Real-Time Simulation of Rail Vehicle Dynamics

By
Michael David Simpson

Submitted to the Newcastle University
School of Computing Science in partial
fulfilment of the requirements for the
Degree of Doctor of Philosophy

September 2016

Abstract

Software simulation is vitally important in a number of industries. It allows engineers to test new products before they leave the drawing board and enables tests that would otherwise be difficult or impossible to perform. Traditional engineering simulations use sophisticated numerical methods to produce models that are highly accurate, but computationally expensive and time-consuming to use. This accuracy is essential in the latter stages of the design process, but can make the early stages - which often involve frequent, iterative design changes - a lengthy and frustrating process. Additionally, the scope of such simulations is often limited by their complexity.

An attempt has been made to produce an alternative, real-time simulation tool, developed using software and development practices from the video games industry, which are designed to simulate and render virtual environments efficiently in real-time. In particular, this tool makes use of real-time physics engines; iterative, constraint-based solver systems that use rigid body dynamics to approximate the movements and interactions of physical entities. This has enabled the near-real-time simulation of multi-vehicle trains, and is capable of producing reasonably realistic results, within an acceptably small error bound, for situations in which a real-time simulation would be used as an alternative to existing methods.

This thesis presents the design, development and evaluation this simulation tool, which is based on NVidia's PhysX Engine. The aim was to determine the suitability of a physics engine-based tool for simulating various aspects of rail dynamics. This thesis intends to demonstrate that such a tool, if configured and augmented appropriately, can produce results that approach those produced by traditional methods and is capable of simulating aspects of rail dynamics that are otherwise prohibitively expensive or beyond the capabilities of existing solutions, and may therefore be a useful supplement to the existing tools used in the rail industry.

Pages: 285
Word Count: 79,761

[BLANK PAGE]

Acknowledgements

The author would like to thank Dr Graham Morgan and Dr Gary Ushaw for their supervision and advice. Thanks also to Robin, William, Calum and the rest of the Systems Team for their support during the development of the Locomotion tool.

Thanks also to Dr Joe Carruthers, Dr Francis Franklin and Conor O'Neil at NewRail for their input and support. The development of the simulation tool has been partly funded by NewRail as part of the SecureMetro project, which was an EC funded project under FP7.

Finally, big thanks to Mum, Dad, Sam, Jo and Alan for all their patience, encouragement and support (both emotional and financial). I couldn't have done it without you.

[BLANK PAGE]

Contents

List of Figures

All Figures are original, unless referenced.

List of Tables

List of Formulas

Abbreviations

The following abbreviations are used throughout this thesis:

- **WRI** - 'Wheel/Rail Interface', referring to the point of contact between the wheel and the rails (see Section 2.1)
- **TM** - 'Timing Multiple', referring to a multiple added to the simulation code to adjust simulation timing parameters (see Section 5.8.1)
- **SIC** - 'Solver Iteration Count', referring to a parameter of rigid bodies in the PhysX engine (see Section 3.4.7)
- **SEV** - 'Solver Extrapolation Value', referring to a parameter of joints in the PhysX engine (see Section 3.4.7)
- **MAV** - 'Max Angular Velocity', referring to a parameter of rigid bodies in the PhysX engine (see Section 3.4.7)
- **SB Wheelset** - The 'Single Body' Wheelset, referring to one of the wheelset design variations (see Section 3.2.2)
- **MB Wheelset** - The 'Multi Body' Wheelset, referring to one of the wheelset design variations (see Section 3.2.2)

Nomenclature

The following symbols are used in this thesis to represent various values and properties. Formulas and diagrams from external sources have been modified to make them consistent with the nomenclature below.

L - lateral load	L_l / L_r - lateral load (left/right)
V - vertical load	V_l / V_r - vertical load (left/right)
N - normal force	N_l / N_r - normal force (left/right)
λ - effective wheelset conicity	λ_l / λ_r - wheel conicity (left/right)
δ - wheel contact angle	L_w - total lateral force (gravitational stiffness)
F_{tan} - tangential friction force	W - total vertical load on a wheelset
r - track curve radius	y - wheelset lateral displacement/offset
v - forward velocity	γ_l - lateral creep force
α - wheelset angle of attack	μ - coefficient of friction (wheel/rail)
V_{lat} - wheel lateral velocity	m - mass
$\dot{y}$ - wheelset lateral velocity	g - acceleration due to gravity
ω - wheelset rotational velocity	r_l / r_r - wheel radius (left/right)
r_0 - wheelset rolling radius	π - Pi (3.14159...)
l_0 - half the track gauge	ϕ - wheelset roll angle

Chapter 1

Introduction

Rail engineers currently employ various methods in the development and testing of rail vehicles and safety features, including a range of software simulation tools. Software simulation is an important research and development tool, and has some clear advantages over real-world testing, but existing simulation tools are complex, slow and expensive to use. This thesis presents an alternative, real-time simulation tool based on development strategies and software used in the video games industry; in particular real-time physics engines. These engines prioritise speed over accuracy, but are based on real-world mathematics and should be capable of producing realistic results.

This thesis describes the design and implementation of 'Locomotion', a rail dynamics simulation tool based on NVidia's PhysX Engine, and the subsequent evaluation of that tool for engineering use. The main goal of this research was to determine whether a simulation tool based on a physics engine is capable of producing realistic data and simulating key aspects of rail dynamics in a way that would be useful to engineers, as well as to attempt to evaluate the error bound of the results produced by the tool and attempt to constrain the error where possible.

1.1 NewRail

The development of the Locomotion simulation tool was conducted in association with NewRail, a research centre at Newcastle University. NewRail is dedicated to research and development in the fields of rail safety and security, working with various international partners on a range of projects [1]. In particular, the tool was developed in consultation with members of the Rail Vehicles Group, whose focus is on the design, development and testing of new materials and processes for transport applications [2].

1.2 Rail Dynamics Simulation

There are a number of simulation tools currently used in the rail industry (examples of which are discussed in Section 2.3), which are highly accurate, but computationally and temporally expensive. There are times when this level of accuracy is critical, but there are also times when true accuracy is less important and when a level of abstraction could speed up the process of developing new products. There are also scenarios, such as the dynamic behaviour (and derailment behaviour) of multi-vehicle trains, which are currently beyond the scope of the traditional tools due to their mathematical complexity.

NewRail engineer Dr. Joe Carruthers explained: *"One of frustrations with conventional numerical-based tools for engineering simulation is the length of time that it can take such models to solve. With finite element crash simulations typically taking hours or even days to solve, iterative design optimisations can become a slow process. Therefore, as a complement to existing analysis software, NewRail is interested in alternative simulation technologies that are capable of providing ballpark estimations in real-time (or near real-time). These would facilitate the rapid evaluation of design concepts, particularly for the early stages of the design process, in which the rapid, approximate evaluation of a wide range of design options is often more important than absolute precision."* [3]

The problem is that rail vehicles are complex, dynamic systems made up of multiple bodies with many degrees of freedom, which makes simulating them a challenging and computationally expensive process.

1.3 A Real-Time Alternative

A real-time simulation tool would have a number of potential advantages over the existing tools. For example, it would allow engineers to rapidly make changes to a vehicle's design, providing near-instantaneous feedback, which would help to speed up the early, more iterative stages of the design process. There have been a number of attempts to speed up the existing simulation techniques currently used by engineers (discussed in Section 2.3), but none of these have achieved the speed or simplicity necessary for real-time simulation. This Thesis presents an alternative approach, which should allow for real-time simulation of rail vehicles. There are a number of potential applications of a real-time simulation tool, including gauging, route design and optimisation, the simulation of the wheel/rail interface and other aspects of rail dynamics, some of which are discussed in Sections 2.1, 2.2 and 2.3.

1.4 Real-Time Physics Engines

The simulation tool presented in this Thesis was developed using physics simulation technology from the video games industry, discussed in Section 2.4, along with other software and development practices that are used to simulate and render virtual environments efficiently in real-time. Physics simulation is used by game developers to add realistic physical behaviours to entities in the game world, and its use has become so widespread that a number of third-party middleware Physics Engines have been developed. Thanks to evolutions in hardware and software in the last decade, these engines have become more sophisticated and may have reached a point where they can produce data that is sufficiently realistic as to be useful to engineers.

The use of games industry software also allows the simulation to run on a standard desktop PC and commodity gaming hardware, making it a relatively low-cost solution that does not require expensive, specialist hardware. NVidia's PhysX engine [4] was chosen because it has become the predominant engine used in the games industry and because it has been shown to produce accurate results in certain circumstances (as discussed in Section 2.3.4). The PhysX Software Development Kit was also freely available for educational and non-commercial use at the start of this project.

1.5 The Locomotion Simulation Tool

'Locomotion', the simulation tool presented in this thesis, represents an attempt to create a real-time simulation tool using an alternative approach to the traditional, numerical-based engineering tools. Locomotion was designed to provide engineers with a testing environment that can be used to rapidly evaluate design changes, as well as the ability to simulate multi-vehicle trains and to produce useful data about behaviours such as gauging, inter-vehicle stability and derailment that are currently prohibitively expensive or are beyond the capabilities of existing tools.

After initial testing, research and development became focussed on evaluating a range of key physics engine parameters that control simulation fidelity, but do not correspond to real-world values. These parameters have been adjusted to see which values produce the highest straight line speeds and most accurate derailment behaviour on curved track, whilst also attempting to find a compromise between performance and accuracy. There are also design choices, such as the wheelset design variations described in Section 3.2.2, where it is unclear which choice will maximise the flexibility and stability of the simulation, and this Thesis presents a comparison between these design choices. Also presented is the development of a new real-time simulation technique, based on a real-world phenomenon and designed to improve the simulation of the wheel/rail interface.

1.6 Evaluating the Simulation Tool

A key aim of this research was to evaluate the capabilities of the physics engine and to determine whether it is suitable for the simulation of rail dynamics. This was achieved by using data collected from the Locomotion tool to evaluate its strengths and weaknesses, and to determine the sort of testing for which it may be suitable. This includes attempts to determine whether the tool is capable of simulating the wheel rail-interface and the dynamic derailment mechanism, as well as other aspects of rail vehicle dynamics, such as gauge testing, and its suitability for use as a rapid prototyping tool.

Limited validation data was available for use in the evaluation of the simulation, but the behaviour of simple objects has been evaluated using traditional mathematics and the Nadal Limit for Wheel-Climb Derailment was used to make predictions about derailment behaviour. These predictions have been used to evaluate the simulation, producing promising results (as discussed in Chapter 5).

