Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

DIPLOMARBEIT

WERKZEUGE ZUR DYNAMISCHEN GEOMETRIESIMULATION VON SCHIENENFAHRZEUGEN

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs unter der Leitung von

O. Univ.-Prof. Dipl.-Ing. Dr. techn. Peter Lugner
E 325/I
Institut für Mechanik

eingereicht an der Technischen Universität Wien
Fakultät Maschinenbau

von

Gerhard Schandl
Matr. Nr. 9627102
Stoßgasse 44, 3100 St. Pölten

Wien, im September 2002

Vorwort

An dieser Stelle möchte ich mich für die Betreuung am Institut für Mechanik während der Abfassung meiner Diplomarbeit bedanken. Mein besonderer Dank gilt Herrn O. Univ.-Prof. Dipl.-Ing. Dr. techn. Peter Lugner für die Durchsicht und die Anregungen zu meiner Arbeit.

Weiters möchte ich mich bei den Mitarbeitern von Siemens SGP Verkehrstechnik, insbesondere Herrn Dipl.-Ing. Dr. techn. Anton Stribersky, Herrn Dipl.-Ing. Nikolaus Böck und Herrn Dipl.-Ing. Dr. techn. Joachim Pargfrieder für die Unterstützung bei meiner Diplomarbeit bedanken.

Mein Dank gilt auch meiner Schwester Angelika Schandl für die Durchsicht der Arbeit auf grammatikalische Fehler und Rechtschreibfehler.

Für die Unterstützung während meines Studiums möchte ich mich bei meiner Familie bedanken.

Gerhard Schandl, Wien, September 2002

Inhaltsverzeichnis

Kapitel 1. Aufgabenstellung

Die Integration bereits bisher vorhandener Werkzeuge zur computerunterstützten Entwicklung von Schienenfahrzeugen wie Finite Elemente Methoden (FEM), Mehrkörperdynamiksimulation (**M**ulti-**B**ody-**S**imulation - MBS) und Computer Aided Design (CAD) in einem Virtual Product Development (VPD) System ermöglicht den Bau virtueller Prototypen (Digital Mock Up, DMU).

Damit können Fehler bereits früh im Entwicklungsprozess, vor dem Bau teurer und zeitaufwendiger realer Prototypen, erkannt und korrigiert werden. Die Entwicklungszeit kann dadurch verkürzt werden.

Die Aufgaben dieser Diplomarbeit sind:

- Durchführung einer Literaturstudie zum Stand der Technik aktiver DMU-Werkzeuge
- Entwicklung und Aufbau einer Strategie zur Kollisionsuntersuchung starrer 3-D-Körper
- Aufbau eines vereinfachten Mehrkörperdynamikmodells eines Schienenfahrzeugs in SIMPACK (MBS-Software), Berechnung der Dynamik in SIMPACK und Einbringung dieser Dynamik- sowie der Geometriedaten dieses Fahrzeugs in eine DMU-Software (Centric Innovation)
- Verifikation der Leistungsfähigkeit eines DMU zur Kollisionserkennung (Centric Innovation)
- Verallgemeinerung der Vorgangsweise für die industrielle Anwendung im Rahmen der Schienenfahrzeugentwicklung

Kapitel 2. Einleitung und Literaturstudie

Seit einigen Jahren werden in der Produktentwicklung zunehmend Softwarelösungen zur Erstellung eines virtuellen Prototypen verwendet. Im Folgenden wird ein Überblick über die Definition des Begriffs Digital Mock-up (DMU), die Anwendung dieser Technik und verfügbare kommerzielle DMU-Softwarepakete gegeben. Weil für die vorliegende Diplomarbeit die Kollisionserkennung innerhalb der DMU-Software benutzt wurde, wird auch ein Überblick über die theoretischen Grundlagen der Kollisionserkennung gegeben.

2.1 Definition und Eigenschaften von DMU Systemen

Eine Definition des Begriffs Digital Mock-up wird in [1] angegeben: *Ein virtueller Prototyp, oder ein Digital Mock-up, ist eine Computersimulation eines realen Produkts, die unter den Aspekten Entwurf/Entwicklung, Herstellung, Einsatz und Recycling* (Anmerkung des Autors: Je nach Verwendung) *ebenso dargestellt und analysiert werden kann wie ein reales physisches Modell. Die Entwicklung und das Testen des virtuellen Prototypen wird Virtual Prototyping (VP) genannt.*

Im Unterschied dazu versteht man unter Virtual Reality die Computersimulation eines Systems, die dem Benutzer ermöglicht, Handlungen in diesem System auszuführen und ihre Effekte in Echtzeit zu erfahren [1]. Bei VR-Systemen erhält der Nutzer durch entsprechende Geräte den optischen und zum Teil auch akustischen und haptischen Eindruck sich innerhalb der Simulation zu befinden. Im Englischen wird dies als *immersion* bezeichnet [2]. Dies ist bei VP-Systemen nicht notwendigerweise der Fall. Allerdings kann VR die Möglichkeiten von VP erweitern.

Eines der wesentlichen Ziele der Einführung von DMU besteht darin, kosten- und zeitaufwendige Musterbauten zu reduzieren oder überhaupt zu vermeiden, indem man sie durch digitale Modelle und Simulationen ersetzt. Es wird dadurch mit erheblichen Verkürzungen in den Produktentwicklungszeiträumen gerechnet. Die schnelle Verfügbarkeit digitaler Prototypen bietet außerdem die Möglichkeit Entscheidungen jeweils aufgrund aktueller Daten des Entwicklungsprozesses zu fällen. Dadurch ist das Restrisiko geringer, dass aufgrund unvollständiger oder veralteter Daten Fehlentscheidungen getroffen werden [2].

Digital Mock-up Techniken ermöglichen damit, Entwicklungszeiten zu verkürzen, die Qualität zu erhöhen und die Kosten zu senken, da 70% der Kosten über den gesamten Produktlebenszyklus durch Entscheidungen, die in der Frühphase der Entwicklung getroffen werden, beeinflusst werden [3].

Die Grundfunktionalität jeder DMU-Software besteht aus einem Modul, mit dem 3D-Modelle, die in verschiedenen CAD-Formaten vorliegen, dargestellt werden können. In diesem Modul können die Bauteile dann manipuliert und mit Anmerkungen versehen werden. Auch wird hier üblicherweise eine Möglichkeit geboten, Schnitte durch die Modelle zu legen. Die Bedienung der Software ist gegenüber einer 3D-CAD-Anwendung

wesentlich einfacher gestaltet und ermöglicht auch Anwendern ohne 3D-CAD-Schulung die Nutzung. Durch die Reduktion der vom CAD-System gelieferten Daten auf Hüllgeometrien kann auch eine große Anzahl von komplexen Elementen dargestellt werden. Häufig wird auch der Zugriff auf und die Verwaltung von Dokumenten in zahlreichen Formaten innerhalb der DMU-Software unterstützt, sodass ein Zugriff auf alle für ein Projekt relevanten Daten über eine einheitliche Oberfläche erfolgen kann.

Zusätzlich ist typischerweise die Möglichkeit vorhanden, Modelle einfache Bewegungsabläufe durchführen zu lassen und sie dabei auf ihre Funktion zu untersuchen. Dies wird zum Beispiel zur Durchführung von Einbauanalysen genutzt. Bei manchen DMU-Programmen ist es auch möglich, den Modellen mit Logiken Funktionalitäten zu verleihen und Resultate aus Mehrkörperdynamik-Simulationsprogrammen einzulesen und damit dynamische Bewegungsabläufe darzustellen.

Weiters enthalten manche der DMU-Programmpakete Module, die die Zusammenarbeit an mehreren Orten in einem Netzwerk ermöglichen. Damit können Entwickler an unterschiedlichen Standorten zeitgleich oder zeitversetzt Simulationen an einem virtuellen Prototyp durchführen.

2.2 Anwendungen der DMU-Technik

DMU-Technik ist heute, ausgehend von der Automobil- und Luftfahrtindustrie, eine in der Produktentwicklung häufig eingesetzte Technik.

Die Anwendungsbereiche sind hier zum Beispiel [3,4]:

- Bewertung von Innen- und Außendesign durch fotorealistische Darstellung
- Bauraumuntersuchungen (zum Beispiel für die Anordnung von Nebenaggregaten im Motorraum eines PKW). Hier muss teilweise auch eine kinematische Simulation der Bauteilbewegung durchgeführt werden (z. B. für Fahrwerksteile). In Kapitel 2.4 wird auch die Vorgangsweise für die Einbindung von dynamischen Bewegungsdaten in eine solche Untersuchung anhand eines Beispiels von Audi erläutert.
- Fertigungssimulation. Hier wird die Herstellbarkeit des Produkts abgesichert. Zu diesem Zweck müssen die einzelnen Bauteile in der Montagereihenfolge auf den bei der Montage verwendeten Bahnen bewegt und auf Kollisionsfreiheit geprüft werden. Dabei muss auch der Bauraumbedarf von für die Montage notwendigen Werkzeugen (z. B. Greifer) berücksichtigt werden.
- Ergonomieuntersuchungen, z. B. am Fahrerplatz. Zu diesem Zweck wird die Darstellung des DMU durch VR-Techniken ergänzt [5].
- Demontageprüfung, z. B. um die Wartbarkeit des Produkts sicherzustellen.

Die Bauteilbewegung, die bei einigen dieser Aufgaben gefordert ist, erfolgt im allgemeinen entweder durch kinematische Vorgaben oder auch (bei VR-Systemen) durch Benutzereingaben.

Um dem DMU tatsächlich gleiche Eigenschaften zu verleihen, wie sie ein physikalischer Prototyp bietet, ist es notwendig, dass auch das dynamische Verhalten des DMU einem realen Prototypen entspricht. Da im DMU nur Informationen über die Geometrie vorliegen, muss zu diesem Zweck ein Datenaustausch mit einer Mehrkörperdynamiksoftware erfolgen. Beim im Rahmen dieser Diplomarbeit durchgeführten Beispiel geschieht dies durch Vorabberechnung der Dynamik und Übergabe der Bewegungsdaten an die DMU-Software.

Für eine realistische *Human-in-the-Loop*-Simulation (z. B. Bedienung des Kraftfahrzeugs durch reale Eingabegeräte, wie Lenkung, Pedale, etc. und Berechnung und Darstellung des DMU entsprechend dieser Eingaben), wie sie zum Teil gefordert wird, ist dies aber nicht ausreichend [6]. Für solche Anwendungen wird eine Echtzeitfähigkeit gefordert, um dem Benutzer eine realistische Rückmeldung über die Auswirkungen seiner Eingaben zu bieten. Dies ist aber mit den aktuell angebotenen Programmpaketen nicht möglich [6].

2.3 Überblick über kommerzielle DMU-Produkte

Im Folgenden wird ein Überblick über vier kommerzielle DMU-Produkte und ihre Eigenschaften geboten.

2.3.1 ENOVIA

ENOVIA ist ein von IBM und Dassault Systems entwickeltes und vertriebenes Digital-Mock-up-System. Die ENOVIA Anwendungen werden folgendermaßen gegliedert [7]:

- ENOVIA Portal
  - ENOVIA DMU
  - ENOVIA 3d com
  - ENOVIA MultiCAx
  - ENOVIA MultiPDM
- ENOVIA Life Cycle Applications
  - ENOVIA LCA V5
  - ENOVIAVPM
  - ENOVIAPM
- ENOVIA PPR Hub
- ENOVIA Enterprise Architecture
- ENOVIA RADEnvironment (Rapid Development Environment)

ENOVIA Portal
ENOVIA Portal ist webbasiert und bietet einen zentralen Zugriff auf Produkt-, Prozess/ und Ressourceinformationen eines Unternehmens, unabhängig vom Ort und Format der gespeicherten Daten.

ENOVIA DMU ermöglicht die Überprüfung des digitalen Modells mit Kollisionsuntersuchungen, Schnitten, Messungen und Abstandsanalysen. Die Simulation und Analyse von Montage- und Demontageprozessen sowie das Erstellen fotorealistischer Bilder sind weitere Funktionen, die hier geboten werden.

ENOVIA 3d com stellt die Visualisierungsumgebung für 2D- und 3D-Konstruktionsdaten bereit, ermöglicht den Zugriff auf alle mit der Produktentwicklung zusammenhängenden Daten, wie zum Beispiel auch MS Office-Dokumente, und ist die einheitliche Oberfläche für alle Benutzer, die in den Produktentwicklungsprozess eingebunden sind.

ENOVIA MultiCAx ermöglicht den Datenaustausch mit einer großen Zahl von 3D- und 2D-CAD-Systemen, FEM-Programmen und universellen Datenaustauschformaten wie

- Pro/ENGINEER (3D-CAD-Software)
- CATIA (3D-CAD-Software)
- Solid Works (3D-CAD-Software)
- AutoCAD (CAD-Software)
- I-DEAS (FEM-Software)
- UniGraphics (beinhaltet unter anderem 3D-CAD-Funktion)
- VRML (**V**irtual **R**eality **M**arkup **L**anguage, Datenformat zum Austausch von 3D-Geometrie-Daten)
- IGES (**I**nitial **G**raphics **E**xchange **S**pecification, Datenformat zum Austausch von 3D-CAD-Daten)

ENOVIA MultiPDM bietet die Möglichkeit auf Produktdaten im PDM-System des Unternehmens zuzugreifen.

ENOVIA Life Cycle Applications
Die ENOVIA Life Cycle Applications bieten die Funktionen herkömmlicher PDM-Systeme wie

- Konfigurationsverwaltung
- Dokumentenverwaltung
- Änderungsverwaltung

und Informationen über das Modell wie zum Beispiel den konfigurierten digitalen Prototypen.

ENOVIA PPR Hub
ENOVIA PPR Hub stellt die Modellerstellungs- und Integrationsfunktionen bereit, die erforderlich sind, um Produkt- und Prozessdaten zu Unternehmenswissen zusammenzuführen. Hierzu werden Objekte, aus denen sich das Projekt zusammensetzt und die es beschreiben, die Fertigungsprozesse und die Ressourcenanforderungen zu einem Netz verbunden [8]. Dadurch kann

- technisches Wissen in der Datenbank erfasst und verwaltet werden.
- eine genaue Projektplanung gewährleistet werden, da beim Navigieren durch Verknüpfungen Objektabhängigkeiten erkannt werden, wobei alle Folgen, die eine Änderung nach sich zieht, ausgewertet werden können.
- eine eindeutige Begründung technischer Entscheidungen gewährleistet werden, da der Weg von Spezifikation zu Ergebnis leicht zu verfolgen ist.

ENOVIA Enterprise Architecture
ENOVIA Enterprise Architecture ermöglicht Produktentwicklungsprozesse in sehr anspruchsvollen Umgebungen. Diese Umgebungen sind zum Beispiel Unternehmen mit einer Vielzahl von Unternehmensstandorten oder eine große Zahl inkompatibler Informationssysteme an diesen Standorten.

ENOVIA RADEnvironment
ist eine Anwendungsentwicklung, mit der Standardentwicklungstools unter Windows in C++ erstellt und erweitert werden können.

2.3.2 VisMockup

Die VisMockup Software wurde von dem Unternehmen Engineering Animation Inc. entwickelt. Dieses Unternehmen ist heute Teil von EDS, angesiedelt in Plano, Texas [9]. VisMockup bietet die für DMU-Systeme üblichen Funktionalitäten, wie Kollisionsuntersuchung, und die Möglichkeit, Schnitte durch ein Modell zu legen. Die Geometrien der Modelle werden durch den Import von 3D-CAD-Daten festgelegt, wobei hier Schnittstellen zu

- I-DEAS Master Series (FEM-Software)
- Pro/ENGINEER (3D-CAD-Software)
- CATIA (3D-CAD-Software)
- Unigraphics (beinhaltet unter anderem 3D-CAD-Funktion)
- CADDS (3D-CAD-Software)
- STEP (**ST**andard for the **E**xchange of **P**roduct Data, internationaler Standard für den Datenaustausch zwischen verschiedenen Systemen (CAD, CAM, CAE, PDM)
- IGES (**I**nitial **G**raphics **E**xchange **S**pecification, Datenformat zum Austausch von 3D-CAD-Daten)

- DXF (**D**rawing E**X**change **F**ormat, Datenformat zum Austausch von 2D-CAD-Daten)
- VRML und (**V**irtual **R**eality **M**arkup **L**anguage, Datenformat zum Austausch von 3D-Geometrie-Daten)
- STL (**ST**ereo**L**ithography Format, Standard Ausgabeformat für 3D-Daten, das von zahlreichen CAD-Systemen unterstützt wird)

vorhanden sind.

Zusätzlich ermöglicht VisMockup den Datenaustausch mit der Mehrkörperdynamiksoftware ADAMS und kann damit komplexe Bewegungsdaten darstellen und den digitalen Prototypen mit einem realistischen Bewegungsverhalten versehen [10].

Als weitere Ergänzung von VisMockup zu einer Programmplattform für Collaborative Engineering werden unter anderem angeboten:

- e-Vis, eine internetbasierte Plattform, die den unternehmensweiten Zugriff auf Produktdaten ermöglicht.
- Vis Publish sorgt dafür, dass immer auf aktuelle Produktdaten zugegriffen wird.
- Vis Quality ermöglicht die Darstellung und Verknüpfung von Messdaten aus der Produktion mit den 3D-Geometrien und ermöglicht damit eine Überwachung der Produktqualität. Weiters kann es zum Beispiel zur Unterstützung bei Montageproblemen eingesetzt werden.

2.3.3 Centric Innovation

Die Centric Innovation Software enthält ein Modul zur Erstellung eines virtuellen Prototypen [11]. Diesem virtuellen Prototyp kann auch eine einfache kinematische Funktionalität verliehen werden. Zusätzlich kann die Dynamik in der Mehrkörperdynamiksoftware SIMPACK vorab berechnet werden, und die resultierende Bewegung kann mit Hilfe einer Schnittstelle SIMPACK/Centric in Centric grafisch dargestellt werden.
Die Centric Innovation Software enthält zusätzlich Funktionen zur webbasierten, unternehmensweiten Zusammenarbeit.
Die Centric Innovation Software wurde in der hier vorliegenden Arbeit verwendet. Eine genaue Beschreibung erfolgt in Kapitel 4.

2.3.4 Tecoplan Virtual Workshop

Virtual Workshop war ein von der in München ansässigen Firma Tecoplan AG angebotenes DMU-Programm. Mit Virtual Workshop waren Montage-, Demontage- und Kollisionsuntersuchungen möglich. Über eine Schnittstelle zu ADAMS konnten auch

Ergebnisse dynamischer Simulationen eingelesen und für die Erstellung von Bauraumuntersuchungen verwendet werden [12]. Diese Software wird heute nicht mehr angeboten.

2.4 Beispiel einer Einbindung von Daten einer Mehrkörperdynamik-Simulation in eine DMU-Umgebung

Beim Automobilhersteller Audi wurde bei der Entwicklung des Audi A4 mit Hilfe von DMU auch eine Kollisionsuntersuchung an dynamisch bewegten Bauteilen durchgeführt [13,14]. Dabei handelte es sich um die Freigängigkeitsuntersuchung von Motor- und Getriebebaugruppen. Motor und Getriebe sind elastisch gegenüber der Karosserie gelagert und führen daher, angeregt zum Beispiel durch Lastwechsel oder Fahrbahnunebenheiten, Relativbewegungen gegenüber der Karosserie aus. Audi setzte zu diesem Zweck Geometriehüllen ein, die den Bewegungsraum dieser Bauteile beschreiben. Die Geometriehüllen werden im DMU-System erzeugt, indem Bewegungsdaten und 3D-Geometriedaten miteinander verknüpft werden. Bei der Hüllbildung wird die Geometrie in konkrete Stellungen transformiert und die einzelnen Geometrieelemente linear miteinander verbunden, sodass eine kontinuierliche Bewegungsraumbeschreibung entsteht. Innere Strukturen werden dabei automatisch eliminiert, wodurch die Datenmenge auf eine handhabbare Größe reduziert wird. Eine Bewegungshülle, die den kompletten Bauteilumfang einer Motor-Getriebe-Einheit abbildet, hat eine handhabbare Dateigröße von rund 30 bis 40 MB (Bild 2.1).

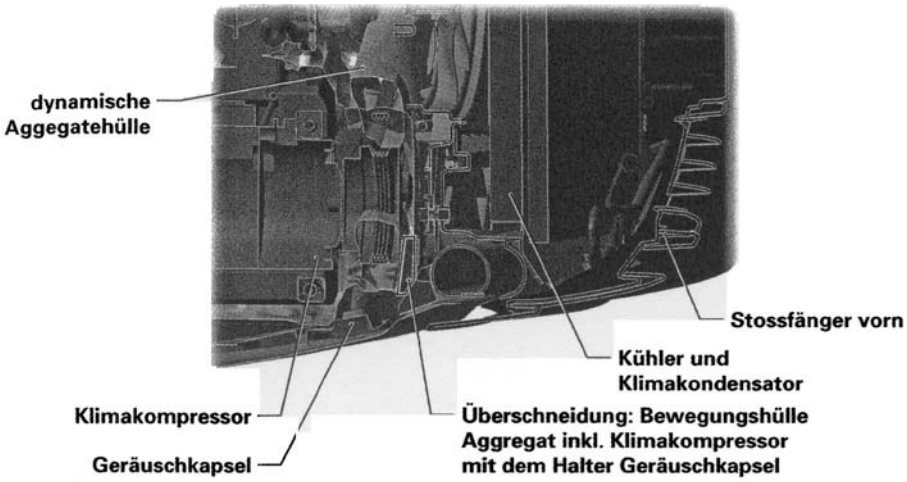

Bild 2.1 Schnitt quer zur Fahrzeug-Längsachse durch den virtuellen Vorbau mit konkretem Kollisionsergebnis beim Audi A4 [14]

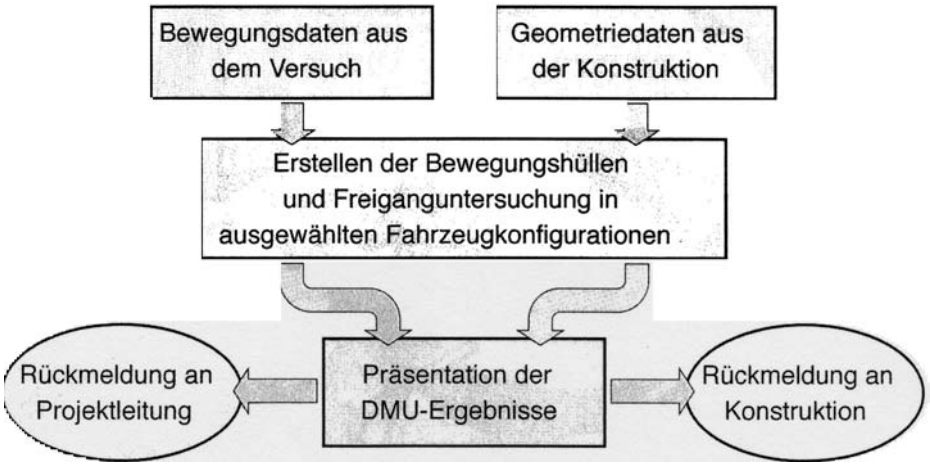

Bild 2.2 Kommunikationsfluss zur Berücksichtigung von Bewegungen im DMU [14]

Die Bewegung der Aggregate wird im Versuch ermittelt. Über ein berührungsfreies Messsystem werden die Transformationsinformationen in eine Datei geschrieben, die über entsprechende Schnittstellen in das DMU-System eingelesen werden kann (Bild 2.2). Zur Berücksichtigung von Einbautoleranzen des Aggregats wird die Bewegungshülle zunächst auf Basis der Bewegungsdaten aus dem Versuch generiert und diese anschließend in einem automatisierten zweiten Schritt innerhalb des Toleranzraums verschoben. Die so entstandene dynamische Bewegungshülle wird mit den statischen Bauteilen auf Freigängigkeit geprüft. Der Aufwand für eine Freigangsuntersuchung des kompletten Aggregats (Motor, Getriebe und Anbauteile) konnte dank DMU von zirka vier bis fünf Manntagen auf einen Manntag verkürzt werden. Zur Ermittlung der Bewegungen ist hier aber immer noch ein physikalischer Prototyp notwendig. Außerdem wird hier nur die Bewegung einer Komponente gegenüber anderen, stationären Komponenten auf Freigängigkeit geprüft. Die Möglichkeit, dass zwei Bauteile den gleichen Raum zu unterschiedlichen Zeitpunkten und daher kollisionsfrei einnehmen, besteht hier nicht.

Auch für Audi ist die Integration von Bewegungsdaten aus dem bei Audi verwendeten Mehrkörperdynamik-Simulationsprogramm ADAMS/car in das DMU-System (bei Audi: Tecoplan „Virtual Workshop“) ein wünschenswertes Ziel. Dieses Ziel wurde durch die Integration einer Schnittstelle zum Einlesen von ADAMS-Bewegungsdaten ab der Version 3.7 von „Virtual Workshop“ erreicht.

Die Funktion wurde anhand eines Audi A6 quattro überprüft. Dabei wurden ADAMS/car-Modelle von Vorder- und Hinterradaufhängung sowie des Antriebsstrangs inklusive aller elastischen Elemente erstellt. Dann wurden verschiedene Fahrmanöver wie zum Beispiel Lastwechsel unter verschiedenen Randbedingungen simuliert. Die dabei entstehende große Datenmenge ist schwer handhabbar. Audi verwendet daher *Envelopes* (Hüllen) des Bewegungsraums um die Teilebewegung im DMU-System darzustellen.

Tecoplan Virtual Workshop stellt hier mehrere verschiedene *Enveloping*-Funktionen zur Verfügung. Die Funktionen VOX-*Surface* und VOX-*Pack* haben sich hier am meisten bewährt. VOX-*Surface* basiert auf dem Prinzip von Oberflächenkopien, wobei von wenigstens sechs Seiten („von oben“, „von unten“, „von links“, „von rechts“, „von vorne“,

„von hinten“) aus nur die im Vordergrund liegende Oberfläche in einen *Envelope* kopiert wird. Auf diese Art werden innere Oberflächen eliminiert. Die Erfahrung hat gezeigt, dass diese Methode ein hohes Maß an Genauigkeit bietet.

Beim VOX-*Pack*-Verfahren (Bild 2.3) wird der virtuelle Raum in Würfel beliebiger Kantenlänge aufgeteilt. Die Abmessungen der Würfel werden durch die Geometriedaten bestimmt. Innerhalb jedes Würfels wird der Umriss der dreidimensionalen Fläche verwendet wenn die Fläche konvex ist. Konkave Oberflächen werden durch ebene Flächen ersetzt, die die Verbindung zum Nachbarwürfel herstellen. Mit dieser Methode wird die Oberfläche vergröbert, so ähnlich als ob man den Gegenstand in Zellophan einwickeln würde. Hohlkörper mit kleinen Öffnungen können mit dieser Methode gut vereinfacht werden. Ein weiterer Vorteil ist, dass damit immer eine geschlossene Oberfläche erzeugt wird.

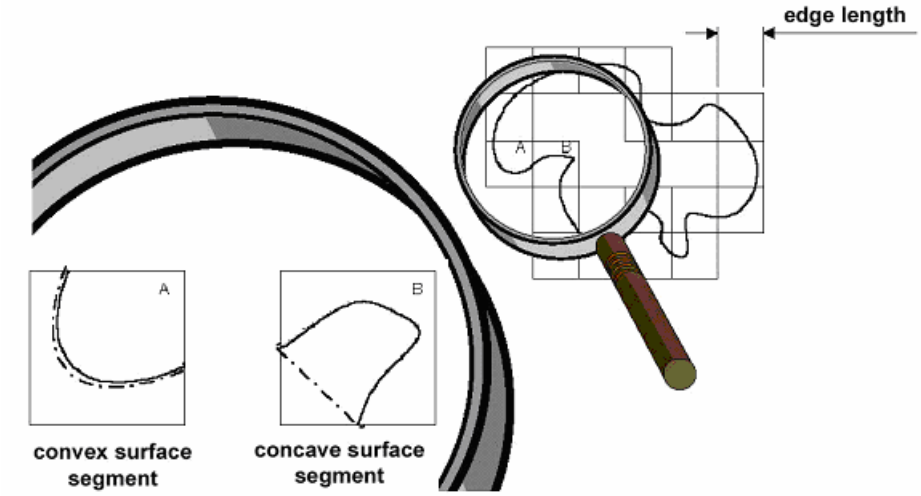

Bild 2.3 VOX-Pack-Verfahren [13]

Audi verwendet die *Envelope*-Funktion um die Datenmenge um 90-95% zu reduzieren. Die beiden folgenden Bilder veranschaulichen (Bild 2.4, Bild 2.5), wie der *Envelope*-Prozess funktioniert.

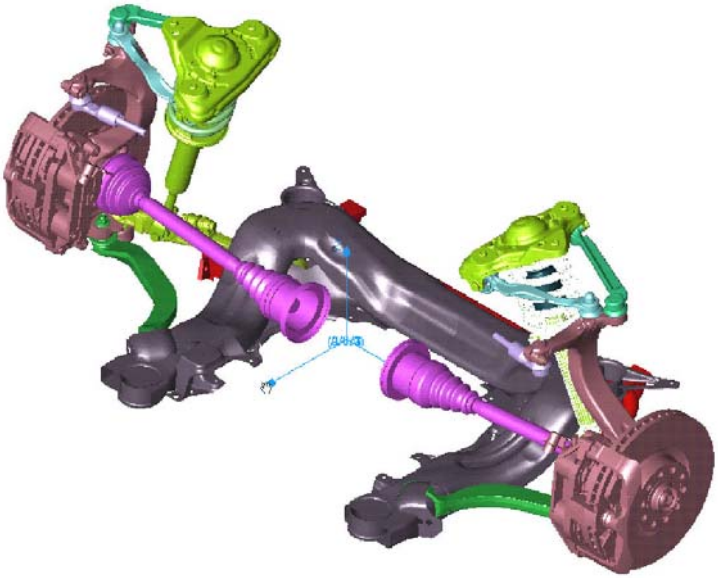

Bild 2.4 Statisches Modell der Vorderradaufhängung [13]

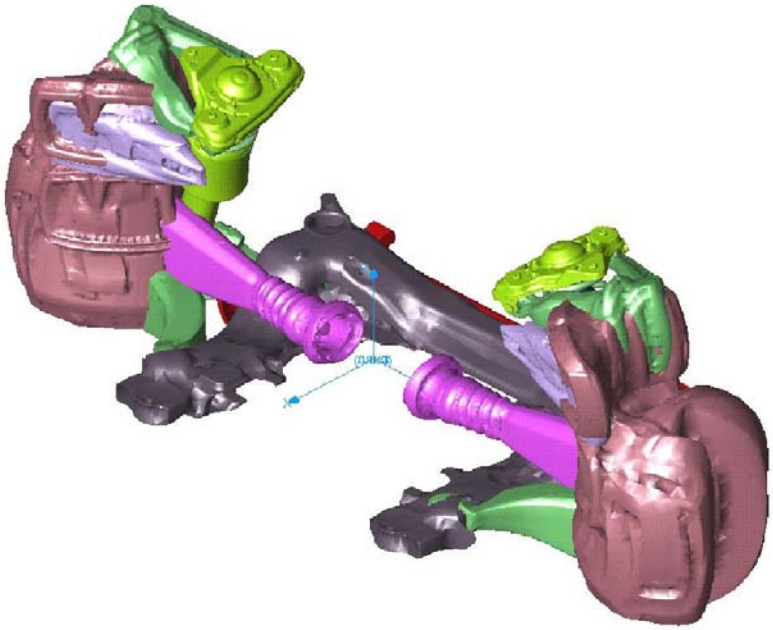

Bild 2.5 Dynamischer Envelope der Vorderradaufhängung (Bewegungsdaten aus ADAMS/car) [13]

Bei Audi wird aktives DMU daher zukünftig in der Entwicklung neuer Modelle eingesetzt werden. Als Aufgabe für die Zukunft wird die Integration flexibler Teile, wie zum Beispiel des Auspuffsystems, in das DMU-System angesehen.

2.5 Grundlagen der Kollisionserkennung

Das Ziel der Kollisionserkennung ist es, einen geometrischen Kontakt von Objekten zu melden [15]. Die geometrischen Modelle können dabei als *Splines*, algebraische Oberflächen oder als Zusammensetzung von Polygonen dargestellt sein. Aus CAD-Daten abgeleitete geometrische Modelle, wie sie in DMU eingesetzt werden, weisen folgende Charakteristiken auf [16]:

- **Modellkomplexität**: Die geometrischen Modelle bestehen aus einer großen Anzahl (10000 – mehrere 100000) von Polygonen
- **Unstrukturierte Darstellung**: Die Modelle werden durch Polygone ohne Topologieinformation dargestellt. Solche Modelle werden als *Polygon Soups* [17] bezeichnet und weisen z. B. Lücken zwischen den Polygonen auf.
- **Geringe Entfernung**: Die Modelle können sich in einem geringen Abstand zueinander befinden. Sie können sich auch an mehreren Stellen überschneiden.

Ein perfekter Kollisionerkennungsalgorithmus sollte folgende Eigenschaften aufweisen [3]:

- Für eine bestimmte gegebene Repräsentationsform der geometrischen Modelle, sollte er die größtmögliche Anzahl von verschiedenen Arten dieser Modelle handhaben können (z. B. deformierbare *Polygon Soups* bei einer Repräsentation durch Polygone)
- Niedrige Rechenzeit (Ziel: 2 x n x 100000 Polygone in $\leq$ 1 Millisekunde)
- Im Fall einer Kollision sollen die schneidenden Polygone und der Ort der Überschneidung ausgegeben werden.
- Bei gegebener aktueller und vorhergehender Position sollte der Algorithmus in der Lage sein, die exakte Zeit und den Ort der Kollision zu bestimmen.
- Der Algorithmus sollte keine großen Hilfsdatenstrukturen benötigen. Insbesondere sollte die Berechnung dieser Strukturen nicht zu lange dauern ($\leq$ 1 Sekunde pro Objekt).

Heute ist kein Algorithmus bekannt, der in der Lage ist, alle diese Anforderungen zu erfüllen.

Als grundlegende Algorithmen kommen für die Kollisionserkennung von *Polygon Soups* zwei Möglichkeiten in Betracht:

- Erkennung der Überschneidung zweier Polygone (*Polygon/Polygon Intersection*)
- Erkennung der Durchdringung der Aussenkante eines Polygons mit einem zweiten Polygon (*Edge/Polygon Intersection*)

2.5.1 *Polygon/Polygon Intersection*

Hier werden die Polygone direkt auf Überschneidungen geprüft. In [18] ist ein solcher Test für die Überprüfung von Dreiecken (Polygone können durch Triangulation durch Dreiecke ersetzt werden) angegeben.

Dabei geht man von folgender Situation aus: Gegeben sind zwei Dreiecke T_1 und T_2 mit den Ecken V_0^1, V_1^1, V_2^1 und V_0^2, V_1^2, V_2^2, die auf Überschneidung überprüft werden sollen. Weiters seien π_1 und π_2 jene Ebenen, in denen die Dreiecke liegen.

Die Gleichung der Ebene π_2:

$$\underline{N_2} \cdot \underline{X} + \underline{d_2} = 0 \tag{2.1}$$

(wobei $\underline{X}$ ein beliebiger Punkt auf der Fläche ist) berechnet sich zu:

$$\begin{aligned} \underline{N_2} &= \left(\underline{V_1}^2 - \underline{V_0}^2\right) \times \left(\underline{V_2}^2 - \underline{V_0}^2\right) \\ \underline{d_2} &= -\underline{N_2} \cdot \underline{V_0}^2 \end{aligned} \tag{2.2}$$

Die Abstände der Ecken von T_1 zur Ebene π_2 (multipliziert mit der Konstante $\left|\underline{N_2}\right|$) berechnen sich dann durch Einsetzen der Ecken in die Gleichung der Ebene π_2:

$$\underline{d_{V_i^1}} = \underline{N_2} \cdot \underline{V_i^1} + \underline{d_2}, \; i = 0, 1, 2 \tag{2.3}$$

Sind nun alle diese Abstände $\underline{d_{V_i^1}} \neq 0$ für $i = 0, 1, 2$ (das bedeutet kein Punkt von T_1 befindet sich auf π_2) und haben alle dasselbe Vorzeichen, dann liegt T_1 auf einer Seite von π_2 und es kann keine Überschneidung stattfinden. Durch diese Überprüfung kann bereits eine große Zahl von Dreiecken von weiteren Überprüfungen ausgeschlossen werden.

Sind alle $\underline{d_{V_i^1}} = 0$ für $i = 0, 1, 2$, dann befinden sich die Dreiecke auf einer gemeinsamen Ebene. Dieser Fall wird später behandelt. Sind die Abstände $\underline{d_{V_i^1}} \neq 0$ und haben sie verschiedene Vorzeichen, dann kann eine Überschneidung zwischen den Dreiecken stattfinden. Um dies zu überprüfen, wird zuerst die Gleichung der Schnittgerade der beiden Ebenen π_1 und π_2 berechnet. Diese berechnet sich zu:

$$\underline{L} = \underline{O} + t \cdot \underline{D} \tag{2.4}$$

wobei $\underline{D} = \underline{N_1} \times \underline{N_2}$ die Richtung der Geraden $\underline{L}$ und $\underline{O}$ ein Punkt auf der Geraden ist. Bedingt durch die vorherigen Berechnungen schneiden beide Dreiecke $\underline{L}$. Diese Überschneidungen bilden Intervalle auf $\underline{L}$. Überlappen sich diese Intervalle, so überschneiden sich auch die Dreiecke (Bild 2.6).

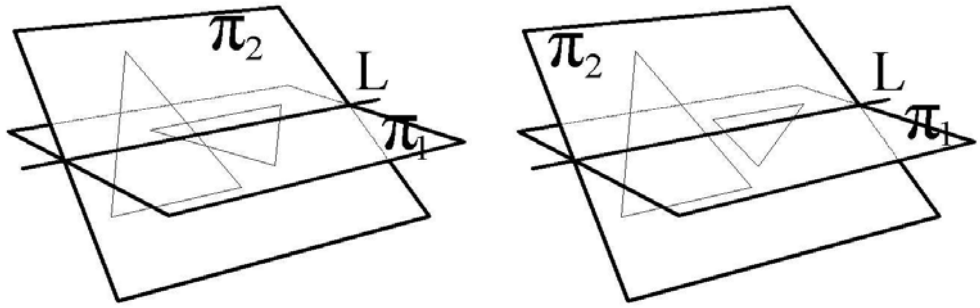

Bild 2.6 Dreiecke und die Ebenen, in denen sie liegen. Links: Die Dreiecke überschneiden sich. Rechts: Die Dreiecke überschneiden sich nicht.

Befinden sich die Dreiecke in einer Ebene, so werden sie auf eine Ebene projiziert, auf der die Flächen der Dreiecke maximiert werden. Danach wird ein zweidimensionaler Dreieck/Dreieck-Überlappungs-Test ausgeführt. Dabei werden alle Kanten von T_1 mit jenen von T_2 auf Überschneidungen überprüft. Wird eine Überschneidung gefunden, überschneiden sich die Dreiecke. Wird keine Überschneidung gefunden, muss noch überprüft werden, ob eines der beiden Dreiecke komplett im anderen enthalten ist. Dies kann durch die Durchführung eines *Point-in-Triangle* Tests (Kapitel 2.5.2.1 und 2.5.2.2) für eine der Ecken von T_1 in T_2 und umgekehrt durchgeführt werden.

