvektor. Während die Basisfunktionen mit den Kontrollpunkten und den dazugehörenden Gewichten die Geometrie der Kurve bestimmen, definiert der Knotenvektor im Wesentlichen den Verlauf des Kurvenparameters innerhalb der Kurve. Für jedes Querschnittsegment wird nachfolgend eine NURBS-Kurve

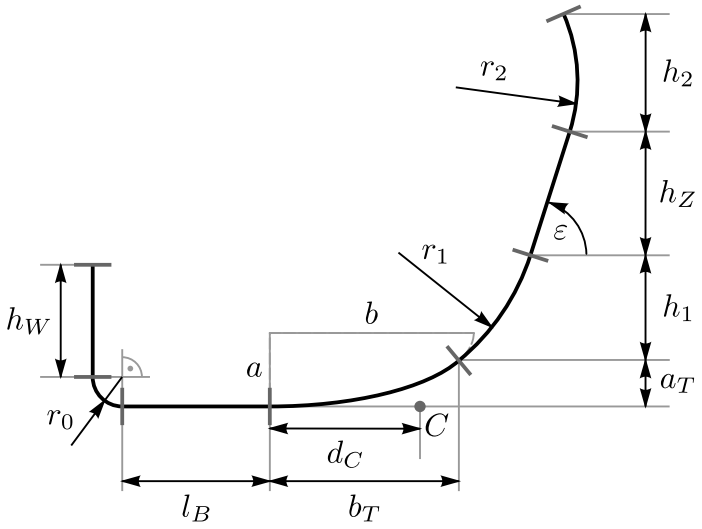

Abbildung 3.11: Definition der Querschnittsparameter wie sie für die Parametrisierung der Geometrie verwendet werden.

zweiten Grades konstruiert. Dazu werden die Positionen der Kontrollpunkte und ihrer Gewichte in Abhängigkeit der in Abbildung 3.11 eingeführten Bemassungen berechnet. Da alle Segmentkurven vom Grad Zwei sind, können sie zu einer Querschnittskurve konkateniert werden. Der Knotenvektor einer solchen Querschnittskurve ist so gewählt, dass jedes Querschnittsegment innerhalb einer Parametereinheit durchlaufen wird./% Für ausführlichere Details zum Knotenvektor sei auf Anhang A verwiesen.%/ Es resultiert, dass alle Querschnittskurven entlang dem Streckenprofil wieder vom Grad Zwei sind und dieselbe Anzahl Kontrollpunkte und denselben Knotenvektor haben. Das ist Voraussetzung für die in Abschnitt 3.2 erfolgende Generierung einer NURBS-Fläche aus den aufgereihten Querschnittskurven.

Jedes Querschnittsegment wird durch zwei der Punkte *0* bis *7* berandet, siehe Abbildung 3.10, weshalb alle diese Punkte als Kontrollpunkte für die jeweiligen Segmente dienen. Die Segmente werden vorerst in der $\boldsymbol{e}_x^{\mathrm{T}}$-$\boldsymbol{e}_y^{\mathrm{T}}$-Ebene konstruiert mit einem im Ursprung liegenden Punkt *3*, d.h. ${}_{\mathrm{T}}\boldsymbol{r}_{O3} = (0,0,0)^{\mathsf{T}}$. Die zu diesen *Randkontrollpunkten* gehörenden Gewichte $w_0 = w_1 = \cdots = w_7 = 1$ sind alle Eins. Die Positionen der Randkontrollpunkte ergeben sich zu

$$
\begin{aligned}
{}_{\mathrm{T}}\boldsymbol{r}_{O0} &= (-c(l_B + r_0), r_0 + h_W, 0)^{\mathsf{T}}, \quad {}_{\mathrm{T}}\boldsymbol{r}_{O4} = (c\,b_T, a_T, 0)^{\mathsf{T}}, \\
{}_{\mathrm{T}}\boldsymbol{r}_{O1} &= (-c(l_B + r_0), r_0, 0)^{\mathsf{T}}, \quad {}_{\mathrm{T}}\boldsymbol{r}_{O5} = (c(b_T + \Delta x_1), a_T + h_1, 0)^{\mathsf{T}} \\
{}_{\mathrm{T}}\boldsymbol{r}_{O2} &= (-c\,l_B, 0, 0)^{\mathsf{T}}, \quad {}_{\mathrm{T}}\boldsymbol{r}_{O6} = (c(b_T + \Delta x_2), a_T + h_1 + h_Z, 0)^{\mathsf{T}}, \\
{}_{\mathrm{T}}\boldsymbol{r}_{O3} &= (0,0,0)^{\mathsf{T}}, \quad {}_{\mathrm{T}}\boldsymbol{r}_{O7} = (c(b_T + \Delta x_3), a_T + h_1 + h_Z + h_2, 0)^{\mathsf{T}},
\end{aligned}
\tag{3.11}
$$

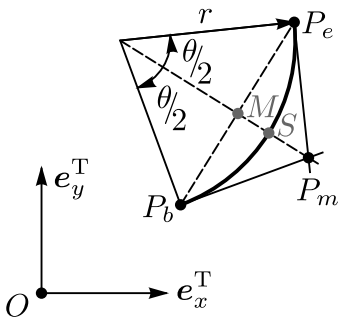

Abbildung 3.12: Skizze zur Konstruktion der Kontrollpunkte P_b, P_m und P_e von Kreissegmenten in Bahnquerschnitten.

wobei

$$\begin{aligned}
\Delta x_1 &= r_1 \left(\cos(\varepsilon)\sin(\theta) + \sin(\varepsilon)(1-\cos(\theta))\right), \\
\Delta x_2 &= \Delta x_1 + h_Z \frac{1}{\tan(\varepsilon)} \quad \text{und} \\
\Delta x_3 &= \Delta x_2 - r_2 \sin(\varepsilon) + \sqrt{r_2^2 \sin^2(\varepsilon) + 2 r_2 h_2 \cos(\varepsilon) - h_2^2}.
\end{aligned} \tag{3.12}$$

Hierbei bezeichnet ε den Winkel des geraden Segments zwischen den Punkten *5* und *6* und θ den Winkel des Kreissegments zwischen den Punkten *4* und *5*, vgl. Abbildungen 3.10 und 3.11. Der Winkel ε stammt aus den Konstruktionsdaten. Der Winkel θ berechnet sich nach Gleichung (A.6) in Anhang A./% Für die Herleitung von Δx_3 sei hier ebenfalls auf Anhang A verwiesen.%/

Zusätzlich zu den Randkontrollpunkten gibt es einen weiteren *inneren Kontrollpunkt* pro Querschnittsegment. Im Falle der geraden Segmente liegt dieser in der Mitte und hat ein Gewicht von Eins. Für Randkontrollpunkte P_b und P_e mit Gewichten $w_b = w_e = 1$ ergibt sich der innere Kontrollpunkt also nach

$${}_{\mathrm{T}}\boldsymbol{r}_{OP_m} = \frac{1}{2}({}_{\mathrm{T}}\boldsymbol{r}_{OP_b} + {}_{\mathrm{T}}\boldsymbol{r}_{OP_e}), \quad w_m = 1. \tag{3.13}$$

Für die kreisförmigen Segmente wird in der Folge ein Vorgehen beschrieben, indem im Wesentlichen nur die für eine Implementierung nötigen Formeln angegeben werden. Das Vorgehen basiert auf Abschnitt 7.3 von [73]. Für ausführlichere Details und geometrische Interpretationen sei hier ebendieses Original empfohlen. Es wird wieder von Randkontrollpunkten P_b und P_e mit Gewichten $w_b = w_e = 1$ ausgegangen. Abbildung 3.12 skizziert die Situation für das Kreissegment. Zuerst wird der Winkel θ des Kreissegments nach

$$\frac{\theta}{2} = c \, \sin^{-1}\left(\frac{\|{}_{\mathrm{T}}\boldsymbol{r}_{OP_e} - {}_{\mathrm{T}}\boldsymbol{r}_{OP_b}\|}{2r}\right)$$

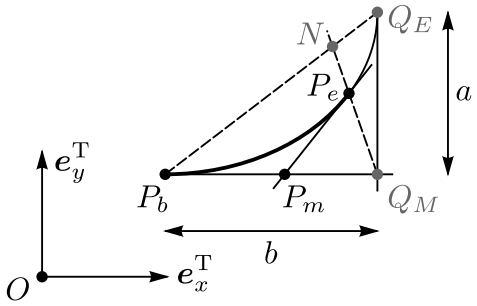

Abbildung 3.13: Skizze zur Konstruktion der Kontrollpunkte P_b, P_m und P_e des Ellipsensegments in einem Bahnquerschnitt. Die Punkte P_b, Q_M und Q_E bilden die Kontrollpunkte der Viertelellipse, die hilfsweise mitkonstruiert wird. Der Punkt N ist ein weiterer Hilfspunkt. Die Längen a und b bezeichnen die Halbachsen der Ellipse und sind in den Konstruktionsdaten angegeben.

bestimmt und danach die Punkte M und S

$$
\begin{aligned}
{}_{\mathrm{T}}\boldsymbol{r}_{OM} &= \frac{1}{2}({}_{\mathrm{T}}\boldsymbol{r}_{OP_b} + {}_{\mathrm{T}}\boldsymbol{r}_{OP_e}),\\
{}_{\mathrm{T}}\boldsymbol{n} &= \begin{bmatrix} 0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 1 \end{bmatrix} \frac{{}_{\mathrm{T}}\boldsymbol{r}_{OP_e} - {}_{\mathrm{T}}\boldsymbol{r}_{OP_b}}{\|{}_{\mathrm{T}}\boldsymbol{r}_{OP_e} - {}_{\mathrm{T}}\boldsymbol{r}_{OP_b}\|},\\
{}_{\mathrm{T}}\boldsymbol{r}_{OS} &= {}_{\mathrm{T}}\boldsymbol{r}_{OM} + c\,r(1-\cos(\theta/2)){}_{\mathrm{T}}\boldsymbol{n}
\end{aligned}
$$

konstruiert. Damit lässt sich schliesslich der Punkt P_m und sein Gewicht w_m bestimmen mit

$$
\begin{aligned}
w_m &= \cos(\theta/2) \quad \text{und} \quad s = \frac{w_m}{1+w_m},\\
{}_{\mathrm{T}}\boldsymbol{r}_{OP_m} &= \frac{1}{s}\,{}_{\mathrm{T}}\boldsymbol{r}_{OS} - \frac{1-s}{s}\,{}_{\mathrm{T}}\boldsymbol{r}_{OM}.
\end{aligned}
\tag{3.14}
$$

Der innere Kontrollpunkt für das elliptische Segment ergibt sich im Falle einer Degeneration zur Linie nach Gleichung (3.13). Für den Normalfall wird nachfolgend wiederum ein Vorgehen knapp beschrieben, so dass eine Implementierung möglich ist. Auf weitere Erklärungen wird verzichtet und auf Abschnitt 7.6 von [73] verwiesen, auf dem der Algorithmus basiert. Abbildung 3.13 illustriert die Situation beim Ellipsensegment. Ausgehend von Randkontrollpunkten P_b und P_e mit Gewichten $w_b = w_e = 1$, werden zuerst die Hilfspunkte Q_E, Q_M und N

$$
\begin{aligned}
{}_{\mathrm{T}}\boldsymbol{r}_{OQ_E} &= {}_{\mathrm{T}}\boldsymbol{r}_{OP_b} + (c\,b, a, 0)^{\mathsf{T}}, \quad w_E = 1,\\
{}_{\mathrm{T}}\boldsymbol{r}_{OQ_M} &= {}_{\mathrm{T}}\boldsymbol{r}_{OP_b} + (c\,b, 0, 0)^{\mathsf{T}}, \quad w_M = \frac{\sqrt{2}}{2},\\
{}_{\mathrm{T}}\boldsymbol{d} &= {}_{\mathrm{T}}\boldsymbol{r}_{OP_e} - {}_{\mathrm{T}}\boldsymbol{r}_{OQ_M} = (c\,(b_T - b), a_T, 0)^{\mathsf{T}},\\
s &= \frac{-(a\,b_T - a_T\,b)}{a(b_T - b) - a_T\,b},\\
{}_{\mathrm{T}}\boldsymbol{r}_{ON} &= {}_{\mathrm{T}}\boldsymbol{r}_{OP_e} + s\,{}_{\mathrm{T}}\boldsymbol{d}
\end{aligned}
$$

konstruiert, wobei b und a die Achsenabschnitte der Ellipse bezeichnen und zusammen mit b_T und a_T als Bemassungsparameter, siehe Abbildung 3.11, gegeben sind. Danach wird mit

$$u = \left(1 + \sqrt{\frac{\|{}_{\mathrm{T}}\boldsymbol{r}_{ON} - {}_{\mathrm{T}}\boldsymbol{r}_{OQ_E}\|}{\|{}_{\mathrm{T}}\boldsymbol{r}_{OP_b} - {}_{\mathrm{T}}\boldsymbol{r}_{ON}\|}}\right)^{-1}$$

das Gewicht w_m und die Position ${}_{\mathrm{T}}\boldsymbol{r}_{OP_m}$

$$\begin{aligned} w_m &= (1-u)w_b + u\, w_M \\ {}_{\mathrm{T}}\boldsymbol{r}_{OP_m} &= \frac{1}{w_m}\left((1-u)w_b\, {}_{\mathrm{T}}\boldsymbol{r}_{OP_b} + u\, w_M\, {}_{\mathrm{T}}\boldsymbol{r}_{OQ_M}\right) \end{aligned} \tag{3.15}$$

des inneren Kontrollpunktes bestimmt. Damit sind alle Kontrollpunkte aller in den Konstruktionsdaten angegebenen Querschnitte berechnet und deren NURBS-Kurven bestimmt.

Es kommt vor, dass das diskrete Streckenprofil Punkte $(s_i, {}_{\mathrm{T}}\boldsymbol{r}_{OC_i}, \beta_i)$ an mehr Stellen $\{s_i\}$ enthält, als Querschnitte gegeben sind. In diesem Fall werden Querschnittskurven interpoliert, in dem an einer Stelle s_i zwischen zwei Querschnitten, deren Kontrollpunkte und Gewichte linear interpoliert werden. Es resultiert also zu jedem zuvor evaluierten Streckenprofilpunkt $(s_i, {}_{\mathrm{T}}\boldsymbol{r}_{OC_i}, \beta_i)$ ein Querschnittspline.

Gemäss Eigenschaft 4.9 aus Kapitel 4 in [73] sind die rationalen Basisfunktionen einer NURBS-Kurve invariant unter affinen Transformationen. D.h. für eine affine Transformation der Kurve wird die affine Abbildung auf die Kontrollpunkte angewandt. Damit werden die Querschnittskurven entlang dem Streckenprofil aufgereiht. Zuerst werden alle Kontrollpunkte um $-c\, d_C$ in $\boldsymbol{e}_x^{\mathrm{T}}$-Richtung verschoben, danach mit der Rotationsmatrix (3.6) um den Winkel β_i um die $\boldsymbol{e}_y^{\mathrm{T}}$-Achse rotiert. Und schliesslich werden nochmals alle Kontrollpunkte um den Vektor ${}_{\mathrm{T}}\boldsymbol{r}_{OC_i}$ verschoben. Damit liegt schliesslich der Aufhängungspunkt C eines Querschnitts, siehe Abbildung 3.11, auf dem Streckenprofil und die Profillinie steht senkrecht zur Ebene des Querschnitts. Abbildung 3.14 zeigt einen Ausschnitt von aufgereihten Querschnitten für die Bahn von Whistler.

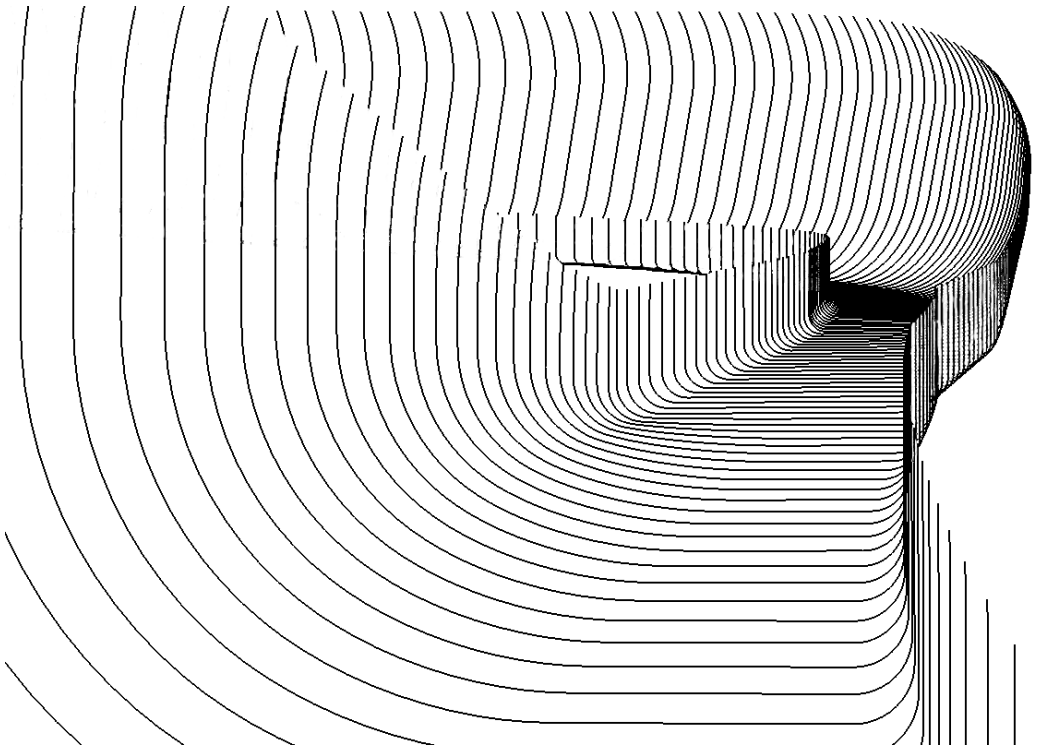

Abbildung 3.14: Ausschnitt von entlang dem Streckenprofil aufgereihten Querschnittskurven.

3.2 Oberflächenrekonstruktion

Nachdem anhand der Konstruktionsdaten Parametrisierungen der spezifizierten Geometrien zusammengesetzt wurden, wird jetzt mit Hilfe der kommerziellen Software *Maya* [5] die Bahnoberfläche digital rekonstruiert. Die mittels Python API [6] erstellten und anhand der Tupel $(s_i, {}_{\mathrm{T}}\boldsymbol{r}_{OC_i}, \beta_i)$ aufgereihten Querschnittsplines werden zu einer NURBS-Fläche zusammengesetzt. Die benutzte Methode nennt sich im Englischen *skinning* oder *lofting* und ist im Abschnitt 10.3 von [73] beschrieben. Dabei wird eine Reihe von aufeinander folgenden Querschnitten, welche alle identisch viele Kontrollpunkte haben, denselben Knotenvektor teilen und vom gleichen Grad sind, zu einer Splinefläche verbunden. Diese entsteht, indem in Längsrichtung zwischen den korrespondierenden Stützpunkten zweier Querschnitte Splinekurven erstellt werden. Die resultierende Oberfläche ist in Abbildung 3.15 mit breiten Linien dargestellt. Sie ist vom Grad zwei in Richtung der Querschnitte und vom Grad eins in Längsrichtung. Das bedeutet, dass zwei aufeinanderfolgende Querschnitte linear interpoliert werden und damit die Punkte *0* bis *7* der jeweiligen Querschnitte durch Geraden verbunden sind.

Die Splineoberfläche wird in einem letzten Schritt trianguliert, wobei jedes Splinesegment von Einheitsparameterlänge durch eine bestimmte Anzahl Geraden approximiert wird. In dieser Arbeit wird jedes Querschnittsegment in fünf Teilstücke aufgeteilt und die Linien zwischen zwei aufeinanderfolgenden Querschnitten gedrittelt. Das resultie-

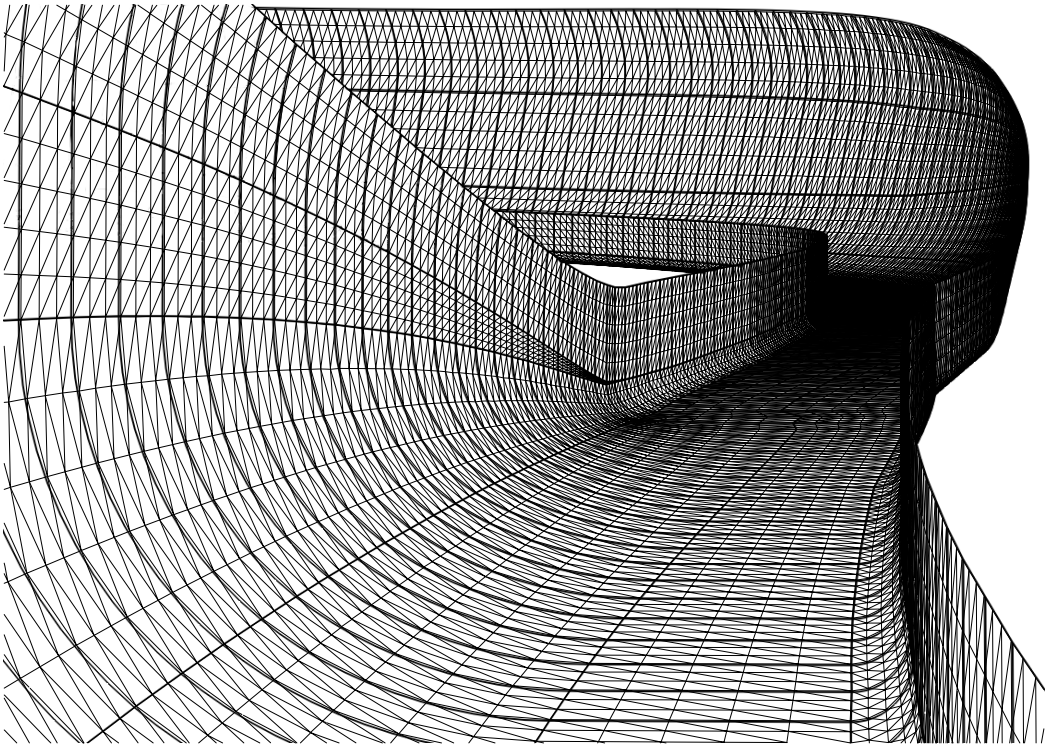

Abbildung 3.15: NURBS Oberfläche (breite) und dessen Tessellierung, das Dreiecksgitter (dünne Linien) einer Bobbahn.

rende Dreiecksgitter ist in Abbildung 3.15 in dünnen Linien gezeichnet und dient als Bahnmodell sowohl beim Rendern der Szenerie, als auch bei der Kollisionserkennung innerhalb der Physiksimulation. Mit der Unterteilung der Geraden in drei Teilstücke zwischen zwei Querschnitten resultieren Dreiecke mit solidem Seitenverhältnis. Dreiecke mit extremen Seitenverhältnissen könnten sonst sowohl beim Rendering, als auch bei der Kollisionserkennung zwischen Bobmodell und Bahn zu numerischen Problemen führen.