A real-time system will always introduce error and this research was intended to determine the error bound of the data produced by the simulation and, in cases where error was high, to attempt to constrain it. This thesis intends to show that, by adjusting the parameters of the engine (rather than modifying the internal workings of the physics engine itself) and by using engineering formulas to augment the system (for example by applying additional forces to the wheels) it is possible to produce results that approximate those produced by other means.

1.7 Contribution(s) of Thesis

This Thesis describes the design and development of a real-time rail dynamics simulation tool based on a Physics Engine, and the subsequent evaluation of that simulation tool for engineering use. It intends to show that a physics engine can simulate aspects of rail vehicle dynamics in real-time and produce meaningful data, if its parameters are adjusted appropriately and if it is augmented with additional forces based on published rail engineering formulas. The key contributions of this research are described below.

1.7.1 *Development of a Real-time Rail Dynamics Simulation Tool*

This Thesis describes the design and implementation of a real-time rail dynamics simulation tool, developed using PhysX and other development tools from the video games industry, and includes a description of how the physics engine was configured and integrated into the application. This simulation tool represents an alternative approach to the simulation methods used in traditional engineering tools and, while less accurate, is significantly faster and capable of producing realistic results, with a certain error bound. Preliminary investigations suggested that the development of such a simulation tool for use in the rail industry has not been attempted before.

1.7.2 *An Evaluation of a Physics Engine-based Tool for Engineering Use*

This Thesis presents an evaluation of the simulation tool. The aim was to determine if the physics engine is capable of producing suitably realistic results, as well as determining the error bound of the results and constraining the error where possible. This evaluation includes tests using simple objects (such as cubes), as well as wheelsets, bogies and full vehicles.

One facet of the research involved evaluating a range of simulation design choices and physics engine parameters, to evaluate their impact on the system and to attempt maximise the running speed of the vehicle, in order to make the simulation useful to rail engineers. Other testing focusses on the simulation of the wheel/rail interface, and shows that the simulation is capable of producing results that approach those predicted using traditional methods. Results are evaluated using traditional mathematics and the Nadal Limit for wheel-climb derailment, a widely-used benchmark from the rail industry. There have been studies into the accuracy of physics engines, and their suitability for other applications (some of which are discussed in Section 2.4.4), but not in a rail engineering context.

1.7.3 *Improving the Simulation of the Wheel/Rail Interface*

Initial testing showed that the physics engine was unable to model the wheel/rail interface correctly using its default parameters, and this thesis describes how the PhysX engine was iteratively configured and tested by adjusting a range of engine parameters to improve the fidelity of the simulation and produce more realistic results. It was ultimately necessary to develop a new technique to improve the results further, and this thesis presents the description of that new real-time technique; an additional corrective force that uses a spline-based approach and published rail engineering equations, which is shown to further improve the results.

1.7.4 *A Simulation of Multi-Vehicle Train Behaviour*

Locomotion is capable of simulating multi-vehicle trains in near-real time, something which would be prohibitively expensive using traditional tools. It was intended that this behaviour would be studied in more detail, and a number of features were developed to enable multi-vehicle tests, but time constraints prevented a full evaluation of these features. However, sample data from multi-vehicle tests is included in Chapter 6 as examples of the capabilities of the tool.

1.7.5 *Alternative Applications*

This thesis also presents an evaluation of the simulation tool's suitability for alternative applications, in addition to the simulation of the wheel/rail interface. This includes an evaluation of its functionality as a rapid prototyping tool and as a gauge testing tool.

1.7.6 *Discussion of Possible Applications of a Real-time Simulation Tool*

Based on the data collected, the potential applications of such a real-time tool are discussed (in Chapter 7), including its potential for use in the areas of rapid prototyping, flange collision tracking, gauge testing and analysing derailment behaviour.

1.8 Thesis Outline

This thesis consists of seven chapters:

- Chapter 2 contains background material and related research in the fields of rail vehicle dynamics, engineering simulation and real-time physics engines.
- Chapter 3 presents the design of the 'Locomotion' simulation tool, including the integration of PhysX, the design of the virtual vehicle and the testing environment, and the new real-time wheel/rail interface simulation technique.
- Chapter 4 describes the implementation of the simulation tool, including a description of some of its key features, a discussion of issues that were encountered during its development and any changes that were made to the design.
- Chapter 5 presents an evaluation of the simulation of the wheel/rail interface in the Locomotion tool. This includes an evaluation of the effect of altering various Physics Engine parameters, as well as a range of design decisions and the new wheel/rail interface simulation technique mentioned in Section 1.7.3.
- Chapter 6 contains sample data from tests conducted in the simulation, which are included to show the sort of data that the tool is capable of producing. This includes testing of multi-vehicle trains and an evaluation of the suitability of the tool for use as a rapid prototyping tool and as a gauge testing tool.
- Chapter 7 presents the conclusions of this research and a discussion of the results presented in Chapters 5 and 6. There is also a discussion of further research that might be conducted, how such a simulation tool might be further developed and how it might be used in the rail industry.

Chapter 2

Background and Related Work

This chapter includes background information and related research in the fields of Rail Vehicle Dynamics, Rail Simulation and Real-time Physics Engines. It introduces some key concepts of rail dynamics, and describes how traditional mathematical modelling is used in existing engineering simulations. Real-time physics engines are then discussed, including their applications inside and outside of the games industry and how they can be used, as part of a different approach to traditional methods, to produce real-time engineering simulations.

2.1 Rail Vehicle Dynamics

This section introduces some of the basic concepts of Rail Vehicle Dynamics that are pertinent to the development of the Locomotion simulation tool, and introduces key terminology that is used in the rest of this thesis.

2.1.1 *Rail Vehicles*

A train is a complicated, dynamic system comprising multiple bodies with many degrees of freedom. Figure 2.1 (below) illustrates a generic passenger train.

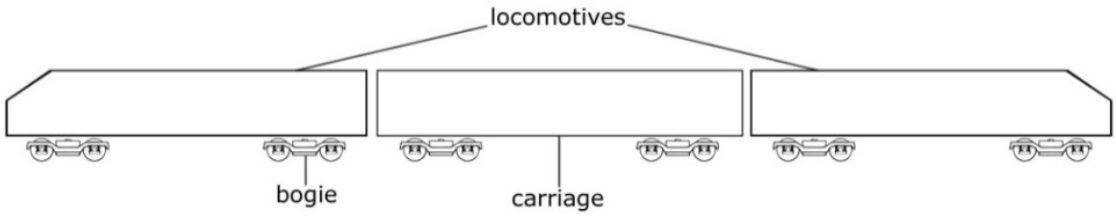


Figure 2.1 - A generic passenger train, consisting of two locomotives and a carriage.

Trains are used for both freight and passenger conveyance, however this research focusses on passenger vehicles. The simulation tool was designed with the ability to simulate freight vehicles, but time constraints prevented the full implementation or evaluation of such vehicles (for more details, see Future Work (Section 7.4)).

Passenger trains commonly comprise two types of rail vehicle; Locomotives and Carriages. A common configuration for passenger trains includes two locomotives, one at each end of the train, with the rest of the train consisting of carriages. Locomotives are vehicles with motorised wheels and often do not have their own payload capacity, their function being to pull (or push) the train along the tracks.

Alternatively, the term 'locomotive' may (as it does in this Thesis) describe a vehicle, also known as a 'motor coach' or 'multiple unit vehicle', which provides both drive and capacity for passengers - commonly used on smaller passenger trains and metro vehicles. Carriages are for passenger conveyance and their wheels aren't commonly motorised (although some trains - such as the 'Bullet Train' that operates on Japan's Shinkansen network [5] - do have motors on each carriage to enable the train accelerate to higher speeds).

Couplings and Buffers

The vehicles that make up the train are joined together by couplings. An example of a rail coupling between two vehicles is illustrated in Figure 2.2, below.

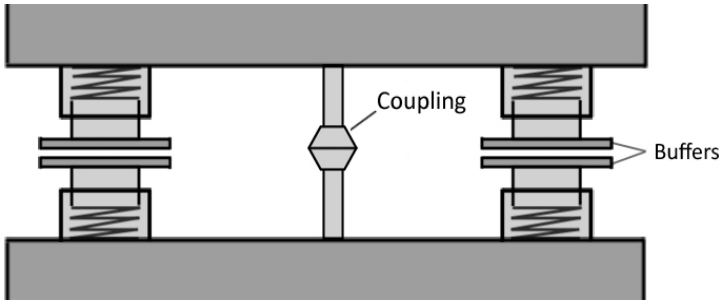


Figure 2.2 - A generic automatic coupling between two vehicles (with buffers)

- *Couplings*

The coupling, as well as connecting two vehicles together, acts as a pivot and allows for relative moment between the connected vehicles, which is necessary to allow them to corner correctly. An example of a coupler commonly used in Europe is the Scharfenberg Coupler [6].

- *Buffers*

Many rail vehicles also have buffers; shock-absorbing pads that limit the slack between the vehicles and lessen any shocks, though they are less common on modern passenger vehicles.

2.1.2 *Bogies*

Bogies are the assemblies to which a rail vehicle's wheels, motors and suspension are attached. The most common passenger vehicle designs use a pair of two-axle bogies on each vehicle [7], as illustrated in Figure 2.1. A simplified diagram of a rail bogie, showing the key components, is shown in Figure 2.3 (overleaf) and a photo of a real-world bogie, an Italian ETR-5000 Bogie, is shown in Figure 2.4 (overleaf).

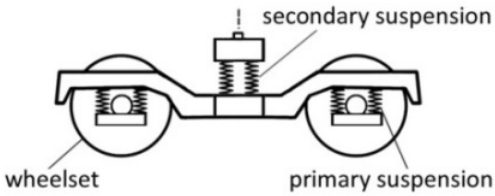


Figure 2.3 - Labelled diagram of a generic rail bogie

Figure 2.4 - An Italian ETR-500 Bogie [7]

The bogie is attached to the chassis of the rail vehicle via a central pivot, allowing it to rotate as the vehicle corners and helping to reduce track forces. On straight track the bogie joint is designed to resist yawing motion for improved stability [8].

Suspension

For passenger bogies, two wheelsets are generally mounted to a rigid H-shaped frame that splits the suspension into two stages. The primary suspension transmits forces from the wheelsets to the bogie frame and its main functions are to guide the wheelsets and to isolate the bogie frame from dynamic loads produced by any irregularities in the track. The secondary suspension transmits forces from the bogie frame to main body of the vehicle and provides a reduction in the dynamic accelerations acting on the body, which helps to improve passenger comfort [7].