2.5.2 *Edge/Polygon Intersection*

Dieser Algorithmus basiert auf der Erkennung, ob eine Linie (Außenkante des einen Polygons) eine Fläche (zweites Polygon) durchdringt. Zur Erkennung, ob eine Linie ein Polygon durchdringt, genügt ein einfacher Zwei-Stufen Test [19]:

- Erstens: Prüfe, ob sich die Endpunkte der Linie an entgegengesetzten Seiten des Polygons befinden (notwendige Bedingung).
- Zweitens: Prüfe, ob der Schnittpunkt der Linie mit der Ebene, in der sich das Polygon befindet, innerhalb des Polygons liegt.

Die Überprüfung, ob die Endpunkte der Linie an entgegengesetzten Seiten liegen, kann z. B. für ein Dreieck folgendermaßen durchgeführt werden [20]:

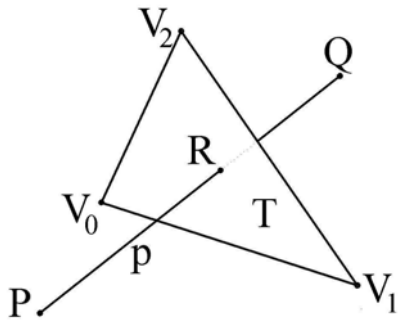

Bild 2.7 Durchdringung Dreieck/Linie

Gegeben sei ein Dreieck T mit den Ecken V_0, V_1, V_2 und eine Strecke p mit den Endpunkten P, Q (Bild 2.7). Zur Überprüfung, ob die Endpunkte P, Q von p auf einer Seite von T liegen, wird zuerst der Normalvektor

$$\vec{n} = \overrightarrow{V_0V_1} \times \overrightarrow{V_0V_2} \tag{2.5}$$

auf die Ebene π, in der T liegt, gebildet. P und Q liegen auf der gleichen Seite von π, wenn die Vorzeichen von $\vec{n} \cdot \overrightarrow{V_0P}$ und $\vec{n} \cdot \overrightarrow{V_0Q}$ gleich sind. Ist $\vec{n} \cdot \overrightarrow{V_0P} = 0$ oder $\vec{n} \cdot \overrightarrow{V_0Q} = 0$, dann liegt jener Punkt, für den das innere Produkt gleich Null ist, auf π. In diesem Fall muss, falls das innere Produkt des anderen Punkts ungleich Null ist, ein weiterer Test durchgeführt werden, um zu überprüfen, ob der auf der Ebene π liegende Punkt innerhalb des Dreiecks liegt. Sind beide innere Produkte gleich Null, dann liegt $\overrightarrow{PQ}$ auf π und es muss ein weiterer Test ausgeführt werden um zu überprüfen, ob die Strecke innerhalb des Dreiecks liegt.

Liegen P und Q auf verschiedenen Seiten von π, dann wird der zweite Teil des Tests durchgeführt.

Für die Lösung des zweiten Problems (*Point-in-Polygon*) gibt es zahlreiche Lösungen, von denen hier zwei der gängigeren Methoden vorgestellt werden.

2.5.2.1 *Ray Intersection Method*

Eine der am häufigsten verwendeten Methoden, um zu bestimmen, ob ein Punkt innerhalb oder außerhalb eines Polygons liegt, ist die *Ray Intersection Method* [21]. Dabei wird ausgehend vom Schnittpunkt ein in der Ebene des betrachteten Polygons liegender Strahl in eine beliebige Richtung konstruiert. Danach wird für jede Begrenzungslinie des Polygons geprüft, ob sie den Strahl schneidet. Ist die Anzahl der Begrenzungslinien, die den Strahl schneiden, gerade (Null eingeschlossen), so liegt der Schnittpunkt außerhalb des Polygons, ist die Anzahl ungerade, so liegt der Schnittpunkt innerhalb des Polygons.

Zusätzliche Prüfungen sind hier für den Fall notwendig, dass der Strahl eine Ecke des Polygons schneidet. Eine andere Möglichkeit ist es, solche Sonderfälle durch eine geeignete Wahl des Strahls zu vermeiden.

2.5.2.2 *Point-in-Polygon* Test mittels baryzentrischer Koordinaten

Ein sehr schneller Algorithmus zur Bestimmung, ob der Schnittpunkt innerhalb des Polygons liegt, wird in [22] vorgestellt. Dieser Test basiert auf der Nutzung von baryzentrischen Koordinaten α und β:

$$\overrightarrow{V_0R} = \alpha \cdot \overrightarrow{V_0V_1} + \beta \cdot \overrightarrow{V_0V_1} \tag{2.6}$$

für ein Dreieck mit den Eckpunkten V_0, V_1, V_2 und dem zu überprüfenden Punkt R (Bild 2.8).

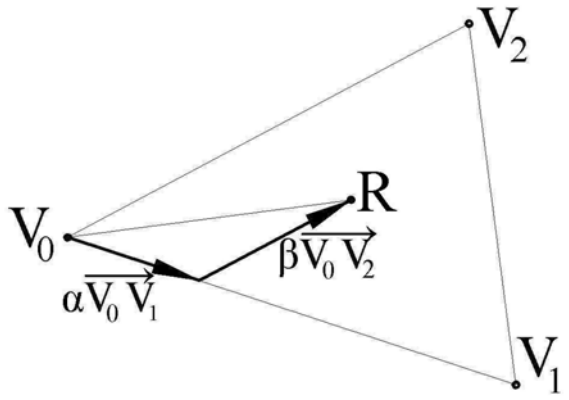

Bild 2.8 Dreieck mit baryzentrischen Koordinaten

Nach der Berechnung des Schnittpunkts werden die baryzentrischen Koordinaten des Schnittpunkts berechnet (hier für den zweidimensionalen Fall):

$$\alpha = \frac{\det\begin{pmatrix} R_x - V_{0x} & V_{2x} - V_{0x} \\ R_y - V_{0y} & V_{2y} - V_{0y} \end{pmatrix}}{\det\begin{pmatrix} V_{1x} - V_{0x} & V_{2x} - V_{0x} \\ V_{1y} - V_{0y} & V_{2y} - V_{0y} \end{pmatrix}} \tag{2.7}$$

$$\beta = \frac{\det\begin{pmatrix} V_{1x} - V_{0x} & R_x - V_{0x} \\ V_{1y} - V_{0y} & R_y - V_{0y} \end{pmatrix}}{\det\begin{pmatrix} V_{1x} - V_{0x} & V_{2x} - V_{0x} \\ V_{1y} - V_{0y} & V_{2y} - V_{0y} \end{pmatrix}} \tag{2.8}$$

Sind $\alpha \geq 0$ und $\beta \geq 0$ (2.9)

und $\alpha + \beta \leq 1$ (2.10)

dann befindet sich der Schnittpunkt innerhalb des Dreiecks.

Zusätzlich zur höheren Geschwindigkeit gegenüber früheren Algorithmen bietet diese Methode noch einen weiteren Vorteil. Befindet sich der Punkt nämlich außerhalb des Dreiecks, so enthalten die baryzentrischen Koordinaten Information über die Position des Schnittpunkts relativ zum getesteten Dreieck. Die Kollision findet dann mit hoher Wahrscheinlichkeit bei einem benachbarten Dreieck statt, das sich in dieser Region befindet. Dieses Dreieck kann dann sofort überprüft werden.

2.5.3 Hierarchische Methoden zur Kollisionserkennung

Der einfachste Algorithmus zur Kollisionserkennung wäre: Überprüfe alle Ecken des ersten Polyeders P auf Durchdringungen mit allen Polygonen des zweiten Polyeders Q und umgekehrt [3]. Alternativ können auch alle Polygone gegeneinander geprüft werden.

Dieser sehr einfach aufgebaute Algorithmus würde aber bei komplexeren Simulationen mit zahlreichen sich bewegenden Objekten, die wiederum aus einer sehr großen Anzahl von Polygonen bestehen, zu sehr langen Rechenzeiten führen. Daher wurden mehrere Verfahren entwickelt, um die Anzahl der durchzuführenden *Polygon/Polygon*- oder *Edge/Polygon*-Tests zu reduzieren.

Das Ziel dieser Verfahren ist es, möglichst viele Polygonpaare, die nicht miteinander kollidieren können, zu erkennen und vom nachfolgenden *Polygon/Polygon*-Test auszunehmen [3]. Dies geschieht mit *bounding volume trees*. Jeder Knoten in diesem Baum enthält mehrere Polygone eines Objekts zusammen mit einem *bounding volume* (BV), das diese Polygone umschließt. Die Reduktion der notwendigen *Polygon/Polygon*-Tests funktioniert folgendermaßen: Nimmt man zwei sich bewegende Objekte und die beiden mit ihnen verknüpften *BV trees* als gegeben, so werden alle BV der *BV trees* auf Überlappungen geprüft. Die Polygone, die in jenen BVs enthalten sind, die nicht überlappen, werden von der weiteren Kollisionsprüfung ausgeschlossen.

Einige Grundbegriffe, die später verwendet werden, sind:

- **Zeitliche und geometrische Kohärenz**. Unter zeitlicher Kohärenz versteht man die Eigenschaft, dass sich der Zustand der Anwendung zwischen zwei Zeitschritten nicht signifikant ändert [23]. Die Objekte bewegen sich nur wenig zwischen zwei Zeitschritten. Daraus folgt die geometrische Kohärenz, da sich die durch die Koordinaten ihrer Eckpunkte definierte Geometrie der Objekte zwischen zwei Schritten nur minimal verändert. Die zugrunde liegende Annahme ist, dass die Zeitschritte so klein sind, dass die Objekte zwischen zwei Zeitschritten keine großen Distanzen zurücklegen.

- ***Axis Aligned Bounding Box* (AABB)**. Diese *Bounding Boxes* sind entsprechend den globalen Koordinaten ausgerichtet – ihre Kanten sind parallel zu den Achsen dieses globalen Koordinatensystems.

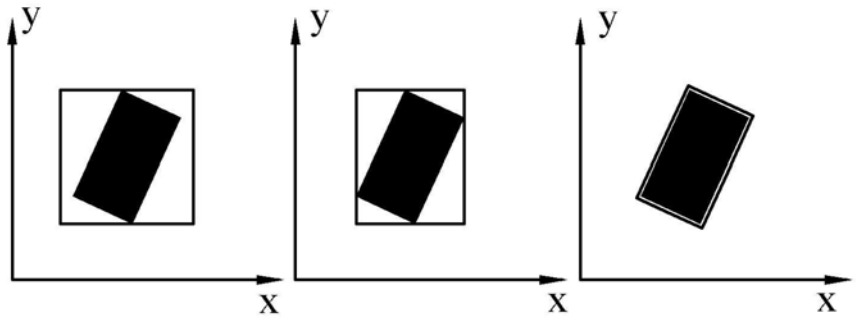

Bild 2.9 Bounding Box Arten: Links: Fixed-Size Bounding Cube. Mitte: Dynamically Rectangular Bounding Box. Rechts: Oriented Bounding Box.

- ***Fixed-Size Bounding Cubes*** [23]. Dies sind AABBs, deren Größe so gewählt wird, dass das umschlossene Objekt bei jeder Orientierung enthalten ist (Bild 2.9 Links). Dieser *Bounding Cube* wird durch ein Zentrum und die halbe Kantenlänge definiert. *Fixed Cubes* können einfach neu berechnet werden, wenn sich die Objekte bewegen.
  In einem *Preprocessing* Schritt wird Zentrum und halbe Kantenlänge berechnet. Für jeden Zeitschritt muss, wenn sich das umschlossene Objekt bewegt, der *Cube* folgendermaßen berechnet werden:
  - Verschiebung des Zentrums
  - Berechnung der minimalen x-, y- und z-Koordinaten durch Subtraktion und Addition der halben Kantenlänge vom Zentrum
- ***Dynamically Rectangular Bounding Boxes*** [23]. Dies sind AABBs, deren Abmessungen so gewählt sind, dass das umschlossene Objekt nur mit der gerade aktuellen Orientierung enthalten ist (Bild 2.9 Mitte). Sie sind definiert durch maximale und minimale x-, y- und z-Koordinaten. Bewegt sich das umschlossene Objekt, so müssen diese Koordinaten entsprechend der neuen Orientierung des Objekts neuerlich berechnet werden. Der Vorteil dieses *Bounding Volumes* ist, dass es die Objekte enger umschließt. Der Nachteil besteht darin, dass die maximalen und minimalen x-, y- und z-Koordinaten für jeden Zeitschritt nicht einfach durch Addition und Subtraktion wie bei *Fixed-Size Bounding Cubes* berechnet werden können.
- ***Oriented Bounding Boxes*** [3]. Dies sind *Bounding Boxes*, deren Orientierung nicht durch ein festes Koordinatensystem vorgegeben ist (Bild 2.9 Rechts). Die Orientierung wird für jedes umschlossene Objekt anders und zwar so gewählt, dass das Volumen der *Bounding Box* möglichst klein ist. Dadurch kann ein Objekt enger umschlossen werden als mit AABBs, die Berechnung dieser *Bounding Boxes* dauert aber länger.

Drei hierarchische Kollisionsprüfungsverfahren, die an der University of North Carolina entwickelt wurden, werden im Folgenden vorgestellt. Diese Lösungen stehen auch als API (**A**pplication **P**rogram **I**nterface, Anwendungsprogrammschnittstelle) zur Verfügung.

2.5.3.1 I-COLLIDE

I-COLLIDE ist ein Kollisionserkennungsalgorithmus für die exakte Kollisionserkennung in komplexen Umgebungen. Mit I-COLLIDE kann die Zahl von $O(n^2)$ möglichen Interaktionen von n sich bewegenden Objekten auf $O(n+m)$ reduziert werden, wobei m die Anzahl jener Objekte ist, die sich in großer Nähe zueinander befindet [23].

Dieser Algorithmus berechnet zuerst AABBs vom Typ *Dynamically Rectangular Bounding Box*. Danach werden die dreidimensionalen *Bounding Boxes* auf die x-, y- und z-Achse projiziert. Da die *Bounding Boxes* achsausgerichtet sind, entstehen durch die Projektion auf die Achsen Intervalle (Bild 2.10).

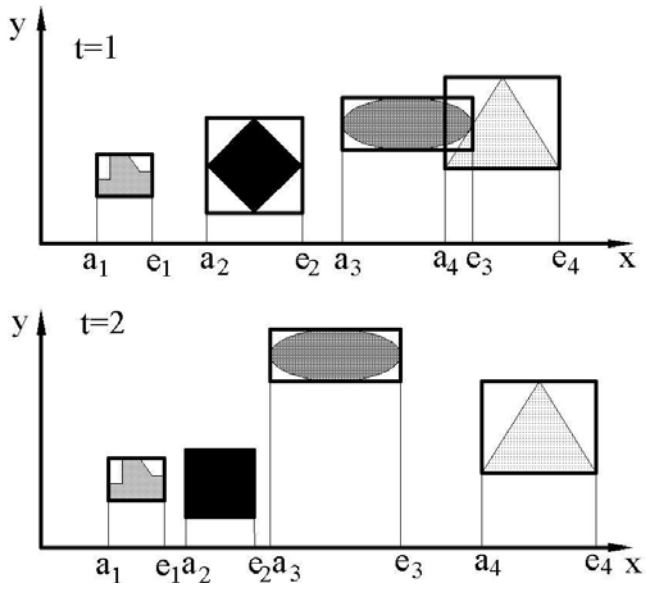

Bild 2.10 Projektion der Bounding Boxes auf die x-Achse mit Intervallen $a_i - e_i$

Das Ziel ist es, die Überlappung von Intervallen herauszufinden, da sich zwei *Bounding Boxes* dann und nur dann überlappen können, wenn sich die Intervalle in allen drei Dimensionen überlappen. Zu diesem Zweck werden drei Listen konstruiert, eine für jede Achsrichtung. Jede dieser Listen enthält die Anfangs- und Endpunkte der Intervalle. Durch Ordnen dieser Listen kann ermittelt werden, welche Intervalle sich überlappen. Im allgemeinen Fall würde dies Zeit in der Größenordnung von $O(n \cdot \log n)$ in Anspruch nehmen, wobei n die Anzahl der Objekte ist. Diese Zeit kann dadurch reduziert werden, dass zum Aufbau der Liste die Liste des vorherigen Zeitschritts herangezogen wird und nur

die Zahlenwerte der Intervallanfangs- und Intervallendpunkte aktualisiert werden. In kohärenten Umgebungen, wo die Objekte nur kleine Bewegungen zwischen zwei Schritten ausführen, ist diese Liste bereits beinahe geordnet, so dass das Herstellen der fertig geordneten Liste nur eine Zeit von $O(n)$ in Anspruch nimmt.

Zusätzlich müssen Änderungen im Überlappungsstatus von Intervallen (z. B. von Überlappung im vorherigen Zeitschritt zu Nicht-Überlappung im aktuellen Schritt oder umgekehrt) ermittelt werden. Dafür wird Rechenzeit der Größenordnung $O(n + e_x + e_y + e_z)$ in Anspruch genommen, wobei e_x, e_y und e_z die Anzahl der Änderungen entlang der x-, y- und z-Achse sind.

Erst wenn durch diese vorangegangene Prüfung Paare von überlappenden *Bounding Boxes* gefunden werden, wird für die Geometrien in diesen *Bounding Boxes* eine exakte Kollisionsberechnung durchgeführt.

2.5.3.2 RAPID

RAPID (**RAP**id **I**nterference **D**etection) ist ein Algorithmus zur exakten Kollisionsberechnung für Anwendungen mit folgenden Charakteristika [15]:

- Große Modellkomplexität
- Unstrukturierte Darstellung
- Große Nähe
- Exakte Kollisionserkennung

Das Ziel war es einen Algorithmus auf der Basis von *Oriented Bounding Boxes* (OBB) zu entwickeln, um die Berechnung der Kollisionserkennung zu beschleunigen. Die Anforderungen an das *Bounding Volume* lauten folgendermaßen:

- Es sollte das originale Modell möglichst eng umschließen.
- Das Testen zweier dieser *Bounding Volumes* auf Überlappung sollte schnell erfolgen können.

Der Algorithmus berechnet zuerst einen OBB-*Tree* (Siehe [16]). Die Berechnung der Überprüfung auf Überlappungen wird folgendermaßen durchgeführt [16]:

Ein trivialer Test auf Überlappung ist es, die *Bounding Volumes* auf eine Achse zu projizieren. Durch diese Projektion bilden die *Bounding Volumes* Intervalle auf der Achse. Überlappen sich die Intervalle nicht, so überlappen sich auch die *Bounding Volumes* nicht, und die Achse wird als Trennungsachse bezeichnet. Überlappen sich die Intervalle, so können sich die *Bounding Volumes* überlappen oder auch nicht überlappen.

Um die Anzahl der notwendigen Tests einzugrenzen, benutzt man folgende Erkenntnis: Zwei nicht überlappende Polytope (=endlich große Vielecke) im dreidimensionalen Raum

können immer durch eine Fläche getrennt werden, die parallel zu einer Oberfläche eines Polytopes oder parallel zu je einer Kante je eines Polytopes ist. Als Konsequenz daraus folgt, dass zwei Polytope sich dann nicht überlappen, wenn eine Trennungsachse existiert, die entweder normal auf eine Oberfläche eines Polytopes oder normal auf je eine Kante je eines Polytopes steht. Jede *Bounding Box* hat je drei Oberflächenorientierungen und je drei Kantenrichtungen. Dies führt zu 15 potentiellen Trennungsachsen (drei Oberflächenorientierungen von jeder *Box* und neun paarweise Kombinationen der Kanten), die überprüft werden müssen. Überlappen sich die *Bounding Boxen*, so existiert keine Trennungsachse. Die Prüfung dieser 15 potentiellen Trennungsachsen ist daher hinreichend zur Bestimmung des Überlappungsstatus zweier *Bounding Boxen*.

Um die Überprüfung auf Überlappung durchzuführen, werden die Zentren der *Bounding Boxen* auf die potentielle Trennungsachse projiziert. Weiters wird der projizierte Abstand vom Zentrum zur äußeren Ecke der *Box* berechnet (Bild 2.11). Ist der projizierte Abstand der beiden Mittelpunkte größer als die Summe der beiden projizierten Abstände zur äußeren Ecke, so überlappen sich die *Bounding Boxen* nicht.

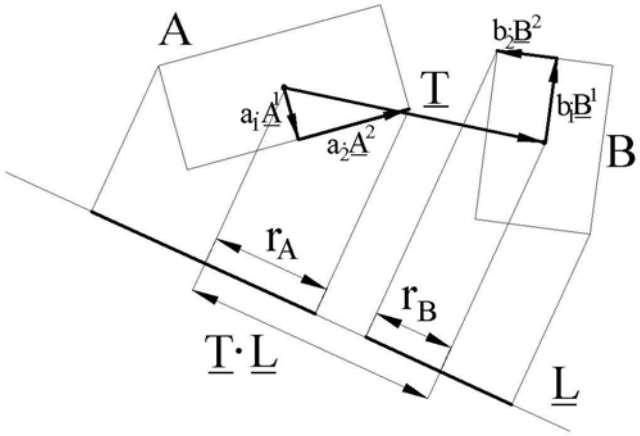

Bild 2.11 OBBs A und B mit Trennungsachse $\underline{L}$. $\underline{L}$ ist eine Trennungsachse, weil sich A und B bei der Projektion auf $\underline{L}$ nicht überlappen

Sind zwei OBBs A und B gegeben und die Position von B relativ zu A beschrieben durch die Rotation $\underline{R}$ und die Translation $\underline{T}$ so wird die Überlappung auf folgende Art überprüft: Die halben Kantenlängen von A und B sind a_i und b_i mit $i = 1,2,3$. Durch Projektion der halben Kantenlängen auf die potentielle Trennungsachse und die Summation ihrer Abbildungen erhält man die halbe Länge des Abbildungsintervalls. Die halbe Länge des Intervalls der Abbildung von A (r_A) ergibt sich damit zu

$$r_A = \sum_i \left| a_i \cdot \underline{A^i} \cdot \underline{L} \right| \tag{2.11}$$

Für r_B ergibt sich ein entsprechender Ausdruck.
Der Abstand zwischen den Mittelpunkten der Intervalle ist $\left|\underline{T} \cdot \underline{L}\right|$. Die Intervalle überlappen sich daher nicht wenn

$$\left|\underline{T} \cdot \underline{L}\right| > \sum_i \left|a_i \cdot \underline{A^i} \cdot \underline{L}\right| + \sum_i \left|b_i \cdot \underline{B^i} \cdot \underline{L}\right| \tag{2.12}$$

erfüllt ist. Durch einige mathematische Vereinfachungen, die in [16] erläutert werden, kann diese Prüfung so vereinfacht werden, dass ein Test für 2 OBBs maximal 200 Rechenoperationen beansprucht.

Erst wenn zwei *Bounding Boxen* sich überlappen, muss wieder wie bei 2.5.3.1 eine exakte Kollisionsprüfung für die involvierten Polygone durchgeführt werden.

2.5.3.3 V-COLLIDE

V-COLLIDE ist ein Algorithmus zur Kollisionserkennung von aus Polygonen zusammengesetzten Modellen, die Starrkörperbewegungen durchführen [24].

V-COLLIDE vereinigt den I-COLLIDE und den RAPID Algorithmus. Zuerst wird nach dem in 2.5.3.1 erläuterten Verfahren die Überlappung von AABBs berechnet. Für jene Paare, bei denen eine Überlappung auftritt wird dann die OBB Hierarchie nach 2.5.3.2 auf Überlappung geprüft. Nur für jene Polygone, die in den OBBs enthalten sind, die sich überlappen, wird dann eine exakte Kollisionsprüfung durchgeführt.

Für den Fall einer Kollision werden von V-COLLIDE nur die involvierten Objekte gemeldet. Auf die prinzipiell vorhandene Möglichkeit genauere Informationen zu liefern wurde bei der Implementierung des Algorithmus verzichtet, um den Speicherbedarf der verwendeten Datenstrukturen zu verringern [24].

Kapitel 3. Die Mehrkörperdynamiksoftware SIMPACK

Die SIMPACK (**SI**mulation of **M**ulti-body systems **PACK**age) Software ist ein von der Firma INTEC angebotenes Programmpaket zur Simulation dynamischer Vorgänge in Mehrkörpersystemen.

Das Grundkonzept von SIMPACK besteht darin, die Bewegungsgleichungen eines mechanischen Systems aufzustellen und zu lösen. Dieses mechanische System wird vom Anwender durch von SIMPACK bereitgestellte Modellierungselemente definiert [25]. Im Folgenden wird ein Überblick über die Grundfunktionalitäten von SIMPACK, speziell im Hinblick auf die für die vorliegende Diplomarbeit verwendeten Funktionen, geboten.

Die Bedienung des Programms erfolgt über eine grafische Benutzeroberfläche, die aus mehreren Fenstern besteht. Bei Programmstart erscheinen zuerst zwei Fenster:

- *SIMPACK-Echo-Area*: Hier werden Meldungen des Programms, wie zum Beispiel Informationen zur Berechnung oder Fehlermeldungen dargestellt.

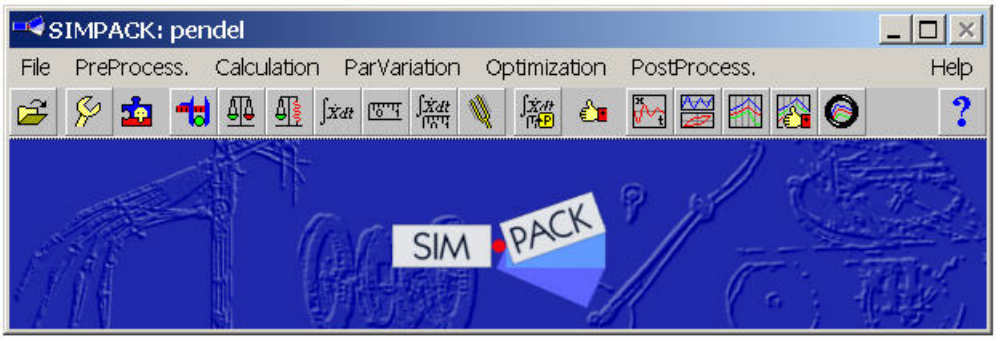

Bild 3.1 SIMPACK User Interface

- *SIMPACK User Interface (Bild 3.1)*: Das *SIMPACK User Interface* ist das Hauptfenster von SIMPACK. Es beinhaltet in einer Menüleiste die Module
  - *File*
  - *Pre-Processing*
  - *Calculation*
  - *Parameter Variation*
  - *Optimization*
  - *Post-Processing*
  - *Help*

  Jedes dieser Module besteht aus mehreren Unterpunkten, die in *Pulldown*-Menüs ausgewählt werden können.

3.1 Das *File*-Modul

In diesem Modul erfolgt die SIMPACK-Modellverwaltung. Hier können neue Modelle erzeugt oder vorhandene Modelle gelöscht, umbenannt oder geöffnet werden.

3.2 Das *Pre-Processing* Modul

Die Aufgabe des *Pre-Processing* Moduls ist es, dem Benutzer den Aufbau eines Modells mit den zugehörigen 3D-Geometrien zu ermöglichen. Dies geschieht in den folgenden Unterpunkten:

3.2.1 *MBS-Setup*

In diesem Fenster erfolgt der Modellaufbau. Hier werden die Daten von Körpern, Gelenken etc. eingegeben.
Bei der Neuerstellung eines Modells werden von SIMPACK die folgenden Objekte automatisch erstellt:

- Ein inertialfestes Bezugssystem (*$B_Isys*) auf das sämtliche folgende Bewegungen bezogen werden
- Ein *Marker* (*$M_Isys*) im Ursprung dieses Bezugssystems
- Ein Körper (*$B_Body1*) mit einem *Marker* (*$M_Body1*) auf dem körperfesten Referenzsystem
- Ein Gelenk (*$J_Body1*) mit 0 Freiheitsgraden, das den Körper mit dem Bezugssystem verbindet
- Ein *Sensor* (*$S_Body1*) zwischen den *Markern $M_Isys* und *$M_Body1*
- Ein Gravitationsvektor
- Eine *Default*-Geometrie für das Bezugssystem und den Körper

3.2.1.1 Bezugssystem

Um die Bewegungsgleichungen formulieren zu können, ist ein Bezugssystem erforderlich. SIMPACK bietet zwei Arten von Bezugssystemen:

- Ein inertialfestes Bezugssystem
- Ein bewegtes Bezugssystem. Die Bewegung dieses Koordinatensystems zum Inertialsystem muss kinematisch vorgegeben sein.

Auf dem Bezugssystem können zusätzliche *Marker* definiert werden.

3.2.1.2 Körper (*Bodies*)

Körper sind die Grundbestandteile eines mechanischen Modells. Sie sind die einzigen Teile in SIMPACK, die Masse aufweisen. In SIMPACK können zwei verschiedene Arten von Körpern erzeugt werden:

- Starrkörper
- Elastische Körper

Für die vorliegende Arbeit wurden nur Starrkörper verwendet, weshalb im Folgenden auch nur auf Starrkörper eingegangen wird.

Wird in SIMPACK ein neuer Körper erzeugt, so werden folgende Elemente automatisch generiert:

- Ein körperfestes Referenzsystem
- Ein *Marker* im Ursprung des körperfesten Referenzsystems mit der gleichen Orientierung wie das Referenzsystem
- Ein 0-Freiheitsgrad-Gelenk zwischen dem Bezugssystem und dem Körper
- Ein *Sensor* zwischen dem Inertialsystem und dem *Marker* am Körper
- Ein *3D-Ensemble* und ein *3D-Primitive* vom Typ Koordinatensystem zur grafischen Darstellung des Körpers. Der *3D-Primitive* stellt das körperfeste Referenzsystem dar.

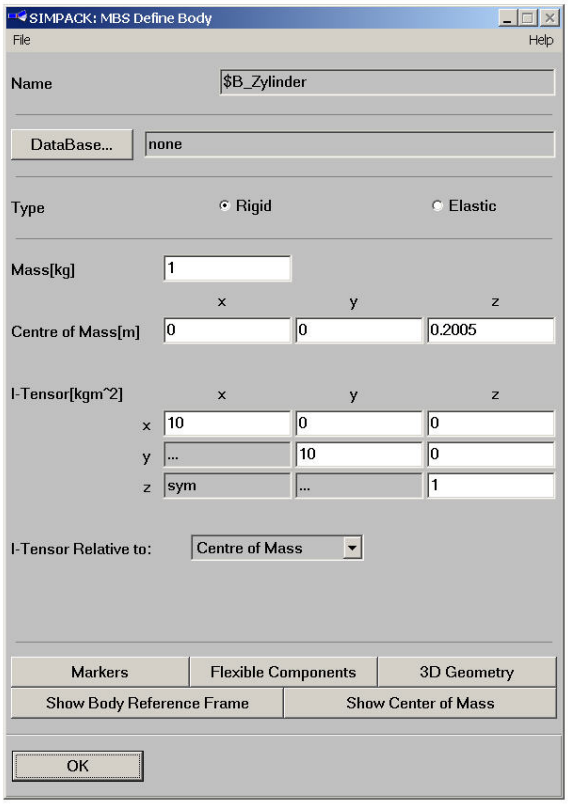

Bild 3.2 Eingabefenster für die Eigenschaften des Körpers

Die Eigenschaften dieses Körpers sind standardmäßig mit einer Masse von 1kg, den Schwerpunktskoordinaten x=0, y=0 und z=0 bezogen auf das körperfeste Referenzsystem und Ixx=Iyy=Izz=1kgm² festgelegt, wobei sich der Trägheitstensor auf das körperfeste Referenzsystem bezieht. Diese Eigenschaften müssen vom Benutzer entsprechend den Eigenschaften des zu modellierenden Systems geändert werden (Bild 3.2).

Um einen Starrkörper zu definieren, werden folgende Daten benötigt:

- Name des Körpers
- Masse des Körpers
- Schwerpunktlage relativ zum körperfesten Bezugssystem
- Trägheitstensor des Körpers bezogen auf das körperfeste Referenzsystem, bezogen auf ein Bezugssystem im Schwerpunkt mit der Orientierung des Referenzsystems oder bezogen auf einen benutzerdefinierten körperfesten *Marker*
- Art des Körpers für die grafische Repräsentation und seine Abmaße
- Falls notwendig zusätzliche *Marker*

Die 3D-Geometrie eines Körpers besteht aus einem oder mehreren *Ensembles*. Jedes *Ensemble* besteht aus einem oder mehreren *Primitives* wie z. B. Quader, Zylinder, Kugeln etc. Zusätzlich können auch komplexere Geometrien, die in CAD-Systemen erstellt wurden, in ein *Ensemble* geladen werden.

Alle Körper werden mit *$B_Name des Körpers* bezeichnet.

3.2.1.3 *Marker*

Marker können auf allen Körpern und Referenzrahmen definiert werden. Ihre Aufgaben sind:

- Definition des Angriffspunkts eines Kraftelements an einem Körper oder Referenzrahmen
- Definition des Anknüpfungspunkts eines Gelenks oder einer *Constraint*
- Grundlage für *Sensor* Messungen, die die Relativbewegung zwischen zwei *Markern* messen

Es gibt zwei Arten von *Markern*:

- Körperfeste *Marker*
- Bewegte *Marker*

Die Definition körperfester Marker erfolgt über die Eingabe der Koordinaten des Markers im körperfesten Referenzsystem. Die Orientierung des Markers ist standardmäßig gleich der Orientierung des körperfesten Referenzsystems. Für die Eingabe einer davon abweichenden Orientierung stehen dem Benutzer drei Möglichkeiten zur Verfügung:

- Eingabe von drei Kardanwinkeln relativ zum körperfesten Referenzsystem
- Eingabe einer Transformationsmatrix, die die Orientierung relativ zum körperfesten Referenzsystem angibt.
- Eingabe von drei Punkten P, Q, R, wobei die Linie PQ die Richtung einer vom Benutzer ausgewählten Achse angibt und R einen Punkt in der Ebene angibt, in dem eine zweite vom Benutzer ausgewählte Achse liegt.

Die Bezeichnung eines *Markers* erfolgt mit *$M_Name des Markers*.

Ein körperfester *Marker* ist damit vollständig definiert. Bewegte *Marker* wurden nicht verwendet, weshalb ich hier auch nicht näher darauf eingehe.

3.2.1.4 Gelenke (*Joints*)

Mit den Körpern werden automatisch masselose Gelenke erzeugt. Mit Gelenken werden Körper mit dem Inertialsystem oder mit einem anderen Körper verbunden. Zu jedem Körper existiert genau ein Gelenk. Art und Befestigungspunkte des Gelenks werden vom Benutzer festgelegt. Als Bezugspunkte, zwischen denen das Gelenk eingefügt wird, werden zwei *Marker* gewählt.

Zur vollständigen Definition eines Gelenks sind folgende Daten notwendig:

- Name des Gelenks
- *Marker* zwischen denen das Gelenk wirkt
- Art des Gelenks (entweder aus SIMPACK-Bibliothek oder benutzerdefiniert)
- Anfangsbedingungen des Gelenks

Die Bezeichnung von Gelenken erfolgt mit *$J_Name des Gelenks*.

3.2.1.5 Zwangsbedingungen (*Constraints*)

Führt in einem mechanischen System mehr als ein Pfad von einem Körper zum nächsten, so spricht man von kinematisch geschlossenen Schleifen. Zur Definition dieser geschlossenen Schleifen werden in SIMPACK *Constraints* verwendet. Die *Constraints* werden dabei so gewählt, dass folgende Gleichung erfüllt ist:

$$\sum DOF_{SYSTEM} = \sum DOF_{JOINTS} - \sum Constr. \qquad (3.1)$$

Dabei steht DOF_{SYSTEM} für die Anzahl der Freiheitsgrade des gesamten Systems, DOF_{JOINTS} für die Bewegungsmöglichkeiten der Gelenke und $Constr.$ für die durch *Constraints* eingeschränkten Bewegungsmöglichkeiten.

Dieses Konzept soll anhand eines einfachen Beispiels (ebener Kurbeltrieb – ein Freiheitsgrad) erläutert werden (Bild 3.3):

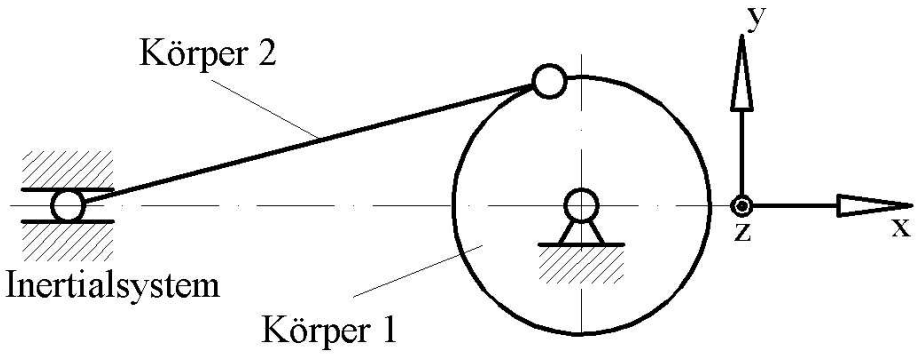

Bild 3.3 Ebener Kurbeltrieb

Dieses Modell wird in SIMPACK mit folgender Struktur dargestellt (Bild 3.4). Dabei werden bei einem Gelenk die Bewegungsmöglichkeiten zwischen den Körpern, die durch das Gelenk verbunden werden, angegeben. Bei einer *Constraint* zwischen zwei Körpern wird die Bewegungsmöglichkeit zwischen diesen Körpern angegeben, die durch die *Constraint* gesperrt wird.

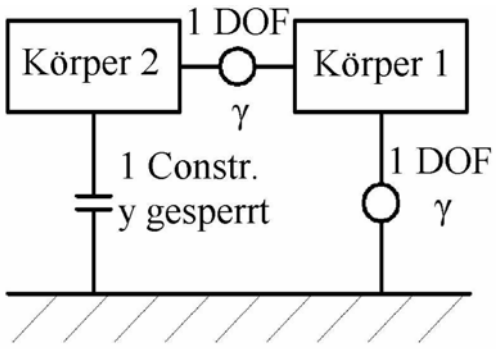

Bild 3.4 Struktur des SIMPACK Modells des ebenen Kurbeltriebs
Gelenk mit Angabe der Freiheitsgrade, Constraint mit Angabe der gesperrten Bewegungen (α, β, γ: Rotation um x-, y- und z-Achse), Symbolik nach SIMPACK Manual [25]

Nach Gleichung (3.1) ergibt sich dann $DOF_{SYSTEM} = 2 \cdot 1 - 1 = 1$. Der ebene Kurbeltrieb wurde damit korrekt modelliert.
Zur Definition einer *Constraint* sind folgende Eingaben erforderlich:

- Name der *Constraint*
- *Marker*, zwischen denen die Schleife geschlossen werden soll

- Art der *Constraint* (aus SIMPACK-Bibliothek oder benutzerdefiniert)
- *Constraint* Parameter
- Abhängige und unabhängige Variable in der geschlossenen Schleife

Die Bezeichnung einer *Constraint* erfolgt mit *$L_Name der Constraint*.