3.3 Berandung der Bahnoberfläche

Zusätzlich zur Bahnoberfläche selbst werden Elemente, die an die Eisoberfläche angrenzen, digital rekonstruiert. In den meisten Kurven wird am oberen Ende der Eisfläche ein Endanschlag gebaut, damit der Bobschlitten oben nicht aus der Bahn fahren kann. Weiter wird häufig die Seitenwand des Eiskanals mittels einer darüber liegenden, bündig anschliessenden Wand überhöht. Abbildung 3.16 zeigt links ein Photo der Bahn von Sochi und rechts denselben Bahnausschnitt als Screenshot. Man sieht beidseits der Bahn die Zusatzwände. Am oberen Bildrand sowie innerhalb der nächsten Kurve sind die Endanschläge zu erkennen. Zudem werden für ein Rennen alle weiter unten liegenden Starteinfahrten, z.B. Damenstart, Juniorenstart und Rodelstarts, mit temporären Wänden verschlossen. Abbildung 3.17 zeigt links ein Photo einer solchen Startzufahrt und rechts ein Screenshot desselben Bahnausschnitts. Für einige Beispielbilder der Bahn von Whistler sei auf Seite

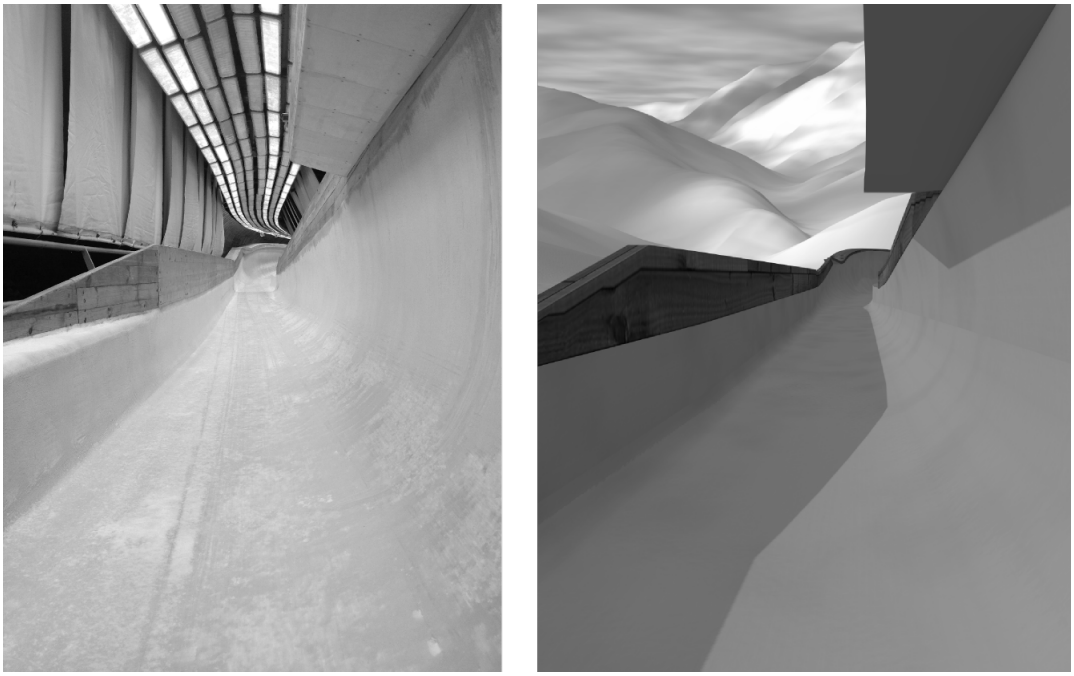

Abbildung 3.16: Randelemente zur Eisfläche. Endanschläge oberhalb von allen Kurven und zusätzliche Wände auf beiden Seiten der Bahn zur Überhöhung der Kanaltiefe. Links ein Foto der Bahn von Sochi, rechts derselbe Bahnabschnitt in der Simulation. (Quelle: Foto von Baumann Gregor, 2013.)

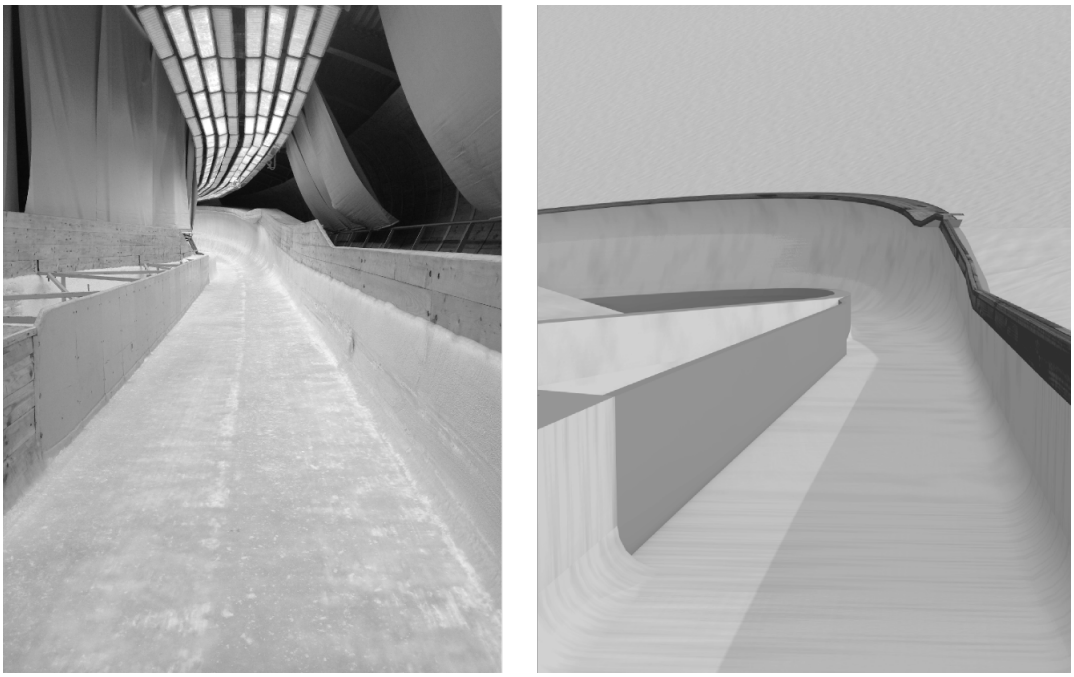

Abbildung 3.17: Randelemente zur Eisfläche. Bretterverschlag bei alternativen Starteinfahrten in die Bahn. Links ein Foto der Bahn von Sochi, rechts derselbe Bahnabschnitt in der Simulation. (Quelle: Foto von Baumann Gregor, 2013.)

316ff von [86] verwiesen. Alle diese Randelemente müssen bei der Kollisionserkennung mitberücksichtigt werden, da ein Schlitten gegen diese Elemente fahren kann.

Anhand von Videos, siehe [18], werden die Positionen und Dimensionen der Randelemente geschätzt und in Datensätzen abgelegt. Dazu werden Standbilder der Videos mit Perspektiven auf die modellierte Bahnoberfläche aus ähnlichen Blickwinkeln verglichen. Wählt man sich, beispielsweise für den Beginn eines Elements, einen Querschnitt im Modell aus, ist die Position durch die zum Querschnitt gehörende Stelle s_i festgelegt. Durch Ausprobieren oder relativen Abgleich zu den Bemassungen der betroffenen Querschnitte werden die Dimensionen der Elemente geschätzt und aufgenommen. Es bleibt zu sagen, dass für die Fahrdynamik des Schlittens und für den Eindruck eines Simulatorpiloten grobe digitale Replikationen der Elemente ausreichen und deshalb auf Detailtreue bewusst verzichtet wird.

Aus den Datensätzen werden die Querschnitte der einzelnen Randelemente als lineare Splinekurven generiert, mit den Bahnquerschnitten zusammen entlang dem Streckenprofil $\boldsymbol{c}(s)$ positioniert, und schliesslich, analog zu den Bahnquerschnitten, zu Splineoberflächen verbunden. Die triangulierten Gitter der Elemente dienen neben dem Bahnmodell als Kollisionsgeometrien in der Physiksimulation.

Kapitel 4

Mehrkörpermodell eines Bobschlittens

Dieses Kapitel beschreibt den Aufbau des Bobmodells, das aus neun starren Körpern zusammengesetzt ist und über mehrere reibungsbehaftete, harte Kontakte mit dem Bahnmodell aus Kapitel 3 interagiert. Viele Parameter des Bobmodells stammen aus CAD Daten und Messreihen des *Citius* Projektes [4]. Einige müssen zusätzlich bestimmt werden und werden mit Hilfe von erfahrenen Bobpiloten im Simulator evaluiert.

4.1 Kinematik

Das mechanische Modell des Bobschlittens besteht aus neun Starrkörpern, die über acht skalare Rotationsgelenke verbunden sind. Abbildung 4.1 zeigt den Kinematikgraphen des Modells ausgehend vom Ursprung eines Inertialsystems. Ein *Mitteltrennungsgelenk* teilt einen Bobschlitten in einen *Vorder-* und einen *Hinterteil*. Abbildung 4.2 zeigt das Modell des Hinterteils. Es besteht aus dem Hinterboot, den zwei Hinterkufen und einem Körper, der *Mannschaft* genannt wird und die Oberkörper der Insassen repräsentiert. Das Hinterboot beinhaltet die Unterleibe der Mannschaft und die starr fixierte Hinterachse. Es ist der Basiskörper des Modells und über ein Freigelenk mit dem Ursprung O des inertialen Koordinatensystems $(\boldsymbol{e}_x^{\mathrm{T}}, \boldsymbol{e}_y^{\mathrm{T}}, \boldsymbol{e}_z^{\mathrm{T}})$ verbunden. Die Position ${}_{\mathrm{T}}\boldsymbol{r}_{OS}$ beschreibt den Schwerpunkt S des Basiskörpers und ${}_{\mathrm{T}}\boldsymbol{v}_S$ dessen Translationsgeschwindigkeit. Die Orientierung des Basiskörpers ist mit einem Einheitsquaternion $\boldsymbol{\zeta}_{\mathrm{TK}}$ parametrisiert, wobei der Index K das körperfeste Koordinatensystem $(\boldsymbol{e}_x^{\mathrm{K}}, \boldsymbol{e}_y^{\mathrm{K}}, \boldsymbol{e}_z^{\mathrm{K}})$ des Basiskörpers referenziert. Die Winkelgeschwindigkeit des Basiskörpers wird mit ${}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{TK}}$ bezeichnet. Jede Hinterkufe sowie die Mannschaft sind über Drehgelenke mit dem Hinterboot verbunden, woraus sich die drei Freiheitsgrade φ_1, φ_2 und δ ergeben. Das Mitteltrennungsgelenk mit Freiheitsgrad α verbindet das Hinterboot mit dem Vorderboot. Abbildung 4.3 zeigt das Modell des Bobschlittenvorderteils, welches das Vorderboot beinhaltet. Im Vorderboot ist der kinematisch aktuierte Lenkkopf angebracht, der um den Steuerwinkel β rotiert. Der Lenkkopf trägt die Vorderachse mit den zwei Vorderkufen. Sowohl die Vorderachse wie auch die Kufen sind drehbar gelagert. Ihre Verdrehungen werden mit γ, φ_3 bzw. φ_4 bezeichnet. Die 13 nicht aktuierten Freiheitsgrade des Bobschlittenmodells werden in den generalisierten

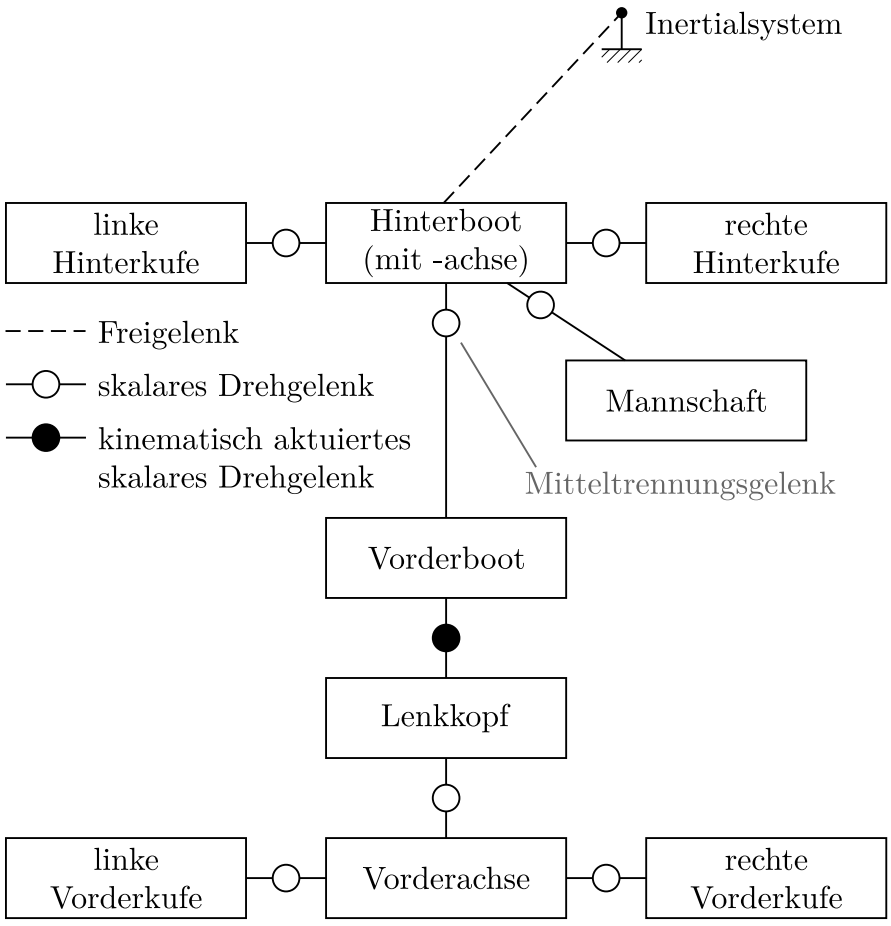

Abbildung 4.1: Kinematischer Baum des Bobschlittenmodells.

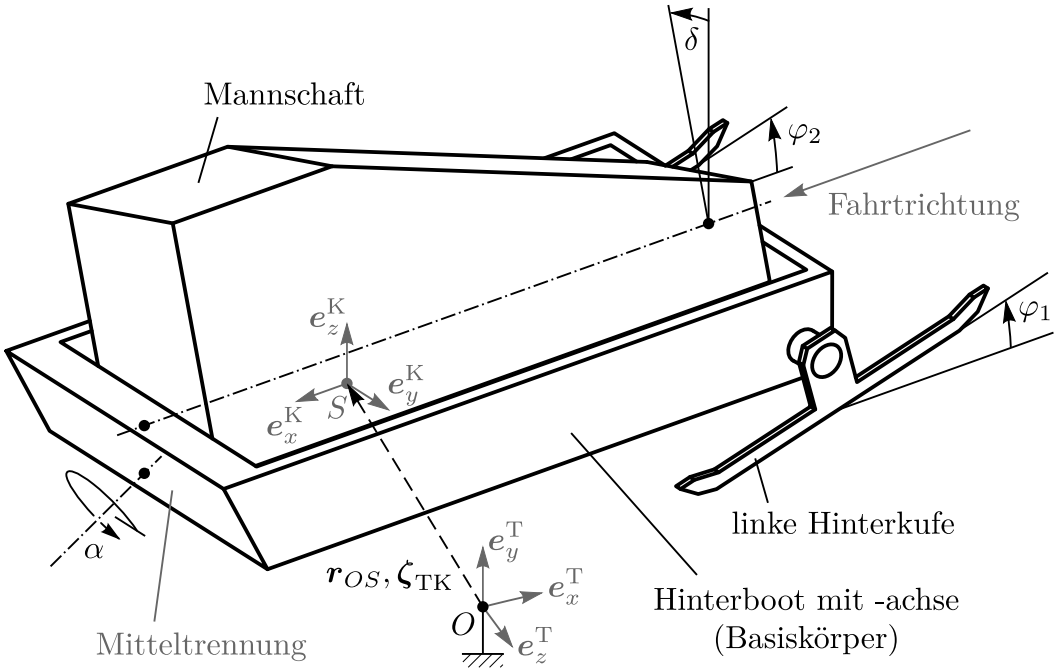

Abbildung 4.2: Freiheitsgrade des Hinterteils des Bobschlittenmodells.

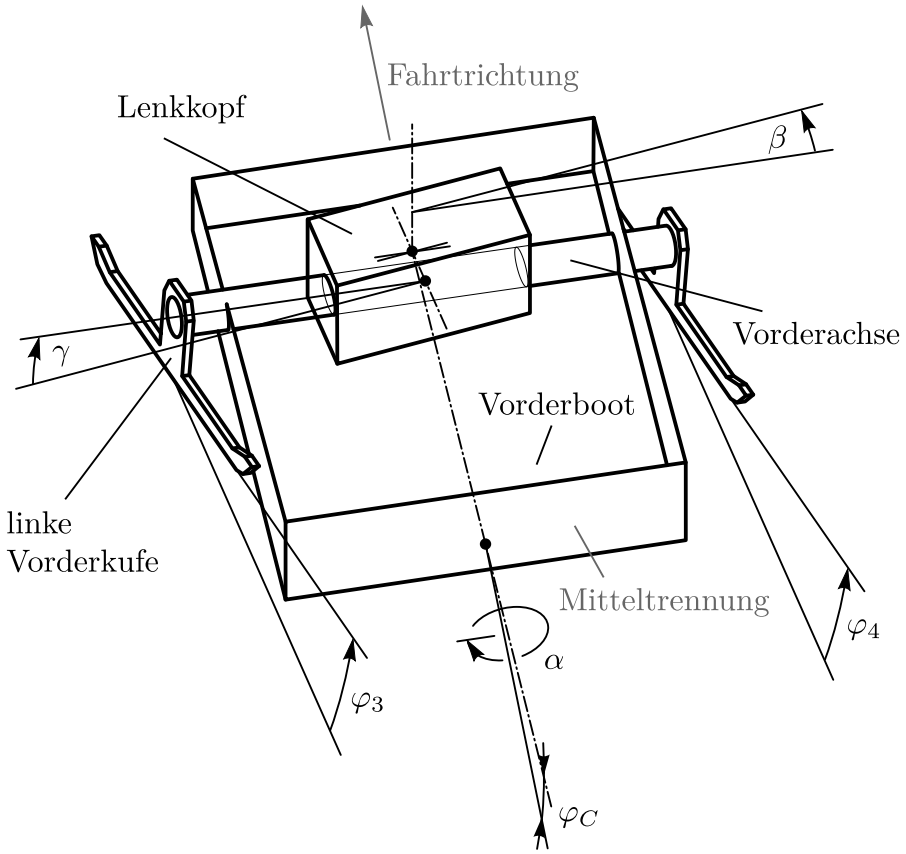

Abbildung 4.3: Freiheitsgrade des Vorderteils des Bobschlittenmodells.

Koordinaten $\boldsymbol{q}$ und Geschwindigkeiten $\boldsymbol{u}$

$$\begin{aligned} \boldsymbol{q} &= ({}_{\mathrm{T}}\boldsymbol{r}_{OS}, \boldsymbol{\zeta}_{\mathrm{TK}}, \alpha, \varphi_1, \varphi_2, \delta, \gamma, \varphi_3, \varphi_4)^{\mathsf{T}}, \\ \boldsymbol{u} &= ({}_{\mathrm{T}}\boldsymbol{v}_S, {}_{\mathrm{K}}\boldsymbol{\omega}_{\mathrm{TK}}, \dot{\alpha}, \dot{\varphi}_1, \dot{\varphi}_2, \dot{\delta}, \dot{\gamma}, \dot{\varphi}_3, \dot{\varphi}_4)^{\mathsf{T}} \end{aligned} \tag{4.1}$$

zusammengefasst. Zusammen mit dem kinematisch aktuierten Gelenk mit Steuerwinkel β ergeben sich total 14 Lagezustände. Man beachte, dass es sich bei $\boldsymbol{\zeta}_{\mathrm{TK}}$ um ein Einheitsquaternion handelt. Obwohl die generalisierten Koordinaten $\boldsymbol{q}$ 14 Einträge haben, beschreiben sie damit nur 13 Freiheitsgrade. Für eine umfassende Behandlung von Quaternionen als Parametrisierung von Rotationen seien [77, 64] empfohlen. In [65] ist zudem eine physikalische Interpretation der Bindung auf Einheitslänge beschrieben.

4.2 Klassische Kraftelemente

Im Modell kommen vier verschiedene klassische[7] Kraftelemente vor. Zum Einen wirken in den Drehgelenken des Modells lineare oder stückweise lineare Rückstellkräfte. Zum Anderen beinhaltet das Bobmodell Gravitationskräfte und einen aerodynamischen Widerstand.

Im Schwerpunkt S_i eines jeden Körpers i wirkt eine Gravitationskraft $\boldsymbol{F}_{G_i} = -m_i g \boldsymbol{e}_y^{\mathrm{T}}$, wobei die Massen m_i und die Positionen der Schwerpunkte bekannt sind aus Messungen und CAD Zeichnungen, und g die Erdbeschleunigung bezeichnet. Der aerodynamische Widerstand ist mit einer Kraft $\boldsymbol{F}_A = -c_w A \rho \boldsymbol{v}_S \|\boldsymbol{v}_S\|$ modelliert. Sie greift repräsentativ im Schwerpunkt S des Hinterboots an. Der kombinierte Parameter $c_w A$ wurde im Rahmen des *Citius* Projektes [4] gemessen und ρ bezeichnet die Luftdichte.