2.1.3 *The Wheel/Rail Interface*

The interaction between the wheels and the rails is a key area of rail vehicle dynamics. It is important in determining the behaviour of the train and, while designed to improve vehicle stability and cornering performance, can lead to behaviours that could cause derailment or damage to infrastructure. The wheel/rail interface is therefore a major area of research and development for rail engineers and modelling it has been a key challenge in the development of the real-time simulation tool presented in this thesis.

The mathematics of the wheel/rail interface are discussed in Section 2.2.

Wheelsets

There are many designs of wheelset, but they all have two common features: the wheel profile and a rigid connection between the wheels through the axle [8]. An illustration of the interface between wheelset and rails, showing the key components, is shown in Figure 2.5 (overleaf).

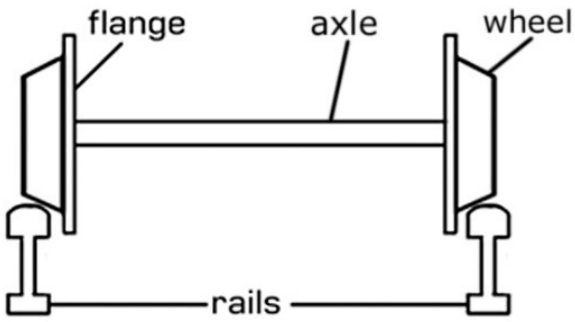


Figure 2.5 - The Wheel/rail Interface (front view)

- *Wheel Shape*

According to The Handbook of Rail Vehicle Dynamics [8], the earliest railway wheels were cylindrical and ran on flanged rails, but it was later discovered that adding a small amount of conicity (exaggerated in Figure 2.5) enhanced the guidance of the wheels around curves in the track. Adding the flange to the wheel instead of the rails also reduces the materials and costs involved in constructing the railway.

Figure 2.6 (below) shows a wheelset with exaggerated conicity, and demonstrates how the effective radius of the wheels (blue) changes with lateral displacement.

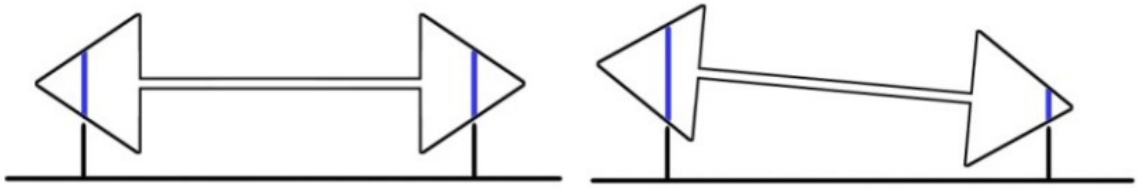


Figure 2.6 - Illustrating the rolling radius (blue) of a centralised (left) and laterally offset wheelset (right)

When the wheelset is centred on the track (Figure 2.6, left), the effective radius (known as the 'rolling radius') of each wheel (blue) is the same. As the wheel drifts away from the centre of the track (Figure 2.6, right), the rolling radius of the outer wheel increases and the radius of the inner wheel decreases, changing the effective size of the wheels at the point of contact with the rails. Since the wheels are rigidly joined by the axle, the outer wheel will travel further, causing the wheelset to follow a curve and/or to push the wheelset back towards the centre of the rails. This self-centring mechanism allows cornering without unnecessary wear-and-tear on the rails or flanges, which can be dangerous and expensive. The wheels are therefore deliberately spaced to allow some lateral displacement before flange contact occurs, as shown in Figure 2.5.

➢ *Wheel Profile*

Some modern wheel profiles are not purely conical but are instead constructed from a series of radii that approximate a part-worn shape. This is intended to give a more stable shape and to prevent the changes in conicity that may occur as a conical wheel profile wears over time [7]. Figure 2.7 (below) illustrates the key elements of the wheel profile:

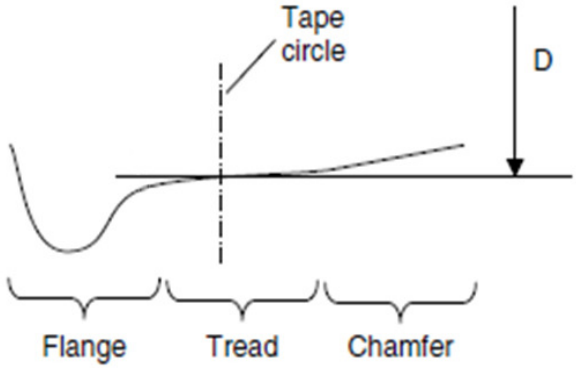


Figure 2.7 - The main elements of a Wheel Profile [7]

The tread is the main contact surface between the wheel and the rail. The tape circle is the position of the contact point when the wheelset is centred between the rails and is the point at which the diameter of the wheel (D) is measured. Some wheels also have a chamfer towards the outer edge, which is designed to lift that part of the wheel off the rail and ease its motion on switches, level crossings etc. Flanges are added to the inner side of the wheel in order to prevent derailment and guide the wheelset when the lateral offset is high. This flange contact is not desirable, but is necessary for times when lateral forces acting on the wheelset exceed the centring forces.

The width of the profile is typically 125 - 135mm and the height of the flange is typically 28 - 30mm. The flange inclination angle is normally between 65 and 70°. Up to 10mm of flange clearance (between the flange and the rail) is commonly allowed [8].

➢ *'Conicity' vs 'Effective Conicity'*

A key term that is used to describe the shape of a wheel is its 'conicity'.

- The 'conicity' of a wheel is based on the slope of the wheel (i.e. 1:20). The conicity of wheels with more complex shapes is often represented as an average of the conicity across the wheel profile [7].
- The 'effective conicity' of a wheelset is the difference in conicity between the two wheels (i.e. it is zero when the wheel is at rest and increases with lateral movement).

Commonly, the conicity of a wheel in the vicinity of the tape circle is 1:10 or 1:20.

Track

In addition to the wheel profile, the shape and properties of the rails are key factors that affect vehicle behaviour. The track can shift or suffer wear and tear over time, but the simulation presented in this thesis is based on the use of rigid body dynamics (described in Section 2.4.2) and assumes that the track is fixed and non-deformable. There are a number of behaviours that can occur on track that is assumed to be rigid, and that engineers would be interested in studying, and it is these scenarios the Locomotion simulation tool was designed to simulate. Some of these scenarios, including Hunting Oscillation and Wheel Climb Derailment are discussed later in this section. In this section, a number of properties of the rail are described that can contribute to these behaviours.

- *Rail Profile*

Modern rails are made of hot-rolled steel and have a cross-sectional profile that typically approximates to an I-beam but is often asymmetrical [8]. Figure 2.8, below, illustrates a typical rail profile.

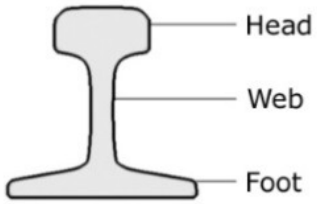


Figure 2.8 - A Typical Rail Profile

The head of the rail is the where wheel/rail contact occurs and is profiled to resist wear and to give a good ride. The foot of the rail is the part that is attached to the sleepers and the web is the connection between head and foot.

- Common rail profile heights range from 145mm to 172mm [9]
- Common rail head widths range from 65mm to 72mm [9]

- *Curve Radius*

Track curves are measured based on the radius (r) at the mid-point between the rails (blue), as illustrated in Figure 2.9, below.

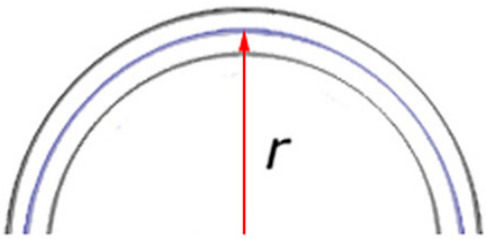


Figure 2.9 - Illustrating Curve Radius (r) - showing rails (black) and the midpoint between the rails (blue).

In the real world, curves generally range from 500 metres up to 1,000+ metres. Smaller radii are less common, but are used in certain locations, such as intracity metro routes and subway lines where limited space is available.

- *Track Gauge*

The gauge defines the width of the track and is measured as the distance between the inner edges of the load-bearing rails [10], as demonstrated in Figure 2.10 (below).

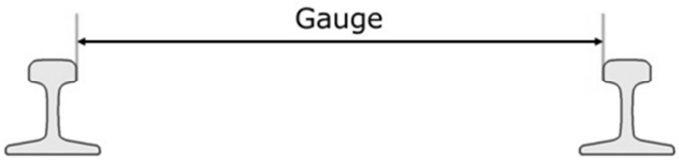


Figure 2.10 - Illustrating Track Gauge

The most common gauge in the world is the Standard Gauge, where the distance between the rails is 1.435m [11], but this can vary between countries and rail operators.

- *Canting*

Most railway organisations tilt the rails inward during bends, so that the inner rail is lower than the outer rail, in order to improve cornering stability. This tilting of the rails is known as canting. Canting redirects the forces at the contact point and helps to neutralise the effect of lateral forces, which in turn reduces wear and improves passenger comfort. The rails are usually canted by a small angle that matches the conicity of the wheel, which in the United Kingdom is 1:20, but cants of 1:30 (for example in Sweden) and 1:40 (in many countries including Germany) are also common [12].

Wheel-Rail Forces

The mathematics of the forces acting on the wheels are discussed in more detail in Section 2.2, but Figure 2.11 (below) shows what are considered to be the main forces acting on the wheel at the point of contact with the rail.

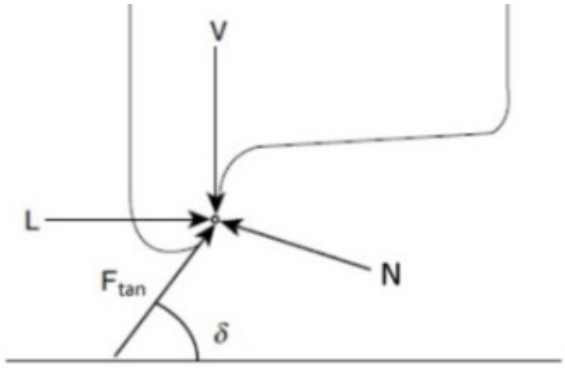


Figure 2.11 - Forces at the flange contact location [13]

These include the lateral (L) and the vertical (V) load, the tangential friction force (F_{tan}) and the Normal Force (N). The figure also shows δ, which is known as the 'contact angle' of the wheel. As this figure illustrates, N is directed inwards - towards the track centre - and is part of the wheelset's aforementioned self-centring mechanism.