3.2.1.6 Kraftelemente (*Force Elements*)

Kraftelemente sind masselose Verbindungen zwischen zwei Körpern oder zwischen einem Körper und dem Inertialsystem. Sie üben Kräfte oder Momente oder beides auf die Körper, an denen sie angebracht sind, aus.
Grundsätzlich kann man zwei Arten von Kraftelementen unterscheiden:

- *Point-to-Point Elements*, die entlang der Verbindungslinie der beiden *Marker* wirken. Alle Ein- und Ausgaben dieser Elemente werden entlang dieser Richtung bestimmt. Die Kräfte und/oder Momente wirken im Ursprung dieser *Marker* mit entgegengesetztem Vorzeichen. *Point-to-Point Elements* sind immer symmetrisch, die Orientierung der Anknüpfungs-*Marker* beeinflusst ihr Verhalten nicht.
- *Component Elements* wirken entlang der Achse jenes *Markers*, der als *From-Marker* definiert wurde. Die Kräfte und/oder Momente wirken auf beide Körper an der Position des *To-Markers* mit unterschiedlichem Vorzeichen. *Component Elements* sind nicht symmetrisch, weil nur die Orientierung des *From-Markers*, nicht aber die Orientierung des *To-Markers* einen Einfluss auf ihr Verhalten hat.

Zur Definition eines Kraftelements sind erforderlich:

- Name des Kraftelements
- *Marker* (*From-Marker* und *To-Marker*) zwischen denen das Kraftelement wirkt
- Art des Kraftelements (aus SIMPACK-Bibliothek, zum Beispiel Feder-Dämpfer parallel oder benutzerdefiniert)
- Parameter wie z. B. Steifigkeiten, Dämpferrate, etc.
- Anfangsbedingungen

Kraftelemente können auch grafisch dargestellt werden. Von SIMPACK werden dafür skalierte *Primitive*s wie z. B. *Scaled Spring PtP* bereitgestellt (siehe auch 3.2.1.9).

Die Bezeichnung von Kraftelementen erfolgt mit *$F_Name* des Kraftelements.

3.2.1.7 *Substructures*

Substructures ermöglichen es, mehrmals verwendete Baugruppen (z. B. ein Drehgestell) nur einmal zu erstellen und dann in das Hauptmodell zu laden. Die Struktur einer *Substructure* wird als SIMPACK Modell definiert und als *Substructure* in einem anderen Modell eingefügt. Dabei ist es auch möglich, die *Substructure* um die xy-, xz- oder yz-

Ebene zu spiegeln. Ist eine *Substructure* in ein Hauptmodell eingebaut, können Elemente der *Substructure* vom Hauptmodell aus nicht editiert werden. Um die *Substructure* zu ändern, muss das Modell der *Substructure* geladen und entsprechend modifiziert werden. Diese Änderung wird dann in allen Hauptmodellen, in denen die *Substructure* eingebaut ist, übernommen.

3.2.1.8 *Input Functions*

Zur Modellierung nichtlinearer Zusammenhänge, zum Beispiel nichtlinearer Federkennlinien, werden *Input Functions* verwendet. Die Kennlinie einer *Input Function* wird durch Punktepaare vom Benutzer definiert. Zwischen diesen Punkten wird interpoliert, wobei SIMPACK dem Benutzer mehrere Auswahlmöglichkeiten vorgibt:

- Keine Interpolation, die Kennlinie verläuft dann stufenweise
- Lineare Interpolation, die Kennlinie verbindet zwei Punkte durch eine Gerade
- Kubische Spline-Approximation
- Akima-Spline-Approximation

3.2.1.9 3D-Geometrien zur Repräsentation von Körpern

Mit 3D-Geometrien kann ein mechanisches System grafisch dargestellt werden. Diese Geometrien sind in einem *Ensemble* abgelegt, das mit dem Element, das grafisch dargestellt werden soll, bewegt wird. Jedes Ensemble enthält eine oder mehrere Geometrien, die durch von SIMPACK bereitgestellte *Primitives* (einfache Geometrien wie Zylinder, Prismen, etc.) oder durch CAD-Geometrien definiert werden.

Diese Geometrien können auch skaliert werden, um bestimmte Elemente, wie z. B. Federn, korrekt darzustellen. Dabei wird dann das zu skalierende Abmaß in der grafischen Darstellung entsprechend den berechneten Werten verkürzt oder verlängert.

3.2.1.10 Gravitation

Die Gravitationsbeschleunigung kann in jede beliebige Richtung zeigen und beliebige Werte annehmen. Standardmäßig ist g= -9.81m/s² in z-Richtung.

Für Schienenfahrzeuge wird üblicherweise ein Koordinatensystem verwendet, bei dem die z-Achse nach unten weist. Die Gravitationskonstante muss dementsprechend auf g= +9.81m/s² in z-Richtung geändert werden.

3.2.1.11 *Sensors*

Sensors werden in SIMPACK verwendet, um die Relativbewegung zwischen zwei *Markern* zu berechnen. Wird ein neuer Körper erstellt, so wird diesem Körper von SIMPACK automatisch ein *Sensor* zugewiesen, der die Bewegung des Körpers zum Inertialsystem berechnet. Dieser *Sensor* wird verwendet, um die Bewegung der 3D-*Ensembles* zu ermitteln (3.2.1.9).

Vom Benutzer können zusätzliche *Sensoren* zwischen zwei beliebigen *Markern* definiert werden. Sie werden verwendet, wenn die Relativbewegung, -geschwindigkeit oder -beschleunigung zwischen zwei Punkten ermittelt werden soll.

Die von einem *Sensor* erfassten kinematischen Größen sind:

- Position
- Geschwindigkeit
- Beschleunigung
- Winkel
- Winkelgeschwindigkeit
- Winkelbeschleunigung

Die Bezeichnung von *Sensoren* erfolgt mit *$S_Name des Sensors*.

3.2.1.12 Rad/Schiene Elemente

Schienenfahrzeuge bewegen sich entlang einer Strecke, die durch die Bogenlänge s parametrisiert wird. In SIMPACK wird daher die Bogenlänge s als Hauptzustandsvariable s(t) gewählt. Alle Rad/Schiene spezifischen Elemente sind in Relation zu dieser Bogenlänge definiert.

Eine Modellierung eines Schienenfahrzeuges beginnt mit der Definition der Strecke (*Track*), parametrisiert durch die Bogenlänge s. Danach wird der Radsatz mit einem *Joint*, der die Bogenlänge s(t) als Hauptzustandsvariable benutzt, definiert.

Eine genauere Erläuterung dieser Elemente erfolgt in Kapitel 3.7 Spezifische Elemente für den Rad/Schiene-Kontakt.

3.3 Das *Calculation* Modul

Im *Calculation* Modul können verschiedene Berechnungsmethoden vom Benutzer konfiguriert und ausgeführt werden.

3.3.1 *Assembly Test*

Hier wird überprüft, ob ein Modell zusammenbaubar ist und die rechte Seite der Bewegungsgleichungen berechnet.

$$\dot{x} = f(x,t) \tag{3.2}$$

mit den Zustandsvariablen $x = \left(\dot{z}^T, z^T\right) = \left(\dot{z}_j^T, \dot{z}_E^T, z_j^T, z_E^T, z_F^T\right)$

dabei ist

t:	Zeit
$\dot{z}_j$:	Ableitungen der Gelenkspositionen
$\dot{z}_E$:	Ableitungen der elastischen Koordinaten
z_j:	Gelenkspositionen
z_E:	Elastische Koordinaten
z_F:	Zustandsvariable der Eigendynamik von Kraftelementen

Als Ausgangswerte werden die angegebenen *Initial States* verwendet.

3.3.2 Inverse Kinematik

Inverse Kinematik wird verwendet, um die Kinematik und Kinetik von Systemen mit kinematischer Schleifenstruktur und vorgegebener Bewegung (0 Freiheitsgrade) zu berechnen.

3.3.3 Statisches Gleichgewicht (*Static Equilibrium*)

Hier wird die statische Gleichgewichtslage eines Systems berechnet. Gleichgewichtslage bedeutet, dass die Beschleunigungen Null sind. Die Bedingung für statisches Gleichgewicht lautet daher:

$$\ddot{z} = f(\dot{z}, z, t) \equiv 0 \tag{3.3}$$

Auch Systeme, die sich translatorisch mit konstanter Geschwindigkeit bewegen, können sich im Gleichgewicht befinden (quasistatisch).

3.3.4 *Nominal Force Parameter*

Mit *Nominal Force Parameter* können unbekannte Parameter von Kraftelementen, wie zum Beispiel Federvorspannkräfte oder Nulllängen, berechnet werden. Dabei werden die *Force Parameters* berechnet, die zur Erreichung des Gleichgewichtszustands notwendig sind. Die Bedingung für den Gleichgewichtszustand ist wieder, dass die Beschleunigungen Null sind.

3.3.5 Eigenwerte (*Eigenvalues*)

Dieser Modul berechnet die Eigenwerte und Eigenvektoren der linearisierten Bewegungsgleichungen des mechanischen Systems. Die nichtlinearen Bewegungsgleichungen werden dafür um einen Punkt linearisiert.

3.3.6 Zeitintegration (*Time Integration*)

Das *Time Integration* Modul löst die Bewegungsgleichungen numerisch durch Integration nach der Zeit. Die Bewegungsgleichungen können gegeben sein durch:

- Gewöhnliche Differentialgleichungen
- Differential-algebraische Gleichungen
- Gewöhnliche Differentialgleichungen oder Differential-algebraische Gleichungen mit zustandsabhängigen Unstetigkeiten

Resultate der Zeitintegration sind die zeitabhängigen Zustandsgrößen $z_j(t)$, $\dot{z}_j(t)$, … des Modells.

Bei Differentialgleichungen und Integrationsmethoden werden folgende Ansätze häufig verwendet:

- *ODE* (**O**rdinary **D**ifferential **E**quation): Explizite gewöhnliche Differentialgleichung $\dot{x} = f(x,t)$, wird in SIMPACK benutzt um Mehrkörpersysteme mit offenen kinematischen Baumstrukturen zu beschreiben.
- *DAE* (**D**ifferential-**a**lgebraric **E**quation): Differential-algebraische Gleichung der Form $\dot{x} = f(x,t,\lambda); c(x,t) = 0$. Werden in SIMPACK benutzt, um Mehrkörpersysteme mit geschlossenen Schleifen zu beschreiben, wobei $c(x,t) = 0$ die Schleifenschließbedingung und λ die Zwangskräfte sind.
- *Stiff*: *ODE* oder *DAE* werden als *Stiff* bezeichnet, wenn sie stark gedämpfte Eigenwerte oder weit auseinanderliegende Eigenfrequenzen aufweisen. Dies tritt auf, wenn Kraftelemente mit hoher Steifigkeit und/oder Dämpfung verwendet werden.
- Explizite Unstetigkeiten über der Zeit (*Explicit Time Discontinuity*): Unstetigkeiten auf der rechten Seite der Gleichung ($f(x,t)$ oder $f(x,t,\lambda)$).
- Implizite Unstetigkeiten über der Zeit (*Implicit Time Disconitnuity*): Unstetigkeiten, die nicht zu einer bestimmten Zeit definiert sind, sondern abhängig von den Zustandsgrößen des Mehrkörpersystems auftreten (z. B. ein Rad, das von der Schiene abhebt)

Je nach Art des Modells werden von SIMPACK unterschiedliche Integrationsmethoden angeboten.

Um eine Zeitintegration durchzuführen, müssen vom Benutzer folgende Eingaben gemacht werden:

- Typ der zu integrierenden Gleichungen
  - Nichtlineare oder linearisierte Differentialgleichungen
  - Numerisch oder symbolisch
- Integrationsmethode
- Anfangs- und Endzeitpunkt der Integration
- Anzahl der Ausgabezeitschritte
- Anfangsposition
- Endposition der Integration Abspeichern (ja/nein)
- Explizite Unstetigkeiten
- Integrationsmethodenspezifische Parameter (Maximale Schrittweite, Anfangsschrittweite…)

3.3.7 Berechnung der Messungen (*Perform Measurements*)

Nach der Zeitintegration oder einer Invers-Kinematik-Analyse sind nur die Positionen und Geschwindigkeiten der Gelenke bekannt. Der Grund dafür ist, dass sie direkte Resultate der Zeitintegration der Differentialgleichungen darstellen.

Das *Measurement* Modul berechnet die abhängigen Variablen, die für das *Post-Processing* Modul benötigt werden. Es berechnet

- Von außen ausgeübte Kräfte
- Zwangskräfte
- Sensormessungen

Um das *Measurement* Modul ausführen zu können, muss zuerst eine Zeitintegration durchgeführt werden.

3.3.8 Schnittstelle SIMPACK/Centric Studio

Die Schnittstelle SIMPACK/Centric Studio ermöglicht den Export von SIMPACK-Simulationsdaten in die Centric Studio Software. Die Resultate können dort mit den CAD-Geometrien dargestellt werden und es können z. B. mit den SIMPACK-Resultaten Kollisionsuntersuchungen durchgeführt werden.

Die Bedienung erfolgt durch das Starten des Moduls *New Studio MBS File*, das sich im Menü *Calculation – Perform Measurements* befindet.

Beim Start dieses Moduls erstellt SIMPACK eine MBS(**M**ulti **B**ody **S**imulation)-Datei mit der Bezeichnung <*Modellname*>.*mbs*. Dieses Datei ist eine komprimierte Datei, die der ZIP(=gängiger Standard für komprimierte Dateien)-Spezifikation entspricht und die daher zum Beispiel mit WinZip geöffnet werden kann.

Diese komprimierte Datei enthält eine XML-Datei (XML=E**X**tensible **M**arkup **L**anguage, ein Textformat für strukturierte Daten) und eine MBR-Datei (MBR=**M**ulti **B**ody **R**esult).

Die XML-Datei enthält die SIMPACK-Modellbeschreibung (Bild 3.5).

```
<?xml version="1.0" ?>
- <MBS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=".\MultiBodySimulation.xsd">
  <VERSION>1.000000</VERSION>
- <MODEL>
- <HEADER>
  <NAME>kurve_ohne_uebergang_2</NAME>
  <DESCRIPTION>This is the model description of the kurve_ohne_uebergang_2</DESCRIPTION>
  <DATE>2002-07-03</DATE>
  <VERSION>1.000000</VERSION>
  <LENGTHUNIT TYPE="m" CONVERSIONFACTOR="1.000000e+000" />
  <ANGLEUNIT TYPE="rad" CONVERSIONFACTOR="1.000000e+000" />
  <TRANSFORMTYPE TYPE="XYZEULER" />
  <COORDINATESYSTEM TYPE="ZUPXBACK" />
  </HEADER>
- <BODY>
  <NAME>$S_bogie1:$E_Isys</NAME>
  <DATE>2002-07-03</DATE>
  <VERSION>1.000000</VERSION>
  <LENGTHUNIT TYPE="m" CONVERSIONFACTOR="1.000000e+000" />
  <ANGLEUNIT TYPE="rad" CONVERSIONFACTOR="1.000000e+000" />
  <TRANSFORMTYPE TYPE="XYZEULER" />
  <TRANSFORM X="0.000000e+000" Y="0.000000e+000" Z="0.000000e+000" ROT1="0.000000e+000"
  ROT2="0.000000e+000" ROT3="0.000000e+000" />
  <CGOFFSET X="0.000000e+000" Y="0.000000e+000" Z="0.000000e+000" />
  </BODY>
- <BODY>
  <NAME>$S_bogie1:$E_wheelset1_axle</NAME>
  <DATE>2002-07-03</DATE>
  <VERSION>1.000000</VERSION>
  <LENGTHUNIT TYPE="m" CONVERSIONFACTOR="1.000000e+000" />
  <ANGLEUNIT TYPE="rad" CONVERSIONFACTOR="1.000000e+000" />
  <TRANSFORMTYPE TYPE="XYZEULER" />
  <TRANSFORM X="1.908000e+001" Y="9.152175e-014" Z="-4.248741e-001" ROT1="1.804158e-009"
  ROT2="8.805020e-003" ROT3="-4.230764e-011" />
  <CGOFFSET X="0.000000e+000" Y="0.000000e+000" Z="0.000000e+000" />
```

Bild 3.5 Ausschnitt aus einer XML-Datei

Die XML-Datei enthält zuerst einen Hinweis auf das verwendete Dateiformat (<MBS xmlns:xsi=…> - unter der angegebenen Internet-Adresse können die Informationen über das verwendete Dateiformat abgerufen werden.) (Bild 3.5). Danach erfolgt neben der Angabe des Namens des SIMPACK-Modells und des Datums, an dem die Datei erstellt wurde eine Angabe über die verwendeten Einheiten (<LENGTHUNIT> und <ANGLEUNIT>, im vorliegenden Fall „m“ und „rad“). Danach wird die verwendete Winkeltransformation (<TRANSFORMTYPE>) angegeben. Nach diesen grundlegenden Informationen folgt nun eine Auflistung aller SIMPACK-*Ensembles*. Zu jedem *Ensemble* werden wieder die verwendeten Einheiten und Winkeltransformationen aufgelistet. Zusätzlich wird für jedes *Ensemble* die Anfangsposition des körperfesten Referenzsystems

in SIMPACK ausgegeben (<TRANSFORM X=…, Y=…, Z=…, ROT1=…, ROT2=…, ROT3=…). Mit diesen Informationen kann dann die Centric-Software den *Configuration Folder* automatisch erstellen (Kapitel 4.7).

In der MBR-Datei sind die Bewegungs- und Skalierungsdaten für jeden Ausgabezeitschritt und für jedes SIMPACK-*Ensemble* enthalten. Diese Datei ist binär codiert und kann daher nicht eingesehen werden.
Notwendige Voraussetzung um ein MBS-File erzeugen zu können ist die vorherige Durchführung einer Zeitintegration.

3.4 Das *Parameter Variation* Modul

Das *Parameter Variation* Modul ermöglicht eine automatisierte Analyse der Auswirkungen von Veränderungen einzelner Parameter auf das Modell.

3.5 Das *Optimization* Modul

Das *Optimization* Modul ermöglicht es dem Benutzer, für einen Parameter den „besten Kompromiss“ hinsichtlich mehrerer, widersprüchlicher Anforderungen zu finden.

3.6 Das *Post Processing* Modul

Das *Post-Processing* Modul ermöglicht die grafische Darstellung der in SIMPACK berechneten Resultate. Im *Post-Processing* Modul werden geboten:

- *2D-Plots*. Das *2D-Plot* Modul ermöglicht die Anzeige von SIMPACK Variablen in einem Diagramm.
- *3D-Animation* bietet eine 3-dimensionale Animation des mechanischen Modells mit den entweder vorher in der Zeitintegration berechneten (*Off-line*) Ergebnissen oder während der Zeitintegration (*On-line*).
- *Fast Fourier Transformation* der Ergebnisse

3.7 Spezifische Elemente für den Rad/Schiene-Kontakt

Die SIMPACK Software beinhaltet zahlreiche Elemente, die zur Modellierung von Schienenfahrzeugen erforderlich sind. Der Aufbau erfolgt zuerst durch die Definition der Strecke (*Track*). Danach wird der Rad/Schiene-Kontakt mit einem *Wheel/Rail-Joint* modelliert. SIMPACK bietet für diese Modellierung eine Bibliothek mit Reibungsgesetzen und Kontaktdefinitionen für den Rad/Schiene-Kontakt.

3.7.1 Streckendefinition (*Track*)

Durch die Streckendefinition wird der Verlauf der Schienenstrecke im dreidimensionalen Raum beschrieben. Diese Geometrie wird von den Modellierungselementen, wie *Joints*, *Marker*, etc. benutzt, um die dynamische Bewegung entlang der Strecke zu beschreiben.

Die kartesischen Koordinaten x(s), y(s) und z(s) der Strecke und die Orientierung der Strecke werden durch die Integration von Streckenvariablen wie der Krümmung κ(s) berechnet. Diese Integration wird von SIMPACK automatisch durchgeführt. Die resultierende Strecke wird durch einen kubischen Spline dargestellt, der während der Simulation verwendet wird.

3.7.1.1 Theoretische Grundlagen

Für die Bewegung längs einer regulären Kurve ist die eindeutige Ortsbestimmung $\mathrm{x}(t)$ nicht nur durch die Vorgabe der „Zeit“ t, sondern auch durch den bis zu diesem Zeitpunkt zurückgelegten Weg $s = s(t)$ möglich. Denn wegen $\dot{s}(t) = |\dot{x}(t)| > 0$ ist $s = s(t)$ als monotone Funktion (zumindest theoretisch) eindeutig nach t auflösbar, $t = t(s)$.
Man sagt die Kurve $\underline{\mathrm{r}} : [0, \mathrm{L}] \to \mathrm{IR}^3$, $\underline{\mathrm{r}}(s) = \underline{\mathrm{x}}(t(s))$ entsteht aus der regulären Kurve $\underline{x}$ durch Umparametrisierung auf Bogenlänge. Die Bogenlänge als Parameter ist dadurch ausgezeichnet, dass der die Kurve begleitende Tangentialvektor $\frac{d}{ds}\underline{\mathrm{r}}(s)$ konstant den Betrag 1 hat; dadurch ergeben sich besonders elegante Formeln [26]:
Formeln von FRENET. Für eine dreimal stetig differenzierbare Kurve $\underline{\mathrm{r}}(s)$, parametrisiert durch die Bogenlänge s, die überall eine Krümmung $\kappa(s) \neq 0$ besitzt, gelten folgende Formeln:

$$\frac{d\underline{\mathrm{T}}}{ds} = \kappa \cdot \underline{\mathrm{N}} \tag{3.4}$$

$$\frac{d\underline{\mathrm{N}}}{ds} = -\kappa \cdot \underline{\mathrm{T}} + \tau \cdot \underline{\mathrm{B}} \tag{3.5}$$

$$\frac{d\underline{\mathrm{B}}}{ds} = -\tau \cdot \underline{\mathrm{N}} \tag{3.6}$$

Dabei ist:
$\underline{\mathrm{T}}$: Tangentialvektor der Kurve $\mathrm{r}(s)$
$\underline{\mathrm{N}}$: Normalvektor der Kurve $\mathrm{r}(s)$
$\underline{\mathrm{B}}$: Binormalenvektor der Kurve $\mathrm{r}(s)$, $\underline{\mathrm{B}} = \underline{\mathrm{T}} \times \underline{\mathrm{N}}$
$\kappa(s)$: Krümmung der Kurve $\underline{\mathrm{r}}(s)$
$\tau(s)$: Torsion der Kurve $\underline{\mathrm{r}}(s)$

Daraus folgt:

$$\underline{\mathrm{T}} = \frac{d\underline{\mathrm{r}}}{ds} = \underline{\mathrm{r}}' \tag{3.7}$$

$$\frac{d\underline{\mathrm{T}}}{ds} = \frac{d^2\underline{\mathrm{r}}}{ds^2} = \underline{\mathrm{r}}'' \tag{3.8}$$

Die Krümmung ergibt sich zu:

$$\kappa^2 = \underline{\mathrm{r}}''^{\mathrm{T}} \cdot \underline{\mathrm{r}}'' \tag{3.9}$$

Die Torsion folgt mit:

$$\tau = \frac{1}{\kappa^2}\underline{\mathrm{r}}' \cdot \left(\underline{\mathrm{r}}'' \times \underline{\mathrm{r}}'''\right) \tag{3.10}$$

Geometrische Betrachtung, dargestellt in der Horizontalprojektion der Bogenlänge $\bar{s}$ [25] für einen Streckenverlauf ohne Steigungen und Gefälle:

Die erste Achse des begleitenden Dreibeins, die sich mit der Bogenlänge $\bar{s}$ bewegt, fällt mit dem Tangentialvektor zusammen. Die Orientierung dieses Dreibeins relativ zum Inertialsystem ist durch folgende drei Winkel gegeben (Bild 3.6):

ψ: Winkel in der Horizontalprojektion der des Elements ds
γ: Steigungswinkel
ϕ: Rotationswinkel um ds

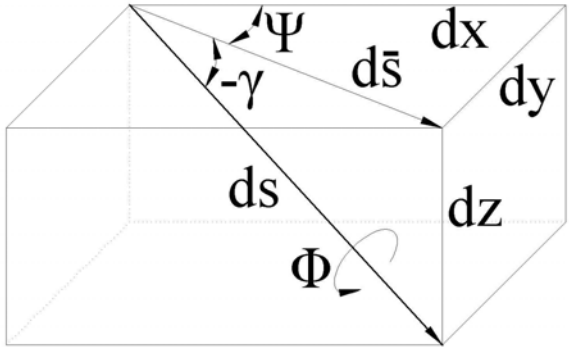

Bild 3.6 Bogenelement

Die Reihenfolge der Rotationen ist 3,2,1:

$$\underline{e}_1 = \underline{A}_{12} \cdot \underline{e}_2 = \underline{A}^z(\psi) \cdot \underline{A}^y(\gamma) \cdot \underline{A}^x(\phi) \cdot \underline{e}_2 \tag{3.11}$$

Die Kurve gibt den Verlauf der Schienenmitte an. Die Ableitungen nach $\bar{s}$ können angeschrieben werden als (Bild 3.6):

$$\frac{d\underline{r}}{d\bar{s}} = \begin{bmatrix} \frac{dx}{d\bar{s}} \\ \frac{dy}{d\bar{s}} \\ \frac{dz}{d\bar{s}} \end{bmatrix} = \begin{bmatrix} \cos\psi \\ \sin\psi \\ -\tan\gamma \end{bmatrix} \tag{3.12}$$

Der Zusammenhang zwischen s und $\bar{s}$ ergibt sich nach (Bild 3.6):

$$d\bar{s} = ds \cdot \cos\gamma \tag{3.13}$$

Ein *Standard Track* (siehe Kapitel 3.7.1.2) ist definiert durch folgende Parameter:

$\bar{R}(\bar{s})$: Radius des Streckenverlaufs in der horizontalen Ebene
$u(\bar{s})$: Überhöhung gemessen in der Vertikalebene (Bild 3.7)
$e_0(\bar{s})$: Referenzlänge für die Überhöhung (Bild 3.7)

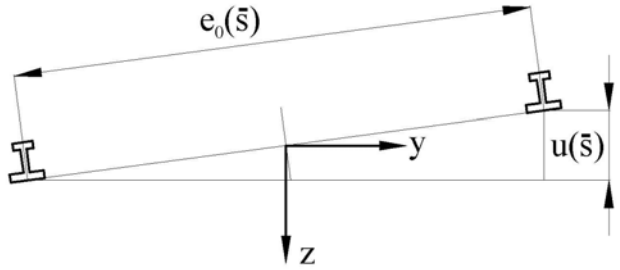

Bild 3.7 Parameter der Überhöhung

Für eine Rechtskurve werden Radius und Überhöhung der Kurve als positiv, für eine Linkskurve als negativ definiert.

Die Höhe über Grund (gemessen in Gleismitte) kann ausgedrückt werden durch:

$$z = -\frac{|u|}{2};\ \frac{dz}{d\bar{s}} = -\frac{1}{2} \cdot \frac{du}{d\bar{s}} \cdot sign(u) = -\tan\gamma;\ \frac{d^2z}{d\bar{s}^2} = -\frac{1}{2} \cdot \frac{d^2u}{d\bar{s}^2} \cdot sign(u) \tag{3.14}$$

Aus der ersten Ableitung der Höhe über Grund (3.14) folgt:

$$\gamma = \tan^{-1}\left(\frac{sign(u)}{2}\cdot\frac{du}{d\bar{s}}\right);\ \frac{d\gamma}{d\bar{s}} = \frac{1}{2}\cdot\frac{d^2u}{d\bar{s}^2}\cdot sign(u)\cdot\cos^2\gamma \tag{3.15}$$

und damit:

$$\frac{d^2\gamma}{d\bar{s}^2} = \left(\frac{1}{2}\cdot\frac{d^3u}{d\bar{s}^3}\cdot\cos^2\gamma - \frac{d^2u}{d\bar{s}^2}\cdot\frac{d\gamma}{d\bar{s}}\cdot\sin\gamma\cdot\cos\gamma\right)\cdot sign(u) \tag{3.16}$$

Nach (3.16) hängt also die zweite Ableitung des Steigungswinkels von der dritten Ableitung der Überhöhung ab.

Für den Rollwinkel ergibt sich aus der Geometrie die folgende Beziehung:

$$\sin\phi = \frac{u(\bar{s})}{e_0(\bar{s})}\cdot\cos\gamma \tag{3.17}$$

Die Ableitungen von (3.17) nach $\bar{s}$ führen zu:

$$\frac{d\phi}{d\bar{s}} = \frac{1}{e_0\cdot\cos\phi}\cdot\left\{\frac{du}{d\bar{s}}\cdot\cos\gamma - u\cdot\frac{d\gamma}{d\bar{s}}\cdot\sin\gamma\right\} - \frac{1}{e_0}\cdot\frac{de_0}{d\bar{s}}\cdot\tan\phi \tag{3.18}$$

$$\begin{aligned}\frac{d^2\phi}{d\bar{s}} = {} & \frac{1}{e_0\cdot\cos\phi}\cdot\left\{\left[\frac{d^2u}{d\bar{s}^2} - u\cdot\left(\frac{d\gamma}{d\bar{s}}\right)^2\right]\cdot\cos\gamma - \left[u\cdot\frac{d^2\gamma}{d\bar{s}^2} + 2\cdot u\cdot\frac{d\gamma}{d\bar{s}}\right]\cdot\sin\gamma\right\} + \\ & \tan\phi\cdot\left\{\left(\frac{d\phi}{d\bar{s}}\right)^2 - \frac{1}{e_0}\cdot\frac{d^2e_0}{d\bar{s}^2}\right\} - \frac{1}{e_0}\cdot\frac{d\phi}{d\bar{s}}\cdot\frac{de_0}{d\bar{s}}\end{aligned} \tag{3.19}$$

Die anderen Variablen erhält man aus der Integration:

$$\frac{d\psi}{d\bar{s}} = \frac{1}{\bar{R}(\bar{s})};\ \frac{d^2\psi}{d\bar{s}^2} = -\frac{1}{\bar{R}^2}\cdot\frac{d\bar{R}}{d\bar{s}} \tag{3.20}$$

Eingabeparameter für *Standard Tracks*

Für *Standard Tracks* sind die Eingaben:

u: Überhöhung

$\frac{1}{\bar{R}}$: Krümmung in der Horizontalebene

Daraus ergeben sich die Differentialgleichungen für den Streckenverlauf folgendermaßen:

Unabhängige Variable: $x_S = (z_1, z_2, z_6)^T$
mit den Differentialgleichungen:

$$z_1' = x' = \cos\psi \tag{3.21}$$
$$z_2' = y' = \sin\psi \tag{3.22}$$
$$z_6' = \psi' = \frac{1}{\overline{R}} \tag{3.23}$$

Abhängige Variable: $x_d = (z_3, z_4, z_5)^T$
als Funktion der Überhöhung u und ihrer Ableitungen:

$$z_3 = z = -\frac{|u|}{2} \tag{3.24}$$
$$z_4 = \phi = \sin^{-1}\frac{u(\bar{s})}{e_0(\bar{s})} \cdot \cos\gamma \tag{3.25}$$
$$z_5 = \gamma = \tan^{-1}\left(\frac{u'}{2}\right) \cdot sign(u) \tag{3.26}$$

Übergangsbogen und Überhöhung

Um zum Beispiel einen geraden Streckenabschnitt mit einer Kurve mit konstantem Radius zu verbinden, verwendet man eine so genannte Klothoide. Eine Klothoide ist eine Kurve mit einer proportional zur Bogenlänge zunehmenden (bzw. abnehmenden) Krümmung. Ihr Verlauf wird durch die FRESNELSCHEN Integrale beschrieben.

Die Start- und Endpunkte des Übergangsbogens entsprechen auch den Start- und Endpunkten der Überhöhungsrampe.

In SIMPACK können für diesen Übergang zwei Funktionen verwendet werden:

- Lineare Rampe (*Straight Ramp*)
- S-förmige Rampe (*S-Shaped Ramp*)

Lineare Rampe (*Straight Ramp*)

Hier wird eine lineare Rampe für den Verlauf der Krümmung und den Anstieg der Überhöhung im Übergangsbogen gewählt (Bild 3.8).

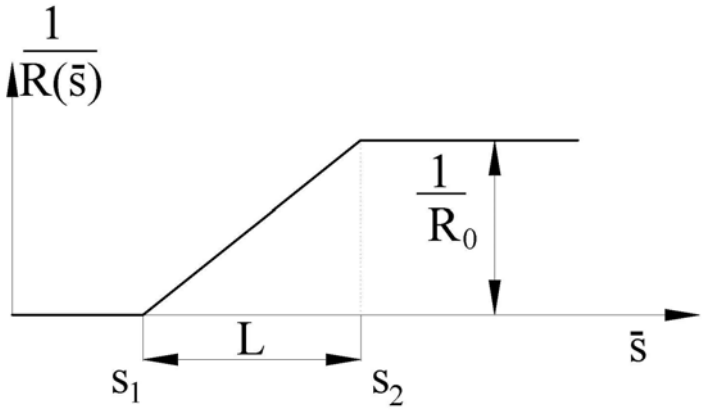

Bild 3.8 Verlauf der Krümmung in der Horizontalebene bei einem Kurveneingang

Im Bereich $s_1 < \bar{s} < s_2$ gilt:

$$\frac{d\psi}{d\bar{s}} = \frac{1}{\bar{R}(\bar{s})} = \frac{1}{R_0 \cdot L}(\bar{s} - s_1) \tag{3.27}$$

Die zweite Ableitung von ψ weist an den Punkten s_1 und s_2 Unstetigkeitsstellen auf. Der Anstiegswinkel für die Überhöhungsrampe verläuft wie die Krümmung in (Bild 3.8) und weist Unstetigkeitsstellen an den Punkten s_1 und s_2 auf.

Im Bereich $s_1 < \bar{s} < s_2$ gilt für den Anstiegswinkel:

$$\tan\gamma = \frac{u_0}{2 \cdot L} \tag{3.28}$$

S-förmige Rampe (*S-shaped Ramp*)

Hier wird der Übergangsbogen mit Hilfe einer parabolischen Funktion (Bild 3.9) dargestellt. Die zweite Ableitung von ψ ist dabei stetig.

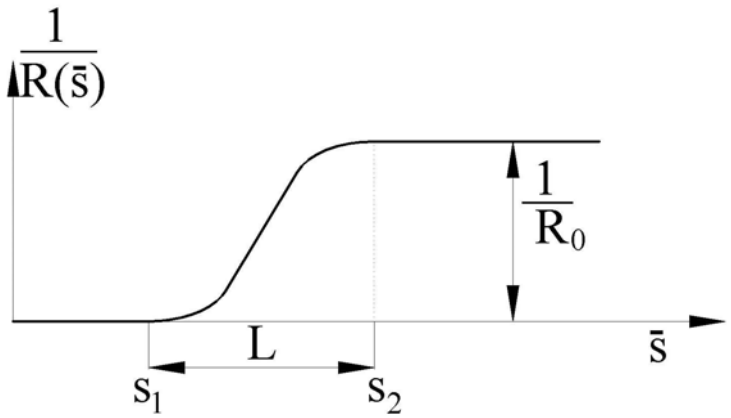

Bild 3.9 Verlauf der Krümmung im Übergangsbogen bei S-förmiger Rampe

Der Anstiegswinkel γ ist hier ebenfalls stetig, nicht aber seine Ableitung.

Da es aber für die Integration notwendig ist, dass sowohl die Winkel ψ und γ als auch ihre Ableitungen stetig sind, wird um die kritischen Punkte s_1 und s_2 eine Glättungsfunktion angewandt. Diese Glättungsfunktion für die Funktion der Überhöhung $u(\bar{s})$ und die horizontale Krümmung $\frac{d\psi}{d\bar{s}}$ muss bis zur dritten Ableitung stetig sein, da $\frac{d^2\gamma}{d\bar{s}^2}$ eine Funktion der dritten Ableitung ist. Um diese Bedingungen zu erfüllen, wird ein Polynom siebenter Ordnung verwendet. Die halbe Länge H des Abschnitts, auf den die Glättungsfunktion angewandt wird, kann vom Benutzer definiert werden (Bild 3.10).

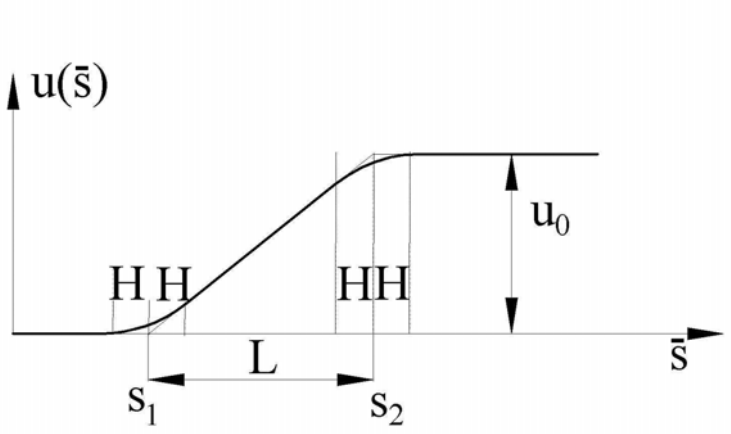

Bild 3.10 Verlauf der Überhöhung in Abhängigkeit von $\bar{s}$ mit Glättungsfunktion

Durch die Anwendung dieser Glättungsfunktion kommt es zu einer Differenz im SIMPACK-Streckenverlauf und dem erwarteten Streckenverlauf ohne diese Glättungsfunktion. Für die Konstruktion des Lichtraumprofils zur Kollisionsuntersuchung in Centric müssen daher die Streckendaten aus SIMPACK verwendet werden.

3.7.1.2 Bedienung des Moduls *Track*

Der Streckenverlauf wird durch folgende Parameter vom Benutzer definiert:

Art des Streckenverlaufs (*Type*):

- *Standard Track*. *Standard Track* umfasst gängige Szenarien wie
  - Gerade Strecke (*Straight Track*)
  - Kreisbogen (*Constant Curvature*)
  - Kurveneingang (Gerade - Übergangsbogen – Kreisbogen) (*Curve Entry*)
  - Kurve mit Ein- und Ausfahrt (Gerade – Übergangsbogen – Kreisbogen – Übergangsbogen – Gerade) (*Curve Passing*)
  - Doppelkurve mit geradem Zwischenstück (Gerade – Übergangsbogen – Kreisbogen – Übergangsbogen – Kreisbogen in Gegenrichtung - Übergangsbogen – Gerade) (*Sign-Change of Curvature*)
  - Doppelkurve ohne gerades Zwischenstück (Gerade – Übergangsbogen – Kreisbogen – Übergangsbogen – Kreisbogen in Gegenrichtung – Übergangsbogen – Gerade) (*Cross Over*)

- *Cartographic Track*. Durch Zusammenbau der Strecke aus mehreren Abschnitten, die jeweils aus Gerade – Übergangsbogen – Kreisbogen – Übergangsbogen bestehen, sowie durch Vorgabe von Überhöhungen und Höhenverlauf der Strecke, können nahezu beliebige Streckenverläufe vorgegeben werden.

- *Measured Track*. Streckenvorgabe durch in einer Datei abgelegte Informationen über den Streckenverlauf.

Art der Übergangsrampe (*Ramp/Transition*)
- Linear oder
- S-förmig (s. 3.7.1.1)

Art der Überhöhung (*Superelevation Type*)

- Rotation um die Gleismitte
- Rotation um die kurveninnere Schiene

Überlagerung von Gleislagestörungen (*Excitation*)

- Keine
- *Track-related* (linke und rechte Schiene weisen gleiche Störungen auf)

- *Rail-related* (Definition für jede Schiene einzeln)

Art des Schienenmodells (*Flexibility*)

- Starr (*Inertia-fixed Rails*). Die Schienen sind inertialfest
- Elastisch (*Discrete Track Model*). Die Schienen sind gegenüber dem Inertialsystem elastisch gelagert.