Mit Ausnahme des kinematisch aktuierten Steuergelenkes werden in allen Drehgelenken Rückstellkräfte modelliert. In der Mitteltrennung und zwischen Lenkkopf und Vorderachse wird ein stückweise lineares Feder-Dämpferelement verwendet, wie es die Unterabschnitte 5.4.3 bzw. 5.3.3 in [4] nahelegen. Die folgenden Erläuterungen sind [4] entnommen. Ein Modell des realen Federmechanismus in der Mitteltrennung ist in Abbildung 4.4 links gezeichnet. Ein masseloser Hebel der Länge l_α ist am unteren Ende drehbar gelagert, was dem Mitteltrennungsgelenk entspricht. Das obere Ende des Hebels ist zwischen zwei horizontal liegenden Federn mit Steifigkeit c angebracht. Die Federn sind an deren äusseren Ende fixiert und liegen am Hebel nur auf. Bei genügend grossen Auslenkungen verliert der Hebel also den Kontakt zu einer Feder. Die Federn können über die fixe Verschiebung z_k des äusseren Endes vorgespannt werden und sie sind so verbaut, dass im Falle von $z_k = 0$ die entspannten Federn gerade an den vertikal stehenden Hebel heranreichen. Im nicht vorgespannten Fall wirkt also für alle Auslenkungen nur gerade eine Feder. Abhängig von

[7] Mit *klassisch* soll in diesem Zusammenhang angedeutet werden, dass es sich um sehr gebräuchliche, einfache Kraftgesetze handelt. Und insbesondere sollen diese Kraftelemente von den im Abschnitt 4.3 behandelten Kontakten separiert werden. Die zu den Kontakten (oder einseitigen Bindungen) gehörenden Bindungskräfte wären in diesem Sinne die *nicht klassichen Kraftelemente*, für deren Beschreibung man mengenwertige Funktionen bzw. Normalkegelinklusionen benötigt.

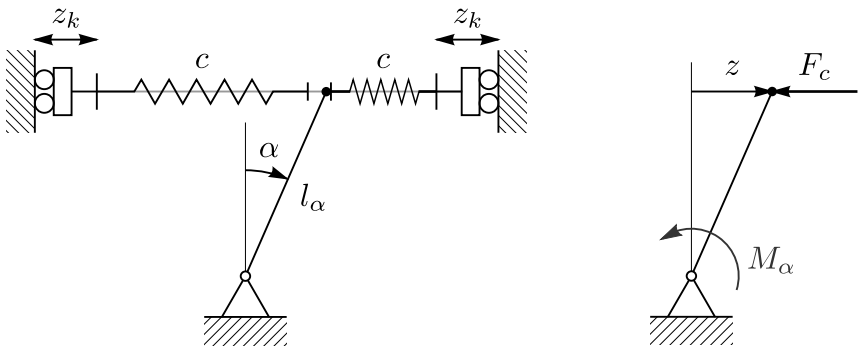

Abbildung 4.4: Modell des Rückstellfedermechanismus des Mitteltrennunggelenks.

der Auslenkung α des Hebels resultiert eine Federkraft F_c nach

$$\begin{aligned} z(\alpha) &= l_\alpha \sin(\alpha), \\ F_c(z) &= \begin{cases} -c(z_k - z), & z < -z_k \\ -c(z_k - z) + c(z + z_k) = 2cz, & z \in [-z_k, z_k] \\ c(z + z_k), & z > z_k \end{cases} \end{aligned} \tag{4.2}$$

und diese induziert im Drehgelenk ein Rückstellmoment $M_\alpha = \cos(\alpha) l_\alpha F_c$. Da die Auslenkung α auf relativ kleine Winkel $|\alpha| \leq 12°$ [4] beschränkt ist, werden die Näherungen $\sin(\alpha) \approx \alpha$ /rad und $\cos(\alpha) \approx 1$ verwendet, womit das Rückstellmoment zu

$$M_\alpha(\alpha) = \begin{cases} -c\, l_\alpha (z_k - l_\alpha \alpha \text{ /rad}), & \alpha < -\alpha_k \\ 2c\, l_\alpha^2\, \alpha \text{ /rad}, & \alpha \in [-\alpha_k, \alpha_k] \\ c\, l_\alpha (l_\alpha \alpha \text{ /rad} + z_k), & \alpha > \alpha_k \end{cases} \tag{4.3}$$

wird, wobei α_k nach

$$\sin(\alpha_k) = \frac{z_k}{l_\alpha} \tag{4.4}$$

den Grenzwinkel bezeichnet für den der Hebel von einer Feder abhebt. Das Rückstellmoment in der Mitteltrennung folgt damit näherungsweise einem stückweise linearen Kraftgesetz, wie es in Abbildung 4.5 skizziert ist. Als Näherung an ein komplexeres Modell schlägt [4] auch für die Rückstellung der Vorderachse ein solches vor und nennt Ersatzsteifigkeiten für die verbauten Teile. Der Winkel Φ in Abbildung 4.5 bezeichnet also entweder den Winkel α in der Mitteltrennung oder den Winkel γ der Pendelachse, vgl. dazu Abbildung 4.3. Das stückweise lineare Kraftgesetz ist charakterisiert über eine *innere* Steifigkeit c_1, eine *äussere* Steifigkeit c_2 und einen Winkel Φ_k, der die Stelle des Knicks definiert und damit den inneren vom äusseren Bereich trennt. Dieses Kraftgesetz ist eine ungerade Funktion, womit strikt $M_c(-\Phi) = -M_c(\Phi)$ gilt. Die real verbauten Teile in der Mitteltrennung und der Pendelachse sind austauschbar und bieten dem Piloten eine Möglichkeit, die Fahrdynamik zu verändern. Sämtliche real verfügbaren Federsätze und Setupoptionen sind über entsprechende Parametersätze im Modell nachempfunden und ihr Effekt kann damit

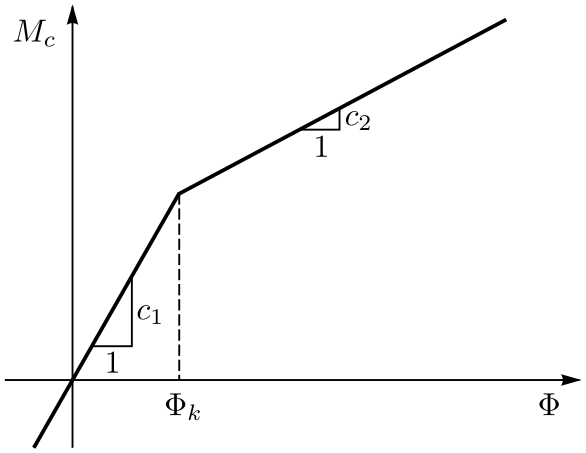

Abbildung 4.5: Stückweise lineares Kraftgesetz, wie es in der Mitteltrennung ($\Phi = \alpha$) und auf die Vorderachse ($\Phi = \gamma$) wirkt.

im Simulator getestet werden. Parallel zu diesen Rückstellmomenten wirkt ein lineares Dämpferelement in den Drehgelenken, das sämtlichen dissipativen Effekten Rechnung trägt.

Die Rückstellkräfte zwischen allen Kufen und den entsprechenden Achsen sind mit linearen Feder-Dämpferelementen modelliert. Weil die Drehfreiheitsgrade der Kufen hauptsächlich aus visuellen Gründen[8] modelliert wurden und ihr Einfluss auf die Bobschlittenbewegung als gering eingestuft wird, werden die Elementparameter grob abgeschätzt und nur anhand visueller Kriterien validiert. Die Rückstellung der Mannschaft innerhalb des Hinterbootes ist ebenfalls mit einem linearen Feder-Dämpferelement modelliert. Da keine Details über die Bewegung und Charakteristiken der Besatzung bekannt sind, werden mit diesem einfachsten Kraftelement am wenigsten neue Parameter eingeführt. Die Parameter bestimmen sich wiederum nach einem visuellen Kriterium und sind so gewählt, dass die Mannschaft visuell in sämtlichen Situationen eines Laufs innerhalb der sie umgebenden Hülle verbleiben.

4.3 Reibungsbehaftete harte Kontakte

Das Modell des Bobschlittens interagiert mit dem statischen Bahnmodell aus Kapitel 3 über mehrere reibungsbehaftete harte Kontakte. Die potentiellen Kontaktpunkte auf dem Bobschlittenmodell sind an bestimmten Positionen auf den jeweiligen Körpern vordefiniert. Die Positionen sind in Abbildung 4.6 illustriert. Zwei Kontaktpunkte befinden sich auf jeder Kufe, ein Kontaktpunkt auf jedem Abweiser und fünf Kontaktpunkte auf dem oberen

[8]Beim Rendern der Szenerie sollten die Kufen immer vollständig über dem Eis erscheinen und nicht visuell durch das Eis brechen. Wenn die Kufen keine Rotationsfreiheitsgrade hätten und nur über je einen einzelnen Kontaktpunkt modelliert wären, würden die Kufen in gekrümmten Bahnbereichen partiell durch das Bahngitter ragen und deshalb nicht dargestellt. Das wäre für den Benutzer störend, obwohl davon auszugehen ist, dass die Bewegung des Bobschlittens mit diesem alternativen Modell nicht markant abwiche.

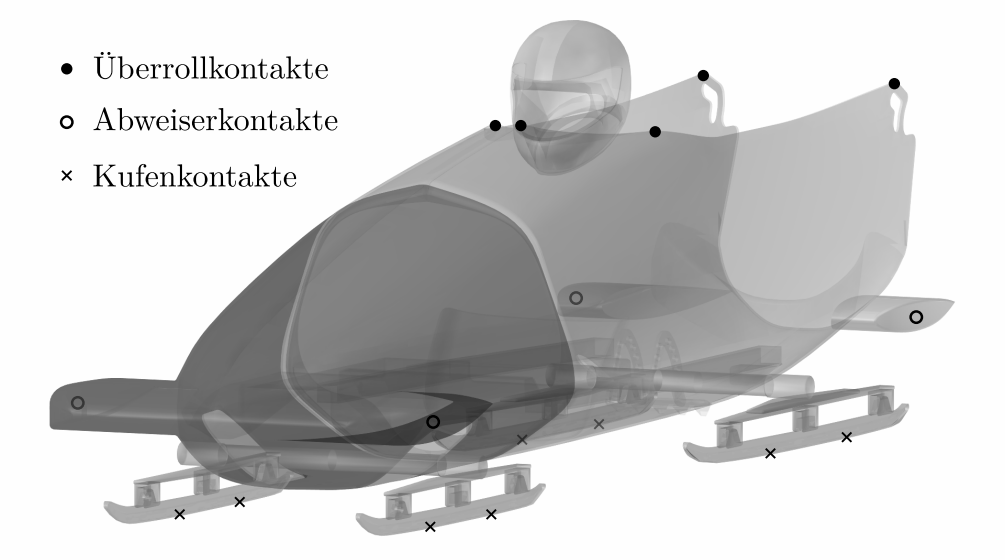

Abbildung 4.6: Vordefinierte Kontaktpunkte auf dem Bobschlitten.

Rand des Hinterboots, welche nur im Falle eines Überrollens des Schlittens schliessen. Die zwei vorderen Abweiserkontakte befinden sich auf dem Vorderboot, die hinteren zwei auf dem Hinterboot. Innerhalb der Kollisionsdetektion, siehe Unterabschnitt 2.6.1, in der Physiksimulation wird bestimmt, welche der potentiellen Kontaktpunkte geschlossen sind. In allen geschlossenen Kontakten wirken potentiell (impulsive) Kräfte im Kontaktpunkt auf den entsprechenden Starrkörper. Die Kräfte geschlossener Kontakte unterliegen den mengenwertigen Kraftgesetzen (2.32). Die darin vorkommende konvexe Menge $\mathscr{C}_f$, in der sich die verschiedenen Kontakttypen unterscheiden, kann als Kraftreservoir der möglichen Reibungskräfte interpretiert werden. Die Abweiserkontakte sind mit einer isotropen Coulombreibung modelliert, für welche $\mathscr{C}_f = \lambda_n \mu \mathscr{B}_2$ ist, wobei $\mathscr{B}_2$ den Einheitskreis im $\mathbb{R}^2$ bezeichnet. Die Reibkraft ist im Fall von Haften durch $\|\boldsymbol{\lambda}_f\| < \mu\lambda_n$ begrenzt. Im Fall von Gleiten ist sie proportional zum Druck in Normalrichtung $\|\boldsymbol{\lambda}_f\| = \mu\lambda_n$ und zeigt entgegen der Gleitgeschwindigkeit, siehe das Beispiel $\boldsymbol{\lambda}_{f1}$ in Abbildung 2.6. Algorithmus 4.1 zeigt die Implementierung der `prox-` und `getTangents`-Funktionen des Kontakttyps mit isotroper Coulombreibung, vgl. Tabelle 2.4. Die Überrollkontakte bestehen nur zu graphischen Zwecken. Da die Simulation und das Rendern der Szenerie im Falle eines Überrollens fortlaufen, soll das Bobmodell immer auf der Bahnoberfläche gehalten werden. Für den Piloten beendet ein Sturz den Lauf, weshalb die Kontakte nicht gesondert modelliert werden und das einfachste implementierte Reibgesetz, die eben besprochene isotrope Coulombreibung, verwendet wird. Für eine umfangreichere Behandlung harter einseitiger Bindungen mit Reibung und weitere Beispiele von Coulombreibung wird auf [25, 26, 28, 55] verwiesen.

Für die Kufenkontakte wird ein neues Reibmodell aufgesetzt, in dem eine spezifische konvexe Menge $\mathscr{C}_f$ kreiert wird. Das Reibmodell imitiert das anisotrope Verhalten der Kufen, das auch in der Realität beobachtet wird [11] und für ein Funktionieren des Lenkens notwendig ist. Das Modell ist von Möller [63] erdacht und auch in [4] erwähnt.

Algorithmus 4.1

```
# only partial implementation of this class is shown, i.e. its
# prox() function and its getTangents() function.
class IsotropicCoulombFrictionContact(AbstractContact):
  def prox(self, lambda_c, rG_clambda, dt):
    # normal part:
    l_n = min(lambda_c.x,0) # min returns the minimum
    # frictional part:
    [l_t, l_b] = self._proxCircle(rG_clambda - lambda_c, -self.mu*l_n)
    return Vector3d(l_n, l_t, l_b)
  def _proxCircle(self, l, radius):
    [l_t, l_b] = [l.y, l.z]
    l_f_squared = l_t*l_t + l_b*l_b
    if (l_f_squared > radius*radius): # project on circle border
      l_t *= radius / sqrt(l_f_squared) # sqrt .. takes
      l_b *= radius / sqrt(l_f_squared) #   square root
    return [l_t, l_b]

  def getTangents(self, normal):
    # return arbitrary tangential vectors to given normal.
    if (abs(normal.x) > abs(normal.y)):
      tangent = cross(Vector3d(0,1,0), normal).normalise()
    else:
      tangent = cross(Vector3d(1,0,0), normal).normalise()
      bitangent = cross(normal, tangent).normalise()
  return [tangent, bitangent]
```

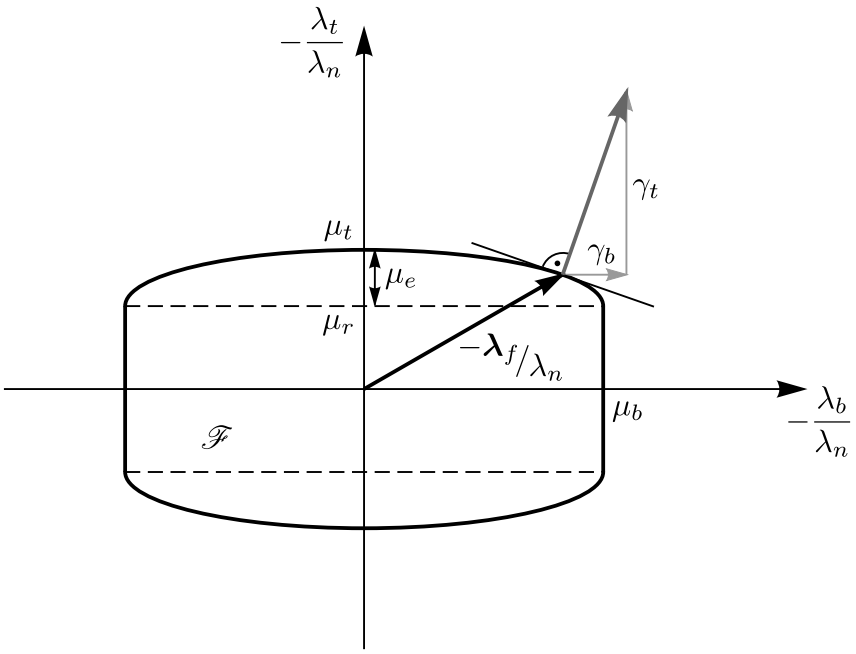

Abbildung 4.7: Reibkraftreservoir der Kufenkontakte.

Die Anisotropie führt zu einer Reibkraft, die bei Gleiten in Längsrichtung um einen Faktor fünf bis zehn kleiner ist als bei Gleiten in Querrichtung. Ein Reibkraftreservoir $\mathscr{C}_f = \lambda_n \mathscr{F}$ mit der heuristischen konvexen Menge $\mathscr{F}$, die in Abbildung 4.7 dargestellt ist, führt zu guten Resultaten. Die Menge $\mathscr{F}$ besteht aus einem rechteckigen inneren Teil und zwei Ellipsenhälften am oberen und unteren Rand. Man beachte, dass, wie im Falle der isotropen Coulombreibung, das Reservoir mit der Normalkomponente λ_n skaliert. Die Menge $\mathscr{F}$ ist breiter als hoch ($\mu_t < \mu_b$), was der geometrischen Anisotropie der Kufe Rechnung trägt. D.h. auf unebenem Eis wird die lange und schmale Kufe in Längsrichtung weniger und in Querrichtung stärker gebremst. Der rechteckige Innenbereich verstärkt diese Anisotropie, denn je schmaler die Halbellipsen sind, umso grösser wird, für eine gegebene Gleitrichtung $\boldsymbol{\gamma}_f = (\gamma_t, \gamma_b)^\mathsf{T}$, der Querkraftanteil λ_b in der Reibkraft $\boldsymbol{\lambda}_f$. Der laterale Reibkoeffizient μ_b ist zusätzlich nicht konstant, sondern liegt innerhalb eines fixen Intervalls $[\underline{\mu}_b, \overline{\mu}_b]$. Innerhalb von diesem Intervall variiert der Wert linear in Abhängigkeit der Normalkraft λ_n. Der Maximalwert wird ab einer vorzugebenden Normalkraft $\lambda_{n,\max}$ erreicht, also

$$\mu_b(\lambda_n) = \underline{\mu}_b + (\overline{\mu}_b - \underline{\mu}_b)\frac{\min(\lambda_n, \lambda_{n,\max})}{\lambda_{n,\max}}. \tag{4.5}$$

Diese zusätzliche Nichtlinearität modelliert das Einsinken der Kufe in das Eis unter starken Fliehkräften. Gräbt sich eine Kufe unter starkem Druck in das Eis, steigt die Spurhaltung überproportional, da in lateraler Richtung zusätzlich zur Reibung selbst auch eine geometrische Hürde überwunden werden muss. Die Implementierung der zum Kufenkontakttyp gehörenden `prox-` und `getTangents`-Funktionen sind in Algorithmus 4.2 gezeigt. Da der Bobschlitten während eines Laufs meistens vorwärts gleitet, ist die Relativgeschwindigkeit $\boldsymbol{\gamma}_f$ in tangentialer Richtung meistens ungleich Null und zeigt vom oberen Rand von $\mathscr{C}_f$

Algorithmus 4.2: (1. Teil)

```
# only partial implementation of this class is shown, i.e. its
# prox() function and its getTangents() function.
class SliderFrictionContact(AbstractContact):
  def prox(self, lambda_c, rG_clambda, dt):
    # normal part:
    l_n = min(lambda_c.x,0) # min returns the minimum
    # frictional part:
    l_n_max = dt * self.lambda_n_max # note that lambda refers to
    # P^ values in the calculation and P^_n_max=lambda_n_max*dt.
    mu_b = (self.mu_b_min + (self.mu_b_max - self.mu_b_min) *
            min(-l_n, l_n_max) / l_n_max)
    # transform ellipse to circle by
    # scaling bitangential direction
    rG_clambda.z *= mu_b/(self.mu_t-self.mu_r)
    lambda_c.z /= mu_b/(self.mu_t-self.mu_r)
    radius = -(self.mu_t-self.mu_r)*l_n
    heightRectangle = -self.mu_r*l_n
    [l_t, l_b] = self._proxScaledF(rG_clambda - lambda_c,
                                   radius, heightRectangle)
    # scale back the bitangential direction
    l_b *= mu_b/(self.mu_t-self.mu_r)
    return Vector3d(l_n, l_t, l_b)
  def _proxScaledF(self, l, radius, heightRectangle):
    # prox function of the scaled set F^ consisting of a
    #   - rectangular part in the middle.
    #   - half-circles at the top and
    #     the bottom border of the rectangle
    [l_t, l_b] = [l.y, l.z]
    widthRectangle = radius
    if(l_t > heightRectangle): # top half-circle subset
      l_e = l_t - heightRectangle
      [l_e, l_b] = self._proxCircle(l_e, l_b, radius)
      l_t = l_e + heightRectangle
    elif (-heightRectangle <= l_t <= heightRectangle):
      # inner rectangular subset
      l_b = self._proxInterval(l_b, -widthRectangle, widthRectangle)
    else: # (l_t < -heightRectangle) bottom half-circle subset
      l_e = l_t + heightRectangle
      [l_e, l_b] = self._proxCircle(l_e, l_b, radius)
      l_t = l_e - heightRectangle
    return [l_t,l_b]
```

Algorithmus 4.2: (Fortsetzung)

```
def _proxCircle(self, x, y, radius):
  v_squared = x*x + y*y
  if (v_squared > radius*radius): # project on circle border
    x *= radius / sqrt(v_squared) # sqrt .. takes
    y *= radius / sqrt(v_squared) #   square root
  return [x, y]
def _proxInterval(self, x, lowerLimit, upperLimit):
  # 'min' is supposed to return the minimum,
  # 'max' to return the maximum.
  return min(max(x,lowerLimit), upperLimit)

def getTangents(self, normal):
  # tangentAxis .. body related, fixed tangent direction.
  tangent = cross(self.tangentAxis, normal).normalise()
  bitangent = cross(normal, tangent).normalise()
  return [tangent, bitangent]
```

nach oben weg. Die Implementierung der `_proxScaledF`-Funktion in Algorithmus 4.2 prüft deshalb zuerst diesen Fall und erst danach die unwahrscheinlicheren Fälle von Haften und Rückwärtsgleiten. Bei anisotropen Reibgesetzen ist es im Allgemeinen aufwendig, die Stelle auf dem Rand der Menge zu finden, an der die Gleitgeschwindigkeit normal auf dem Rand steht. Der Sonderfall von Ellipsen lässt sich jedoch nach [66] effizient transformieren in einen skalierten Raum, wo die Ellipse zu einem Kreis wird. In diesem Raum ist die Projektion an die Menge numerisch günstig zu bewerkstelligen. Liegt ein zu projizierender Vektor ausserhalb des Kreises, wird seine Länge auf den Radius des Kreises skaliert und die Richtung beibehalten, siehe Algorithmus 4.1. Nach der Evaluation wird wieder zurück in den unskalierten Ursprungsraum transformiert.