Hunting Oscillation

A potential side effect of conical wheels is a phenomenon known as 'Hunting Oscillation', which is caused when the wheelset overshoots its equilibrium and can cause the vehicle to rock from side-to-side, even on straight track. This behaviour is limited by flange contact but can lead to derailment, wear-and-tear or deformation of the track. This oscillation of the wheelset is illustrated in Figure 2.12, below.

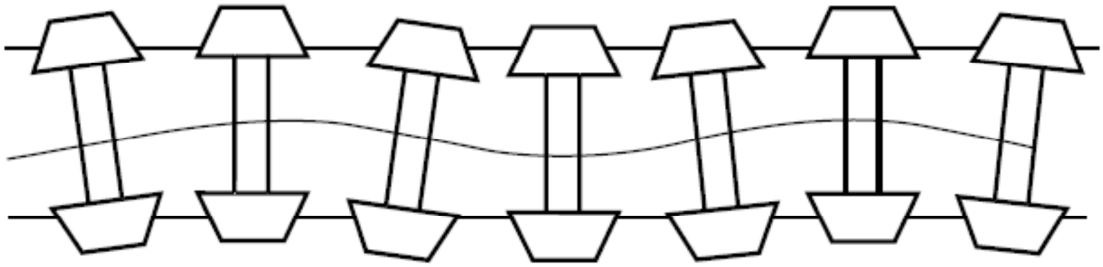


Figure 2.12 - Illustrating Hunting Oscillation of a Wheelset [8]

The speed at which hunting occurs is known as the critical speed and vehicle designers must ensure that the critical speed of the vehicle is above its maximum running speed [12]. The shape of the wheels can be optimised based on the speeds the vehicle is expected to reach and the sharpness of the bends that it is expected to negotiate. Increasing the conicity lowers the curve radius for which perfect curving will be possible, but also lowers the speed at which the wheelset becomes unstable. High-speed rail vehicles have a lower conicity of around 1:40 or 1:50 in an attempt to prevent hunting [7]. If the wheel/rail interface is being correctly simulated in the Locomotion tool, then there may be some evidence of hunting oscillation occurring in the simulation.

Wheel-Climb Derailment

Derailment occurs when a train's wheels run off the rails. Most derailments are relatively minor but can cause delays and damage to infrastructure. There is a type of derailment, known as Wheel-Climb Derailment, that occurs when high lateral forces cause the wheel to climb the rail and, eventually, to derail. This form of derailment can occur even with new, undamaged wheels and rails, and is therefore a form of derailment that a real-time tool, such as the tool presented in this Thesis, could be used to simulate.

Wheel-Climb Derailment is usually the result of high lateral forces, usually caused by a vehicle travelling at too high a speed around a curve, or following an event such as an impact or explosion, and is believed to have been responsible for 16 derailments in the USA alone between 1998 and 2000 [13].

2.1.4 *Gauging*

Another important issue in rail dynamics is gauging. The gauge defines the maximum height and width for rail vehicles to ensure safe passage through the network while avoiding collisions with other vehicles and infrastructure. Gauging systems vary between countries and rail operators and even in rail systems where the track gauge does not vary. Figure 2.13 (below) is an illustration of a gauging failure. It shows an incident in which a Norwegian locomotive became stuck in a tunnel in Kosovo because it was too large.

Figure 2.13 - A Gauging Failure - an incident in which a locomotive became stuck in a tunnel [14].

It is not always simple to estimate the gauge of a bridge or tunnel, however, as the infrastructure may by irregular or may shift over time. Being able to model a rail vehicle (or multi-vehicle trains) and infrastructure such as tunnels will enable gauge testing to be carried out, and this is one possible application for a real-time simulation tool.

2.1.5 *Summary*

This section has introduced some of the concepts and terminology of rail dynamics and the wheel/rail interface that are discussed in the rest of this Thesis.

The next section discusses the mathematics of the wheel/rail interface, in order to demonstrate the mathematical complexity of the problem and to introduce some of the formulas that are used in the existing tools, and that are (or were intended to be) used in the development and evaluation of the Locomotion tool.

2.2 The Mathematics of the Wheel/Rail Interface

The mathematics of the wheel/rail interface are fairly well known. This section presents examples of a few of the formulas that are used in the rail industry, and in the development of existing rail simulation tools. Some of these formulas are used (or were intended to be used) in the development and evaluation of the simulation tool presented in this Thesis, or are included to illustrate the mathematical complexity of the problem and why the traditional engineering approach is so computationally expensive.

2.2.1 *Forces*

This section presents some of the formulas that describe how some of the forces acting on a wheelset are calculated.

Lateral Velocity and Angle of Attack

Figure 2.14 (below) shows a plan view of a wheelset. If the track is considered to be rigid, then the wheelset has two main degrees of freedom: the lateral displacement (y) and the yaw angle (α) relative to the track. This angle, which is sometimes known as the 'angle of attack', contributes to the lateral movement of the wheelset based on a component of its rotational velocity.

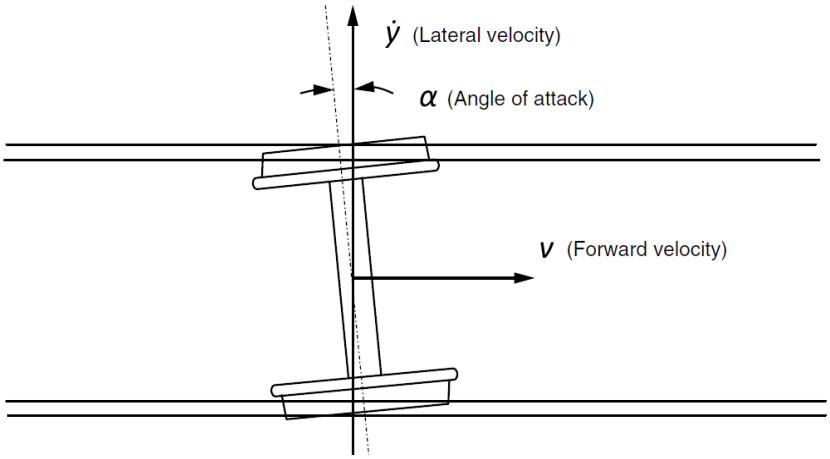


Figure 2.14 - A plan view of a wheelset, showing Forward Velocity, Lateral Velocity and Angle of Attack [13]

The lateral velocity (V_{lat}) of each wheel is given, in its simplest form, by Formula 2.1:

$$V_{lat} = -\omega\, r\, sin(\alpha)$$

Formula 2.1 - Lateral Velocity of a Wheel [13]

Where 'ω' is the rotational velocity of the wheel, 'r' is the rolling radius, and 'α' is the angle of attack. The lateral velocity of the wheelset ('ẏ') can then be calculated as the relative lateral velocity between the two wheels.

Creepages and Creep Forces

The wheels experience a number of different creepages and creep forces, caused by the relative speeds between the wheel and rail. These include longitudinal creepage, spin creepage and lateral creepage [10], all of which can exert a force on the wheelset. One example is the lateral creep force, which occurs as the wheel moves towards flange contact and a force is produced. The direction of the creep force depends on the lateral and spin creepages of the wheel, which are described below.

- Lateral creepage is the yaw angle common to the two wheels.
- Spin creepage is calculated as $sin(\delta / r_0)$, where δ is the contact angle and r_0 is the rolling radius of the wheel.

Figure 2.15 (below) illustrates the three phases of wheel-climb derailment and how the lateral creep force (γ_l) changes in each phase.

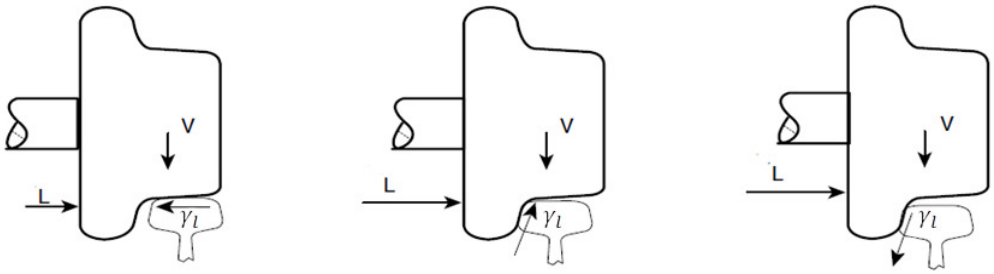


Figure 2.15 - The Three Phases of Wheel-Climb Derailment [13]

In phase 1 (left), the wheel is moving toward flange contact. Phase 2 (centre) is when flange contact occurs and Phase 3 (right) shows the wheel climbing the rail. In phases 1 and 3 the lateral creep force opposes flange climb, but in phase 2 the creep force reverses direction due to the change in the angle of attack and assists the wheel climb process.

The lateral creep force (γ_l) can be calculated based on the wheelset's lateral and forward velocities, as shown in Formula 2.2, below.

$$\gamma_l = \left(\alpha - \frac{\dot{y}}{v}\right) sec(\delta)$$

Formula 2.2 - Wheelset Lateral Creep Force [13]

where α is the angle of attack, δ is the contact angle, v is forward velocity and $\dot{y}$ is the lateral velocity.

The Gravitational Stiffness Force

Section 2.1.3 described how the conical shape of the wheels produces a self-centring effect; the difference in the effective size of each wheel resulting from lateral movement causes the wheelset to yaw about the vertical axis and centre itself between the rails. This effect can also be described in relation to the forces acting on the wheels: an increased lateral offset causes a change in the direction of the Normal force between the wheel and rail, and a component of this force is directed towards the track centre. This is sometimes known as the *Gravitational Stiffness Force* (GSF) [12] and depends on the lateral displacement and roll angle of the wheelset. Figure 2.16 (below) shows a displaced wheelset, along with the angles and forces that contribute to the GSF:

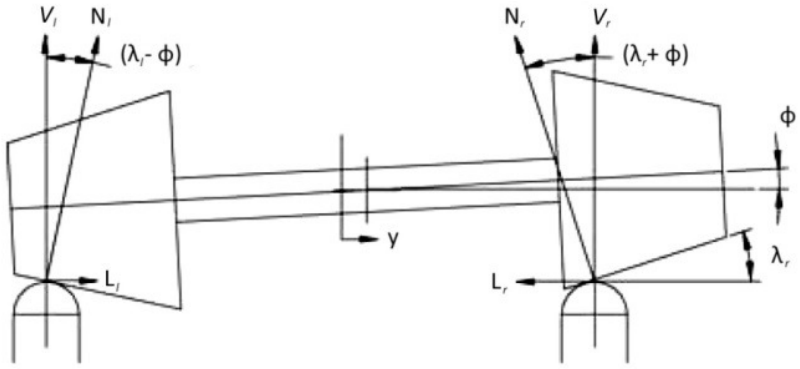


Figure 2.16 - Forces and properties of the wheelset that contribute to the GSF [12]

L_l/L_r and V_l/V_r are the lateral and vertical loads at the point of contact of each wheel, ϕ is known as the roll angle of the wheelset, λ_l/λ_r represent the conicity of each wheel and N_l/N_r represent the normal forces acting on each wheel.