Diskretisierung (*Discretization*)

Vorgabe der Anzahl der Stützpunkte für die gesamte Strecke und für jene Abschnitte, auf die Glättungsfunktionen angewendet werden.

Parameter

Eingabe der jeweils erforderlichen Parameter wie Kurvenradien, Länge von Übergangsbögen, etc.

Als Ergebnisse aus dem *Track* Modul können jene Funktionen, die die Lage der Strecke definieren, ausgegeben werden:

- Positionen: $x(s), y(s), z(s), \phi(s), \gamma(s), \psi(s), u(s)$
- Erste Ableitungen: $x'(s), y'(s), z'(s), \phi'(s), \gamma'(s), \psi'(s), u'(s)$
- Zweite Ableitungen: $x''(s), y''(s), z''(s), \phi''(s), \gamma''(s), \psi''(s), u''(s)$

Die Definition der Streckenlage wird in einem für den Anwender nicht lesbaren Format in der Datei <*Modellname*>.*trk* abgelegt. Für *Standard Tracks* besteht optional die Möglichkeit, die Information in einem lesbaren ASCII-*File* zu speichern.

3.7.2 Definition des Rad/Schiene-Kontakts

Für die Definition des Rad/Schiene-Kontakts wird von SIMPACK der *Joint Type 07: Wheel/Rail Joint* bereitgestellt. Dieser *Joint* kann ohne Rad/Schiene-Funktionalität verwendet werden und ist dann ein Sechs-Freiheitsgrad-Gelenk, das aber anstatt des Koordinatensystems x-y-z und der zugehörigen Rotationen ein mit dem Streckenverlauf bewegtes Koordinatensystem s-y-z mit den entsprechenden Rotationen verwendet.

Wird die Rad/Schiene-Funktionalität aktiviert, so wird von SIMPACK automatisch ein Rad/Schiene Kontakt mit den zugehörigen Bezugssystemen, *Markers*, *Sensors*, Kräften und Zwangsbedingungen hergestellt (Bild 3.11).

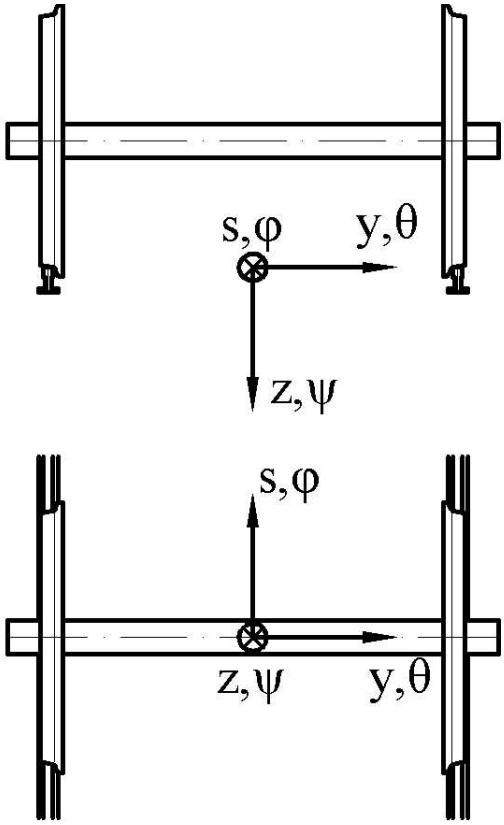

Bild 3.11 Koordinatensystem beim SIMPACK Wheel/Rail Joint 07 mit Rad/Schiene-Funktionalität

Bei aktivierter Rad/Schiene-Funktionalität werden die 6 Freiheitsgrade des *Wheel/Rail Joints* durch Einfügen zweier Zwangsbedingungen verringert (Bild 3.11). Nach (3.1) verbleiben damit 6-2=4 Freiheitsgrade.

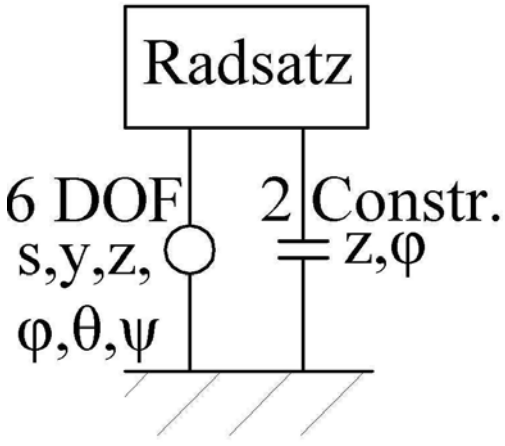

Bild 3.12 Kinematisches Schema des Radsatzes
Gelenk mit Angabe der Freiheitsgrade, Constraint mit Angabe der gesperrten Bewegungen

Dabei sind s, y, θ und ψ unabhängige Variable, z und φ abhängige Variable.

Mit dem Rad/Schiene Kontakt werden zwei *Ensembles ($E_Radsatzname_ProfRef_Right und $E_Radsatzname_ProfRef_Left*) erzeugt, um die Räder grafisch darzustellen.

Weiters werden von SIMPACK mehrere *Marker* generiert:

- *$M_Wheel_ProfRef*: Liegt im Mittelpunkt des Rades
- *$M_Wheel_Contact*: Ursprung im Kontaktpunkt des Rades mit der Schiene am Rad. Die Position ist das Resultat der Kontaktpunktberechnung (Punkt, an dem die Reibungskräfte und Rad/Schiene-Zwangsbedingungen angreifen)
- *$M_Rail_ProfRef*: Bewegt sich entlang der Schiene
- *$M_Rail_Contact*: Ursprung im Kontaktpunkt des Rades mit der Schiene an der Schiene. Die Position ist das Resultat der Kontaktpunktberechnung (Punkt, an dem die Reibungskräfte und Rad/Schiene-Zwangsbedingungen angreifen)
- *$M_Rail_Track_Frame*: Bewegt sich entlang der Gleismitte
- *$M_Rail_Track_Camera*: Bewegt sich entlang der Gleismitte, aber ohne Überhöhungen

3.7.3 Globale Fahrzeugwerte (*Vehicle Globals*)

In diesem Fenster werden die fahrzeugspezifischen Parameter eingegeben. Diese Parameter sind:

- Geschwindigkeit des Fahrzeugs mit v=const. im vorgegebenen Fall.
- Spurweite
- Profile von Rad und Schiene
- Radius des Rads
- Schienenneigung
- Reibungsgesetz und –parameter
- Kontaktmodell (z. B. starrer/elastischer Kontakt)

Kapitel 4. Das Virtual Product Development Program Centric Innovation

Centric Innovation ist ein vom in San Jose, Kalifornien ansässigen Unternehmen Centric Software entwickeltes Programm zur virtuellen Produktentwicklung. Centric Innovation ermöglicht die Zusammenarbeit mehrerer Teams über das Internet. Mit Centric Innovation können zahlreiche Projektinformationen in einem standardisierten Umfeld unternehmensweit abgerufen werden [27].

Die Centric Innovation Software besteht aus mehreren Modulen:

Studio

- ***Studio*** ist das Basismodul. Von hier aus kann der Benutzer ein Objekt darstellen, manipulieren und mit kinematischen und logischen Verknüpfungen animieren. Die Bedienoberfläche bietet die übliche Microsoft Windows Oberfläche und Funktionalität.
- ***2D*** erlaubt es, zweidimensionale Daten wie CAD-Zeichnungen, Bilder oder Logos einzufügen und zu verwalten.
- Mit ***Behavior*** kann ein Modell mit kinematischen und logischen Verknüpfungen animiert werden.
- ***Local Connectors*** ermöglichen es, verschiedene Datenformate zu importieren und zu exportieren. Zu den unterstützten Formaten gehören zahlreiche CAD/CAM, CAE, PDM, ERP-Anwendungen und die Microsoft Office Produkte (siehe auch Unterpunkt *Connector Hub*).

Connector Hub

Der Centric Innovation *Connector Hub* sammelt die Produktdefinitionen und Strukturinformationen verteilter CAD-Quellen und macht sie in Centric Studio zugänglich.

- ***Connector Hub Server*** verwaltet Produktdaten und macht Struktur- und Geometrieinformationen in Centric Studio zugänglich.
- ***Connector Hub Remote Connector*** konvertiert Daten von zahlreichen CAD-Systemen und ermöglicht den Centric Studio Nutzern den Zugriff auf sie. Unterstützt werden dabei:
  - AutoCAD (2D-CAD-Software)
  - CATIA (3D-CAD-Software)
  - I-DEAS MS7 (FEM-Software)
  - I-DEAS MS8 (FEM-Software)
  - IGES (**I**nitial **G**raphics **E**xchange **S**pecification, Datenformat zum Austausch von 3D-CAD-Daten)
  - Pro/ENGINEER (3D-CAD-Software)
  - Pro/INTRALINK (Software zur webbasierten Zusammenarbeit mit Pro/ENGINEER)
  - Solid Works (3D-CAD-Software)

Collaboration Hub Product Modules

Centric Innovation *Collaboration Hub* ermöglicht Entwicklerteams den Austausch von Produktdaten und –definitionen über das Internet.

- Mit dem ***Site Administrator*** können einzelnen Benutzern und Benutzergruppen bestimmte Zugriffsrechte zugewiesen werden.
- ***Publisher*** ermöglicht es, für jedes gewünschte Element oder Produkt Änderungen zu verwalten.
- ***Team*** erlaubt Teammitgliedern die Teilnahme an Echtzeit Audio/Video-Konferenzen.
- ***Web*** ermöglicht es, große Datenmengen interaktiv über das Web zu übertragen.
- Mit ***Package Transfer*** kann ein sicherer Austausch von komprimierten Centric Studio Projekten über das Internet stattfinden.

4.1 Centric Studio

Die Centric Studio Bedienoberfläche (Bild 4.1) besteht aus mehreren Fenstern:

- *Information Browser*
- *3D Window*
- *Component Browser*
- *Active Project Browser*
- *Workspace Browser*
- *Output Window*

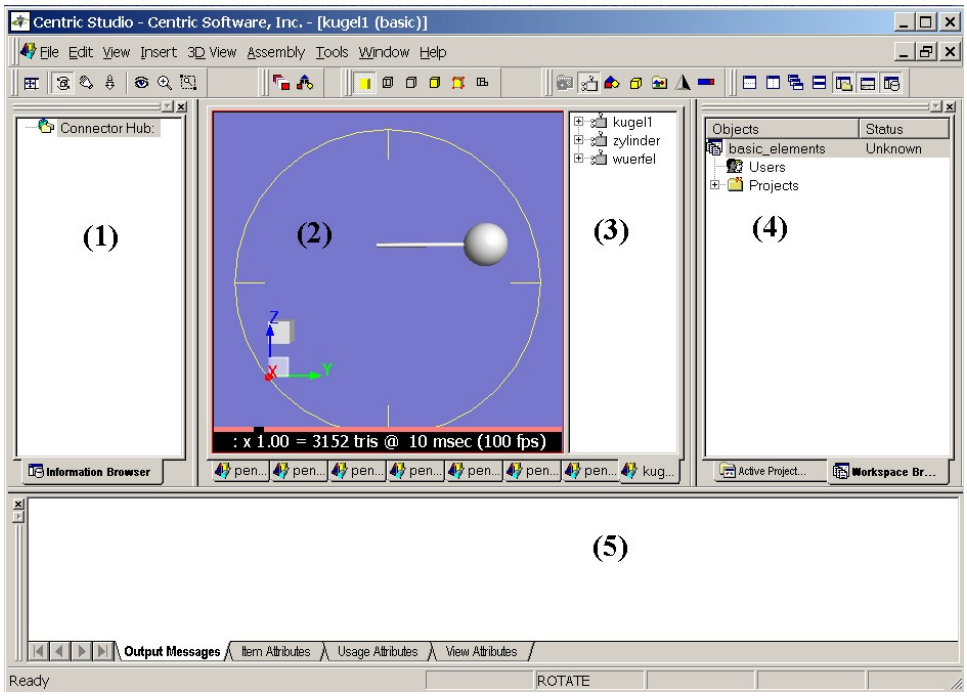

Bild 4.1 Centric Studio Bedienoberfläche: (1) Information Browser (2) 3D Window (3) Component Browser (4) Active Project Browser/Workspace Browser (5) Output Window

Information Browser

Der *Information Browser* ermöglicht den Zugriff auf Projektdaten über die *Centric Innovation Collaboration* und *Connector Hub*.

3D Window

Im *3D Window* werden die Modelle dargestellt. Die Anwendung von Zoomfunktionen und die Auswahl verschiedener Ansichten kann hier durchgeführt werden.

Component Browser

Im *Component Browser* werden die Komponenten des im *3D Window* angezeigten Modells mit ihrer hierarchischen Struktur dargestellt.

Active Project Browser

Im *Active Project Browser* werden alle Komponenten und Elemente des aktiven Projekts angezeigt. Ein Projekt ist eine Untergruppe eines *Workspac*e (z. B. *Workspace*: Triebwagen, Projekte: Drehgestell, Wagenkasten, …)

Workspace Browser

Im *Workspace Browser* werden alle Projekte und Benutzer in einem *Workspace* aufgelistet. Im *Workspace* sind die für einen Auftrag relevanten Informationen abgelegt und in einzelnen *Projects* organisiert. In der Bedienoberfläche kann im Fenster (4) (Bild 4.1) wahlweise entweder der *Active Project Browser* oder der *Workspace Browser* angezeigt werden.

Output Window

Im *Output Window* können folgende Fenster angewählt werden:

Output Messages zeigt Informationen über die Interaktion mit dem Benutzer, wie zum Beispiel Fehlermeldungen oder Resultate, an.

Im ***Item Attributes*** Fenster können Attribute wie zum Beispiel Teilenummern editiert und gelesen werden. Attribute, die im *Item Attributes* Fenster geändert werden, werden auch bei den Kopien des ausgewählten Objekts verändert (Wenn eine Geometrie mehrmals verwendet wird, kann diese in Centric mittels *Instancing* als Kopie aus der nur einmal vorhandenen Ursprungsgeometrie erstellt werden).

Im ***Usage Attributes*** Fenster geänderte Attribute werden nur bei der ausgewählten Komponente und nicht bei *Instance*-Geometrien verändert.

Im ***View Attributes*** Fenster können die für die Positionierung der Elemente im Raum verantwortlichen Attribute (x-, y-, z-Verschiebung, Rotation um x-, y- und z-Achse, Maßstab in x-, y- und z-Richtung) manuell geändert werden (Bild 4.2).

Name	Type	Value	Unit
Type	Label	Simulation Configuration Group	
Name	Label	$E_Kugel	
Window Visibility	Status	☑	
X Translation	Length	0.000000	mm
Y Translation	Length	-311.148942	mm
Z Translation	Length	148.042798	mm
X Scale	Scalar	1.000000	
Y Scale	Scalar	1.000000	
Z Scale	Scalar	1.000000	
X Rotation	Angle	-50.889653	deg
Y Rotation	Angle	0.000000	deg
Z Rotation	Angle	0.000000	deg

Bild 4.2 View Attributes: X Translation, Y Translation, Z Translation: Verschiebungen in x, y, z-Richtung; X Scale, Y Scale, Z Scale: Skalierungsfaktoren (Maßstab) in x, y, z-Richtung; X Rotation, Y Rotation, Z Rotation: Rotationen um x, y, z-Achse

4.2 *Cross Sections*

Mit *Cross Sections* können Schnitte durch ein Modell gelegt werden.

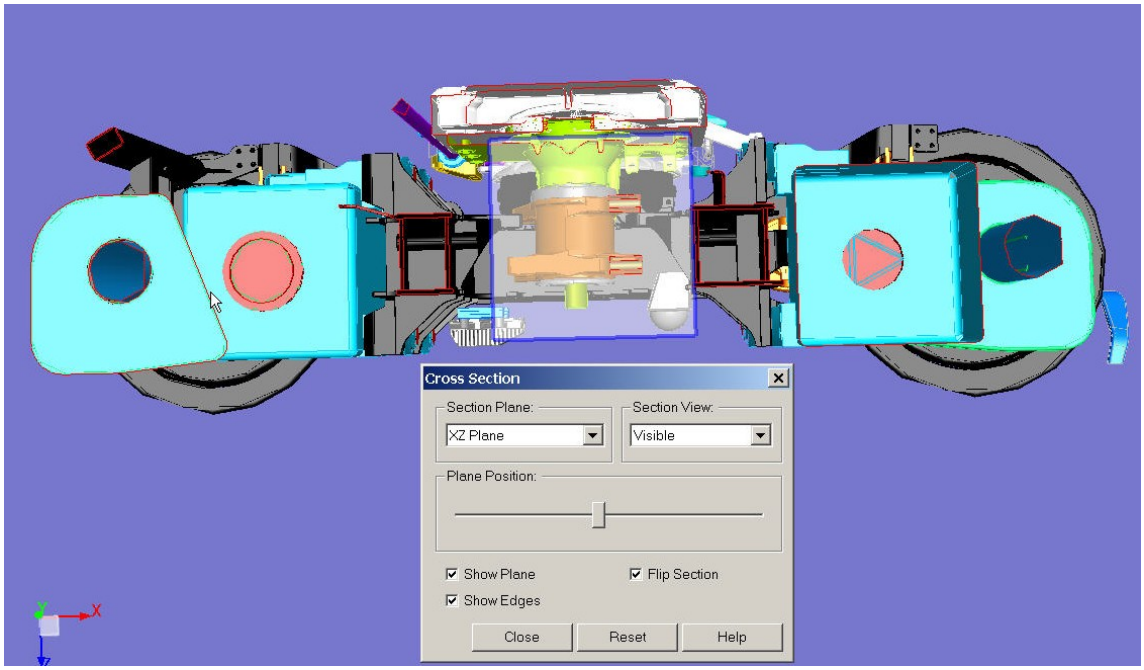

Bild 4.3 Anwendung von Cross Section – Längsschnitt eines Drehgestells

In der *Cross Section* Dialogbox (Bild 4.3) kann die Position der Schnittebene mit dem Schieber *Plane Position* ausgewählt werden. Die Schnittebene kann als xy-, yz- oder xz-Ebene bezogen auf das Weltkoordinatensystem oder durch benutzerdefinierte Punkte oder Geraden (zum Beispiel Tangenten an einen beliebigen Punkt einer Oberfläche) definiert

werden. Weiters kann die Position der Schnittebene auch durch Eingabe von Koordinaten festgelegt werden. Ein einmal erstellter Schnitt kann im *Active Project Browser* im *Cross Sections Folder* abgespeichert werden.

4.3 Kollisionserkennung

In Centric Studio können Bauteile auf Kollisionen geprüft werden. Im *Active Project Browse*r können *Component-Folder* angelegt werden, in die bestimmte Geometrien abgelegt werden. Damit sind diese Geometrien dann als Einzelteile definiert. Bei einer Kollisionsprüfung werden dann diese *Components* entsprechend den Einstellungen im *Assembly* Menü überprüft. Es können entweder alle Teile gegeneinander auf Kollisionen geprüft werden, oder es können in der *Collision Settings* Dialogbox in *Intra-Component-Collision* jene Teile ausgewählt werden, die auf Kollisionen untersucht werden sollen. In der Dialogbox kann außerdem eingestellt werden, dass bereits die Unterschreitung eines Mindestabstandes zwischen Komponenten als Kollision gemeldet wird. Die Meldung einer Kollision geschieht durch eine Farbänderung (Bild 4.4) der in die Kollision involvierten Bauteile, wobei die Farbe, die die Teile dann annehmen, in *Collision Settings* eingestellt werden kann.

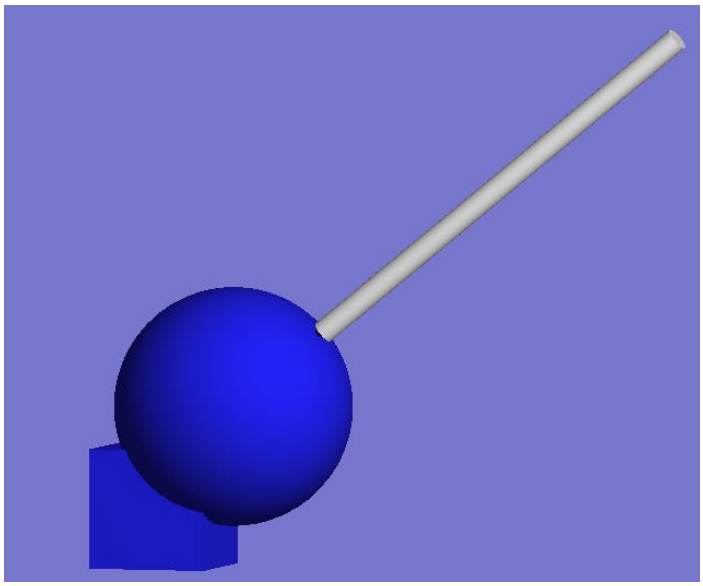

Bild 4.4 Anzeige einer Kollision durch Farbänderung (Blaue Teile kollidieren)

Weiters können sowohl das Auftreten einer Kollision wie auch das Ende einer Kollision durch Logikfunktionen abgefragt werden. Nicht angezeigt werden kann die exakte Stelle der Kollision.

Die Kollisionsberechnung wird mit Hilfe von Programmbibliotheken durchgeführt, die an der University of North Carolina entwickelt wurden (siehe dazu auch Kapitel 2.5) [16,17,23,24].

4.4 *Behaviors*

Mit *Behaviors* können Modellen einfache Funktionen verliehen werden. Es gibt zwei Arten von *Behaviors*: *Mechanisms* definieren kinematische Zusammenhänge zwischen einzelnen Geometrieteilen. Mit *Mechanisms* können einfache Gelenke zwischen Teilen dargestellt werden. *Logics* ermöglichen es, einzelnen Teilen ein logisches Verhalten aufzuprägen.

4.4.1 *Logics* (Logiken)

Die Struktur (Bild 4.5) einer *Logic* besteht aus mehreren *Modes*, die sogenannte *Activities* enthalten. Die *Activities* enthalten die Anweisungen, wie sich das Modell zu verhalten hat.

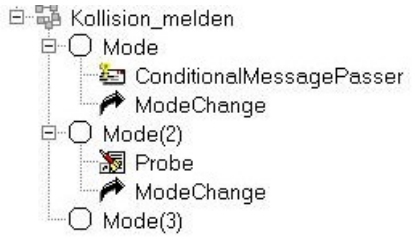

Bild 4.5 Aufbau einer Logic

4.4.1.1 *Activities*

Von Centric wird eine große Auswahl von *Activities* angeboten:

- *Attribute Changer* lesen ein oder mehrere Attribut(e) einer oder mehrerer Komponente(n) aus, wenden einen vom Benutzer definierten mathematischen oder logischen Ausdruck darauf an und weisen das Ergebnis einem oder mehreren Attribut(en) zu. Auf diese Weise kann zum Beispiel durch Ändern der *View Attributes* eine Komponente bewegt werden.
- *Message Passer* senden eine Botschaft solange der Mode, in dem sie sich befinden, aktiv ist.
- *Timed Message Passer* senden ebenfalls eine Botschaft solange der *Mode*, in dem sie sich befinden, aktiv ist. Der Beginn der Sendung ist allerdings gegenüber dem Aktivwerden des *Modes* um eine vom Benutzer gewählte Zeitspanne verzögert.
- *Conditional Message Passer* senden eine Botschaft, wenn ein vom Benutzer definierter Ausdruck „wahr“ ist.
- *Pair Distance Sensor* berechnet die kürzeste Distanz zwischen zwei Komponenten und sendet das Ergebnis zu einem benutzerdefinierten Zielattribut.
- *Environment Distance Sensor* berechnet die kürzeste Distanz zwischen einer Komponente und jener Komponente, die ihr am nächsten ist. Das Ergebnis wird zu einem Zielattribut gesendet.

- *Path Executor* bewegt eine Komponente entlang eines vorab punktweise definierten Pfades.
- *Pick Activity* überträgt eine Botschaft, wenn eine benutzerdefinierte Komponente oder eine Geometrie innerhalb einer Komponente mit einem Mausklick ausgewählt wird.
- *Data Probe Activity* ermöglicht es, *Input* oder *Output Data Probes* (siehe Abschnitt 4.5) innerhalb einer Logik aufzurufen.
- Mit *Sound Player* können Geräusche (vorgegeben durch Dateien im WAV-Format) abgespielt werden, solange der *Mode*, in dem sich die *Sound Player Activity* befindet, aktiv ist.

4.4.1.2 Aufbau einer Logik

Beim Start der Logik wird der als *Startmode* ausgewählte *Mode* aktiv. Als Default-Einstellung ist der erste *Mode Startmode*, es kann vom Benutzer aber ein beliebiger anderer *Mode* als *Startmode* definiert werden. Ist der *Mode* aktiv, werden alle darin enthaltenen *Activities* durchgeführt. Der Wechsel auf einen anderen *Mode* geschieht durch einen *Mode Change*. Der *Mode Change* wird durch eine Botschaft, die zum Beispiel von einem *Conditional Message Passer* gesendet wird, ausgelöst. Der ursprünglich aktive *Mode* wird deaktiviert und der vom Benutzer im *Mode Change* definierte *Zielmode* wird aktiv. Anstatt einer von einer *Acitivity* erzeugten Botschaft kann auch der Kollisionsbeginn oder das Kollisionsende entsprechend den eingestellten *Collision Settings* als Auslöser für einen *Mode Change* verwendet werden. Durch das Definieren eines *Logic Controller* in einem *Mode Change* kann auch die gesamte Logik abgeschaltet werden.

4.5 *Data Probes*

Man unterscheidet zwischen zwei verschiedenen Arten von *Data Probes*:

- *Input Data Probes*
- *Output Data Probes*

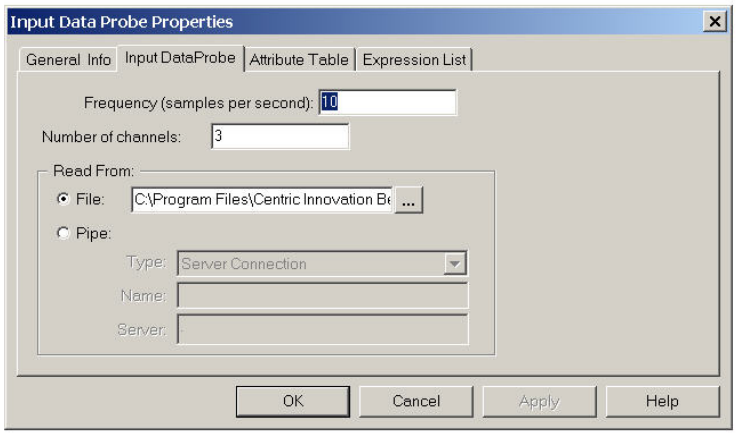

Bild 4.6 Input Data Probe Definitionsfenster

Input Data Probes lesen Daten aus einer vorhandenen Datei aus und ordnen sie vom Benutzer ausgewählten Attributen einer Komponente zu. Die Zuordnung geschieht mit einer vom Benutzer einstellbaren Frequenz (Bild 4.6), wobei aber nach oben Grenzen durch die Zeit, die für die Berechnung des neuen Bildes und, bei eingeschalteter Kollisionsprüfung, für die Überprüfung auf Kollisionen benötigt wird (*Frame Rate*), gesetzt sind. Auf diese Weise kann in Centric eine in einem anderen Programm berechnete Bewegung dargestellt werden: In dem für die Berechnung der Bewegung verwendeten Programm werden die translatorischen und rotatorischen Bewegungen der Teile, die in Centric dargestellt werden sollen, bezogen auf das inertialfeste Ursprungskoordinatensystem, ausgelesen und in eine für Centric lesbare Datei geschrieben, in Centric wird diese Datei dann mit einer *Input Data Probe* eingelesen und die Koordinaten den entsprechenden *View Attributes* zugeordnet. Durch die Wahl der Frequenz, mit der die Daten eingelesen werden, kann die Simulation gegenüber den Originaldaten verlangsamt und sozusagen in „Zeitlupe“ oder beschleunigt im „Zeitraffer“ dargestellt werden.

Output Data Probes schreiben vom Benutzer ausgewählte Attribute in eine Datei. Die Frequenz, mit der die Attribute gelesen und in die Datei geschrieben werden, kann auch hier vom Benutzer gewählt werden. Die Attribute werden hier in eine ASCII-Datei geschrieben, wobei die Werte einer einzelnen Abtastung in einer Zeile stehen und durch Tabulator getrennt sind.

Sowohl bei *Input* wie auch bei *Output Data Probes* können die aus der Datei eingelesenen bzw. aus den Attributen gelesenen Daten bevor sie den Attributen zugewiesen bzw. in die Datei geschrieben werden, noch mit mathematischen Funktionen verändert werden.

4.6 Viewpoints

Will man eine bestimmte Ansicht eines Modells später wieder auffinden, so kann man sie als *Viewpoint* speichern. Auch ein *Viewpoint* ist durch *View Attributes* definiert. Die Koordinaten in den *View Attributes* eines *Viewpoints* geben hier Position und Orientierung des Beobachters an. Ein *Viewpoint* kann auch durch manuelles Ändern der *View Attributes* modifiziert werden. Nach der Eingabe der neuen Position des Beobachters wird der *Viewpoint* aber erst durch Doppelklicken aktualisiert. Es ist daher momentan nicht möglich, einen bewegten *Viewpoint* durch Ändern der *View Attributes* mit einer Logik zu erzielen.

4.7 *Simulations*

Mit *Simulations* können verschiedene *Behaviors* an einer Struktur angewendet werden. Eine *Simulation* besteht aus einer *Configuration* und einem *Scenario*.
In der ***Configuration*** befindet sich die gesamte Geometrie der *Simulation*. Die einzelnen Teile werden dabei aus den ursprünglichen Geometrien mittels *Drag and Drop* kopiert.

Die Ursprungsgeometrien und ihre Anordnung werden durch die Simulation nicht verändert.
Der ***Scenario Folder*** enthält alle *Behaviors*, wie *Logics, Data Probes* etc. Alle diese *Behaviors* wirken nur auf die Geometrien im *Configuration Folder* und nicht auf die Ursprungsgeometrien.

Weiters können in einer Simulation **Simulationsergebnisse aus SIMPACK** importiert werden. Dabei wird eine von SIMPACK erstellte MBS-Datei in Centric eingefügt. Hier wird dann automatisch ein *Configuration* und ein *Scenario Folder* erstellt. Der *Configuration Folder* enthält alle Teile, für die in SIMPACK ein eigenes *Ensemble* definiert worden ist. Im *Scenario Folder* sind dann die Positionsdaten sowie (bei in SIMPACK skalierten Elementen wie zum Beispiel Federn) eventuelle Skalierungsdaten dieser Objekte zu den jeweiligen Ausgabezeitschritten in SIMPACK in *Data Probes* abgelegt.

Mittels *Drag and Drop* werden den *Ensembles* im *Configuration Folder* die entsprechenden Teile in Centric Innovation zugeordnet. Wichtig ist daher, dass die Koordinatensysteme und die Anordnung der Einzelteile in diesen Koordinatensystemen in SIMPACK und Centric übereinstimmen bzw. durch Ändern der *View Attributes* in Centric zur Übereinstimmung gebracht werden. Natürlich können im *Configuration Folder* auch zusätzliche *Logics* eingefügt werden.

Kapitel 5. Modellaufbau in SIMPACK und Centric Innovation. Überprüfen der Schnittstelle SIMPACK/Centric und der Kollisionserkennung.

Die für eine Simulation in Centric notwendige Berechnung der Bewegung eines Modells muss vorab in SIMPACK durchgeführt werden. Dabei wird zuerst ein Mehrkörpermodell des zu prüfenden Systems erstellt. Die Dynamik wird dann in SIMPACK berechnet. Mit der in SIMPACK integrierten Schnittstelle zu Centric wird eine MBS-Datei erstellt. In dieser Datei sind die Koordinaten der körperfesten Referenzsysteme der einzelnen Bestandteile des Mehrkörpermodells zu den jeweiligen Ausgabezeitschritten enthalten. Weiters werden auch Informationen über die Struktur des Modells, die Centric zum Aufbau des *Configuration Folders* benötigt, in die Datei geschrieben.

In Centric wird diese Datei importiert und automatisch ein *Simulation Folder* erstellt. Hier müssen jetzt die Komponenten des Centric-Modells den jeweiligen Äquivalenten in der SIMPACK-Simulation, die im *Configuration Folder* automatisch eingetragen werden, mittels *Drag and Drop* zugewiesen werden. Danach kann die Simulation gestartet werden.

Zur Überprüfung der Funktion und zur Erarbeitung eines Konzeptes zur Vorgangsweise bei der Kollisionsüberprüfung wurden zwei Modelle verwendet:

- Ein Pendel mit einer Dämpfung im Aufhängungspunkt, das aus einem ausgelenkten Zustand eine freie, gedämpfte Schwingung ausführt und dabei mit einem Würfel, der sich in Centric in der Bahn des Pendels befindet, kollidiert.
- Ein vereinfachtes Modell eines Zuges. Die Daten wurden von Siemens SGP Verkehrstechnik zur Verfügung gestellt und sind an einen DESIRO UK Triebwagenzug angelehnt. Dieser Zug wird entlang verschiedener Strecken bewegt und in Centric auf Kollisionen mit einem vorgegebenen Lichtraumprofil geprüft.

5.1 Simulationen und Modellaufbau des Pendels

5.1.1 Modellaufbau in SIMPACK

Das Pendel wurde in SIMPACK aus einem Zylinder und einer Kugel aufgebaut. Diese Körper wurden mit folgenden Massen und Trägheitsmomenten versehen (Abmessungen nach (Bild 5.1)):

Körper	Masse [kg]	I_{XX} [kgm²]	I_{YY} [kgm²]	I_{ZZ} [kgm²]
Zylinder	1	10	10	1
Kugel	20	50	50	50

(Die Trägheitsmomente beziehen sich auf den Schwerpunkt des jeweiligen Körpers, die Achsangaben auf das körperfeste Koordinatensystem $\bar{x}, \bar{y}, \bar{z}$ bzw. ein analoges System für den Zylinder und S_1)

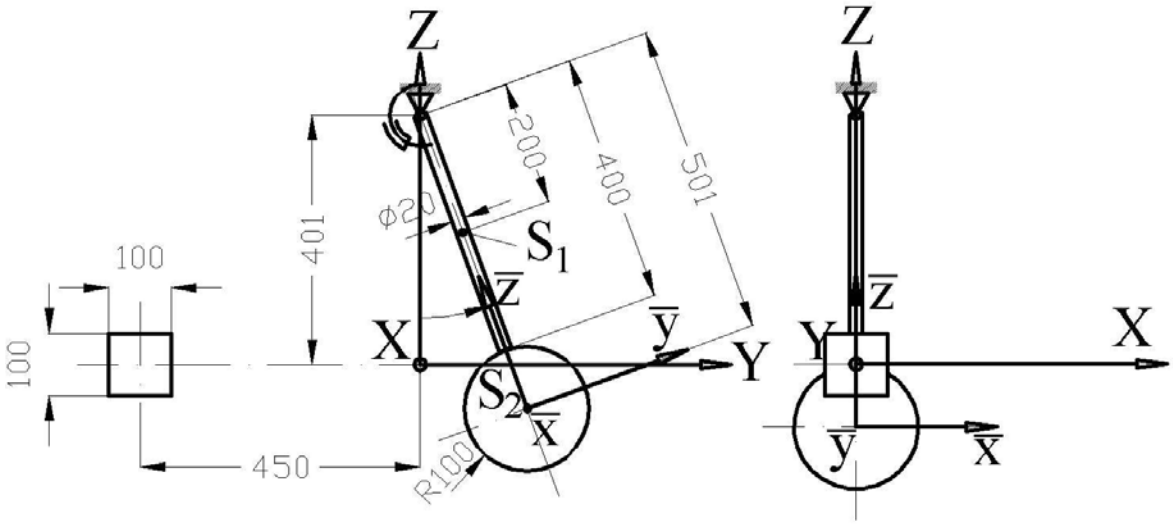

Bild 5.1 Pendel mit Würfel (S_1 ...Schwerpunkt des Zylinders, S_2 ...Schwerpunkt der Kugel , der Würfel wird in SIMPACK nicht dargestellt, da er nur für die Kollisionsprüfung in Centric erforderlich ist und auf die Berechnung der Dynamik keinen Einfluss hat)

Der Würfel wird in SIMPACK nicht modelliert, da er nur für die Kollisionsprüfung in Centric benötigt wird und keinen Einfluss auf die Dynamik der Bewegung hat (bei einer Kollision wird der Würfel widerstandslos durchdrungen).
Das Gelenk am Aufhängungspunkt zwischen Pendel und Inertialsystem ist ein Scharniergelenk. Am Scharniergelenk wurde eine Rotationsdämpfung mit einer Dämpfungskonstante von 5 Nms/rad (D=0.031) eingefügt.

Zur Überprüfung der Kollisionserkennung wurde zuerst in SIMPACK folgendes Szenario errechnet:

- Auslenken des Pendels auf $\varphi = \pi/2$ und in Folge
- Beginn einer abklingenden Schwingung um die Nulllage, bei der das Pendel in der Centric Simulation mehrmals den Würfel durchdringt.

Entsprechend dieser Vorgabe stellt sich ein Schwingungsverlauf nach (Bild 5.2) ein.

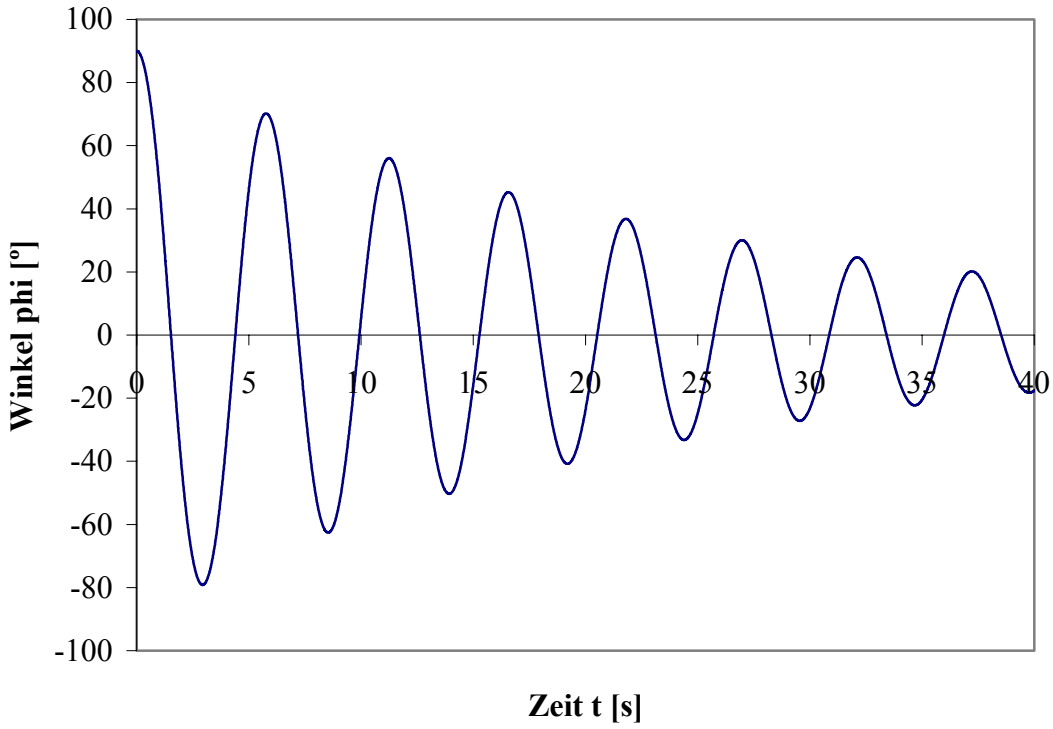

Bild 5.2 Errechnete Schwingung des Pendels in SIMPACK

Bei der Berechnung wurde der Schwingungsverlauf für eine Zeit t=40 Sekunden mit 4501 Ausgabezeitschritten ermittelt. Nach der Berechnung wurden diese Daten über die Schnittstelle SIMPACK/Centric an die Centric Software übergeben.