Die gewählte Form des Reibkraftreservoirs $\mathscr{C}_f$ bietet drei Vorteile. Erstens lässt sich dieses Reibungsmodell numerisch effizient verarbeiten. Der sich in relativ engen Grenzen bewegende Betriebspunkt $\boldsymbol{\gamma}_f$ führt zweitens zu beständigen Kraftverläufen $\boldsymbol{\lambda}_f$, was die benötigten Iterationsschritte im Inklusionsproblem reduziert. Und drittens führt dieses Modell zu einer realistischen Eisinteraktion, was im folgenden Abschnitt 4.4 noch näher erläutert wird.

4.4 Parameterevaluierung mittels qualitativer Aussagen von Bobpiloten

Die meisten Parameter im Bobschlittenmodell sind bekannt aus CAD Daten oder Messreihen des Projektes *Citius* [4]. Einige wenige Parameter, wie zum Beispiel Masse und Position der Mannschaft, sind einmalig abgeschätzt und nicht weiter eruiert oder variiert worden.

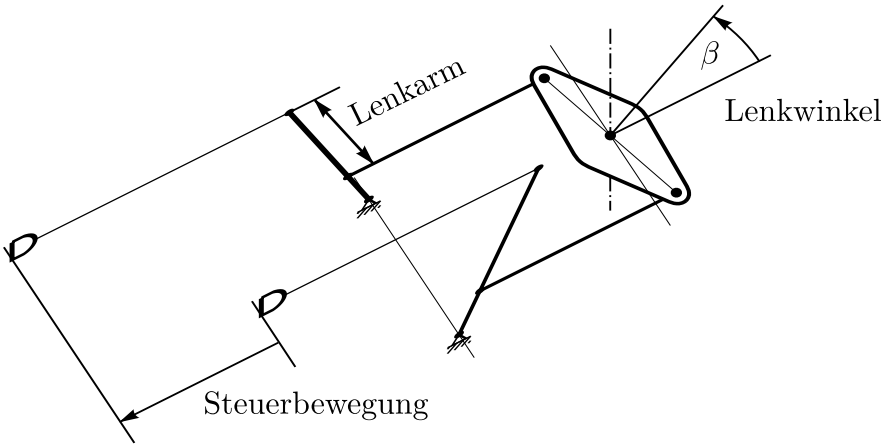

Abbildung 4.8: Steuermechanismus eines Bobschlittens.

Es verbleiben einige wenige Parameter, die nicht über Messungen zugänglich sind und sorgfältig abgeschätzt werden müssen, um die realen Effekte in den Modellen möglichst treu abzubilden. Einige solcher Parameter werden mit Hilfe von professionellen Bobpiloten und ihrem qualitativen Feedback über das Fahrgefühl im Simulator experimentell bestimmt.

Da üblicherweise jeder Pilot seinen eigenen Schlitten fährt, müssen deren individuellen Einstellungen im Simulatorschlitten nachgestellt werden, um die Aussagen über Fahr- und Echtheitsgefühl vergleichbar zu machen. Das betrifft insbesondere den Lenkmechanismus, der in Abbildung 4.8 skizziert ist. Jeder Pilot montiert die Steuerseile individuell an den Lenkarmen und variiert damit deren Länge. Dies verändert das Übersetzungsverhältnis von Lenkbewegung zu resultierendem Lenkwinkel. Je kürzer der Lenkarm, desto *direkter* die Lenkung. Mit einer direkteren Einstellung genügen also kleinere Lenkbewegungen, um einen bestimmten Lenkwinkel zu erreichen. Diese individuelle Einstellung bestimmt den Bereich von Lenkbewegungen, die ein Pilot während einer Fahrt tätigt. Dieser Bereich sollte im Simulator übereinstimmen. Das Montieren der Steuerseile ist zeitaufwändig und mühselig, weshalb dem Bobmodell ein Parameter *Steuerdirektheit* d hinzugefügt ist. Dieser Softwareparameter kann schnell geändert und für jeden Piloten gespeichert und wieder geladen werden. Der Lenkwinkel $\beta(i)$ berechnet sich nach

$$\beta(i) = \beta_{\max} \operatorname{sign}(i)|i|^{\frac{1}{d}}, \quad \beta_{\max} = 10^\circ, \quad i \in [-1, 1], \tag{4.6}$$

wobei i der normalisierte Sensorwert ist und $\beta_{\max}$ den maximal erreichbaren Lenkwinkel bezeichnet, der aus konstruktiven Gründen auf einen relativ kleinen Wert beschränkt ist. Ein neutrales Steuerverhalten wird durch $d = 1$ erreicht, eine Steuerdirektheit $0 < d < 1$ imitiert eine reduzierte und $d > 1$ eine erhöhte Direktheit. Man beachte, dass die Implementierung $\beta(i) \in [-\beta_{\max}, \beta_{\max}]$ garantiert. Dies ist eine Voraussetzung, da diese gegebenen Grenzen nicht virtuell umgangen werden sollen. Während der Experimente mit den Piloten haben alle Steuerdirektheiten $d \in [\frac{5}{8}, \frac{5}{7}]$ verwendet. Mit der Abstimmung dieser Einstellung auf die einzelnen Bobfahrer, stuften alle die nötigen Lenkbewegungen im Simulator als realistisch ein.

Ein erstes Experiment widmet sich dem Verhalten des virtuellen Schlittens über ein Stossen der Abweiser an der Bahnseitenwand. Die Experten bemerkten bald, dass der Rückstosseffekt nach einem Touchieren der Seitenwand mit den Abweisern entscheidend ist für die wahrgenommene Realität der Simulation. Im Experiment werden die Restitutionskoeffizienten ε_n in Normalrichtung der Abweiserkontakte variiert, was den Rückstosseffekt direkt beeinflusst. Aufgrund der grossen Erfahrung der Piloten in der Beobachtung von Bobfahrten an der Bahn oder in Videos, können sie die Authentizität des Rückstossverhaltens im Simulator gut anhand von aufgezeichneten Simulatorsequenzen beurteilen. Mit verschiedenen Piloten konnte nach mehreren Iterationen jeweils ein realistisches Verhalten erreicht werden. Das Experiment ergab wiederholt Werte $\varepsilon_n = 0.05 \pm 0.01$.

In einem zweiten Experiment wird die optimale Form der Menge $\lambda_n \mathscr{F}$ der zulässigen Reibkräfte der Kufenkontakte, siehe Abbildung 4.7, gesucht. Das Verhältnis der Reibkoeffizienten in Längs- und Querrichtung $\frac{\mu_t}{\mu_b}$ und der rechteckige Anteil in der Längsrichtung $\frac{\mu_r}{\mu_t}$ werden im Experiment variiert. Dies verändert einerseits das Längenverhältnis der Form von $\mathscr{F}$ und andererseits die Flachheit des elliptischen Randes. Im Allgemeinen steigt für eine bestimmte Gleitrichtung die Reibkraft in Querrichtung für breitere Mengen und flachere Ellipsen an. Eine erhöhte Grenze für die Reibkraft in Querrichtung lässt das Heck des Schlittens weniger ausbrechen und erhöht die Lenkbarkeit des Schlittens. Beide Effekte werden von den Experten aufgrund ihrer grossen Erfahrung präzise beurteilt. Das Ziel des zweiten Experimentes ist es, ein realistisches Fahr- und Lenkverhalten zu erreichen. Dies ist essentiell, damit die Piloten den Simulator als Trainingsgerät akzeptieren und nutzen. Daneben ist die Bestimmung der Reibwerte in Längs- und Querrichtung an Werten aus der Literatur orientiert. Ein untergeordnetes Ziel ist es, in der Nähe dieser Werte zu bleiben. Die Literatur erwähnt relativ grosse Bereiche für die Reibwerte. Der Grösste $\mu_t \in [0.0, 0.075]$ findet sich in [31]. Jüngere Studien [8, 74, 76] schränken den Längsreibungskoeffizient auf $\mu_t \in [3.3 \cdot 10^{-3}, 0.018]$ ein. Ein Querreibungskoeffizient wird nur in den Arbeiten [83, 4, 11] beschrieben. Diese nennen Werte $\mu_b \in [0.05, 0.3]$. Das Experiment im Simulator führte auf Reibwerte $\mu_t = 4 \cdot 10^{-3}$, $\mu_b \in [0.02, 0.1]$ und $\frac{\mu_r}{\mu_t} = 0.66$. Diese Werte liegen am unteren Ende der Bereiche aus der Literatur. Ein Effekt, der auch in der Simulation in [4] beobachtet wird.

Kapitel 5

Bobsimulator

Dieses Kapitel präsentiert die Bestandteile und den Aufbau des Bobsimulators, so wie er im Labor steht und benutzt wird. Die Software besteht aus den in den Kapiteln 2 bis 4 vorgestellten drei Bestandteilen, einem Bahnmodell als Kollisionsgeometrie, dem Bobmodell und einem Treiber, der die Simulation rechnet. Ein weiterer wichtiger Bestandteil ist die grafische Darstellung der Szenerie und die detaillierte Ausgestaltung der Umgebung. Darauf wird hier allerdings nicht eingegangen. Neben dem Simulationsrechner und einem Beamer mit Leinwand besteht die Hardware aus einem echten Bobschlitten, der mit einem Sensor zur Messung des Lenkwinkels ausgestattet ist. Weiter ist ein System für haptisches Feedback verbaut, so dass der Pilot die Kräfte auf den Lenkseilen, die durch die (simulierten) Kontaktkräfte induziert werden, in seinen Händen spüren kann. Schliesslich bildet eine Audioanlage bestehend aus Subwoofer und zwei Boxen einen zusätzlichen Rezeptionskanal für den Piloten.

Das Bahnmodell aus Kapitel 3 wird anhand der Konstruktionsdaten automatisch mit einem Pythonskript innerhalb der 3D-Modellierungssoftware *Maya* [5] generiert und per Export als Dreiecksgitter gespeichert. Die Konstruktionsdaten sind dabei in Textdateien hinterlegt. Dieses Modell wird danach in der Simulatorsoftware im Terrain platziert und bei Bedarf mit Details ausgeschmückt. Es können beispielsweise Lampen oder Sonnensegel entlang der Bahn angebracht werden. Dies erlaubt einerseits bei Tageslicht (mit einstellbarem Sonnenstand) mit geöffneten oder geschlossenen Sonnensegeln zu fahren und andererseits ein Nachtrennen mit künstlicher Beleuchtung zu simulieren. Unter künstlicher Ausleuchtung ist die Bahn in allen Details erkennbar, während bei natürlicher Beleuchtung Licht- und Schattenwechsel die Wahrnehmung beeinflussen können. Geschlossene Sonnensegel wiederum schränken die Sicht des Piloten erheblich ein und erschweren präzise Fahrten enorm. Diese Optionen sind nützlich, um auf die Steuerfähigkeiten und das Trainingsziel des Benutzers eingehen zu können.

Die Simulatorsoftware implementiert das in Kapitel 4 vorgestellte Modell des Bobschlittens. Über das in Abbildung 5.1 gezeigte Fenster können einige gebräuchliche Parameter verändert und das Bobsetup gespeichert bzw. wieder geladen werden. So können beispielsweise die verschiedenen Federsätze in der Mitteltrennung und der (vorderen) Pendelachse aus einer Liste ausgewählt werden. Über Schieberegler können der Stosskoeffizient ε_n der

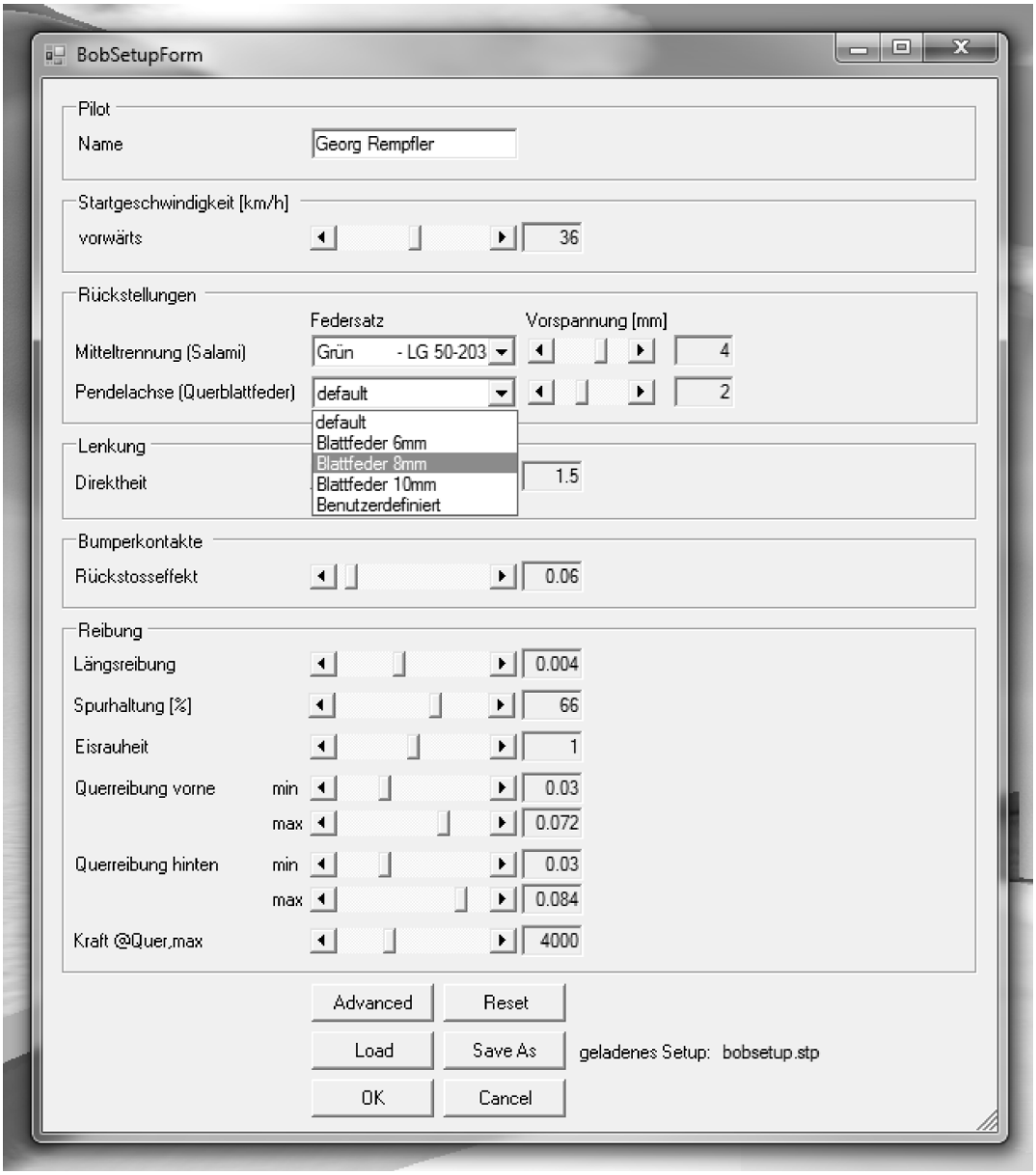

Abbildung 5.1: Screenshot der Einstellmöglichkeiten des Bobmodells. Hier können u.a. die Startgeschwindigkeit und Reibmodellparameter verändert werden. Für die Mitteltrennung und die Pendelachse stehen ausserdem verschiedene Federsätze zur Auswahl.

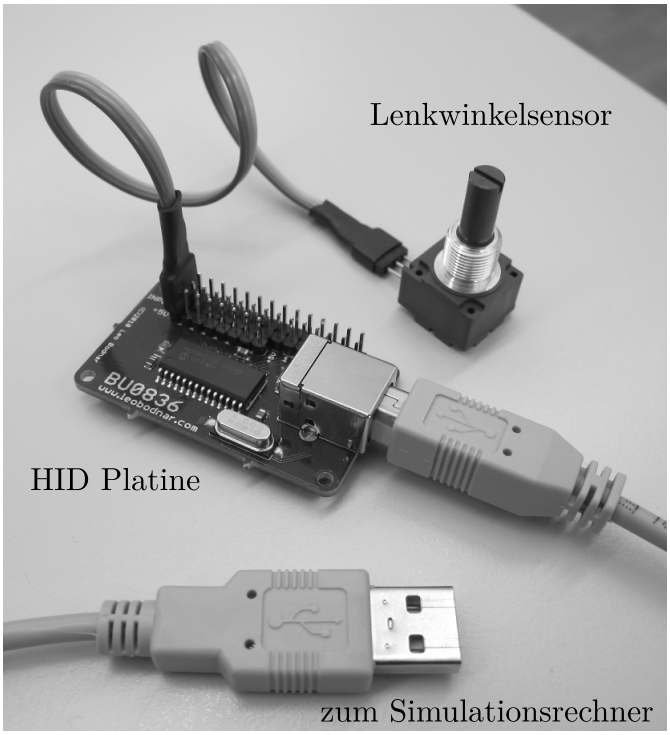

Abbildung 5.2: Human Interface Device (HID) Platine mit angeschlossenem Lenkwinkelsensor und USB Kabel. Der Sensor wird von Windows automatisch als einachsiger Joystick erkannt. Nach einer Kalibration steht die Position als 12bit-Signal normiert auf einen Bereich $[-1, 1]$ zur Verfügung.

Bumperkontakte oder die diversen Parameter des Kufenreibmodells verändert werden. An den Simulationsrechner ist über eine *Human Interface Device*-Platine (HID) ein Potentiometer angeschlossen, welches den Lenkwinkel misst. Diese Komponenten sind in Abbildung 5.2 gezeigt. Der USB Anschluss wird vom Rechner als einachsiger Joystick erkannt. Dieser muss einmalig kalibriert werden. Danach kann das Positionssignal als 12bit-Signal im Bereich $[-1, 1]$ vom Treiber ausgelesen werden, wobei die Werte -1 und 1 dem konstruktiv begrenzten und bekannten Maximaleinschlag entsprechen. Bei einem maximalen Lenkwinkel von $\pm 10°$ ergibt sich eine Signalauflösung von ungefähr $0.005°$.

Pro Durchlauf der Hauptprogrammschleife wird ein Bild gerendert und der Treiber rechnet 83 Zeitschritte. Bei einer Bildrate von 60Hz ergibt dies einen Zeitschritt von $\Delta t = (60\text{Hz} \cdot 83)^{-1} \approx 0.2\text{ms}$. Der exakte Wert variiert mit der Bildrate, da der Treiber in den 83 Zeitschritten immer gerade soweit rechnet, wie der letzte Schleifendurchlauf gedauert hat. So deckt sich die Simulationszeit immer in etwa mit der Echtzeit. Für das Rechnen aller Zeitschritte benötigt der Treiber grob zwischen 4ms und 6ms. Von den 17 Kontakten im Modell sind zu den meisten Zeiten sechs bis acht Kufenkontakte und zusätzlich bis zu zwei Abweiserkontakte geschlossen und damit aktiv in einem Zeitschritt. Die Massenmatrix hat die Dimension $\mathbf{M} \in \mathbb{R}^{13\times 13}$. Die Dimensionen der beteiligten Matrizen und Vektoren sind damit relativ klein. Alle Simulationen laufen auf einem Computer mit *Windows 7* 64-bit, 16GB RAM und einem *i5-3570K*@3.4GHz Prozessor.

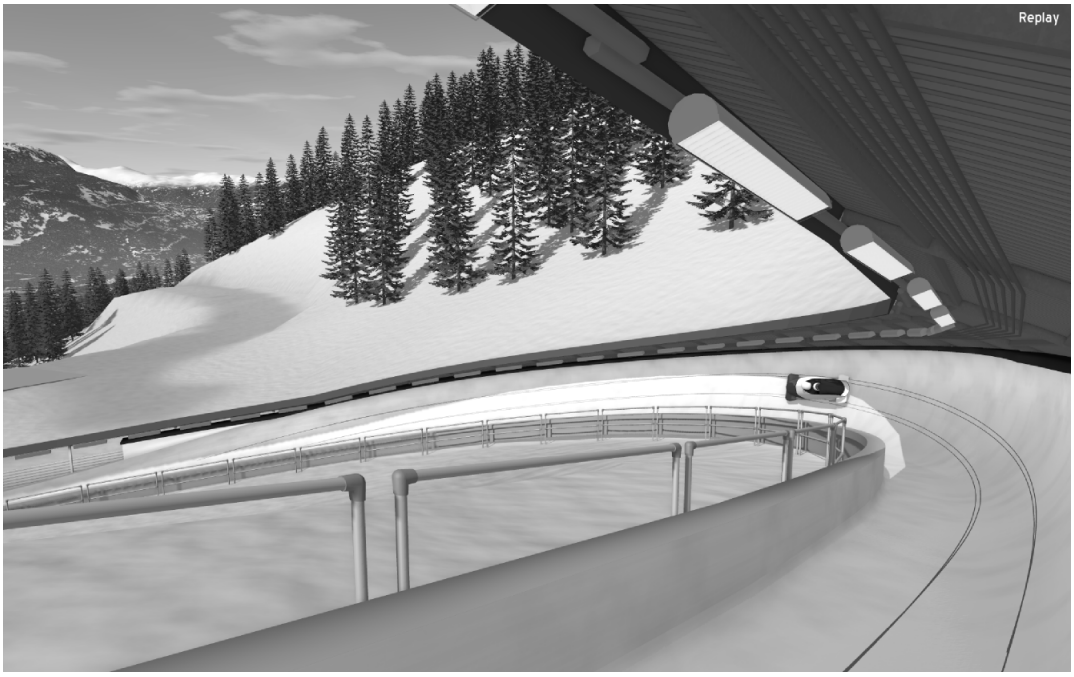

Abbildung 5.3: Screenshot der Simulation im Replay Modus mit eingeschalteter Bahnbeleuchtung und offenen (nicht sichtbaren) Sonnenblenden.