When the lateral displacement is small, the GSF can be calculated by ignoring the differences in the conicity of the wheels [12]. The lateral forces on each wheel can be calculated using the following formulas:

$$L_l = N_l \, sin(\lambda_l - \emptyset) \qquad L_r = N_r \, sin(\lambda_r + \emptyset)$$

The vertical forces are calculated as follows:

$$V_l = N_l \, cos(\lambda_l - \emptyset) \qquad V_r = N_r \, cos(\lambda_r + \emptyset)$$

And so the total lateral force (L_w) can be calculated as

$$L_w = L_r - L_l = [\, V_l \, tan(\lambda_l - \emptyset) - V_r \, tan(\lambda_r + \emptyset)]$$

For small angles, this can be simplified to: $L_w = -W\emptyset$ where $W = V_l + V_r$ (the total vertical load acting on the wheelset).

The roll angle of the wheelset (∅) is calculated using the following equation:

$$\emptyset = \frac{r_r - r_l}{2\,l_0} = \frac{\lambda y}{l_0}$$

Where r_l and r_r are the effective radius of the left and right wheel, λ is the effective conicity of the wheelset and l_0 is half the track gauge.

So the GSF (for small offsets) can be calculated using the formula below (Formula 2.3):

$$L_W = -\frac{W\lambda y}{l_0}$$

Formula 2.3 - Gravitational Stiffness Force [12]

where L_w = Total Lateral Force, λ = effective wheelset conicity, y = wheelset lateral displacement, l_0 = half the track gauge.

Hertz Contact Theory

Determining the dynamic behaviour of the wheelset involves the calculation of the Normal forces between the wheel and rail - and the Hertz contact model is commonly used to solve this. Hertz contact theory is concerned with the contact between elastic, curved bodies, the local stresses and the deformation that is caused. A discussion of how Hertz contact theory can be applied to wheel/rail contact problems is described in 'Applicability of the Hertz contact theory to rail-wheel contact problems' by *Yan and Fisher [2000]* [15].

Kalker's Linear Theory, CONTACT and FASTSIM

Another step in calculating a wheelset's dynamic behaviour is calculating tangent components and creep forces, which are related to the relative speed between the wheel and rail. Kalker proposed several methods to solve the contact problem with models based on the surface description, including his Linear Creep Theory and the 'CONTACT' algorithm (described in Kalker [1982] [16]), before developing the FASTSIM algorithm [17], which is commonly used in the rail industry. The total relative force calculated in FASTSIM, for example, differed only slightly from other implementations and was as much as 15-25% faster.

2.2.2 *Thresholds and the Nadal Limit*

One way to analyse the behaviour of the wheelset, other than simply measuring forces (as in the previous formulas), is through the definition of thresholds and limits. Thresholds describe the minimum conditions required for particular behaviours (such as wheel-climb derailment) to occur and can lead to the derivation of limits to prevent those behaviours from happening (such as limiting the maximum speed of the vehicle on a particular curve radius). One example of this is discussed below.

- *The Nadal Limit*

A commonly-used formula that describes wheel-climb derailment behaviour is *The Nadal Single-Wheel L/V Limit Criterion* [18] (herein referred to as 'The Nadal Limit'). It was developed as a limit to the ratio between the lateral and vertical forces acting on a wheelset, in order to minimise the risk of derailment, and describes the minimum conditions at which wheel-climb derailment is likely to occur. Figure 2.11 (on page 13) shows what Nadal considered to be the key forces acting on the wheel at the contact point. Nadal describes how the ratio between the lateral (L) and vertical (V) load of the wheelset may lead to derailment. The contributing factors are: the flange contact angle (δ), the tangential friction force (F_{tan}) and the normal force (N) [13].

He assumed that there is initially a two-point contact between the wheel tread, the flange and the rail, with the flange contact point located ahead of the tread contact point in the direction of travel, as shown in Figure 2.17 (below). This becomes a single-point contact as the flange climbs the rail and contact with the wheel tread ceases.

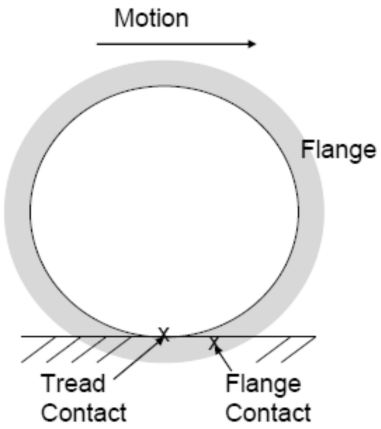


Figure 2.17 - A wheel in two-point contact with the rail [19]

Based on this assumption and a simple equilibrium of the forces at the point of contact, the following equations can be derived (which include the coefficient of friction between the wheels and rails (μ)).

$$N \quad = V\ cos(\delta) + L\, sin(\delta) = V\left(cos(\delta) + \frac{L}{V}\, sin(\delta)\right)$$

$$\left\{ \begin{array}{ll} F_{tan} = V\ sin(\delta) - L\, cos(\delta) = V\left(sin(\delta) - \frac{L}{V} cos(\delta)\right) & when\ (V\ sin\ (\delta) - L\, cos(\delta) < \mu * N \\ F_{tan} = \mu * N & when\ (V\ sin\ (\delta) - L\, cos(\delta) \geq \mu * N \end{array} \right\}$$

From these equations, the L/V ratio can be expressed as:

$$\frac{L}{V} = \frac{tan(\delta) - \frac{F_{tan}}{N}}{1 + \frac{F_{tan}}{N}\ tan(\delta)}$$

As the wheelset travels forwards, the flange contact point is continuously sliding down the rail. Nadal theorised that wheel-climb cannot occur unless the downward motion ceases, when the friction force becomes saturated at the contact point. When the friction force drops below its saturated value, the wheel starts to climb the rail along the flange in a single point contact condition. Therefore, the saturated value can be considered to be the minimum value at which wheel-climb can occur.

If F_{tan} is saturated (i.e. $F_{tan}/N = \mu$), and if the maximum flange contact angle is used for δ, this produces the Nadal Limit criterion, shown in Formula 2.4 (below).

$$\frac{L}{V} = \frac{tan(\delta) - \mu}{1 + \mu\ tan(\delta)}$$

Formula 2.4 - The Nadal Single-Wheel L/V Limit Criterion [18]

Where *L/V* is equal to the value from the right hand side of the equation, this represents the minimum derailment conditions for the wheelset. This formula is used in Chapter 5 to evaluate the derailment speed of vehicles in the simulation tool.

A Conservative Estimate

The Nadal limit assumes that derailment is instantaneous once the limit has been exceeded, whereas it has actually been shown to occur only when this limit has been exceeded for a certain time or distance limit [13]. Wheel-climb generally occurs on curves and the Nadal criterion applies to the outer, flanging wheel and does not consider the effect of the non-flanging wheel. As such, and based on data collected during real-world testing, The Nadal Limit is often considered to be a conservative estimate of wheel-climb behaviour [19] but is still widely used in the rail industry and should produce a suitable benchmark for use in the initial evaluation of the Locomotion tool.

2.2.3 *Mathematical Modelling*

Engineers can use mathematical modelling to analyse and evaluate scenarios by combining formulas, such as those presented in this section, to describe a problem, which can then be solved for a range of input parameters. Such models can be used to predict the forces acting on an object, or to determine under which conditions certain safety thresholds are likely to be exceeded.

The Nadal Limit is one example of how it is possible to predict derailment behaviour mathematically, but in order to make use of this limit it is necessary to calculate all of the lateral and vertical forces acting on the wheelset, using the formulas from Section 2.2.1 (and more). Deriving and solving all of these formulas is a very complex process. Additionally, such a combination of formulas would only represent a single wheelset and wouldn't take into account interactions with other vehicle components, the two-phase suspension system on the bogie, the weight distribution of the vehicle, the effect of vehicle stability on the stability of adjacent vehicles via the couplings, or external factors like wind and air resistance. Deriving and solving a model for even a single vehicle, let alone a multi-vehicle train, quickly becomes extremely difficult and time-consuming.

Also, while it is useful to be able to calculate object forces and properties in particular circumstances (such as the size of the GSF for a given wheel profile and lateral offset), these formulas do not describe the dynamic behaviour of the vehicle, or consider how certain aspects of that behaviour can change over time.

2.2.4 *Summary*

This section has presented a few examples of the mathematical formulas that are used in the rail industry to describe the behaviour of the wheel/rail interface. Some of these formulas are used in the development and evaluation of the simulation tool presented in this Thesis. This is a small sample of relatively simple examples, and barely scratches the surface of the problem. This section was also intended to show why deriving formulas for the wheel/rail interface is so mathematically expensive (and why doing so for whole vehicles and multi-vehicle trains is currently prohibitively expensive).

This section has also discussed how mathematical modelling can be a useful tool for evaluating the forces acting on the wheelset and determining whether those forces exceed certain thresholds. However, these mathematical models do not describe the dynamic behaviour of an object.

The next section shows how these formulas can be converted and implemented as part of a dynamic, computer simulation, as well as looking at examples of the traditional engineering simulation tools.

2.3 Rail Dynamics Simulation and Testing

This section describes some of the testing solutions that are currently used in the rail industry, including how the mathematics of the wheel/rail interface are adapted and used in the existing software simulation packages, and identifies the niche that could be filled by a real-time simulation tool.

2.3.1 *Real-world Testing*

One method used in the development and testing of vehicles and safety features is real-world testing. This involves testing real vehicles, models or materials under controlled conditions, using a range of cameras and sensor equipment to record the results. Alternatively, engineers can study the scene of an accident (or any video footage of the incident that may have been captured) and attempt to determine the cause. These tests can provide engineers with the most realistic and useful data, and most of the mathematical formulas described earlier have been derived in this way.