5.1.2 Modellaufbau in Centric und Durchführung von Simulationen

5.1.2.1 Modellaufbau

In Centric wurden die mit Pro/ENGINEER erstellten Geometrien des Pendels eingefügt und im Koordinatensystem entsprechend der Anordnung in SIMPACK platziert (Bild 5.3).

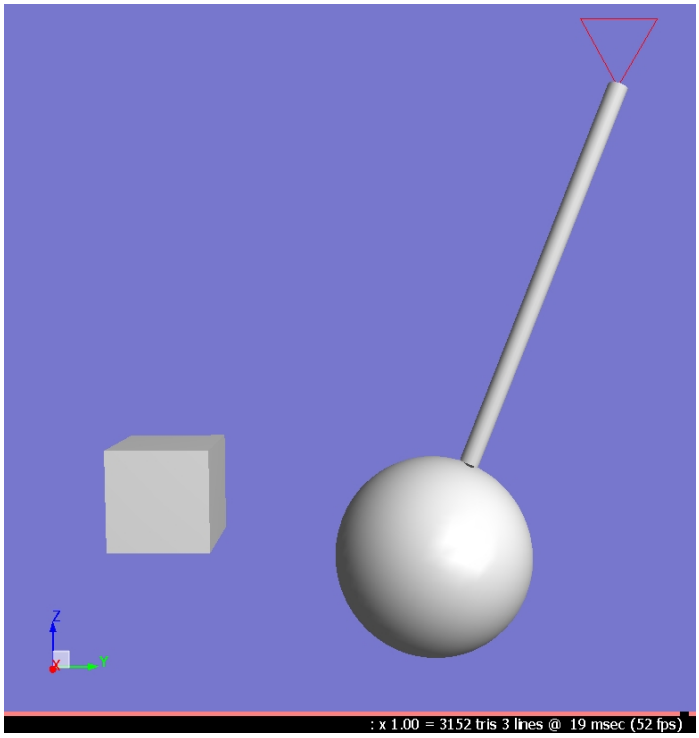

Bild 5.3 Darstellung des Pendels in Centric

Die Simulation der in SIMPACK berechneten abklingenden Schwingung führt zu einer Kollision der Kugel des Pendels mit dem inertialfesten Würfel. Dieser Kontakt hat keine Auswirkung auf den Verlauf der Schwingung, da die Vorgänge in Centric keine Auswirkung auf die Berechnung der Dynamik in SIMPACK haben. Der Würfel wird von der Kugel widerstandslos durchdrungen. Bei eingeschalteter Kollisionserkennung ändert sich während der Kollision (einer zumindest teilweisen Überdeckung der Volumina) die Farbe des Pendels und des Würfels. Weiters wurde die Möglichkeit genutzt, die Kollision in einer Logik abzufragen (siehe Kapitel 4.3) und ein vorab definiertes Attribut im Fall einer Kollision auf „1“, ansonsten auf „0“ zu setzen. Dieses Attribut wurde zusammen mit dem zugehörigen Winkel φ ausgelesen und mit einer *Output Data Probe* in einer Datei abgespeichert.

5.1.2.2 Berechnung der Winkel, bei denen eine Änderung des Kollisionsstatus stattfindet

Entsprechend der vorgegebenen Geometrien ergibt sich als Winkel für Kollisionsbeginn bzw. –ende:

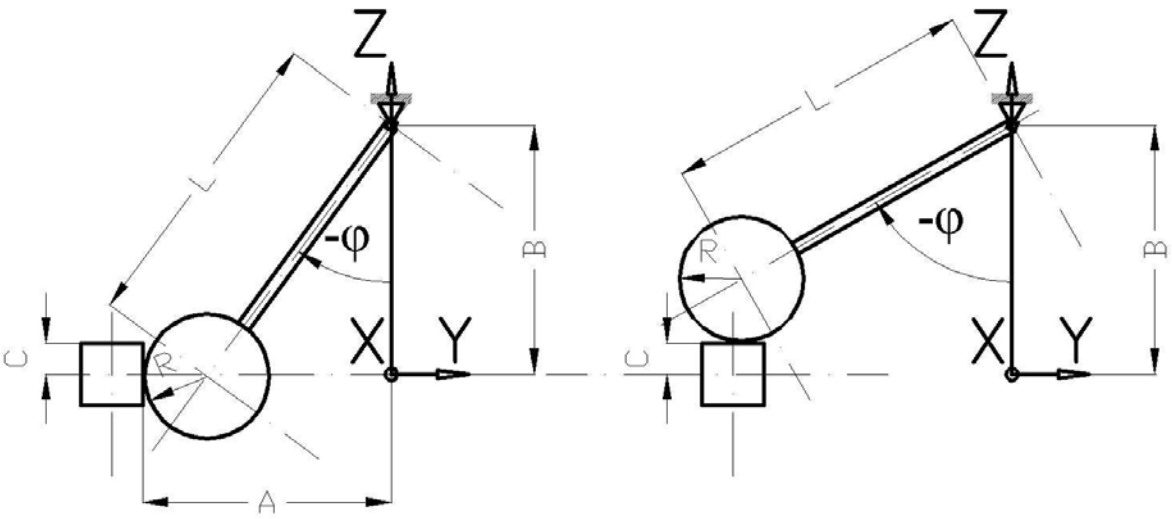

Bild 5.4 Ansicht des Pendel: Links: Kollisionsbeginn, rechts: Kollisionsende

Für den Beginn der Kollision bei der Aufwärtsbewegung ergibt sich nach (Bild 5.4):

$$L \cdot \sin(-\varphi) + R = A \qquad (5.1)$$

und für das Kollisionsende bei der Aufwärtsbewegung:

$$L \cdot \cos(-\varphi) + R = B - C \qquad (5.2)$$

Damit folgt mit A=400mm, B=401mm, C=50mm, L=501mm und R=100mm für den Kollisionsbeginn nach (5.1):

$$\varphi_{BEGINN} = -\arcsin\frac{300}{501} = -0.6420049 rad$$

und für das Kollisionsende nach (5.2):

$$\varphi_{ENDE} = -\arccos\frac{251}{501} = -1.0460448 rad$$

5.1.2.3 Vergleich der errechneten Werte mit der Simulation

In den folgenden Simulationen wurden die Positionswerte mit unterschiedlichen Abtastfrequenzen eingelesen. Diese Abtastfrequenz kann, wie in Kapitel 4.5 erläutert, für die *Data Probes* vom Benutzer eingestellt werden. Damit kann die Simulation gegenüber der Berechnung in SIMPACK in „Zeitraffer“ (bei hohen Abtastfrequenzen) oder „Zeitlupe“ (bei niedrigen Abtastfrequenzen) ablaufen. Der Geschwindigkeit ist aber nach oben dadurch eine Grenze gesetzt, dass die Zeit, die benötigt wird um den neuen Frame zu errechnen, gleichzeitig auch die Mindestzeit, die für einen Schritt benötigt wird, darstellt. Eine Frequenz, die höher ist als die sich aus dem Kehrwert dieser Berechnungszeit ergebende maximale Darstellungsfrequenz, kann nicht erreicht werden.

Überprüft wurden folgende Szenarien:

- Die Simulation in SIMPACK wurde mit einer Integrationszeit von t=40s und 4501 Ausgabezeitschritten durchgeführt. Die Positionsdaten werden in Centric mit einer Abtastrate von 100 Hz eingelesen.
- t=40s, 4501 Ausgabezeitschritte, Abtastrate 10 Hz
- t=40s, 4501 Ausgabezeitschritte, Abtastrate 1 Hz

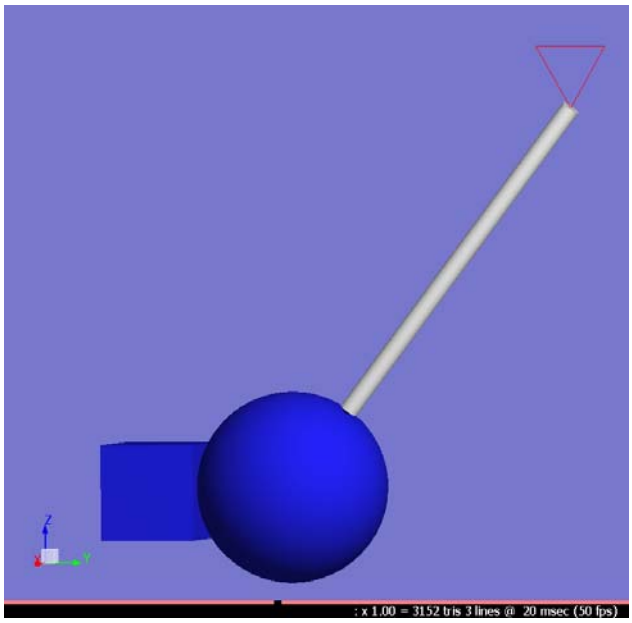

Bild 5.5 Pendel bei der Aufwärtsbewegung am Beginn der Kollision

Für das Szenario mit 4501 Schritten und **100 Hz** Abtastrate ergibt sich für die **Aufwärtsbewegung**:

Kollisionsstatus	**Winkel φ [rad]**	**Kommentar**
0 0 0 1 1	-0.63153195 -0.64413100 -0.65664822 -0.66908228 -0.68143219	Kollision sollte bereits hier angezeigt werden
1 1 1 0	-1.04455709 -1.05340576 -1.06214261 -1.07076716	Hier findet keine Kollision mehr statt

Und für die **Abwärtsbewegung**:

Kollisionsstatus	**Winkel φ [rad]**	**Kommentar**
0 0 0 1 1	-1.04723382 -1.03862011 -1.02990735 -1.02109623 -1.01218712	Kollision sollte bereits hier angezeigt werden
1 1 1 0	-0.64465755 -0.63276726 -0.62081295 -0.60879588	Hier findet keine Kollision mehr statt

Die Kollision wird sowohl bei der Aufwärts- wie auch bei der Abwärtsbewegung um einige Schritte zu spät erkannt – es kommt daher auch zu einer Differenz zwischen den Kollisionsmeldungen bei Auf- und Abwärtsbewegung. Ebenso wird das Kollisionsende in beiden Richtungen zu spät gemeldet.

Für das Szenario mit **10 Hz** Abtastrate ergibt sich bei der **Aufwärtsbewegung**:

Kollisionsstatus	**Winkel φ [rad]**	**Kommentar**
0 0 1 1	-0.63153195 -0.64413100 -0.65664822 -0.66908228	Kollision bereits ab hier
1 1 0 0	-1.04455709 -1.05340576 -1.06214261 -1.07076716	Kollisionsende bereits hier

Verglichen mit dem Szenario mit 100 Hz Abrastrate ist der Fehler hier kleiner, Kollisionsbeginn und –ende werden jeweils einen Schritt zu spät gemeldet.

Für das Szenario mit **1 Hz** Abtastrate ergibt sich für die **Aufwärtsbewegung**:

Kollisionsstatus	**Winkel φ [rad]**	**Kommentar**
0 1 1 1	-0.63153195 -0.64413100 -0.65664822 -0.66908228	Kollisionserkennung hier richtig
1 1 0 0	-1.03559709 -1.04455709 -1.05340576 -1.06214261	Kollisionserkennung hier richtig

Die Ursache für diese Abweichungen wird in Kapitel 5.1.2.5 erklärt.

5.2.1.4 Kollisionsprüfung gegen Ecke bei geringer Überdeckung

Als weiteres Szenario wurde die Position des Würfels so verändert, dass die Durchdringung von Kugel und Würfel nur eine kurze Zeitperiode andauert. Außerdem wurde der Würfel so verschoben, dass die Kollision der Kugel mit dem Würfel an einer Ecke des Würfels stattfindet (Bild 5.6).

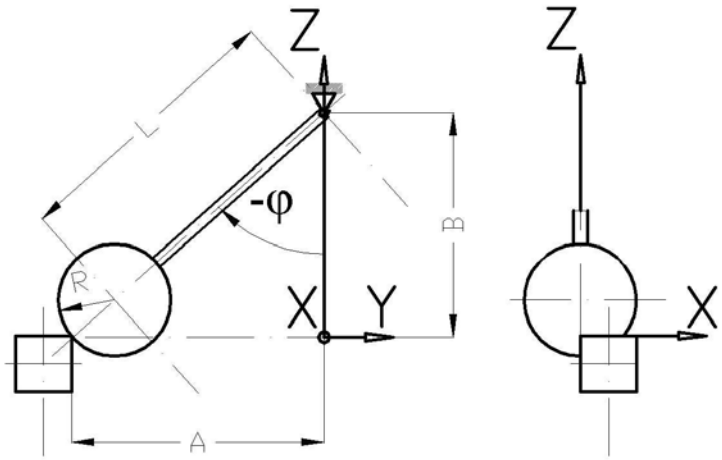

Bild 5.6 Konfiguration für neues Szenario

Für den Beginn der Kollision ergibt sich:

$$\varphi_{BEGINN} = -\left(\arctan\frac{A}{B} - \arccos\frac{A^2 + B^2 + L^2 - R^2}{2 \cdot \sqrt{A^2 + B^2} \cdot L} \right) \tag{5.3}$$

und für das Kollisionsende:

$$\varphi_{ENDE} = -\left(\arctan\frac{A}{B} + \arccos\frac{A^2 + B^2 + L^2 - R^2}{2 \cdot \sqrt{A^2 + B^2} \cdot L} \right) \tag{5.4}$$

Daraus folgt mit A=447.5mm, B=501mm, L=501mm und R=100mm für den Kollisionsbeginn nach (5.3):

$$\varphi_{BEGINN} = -\left(\arctan\frac{447.5}{401} - \arccos\frac{447.5^2 + 401^2 + 501^2 - 100^2}{2 \cdot \sqrt{447.5^2 + 401^2} \cdot 501} \right) = -0.8312346 rad$$

und für das Kollisionsende nach (5.4):

$$\varphi_{ENDE} = -\left(\arctan\frac{447.5}{401} + \arccos\frac{447.5^2 + 401^2 + 501^2 - 100^2}{2 \cdot \sqrt{447.5^2 + 401^2} \cdot 501} \right) = -0.8490574 rad$$

Der während der Kollision (Durchdringung von Kugel und Würfel) vom Pendel überstrichene Winkel beträgt also nur 0.0178228rad (=1.0212°).

Wie zuvor wird der Kollisionsstatus mit einer Logik abgefragt und gespeichert. Mit einer Abtastfrequenz von 10 Hz ergibt sich:

Kollisionsstatus	**Winkel φ [rad]**	**Kommentar**
0	-0.83270419	Kollision bereits hier
0	-0.84203762	
1	-0.85126740	Kollisionsende schon hier
0	-0.86039305	

Die Kollision wird also auch in diesem Fall erkannt, die Kollisionserkennungslogik hinkt aber auch hier hinter den tatsächlichen Kollisionszeitpunkten nach. Wird die Abtastfrequenz weiter abgesenkt, so erfolgt, wie zuvor, die Kollisionsmeldung zum richtigen Zeitpunkt.

An diesem Beispiel kann man besonders gut erkennen, dass die Abweichung zwischen berechneter Kollision und Ausgabe der Kollision in Centric nicht durch Ungenauigkeiten in der Kollisionsprüfung selbst, sondern durch Synchronisationsprobleme (s. Kapitel 5.1.2.5) zwischen Kollisionserkennung und der Logik, die zur Änderung des Kollisionsstatus führt, verursacht wird. Ändert man nämlich die Logik so, dass bei erstmaliger Kollision die Simulation angehalten wird, so findet die Kollision statt und wird auch durch Farbänderung (die Farbänderung zeigt die Kollision zum richtigen Zeitpunkt an) angezeigt, die Simulation wird aber erst angehalten, wenn die Kollision bereits wieder beendet ist (Bild 5.7).

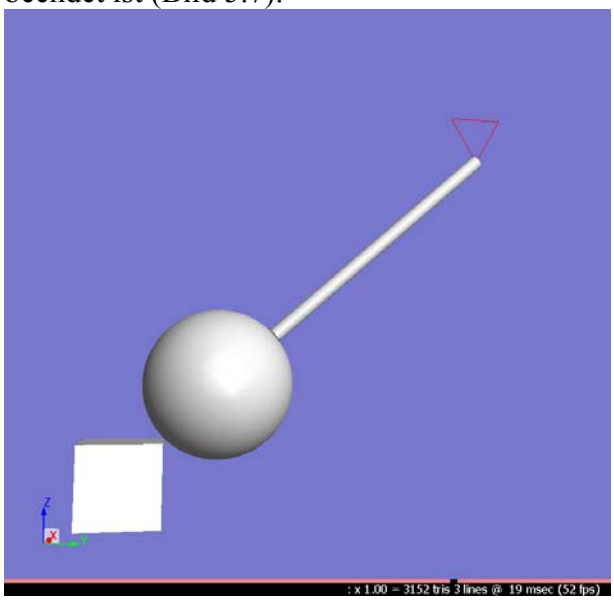

Bild 5.7 Durch Logik angehaltene Simulation. Das Pendel sollte eigentlich mit Beginn der Kollision gestoppt werden, wird aber durch die Zeitverzögerung erst angehalten, wenn die Kollision bereits wieder beendet ist.

5.1.2.5 Interpretation der Ergebnisse

Die Kollisionserkennung funktioniert auch in Fällen, wo die Überdeckung nur sehr klein und nur für eine kurze Zeitspanne vorhanden ist, zuverlässig. Begrenzt wird die Genauigkeit nur dadurch, dass Kollisionen, die kürzer andauern, als das Zeitintervall zwischen zwei Ausgabeschritten, nicht mehr sicher erkannt werden (Die Kollision wird für jeden Zeitschritt geprüft. Ist das Zeitintervall, in dem eine Kollision stattfindet, kürzer als das Zeitintervall zwischen zwei Ausgabeschritten, so liegt die Kollision unter Umständen zwischen zwei Schritte die noch bzw. wieder kollisionsfrei sind und wird nicht erkannt).

Bei hohen Abtastraten ergeben sich Fehler in der Ausgabe der Kollision. Diese Fehler werden aber nicht von der Kollisionserkennung selbst verursacht (z. B. durch geometrische Ungenauigkeiten), da sie bei einer Absenkung der Abtastfrequenz verschwinden. Die Ursache für diese Fehler liegt in der Ablaufstruktur der Kollisionserkennung in Centric Innovation. Die Kollisionserkennung läuft als so genannter separater *Thread* (abgesplitteter NT Background Process). Das bedeutet, dass mehrere Prozesse (z. B. Kollisionserkennung und Logik zur Kollisionserfassung) gleichzeitig und nicht sequentiell laufen. Bei hoher Rechenlast kann es hier zu Problemen bei der Synchronisierung dieser Prozesse kommen, was dann zu den zuvor festgestellten Abweichungen führt (Bild 5.9).

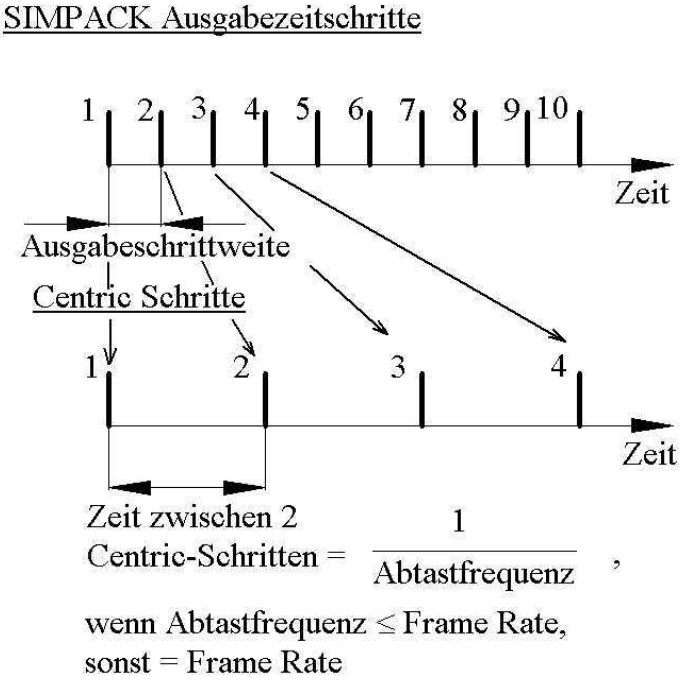

Bild 5.8 Zusammenhang SIMPACK Ausgabeschritte – Centric Darstellung

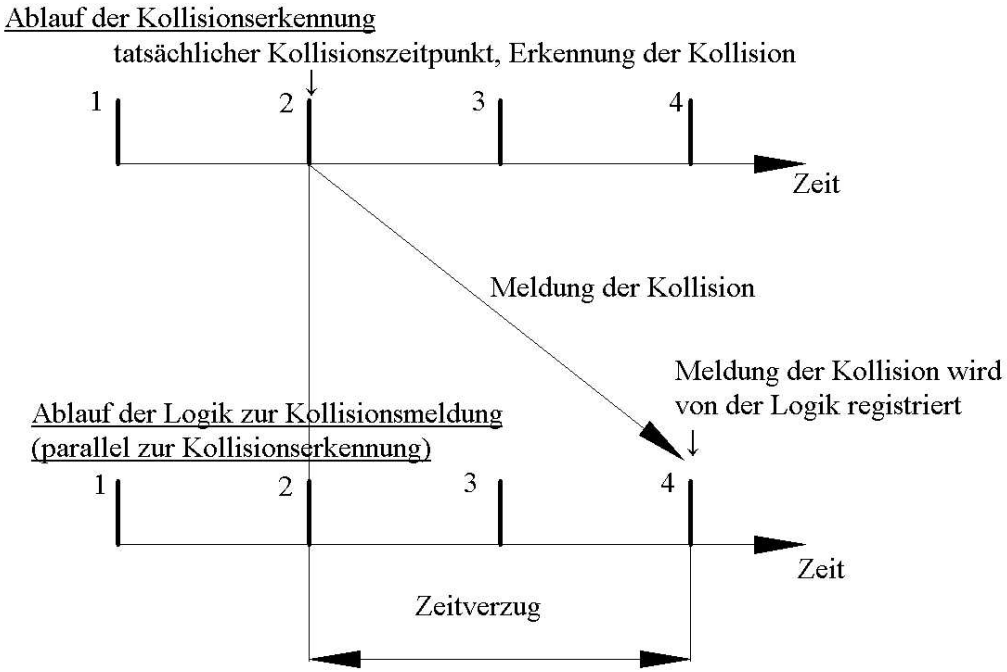

Bild 5.9 Verzögerung der Kollisionsmeldung in Centric

Um den sich daraus ergebenden Fehler klein zu halten oder überhaupt auszuschalten, sollte die Abtastfrequenz nicht höher als die *Frame Rate* (Maximale von Centric errechenbare Anzahl der Bilder pro Sekunde, wird am unteren Rand des *3D Windows* angezeigt) der jeweiligen Centric Simulation sein (Höhere Abtastfrequenzen sind ohnehin nicht sinnvoll, da die Obergrenze für die Simulationsgeschwindigkeit durch die Berechnungszeit von Centric für einen Einzelschritt vorgegeben ist) (Bild 5.8). Eine weitere Möglichkeit zur Verbesserung der Genauigkeit ist die Verringerung der Ausgabezeitschrittweite, da der Fehler, wenn er bei einer bestimmten Abtastfrequenz auftritt, ein „Nacheilen“ der Kollisionsanzeige um einen oder mehrere Schritte bewirkt und eine Verringerung der Schrittweite eines Einzelschritts damit eine Verringerung des absoluten Fehlers bewirkt.

5.1.2.6 Demonstration einer einfachen Parameterstudie anhand der Simulation des gedämpften Pendels

Als einfaches Beispiel für eine Parameteroptimierung wurde das vorher definierte Pendel gewählt. Ziel war es, die Dämpfung so zu verändern, dass keine Kollision des Pendels mit dem Würfel mehr auftritt, der vorhandene Raum aber optimal genutzt wird. Die Ermittlung des entsprechenden Dämpfungswertes wurde mittels *Trial and Error*-Verfahren ermittelt.

Als Ausgangspunkt wurde der Dämpfungswert von 5 Nms/rad (D=0.031) verwendet. Dabei kommt es zu einer nur schwach gedämpften Schwingung, die zu einer Kollision führt, bei der der Würfel komplett durchdrungen wird (Bild 5.10).

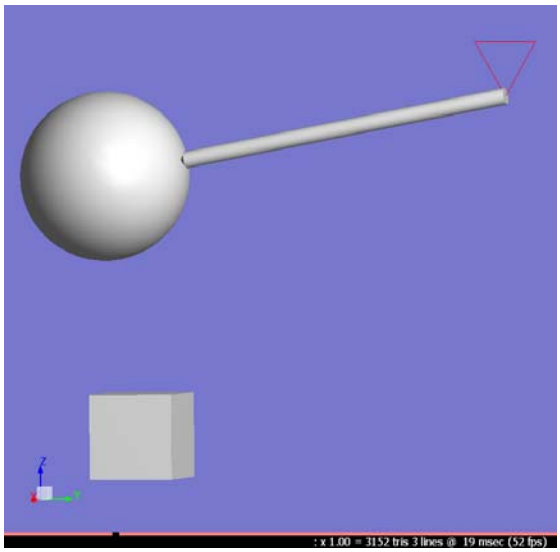

Bild 5.10 Ausgangssituation mit Dämpfungswert 5 Nms/rad – oberer Umkehrpunkt bei erster Schwingung

Mit zunehmender Dämpfung nimmt die Eindringtiefe ab, so dass sich als endgültiges Ergebnis beim fünften Iterationsschritt eine Dämpfung von 39 Nms/rad (D=0.241) ergibt. Damit ergibt sich folgende Schwingung, bei der keine Kollision mehr auftritt und der zur Verfügung stehende Raum gut ausgenutzt wird (Bild 5.11):

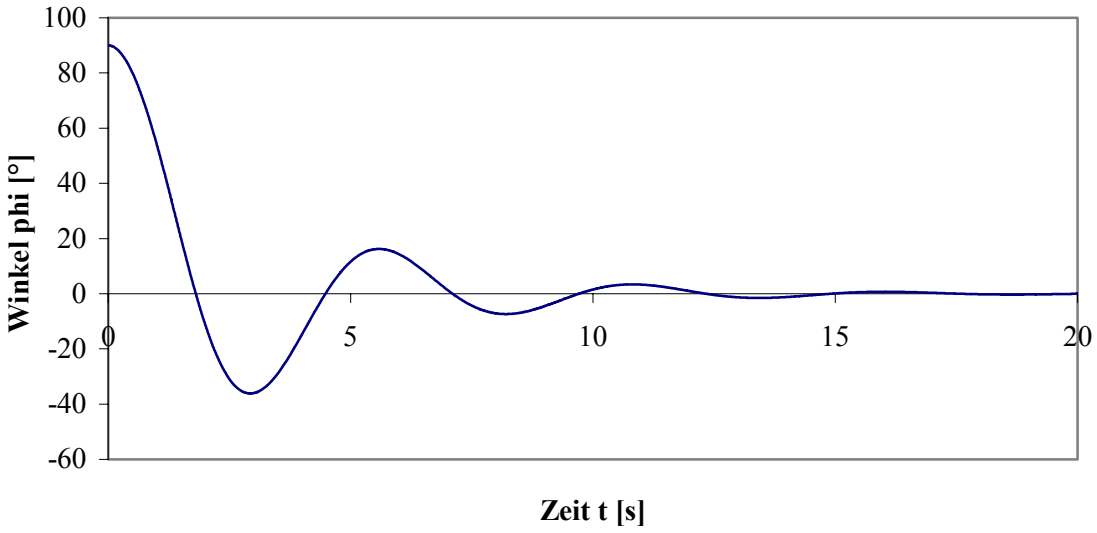

Bild 5.11 Schwingung des Pendels bei endgültigem Dämpfungswert

Insgesamt konnte dieses Ergebnis, trotz der unsystematischen Vorgangsweise, innerhalb von etwa 30 Minuten erzielt werden. Dies stellt verglichen mit anderen Lösungsmöglichkeiten ein gutes Ergebnis dar.

5.2 Aufbau eines vereinfachten Modells eines Triebzugendwagens in SIMPACK

Um eine Vorgangsweise für die Kollisionsprüfung bei Schienenfahrzeugen zu erarbeiten, wurde ein Modell eines Triebzugs in SIMPACK aufgebaut. Die Parameter sind an den Triebzug „Siemens Desiro UK“ angelehnt und wurden für diese Diplomarbeit von Siemens SGP Verkehrstechnik zur Verfügung gestellt. Dieser Triebzug ist mit Drehgestellen vom Typ „SF 5000 UK-TDG“ ausgestattet (Bild 5.12).

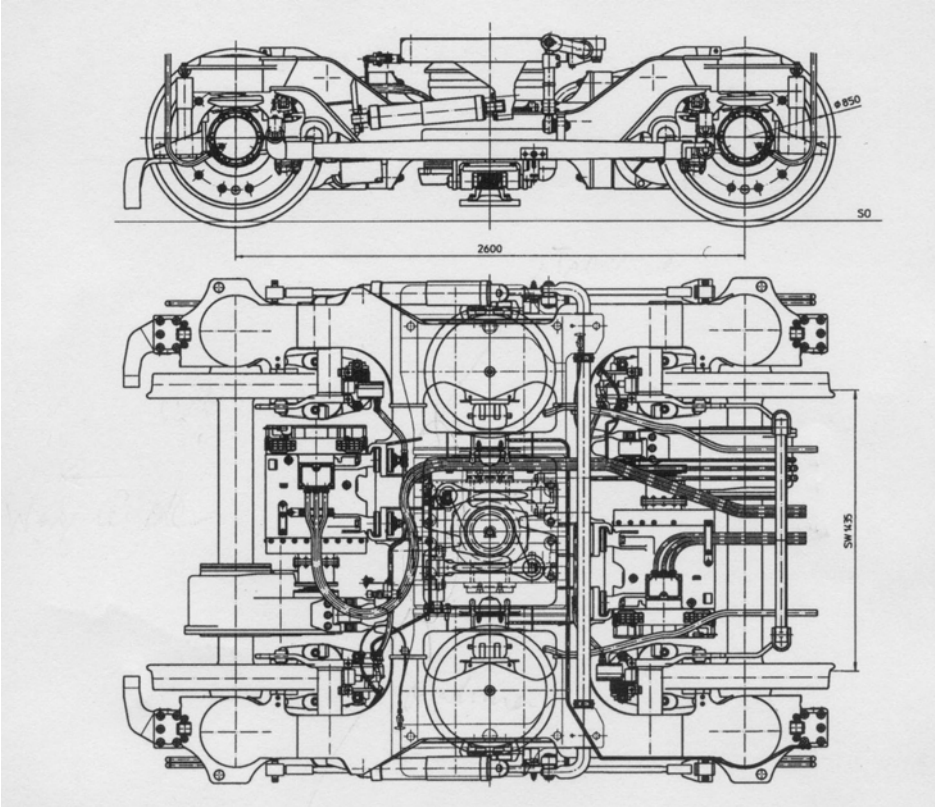

Bild 5.12 Drehgestell SF 5000 UK-TDG [28]

Technische Daten SF 5000 UK-TDG [28]

Betriebsgeschwindigkeit	160 km/h
Maximale Radsatzlast	16,5 to
Radsatzstand	2600 mm
Spurweite	1435 mm
Raddurchmesser neu/abgenutzt	850/786 mm
Minimaler Bogenradius im Betrieb	120 m
Eigenmasse	ca. 8480 kg

Um die Komplexität des Modells zu verringern, wurden die einzelnen Körper als Starrkörper definiert. Dadurch werden Einflüsse z. B. durch Biegeschwingungen des Wagenkastens oder Torsion des Wagenkastens bei Einfahrt in überhöhte Kurven nicht

berücksichtigt. Für die Untersuchung der Vorgangsweise bei einer dynamischen Geometriesimulation zur Kollisionsuntersuchung ist dieses Modell aber ausreichend.

Grundsätzlich kann das Modell in drei Hauptgruppen unterteilt werden:

- **Ungefederte Massen** (Radsätze, Anteile der Getriebe und Anteile der Radsatzführungsschwingen)
- **Primär gefederte Massen** (Drehgestellrahmen, Fahrmotore, Anteile der Getriebe und der Radsatzführungsschwingen)
- **Sekundär gefederte Massen** (Wagenkasten, Traversen)

Diese Gruppen sind untereinander durch Gelenke (*Joints*), Zwangsbedingungen (*Constraints*), wie sie bei geschlossenen Schleifen auftreten, und Feder-Dämpfer-Elemente verbunden.

Die in SIMPACK gebotene Möglichkeit, mehrmals verwendete Teile als *Substructures* zu erstellen und dann in das Hauptmodell einzubauen, wurde für die Drehgestelle genutzt.

5.2.1 Aufbau des Drehgestells

Das Drehgestell besteht aus den Körpern Drehgestellrahmen, Fahrmotoren, Getrieben, Radsatzführungsschwingen, Radsätzen und Traverse (Bild 5.13, Bild 5.14).

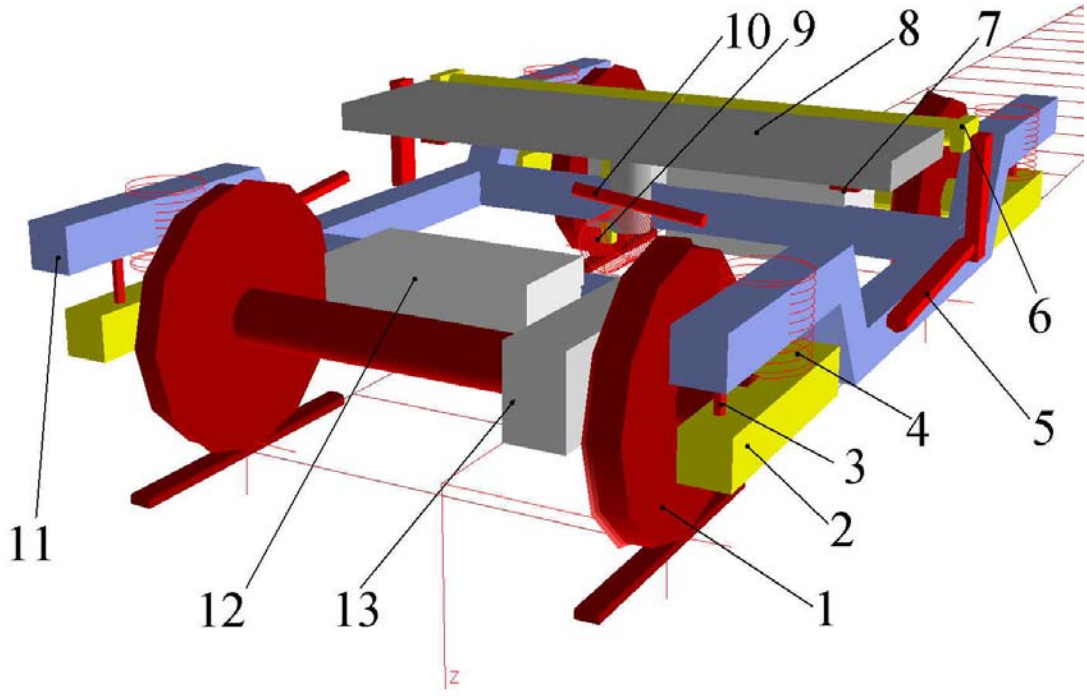

Bild 5.13 SIMPACK Modell des Drehgestells
(1) Radsatz, (2) Radsatzführungsschwinge, (3) Primärdämpfer, (4) Primärfeder, (5) Schlingerdämpfer, (6) Wankstabilisator, (7) Sekundärfeder, (8) Traverse mit Drehzapfen, (9) Längsmitnahme, (10) Querdämpfer, (11) Drehgestellrahmen, (12) Motor, (13) Getriebe

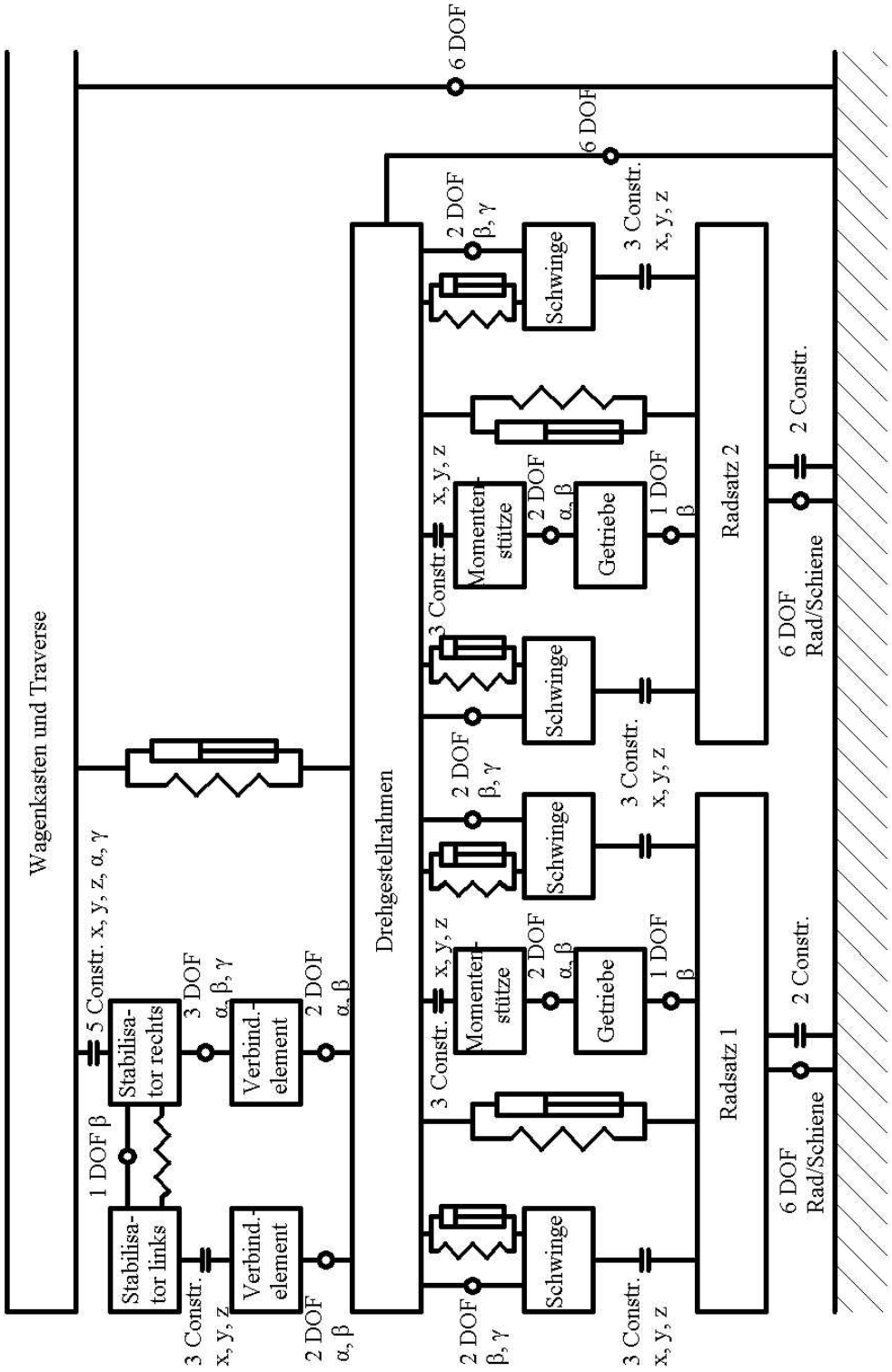

Bild 5.14 Struktur der SIMPACK Modellierung des Drehgestells

Gelenk mit Angabe der Freiheitsgrade, Constraint mit Angabe der gesperrten Bewegungen, Rad-Schiene Kontakt mit Joint Type 07 (s. Kap. 3.7.2)

5.2.1.1 Radsatz

Der Radsatz (Bild 5.15) wird je nach Lage im Drehgestell mit 1 (vorne) oder 2 (hinten) bezeichnet. Der Radsatz besteht aus der Radsatzachse *wheelset1(2)_axle* und dem von SIMPACK bereitgestellten *Joint Type 07: General Wheel/Rail Joint* mit sechs Freiheitsgraden. Die Räder werden von diesem *Wheel-Rail-Joint* automatisch generiert und sind masselos. Die Masse und die Trägheitsmomente der Räder müssen daher bereits in der Radsatzachse enthalten sein.