Die Grafik wird mit einer *GeForce GTX 680* Grafikkarte mit 4GB Speicher gerendert. Basierend auf dieser Hardware werden robuste Bildraten über 60 Bildern pro Sekunde bei einer Auflösung von 1920×1200 Pixeln erreicht. Diese Werte werden mit aktiven Supersampling Techniken erreicht, welche das gerenderte Bild erheblich verschönern. Die Bildraten könnten ohne Weiteres noch verdoppelt werden, indem auf diese graphischen Verbesserungen verzichtet würde. Abbildung 5.3 zeigt einen Screenshot der virtuellen Szenerie einer Simulation im Replay Modus. Im Replay Modus werden die Spuren der geschlossenen Kufenkontakte auf der Eisoberfläche dargestellt. Man erhält damit einen klaren Eindruck der Fahrlinie des Schlittens. Fahrten können abgespeichert und wieder geladen werden. Beim Abspeichern in eine Datei werden für spätere Vergleiche das gefahrene Bobsetup, die Lauf- und Zwischenzeiten und die Fahrspuren abgelegt. Zusätzlich zeichnet der Fahrtenrecorder alle 10ms sämtliche Zustände, d.h. t, $\boldsymbol{q}$, $\boldsymbol{k}$, $\boldsymbol{u}$, $\boldsymbol{m}$, $\dot{\boldsymbol{m}}$, $\boldsymbol{i}$ und $\boldsymbol{j}$ auf. Im gegebenen Takt liest der Fahrtenrecorder die aktuellen Lagen und Geschwindigkeiten aus. Die induzierten Kräfte hingegen werden über die Ausleseperiode gemittelt. Der Treiber arbeitet auf Geschwindigkeitsebene. Bei den Kontaktperkussionen $\hat{P}_n$ und $\hat{\boldsymbol{P}}_f$ handelt es sich deshalb um Impulse bzw. die über den Zeitschritt integrierten Kontaktkräfte. Da der Fahrtenrecorder den Zustand nicht für jeden Zeitschritt aufzeichnet, wird es als sinnvoll erachtet, nicht die (stark variierenden) Perkussionen des aktuellen Zeitschritts auszulesen, sondern die Perkussionen aller Zeitschritte seit der letzten Aufzeichnung zu mitteln.

Als Simulatorplattform ist ein moderner *Citius*-Zweimannschlitten auf einem statischen Gestell aufgebockt. Ein Foto davon ist in Abbildung 5.4 zu sehen. Der Pilot sitzt im Schlitten und ein Beamer projiziert die dreidimensionale Szenerie auf eine grosse Leinwand.

Abbildung 5.4: Die Simulatorplattform im Labor.

Die Leinwand befindet sich unmittelbar vor dem Piloten, damit das virtuelle Bild möglichst viel von dessen Gesichtsfeld abdeckt. Hinter der Leinwand bzw. unter dem Bobschlitten sind zwei Boxen und ein Subwoofer aufgestellt. Das Audiosystem wird, wegen der fehlenden Fliehkräfte und Bewegungen im statischen Bob, als alternativer Wahrnehmungskanal genutzt. So ertönen Fahrgeräusche, die mit der Geschwindigkeit skalieren und den Fahrtwind imitieren. Knallt der Bob mit den Bumpern gegen die Berandung des Eiskanals ertönen Stossgeräusche, die vor allem vom Subwoofer getragen werden. Am wichtigsten jedoch ist das Ertönen eines Kratzens, wenn der Schlitten seitlich über das Eis schiebt. Man bemerkt damit sofort, wenn das Heck des Schlittens ausbricht. Mit zunehmendem Schlupfwinkel ist das Signal deutlicher hörbar. Bei einer realen Fahrt fühlt der Pilot mit dem ganzen Körper und allen Sinnen eine Unzahl von Faktoren. In jedem Simulator fallen viele dieser Informationen weg und die wichtigsten müssen imitiert werden. Häufig spielt dabei die Art der Wahrnehmung eine untergeordnete Rolle, weshalb das Ausweichen auf Audiosignale bei Vorgängen, die im Realen eher mechanisch auf den Körper wirken, zufriedenstellend funktioniert.

Ein im Vorderboot verbauter Motor rendert die Haptik in den Steuerseilen. D.h. er bringt das von den (in der Simulation berechneten) Kontaktkräften induzierte Moment im Steuergelenk auf. Dieses Moment spürt der Pilot unmittelbar in seinen Händen an den Lenkseilen. Der Aufbau ist in Abbildung 5.5 gezeigt. Der Motor treibt über ein Spindelgetriebe eine Linearführung an, die zwischen der Lenkplatte und dem Vorderboot montiert ist. Hinter der Linearführung ist auf der Hülse ein Dehnmessstreifen angebracht, der die Kraft misst. Das Kraftsensorsignal geht über eine A/D-Wandlerkarte an einen

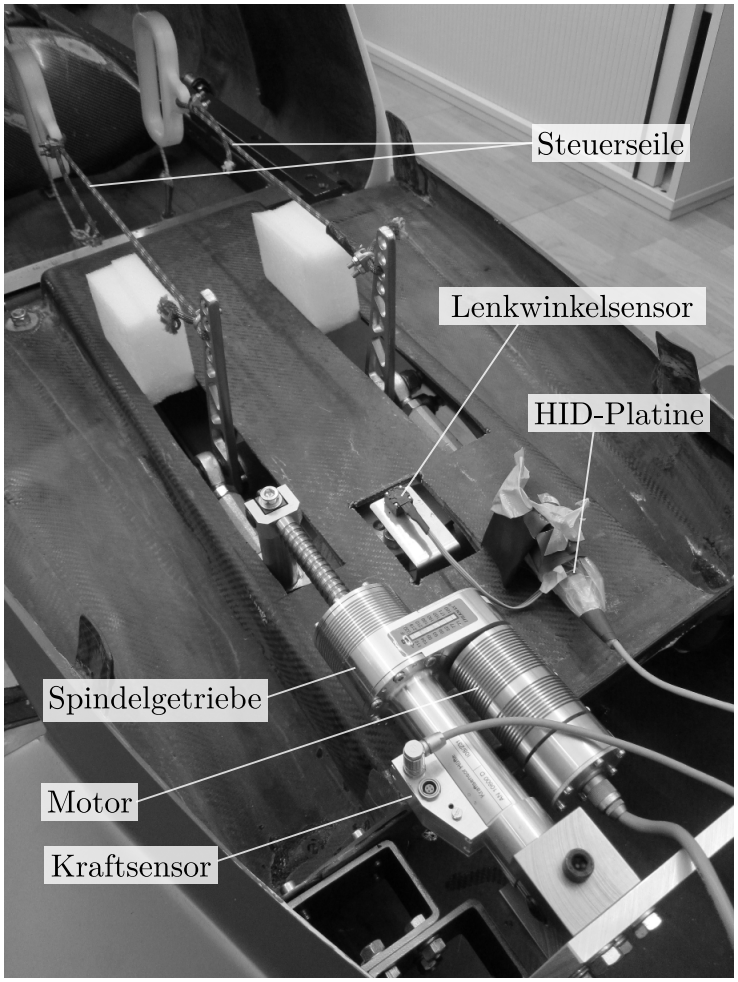

Abbildung 5.5: Das Force-Feedback-System. Ein Motor bringt über ein Spindelgetriebe eine Kraft auf die Lenkplatte des Bobs und induziert so ein Moment im Steuergelenk. Die aufgebrachte Kraft wird mit einem Dehnmessstreifen auf der Hülse hinter der Spindel gemessen.

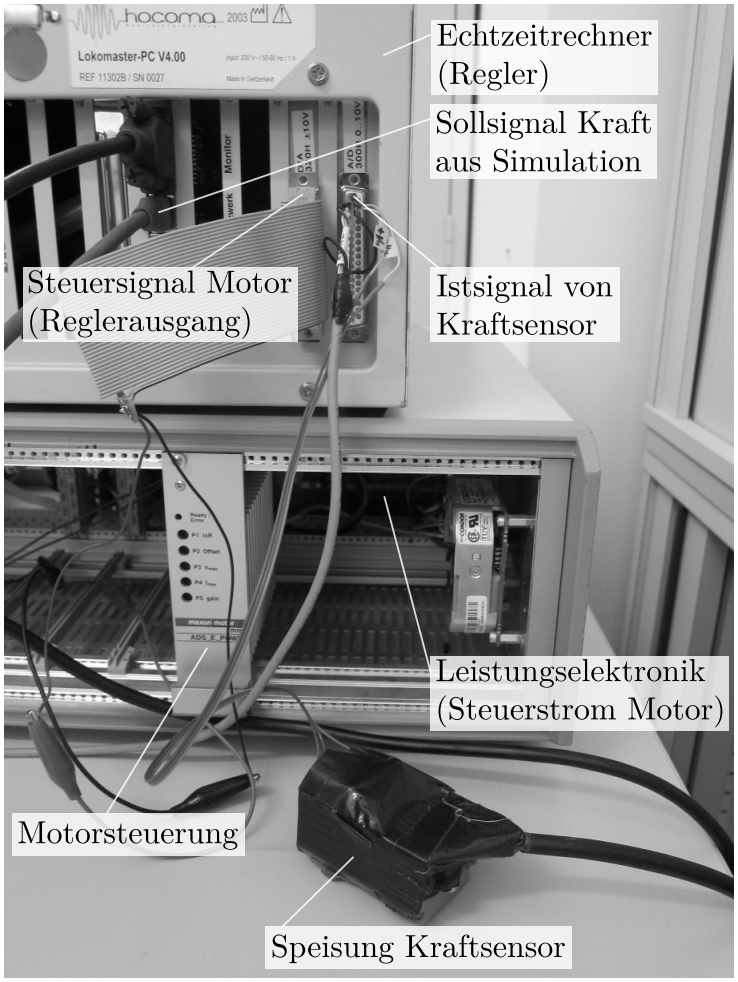

Abbildung 5.6: Elektronikkomponenten des Force-Feedback Systems. Der Echtzeitrechner hat eine Ethernetkarte, eine A/D- und eine D/A-Wandlerkarte. Darauf ist der Regler implementiert, der die Motorsteuerung ansteuert. Über eine Leistungselektronik wird von dieser der Steuerstrom für den Motor im Vorderboot des Bobschlittens generiert.

Echtzeitrechner hinter der Leinwand, siehe Abbildung 5.6. Dieser Echtzeitrechner fungiert als Regler. Das Ausgangssignal des Reglers ist eine Spannung zwischen -10V und 10V, welche über eine D/A-Wandlerkarte an der Motorsteuerung hängt. Diese liefert aufgrund der Steuerspannung über eine Leistungselektronik den Steuerstrom für den Motor. Es handelt sich um eine Stromregelung, weil die Motorkraft gesteuert werden soll. Der angetriebene Motor führt zu einem Moment auf der Lenkachse, das der Pilot über die Hände an den Steuerseilen wahrnimmt. Der Kraftsensor wiederum misst die aufgebrachte Kraft inklusive der Reaktion des Piloten. Damit ist der Regelkreis geschlossen. Das Sollsignal für den Regler liefert der Simulationsrechner. In jedem Zeitschritt werden vom Treiber für jeden Körper die Summe aller angreifenden Kontakt- und Reibkräfte $\boldsymbol{F}_{\boldsymbol{\lambda},S}$ und deren bezüglich dem Schwerpunkt geliefertes Moment $\boldsymbol{M}_{\boldsymbol{\lambda},S}$ aktualisiert, nachdem die Kontakt- und Reibkräfte selbst bestimmt sind. Das von diesen Grössen im (kinematisch aktuierten) Lenkgelenk induzierte Moment M_{soll} berechnet sich nach Gleichung (2.48).

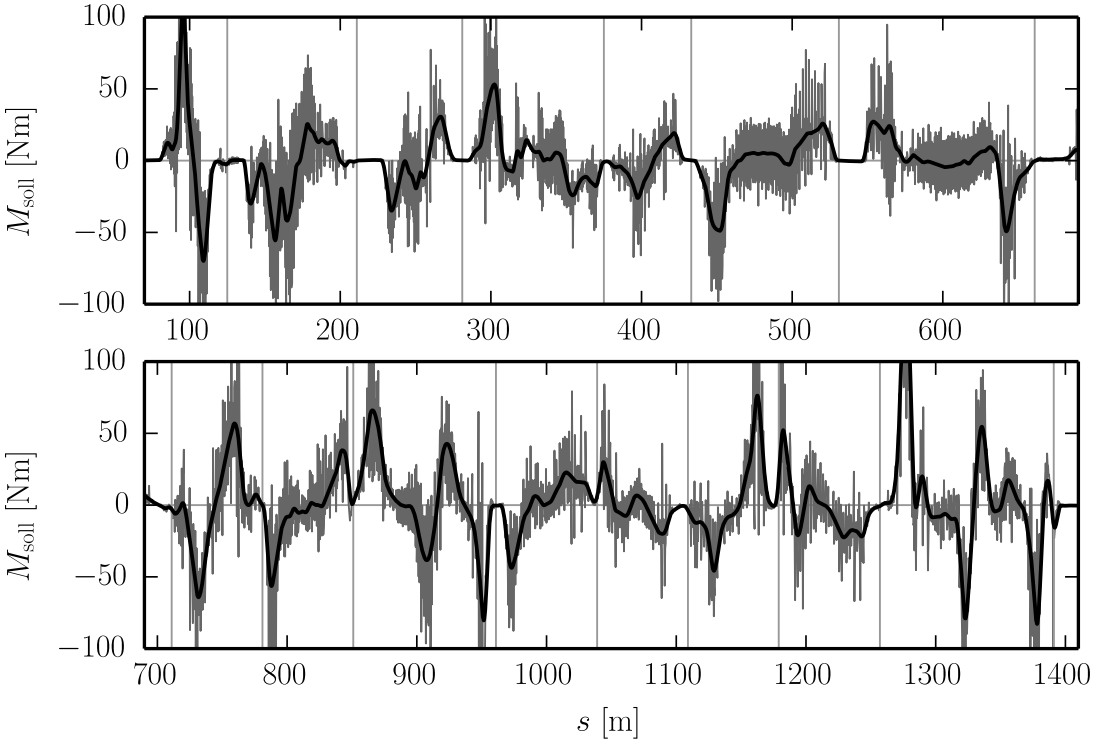

Abbildung 5.7: Verlauf des Stellmoments M_{soll} über eine Simulatorfahrt in Whistler. Die vertikalen Linien (hellgrau) separieren die Kurven der Strecke. Das Rohsignal (grau) ist stark verrauscht, wie man das auch von Messungen kennt. Das gefilterte Signal (schwarz) zeigt, dass pro Kurve meist zwei, manchmal vier Ausschläge mit jeweils entgegengesetzter Richtung auftreten. Diese Spitzen liegen grundsätzlich bei den Kurveneinfahrten und -ausgängen, wo die Rollbeschleunigung gross ist.

Mit der Hebellänge $l = 0.1$m der Lenkplatte, also dem Abstand des Motors zur Lenkachse, ergibt sich die Sollkraft des Motors $F_{\text{soll}} = M_{\text{soll}}/l$. Der Simulationsrechner sendet den Wert der Sollkraft über ein Ethernetkabel zum Echtzeitrechner. Da dieser Sendevorgang relativ lange dauert, wird er erstens asynchron programmiert und werden zweitens, wie schon beim Zustandsrecorder, die zugrundeliegenden Kontaktperkussionen über mehrere Zeitschritte gemittelt. Der Filter mittelt über ein Fenster von 10ms. Die Sendedauer variiert zwischen 14ms und 20ms. Abbildung 5.7 zeigt den Verlauf des Stellmoments einer Simulatorfahrt in Whistler. Das verrauschte Signal (grau) ist das über ein Fenster von 10ms gemittelte. Dieses enorme Rauschen ist nicht ungewöhnlich. Es zeigt sich beispielsweise auch in den Messungen der Fliehkräfte von [9, 68] oder den Messungen der Kufenaufstandskräfte von [4]. Der darübergelegte Verlauf (schwarz) ist stärker (2Hz) gefiltert und zeigt, dass Ausschläge vor allem bei Kurveneingängen und -ausfahrten auftreten. An diesen Stellen führt die enorme Rollbeschleunigung zu einer einseitigen Belastung einer (Vorder-)Kufe, was durch die daraus resultierende erhöhte Reibkraft an der Kufe zu einem Moment um die Lenkachse führt. Im Eingang tendiert ein Schlitten nach oben zu lenken, was

meist durch den Piloten unterbunden werden sollte. Zum Ausgang lenkt ein Schlitten aus der Kurve hinaus (nach unten). Hier sind, je nach Richtung des auf die Kurve folgenden Geradenstücks, ebenfalls Übersteuerungen durch den Piloten zweckdienlich. Die Momentenspitzen indizieren den Kurveneingang und -ausgang deutlich spürbar für die Piloten, welche auch ihr Lenkprogramm bzw. das Timing nach diesem Gefühl in den Händen richten. Diese Information ist deshalb eine wichtige Ergänzung zum Visuellen und erlaubt dem Benutzer eine gefühlte Orientierung in der Bahn.

Kapitel 6

Modellvalidierung

Zusätzlich zum positiven Feedback der beteiligten ehemaligen und gegenwärtigen Profipiloten und ihrer Bereitschaft den Simulator als Trainingsgerät zu nutzen, wird das Modell in diesem Kapitel quantitativ gegen echte Rennresultate validiert.

6.1 Datengrundlage

Als Vergleichsdaten dienen die Zwischen- und Laufzeiten des olympischen Zweierbobrennens der Herren von 2010 in Whistler, Kanada [21]. Dieses Rennen wurde ausgewählt, weil olympische Rennen aus vier Läufen bestehen, anstatt den bei Weltcuprennen üblichen zwei, und weil die Bahn seit den olympischen Spielen 2010 diverse kleine bauliche Veränderungen erfahren hat, die in den verwendeten Konstruktionsdaten nicht berücksichtigt sind. Es ist deshalb davon auszugehen, dass die vorliegenden Konstruktionsdaten am ehesten den Originalzustand der Bahn beschreiben. In den Vergleichsdatensatz werden die Zeiten der ersten 20 Piloten[9] des Schlussklassements genommen. Bei vier Läufen resultiert das in 80 Laufzeitdatensätzen.

Die realen Daten dieses Rennens werden mit den Ergebnissen von 77 Simulatorläufen verglichen, die vom Autor gefahren worden sind. Man beachte, dass zusätzliche drei Läufe im Simulator in einem Sturz endeten und diese deshalb in der Validierung nicht mitberücksichtigt werden. Aus diesem Grund stimmt die Anzahl der verglichenen Laufzeitdatensätze nicht ganz überein.

6.2 Messungen

Bei Bobrennen im Eiskanal von Whistler werden fünf Zwischenzeiten und die Laufzeit selbst gestoppt. Die Positionen der Zeitmessungen sind in Abbildung 6.1 aufgezeigt und liegen an den Stellen $s \in \{50, 280, 660, 960, 1244, 1450\}$ entlang dem Streckenprofil. Die

[9]Zum vierten Lauf dürfen nur die 20 nach drei Läufen bestrangierten Mannschaften antreten. Vorausgesetzt es gibt keine Aus- bzw. Zwischenfälle im vierten Lauf, beenden deshalb immer genau und nur 20 Piloten das Rennen.

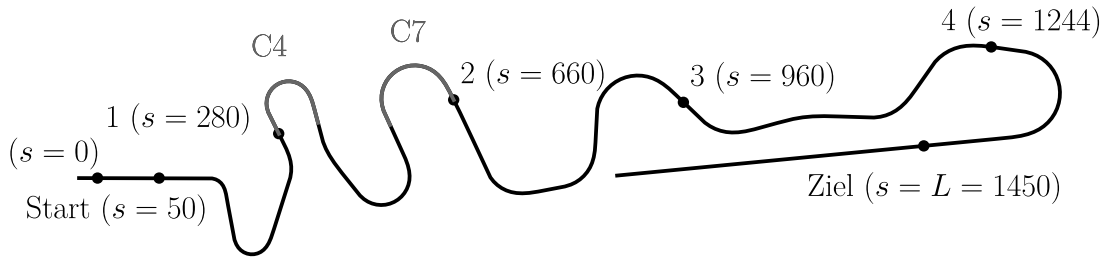

Abbildung 6.1: Messstellen der Zwischen- und Laufzeiten in Whistler, Kanada. Alle Stellenangaben s sind in Metern, [m] angegeben.

Laufzeit wird mit dem Passieren der Stelle $s = 0$ gestartet. Die Messstellen sind anhand von Videoaufnahmen des Rennens geschätzt. Da die Messresultate bei der Übertragung direkt eingeblendet werden, kann anhand der Bilder die Messstelle sehr gut abgeschätzt werden.

6.3 Histogramme und Validierungsresultate

Zuerst werden die Zwischenzeiten aller Läufe in Histogrammen dargestellt. Danach werden die Mittelwerte und Standardabweichungen der einzelnen Zwischenzeiten analysiert und der Mittelwertsfehler zwischen Rennresultat und Simulation eruiert.

Abbildung 6.2 zeigt die Histogramme der Zwischenzeiten des Rennens. Die Varianz der Zwischenzeiten nimmt mit zunehmender Laufzeit zu. Während die Startzeiten noch innerhalb von ungefähr 0.2s liegen, verteilen sich die Laufzeiten zum Schluss, von einem Ausreisser abgesehen, auf etwa 1.6s. Der Anstieg der Varianz ist zu erwarten. Während die Toppiloten ohne grössere Fehler ins Ziel kommen, kumulieren sich bei den restlichen Piloten mehr oder weniger Fahrfehler über die Strecke. Verstärkend auf die Varianz der unteren Zwischenzeiten wirkt zudem die Tatsache, dass Fahrfehler in oberen Streckenteilen zu niedrigeren Geschwindigkeiten führen können, was sich in der Folge erheblich auf die Zeiten auswirkt.