One example of real-world testing is a blast test performed on the *Metro de Madrid* 5000 series vehicle (which the virtual vehicle used in the Locomotion tool is based on - as described in Section 4.2), which was conducted by *NewRail* as part of the *SecureMetro* project [20]. Figure 2.18, below, is taken from a video of this test.

Figure 2.18 - From a video of a recent blast test on a Metro Madrid 5000 Series Vehicle *[NewRail]*

The data from these real-world tests is invaluable, but they are extremely expensive, often dangerous and cannot be repeated easily (at least not without great expense). They require the use of specialist sensors and recording equipment, which are expensive and may be damaged during the test. They also require the construction of bespoke testing environments and either the use of real life rail vehicles or specially constructed models. Additionally, the scope of these tests is often limited by financial, practical or safety concerns (which, for example, restrict engineers to testing to a stationary vehicle, as was the case in the above blast test, rather than a moving one).

2.3.2 *Virtual Simulation*

An alternative to real-world testing is the use of virtual simulation, which has a number of advantages over full-scale testing;

- Simulation is cheaper than track testing.
- Simulations are repeatable.
- Simulation can be used to test a design before it leaves the drawing board.
- Simulation can explore a range of input conditions that are difficult or impossible to test in real life.
- Simulation can yield information about conditions which would be difficult or impossible to measure in the real world.

Simulation also has an advantage over purely mathematical solutions in that it can produce full 3D, interactive visualisations of the problem.

A Brief History of Rail Simulation

Rail Dynamics as a science didn't really start until the 1960s, when the Research Department at British Rail was established. Before this, rail vehicles were designed (according to Iwnicki (2003) [12]) by a combination of *'evolution and educated guesswork'*. Rail simulations started as research tools, but soon began to find other applications, such as troubleshooting, optimising vehicle designs and, more recently, virtual testing and acceptance. Later, the use of computers and the development of time-stepping integration (discussed in more detail later in this section) allowed complex equations to be solved more rapidly, leading to a better understanding of many elements of rail vehicle dynamics, such as wheel/rail interaction, contact wear and rolling friction.

Current Applications of Virtual Simulation Tools

Current applications of rail simulation software are listed below, and include a number of potential applications of a real-time simulation tool. They are currently used to investigate a range of rail dynamics issues, including [14]:

- Running Stability, Ride Quality and Curving Behaviour.
- The behaviour of the Wheel/Rail Interface.
- Vehicle Acceptance issues, such as low speed flange climbing, high speed stability and derailment resistance.
- Gauging (as described in Section 2.1.4)

Challenges in the Simulation of Rail Vehicle Dynamics

Evans and Berg [2009] discuss the challenges involved in the development of rail dynamics simulations. They say that it is very easy to get wrong answers in dynamic simulations and that, even if the model is correct, the simulation results still depend on selecting the right track input, wheel and rail profiles and friction conditions. Results can also be very sensitive to simulation parameters such as the integration algorithm, timestep or smoothing effects.

Challenges include the modelling of suspension, track models and inter-vehicle connections. Some of these elements can be modelled in real-time and are included in the Locomotion simulation tool, while others, such as suspension are not (more details in Section 6.2 - Future Work).

However, they say that the key issue is validation; there is very little real world data with which to validate simulation results. This has certainly been a key challenge in this research. They also explain that validating against available real-world data is a process prone to error or bias if not done correctly.

'Discretisation' - Adapting the Mathematical Models

Rail simulations are based on a range of engineering formulas, such as those discussed in Section 2.2. Implementing these mathematical descriptions of rail dynamics and wheel/rail interface behaviour in a dynamic computer simulation requires a process of discretisation. Discretisation is the process of turning a continuous problem into discrete sub-problems, which can then be solved more easily. One such technique is the Finite Element Method (FEM) which involves subdividing a problem domain into smaller subdomains, known as finite elements. The combination of the solutions to the simpler element equations of the subdomains results in an approximation of the solution to the more complex equation of the larger domain.

- *Discretisation Example*

Consider the following example, illustrated below (Figure 2.19), in which discretisation is used to calculate the area under a graph.

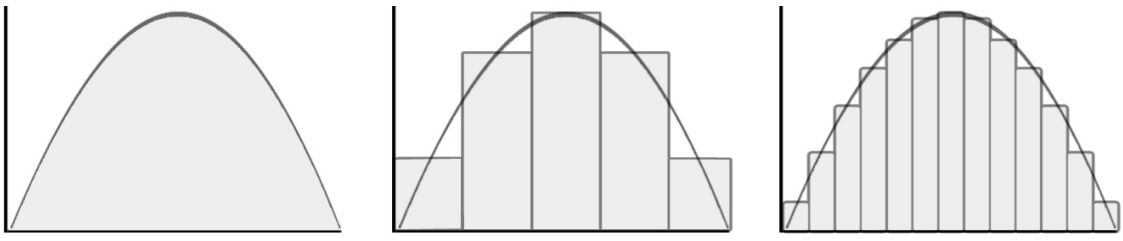


Figure 2.19 - Calculating the area under a graph

A simpler, less mathematically intense solution that calculating the true area under the graph (Figure 2.19 - left) is to create a series of rectangles that are each the same 'height' as the graph. The sum of the area of these rectangles produces an approximation of the area under the graph (Figure 2.19 - middle) and is less complex than the full solution. Such a solution will always introduce a certain amount of error, which can be reduced at the expense of mathematical complexity. In this example, the more rectangles that are used, the closer the approximation will be to the true area (Figure 2.19 - right).

- *Differential Equations of Motion*

In order to simulate the dynamic behaviour of an object, the equations that describe the forces acting on the object and how they cause the state of that object to change over time need to be considered. This requires the creation of complete (or partial) time-dependent differential equations, which can then be solved with respect to time using integration.

Consider the equation of linear motion for a particle (Newton's Second Law): F = ma. This states that the force (F) on an object is the product of its mass (m) and acceleration (a). Acceleration is the rate of change of velocity over time. The equation can therefore be rewritten as:

$$F = m\,\frac{dv}{dt}$$

This formula can be rewritten thus:

$$\frac{\mathrm{dv}}{\mathrm{dt}} = \mathrm{F/m} \qquad \therefore \qquad \mathrm{dv} = \left(\frac{\mathrm{F}}{\mathrm{m}}\right)\mathrm{dt}$$

which allows the difference in the particle's velocity to be calculated based on the timestep (dt), which is an important part of the process of modelling the object's behaviour in a computer simulation.

- *Numerical Integration*

Numerical integration techniques are used to approximately integrate the time dependent differential equations of motion, such as the example above. There are two approaches to integration; implicit and explicit integration. Implicit methods involve calculating a solution based on the current state of the system and a later one. Implicit functions can produce more accurate results using larger timesteps than explicit methods.

However, they are more difficult to derive and require additional computations to solve, making them more expensive than explicit methods. As such, implicit methods are prohibitively expensive for use in problems as complex as the dynamics of a rail vehicle, especially in a real-time context, and so they tend not to be used.

Explicit methods calculate the state of a system at the current time from the state of the system at a previous time and an additional calculation. Most existing simulation tools make use of explicit integration using timesteps that are as small as possible in order to maximise accuracy and minimise error. A real-time system would also have to make use of explicit methods, to avoid compromising the performance of the simulation.

In numerical integration, it is necessary to take finite steps in time, thus dt goes from being infinitely small to some discrete amount (Δt):

$$\Delta v = \left(\frac{F}{m}\right)\Delta t$$

This approximates the change in velocity (Δv) based on the time change (Δt). The position of the object can then be calculated using its previous position and its velocity.

- *Taylor's Theorem*

Integration methods are based on the use of Taylor's theorem, which allows the approximation of the value of a function at a particular point to be calculated by knowing something about the function and its derivatives at an earlier point. The approximation of the function takes the form of an infinite polynomial series, such as the one below.

$$y(x + \Delta x) = y(x) + (\Delta x)y'(x) + \left[\frac{(\Delta x)^2}{2!}\right]y''(x) + \left[\frac{(\Delta x)^3}{3!}\right]y'''(x) + \ \ldots$$

Where y is a function of x, (x +Δx) is the new value of x at which you want to approximate y and y'(x) is the first derivative of y(x), y''(x) is the second derivative of y and so on. [21]

- *Euler's Method*

A simple example of an explicit integration method is Euler's method, which involves calculating the initial conditions (x_0) and then explicitly calculating x_1 to x_n for each timestep (Δt) from t_0 to $t_0 + n\Delta t$. i.e. x_n is calculated based on the value of x_{n-1} and an additional computation. Using Euler's method to integrate the equation of motion above produces the following formula:

$$v(t + \Delta t) = v(t) + (\Delta t)v'(t)$$

Euler's method is known as an order 2 solution, which means that it considers the first two derivatives in the polynomial series and ignores, or 'truncates', the rest. This reduces the computational complexity of the method, but leads to so-called truncation error.

The main advantage of Euler's method is its simplicity. The downside is the amount of error and its lack of stability and accuracy in many practical examples [22]. The simplicity of Euler's method leads to errors that are proportional to the step size used; i.e. a larger timestep results in larger error.

- *Other Integration methods*

Other examples of integration methods include the 'Improved Euler Method', which is an order higher than the standard method - error order 3 - and produces better results at the cost of additional complexity. Improved Euler is calculated using Formula 2.5, below.

$$k1 = (\Delta x)\ \mathrm{y'(x,y)}$$

$$k2 = (\Delta x)y'(x + x, y + k1)$$

$$y(x + \ \Delta x) = y(x) + 1/2\ (k1 + k2)$$

Formula 2.5 - The 'Improved Euler' Integration Method

Other methods include the family of Runge-Kutta methods, which are higher-order methods. One variation of Runge-Kutta is an order 5 solution. The integration formula for this method is shown in Formula 2.6, below.

$$k1 = (\Delta x)y'(x, y)$$

$$k2 = (\Delta x)y'\left(x + \frac{\Delta x}{2}, y + \frac{k1}{2}\right)$$

$$k3 = (\Delta x)y'\left(x + \frac{\Delta x}{2}, y + \frac{k2}{2}\right)$$

$$k4 = (\Delta x)y'(x + \Delta x, y + k3)$$

$$y(x + \Delta x) = y(x) + 1/6[k1 + 2(k2) + 2(k3) + k4]$$

Formula 2.6 - The Runge-Kutta (Order 5) Integration Method

This is a more stable solution than Euler's - meaning that it converges well with the exact solution - and avoids the calculation of partial derivatives in order to solve. [21]

Another popular example is Verlet, which provides higher numerical stability than Euler's - it is 4^{th} order accurate - and, while less accurate than Runge-Kutta, requires fewer calculations per timestep and is reversible, allowing calculations to be performed forward and backward in time.