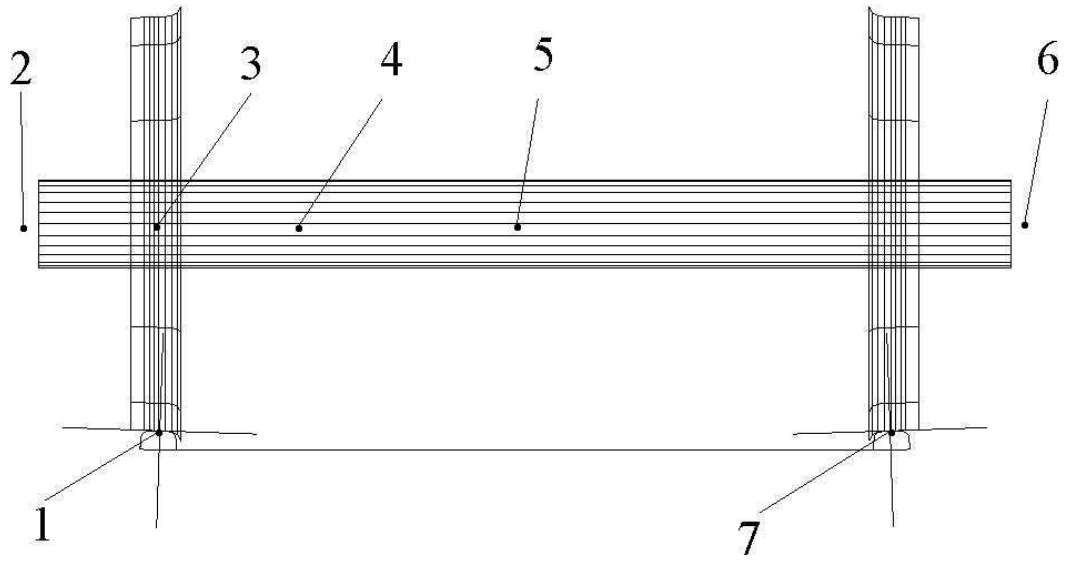

Bild 5.15 Vorderer Radsatz in SIMPACK

Auf dem Radsatz sind folgende *Marker* definiert:

Nr.	Bezeichnung	Funktion
1	*$M_wheelset1_axle_Contact_Left*	Wird von SIMPACK automatisch für den *Wheel-Rail-Joint* generiert
2	*$M_wheelset1_axle_left*	Einbauposition des linken Radsatzlagers
3	*$M_wheelset1_axle_ProfRef_Left*	Wird von SIMPACK automatisch für den *Wheel-Rail-Joint* generiert
4	*$M_wheelset1_axle_gearbox*	Radsatzseitiger Abstützpunkt des Getriebes
5	*$M_wheelset1_axle*	Körperfestes Referenzsystem, wird von

		SIMPACK beim Definieren eines Körpers automatisch erstellt
6	*$M_wheelset1_axle_ProfRef_Right*	Wird von SIMPACK automatisch für den *Wheel-Rail-Joint* generiert
7	*$M_wheelset1_axle_right*	Einbauposition des rechten Radsatzlagers
8	*$M_wheelset1_axle_Contact_Right*	Wird von SIMPACK automatisch für den *Wheel-Rail-Joint* generiert

5.2.2.2 Aufbau des Drehgestellrahmens mit Radsatzführungsschwingen, Primärfedern und -dämpfern

Der Drehgestellrahmen ist als Starrkörper modelliert (Bild 5.16). Am Drehgestellrahmen sind die Radsatzführungsschwingen mit einem *Joint Type 13: Universal Joint y-z* befestigt. Dieses Gelenk erlaubt eine Rotation um die y- und die z-Achse.

Die Befestigung des Radsatzes an den Radsatzführungsschwingen bildet eine geschlossene Schleife (siehe Bild 5.14). Die Modellierung der Radsatzlager erfolgt daher mit *Constraints*. Verwendet werden *Constraints* vom *Typ 10: Spherical*, die Rotationen um x-, y- und z-Achse erlauben und Translationen um alle Achsen sperren.

Die Primärfederstufe zwischen Radsatzführungsschwinge und Drehgestellrahmen wird durch Kraftelemente vom *Typ 01* bzw. *02* (*Spring Point to Point* bzw. *Damper Point to Point*) nachgebildet. Primärfedern und –dämpfer haben jeweils unterschiedliche Angriffspunkte.

Die Fahrmotore werden durch einen *Joint Type 00: 0 Degrees of Freedom* am Drehgestellrahmen befestigt. Diese Gelenke erlauben keine Bewegung zwischen Motor und Drehgestellrahmen, die Elastizität der Motoraufhängung wird somit im Zuge der Vereinfachung nicht berücksichtigt.

Die Getriebe stützen sich an einem Ende am Radsatz ab und werden mit dem Radsatz durch einen *Joint Type 02: Revolute Joint be* verbunden. Dieses Gelenk erlaubt eine Drehbewegung um die y-Achse zwischen Getriebe und Radsatz. Am anderen Ende erfolgt die Abstützung über ein mit geringer Masse versehenes Verbindungselement am Drehgestellrahmen. Dabei wird die Verbindung zwischen Getriebe und Verbindungselement durch einen Joint *Type 12: Universal Joint x-y* hergestellt, der Rotationen um die x- und y-Achse freigibt. Mit der Verbindung des anderen Endes dieser Getriebe-Drehmomentstütze mit dem Drehgestellrahmen tritt eine geschlossene Schleife auf (siehe auch Bild 5.14). Diese Schleife wird mit einer *Constraint* vom *Type 25: User Defined Constraint* geschlossen. Mit dieser Zwangsbedingung werden Rotation um x-, y- und z-Achse freigegeben und alle Translationen gesperrt.

Die Verbindung des Drehgestellrahmens selbst mit dem Inertialsystem erfolgt mit einem *Joint Type 07: General Wheel/Rail Joint* mit sechs Freiheitsgraden. Im Gegensatz zum Radsatz wird hier aber kein Rad/Schiene-Kontakt generiert.

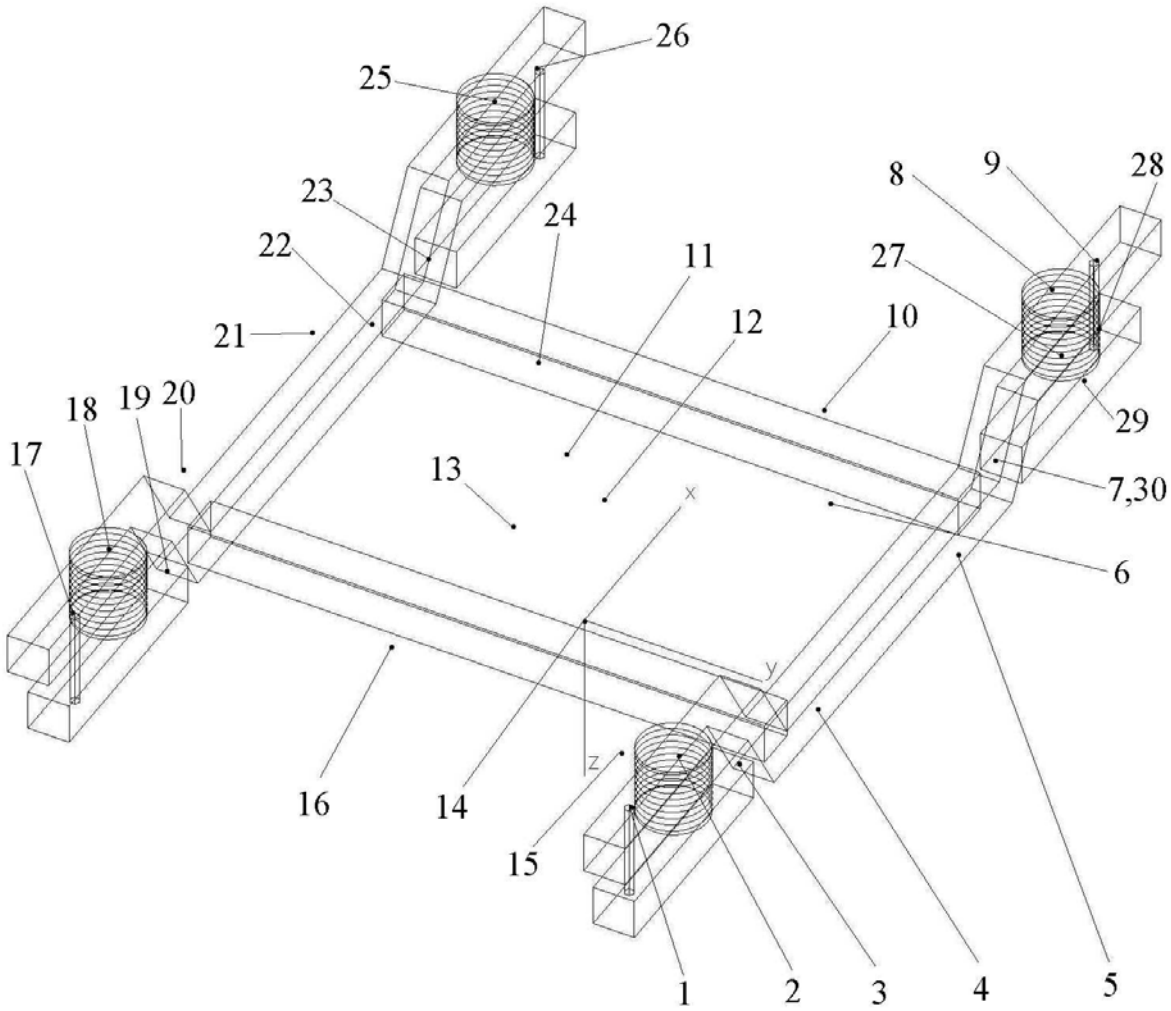

Bild 5.16 Drehgestellrahmen mit Radsatzführungsschwingen und Primärfedern und -dämpfern in SIMPACK

Um die Anbindung von Kraftelementen, Gelenken und Zwangsbedingungen zu ermöglichen, wurden die Körper mit folgenden Markern versehen:

Nr.	Bezeichnung	Funktion
1	*$M_bogie_frame_prim_damp_br*	Befestigungspunkt des Primärdämpfers am Drehgestellrahmen, hinten rechts
2	*$M_bogie_frame_prim_susp_br*	Befestigungspunkt der Primärfeder am Drehgestellrahmen, hinten rechts
3	*$M_bogie_frame_bbr*	Befestigungspunkt der Radsatzführungsschwinge am Drehgestellrahmen, hinten rechts

4	*$M_bogie_frame_schlingerd_right*	Befestigungspunkt des rechten Schlingerdämpfers am Drehgestellrahmen
5	*$M_bogie_frame_stabi_right*	Befestigungspunkt des Wankstabilisator-Verbindungselements am Drehgestellrahmen, rechts
6	*$M_bogie_frame_sek_susp_right*	Position der rechten Sekundärfeder
7	*$M_bogie_frame_bfr*	Befestigungspunkt der Radsatzführungsschwinge am Drehgestellrahmen, vorne rechts
8	*$M_bogie_frame_prim_susp_fr*	Befestigungspunkt der Primärfeder am Drehgestellrahmen, vorne rechts
9	*$M_bogie_frame_prim_damp_fr*	Befestigungspunkt des Primärdämpfers am Drehgestellrahmen, vorne rechts
10	*$M_bogie_frame_engine1*	Befestigungspunkt des vorderen Fahrmotors am Drehgestellrahmen
11	*$M_bogie_frame_sek_susp_lat*	Befestigungspunkt des Queranschlags am Drehgestellrahmen
12	*$M_bogie_frame_querd*	Befestigungspunkt des Querdämpfers am Drehgestellrahmen
13	*$M_bogie_frame_laengsmit*	Angriffspunkt der Längsmitnahme am Drehgestellrahmen
14	*$M_bogie_frame*	Körperfestes Referenzsystem des Drehgestellrahmens, von SIMPACK automatisch generiert
15	*$M_bogie_frame_gearbox2_mount*	Befestigungspunkt der Drehmomentstütze des hinteren Getriebes am Drehgestellrahmen
16	*$M_bogie_frame_engine2*	Befestigungspunkt des hinteren Fahrmotors am Drehgestellrahmen
17	*$M_bogie_frame_prim_damp_bl*	Befestigungspunkt des Primärdämpfers am Drehgestellrahmen, hinten links
18	*$M_bogie_frame_prim_susp_bl*	Befestigungspunkt der Primärfeder am Drehgestellrahmen, hinten links
19	*$M_bogie_frame_bbl*	Befestigungspunkt der Radsatzführungsschwinge am Drehgestellrahmen, hinten links
20	*$M_bogie_frame_schlingerd_left*	Befestigungspunkt des linken Schlingerdämpfers am Drehgestellrahmen
21	*$M_bogie_frame_stabi_left*	Befestigungspunkt des Wankstabilisator-Verbindungselements am Drehgestellrahmen, links
22	*$M_bogie_frame_sek_susp_left*	Position der linken Sekundärfeder
23	*$M_bogie_frame_bfl*	Befestigungspunkt der

		Radsatzführungsschwinge am Drehgestellrahmen, vorne links
24	*$M_bogie_frame_gearbox1_mount*	Befestigungspunkt der Drehmomentstütze des vorderen Getriebes am Drehgestellrahmen
25	*$M_bogie_frame_prim_susp_fl*	Befestigungspunkt der Primärfeder am Drehgestellrahmen, vorne links
26	*$M_bogie_frame_prim_damp_fl*	Befestigungspunkt des Primärdämpfers am Drehgestellrahmen, vorne links
27	*$M_schwinge_fr_prim_susp*	Befestigungspunkt der Primärfeder an der Radsatzführungsschwinge, vorne rechts
28	*$M_schwinge_fr_prim_damp*	Befestigungspunkt des Primärdämpfers an der Radsatzführungsschwinge, vorne rechts
29	*$M_schwinge_fr_bearing*	Position des Radsatzlagers an der Radsatzführungsschwinge
30	*$M_schwinge_fr_frame_bearing*	Befestigungspunkt der Radsatzführungsschwinge am Drehgestellrahmen

(An den drei anderen Radsatzführungsschwingen gleiche *Marker* mit *fl* für vorne links, *bl* für hinten links und *br* für hinten rechts)

5.2.2.3 Traverse und Drehzapfen

Die Traverse ist das Verbindungselement zwischen Drehgestell und Wagenkasten. Die Traverse ist mit dem Wagenkasten über ein Null-Freiheitsgrad-Gelenk fest verbunden. Die Verbindung mit dem Drehgestellrahmen wird über die Elemente der Sekundärfederstufe und die Längsmitnahme hergestellt (Bild 5.17).
Die Längsmitnahme überträgt die Längskräfte zwischen Traverse und Drehgestellrahmen und wurde als Kraftelement *Typ 04: Spring-Damper Parallel Point to Point* modelliert.
Die Vertikalkräfte werden von den Sekundärfedern aufgenommen, die als Kraftelemente vom Typ *05: Spring-Damper Parallel Component* modelliert wurden.
Zusätzlich sind Schlingerdämpfer (modelliert als Kraftelement *Typ 02: Damper Point to Point*), Querdämpfer (*Typ 02: Damper Point to Point*) und ein Queranschlag vorhanden. Der Queranschlag begrenzt die Querverschiebung zwischen Traverse und Drehgestellrahmen durch eine ab einem bestimmten Verschiebungswert einsetzende, progressive Federwirkung und einem Anschlag bei Erreichen des maximal zulässigen Verschiebungswerts. Der Queranschlag wird grafisch nicht dargestellt und ist daher in (Bild 5.17) nicht zu sehen.
Die Querkräfte werden nur durch die Quersteifigkeiten der Sekundärfedern und, wenn eine bestimmte Querverschiebung überschritten wird, vom Queranschlag aufgenommen.

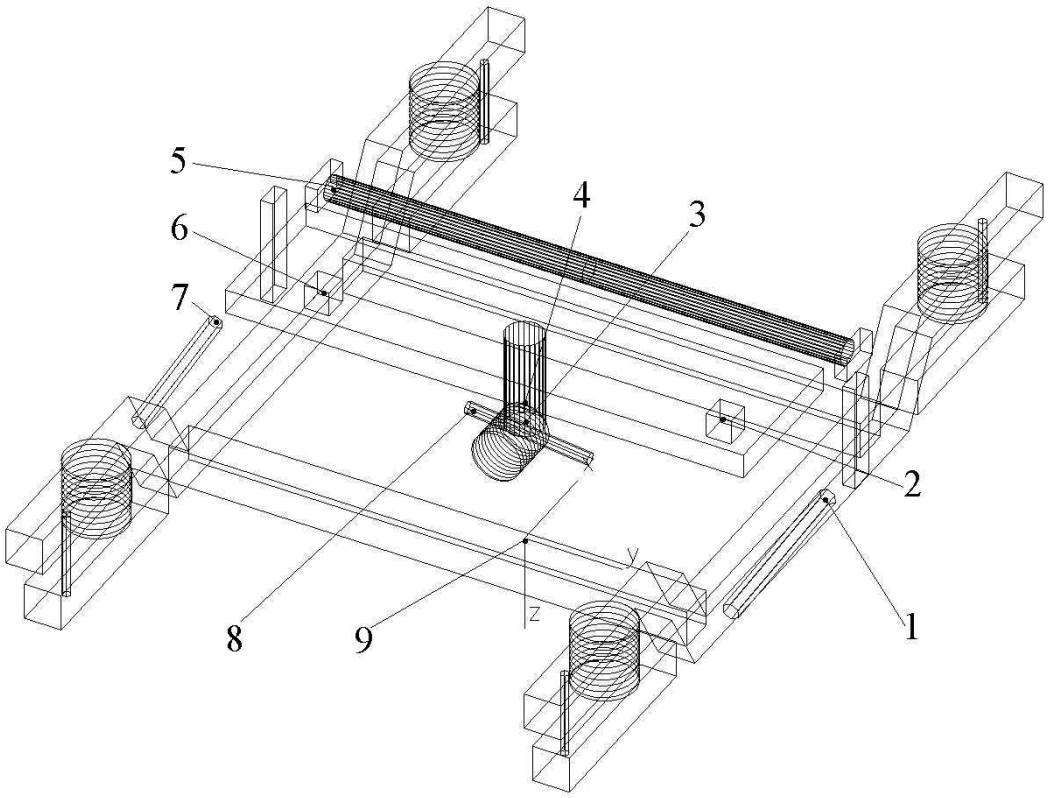

Bild 5.17 Traverse und Wankstabilisator eingebaut im SIMPACK-Modell

Die Anbindung der Kraftelemente und des Wankstabilisators an die Traverse wird an folgenden Markern hergestellt:

Nr.	Bezeichnung	Funktion
1	*$M_traverse_schlingerd_right*	Befestigungspunkt des rechten Schlingerdämpfers an der Traverse
2	*$M_traverse_sek_susp_right*	Einbauposition rechte Sekundärfeder
3	*$M_traverse_laengsmit*	Traversenseitiger Befestigungspunkt der Längsmitnahme
4	*$M_traverse_sek_susp_lat*	Einbauposition des Queranschlags
5	*$M_traverse_stabi_left*	Befestigunspunkt des Stabilisators an der Traverse
6	*$M_traverse_sek_susp_left*	Einbauposition linke Sekundärfeder
7	*$M_traverse_schlingerd_left*	Befestigungspunkt des linken Schlingerdämpfers an der Traverse
8	*$M_traverse_querd*	Befestigungspunkt des Querdämpfers an der Traverse

9	*$M_traverse*	Körperfestes Referenzsystem der Traverse, wird von SIMPACK automatisch generiert

5.2.2.4 Wankstabilisator

Der Drehstab des Wankstabilisators wurde aus zwei Starrkörpern modelliert, die durch ein Federelement verbunden sind. Die Verbindung zwischen den jeweiligen Hälften und dem Drehgestellrahmen wird über je ein Verbindungselement pro Seite hergestellt (Bild 5.18).
Die Verbindung zwischen Drehgestellrahmen und diesen Verbindungselementen erfolgt über einen *Joint Type 12: Universal Joint x-y*, der eine Rotation um die x- und die y-Achse zulässt.
Die beiden Hälften des Stabilisators sind durch einen *Joint Type 02: Revolute Joint be* verbunden, der eine Rotation um die y-Achse freigibt.
Die Verbindung zwischen (in Fahrtrichtung gesehen) rechtem Verbindungselement und rechter Drehstabhälfte wird durch einen *Joint Type 10: Spherical Joint x-y-z* hergestellt, der eine Rotation um alle drei Achsen erlaubt.
Die Schleife (siehe auch Bild 5.14) wurde zwischen dem oberen Ende des linken Verbindungselements und der linken Drehsstabhälfte durch eine *Constraint Type 25: User Defined Constraint* geschlossen, die die Rotation um alle drei Achsen freigibt und die Translationen sperrt.
Die Verbindung des Stabilisators mit der Traverse erfolgt mit einer *Constraint Type 25: User Defined Constraint*, die die Rotation um die y-Achse freigibt und alle anderen Bewegungsmöglichkeiten sperrt.

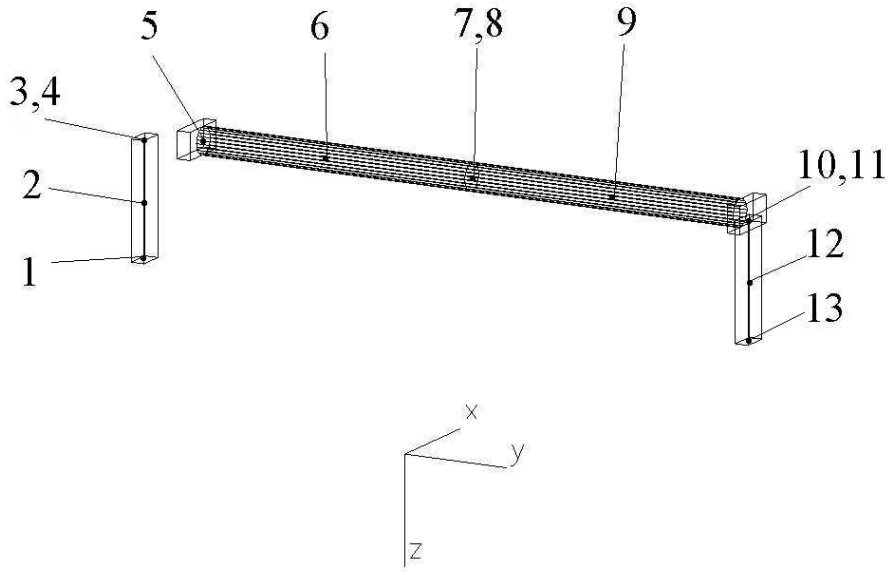

Bild 5.18 Wankstabilisator in SIMPACK

Die Anbindung der Kraftelemente und Gelenke erfolgt an folgenden *Markern*:

Nr.	Bezeichnung	Funktion
1	*$M_stabi_left_link_lower*	Befestigungspunkt des linken Verbindungselements am Drehgestellrahmen
2	*$M_stabi_left_link*	Körperfestes Referenzsystem des linken Verbindungselements, von SIMPACK generiert
3	*$M_stabi_left_link_upper*	Verbindung zwischen linkem Verbindungselement und linker Drehstabhälfte
4	*$M_stabi_left_arm*	Verbindung zwischen linkem Verbindungselement und linker Drehstabhälfte
5	*$M_stabi_left_front*	Verbindung zwischen linker Drehstabhälfte und Traverse
6	*$M_stabi_left*	Körperfestes Referenzsystem des linken Drehstabes, von SIMPACK generiert
7	*$M_stabi_left_back*	Einbaupunkt der Feder und des Gelenks zwischen linker und rechter Drehstabhälfte
8	*$M_stabi_right_back*	Einbaupunkt der Feder und des Gelenks zwischen linker und rechter Drehstabhälfte
9	*$M_stabi_right*	Körperfestes Referenzsystem des rechten Drehstabes, von SIMPACK generiert
10	*$M_stabi_right_bearing*	Verbindung zwischen rechtem Verbindungselement und rechter Drehstabhälfte
11	*$M_stabi_right_link_upper*	Verbindung zwischen linkem Verbindungselement und linker Drehstabhälfte
12	*$M_stabi_right_link*	Körperfestes Referenzsystem des rechten Verbindungselements, von SIMPACK generiert
13	*$M_stabi_right_link_lower*	Befestigungspunkt des rechten Verbindungselements am Drehgestellrahmen

5.2.2.5 Fahrmotor und Getriebe

Der Fahrmotor wurde zur Vereinfachung mit einem Null-Freiheitsgrad-Gelenk mit dem Drehgestellrahmen verbunden (Bild 5.19).
Das Getriebe stützt sich auf einer Seite am Radsatz ab und ist auf der anderen Seite über ein Verbindungselement mit dem Drehgestellrahmen verbunden (siehe auch Bild 5.14 und Kapitel 5.2.2.2). Die Verbindung mit dem Radsatz erfolgt mit einem Gelenk, das eine Drehbewegung um die y-Achse ermöglicht. Die Verbindung mit dem Drehgestellrahmen wird über ein Verbindungselement hergestellt, wobei zwischen Getriebe und Verbindungselement ein Gelenk, das die Rotation um x- und y-Achse ermöglicht, eingebaut ist und zwischen Verbindungselement und Drehgestellrahmen die entstehende Schleife mit einer Zwangsbedingung, die Translationsbewegungen in allen drei Achsen sperrt, geschlossen wird.

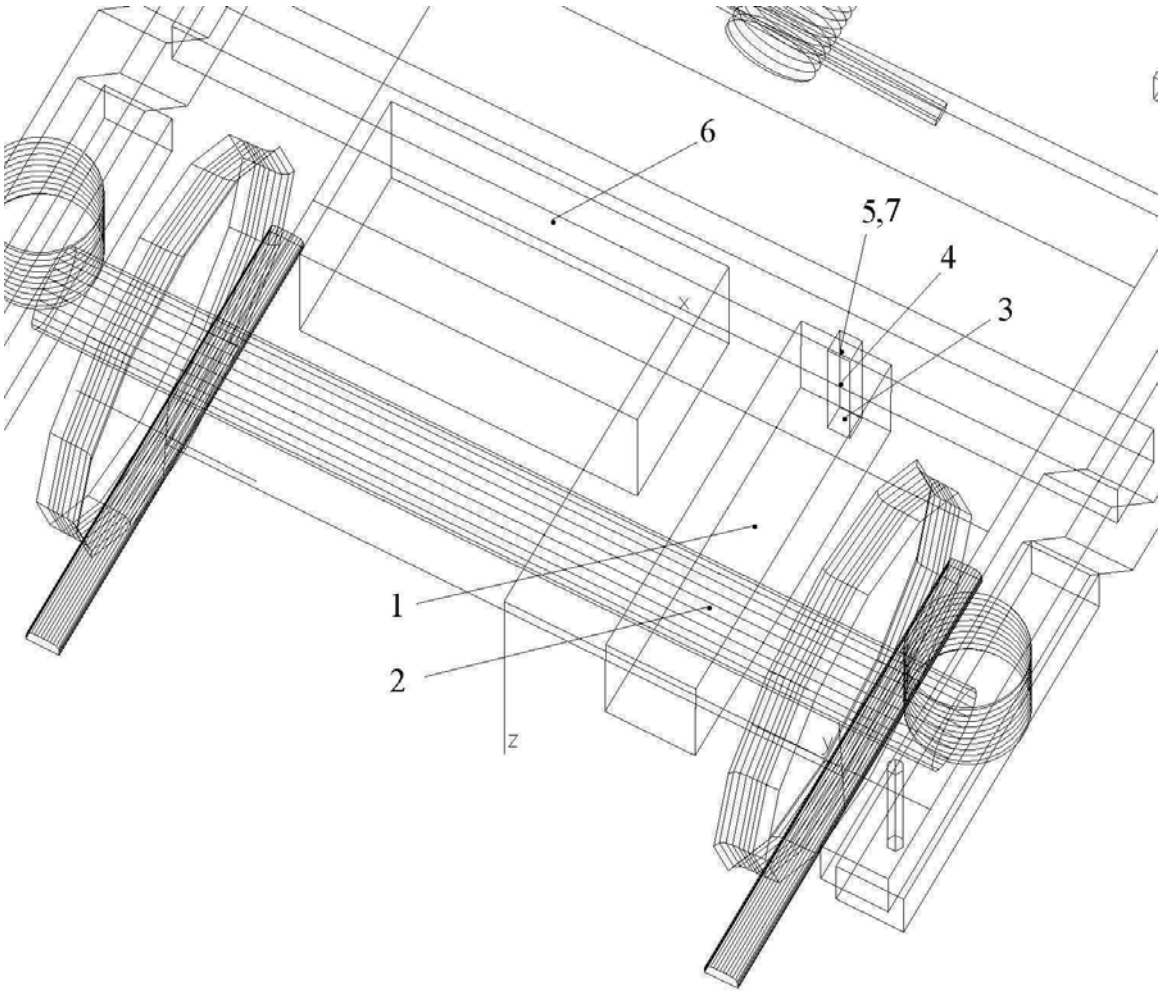

Bild 5.19 Einbau von Motor und Getriebe im Drehgestell

Nr.	Bezeichnung	Funktion
1	*$M_gearbox2*	Körperfestes Referenzsystem des hinteren Getriebes, von SIMPACK generiert
2	*$M_gearbox2_axle*	Befestigungspunkt des hinteren Getriebe am Radsatz
3	*$M_gearbox2_link_lower_mount*	Befestigungspunkt zwischen Drehmomentstütze und Drehgestellrahmen
4	*$M_gearbox2_link*	Körperfestes Referenzsystem der hinteren Drehmomentstütze, von SIMPACK generiert
5	*$M_gearbox2_mount*	Verbindung zwischen Getriebe und Drehmomentstütze
6	*$M_engine2*	Körperfestes Referenzsystem des hinteren Motors und Verbindungspunkt zwischen Motor und Drehgestellrahmen, von SIMPACK generiert
7	*$M_gearbox2_link_upper_mount*	Verbindung zwischen Getriebe und Drehmomentstütze

5.2.2 Gesamtfahrzeug

Das Gesamtfahrzeug besteht aus dem Wagenkasten und zwei Drehgestellen, wobei das vordere Drehgestell gegenüber dem zuvor beschriebenen hinteren Drehgestell um 180° um die z-Achse gedreht ist (Bild 5.20).

Die Anbindung des Wagenkastens an die Drehgestelle erfolgt über die Traversen mit einem Null-Freiheitsgrad-Gelenk. Der Wagenkasten ist mit dem Inertialsystem mit einem *Joint Type 07: General Wheel/Rail Joint* mit sechs Freiheitsgraden verbunden, wobei auch hier, wie beim Drehgestellrahmen, kein Rad/Schiene Kontakt generiert wurde.

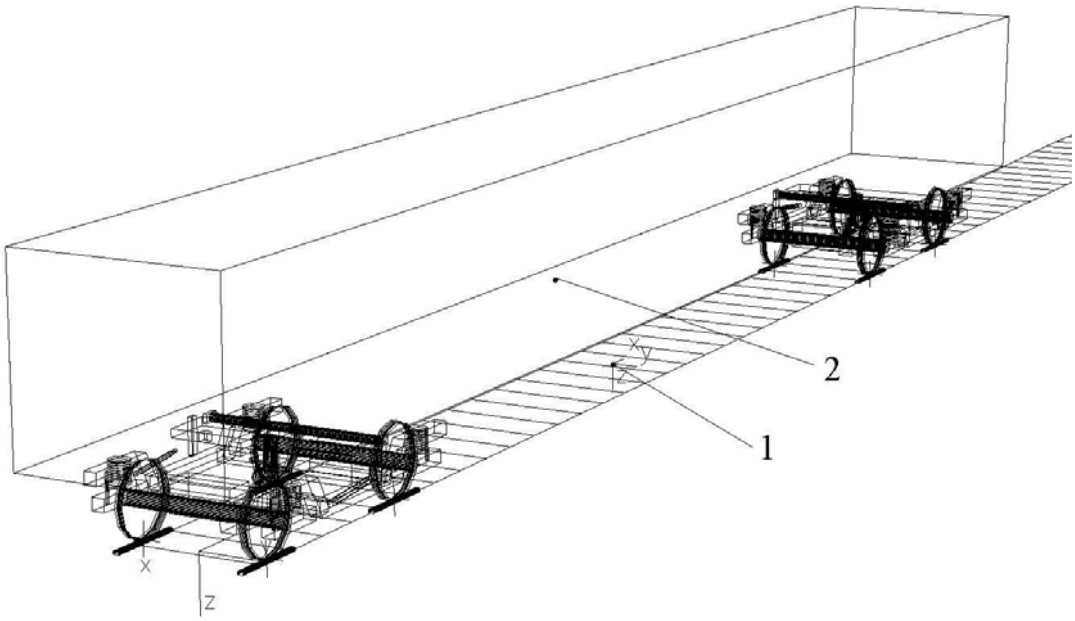

Bild 5.20 Gesamtfahrzeug: (1) körperfestes Referenzsystem des Wagenkastens (2) Schwerpunkt des Wagenkastens

Der Schwerpunkt des Wagenkastens ist gegenüber der Mitte des Wagenkastens geringfügig nach hinten verschoben, da der Wagenkasten unsymmetrisch ist.

5.3 In SIMPACK für die Übergabe an Centric simulierte Fahrszenarien

Für die Übergabe an Centric und die Durchführung einer virtuellen Fahrt des Prototypen mit Kollisionsüberprüfung wurden zwei Fahrten simuliert:

- Geradeausfahrt mit Gleislagestörung
- Einfahrt in eine Kurve ohne Übergangsbogen, um einen extremen Fall zu simulieren

Zusätzlich wurde eine Geradeausfahrt mit Gleislagestörungen simuliert, um in Centric die Darstellung skalierter Elemente (in diesem Fall die Primärfedern) zu erproben.

5.3.1 Geradeausfahrt

Bei der Geradeausfahrt wurde dem Gleis mit *Track-related Excitation* eine Gleislagestörung überlagert. Diese **Gleislagestörung** wird durch folgende Funktion beschrieben:

$$y = 0.02 \cdot \left[sign(s-20) \cdot sign(s-(20+8\cdot\pi)) - 1\right] \cdot \left[\sin\left(0.25\cdot(s-20) - 3/2 \cdot \pi\right) - 1\right] \quad (5.5)$$

y…Gleislage in y-Richtung
s…zurückgelegter Weg in x-Richtung

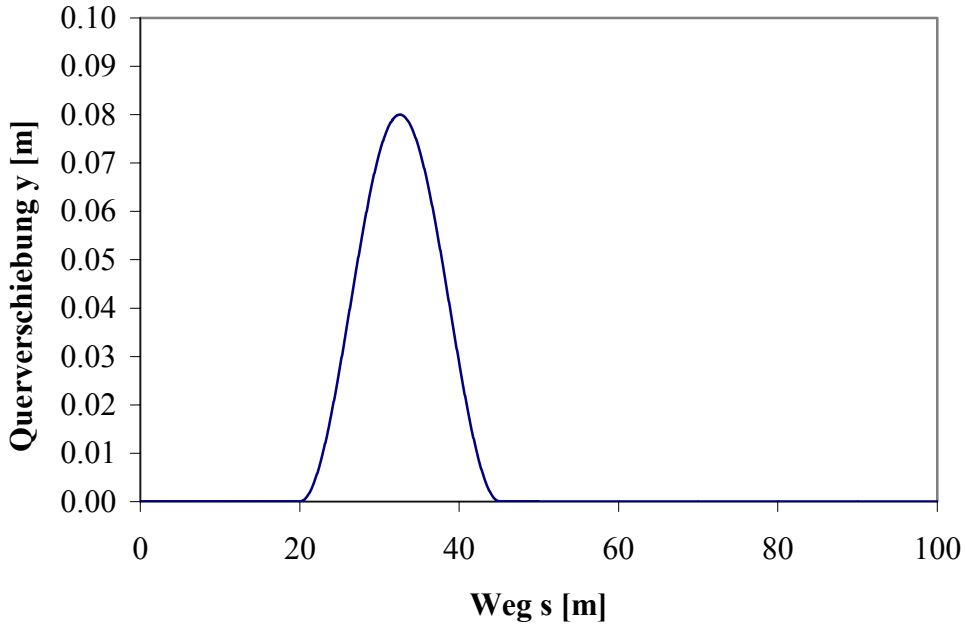

Bild 5.21 Gleislage der in SIMPACK definierten Strecke

Wie man an dem Diagramm in (Bild 5.21) erkennen kann, ist diese Störung nicht realitätsnah und dient nur dazu, um eine starke Wankbewegung des Wagenkastens sowie eine Querverschiebung zwischen Wagenkasten und Drehgestell hervorzurufen und damit in Centric eine Kollision zu provozieren.

Im Weiteren wird der zurückgelegte Weg des Wagenkastenreferenzsystems mit s bezeichnet.

Im *Vehicle Globals* Menü wurde als **Fahrgeschwindigkeit** des Schienenfahrzeuges v=36m/s gewählt. Für das Radprofil wurde S1002, für das Schienenprofil UIC 60 gewählt. Der Reibungskoeffizient μ wurde auf 0.05 herabgesetzt, um ein Befahren dieser Strecke ohne Abheben eines Rades zu ermöglichen.

Der Integrator wurde auf ein Integrationsintervall von 7s entsprechend einer zurückgelegten Strecke von 252m gesetzt. Für dieses Zeitintervall wurde eine Anzahl von 3501 Ausgabezeitschritten festgelegt.