Verglichen mit den Rennresultaten sind die Histogramme der simulierten Läufe einiges konzentrierter. Sie sind in Abbildung 6.3 dargestellt. Speziell die Startzeiten aller Läufe verteilen sich nur auf einzelne Tausendstelsekunden. Dies ist auf die konstanten Anfangsbedingungen in der Simulation zurückzuführen und auf die Tatsache, dass die Streckenführung auf den ersten 50m bis zur Messung der Startzeit ausschliesslich gerade ist. In diesem kurzen Teilstück wird, wenn überhaupt, nur minimal gelenkt. Deshalb resultieren in allen Läufen praktisch identische Startzeiten. In der Realität verhält es sich grundsätzlich anders, weil jede Mannschaft individuell startet. Je nach Sprintleistung und Ablauf des Einsteigens resultieren verschiedene Startzeiten und insbesondere

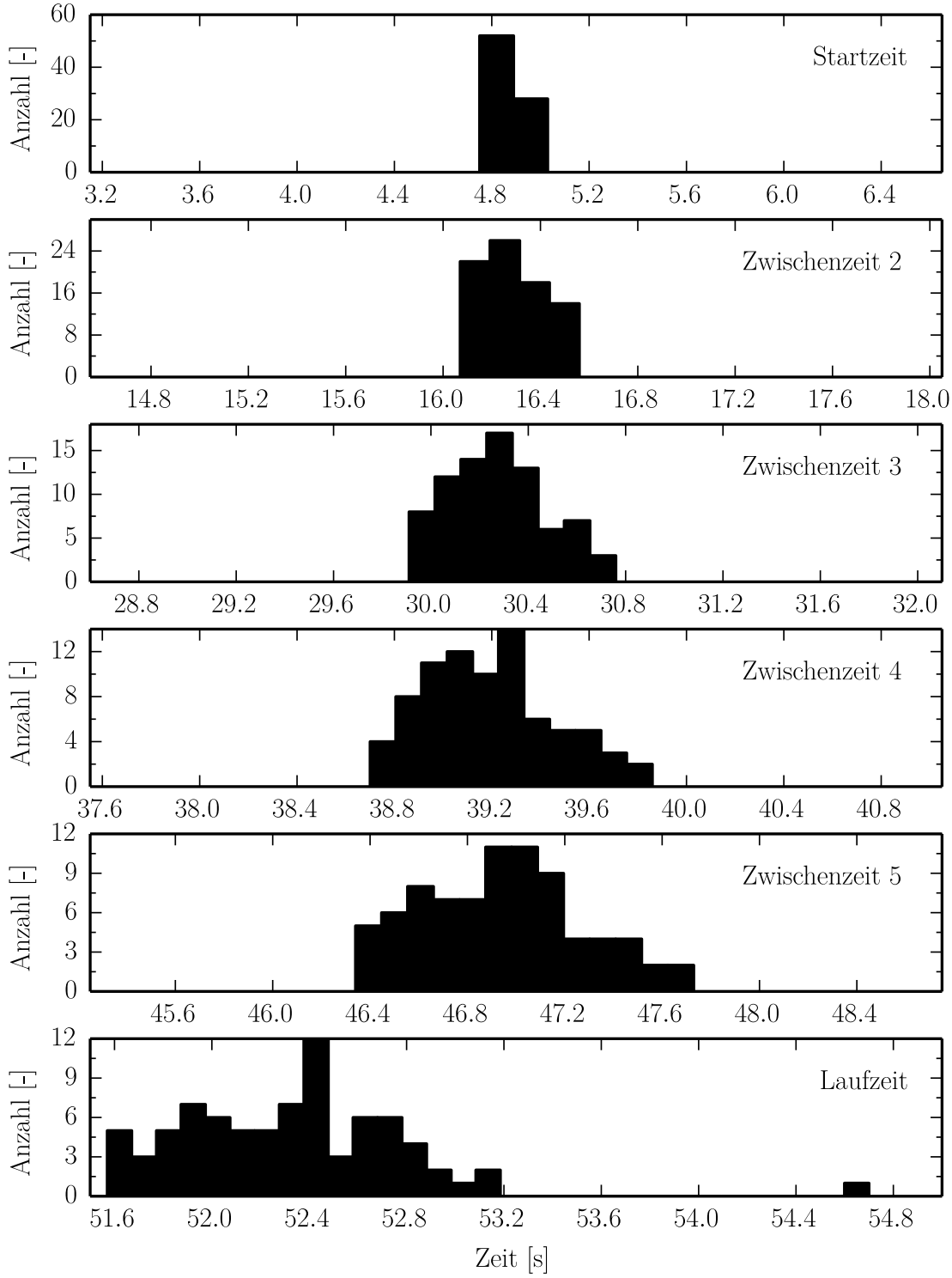

Abbildung 6.2: Histogramme der Zwischenzeiten des Herren-Zweierbobrennens der olympischen Winterspiele 2010.

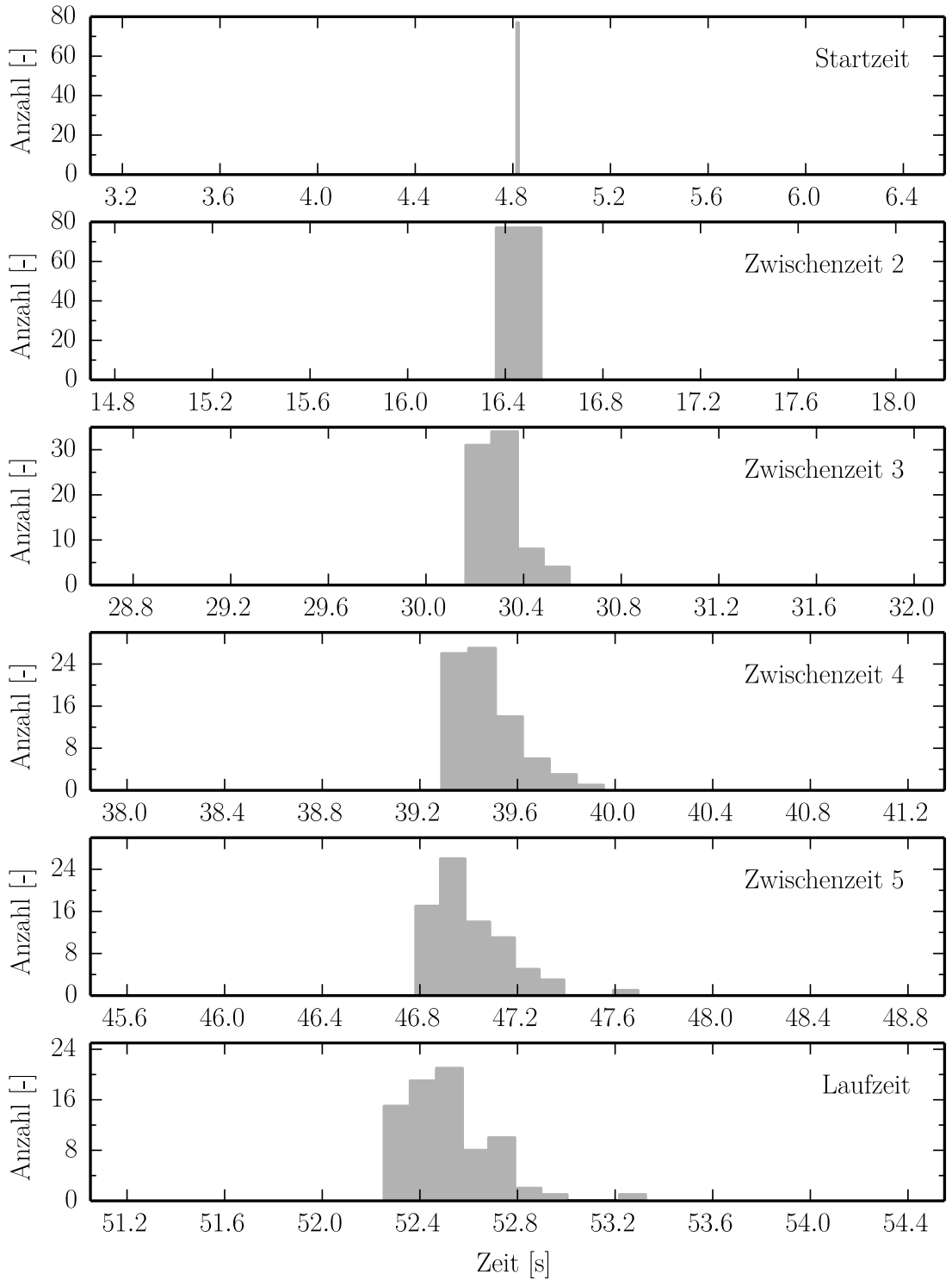

Abbildung 6.3: Zwischenzeiten von 77 Simulationsläufen.

Tabelle 6.1: Statistik von 80 Rennläufen verglichen mit 77 Simulationsläufen. Alle Zeiten sind in Sekunden, [s] angegeben.

Zwischenzeit		Startzeit	2	3	4	5	Laufzeit
Rennen	∅	4.8732	16.2883	30.2721	39.1948	46.9438	52.3270
	σ	0.0691	0.1302	0.2054	0.2649	0.3235	0.4730
Simulation	∅	4.8182	16.4049	30.3003	39.4701	47.0068	52.5225
	σ	1.09e-03	0.0320	0.0922	0.1349	0.1612	0.1792
Fehler [%]	∅	1.1297	-0.7160	-0.0929	-0.7026	-0.1343	-0.3736

auch unterschiedliche Abgangsgeschwindigkeiten nach 50m. Die Startzeiten sind folglich breiter verteilt und die Varianz in der Abgangsgeschwindigkeit führt, nebst potentiell hinzukommenden Fahrfehlern, zusätzlich zu einer grösseren Varianz in den folgenden Zwischenzeiten.

Ein weiterer Grund für die breiteren Histogramme der Rennresultate ist die Tatsache, dass sich die Wetter- und Eisbedingungen zwischen den Renntagen und -läufen ändern und die Bahn damit schneller oder langsamer werden kann. Dagegen sind die Bedingungen in der Simulation immer konstant bzw. werden im Modell gar nicht berücksichtigt.

Die Tabelle 6.1 listet die Durchschnitte (∅) und Standardabweichungen (σ) der analysierten Läufe auf und bestätigt was in den Abbildungen 6.2 und 6.3 ersichtlich ist. Die Standardabweichung wächst sowohl im Rennen wie auch im Simulator stetig an. Dabei liegen die Simulationsresultate durchgängig im Bereich der Rennresultate und die entsprechenden Mittelwerte liegen nahe beieinander. Die unterste Zeile der Tabelle 6.1 listet die Abweichung dieser Mittelwerte in Prozent auf. Der Fehler liegt, mit Ausnahme der Startzeit, unter einem Prozent und absolut unter 0.2s über eine Laufzeit von etwas mehr als 52.5s.

Kapitel 7

Analyse von Fahrlinien simulierter Fahrten

Ein Ziel der meisten Bobsimulatorprojekte ist die Optimalität irgendeiner Art zu beurteilen. Einige Projekte [58, 87] leiten theoretisch eine Beschreibung der zeit- oder wegoptimalen Kurve entlang der Bahn her. Diese Projekte benutzen stark vereinfachende (Punktmassen-)Modelle des Bobschlittens und binden dieses üblicherweise bilateral auf die Bahnoberfläche. Andere Projekte analysieren mit statistischen Methoden die Resultate von vielen Simulationen. Während in [4] und [10] Pilotensteuermodelle verwendet werden, um Simulationen offline zu berechnen, werden im Bobsimulator die Läufe von Menschenhand in Echtzeit gesteuert. Die Verwendung von Pilotenmodellen führt zu reproduzierbaren Resultaten, was für die Evaluierung von Setups oder Konstruktionsvarianten von Vorteil ist. In Simulationen, die menschliches Steuern beinhalten, geht die Reproduzierbarkeit im Allgemeinen verloren. Dafür kann mit dem Betrachten von einem Satz von Läufen die natürliche Varianz der Fahrlinien und -zeiten analysiert werden. Die Varianz kann hervorgerufen werden durch individuelles Steuerverhalten, unterschiedliche Fahrlinienstrategien der Piloten oder deren spontaner Reaktion auf Fahrfehler und auf sich ergebende Kurveneinfahrtskonditionen.

Im Folgenden werden sechs schnelle Läufe, die der Autor auf der Bahn von Whistler gefahren ist, untereinander verglichen. Untersucht wird der Effekt verschiedener Fahrlinien auf die Geschwindigkeiten und (Durch-)Fahrzeiten in den Kurven 4 und 7. Diese Kurven sind in Abbildung 6.1 mit C4 bzw. C7 gekennzeichnet. Anhand dieser Abbildung kann man sehen, dass beide Kurven nach rechts gehen und ungefähr dieselbe Länge haben. Die Fahrlinie wird beurteilt anhand des Schwerpunkts S des Hinterboots. Um verschiedene Läufe vergleichen zu können, müssen sämtliche Grössen über die zur Position $\boldsymbol{r}_{OS}(t)$ korrespondierende Stelle $s(t)$ auf dem Streckenprofil $\boldsymbol{c}(s)$ aufgetragen werden. Jeder Eintrag des gespeicherten Verlaufs $\boldsymbol{r}_{OS}$ wird deshalb vorab rechtwinklig auf das Streckenprofil projiziert. Dies sind die Punkte mit minimalem Abstand. Daraus resultieren zu jedem Punkt $\boldsymbol{r}_{OS}(t)$ die korrespondierende Stelle $s(t)$ und der dazugehörige Punkt $\boldsymbol{c}(s(t))$. Die Abbildungen 7.1 und 7.2 zeichnen beide in den oberen zwei Subplots jeweils die Geschwindigkeiten $v_S = \|\boldsymbol{v}_S(t)\|$ und Laufzeiten t relativ zum schnellsten Lauf *02* über die Stelle $s(t)$. Die unteren zwei Subplots stellen die Fahrhöhe h_S über dem Streckenprofil $\boldsymbol{c}(s)$ und die laterale Verschiebung b_S zum Streckenprofil dar. Die Fahrhöhe h_S des Bobschlittenmodells

in einer Kurve berechnet sich nach

$$h_S(t) = (\boldsymbol{r}_{OS}(t) - \boldsymbol{c}(s(t))) \cdot \boldsymbol{e}_y^{\mathrm{T}} \tag{7.1}$$

und ist ein guter Indikator für die Fahrlinie. Er widerspiegelt die im Bobsport gebräuchlichen Ausdrücke *früh* oder *spät* in einer Kurve Höhe zu haben und *eine* oder *mehrere Wellen* darin zu fahren. Die laterale Verschiebung b_S ist mit

$$b_S(t) = (\boldsymbol{r}_{OS}(t) - \boldsymbol{c}(s(t))) \cdot \boldsymbol{e}_x^{\mathrm{P}}(s(t)) \tag{7.2}$$

definiert, wobei sich $\boldsymbol{e}_x^{\mathrm{P}}(s(t))$ mit $\beta(s(t))$ aus Gleichung (3.2) berechnen lässt. Die laterale Verschiebung zum Streckenprofil stimmt überein mit den gebräuchlichen Ausdrücken *früh* (auf der *Aussenseite*) oder *spät* (auf der *Innenseite*) in eine Kurve zu fahren. Diese laterale Position eines Schlittens eingangs einer Kurve beeinflusst entscheidend die Fahrlinie innerhalb der Kurve, siehe [4].

Abbildung 7.1 vergleicht die sechs Läufe in Kurve 4, welche von $s = 280\text{m}$ bis $s = 380\text{m}$ reicht. Eine positive laterale Verschiebung b_S entspricht einem Versatz des Schlittens nach links in Fahrtrichtung gesehen. An den unteren zwei Diagrammen kann abgelesen werden, dass positive b_S zur Kurveneinfahrt (Läufe *08*, *06* und *07*) im Allgemeinen zu einem einzelnen Bogen in der Fahrhöhe h_S führen. Der Bogen erreicht die maximale Höhe ungefähr in der Mitte der Kurve. Eine negative laterale Verschiebung (Läufe *02*, *09* und *08*) führt andererseits zu einem verfrühten Erreichen der Maximalhöhe und kann zusätzlich einen zweiten Bogen nach der Mitte der Kurve provozieren. Die Konsequenz dieser zwei charakteristisch verschiedenen Fahrlinien kann aus den oberen zwei Diagrammen der Abbildung 7.1 abgelesen werden. Vor der Kurve bei $s < 290\text{m}$ sind die Geschwindigkeiten aller Läufe höher oder vergleichbar zum Referenzlauf *02* und alle Läufe haben einen Zeitvorsprung $\Delta t < 0$. Zum Ende der Kurve, bei $370\text{m} < s < 380\text{m}$, haben nur noch die Läufe *08*, *06* und *07* mit *02* vergleichbare Geschwindigkeiten und bewahren ihren Zeitvorsprung. Man beachte, dass Lauf *07* direkt nach der Kurve 4 bei $s = 380\text{m}$ auf der rechten Seite gegen die Bande stösst, was die Geschwindigkeit sprunghaft reduziert und weshalb sich in der Folge Zeitrückstand akkumuliert. Lauf *08*, der einen hohen, einzelnen Bogen fährt, verlässt die Kurve mit höher Geschwindigkeit als eingangs Kurve gehabt, und baut deshalb in der Folge den Zeitvorsprung kontinuierlich aus. Innerhalb der Kurve verliert der Schlitten allerdings Geschwindigkeit, während er auf die Maximalhöhe zusteuert und büsst deshalb innerhalb der Kurve etwas an Zeitvorsprung ein. Dies ist bei Lauf *06*, der ebenfalls einen einzelnen Bogen fährt, etwas anders. Lauf *06* bleibt durch die ganze Kurve etwas tiefer und hält dabei die Geschwindigkeit hoch. Damit wird der Zeitvorsprung auch innerhalb der Kurve kontinuierlich ausgebaut. Zusammenfassend kann statuiert werden, dass Fahrlinien mit einem einzelnen Bogen, der vorzugsweise ungefähr in der Mitte die maximale Fahrhöhe erreicht, andere Fahrlinien übertrifft. Man sollte sich eingangs von Kurve 4 links halten, da man sonst unausweichlich die Maximalhöhe zu früh in der Kurve erreicht und eventuell sogar einen zweiten Bogen fährt. Ein erfahrener Pilot könnte mit feinen Lenkbewegungen versuchen, die erreichte Maximalhöhe zu minimieren, und mit einer tiefen Linie die Geschwindigkeit innerhalb der Kurve hoch zu halten. Man beachte jedoch, dass es viel wichtiger ist, eine Kurve mit hoher Geschwindigkeit (in geeigneter

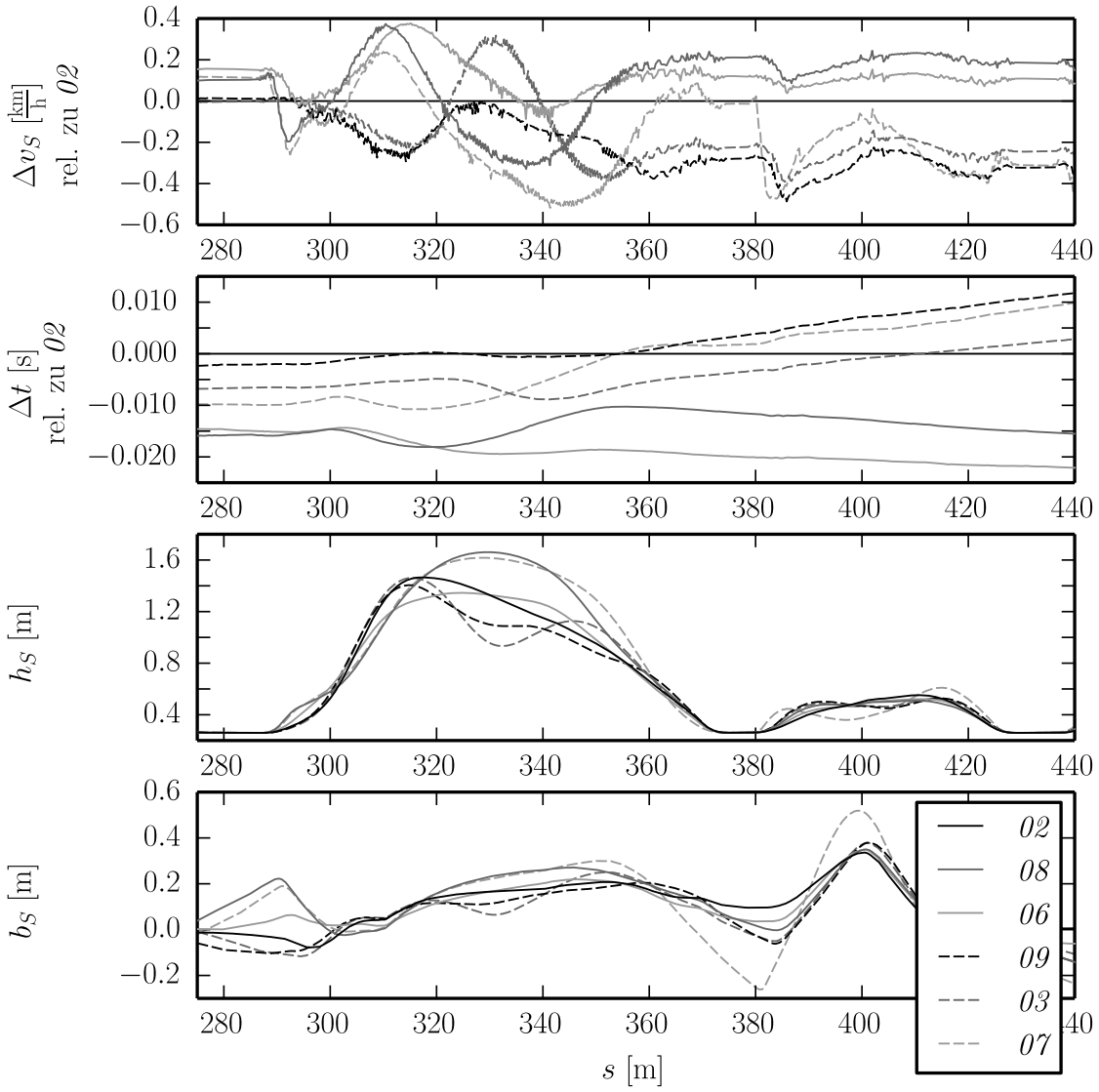

Abbildung 7.1: Vergleich von Fahrlinien in Kurve 4. Der Index S bezieht sich auf den Schwerpunkt des Hinterboots. Die Grösse Δv_S bezeichnet die Geschwindigkeitsdifferenz zu Lauf *02*, Δt die Laufzeitdifferenz, h_S die Höhe über dem Streckenprofil und b_S die Abweichung vom Streckenprofil nach links.

Richtung) zu *verlassen*, denn damit gewinnt man über die ganze Distanz bis zum Eingang der nächsten Kurve Zeit.

Abbildung 7.2 zeigt dieselben vier Masse derselben sechs Läufe über die Kurve 7, die von $s = 550$m bis $s = 660$m reicht. Das dritte Diagramm zeigt wiederum zwei charakteristisch unterschiedliche Fahrlinien. Wie auch in Kurve 4 fährt der Schlitten entweder einen hohen einzelnen Bogen oder erreicht die maximale Höhe vor der Kurvenmitte und fährt einen zweiten, mehr oder weniger ausgeprägten Bogen, ausgangs Kurve. Vor dieser Kurve liegen die zweiten drei Läufe *09*, *03* und *07* zurück und sind etwas langsamer als die Referenz *02*. Die Läufe *08* und *06*, die einen einzelnen Bogen fahren, verlieren durch die Kurve ihren halben bzw. ganzen Zeitvorsprung. Im Gegensatz zu den Erkenntnissen für die Kurve 4, sind in Kurve 7 Fahrlinien mit zwei Bögen (Lauf *02*) zu bevorzugen. Eine solche Linie hält die Geschwindigkeit über die ganze Kurve hoch. Ein einzelner Bogen führt in dieser Kurve nicht zu einer signifikant höheren Ausgangsgeschwindigkeit, wie es in Kurve 4 der Fall ist.