When building a computer simulation, it is necessary for the developer to select the integration method that is best suited to their aims. Because these simulation tools (and physics engines) are commercial software, it is not always possible to determine which specific method they use. There is a discussion of physics engine methods in Sections 2.4.2 and 2.4.4.

2.3.3 *Existing Rail Simulation Solutions*

A number of programming languages and simulation tools are currently used in the rail industry for simulating various aspects of rail safety and security. A few examples of these software packages are discussed in this section.

MATLAB

A number of the existing simulation solutions and research projects have used MATLAB in their development. MATLAB [23] is a high-level language and interactive environment for numerical computation, visualisation, and programming. MATLAB can also be used for Numeric Computation; Data Analysis and Visualisation; Programming and Algorithm Development; Application Development and Deployment; signal processing and communications; image and video processing; control systems; test and measurement, computational finance; and computational biology. [23]

Other similar software is available, including GNU Octave [24], an open-source alternative to MATLAB used in a number of research fields. Octave is also a high-level programming language, used primarily for numerical and scientific computing, and is useful for manipulating data and other numerical functions. [24]

Existing Simulation Tools

There are a number of rail engineering simulation tools, some based on MATLAB and others using bespoke simulation engines. These include:

- RADIOSS [25] (shown in Figure 2.20, left (overleaf)), LS-DYNA [26] and PAM-CRASH [27], which are used to model structural deformation.
- MADYMO [28], which is used to model passengers in crash conditions.
- VAMPIRE [29] (shown in Figure 2.20, right (overleaf)), which is used to simulate the behaviour of a rail vehicle running on a track (discussed in more detail overleaf).

Many of these tools are general-purpose engineering simulation tools that have been modified to simulate rail vehicles (and related issues such as passenger safety and the deformation of materials). Others are purpose-built rail simulation tools.

Below are screenshots from the RADIOSS and VAMPIRE simulation tools. These show a train crash being simulated in the RADIOSS tool (left) and a train running on uneven track in VAMPIRE (right).

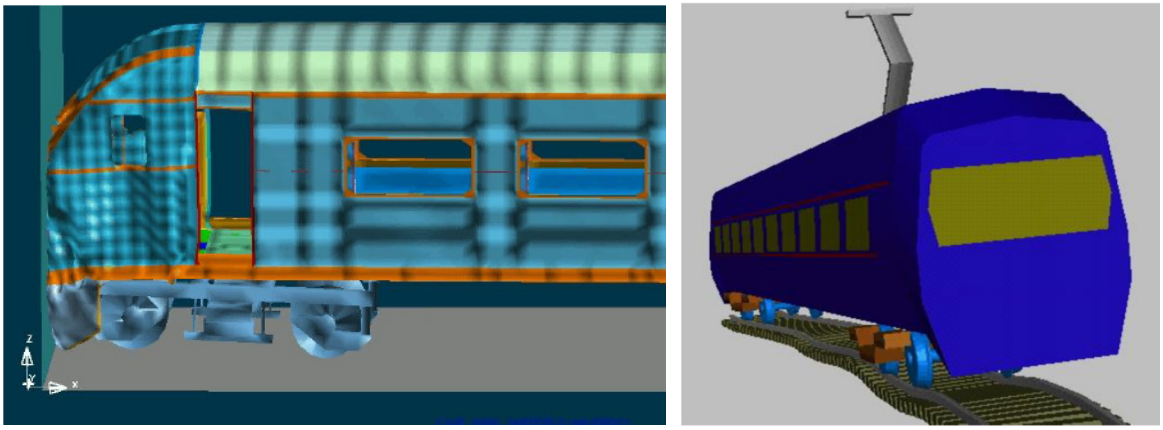


Figure 2.20 - Screenshots from '*RADIOSS*' (left) and *VAMPIRE* (right)

- *VAMPIRE*

VAMPIRE is a special-purpose rail engineering package that is designed to simulate rail vehicles travelling over track, and therefore warrants closer examination. It is probably one of the most relevant software packages to this research as it allows for simulations of entire vehicles, despite not being a real-time solution. It has also been used for a number of similar intended purposes, such as assessing safe operating limits and derailment risk, as well as evaluating vehicle and track designs. VAMPIRE *'allows the user to build a dynamic model of any rail vehicle and study the response of the vehicle to real measured track geometry or user specified inputs in the form of track displacements and external force inputs.'* [29]. Rail vehicles can be modelled with simulated instrumentation, allowing almost any aspect of behaviour to be studied, or for results from real-world experiments/sensors to be replicated.

VAMPIRE is also capable of producing 3D visualisations, including the visualisation of contacts, stresses and forces, as shown in Figure 2.21 (overleaf). These visualisations can be replayed in 3D (including the ability to rewind the visualisation, step through it frame-by-frame and explore the scene using a dynamic camera), but they are produced by the software while the simulation is running and can only played back afterwards. The execution of the simulation and the generation of the visualisation data does not take place in real-time.

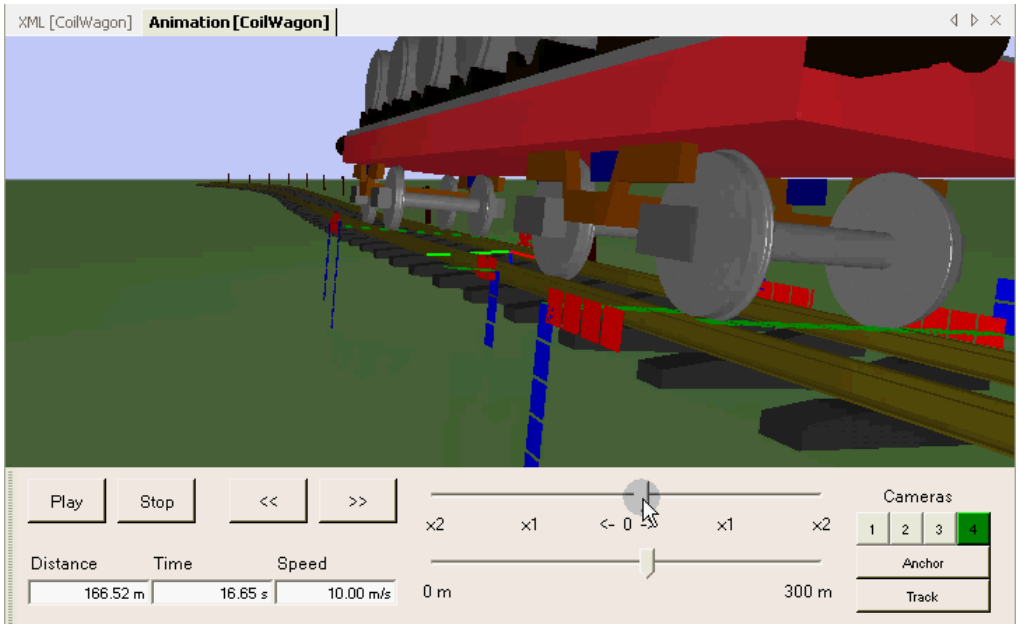


Figure 2.21 - Screenshot from a demo of VAMPIRE [29]

- *Derailment Investigation*

According to a case study available from the VAMPIRE Dynamics website [29], VAMPIRE has been successfully used to recreate the conditions of a real-world derailment. The derailment in question occurred in the Liverpool Central underground station in 2005 (referenced in a Rail Accident Investigation Branch (RAIB) Report [30]). Their analysis, which included a recreation of the incident in VAMPIRE, demonstrated that track forces were not sufficient to derail the vehicle, but showed that material deformation of the rails over time had caused the gauge to widen by 50mm. Further investigation was undertaken to investigate how such incidents could be prevented at similar sites in the future. Interestingly, the simulation showed that lowering the speed of the vehicle actually increased the chance of derailment in this scenario [30].

Limitations of Existing Tools

While it is possible to build up a dynamic simulation of a wheelset, doing so requires the derivation and solving of the time-dependent equations, the application of forces to the wheelset and the calculation of the total lateral and vertical forces acting on the wheelset. By the time lateral velocity, gravitational stiffness, creep forces and all of the other forces acting on each wheel at any given moment are considered, modelling even a single wheelset becomes highly complicated and time-consuming. If such a model was extended to include the dynamics of an entire rail vehicle, the complexity would be immense and it would be prohibitively expensive to solve all of the equations for every part of the vehicle at every timestep.

This mathematical complexity prevents the existing simulations from modelling multi-vehicle trains, dynamic loads or other scenarios that engineers may wish to study, at least not without taking a prohibitively long time to produce their results. Engineers may be willing to wait for several hours (or even days) for safety-critical results in the latter stages of the design process, but this level of accuracy is less critical in the earlier stages and can slow the more-iterative stages of the process.

2.3.4 *Related Work*

There are a number of techniques for calculating contact geometry, forces and properties of the wheel/rail interface that are currently used by rail engineers (many of which are used in the existing simulation tools). The following techniques are examples that are commonly used in the rail industry, along with a number of projects focussed on the development of new, faster techniques (or faster versions of existing techniques). Also included is some research into real-time simulation techniques used in other engineering disciplines.

Faster Techniques

The following are examples of attempts to develop new, faster techniques, or to improve the efficiency of existing ones (including the techniques discussed in Section 2.2.1).

- *Polach*

Polach [31] presents a fast algorithm for the computation of wheel rail forces under known contact geometry, creep and spin conditions. His technique is used in the ADAMS/Rail [32] application as an alternative to FASTSIM and other approaches. It is claimed that this technique has a lower computation time than FASTSIM and is, in some cases, more realistic.

- *Wang & Li*

Wang & Li [33] present a derailment simulation designed to study vehicles travelling at high speed in order to study *'the dynamic derailment mechanism, analyse derailment conditions and influence factors and determine the key cause of the incident'*. The models that make up their solution were developed in MATLAB. They present an improved version of an existing three-dimensional contact trace method for obtaining the contact point, along with a fast, accurate method for measuring the contact force, using a variation of Polach's method, with the inclusion of spin creep moment of the wheel and an improved version of the Hertz contact method. They claim that their solution can obtain the dynamic behaviour of a vehicle on a straight or curved track more quickly and more accurately than other existing methods and they used LS-DYNA to verify their results [34].