Es ergeben sich für die Position des Wagenkastens folgende Resultate aus der Simulation:

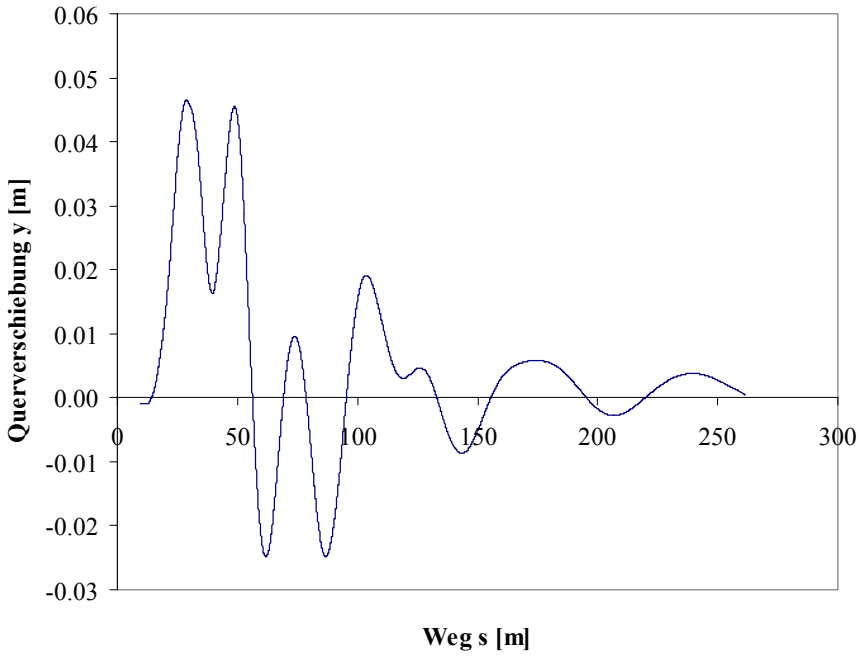

Bild 5.22 Querverschiebung des Wagenkastens zur Gleismitte bei idealer Gleislage (ohne Track Excitations) (Weg s gemessen als Position des Wagenkasten-Referenzsystems)

Der Verlauf der Querverschiebung des Wagenkastens (Bild 5.22) erreicht ihr erstes Maximum bei der Einfahrt des vorderen Drehgestells in den Störungsbereich und ein zweites Maximum bei der Einfahrt des hinteren Drehgestells. Danach stellt sich eine abklingende Schwingung ein. Der Betrag der Querverschiebung ist auch dadurch geringer als der Betrag der Gleislagestörung, dass die Querverschiebung in Wagenkastenmitte gemessen wird und sich dadurch geometrisch eine Verringerung der Verschiebung ergibt.

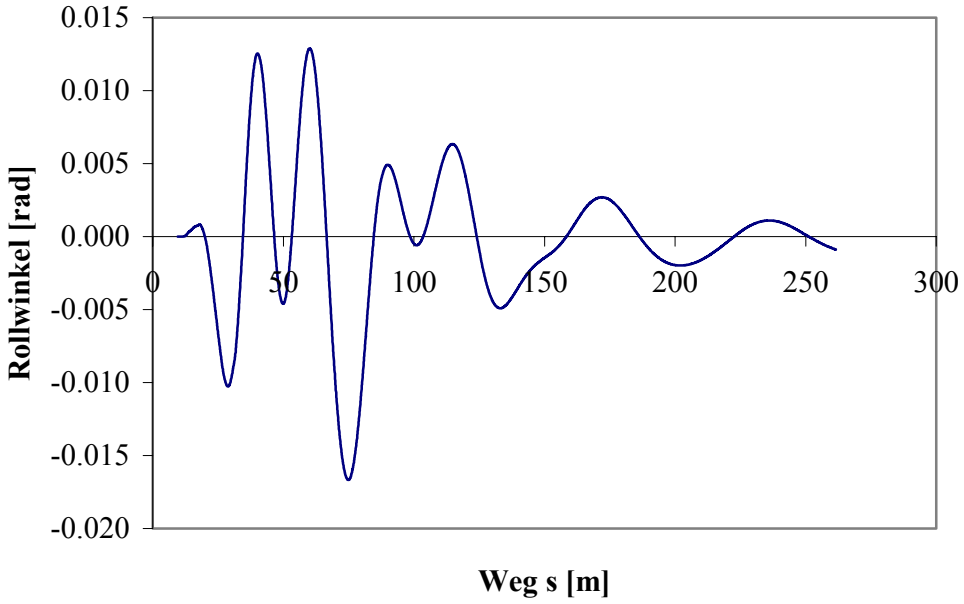

Bild 5.23 Rollwinkel des Wagenkastens (Weg s gemessen als Position des Wagenkasten-Referenzsystems)

Der Rollwinkel (Bild 5.23) fällt zuerst mit der Einfahrt des vorderen Drehgestells in den Störungsbereich ab, steigt dann mit der Ausfahrt aus diesem Bereich an (Gegenschwingung) und fällt mit der Einfahrt des hinteren Drehgestells wieder ab. Danach gibt es bei der Ausfahrt des hinteren Drehgestells aus dem Störungsbereich erneut einen Anstieg und bei der anschließenden Fahrt auf der ungestörten Strecke ein Abklingen der Schwingung.

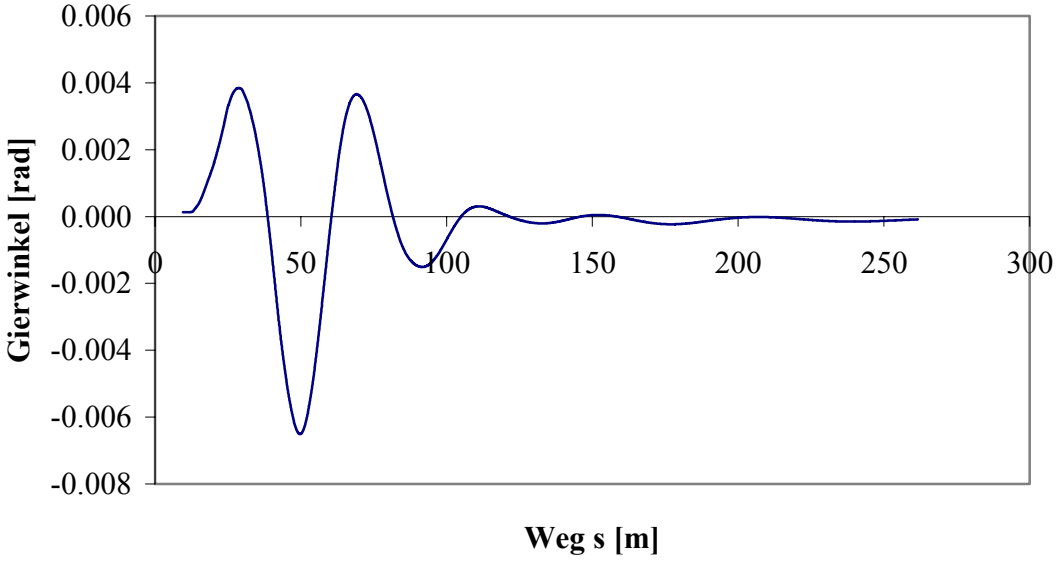

Bild 5.24 Relativer Gierwinkel des Wagenkastens gegenüber dem Streckenverlauf (Weg s gemessen als Position des Wagenkasten-Referenzsystems)

Der Gierwinkel des Wagenkastens (Bild 5.24) steigt zuerst mit der Einfahrt des vorderen Drehgestells in den Bereich mit Gleislagestörung an. Mit dem Verlassen dieses Bereichs und der dann anschließenden Einfahrt des hinteren Bereichs wird ein Minimum erreicht. Danach stellt sich eine abklingende Schwingung ein.

Dabei tritt folgende Querverschiebung zwischen dem Wagenkasten und den Drehgestellen auf (Bild 5.25, Bild 5.26):

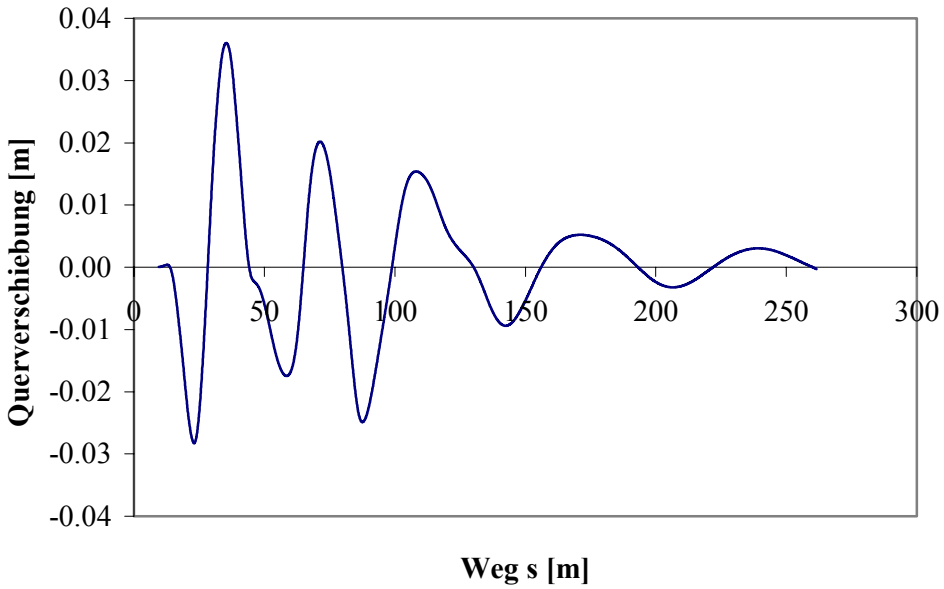

Bild 5.25 Querverschiebung zwischen vorderem Drehgestell und Wagenkasten (Weg s gemessen als Position des Wagenkasten-Referenzsystems)

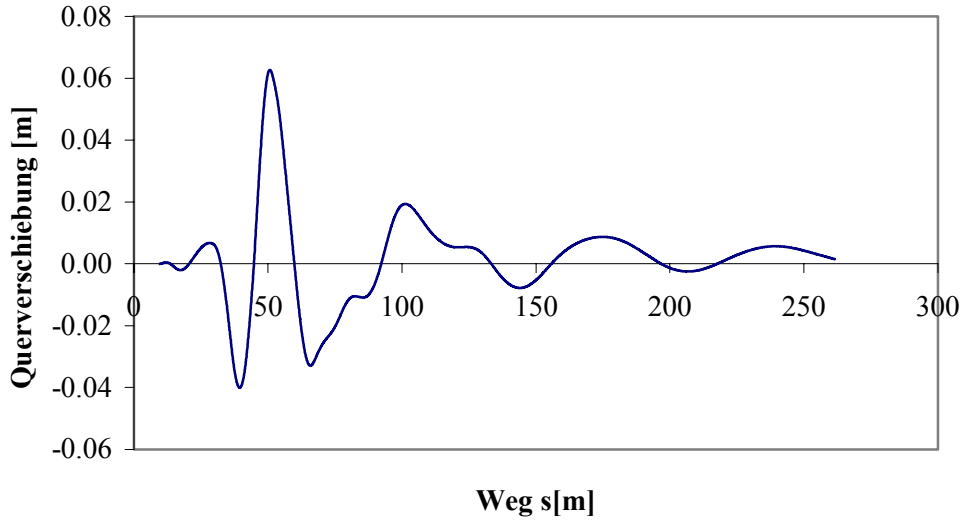

Bild 5.26 Querverschiebung zwischen hinterem Drehgestell und Wagenkasten (Weg s gemessen als Position des Wagenkasten-Referenzsystems)

Mit der Schnittstelle SIMPACK/Centric Innovation wurden die für die Erstellung einer Centric Simulation notwendigen Daten in eine MBS-Datei geschrieben, die dann in eine Centric Innovation Simulation integriert werden konnte (siehe Kapitel 5.4).

5.3.2 Einfahrt in Kurve ohne Übergangsbogen

Als zweites Szenario wurde die Einfahrt in eine Kurve ohne Übergangsbogen gewählt. Auf den Übergangsbogen wurde verzichtet, um einen Extremfall durch den abrupten Anstieg der Krümmung darzustellen. In SIMPACK kann eine Kurveneinfahrt mit vollständigem Verzicht auf den Übergangsbogen nicht definiert werden (siehe dazu Kapitel 3.7.1.1), es wurde daher ein sehr kurzer Übergangsbogen gewählt.

Die **Streckenvorgabe** (Bild 5.27) wurde in SIMPACK mit folgenden Parametern definiert:

- Streckentyp: *Curve Passing* (Gerade – Übergangsbogen – Kreisbogen – Übergangsbogen –Gerade)
- *Ramp*: Linear (gibt den Verlauf des Krümmungsanstiegs im Übergangsbogen an, siehe dazu Kapitel 3.7.1.1)
- Länge der Eingangsgeraden: 29.7m
- Länge des Übergangsbogens: 0.6m
- Länge des Kreisbogens: 502.05481m (entspricht einem Halbkreis, der in Pro/ENGINEER für die Übernahme in Centric definierte Verlauf des Lichtraumprofils besteht aus einer Eingangsgeraden mit einer Länge von 30m und einem Halbkreisbogen – der in SIMPACK notwendige Übergangsbogen wird je zur Hälfte der Geraden und dem Halbkreisbogen zugeschlagen, was den Verlauf sehr gut annähert)
- Radius des Kreisbogens: 160m
- Überhöhung: Keine
- *Half of smoothing length*: 0.2m (Halbe Streckenlänge, auf der eine Glättungsfunktion auf die Krümmung angewandt wird, siehe Kapitel 3.7.1.1)

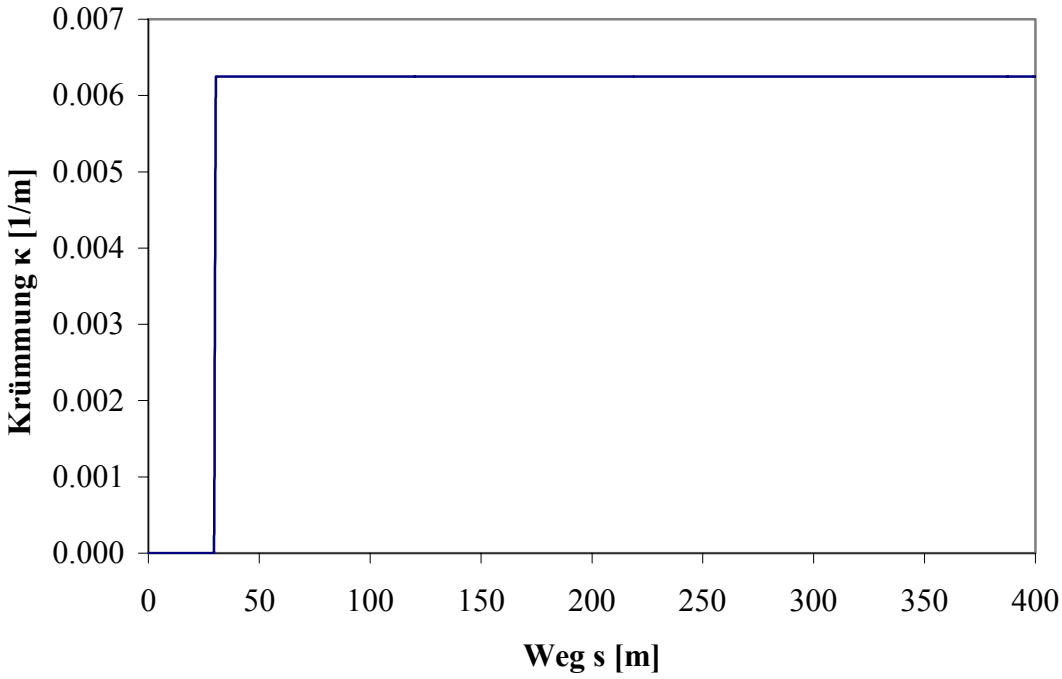

Bild 5.27 Verlauf der Krümmung der Strecke in SIMPACK

Die Strecke wird mit einer **Geschwindigkeit** von v=10m/s durchfahren. Das entspricht im Kurvenabschnitt einer Seitenbeschleunigung von $v^2/R = 0.625 m/s^2$. Eine höhere Geschwindigkeit war wegen des sehr steilen Anstiegs der Krümmung durch den kurzen Übergangsbogen nicht möglich. Die Profilpaarung Rad/ Schiene ist wie zuvor bei der Geradeausfahrt S1002/UIC 60. Der Reibungskoeffizient wurde auch hier auf μ=0.05 herabgesetzt.
Mit einer Integrationszeit von 30s (entsprechend einer Fahrtstrecke von 300m) und 2001 Ausgabezeitschritten ergeben sich folgende Resultate:

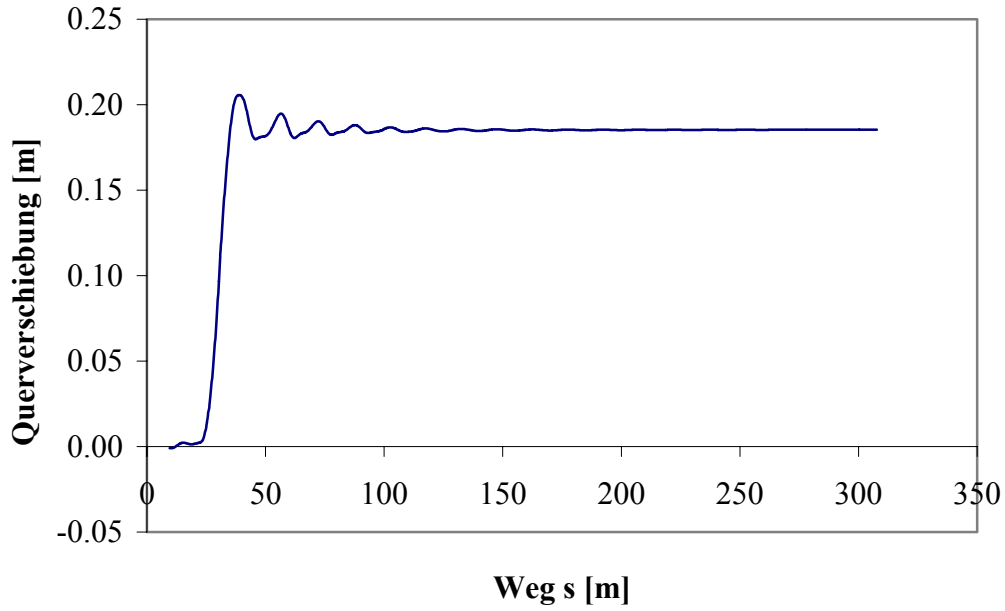

Bild 5.28 Querverschiebung des Wagenkastens zur Gleismitte (Weg s gemessen als Position des Wagenkasten-Referenzsystems)

Der große Betrag der Querverschiebung nach innen (Bild 5.28) ergibt sich, weil die Querverschiebung am Wagenkasten-Referenzsystem in der Wagenkastenmitte gemessen wird und sich hier durch die Sehnenstellung des Wagenkastens in der Kurve bereits geometrisch eine Verschiebung ergibt.

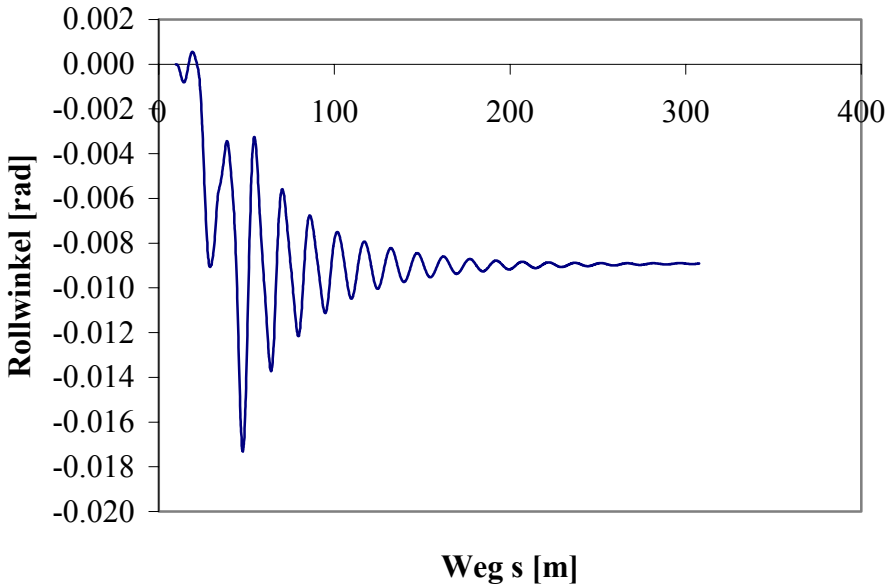

Bild 5.29 Rollwinkel des Wagenkastens (Weg s gemessen als Position des Wagenkasten-Referenzsystems)

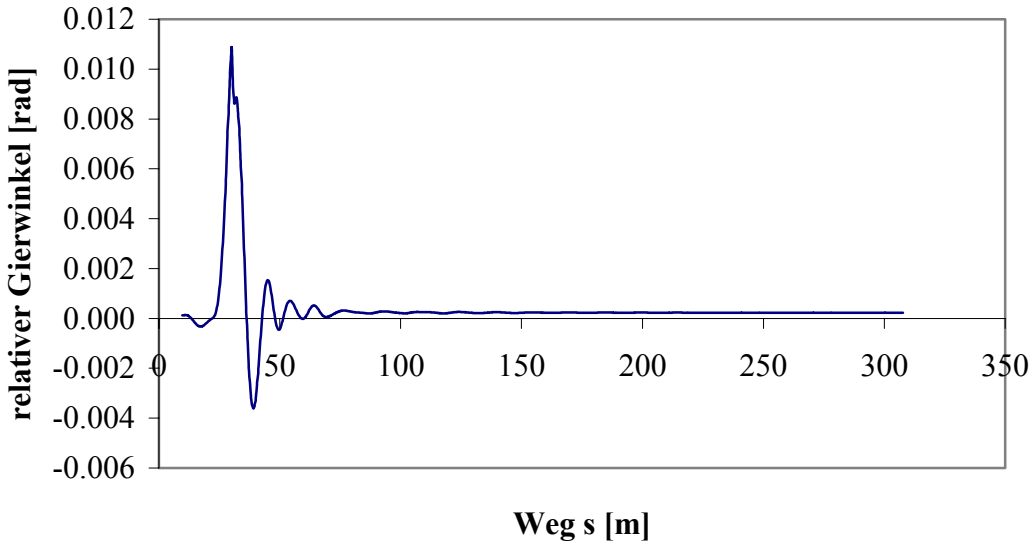

Bild 5.30 Relativer Gierwinkel des Wagenkastens bezogen auf den Streckenverlauf (Weg s gemessen als Position des Wagenkasten-Referenzsystems)

Wie man erkennen kann (Bild 5.29, Bild 5.30), tritt am Übergang zwischen der Geraden und dem Kreisbogen, bedingt durch das weitgehende Fehlen des Übergangsbogens, vor allem beim Rollwinkel ein starkes Schwingen auf.

Auch hier wurden die Ergebnisse über die SIMPACK/Centric Schnittstelle in eine für Centric lesbare Datei geschrieben.

5.3.3 Geradeausfahrt mit Gleislagestörungen für die Simulation mit skalierten Elementen in Centric

Um eine starke Ein- und Ausfederbewegung zu erzielen, wurde mit *Track-related Excitation* eine Gleislagestörung (Rollbewegung) vorgegeben. Folgende Streckenvorgabe wurde verwendet:

- Streckentyp: *Straight Track*
- *Excitation: Track related*
- *Track Excitation Type*:
  Initial Value of Excitation: 20m (Störung beginnt bei s=20m)
  Length of Initial Smoothing: 5m (Länge der Strecke, auf der eine Glättungsfunktion angewendet wird)

  - *Roll*: $\alpha = 0.07 \cdot \sin\left(\frac{2 \cdot \pi}{2 \cdot 2.6} \cdot s\right)$ [rad] (5.6)

    Die Störung wurde so gewählt, dass die halbe Wellenlänge gleich dem Radstand des Drehgestells ist (Bild 5.31). Steht ein Rad am oberen

Scheitelpunkt, so befindet sich das andere Rad der gleichen Seite daher am unteren Scheitelpunkt und es kommt zu maximalen Ein- und Ausfederwegen.

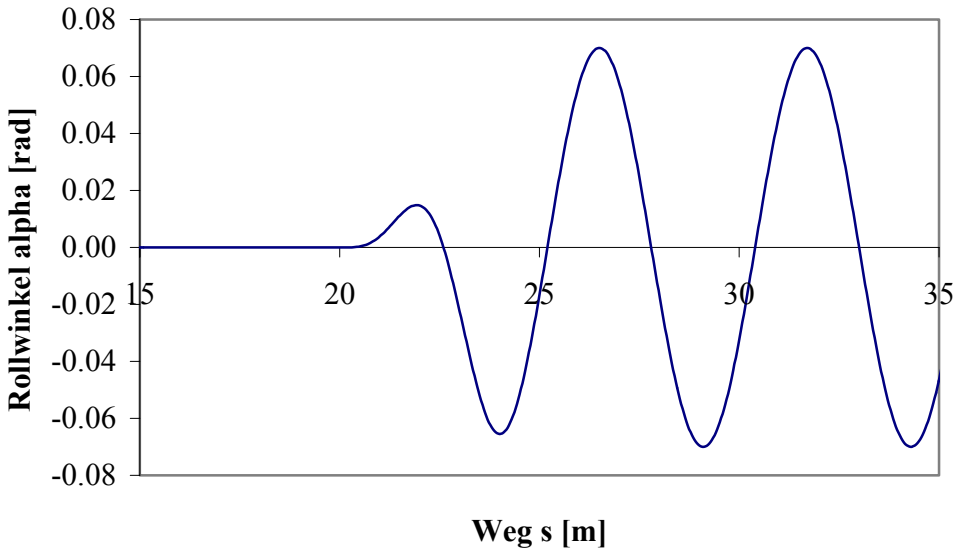

Bild 5.31 Rollwinkel des Gleises um die Mittellinie

Die Strecke wird mit einer **Geschwindigkeit** von v=25m/s durchfahren. Die Profilpaarung Rad/Schiene ist auch hier S1002/UIC 60. Der Reibungskoeffizient wurde auf dem *Default*-Wert von μ=0.4 belassen. Mit einer Integrationszeit von 10s (entsprechend 250m gefahrener Strecke) und 2001 Ausgabezeitschritten ergibt sich (Bild 5.32):

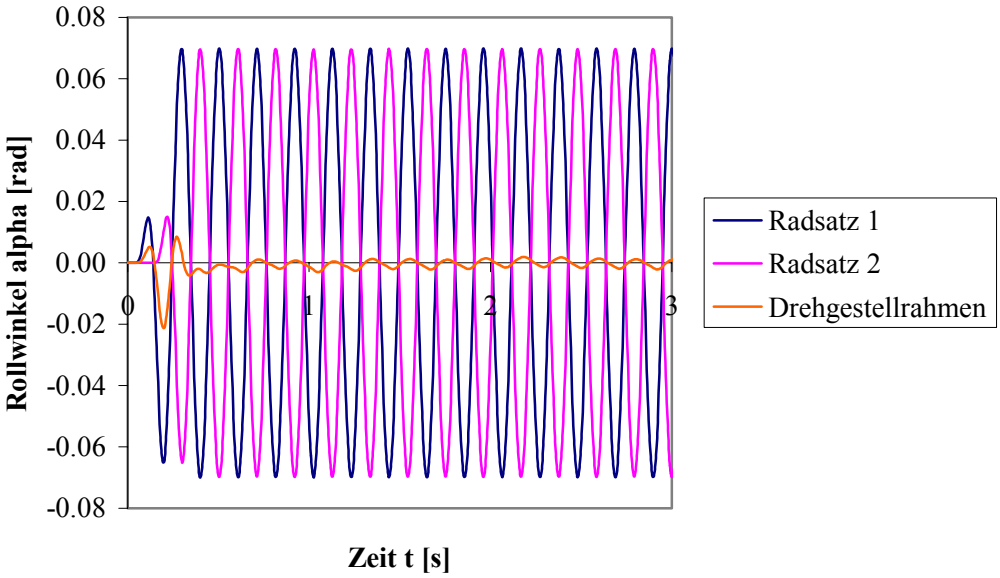

Bild 5.32 Rollwinkel am vorderen Drehgestell

Bedingt durch die Rollbewegungen der Radsätze relativ zum Rahmen des vorderen Drehgestells (Bild 5.32) kommt es zu einem starken Ein- und Ausfedern der Primärfedern, das in Centric dargestellt werden kann.

5.4 Simulation der Fahrten in Centric Innovation

In Centric Innovation wurden die in SIMPACK berechneten Ergebnisse an einem virtuellen Prototypen, der über die Geometrie des realen Fahrzeuges verfügt, dargestellt. Die Geometrie wurde in einem 3D-CAD Programm erstellt werden. Für die hier vorliegende Arbeit wurde von Siemens SGP Verkehrstechnik das Modell des Desiro UK Endwagens (Bild 5.33) zur Verfügung gestellt. Dabei wurden vom Modell einige für die hier durchzuführenden Simulationen nicht erforderliche Teile, wie zum Beispiel die Innenausstattung oder die Türen, entfernt, um die Modellkomplexität zu verringern und damit die Rechenperformance zu verbessern. Weiters wurde die Komplexität auch durch Hüllbildung und Eliminierung innerer Oberflächen verringert.

Bild 5.33 Centric Modell des Desiro UK Endwagen

Die Struktur des aus den konvertierten CAD-Daten entstandenen Modells wurde zuerst in die für die Simulation erforderlichen Komponenten zerlegt. Für die Simulation muss den Teilen im *Configuration Folder* die entsprechende Geometrie zugewiesen werden. Dazu muss jeder Teil zuerst korrekt positioniert werden. So sind zum Beispiel die Radsätze in den konvertierten CAD-Daten als eine Geometrie innerhalb des Drehgestells definiert. Die Beschreibung der Einbaulage wird hier einerseits durch die Position des Radsatzes relativ zum Drehgestell und dann durch die Positionierung der übergeordneten Struktur „Drehgestell“ bestimmt. Das heißt, dass in den *View Attributes* der Struktur Radsatz zum Beispiel für die x-Translation der halbe Radstand eingetragen ist (Abstand Drehzapfenmitte – Mitte Radsatz) und in den *View Attributes* der übergeordneten Struktur Drehgestell die Position des Drehzapfens. Wird nun die Geometrie Radsatz der entsprechenden Komponente im *Configuration Folder* zugewiesen, wird nur die Positionierung innerhalb des Drehgestells (im vorigen Beispiel halber Radstand) übernommen, woraus sich eine falsche Position ergibt.

Daher muss die Struktur zuerst in jene Komponenten zerlegt werden, die als eigenständige Teile bewegt werden sollen, und diese Teile müssen danach korrekt positioniert werden. Zusätzlich muss darauf geachtet werden, dass die Koordinatensysteme zwischen SIMPACK und Centric übereinstimmen. Im vorliegenden Beispiel wurde in SIMPACK ein bei der Simulation von Schienenfahrzeugen übliches Koordinatensystem mit nach unten gerichteter z-Achse und nach vorne gerichteter x-Achse verwendet, während in Centric Innovation das Koordinatensystem aus den CAD-Daten übernommen wird, in diesem Fall mit z-Achse zur Seite nach rechts und x-Achse nach vorn. Die Teile mussten daher in Centric um 90° um die x-Achse gedreht werden, um ein mit SIMPACK übereinstimmendes Koordinatensystem zu erhalten.

Für die Zuweisung an die entsprechenden Komponenten im *Configuration Folder* wurden folgende *Component Files* erstellt:

- Drehgestellrahmen des hinteren Drehgestells
- Drehgestellrahmen des vorderen Drehgestells
- Radsatzführungsschwinge des hinteren Drehgestells, hinten rechts
- Radsatzführungsschwinge des hinteren Drehgestells, hinten links
- Radsatzführungsschwinge des hinteren Drehgestells, vorne rechts
- Radsatzführungsschwinge des vorderen Drehgestells, vorne links
- Radsatzführungsschwinge des vorderen Drehgestells, hinten rechts
- Radsatzführungsschwinge des vorderen Drehgestells, hinten links
- Radsatzführungsschwinge des vorderen Drehgestells, vorne rechts
- Radsatzführungsschwinge des vorderen Drehgestells, vorne links
- Getriebe 1 (1=vorne im vorderen Drehgestell, 4=hinten im hinteren Drehgestell
- Getriebe 2
- Getriebe 3
- Getriebe 4
- Motor 1 (Die Fahrmotoren sind in SIMPACK mit einem Null-Freiheitsgrad-Gelenk am Drehgestellrahmen befestigt und führen daher die gleiche Bewegung aus, die Definition als eigene Komponente wäre daher eigentlich nicht notwendig)
- Motor 2
- Motor 3
- Motor 4
- Radsatz 1
- Radsatz 2
- Radsatz 3
- Radsatz 4
- Traverse hinten (Die Traversen sind mit einem Null-Freiheitsgrad-Gelenk am Wagenkasten befestigt, man könnte sie daher auch dem Wagenkasten zuweisen und nicht als eigenständige Komponente definieren)
- Traverse vorne
- Wagenkasten

- Verbindungselement zwischen Drehgestellrahmen und Wankstabilisator-Drehstabfeder am hinteren Drehgestell, rechts
- Verbindungselement zwischen Drehgestellrahmen und Wankstabilisator-Drehstabfeder am hinteren Drehgestell, links
- Verbindungselement zwischen Drehgestellrahmen und Wankstabilisator-Drehstabfeder am vorderen Drehgestell, rechts
- Verbindungselement zwischen Drehgestellrahmen und Wankstabilisator-Drehstabfeder am vorderen Drehgestell, links
- Primärfeder vorderes Drehgestell, vorne links
- Primärfeder vorderes Drehgestell, vorne rechts
- Primärfeder vorderes Drehgestell, hinten links
- Primärfeder vorderes Drehgestell, hinten rechts
- Primärfeder hinteres Drehgestell, vorne links
- Primärfeder hinteres Drehgestell, vorne rechts
- Primärfeder hinteres Drehgestell, hinten links
- Primärfeder hinteres Drehgestell, hinten rechts

Diese Komponenten werden nach der korrekten Positionierung mittels *Drag and Drop* den entsprechenden SIMPACK-*Ensembles* im *Configuration Folder* zugewiesen.

5.4.1 Definition von Lichtraumprofilen für die Simulation

Um der Simulation ein realistisches Erscheinungsbild zu geben und um die Kollisionskontrolle zu ermöglichen wurden zwei in Pro/ENGINEER erstellte Lichtraumprofile (Bild 5.34, Bild 5.35) angewendet.

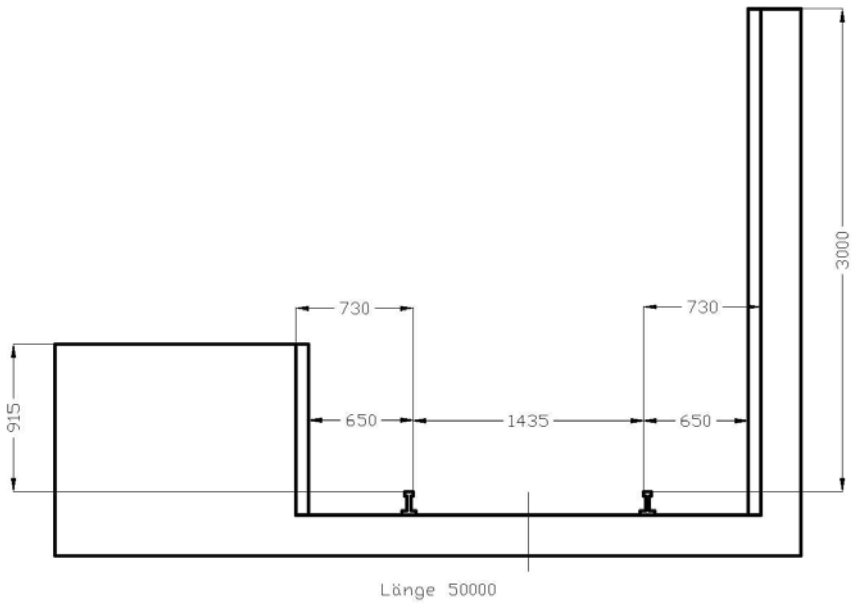

Bild 5.34 Lichtraumprofil, in Fahrtrichtung betrachtet

Für die **Geradeausfahrt** (Bild 5.34) wurde ein Profil erzeugt, das gegenüber dem für den Desiro UK vorgeschriebenen Lichtraumprofil dadurch modifiziert wurde, dass sich die Breite im Verlauf der Strecke verringert. Durch diese Verengung soll eine Kollision provoziert werden. Zusätzlich wurde auf der in Fahrrichtung rechten Seite das Profil, das die Bahnsteigbreite vorschreibt, weiter nach oben hochgezogen.

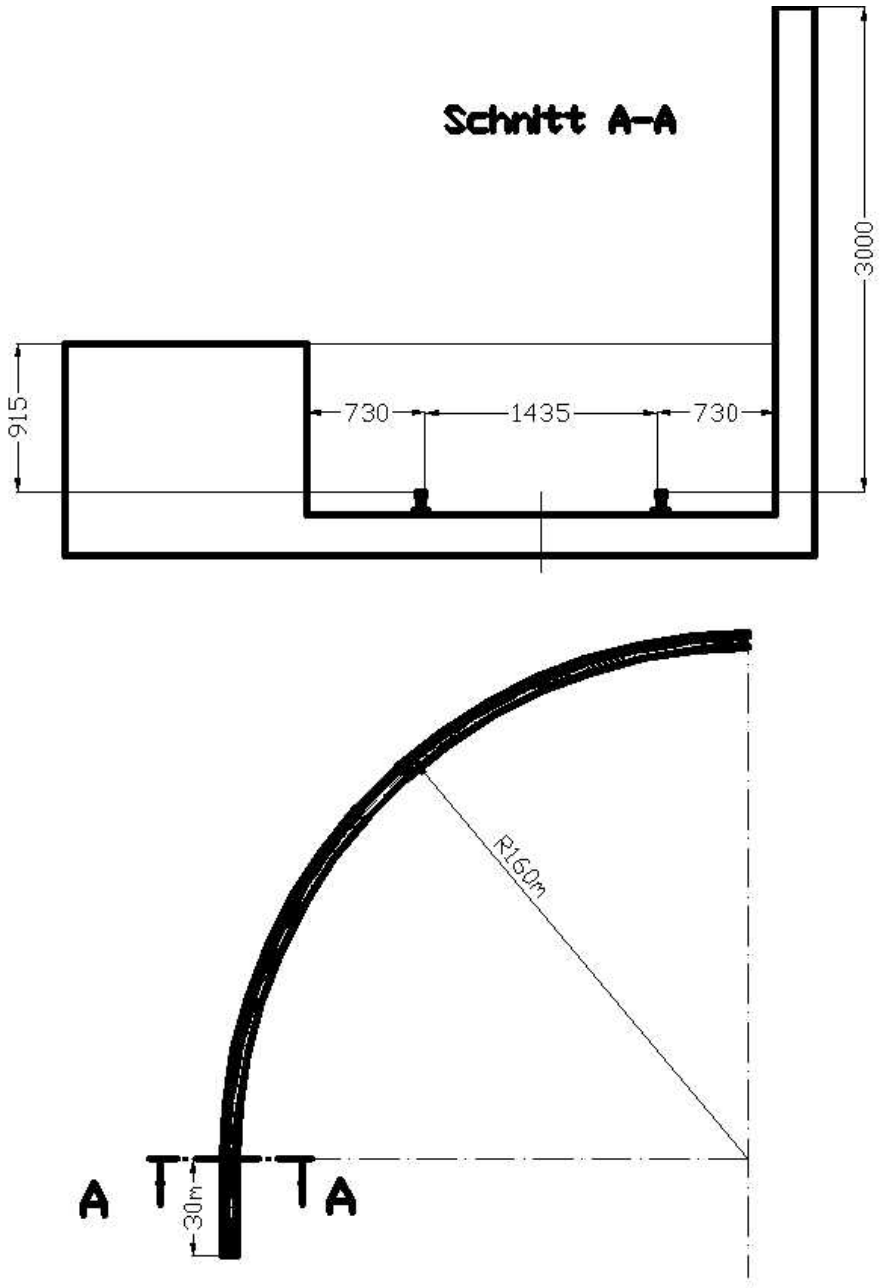

Bild 5.35 Lichtraumprofil für Kurvenfahrt (eine Hälfte und Querschnitt), Maße wenn nicht anders angegeben in mm

Für die **Kurvenfahrt** (Bild 5.35) wurde ein aus einer Geraden und einem Kreisbogen bestehendes Profil erzeugt. Als Kurvenradius wurde R=160m gewählt.
Für Radien unter 360m ist in den Vorgaben, nach denen der Desiro UK ausgelegt wurde [29], eine Erweiterung des Lichtraumprofils vorgesehen, um den vergrößerten Raumbedarf in der Fahrzeugmitte und an den Fahrzeugenden auszugleichen. Diese Erweiterung wurde hier aber nicht verwendet, um eine Kollision des Fahrzeugs mit dem Lichtraumprofil zu provozieren.