Man beachte, dass im Bobsport Zeitdifferenzen von wenigen Hundertstelsekunden entscheidend sein können. Es ist bemerkenswert, dass ein Lauf mit einer falschen Fahrlinienwahl in einer einzelnen Kurve ruiniert sein kann. Ein Toppilot muss sein Fahrprogramm allerdings nicht nur in der Theorie beherrschen und die korrekte Linie *wählen*. Denn schon kleine (zufällige) Ungereimtheiten zum Ausgang einer Kurve können dazu führen, dass man nach der Geraden zum Eingang der nächsten Kurve schlecht steht, wie beispielsweise die Läufe *09* und *03* eingangs Kurve 4. Eine suboptimale Fahrlinie ist in solchen Situationen unausweichlich. In diesen Fällen ist ein schnelles Reagieren und das Kennen der besten Alternativen gefragt. Dieser Umstand macht schnelles Bobfahren zu einer Erfahrungs- und Routinesache.

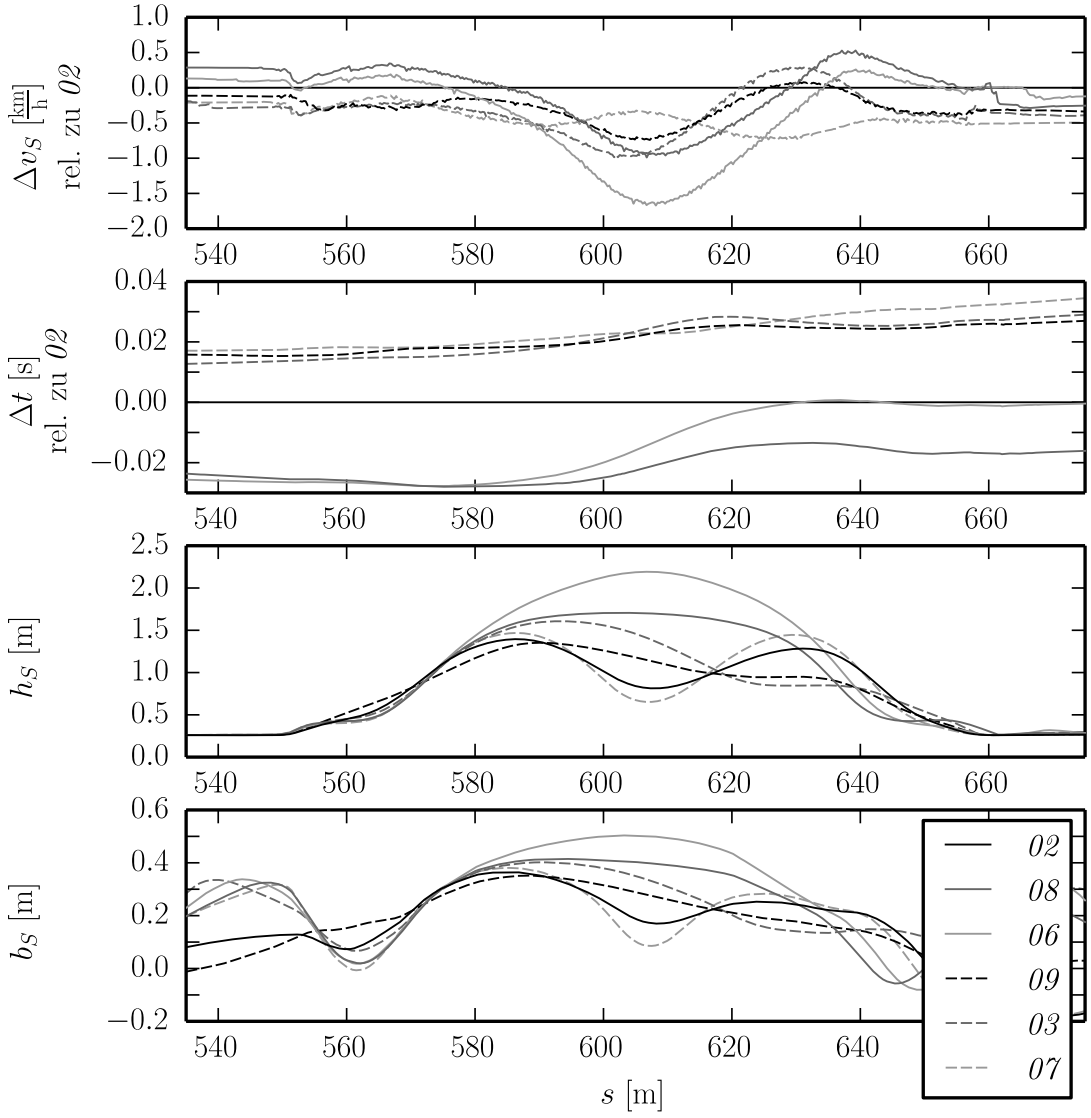

Abbildung 7.2: Relativer Vergleich von sechs Fahrten in Kurve 7. Der Index S bezieht sich auf den Schwerpunkt des Hinterboots. h_S bezeichnet die Höhe über dem Streckenprofil und b_S die Abweichung vom Streckenprofil nach links.

Kapitel 8

Schlussfolgerung

Das in dieser Arbeit beschriebene Simulatorprojekt kombiniert verschiedene Elemente von bestehenden Simulatoren bzw. Simulationsprojekten und bringt neue hinzu. Im Unterschied zu den echtzeitfähigen Simulatoren in Kalifornien [52], Chemnitz [30] und (vermutlich auch in) Japan [71], ist das Bobmodell nicht zweiseitig auf die Bahnoberfläche gebunden. Und der Bobschlitten ist nicht nur als Punktmasse, sondern als Mehrkörpermodell implementiert.

Das Mehrkörpermodell dieser Arbeit verwendet Konstruktions- und Messdaten aus dem Projekt *Citius*, welches moderne Schlittendesigns und heute sehr begehrte, schnelle Bobs hervorgebracht hat. Es widerspiegelt demnach den aktuellen Stand der Technik. Das Modell ist denjenigen aus Braghins [10] oder Maissers [59] Simulationsanalysewerkzeug ähnlich, unterscheidet sich aber in einigen Freiheitsgraden und vor allem in der Verwendung harter, einseitiger Kontakte für die Interaktion mit der Bahnoberfläche. In den Kontakten sind zudem Coulombreibelemente enthalten, die verschiedene Effekte der Kufe-Eis-Interaktion nachbilden. Die mathematische Beschreibung dieser Modelle enthält sowohl den Haftfall, wie auch den Fall von Gleiten (in beliebiger Richtung). Sie führt zu einer numerisch effizienten Implementierung, womit die Bewegung bei vorliegendem Projekt in Echtzeit gerechnet werden kann. Es soll weiter erwähnt werden, dass es sich beim Treiber um ein Time-stepping Verfahren handelt. Im Gegensatz dazu verwenden Maisser [61](S. 10) und Hubbard [38](S. 265) vermutlich Event-driven Algorithmen.

Die Verwendung von unilateralen Kontakten stellt, im Gegensatz zu den bilateral gebundenen Punktmassen, keine Anforderungen an die Glattheit des Bahnmodells. Wie bei anderen Projekten liegt auch hier der Konstruktion die exakte Geometrie der Bahnoberfläche zugrunde. Letztendlich wird aber auf eine diskrete Approximation als Dreiecksgitter übergegangen. Die Abweichungen sind tolerierbar in Anbetracht der vorhandenen Unsicherheit der Eisschichtdicke und Eisrauigkeit. Man beachte, dass auch die Splinemodelle anderer Arbeiten letztlich nur Näherungen bleiben. Das Dreiecksgitter kann günstigerweise sowohl für die Physik, wie für die Grafik verwendet werden. Es erlaubt ausserdem eine effiziente Kollisionsdetektion, was bei Splineoberflächen ein aufwendigeres Problem ist, vgl. [15, 72].

Weil der vorliegende Simulator an einen echten Zweierbob angebunden ist, benutzt der

(Schweizer) Pilot sein gewohntes Lenksystem. Dabei sind auch alle Trägheiten, Rückstellungen und Reibungs- sowie Spielungenauigkeiten enthalten. Man kann deshalb festhalten, dass damit der maximal mögliche Systemanteil der echten Situation entspricht. Es müssen zusätzlich zur Simulation der Fahrt, der Darstellung der Szenerie und der Rückführung diverser daraus resultierender Grössen keine weiteren Elemente nachgeahmt werden. Beim Simulator der Universität von Kalifornien ist dies gemäss [52] der Fall.

In der gerenderten Szenerie ist nicht nur die Bahnoberfläche mit dem Schlitten dargestellt, sondern die komplette Umgebung mit beliebigen Details enthalten. Hier profitiert das Projekt von den heute verfügbaren Grafikkarten und dem Stand der Technik in der Computergrafik. Das Einbinden der Landschaft und z.B. der Randelemente der Bahn führt zu einer erhöhten Akzeptanz bei den Piloten und ist nach [30], S. 245f, für das „subjektive Empfinden [der Piloten] von grosser Wichtigkeit“.

Auf akustischem Weg wird neben einem generellen Fahrgeräusch, das mit der Geschwindigkeit moduliert, speziell auf die ungünstigen Fahrsituationen bei Stössen gegen die Bande und bei schiefem Gleiten der Kufen aufmerksam gemacht. Beide Signale nehmen in der Intensität zu, je grösser die Geschwindigkeit in Normalrichtung vor dem Stoss bzw. je grösser der Schlupfwinkel ist. Insbesondere die subtile Wahrnehmung von Quergleiten ist essentiell für eine schnelle Fahrt. Das Darstellen über einen akustischen Kanal bietet eine günstige Alternative zu einem Bewegen des Schlittens. Eine weitere sensorische Stimulierung des Piloten geschieht über die haptische Darstellung des Reaktionsmoments in der Lenkachse. Dieses Moment wird auf natürliche Weise von den Kontakt- und Reibkräften an den Kufen induziert. Auch beim Simulator in Kalifornien gibt es eine solche Kraftrückführung, vgl. [52]. Allerdings wird nicht klar, wie dieses Moment berechnet wird. Da dort ein Punktmassenmodell mit einer Lenkkraft verwendet wird, ist auch nicht einfach ersichtlich, wie dieses Moment im Modell entstehen soll.

Die Bedienung des Simulators ist einfach gehalten und erlaubt es auch Benutzern, für die der Umgang mit Computertechnik nicht unbedingt alltäglich ist, selbständig und unbeaufsichtigt mit dem Gerät zu interagieren. Über Menüeinträge können Fahrten gespeichert und geladen werden. Mit der Darstellung einer Vergleichsfahrt als transparentem Bob (Geist), gelingt der direkte, visuelle Vergleich von Fahrlinien. Das bietet eine schnelle und einfache Möglichkeit der Analyse und ist insbesondere auch für *Fachfremde* intuitiv zu verstehen.

Weiter besteht die Möglichkeit, gespeicherte Fahrten anhand der Verläufe von beliebigen Zustandsgrössen zu analysieren. Die Laufdateien liegen in einem xml-Format vor und sind damit mit beliebigen Textprogrammen zu öffnen. Sie enthalten u.a. mit 100Hz aufgezeichnete Zustandsverläufe, die Parameter des gefahrenen Setups, sowie die Geschwindigkeitsmessungen und Lauf- und Zwischenzeiten.

Zusammenfassend kann gesagt werden, dass es sich beim Simulator um eine erfolgreiche Demonstration der Möglichkeiten und Anwendung der Methoden der nicht glatten Dynamik handelt. Die Modellierung mit mengenwertigen Kraftgesetzen ist präzise und verständlich, der angewendete Time-Stepping Algorithmus zielführend und effizient.

Der Simulator ist von Schweizer Piloten gut akzeptiert und wird als Trainingsgerät

genutzt. Er war bei sämtlichen Schweizer Olympiakandidaten für die Spiele von 2014 in Sochi Bestandteil der Vorbereitungen auf die internationalen Trainingswochen und schliesslich auf die olympischen Rennen selbst. Auch von ehemaligen Profipiloten wurde mehrfach attestiert, dass es sich hier um den realistischsten Simulator handelt. Weiter sind Trainer interessiert, das Gerät bei der Förderung junger Nachwuchstalente einzusetzen und junge Sportler sind motiviert, den Simulator zu nutzen. Mit der zukünftigen Nutzung wird eine stetige Weiterentwicklung und ein Ausbau der Möglichkeiten einhergehen. Es bleibt zu erwähnen, dass neben allen jetzigen und zukünftigen Evaluierungen, der schlichte, mit jeder Fahrt einhergehende Erfahrungsgewinn nicht nur die Leistung fördert, sondern auch zur Sicherheit der Sportler beiträgt. Sowohl [86], wie [57] statuieren, dass das Unfallrisiko mit zunehmender Erfahrung auf einer Bahn abnimmt.

Anhand der nachträglichen Analyse gespeicherter Fahrten können optimale Strategien und gute Alternativen für einzelne Bahnen ausgearbeitet werden. Der Simulator dürfte aber auch zu einem effektiven Werkzeug für die Beratung von Bahnkonstrukteuren werden. Die von Hubbard [34] unabhängig erstellte Analyse eines tödlichen Unfalls legt Nahe, dass zukünftig Simulationen vermehrt in die Analyse und Evaluierung von Bahnen einzubeziehen sind. Neben den reglementierten Fliehkraft- und Geschwindigkeitslimiten, die sehr gut mit Punktmassenmodellen abgeschätzt werden können, wird der erhöhten Sturzgefahr bei ungünstigen Designs oder auf extremen Fahrpfaden nach Ansicht des Autors heute (noch) zu wenig Aufmerksamkeit geschenkt.

Literaturverzeichnis

[1] Abramowitz, M. und Stegun, I. A. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. 3rd Printing with Corrections. Bd. 55. Applied Mathematics Series. National Bureau of Standards, 1965.

[2] Aeberhard, U. „Geometrische Behandlung idealer Stösse“. Dissertation. ETH Zürich, 2008.

[3] Arnold, P. D. *Persönliche Kommunikation*. 2009 - 2013.

[4] Arnold, P. D. „Analyse und Konzeption von Bobfahrwerken“. Dissertation. ETH Zürich, 2013.

[5] Autodesk. *Maya*. Software (64-bit). 2011. URL: http://www.autodesk.com/products/autodesk-maya/overview (besucht am 03. 07. 2015).

[6] Autodesk. *Maya Python API*. Python Command Reference. 2011. URL: http://download.autodesk.com/us/maya/2011help/index.html?url=./files/Maya_Python_API_Using_the_Maya_Python_API.htm,topicNumber=d0e678623 (besucht am 03. 07. 2015).

[7] Baumann, W. „The influence of mechanical factors on speed in Tobogganing“. In: *Biomechanics III*. Hrsg. von Cerquiglini, S. und Venerando, A. Bd. 8. Medicine and Sport. Karger, 1973, S. 453–459.

[8] Böttcher, R. und Scherge, M. *Reibmessungen zum System Kufe/Eis*. Techn. Ber. Fraunhofer Institut für Werkstoffmechanik, 2011.

[9] Braghin, F. et al. „Design and Verification of Bobsleigh Track“. In: *Proceedings of the ASME 2010 10th Biennial Conference on Engineering Systems Design and Analysis*, 2010.

[10] Braghin, F. et al. „Multi-body model of a bobsleigh: comparison with experimental data“. In: *Multibody System Dynamics* 25, 2011, S. 185–201.

[11] Braghin, F. et al. „Experimental assessment of bobsleigh dynamics and ice-skate contact forces“. In: *Topics in Modal Analysis II*. Hrsg. von Allemang, R. et al. Conference Proceedings of the Society for Experimental Mechanics Series. Springer New York, 2012, S. 487–498. DOI: 10.1007/978-1-4614-2419-2_50.

[12] Bremer, H. *Dynamik und Regelung mechanischer Systeme*. B. G. Teubner, 1988.

[13] CAD Center. *Chukyo University Bobsled Simulator*. URL: http://www.cadcenter.co.jp/business_en/simulator/case_vr0027/ (besucht am 04. 12. 2014).

[14] Dom, M., Hüffner, F. und Niedermeier, R. „Tiefensuche (Ariadne und Co.)“ In: *Taschenbuch der Algorithmen*. Hrsg. von Vöcking, B. et al. eXamen.press. Springer, 2008, S. 61–73. DOI: 10.1007/978-3-540-76394-9_7.

[15] Dyllong, E. und Luther, W. „Distance Calculation Between a Point and a NURBS Surface“. In: *Proceedings of International Conference on Curves and Surfaces*. Hrsg. von Laurent, P.-J., Sablonnière, P. und Schumaker, L. L. Bd. 1. Curve and Surface Design. Saint Malo, France: Vanderbilt University Press, Nashville, TN, Juli 1999, S. 55–62.

[16] Ericson, C. *Real-Time Collision Detection*. The Morgan Kaufmann Series in Interactive 3D Technology. Elsevier Science, 2004.

[17] Eule, G. „Bob-Fahrsimulator: Ein Programm zur Simulation von Fahrten gesteuerter Bobschlitten auf realen Wettkampfbahnen“. In: Bundesinstitut für Sportwissenschaft(BISp). *BISp-Jahrbuch 2000*. 2000, S. 95–100. URL: http://www.bisp.de/SharedDocs/Downloads/Publikationen/Jahrbuch/Jahrbuch2000_Inhalt.html?nn=3003714 (besucht am 06. 12. 2014).

[18] FIBT. *Videos*. URL: http://www.fibt.com/races-results/results/multimedia/videos.html (besucht am 29. 07. 2014).

[19] FIBT. *Two-men Bob Results - World Championship Calgary*. Season 2004/2005. URL: http://www.fibt.com/races-results/results.html (besucht am 23. 07. 2014).

[20] FIBT. *Four-men Bob Results - Olympic Games Cesana*. Season 2005/2006. URL: http://www.fibt.com/races-results/results.html (besucht am 23. 07. 2014).

[21] FIBT. *Two-men Bob Results - Olympic Games Whistler*. Season 2009/2010. URL: http://www.fibt.com/races-results/results.html (besucht am 23. 07. 2014).

[22] FIBT. *Four-men Bob Results - World Cup Sochi*. Season 2012/2013. URL: http://www.fibt.com/races-results/results.html (besucht am 23. 07. 2014).

[23] FIBT. *Two-men Bob Results - World Cup Whistler*. Season 2012/2013. URL: http://www.fibt.com/races-results/results.html (besucht am 23. 07. 2014).

[24] Freudenberg, H., Grüllich, F. und Maisser, P. „Expertensystem zur Abstimmung der Dynamik von Bobschlitten“. In: Bundesinstitut für Sportwissenschaft (BISp). *BISp-Jahrbuch Forschungsförderung 2005/06*. 2006, S. 359–362. URL: http://www.bisp.de/SharedDocs/Downloads/Publikationen/Jahrbuch/Jahrbuch_2005_06_Inhalt.html?nn=3003714 (besucht am 06. 12. 2014).

[25] Glocker, Ch. *Set-valued force laws: dynamics of non-smooth systems*. Lecture notes in applied mechanics. Springer, 2001.

[26] Glocker, Ch. „Introduction to Impacts“. In: *Nonsmooth Mechanics of Solids. CISM Courses and Lectures*. Hrsg. von Haslinger, J. und Stavroulakis, G. Bd. 485. Springer, 2006, S. 45–101.

[27] Glocker, Ch. „Simulation von harten Kontakten mit Reibung: Eine iterative Projektionsmethode“. In: *VDI-Berichte No. 1968: Schwingungen in Antrieben 2006 Tagung, Fulda*. Bd. 1968. Düsseldorf: VDI Verlag, 2006, S. 19–44.

[28] Glocker, Ch. „Simulation of Hard Contacts with Friction: An Iterative Projection Method“. In: *Recent Trends in Dynamical Systems*. Hrsg. von Johann, A. et al. Bd. 35. Springer Proceedings in Mathematics & Statistics. Springer Basel, 2013, S. 493–515. DOI: 10.1007/978-3-0348-0451-6_19.

[29] Günther, M., Kielau, G. und Maisser, P. „Simulation von Fahrten gesteuerter Bobschlitten“. In: *Zeitschrift für Angewandte Mathematik und Mechanik* 74(9), 1994, S. 434–435.

[30] Günther, M. und Maisser, P. „Anpassung eines Programmes zur Simulation von Fahrten gesteuerter Bobschlitten (Bob-Fahrsimulator) an die Erfordernisse des Trainingsbetriebes“. In: Bundesinstitut für Sportwissenschaft (BISp). *BISp-Jahrbuch 1999*. 1999, S. 239–246. URL: http://www.bisp.de/SharedDocs/Downloads/Publikationen/Jahrbuch/Jahrbuch1999_Inhalt.html?nn=3003714 (besucht am 06. 12. 2014).

[31] Hainzlmaier, C. „A new tribologically optimized bobsleigh runner“. Dissertation. Technische Universität München, 2005.

[32] Hamel, G. *Theoretische Mechanik*. Die Grundlehren der mathematischen Wissenschaften LVII. Springer, 1949. DOI: 10.1007/978-3-642-88463-4.

[33] Hubbard, M. „Simulating Sensitive Dynamic Control of a Bobsled“. In: *SIMULATION* 65(2), 1995, S. 147–151.

[34] Hubbard, M. „Luge Track Safety“. arXiv:1212.4901. 2012. URL: http://arxiv.org (besucht am 09. 12. 2014).

[35] Hubbard, M., Kallay, M. und Rowhani, P. „Three-dimensional bobsled turning dynamics“. In: *Journal of applied biomechanics* 5(2), 1989, S. 222–237.

[36] Hubbard, M. et al. „Simulation of vehicle and track performance in the bobsled“. In: *Biomechanics Symposium*. Third ASME/ASCE Mechanics Symposium. Hrsg. von Torzilli, P. A. und Friedman, M. H. San Diego: ASME, 1989, S. 373–376.

[37] Huffman, R. K. und Hubbard, M. „A motion based virtual reality training simulator for bobsled drivers“. In: *The Engineering of Sport*. Hrsg. von Haake, S. Balkema, 1996, S. 195–203.

[38] Huffman, R. K., Hubbard, M. und Reus, J. „Use of an interactive bobsled simulator in driver training“. In: *Advances in Bioengineering*. ASME Winter Annual Meeting. New Orleans, Nov. 1993.

[39] Ingenieurbüro Gurgel. *2010 Olympics - Whistler Sliding Centre. Part 01*. Final Design. WSC.I.001-002.3-centreline-a. F.I.L. und F.I.B.T., 5. Juli 2004.

[40] Ingenieurbüro Gurgel. *Whistler Sliding Centre - General Data of Curve*. WSC.I.005.0. 10. Sep. 2004.

[41] Ingenieurbüro Gurgel. *Whistler Sliding Centre - Profile data*. Datensatz. 10. Sep. 2010.

[42] Ingenieurbüro Gurgel. *Whistler Sliding Centre - Drawing of Centreline*. WSC.I.001-002.3-centreline-a. undatiert.

[43] Ingenieurbüro Gurgel. *Whistler Sliding Centre - Drawing of Longitudinal Section*. WSC.I.004.4-longitudinal-section. undatiert.

[44] Ingenieurbüro Gurgel+Partner. URL: `http://www.ibg-gurgel.de/` (besucht am 02. 10. 2014).

[45] Ingenieurbüro Gurgel+Partner. *Sochi Sliding Center - Bobsleigh, Skeleton and Luge Track*. Preliminary Design Phase. Version Version 5.2. SSC.FD.T.003.0. 1. Juli 2009.

[46] Ingenieurbüro Gurgel+Partner. *Sochi Sliding Centre - Concrete Profile*. SSC.FD.T-RD.300-322.0/1. 11. Nov. 2009.

[47] Ingenieurbüro Gurgel+Partner. *Sochi Sliding Centre - General Data of Profiles*. SSC.FD.T.005.0. 11. Nov. 2009.

[48] Ingenieurbüro Gurgel+Partner. *Sochi Sliding Centre - Longitudinal Section*. SSC.FD.T.004.0. 11. Nov. 2009.

[49] Ingenieurbüro Gurgel+Partner. *Sochi Sliding Centre - Ground Plan*. SSC.FD.T.003.0. Jan. 2010.

[50] Iwashita, K., Taniguchi, H. und Konno, E. „Development of a simulation system for bobsleigh sliding“. In: *Developments in computational techniques for civil engineering*. Hrsg. von Topping, B. H. V. Civil-Comp Press, 1995, S. 273–277. DOI: `10.4203/ccp.32.10.1`.

[51] Jahnke, E. und Emde, F. *Funktionentafeln (Tables of Functions). mit Formeln und Kurven (with Formulae and Curves)*. 3. neubearbeitete Auflage (3rd revised Edition). zweisprachig (bilingual). B. G. Teubner, 1938.

[52] Kelly, A. und Hubbard, M. „Design and construction of a bobsled driver training simulator“. In: *Sports Engineering* 3, 2000, S. 13–24.

[53] Kielau, G. und Maisser, P. „Nonholonomic Multibody Dynamics“. In: *Multibody System Dynamics* 9, 2003, S. 213–236.

[54] Krumke, S. O. und Noltemeier, H. *Graphentheoretische Konzepte und Algorithmen*. 3. Auflage. Leitfäden der Informatik. Vieweg + Teubner, 2012.

[55] Leine, R. I. „On the Stability of Motion in Non-smooth Mechanical Systems“. Habilitation. ETH Zürich, 2006.

[56] Levien, R. „The Euler spiral: a mathematical history“. Online. 30. Aug. 2008. URL: `http://www.levien.com/phd/` (besucht am 16. 10. 2014).

[57] Levy, R. M. und Katz, L. „Virtual Reality Simulation: Bobsled and Luge“. In: *IACSS International Symposium Computer Science in Sport*, 2007.

[58] Maisser, P. „Brachystochronen als zeitkürzeste Fahrspuren von Bobschlitten“. In: *Zeitschrift für Angewandte Mathematik und Mechanik* 78(5), 1998, S. 311–319.

[59] Maisser, P. und Grüllich, F. „Optimales Layout von Bobschlitten (Bobdynamik) - Bob-Expertensystem“. In: Bundesinstitut für Sportwissenschaft (BISp). *BISp-Jahrbuch 2003*. 2003, S. 403–408. URL: http://www.bisp.de/SharedDocs/Downloads/Publikationen/Jahrbuch/Jahrbuch_2003_Inhalt.html?nn=3003714 (besucht am 06. 12. 2014).

[60] Maisser, P. *Tätigkeitsbericht des Institutes für Mechatronik e.V. an der Technischen Universität Chemnitz*. TU Chemnitz, 2001. URL: https://www.tu-chemnitz.de/ifm/publikat/ifm_tb01.pdf (besucht am 06. 12. 2014).

[61] Maisser, P. *Tätigkeitsbericht des Institutes für Mechatronik e.V. an der Technischen Universität Chemnitz*. TU Chemnitz, 2002. URL: https://www.tu-chemnitz.de/ifm/publikat/ifm_tb02.pdf (besucht am 06. 12. 2014).

[62] Meek, D. S. und Walton, D. J. „An arc spline approximation to a clothoid“. In: *Journal of Computational and Applied Mathematics* 170, 2004, S. 59–77.

[63] Möller, M. *Persönliche Kommunikation*. 2009 - 2012.

[64] Möller, M. „Consistent Integrators for Non-Smooth Dynamic Systems“. Dissertation. ETH Zürich, 2011.

[65] Möller, M. und Glocker, Ch. „Rigid body dynamics with a scalable body, quaternions and perfect constraints“. In: *Multibody System Dynamics* 27(4), 2012, S. 437–454. DOI: 10.1007/s11044-011-9276-5.

[66] Möller, M., Leine, R. und Glocker, Ch. „An efficient approximation of orthotropic set-valued force laws of normal cone type“. In: Hrsg. von et.al., J. A. 7th EUROMECH Solid Mechanics Conference. Lisbon, Portugal, Juli 2009.

[67] Moreau, J. J. „Some Numerical methods in multibody dynamics: application to granular materials“. In: *European Journal of Mechanics A/Solids* 13(4 - suppl.), 1994, S. 93–114.

[68] Mössner, M. et al. „An approximate simulation model for initial luge track design“. In: *Journal of Biomechanics* 44, 2011, S. 892–896.

[69] Ogino, M. et al. „A Real-feel Bobsleigh Simulator Based on Motion and Surround Screen“. In: *Technical report of IEICE. Multimedia and virtual environment* 104(390), 2004, S. 31–35. URL: http://ci.nii.ac.jp/naid/110003271105/en/.

[70] Ogino, M. et al. „Development of a Ride Training Simulator for Bobsleigh Drivers“. In: *Transactions of the Virtual Reality Society of Japan* 11(4), 2006, S. 469–477. URL: http://ci.nii.ac.jp/naid/110008728990/en/.

[71] Ogino, M. et al. „Enhancement of bobsleigh simulation reactive force (International Workshop on Advanced Image Technology 2009)“. In: *IEICE technical report. Image engineering* 108(373), 2009, S. 143–147. URL: http://ci.nii.ac.jp/naid/110007123147/en/.

[72] Pagé, F. und Guibault, F. „Collision Detection Algorithm for NURBS Surfaces in Interactive Applications“. In: *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*. Bd. 2. Mai 2003, S. 1417–1420. DOI: 10.1109/CCECE.2003.1226166.

[73] Piegl, L. und Tiller, W. *The NURBS Book*. 2nd Edition. Springer, 1997.

[74] Poirier, L. et al. „Experimental analysis of ice friction in the sport of bobsleigh“. In: *Sports Engineering* 14(2–4), 2011, S. 67–72. DOI: 10.1007/s12283-011-0077-0.

[75] Python Software Foundation. *Python programming language*. URL: https://www.python.org (besucht am 01. 12. 2014).

[76] Scherge, M. et al. „High-Speed Ice Friction Experiments under Lab Conditions: On the Influence of Speed and Normal Force“. In: *ISRN Tribology* 2013, 2013. DOI: 10.5402/2013/703202.

[77] Schweizer, A. P. „Ein nichtglattes mechanisches Modell für Steinschlag“. Dissertation. ETH Zürich, 2015.

[78] SciPy developers. *SciPy library*. Python library for efficient numerical routines. 2014. URL: http://scipy.org/scipylib/index.html (besucht am 02. 07. 2015).

[79] Studer, C. „Augmented time-stepping integration of non-smooth dynamical systems“. Dissertation. ETH Zürich, 2008.

[80] Studer, C. und Glocker, Ch. „Representation of normal cone inclusion problems in dynamics via non-linear equations“. In: *Archive of Applied Mechanics* 76, 2006, S. 327–348.

[81] TU Chemnitz. *Bobsimulator*. 1992-2004. URL: https://www.tu-chemnitz.de/ifm/produkte-html/Simulator_Bob.html (besucht am 06. 12. 2014).

[82] UC Davis. *Virtual Reality Bobsled Simulator*. 1991-1998. URL: http://biosport.ucdavis.edu/sample-page/test-page-1/virtual-reality-bobsled-simulator/ (besucht am 06. 12. 2014). Vorstudie: Bobsled 3-D Motion Dynamics, 1988.

[83] Ulman, D. G. und Cross, C. „Engineering a New-Generation Bobsled“. In: *ASME paper*, 1979. 79-DE-E-5.

[84] Volkmann, L. *Fundamente der Graphentheorie*. Springer-Lehrbuch: Mathematik. Springer, 1996.

[85] Wittenburg, J. *Dynamics of Multibody Systems*. 2nd Edition. Originally published under: Dynamics of Systems of Rigid Bodies, in the LAMM series, Teubner 1977. Springer, 2008.

[86] Zahavich, A. et al. *Whistler Sliding Centre Sled Trajectory and Track Construction Study*. Techn. Ber. SAIT Polytechnic, Okt. 2012.

[87] Zhang, Y., Hubbard, M. und Huffmann, R. „Optimum Control of Bobsled Steering“. In: *Journal of Optimization Theory and Applications* 85(1), 1995, S. 1–19.

Anhang A

Querschnittskonstruktionsdaten

Das *Ingenieurbüro Gurgel+Partner* [44] aus Leipzig in Deutschland ist auf den Entwurf, Planung und Berechnung von Bob- und Rodelbahnen spezialisiert und war am Bau von vielen Kunsteisbahnen beteiligt. Unter Anderem konzipierte es die Bahnen der letzten drei Winterolympiaden in Turin, Whistler und Sochi. Es kann deshalb als der führende Bahnkonstrukteur bezeichnet werden. Zu Beginn des Bobsimulatorprojektes standen Baupläne der Bahn von Whistler, Kanada zur Verfügung, die aus dem Projekt *Citius* [4] stammten. Während des Projektes kamen wir zwischenzeitlich mit dem *Ingenieurbüro Gurgel+Partner* in Kontakt zwecks eines Gutachtens für den Bau der Bahn in Sochi für die Winterolympiade 2014. Im Zuge dieses Kontaktes erhielten wir detailliertere Baupläne der Bahn von Whistler und insbesondere vertieften Einblick in die Konstruktion von Kunsteisbahnen. Später sind wir auch in den Besitz von Bauplänen der Bahn in Sochi gelangt. Diese Arbeit beschränkt sich aus Gründen der Verfügbarkeit auf Konstruktionsdaten, wie sie das *Ingenieurbüro Gurgel+Partner* verwendet. Aufgrund der Marktdominanz dieses Konstrukteurs erscheint diese Beschränkung aber nicht schwerwiegend und es zeigt sich, dass andere Konstrukteure grossenteils mit denselben Geometrien und Parametern arbeiten.

Abbildung A.1 stammt vom *Ingenieurbüro Gurgel+Partner* und zeigt dessen Bemassung von Querschnittsgeometrien. Für Querschnitte listen die Konstruktionsrohdaten 27 Bemassungen pro Querschnitt, wobei weitere fünf Grössen für die definitive Positionierung auf dem Streckenprofil nötig sind. Vier setzen sich aus dem Streckenprofilpunkt $(s_i, {}_{\mathrm{T}}\boldsymbol{r}_{OC_i}, \beta_i)$ an der zum Querschnitt gehörenden Stelle $s_i = \mathsf{Sx}$ zusammen. Ausserdem muss anhand der Richtung c der Krümmung derjenigen Kurve, in der dieser Querschnitt zu liegen kommt, entschieden werden, ob die überhöhte Seite des Querschnitts rechts oder links liegt. Allein aus der Bemassung lässt sich nämlich nicht schliessen, ob es sich um eine Rechts- oder Linkskurve handelt. Von den resultierenden 32 Parametern pro Querschnitt sind, unter der Annahme, dass ein Querschnitt eine geometrisch glatte Kurve sein soll, 14 Angaben redundant. Die Annahme eines glatten Querschnitts lässt sich damit motivieren, dass man im Hinblick auf eine möglichst gleichmässige Eisoberfläche, wohl schon die darunterliegende Betonoberfläche geometrisch glatt konstruiert. D.h. die Betonoberfläche wird ohne Kanten oder gar Ecken konstruiert. Diese Annahme legt auch Nahe, dass im

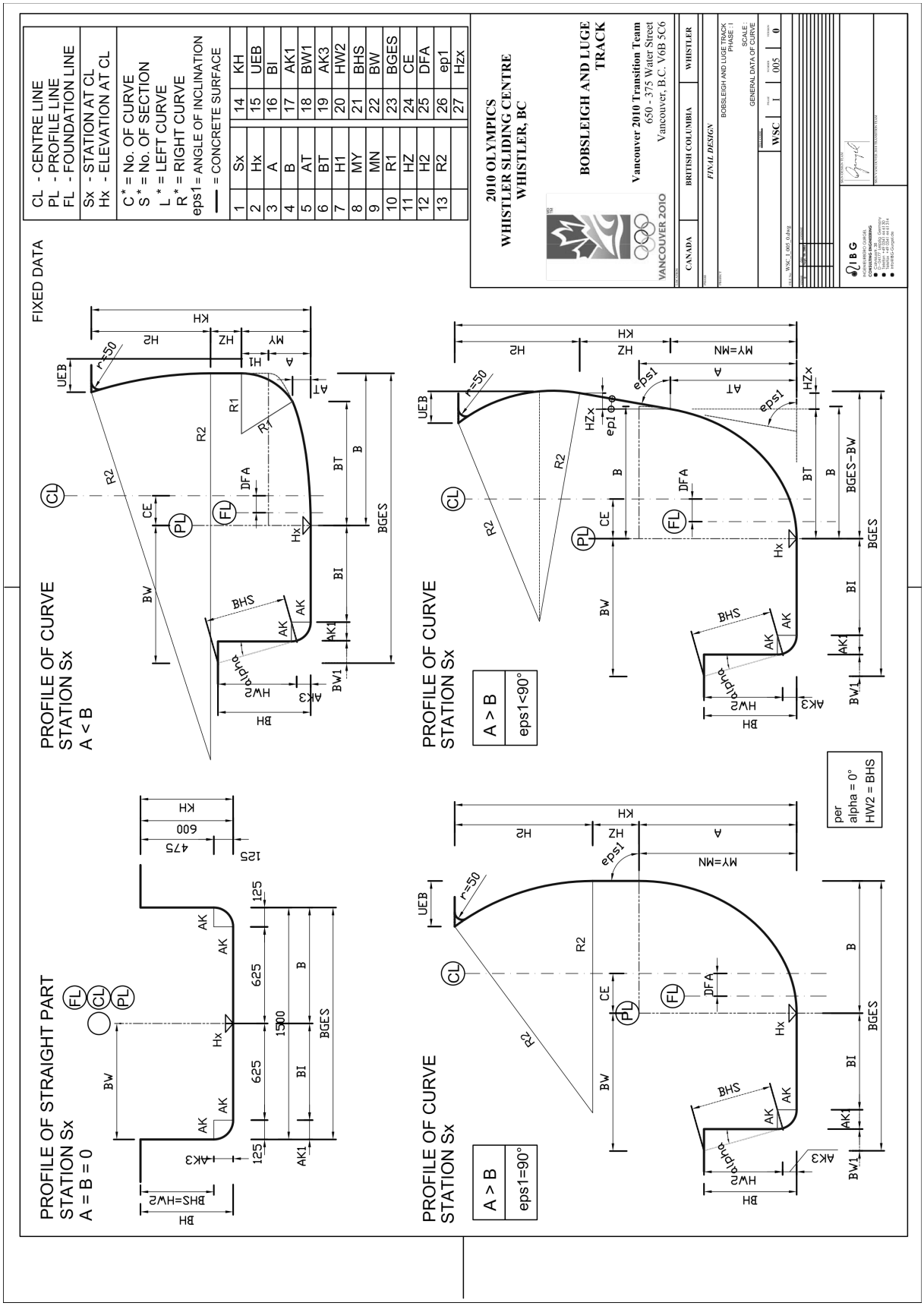

Abbildung A.1: Bemassungen der Bahnquerschnittsgeometrie gemäss Ingenieurbüro Gurgel, Leipzig. (Quelle: IBG, Leipzig, 2004 [40].)

Bobsimulator die Betonoberfläche als Fahroberfläche hergenommen werden kann und nicht zusätzlich auch eine Eisschicht modelliert werden muss.

Die digitale Rekonstruktion eines Querschnitts, wie sie in Abschnitt 3.1.4 beschrieben ist, basiert auf einer Auswahl von 20 von den verfügbaren 32 Konstruktionsparametern. Da die Rohdaten gerundete Werte angegeben, werden aus Gründen der Robustheit bewusst zwei redundante Parameter miteinbezogen. Die Rundungsungenauigkeiten können sonst zu inakzeptablen Deformationen der Querschnittsgeometrien führen. Mit einer situativen Neuberechnung der voneinander abhängigen Masse können die Auswirkungen der Rundungsfehler minimiert werden. Die ersten fünf Parameter werden, wie bereits erwähnt, für die Positionierung des Querschnitts benötigt. Es handelt sich dabei um den Streckenprofilpunkt $(s_i, {}_{\mathrm{T}}\boldsymbol{r}_{OC_i}, \beta_i)$ an der Stelle $s_i = \mathsf{Sx}$. Ein weiterer Parameter ist die Kurvenkrümmung c, der sich anhand von $s = \mathsf{Sx}$ aus den Konstruktionsdaten für die Profillinie extrahieren lässt. Elf Konstruktionsparameter

$$\begin{aligned} h_W &= \mathsf{HW2}, \quad r_0 = \mathsf{AK}, \quad l_B = \mathsf{BI}, \quad d_C = \mathsf{CE}, \\ b &= \mathsf{B}, \quad a = \mathsf{A}, \quad r_1 = \mathsf{R1}, \quad h_Z = \mathsf{HZ}, \\ r_2 &= \mathsf{R2}, \quad h_2 = \mathsf{H2}, \quad \varepsilon = \frac{\pi}{180^\circ} \begin{cases} \mathsf{eps1}, & \mathsf{eps1} \geq 45^\circ, \\ 90^\circ - \mathsf{eps1}, & \mathsf{eps1} < 45^\circ \end{cases} \end{aligned} \tag{A.1}$$

werden von den Konstruktionsrohdaten[10] direkt übernommen, wobei inkonsistenterweise manchmal anstelle des Winkels $\mathsf{eps1}$ dessen Komplement zum rechten Winkel, eigentlich $\mathsf{ep1}$ genannt, angegeben ist. Man vergleiche dazu auch Abbildung A.1. Bei den verbleibenden drei Konstruktionsparametern handelt es sich um die redundant zu berechnenden. Das Ellipsensegment wäre mit drei, anstelle von vier Parametern bereits vollständig definiert. Im Falle einer Degeneration der Ellipse zu einem Liniensegment auf geraden Bahnabschnitten reichte bereits ein Parameter, der die Länge definiert. Unter der Annahme, dass die Querschnittsgeometrie eine glatte Kurve sein soll, sind weiter der Winkel ε und die Höhe h_1 voneinander abhängige Parameter. Die Miteinbeziehung des Winkels vereinfacht die Parametrisierung eines Querschnitts allerdings erheblich und reduziert den Rechenaufwand. Im Falle einer zur Linie degenerierten Ellipse spezifizieren die Konstruktionsdaten $\mathsf{A} = \mathsf{AT} = \mathsf{BT} = 0$ mit Null, was insbesondere für BT unintuitiv sein mag. Die Konstruktionsparameter b_T und a_T werden in diesem Fall nach

$$\begin{aligned} \text{(degenerierte Ellipse) } \mathsf{AT} = 0: \quad b_T &= \mathsf{B} - \mathsf{R1}, \\ a_T &= 0 \end{aligned} \tag{A.2}$$

[10] Der Parameter $d_C = \mathsf{CE}$ bezeichnet den lateralen Versatz zum Punkt C auf dem Streckenprofil. Das Streckenprofil wird im Englischen *centre line* genannt und deshalb auch in Abbildung A.1 so vermerkt und mit CL abgekürzt.

eruiert. Ist eine (nicht degenerierte) Ellipse vorhanden werden diese Parameter nach

$$\begin{aligned}\text{(Ellipse) } \mathsf{AT} \neq 0: \quad b_T &= \begin{cases} \mathsf{BT}, & \mathsf{BT} > \mathsf{AT} \\ \mathsf{B}\sqrt{1-\left(\frac{\mathsf{AT}-\mathsf{A}}{\mathsf{A}}\right)^2}, & \mathsf{BT} \leq \mathsf{AT} \end{cases} \\ a_T &= \begin{cases} \mathsf{A}\left(1-\sqrt{1-\left(\frac{\mathsf{BT}}{\mathsf{B}}\right)^2}\right), & \mathsf{BT} > \mathsf{AT} \\ \mathsf{AT}, & \mathsf{BT} \leq \mathsf{AT} \end{cases} \end{aligned} \tag{A.3}$$

bestimmt, wobei sich diese Gleichungen aus der Ellipsengleichung

$$\frac{x}{\mathsf{B}^2} + \frac{(y-\mathsf{A})^2}{\mathsf{A}^2} = 1 \tag{A.4}$$

ergeben. Die Höhe h_1 des zweiten Kreissegments wird nach

$$h_1 = \begin{cases} \mathsf{H1}, & \mathsf{H1} = 0, \\ r_1\left(\sin(\varepsilon)\sin(\theta) - \cos(\varepsilon)(1-\cos(\theta))\right), & \mathsf{H1} \neq 0 \end{cases} \tag{A.5}$$

in Abhängigkeit des Winkel ε bestimmt, wobei

$$\varepsilon_0 = \tan^{-1}\left(\frac{a\, b_T}{b\sqrt{b^2 - b_T^2}}\right) \qquad \text{und} \qquad \theta = \varepsilon - \varepsilon_0. \tag{A.6}$$

Hierbei bezeichnet $\tan(\varepsilon_0)$ die Steigung im Ellipsenendpunkt $\boldsymbol{r}_{34} = (b_T, a_T, 0)^\mathsf{T}$ und θ den Winkel des anschliessenden Kreissegments. Aus der Ellipsengleichung (A.4) lässt sich die Funktion $y(x) = \mathsf{A} - \sqrt{\mathsf{A}^2(1-\frac{x^2}{\mathsf{B}^2})}$ herleiten, welche für $x \in [0; \mathsf{b}]$ das Ellipsensegment beschreibt. Die Gleichung (A.6) für ε_0 folgt aus der Ableitung dieser Funktion nach x ausgewertet an der Stelle $x = b_T$.