5.4.2 Resultate der Simulation in Centric

5.4.2.1 Geradeausfahrt

Bei der Geradeausfahrt auf der in 5.3.1 definierten Strecke und dem Lichtraumprofil nach (Bild 5.34) kommt es nach einer zurückgelegten Strecke von 11.394m zu einer Kollision (Dargestellt durch Farbänderung der beteiligten Komponenten, siehe Bild 5.36). In dieser Position befindet sich der vorderste Radsatz bei s=29.174m, der hintere Radsatz des vorderen Drehgestells bei s=26.574m, also nahe des Scheitelpunkts des Gleislagefehlers (siehe Bild 5.23). Wie man in (Bild 5.36) erkennen kann, kommt es zu einer Kollision zwischen dem Lichtraumprofil und dem vorderen Drehgestellrahmen.

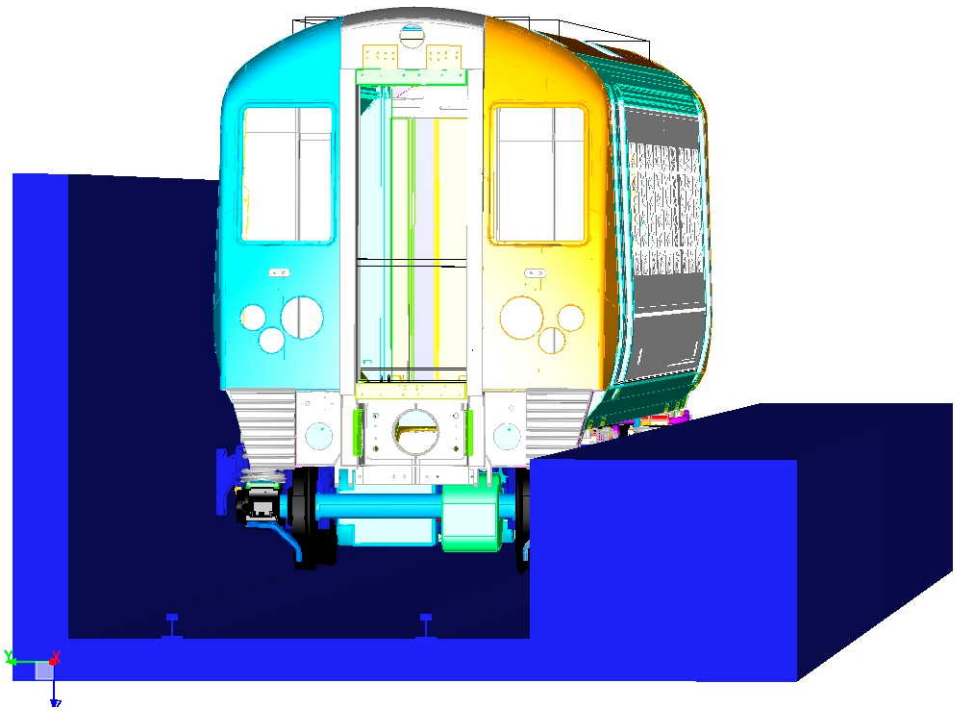

Bild 5.36 Beginn der Kollision bei Geradeausfahrt (Dunkelblau eingefärbte Komponenten (Drehgestellrahmen und Lichtraumprofil) an Kollision beteiligt)

Um die Position, an der die Kollision stattfindet, herauszufinden muss mit *Cross Section* ein Schnitt durch das Modell gelegt werden. Eine Anzeige des genauen Orts der Kollision wird von Centric Innovation momentan nicht angeboten.

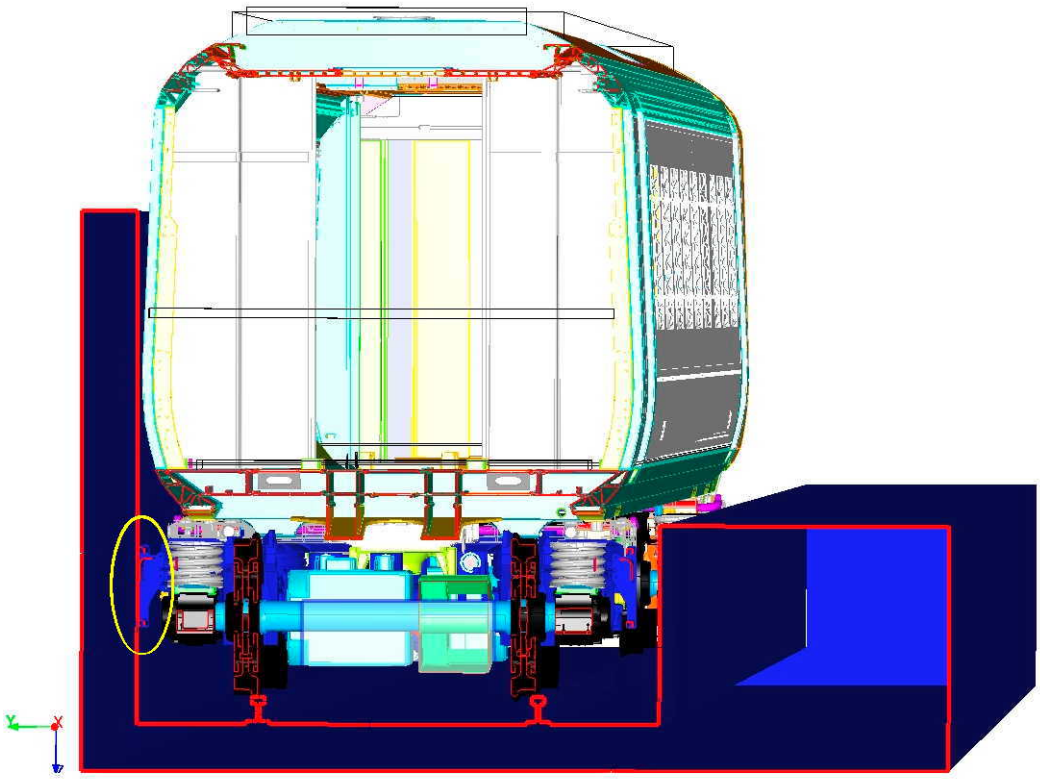

Bild 5.37 Schnitt durch das Fahrzeug und den Bahnsteig (Ansicht von vorne, gegen die Fahrtrichtung)

Wie man in (Bild 5.37) erkennen kann, kommt es zu einer Kollision zwischen der am Drehgestellrahmen befestigten (in Fahrtrichtung) rechten vorderen Trittstufe (in (Bild 5.37) links).

Die Position der Schnittebene wird wie in Kapitel 4.2 erläutert in einer Dialogbox mit einem Schieber festgelegt. Als Schnittebene wurde die y-z-Ebene gewählt. Damit ergibt sich das in (Bild 5.37) gezeigte Ergebnis.

5.4.2.2 Kurvenfahrt ohne Übergangsbogen

Bei der Kurvenfahrt ohne Übergangsbogen nach 5.3.2 mit dem Lichtraumprofil nach (Bild 5.35) kommt es nach einer gefahrenen Strecke von 22.194m zur Kollision des Wagenkastens mit dem Lichtraumprofil. Dabei befindet sich das Fahrzeug mit dem hinteren Drehgestell noch in der Geraden, das vordere Drehgestell läuft bereits im Kreisbogen. Mit Hilfe von *Cross Sections* wurde der Kollisionsbereich aufgefunden (Bild 5.38)

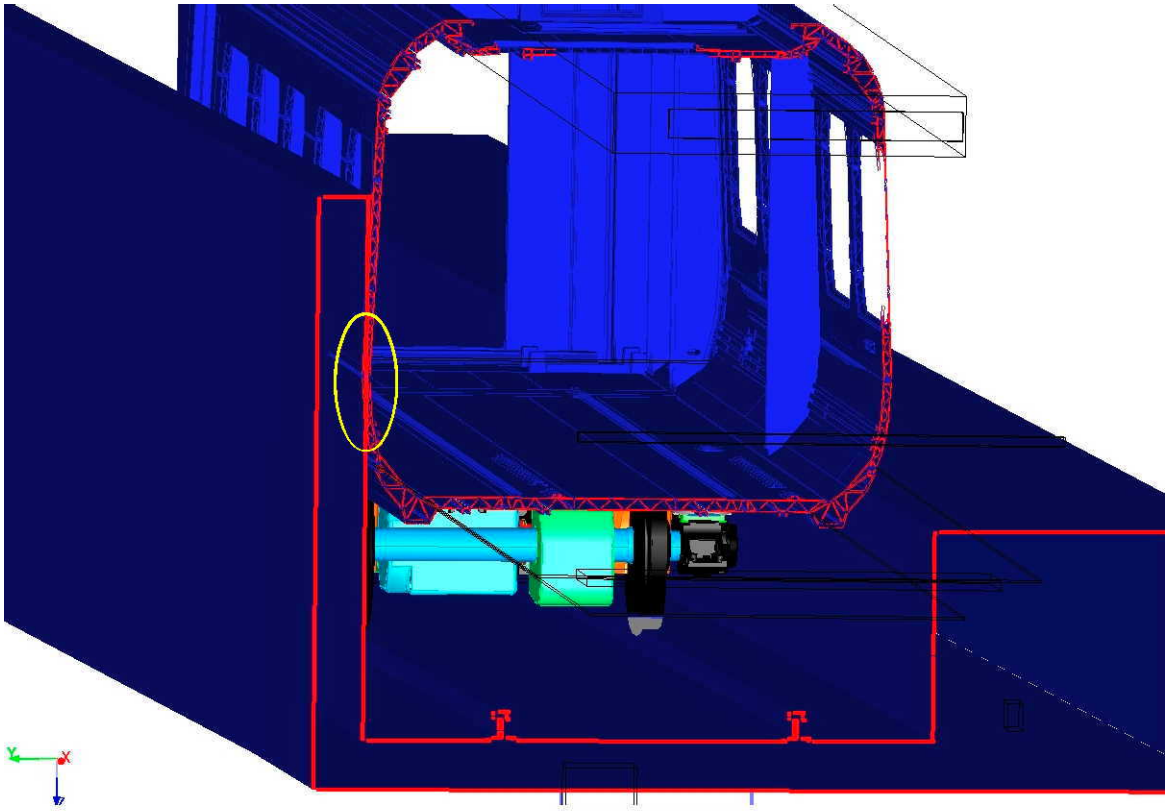

Bild 5.38 Kollision des Wagenkastens mit dem Lichtraumprofil im Kurveneingang im Schnitt dargestellt

Bei dieser Simulation bleibt die Kollision dann über die Fahrt durch den Kreisbogen bestehen.

5.4.2.3 Geradeausfahrt mit Gleislagestörung und skalierten Elementen

Bei der Geradeausfahrt nach 5.3.3 kommt es, ausgelöst durch die Gleislagestörungen, zu starken Ein- und Ausfederbewegungen der Primärfedern. Diese können, wie hier gezeigt werden soll, in Centric dargestellt werden. Durch die Schnittstelle SIMPACK/Centric wird, wie in Kapitel 3.3.8 erläutert, in solchen Fällen zusätzlich auch die Skalierung der Komponente übertragen, sodass die Feder oder ein anderes skaliertes Element in Centric seine Länge ändert. Die Möglichkeit, skalierte Objekte in Centric korrekt darzustellen, ist nicht nur zur Verbesserung der optischen Darstellung wünschenswert, sondern auch, um

eine korrekte Kollisionsprüfung zu ermöglichen. So können zum Beispiel die Schlingerdämpfer, die als seitlich überstehende Teile Kollisionen mit dem Lichtraum auslösen können, auf diese Weise korrekt dargestellt werden.

Bei der Verwendung skalierter Elemente muss in SIMPACK zuerst ein skaliertes *Ensemble* für das flexible Element erstellt werden (z. B. *Scaled Spring Point to Point*). Ist ein solches skaliertes Element in SIMPACK vorhanden, so wird beim Erstellen der MBS-Datei automatisch auch die Skalierungsinformation (Verkürzung oder Verlängerung des Elements) in die MBR-Datei geschrieben (Kapitel 3.3.8). Wird nun in Centric die entsprechende Geometrie dem *Ensemble* im *Configuration Folder* zugewiesen, so wird bei einer Simulation die Geometrie durch Ändern des Maßstabs in x-, y- und z-Richtung entsprechend der SIMPACK-Berechnung verkürzt oder verlängert. Auf diese Weise lassen sich einfache flexible Elemente, wie z. B. Federn, in Centric korrekt darstellen. Die Vorgangsweise bei der Zuweisung der Geometrien zu den *Ensembles* im *Configuration Folder* ist dabei bei flexiblen Körpern gleich wie bei starren Körpern. Die Entscheidung über die Art der Darstellung geschieht in SIMPACK durch die Auswahl eines passenden *Ensembles* für das flexible Element.

In den folgenden Bildern (Bild 5.39, Bild 5.40) kann man die korrekte Abbildung skalierter Elemente in Centric erkennen. Dargestellt wurde hier nur das vordere Drehgestell ohne Wagenkasten.

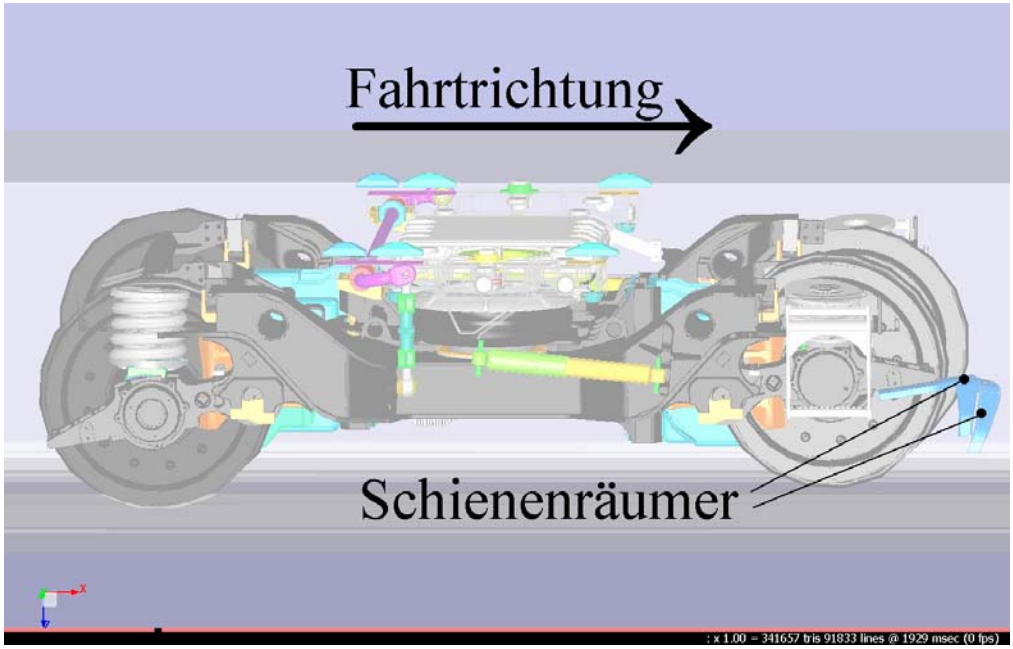

Bild 5.39 Vorderes Drehgestell, hintere Primärfeder federt aus

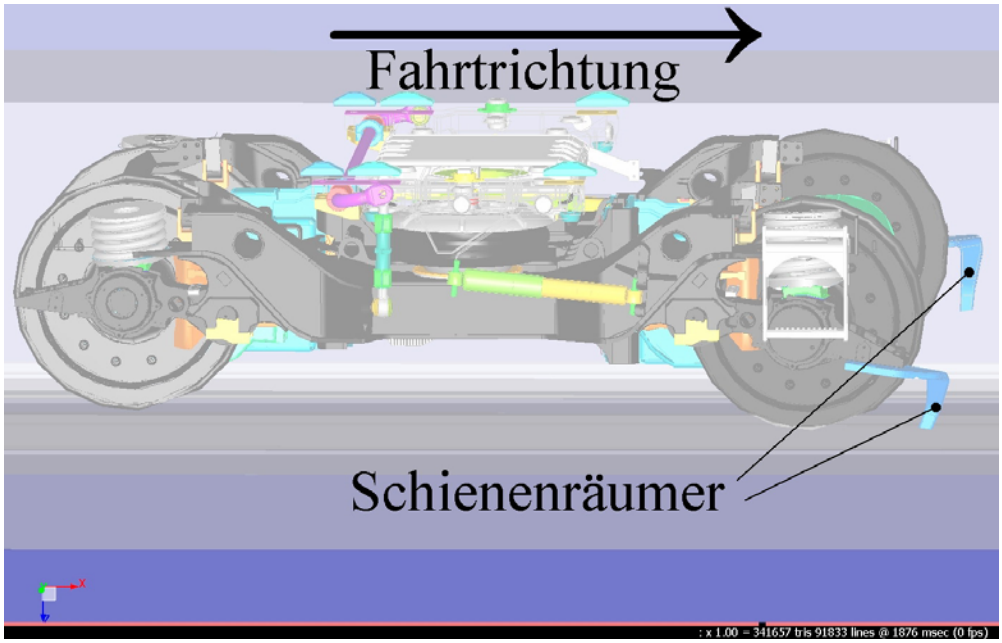

Bild 5.40 Vorderes Drehgestell, hintere Primärfeder federt ein

Kapitel 6. Vorgangsweise für die Geometriesimulation von Schienenfahrzeugen mit der SIMPACK/Centric Schnittstelle

Um eine korrekte Geometriesimulation durchführen zu können, müssen sowohl bei der Modellierung in SIMPACK als auch beim Aufbau des Modells in Centric Innovation einige Punkte beachtet werden.

6.1 Modellaufbau und Vorgangsweise in SIMPACK

Beim Aufbau des Modells in SIMPACK müssen für alle Bauteile, die später in Centric eine eigenständige Bewegung ausführen sollen, eigene *Ensembles* erstellt werden. Soll zum Beispiel ein Dämpfer in der Centric Simulation korrekt mit einem im Zylinder gleitenden Kolben dargestellt werden, so muss auch in SIMPACK ein *Ensemble* für den Kolben und ein *Ensemble* für den Zylinder definiert werden. Eine andere Möglichkeit wäre in diesem Fall den Dämpfer in SIMPACK durch ein skaliertes Element darzustellen. In diesem Fall werden dann in Centric Zylinder und Kolben als ein Teil, der entsprechend der Bewegung verkürzt oder verlängert wird, dargestellt. Die Bewegung von Kolben und Zylinder untereinander wird hier nicht dargestellt.

Bei der Einstellung des Integrators ist zu beachten, dass die Anzahl der eingestellten Ausgabezeitschritte von der Schnittstelle SIMPACK/Centric übernommen wird und im Nachhinein nicht mehr verändert werden kann (aktueller Stand). Hier muss ein Kompromiss gefunden werden zwischen den sehr langen Rechenzeiten in Centric bei einer großen Anzahl von Ausgabezeitschritten einerseits und der für eine hinreichende Genauigkeit erforderlichen Mindestanzahl von Ausgabezeitschritten andererseits. Hierbei ist zu beachten, dass in Centric die Kollisionsprüfung für jeden Ausgabezeitschritt durchgeführt wird. Bei zu grober Wahl der Ausgabeschrittweite wird daher eine nur sehr kurz andauernde Kollision, die zwischen zwei Schritte liegt, nicht gemeldet.

Um bei einer Kollisionsprüfung ein mit dem berechneten Streckenverlauf übereinstimmendes Lichtraumprofil erzeugen zu können, müssen die Informationen über den Streckenverlauf aus SIMPACK in eine Datei geschrieben werden. Bei der vorliegenden Software-Version muss dies manuell, z. B. mit den Resultaten des Sensors *$S_Rail_Track_Frame* für den Verlauf der Gleismitte ohne Gleislagestörungen und *$S_Rail_ProfRef_Right/Left_of_<Radsatzname>* für den Schienenverlauf mit Gleislagestörungen geschehen. Diese Resultate können im 2D-Plot Modul in eine ASCII-Datei geschrieben und dann in eine für ein 3D-CAD-System verwertbare Form gebracht werden.

Das Erstellen des MBS-Dateien erfolgt automatisch durch die Auswahl des Punkts *New Studio MBS File* im Menü *Calculation – Perform Measurement*. Vor dem Erstellen der MBS-Datei muss eine Zeitintegration durchgeführt werden.

6.2 Modellaufbau und Vorgangsweise in Centric Innovation

In Centric muss zuerst ein *Simulation Folder* mit der durchzuführenden Simulation erstellt werden. Dies geschieht durch *Add Files* und Angabe der MBS-Datei (Kapitel 4.7). Dann wird von Centric automatisch ein *Configuration Folder* mit den SIMPACK *Ensembles* und ein *Scenario Folder*, der *Data Probes* mit den Positionsdaten der einzelnen SIMPACK-Ausgabezeitschritte enthält, erzeugt.

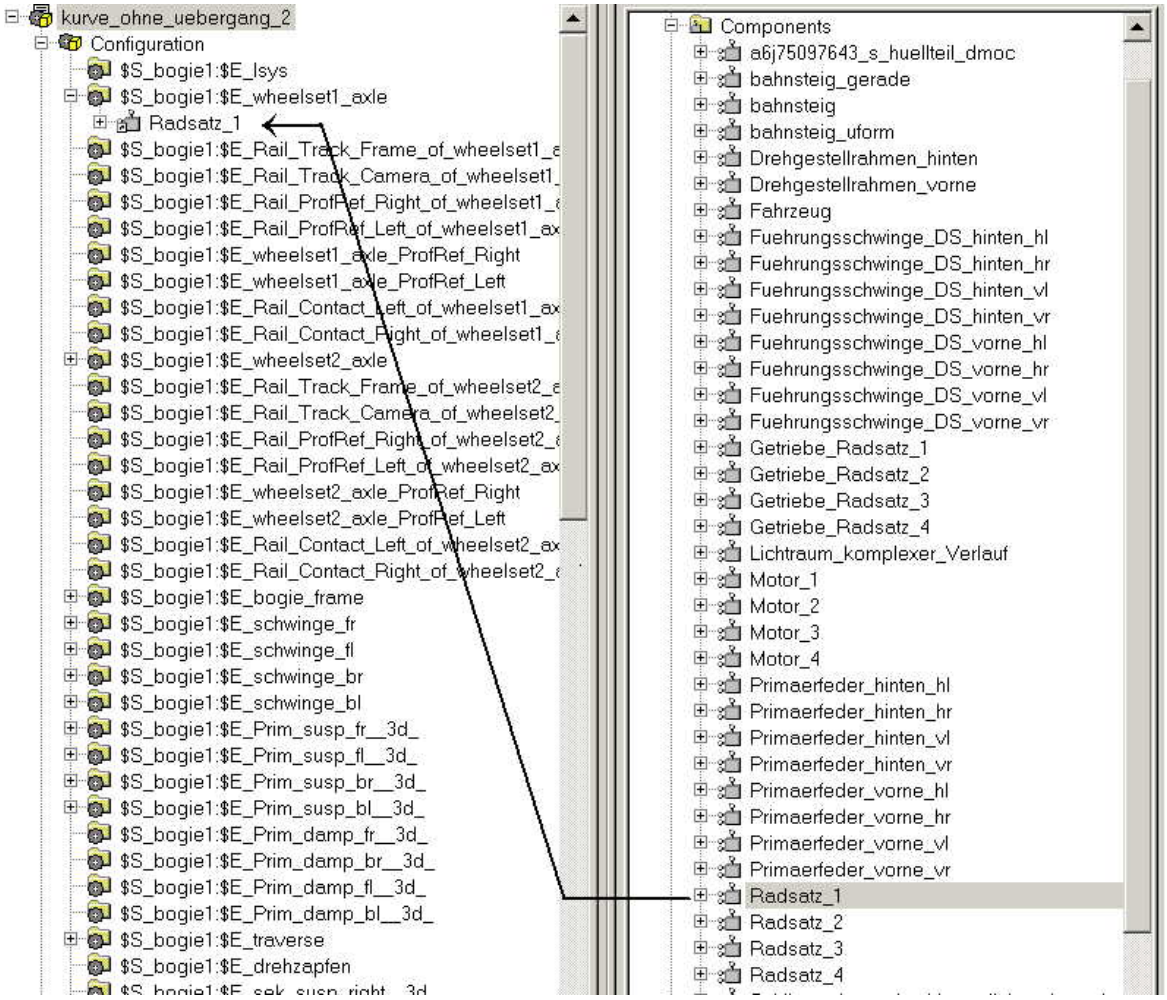

Bild 6.1 Zuweisung der einzelnen Geometrien zu den Ensembles im Configuration Folder. Linkes Fenster: Configuration Folder. Rechtes Fenster: Components in Centric

Den SIMPACK *Ensembles* im *Configuration Folder* muss dann mittels *Drag and Drop* die entsprechende Centric-Geometrie zugewiesen werden (Bild 6.1). Dabei ist zu beachten, dass in Centric das gleiche Koordinatensystem wie in SIMPACK verwendet wird. Stimmen diese Koordinatensysteme nicht überein (was meistens der Fall sein wird), muss die Positionierung der Komponenten in Centric manuell durch Verändern der *View Attributes* so modifiziert werden, dass die Koordinatensysteme übereinstimmen. Weiters wird bei der aktuellen Version (3.5) von Centric Innovation beim Zuweisen von Geometrien, die sich in einer Baumstruktur befinden, zu den *Ensembles* im *Configuration*

Folder nur ihre Relativposition innerhalb der Baumstruktur und nicht ihre Absolutposition übernommen. Die Positionierung stimmt daher nicht mehr und muss manuell durch Ändern der *View Attributes* korrigiert werden.

Soll nur eine grafische Animation erzeugt werden, kann die Simulation jetzt gestartet werden. Um eine Video-Datei zu erzeugen, muss dabei auf externe Software zurückgegriffen werden, z. B. auf das als Shareware erhältliche SnagIt.

Soll eine Kollisionsprüfung durchgeführt werden, so müssen noch die Einstellungen der Kollisionsprüfung angepasst werden. Dies geschieht im Menü *Collision Settings* unter der Registerkarte *Intra Component Collision*. Hier kann die Kollisionsprüfung für einzelne Komponenten und Geometrien ein- und ausgeschaltet werden. Die *Inter Component Collision*, bei der bestimmte Paare für die Kollisionsprüfung ein- oder ausgeschaltet werden können, funktioniert bei der aktuellen Version (3.5) innerhalb der Simulation nicht. Danach muss die Kollisionsprüfung aktiviert werden. Beim Start von Centric Innovation ist die Kollisionsprüfung standardmäßig deaktiviert.

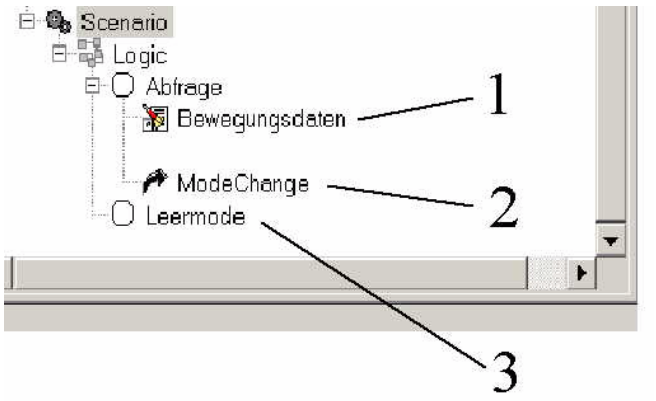

Bild 6.2 Logic, um die Simulation bei einer Kollision anzuhalten

Um die Simulation beim erstmaligen Auftreten einer Kollision anzuhalten, kann eine *Logic* erstellt werden (Bild 6.2). Dieses *Logic* ist folgendermaßen gestaltet: In einem neuen *Scenario Folder* innerhalb der Simulation wird eine *Logic* erstellt. Diese *Logic* enthält zwei *Modes*: Im ersten *Mode* wird eine *Data Probe Activity* (1) erstellt, die die *Data Probe* der Simulation aufruft. Weiters wird ein *Mode Change* (2) erstellt, der beim Auftreten einer Kollision den zweiten *Mode* (3) aktiv schaltet. Dieser *Mode* ist leer, damit wird die Simulation an der Stelle der erstmaligen Kollision angehalten.

Die Auswertung der Kollision muss durch das Legen von Schnitten durch die betroffenen Komponenten geschehen. Eine *Fly-to-Collision*-Funktion, die den genauen Ort der Kollision angibt, ist momentan (Version 3.5) nicht vorhanden. Diese Schnitte können in Centric mit *Cross Section* erstellt werden. Die Position der Schnittebene kann durch Angabe durch einen Schieber oder durch Eingabe der numerischen Position der Schnittebene festgelegt werden. Ist der Ort der Kollision aufgefunden, so kann er markiert und mit Anmerkungen versehen werden.

Soll nicht nur die erste Kollision, sondern sollen (falls vorhanden) mehrere Kollision während einer Simulation ausgewertet werden, so darf die Simulation nach der ersten Kollision nicht durch eine Logik angehalten werden, da eine einmal gestoppte Simulation beim neuerlichen Start wieder in der Ausgangslage beginnt. Um nicht bei der ersten, sondern erst bei der zweiten, dritten,… Kollision die Simulation zu stoppen, muss die Logik entsprechend geändert und eine erneute Simulation durchgeführt werden.

Kapitel 7. Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde zuerst eine Literaturstudie zum Stand der Technik aktiver DMU-Werkzeuge durchgeführt (Kapitel 2). Dabei wurde festgestellt, dass die Einbindung von Dynamikinformationen in DMU-Werkzeuge teilweise möglich ist, dass aber bei der Auswertung hinsichtlich Kollisionserkennung die heute zur Verfügung stehenden Algorithmen mit der heute verfügbaren Hardware bei komplexeren Modellen noch beträchtliche Rechenzeit in Anspruch nehmen.

Um die Einbindung von Dynamikinformationen aus dem Mehrkörperdynamikprogramm SIMPACK in die DMU-Software Centric Innovation zu testen, wurde zuerst ein einfaches Modell (gedämpftes Pendel) in SIMPACK aufgebaut und berechnet (Kapitel 5.1). Die Dynamikdaten dieses Modells wurden dann mit Hilfe der SIMPACK/Centric-Schnittstelle in die Centric Innovation Software eingebracht und mit einfachen Geometrien verknüpft. In Centric wurde dann die Leistungsfähigkeit der Kollisionserkennung getestet. Dabei wurde festgestellt, dass Kollisionen auch bei nur geringen Überdeckungen zuverlässig erkannt werden. Weiters wurde festgestellt, dass die Kombination der Kollisionserkennung mit den von Centric zur Verfügung gestellten Logiken zu Verfälschungen des Ergebnisses hinsichtlich des Kollisionszeitpunktes führen kann, d. h. dass Kollisionen zwar erkannt, aber zu spät gemeldet werden (Kapitel 5.1.2.2 bis 5.1.2.5).

Danach wurde die Vorgangsweise für ein komplexeres Modell (Schienenfahrzeug) verallgemeinert. Dazu wurde zuerst in SIMPACK ein vereinfachtes Modell eines Schienenfahrzeugs erstellt (Kapitel 5.2). Mit diesem Modell wurden in SIMPACK Fahrten auf verschiedenen Strecken simuliert (Kapitel 5.3). Die Ergebnisse dieser Fahrten wurden in Centric eingebracht und dort mit den Geometrien eines Schienenfahrzeugs verknüpft. Zusätzlich wurden die Geometrien von Lichtraumprofilen entsprechend den in SIMPACK simulierten Streckenverläufen geladen. Damit wurden die Fahrten dann in Centric simuliert und das Fahrzeug auf Kollisionen mit dem Lichtraumprofil überprüft (Kapitel 5.4). Dabei wurde festgestellt, dass bei solchen komplexeren Modellen mit großen Baugruppen die von der Kollisionserkennung zur Verfügung gestellten Ergebnisse noch unbefriedigend sind, da als Ergebnis nur die an der Kollision beteiligten Komponenten angezeigt werden. Bei Schienenfahrzeugen sind diese Komponenten (z. B. Wagenkasten) aber oft sehr groß und das Auffinden des Kollisionsorts durch das Legen von Schnitten ist entsprechend zeitaufwendig.

Weiters wäre für solche Anwendungen die Einführung einer bewegten Ansicht (*Viewpoint*) und die Möglichkeit einer umfangreicheren Abspielkontrolle für die Simulation in Centric (nicht nur Start und Stopp) wünschenswert.

Mit den Erfahrungen aus diesen Simulationen wurde dann eine Anleitung für die Durchführung einer Geometriesimulation von Schienenfahrzeugen mit der derzeit vorhandenen Software (SIMPACK 8.5.10, Centric Innovation 3.5) erstellt (Kapitel 6).

Literaturverzeichnis

[1] G. Wang: Definition and review on virtual prototyping. Pittsburgh: Proceedings of international design engineering technical conferences and design automation conference 2001. 2001
Im Internet:
http://www.umanitoba.ca/faculties/engineering/mech_and_ind/prof/wang/DETC01_CIE21265.PDF

[2] J. Rix, H. Kress, K. Schroeder: Das virtuelle Produkt – neue Präsentations- und Interaktionstechniken in der Produktentwicklung. In: Simulation in der Praxis – neue Produkte effizienter entwickeln. Düsseldorf: VDI-Verlag. 1995

[3] G. Zachmann: Virtual reality in assembly simulation – collision detection, simulation algorithms and interaction techniques. Technische Universität Darmstadt, Abteilung für Computerwissenschaften: Dissertation. 2000
Im Internet:
http://web.informatik.uni-bonn.de/II/ag-klein/people/zach/papers/zachmann_diss.pdf

[4] B. Balasubramanian, A. Katzenbach: Simulation im Automobilbau – von der Idee bis zum Kundenfahrzeug. In: Simulation in der Praxis – neue Produkte effizienter entwickeln. Düsseldorf: VDI-Verlag. 1995

[5] P. Buttolo, P. Stewart, A. Marsan: A haptic hybrid controller for virtual prototyping of vehicle mechanism. In: Proceedings of the 10th Symp. on haptic interfaces for virtual environment & Teleoperator Systems (HAPTICS '02). IEEE. 2002

[6] G. Granholm, J. Säilä: Real time simulation in product development. Espoo: Virtual prototyping Symposium 210. 2001
Im Internet:
http://www.vtt.fi/vtt/tutkimus/virtuaaliprototypointi/loppuraportti/s210tekstirp.pdf

[7] http://www.enovia.com am 20.6.2002

[8] http://www-5.ibm.com/de/catia/produkte/enovia_overview.html am 18.6.2002

[9] http://www.plmsolutions-eds.com/products/colsol/ am 20.6.2002

[10] http://www.adams.com/news/releases/1998/news_092498.htm am 20.6.2002

[11] http://www.centricsoftware.com am 19.7.2002

[12] http://www.opendmu.de/Partners/Tecoplan/tecoplan.htm am 15.4.2002

[13] R. Spies, J. Hudi: Integration of Digital Mock-Up and multibody simulation in the product-development process. Berlin: International Adams Users' Conference. November 17-18 1999

[14] Produktentwicklung mit Digital Mock-up. In: Automobiltechnische Zeitschrift Sonderausgabe Audi A4 S. 78-83. Wiesbaden: Vieweg Verlag. November 2000

[15] M. C. Ling, S. Gottschalk: Collision detection between geometric models: a survey. In: Proceedings of IMA Conference on Mathematics of Surfaces. 1998
Im Internet: ftp://ftp.cs.unc.edu/pub/users/manocha/PAPERS/COLLISION/cms.pdf

[16] S. Gottschalk, M. C. Lin, D. Manocha: OBBTree: A hierarchical structure for rapid interference detection. In: Proceedings of ACM Siggraph 1996. 1996
Im Internet:
ftp://ftp.cs.unc.edu/pub/users/manocha/PAPERS/COLLISION/sig96.pdf

[17] M. C. Lin, D.Manocha, J. Cohen, S. Gottschalk: Collision detection: algorithms and applications. In: Proceedings of Algorithms for Robotics Motion and Manipulation S. 129-142 eds. Jean-Paul Laumond and M. Overmars, A.K. Peters (invited submission). Im Internet:
ftp://ftp.cs.unc.edu/pub/users/manocha/PAPERS/COLLISION/wafr.pdf

[18] T. Möller: A Fast Triangle-Triangle Intersection Test. In: Journal of Graphics Tools 2 (2) S. 25-30. 1997

[19] P. Jimenez, F. Thomas, C. Torras : 3D collision detection : a survey. In: Computers & Graphics 25 (2001) S. 269-285. 2001

[20] M. Held: ERIT – A collection of efficient and reliable intersection tests. In: Journal of Graphics Tools 2 (4) S. 25-44. 1997

[21] G. Taylor : Point in polygon test. In: Survey Review 32 (254) S. 479-484. 1994
Im Internet: http://www.comp.glam.ac.uk/pages/staff/getaylor/publications.htm

[22] H. König, T. Strothotte: Fast collision detection for haptic displays using Polygonal models. Ghent: Proceedings of the Conference on Simulation and Visualization 2002 S. 289-300. 2002
Im Internet:
http://isgwww.cs.unimagdeburg.de/graphik/pub/files/Koenig_2002_FCD.pdf

[23] J. Cohen, M. C. Lin, D. Manocha, M. K. Ponamgi: I-COLLIDE: An interactive and exact collision detection system. In: Proceedings of ACM International 3D Graphics Conference S. 189-196. 1995
Im Internet: http://www.cs.unc.edu/~geom/I_COLLIDE/index.html

[24] T. C. Hudson, M. C. Lin, J. C. Cohen, S. Gottschalk, D. Manocha: V-COLLIDE: Accelerated collision detection for VRML. In: Proceedings of VRML 1997. 1997 Im Internet: http://www.cs.unc.edu/~geom/V_COLLIDE/

[25] SIMPACK Dokumentation – Release 8.5. Wessling: INTEC. 2002

[26] Meyberg, Vachenauer: Höhere Mathematik 1, 3. korrigierte Auflage. Berlin: Springer Verlag. 1995

[27] Getting Started with Centric Innovation – Release 3.5. San Jose: Centric Software Incorporated. 2002

[28] Datenblatt Nr. 8001/Stand 11/00. Graz: Siemens SGP Verkehrstechnik GmbH. 2000

[29] Management of Clearances and Gauging Appendix C S. 25f. Railway Group Standard Issue One. Railtrack. August 2000