- *Anyakwo et al. (2012)*

In Anyakwo et al. [35], they present another method for modelling the dynamic behaviour of the wheel/rail interface. They devised a model that incorporates the wheel-rail contact geometry and other elements using an approach that does not require the solving of complex mathematical equations to estimate the model's critical velocity. They created what they describe as a 'novel, two-dimensional' model, the first phase of which includes the use of the Hertz contact model and Kalker's linear theory to calculate the contact patch and creep forces. Their technique also includes variable timesteps to improve computational efficiency. The mathematical model is implemented in *MATLAB,* using a numerical differentiation method, and is validated against results attained using traditional techniques (such as Kalker's linear model) and the simulated results compared well to the estimates.

Monga et al.

Monga et al. [36] present a new method for designing and implementing real-time simulation algorithms, used to create dynamic vehicle models, based on the use of Field-Programmable Gate Array technology. They produced a high-performance reconfigurable platform to run various simulations and to analyse the results. They successfully compared results from their simulation environment to the normal software-based approach (using MATLAB) and showed that their system was capable of real-time execution. In their tests, the hardware accelerated solutions were twice as fast as standard software simulation.

Conclusions from Related Work

While these solutions may be improved or faster versions of their predecessors, none of them approaches the speed or simplicity necessary to implement in a real-time context and are included as examples of existing solutions used by engineers. The engineering approach is concerned with improving the performance of existing solutions with as little compromise in accuracy as possible, and so the performance gains from these new techniques are limited. Additionally, while Monga et al.'s solution [36] is capable of real-time simulation, it also requires specialised and expensive hardware, whereas the use of real-time physics engines enables the simulation to run on a desktop PC using commodity gaming hardware (as discussed in Section 2.4).

2.3.5 *Summary*

This section has introduced and discussed some of the existing simulation tools used in the rail industry, and some of the techniques that are used to create those tools. It has discussed the usefulness of virtual simulation over real-world testing and identified the limitations of the existing tools; namely the fact that their mathematical complexity causes them to be computationally and temporally expensive to use.

This section has discussed existing numerical programming languages, such as MATLAB and Octave, as well as rail engineering tools like RADIOSS and VAMPIRE. These tools are powerful, but incapable of real-time simulation. This section has also presented a number of 'faster' techniques used in a range of scientific and research software packages; however they are not designed for real-time simulation and do not approach the speed or simplicity necessary to implement in a real-time context.

This research does not focus on trying to create a new, faster simulation tool using existing techniques, or on making the existing techniques faster, but on a new approach using alternative technology; real-time physics engines, which are discussed in the next section, along with how they may be used as an alternative to the existing techniques to produce a real-time simulation tool.

2.4 Real-time Physics Engines

The alternative solution proposed in this thesis is based on the use of real-time physics engines, as used in the video games industry. In this section, physics engines are discussed, along with academic studies into their performance and suitability for other purposes.

2.4.1 *Introduction*

Physics engines are used by games developers to create more realistic virtual environments by adding seemingly realistic physical behaviours to entities in the game world. They are iterative, constraint-based solver systems that prioritise speed over accuracy, in order not to interfere with the gameplay/performance of the game [21]. Physics Engines enable features such as realistic explosions and debris, lifelike character animation and physics-based puzzles, as well as simulating vehicle dynamics in racing games. Physics simulation in games was initially designed to enable certain in-game behaviours and to reduce the cost of manual animation, leading to "perceptually correct" approximations rather than truly accurate simulations. Additionally, forces and physics properties are often exaggerated for effect in video games, but third-party physics engines were developed based on realistic principles from mathematics and engineering, such as Newtonian dynamics and improvements in hardware and software in the last decade have led to improvements in realism and performance of these engines. [22]

NVidia's PhysX engine is the most commonly used in the games industry and is being constantly developed and updated [4]. Havok [37] is also a popular engine, but is not as widely used as PhysX. There is also a range of Open Source Collision Detection and Rigid Body Dynamics Libraries such as Bullet [38], Newton [39] and the Open Dynamics Engine (ODE) [40], but they are also not as widely used or as well-supported as PhysX.

2.4.2 *Physics Engine Simulation*

Physics engines are based on Newtonian mechanics, but discretise dynamics and forces differently from the classical mathematical modelling techniques used in existing engineering simulation tools. Physics engines are designed around the use of Rigid Body Dynamics and their main functions are collision detection and collision resolution. These key physics engine concepts are discussed in this section.

Rigid Body Dynamics

Rigid bodies are non-deformable, polygonal mesh objects that collide and interact with each other based on various parameters, including mass, velocity, acceleration due to gravity, friction and other material properties.

Rigid Body motion describes the motion of an object travelling and rotating through space. For modelling purposes, the motion can be separated into directional and rotational components and the two components can be analysed independently. The centre of mass and the linear and rotational velocities of a spherical rigid body are illustrated in Figure 2.22, below.

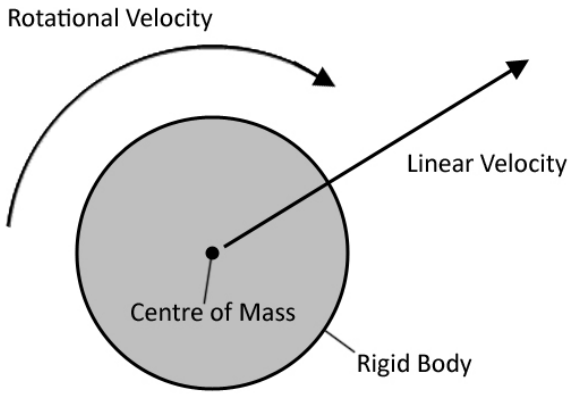


Figure 2.22 - Diagram of a Rigid Body, showing Centre of Mass, Rotational and Linear Velocities

The mathematics of rigid bodies is based around their centre of mass. According to Newton's second law of motion, the movement of an object is a function of its mass and acceleration, and is not affected by its size. The linear and rotational motion of an object, including resulting changes in acceleration and velocity, can therefore be applied to the centre of mass of the object, which moves linearly as if it were an infinitely small particle. Linear and rotational velocity and acceleration values are then applied to the centre of mass, based on integration over time. Forces are often applied to rigid bodies using impulses; an impulse being something that changes the momentum of an object; momentum being the integral of a force with respect to time. The applied forces determine the acceleration of an object, and its velocity can then be updated and its position obtained by integration.

The centre of mass can be adjusted to simulate different properties and could be outside of the object, depending on its composition and geometry. A rail vehicle, for example, where the main body is comparatively light and the main weight is in the chassis and the components underneath it, could be represented by a single rigid body with a lowered centre of mass. This allows the vehicle to be modelled in an abstract fashion, using a single rigid body rather than a series of more complex ones. Similarly, to simulate non-uniform mass distribution (i.e. a carriage that is heavier at the front or back) the centre of mass can be moved forward or backward accordingly. Or to simulate dynamic mass distribution (i.e. passengers moving around the train or fluid cargo), the centre of mass can be adjusted dynamically.

Collision Detection and Resolution

The process of detecting collisions between rigid bodies generally takes place in two phases: broad-phase and narrow-phase. Broad-phase collision detection is generally performed using simple shapes such as spheres and axis-aligned bounding boxes, as these collisions can be solved rapidly and efficiently. Figure 2.23 shows a simplified example of how broad-phase and narrow-phase collision detection may be used for a complex rigid body using simplified collision volumes (grey cuboids).

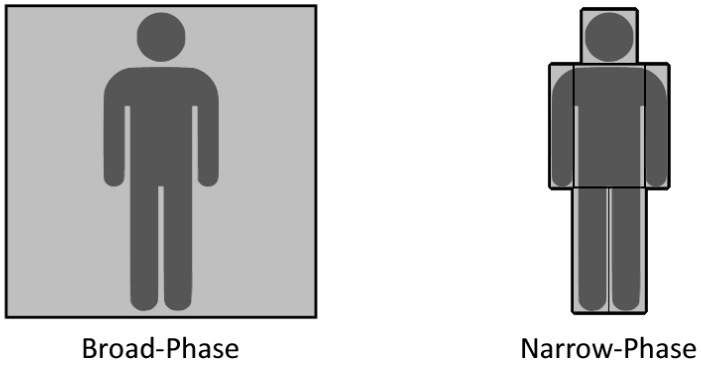


Figure 2.23 - Illustrating broad-phase (left) vs. narrow phase (right) collision detection for a complex rigid body

If collision with these simple objects is encountered, then narrow-phase collision detection will be performed, which typically involves a number of convex polygons that more closely approximate the more complex shape of the object.

Collision resolution then defines what happens to the colliding bodies after the collision has been detected. If a ball were to come into contact with a flat ground plane, for example, a force (equivalent to the normal force (N) between the two objects) is applied to the ball to prevent it from falling, as illustrated in Figure 2.24 (overleaf).

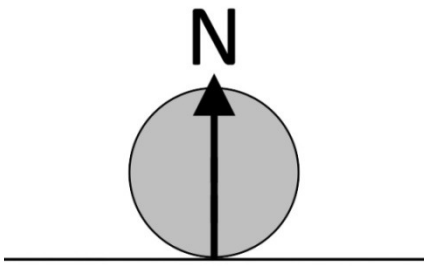


Figure 2.24 - Normal force applied to a ball resting on a ground plane

If the ball comes to rest for a set period of time, then the physics engine will 'put the object to sleep'; meaning that it is not processed further unless the object is involved in another collision, or a force is applied to it, in order to avoid unnecessary computation.

- *Penetration Issues*

The size of the timestep used in the solver can lead to penetration issues. If an object is travelling at sufficiently high velocities and/or the timestep between frames is large, then the two objects may penetrate each other, as illustrated in Figure 2.25 (below).

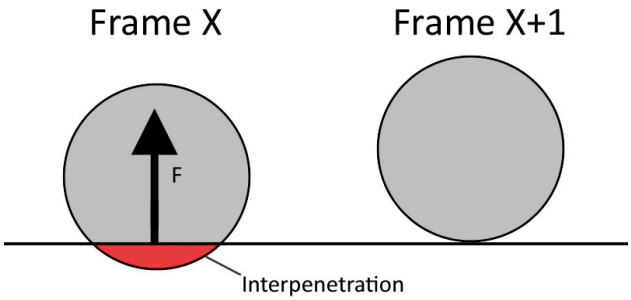


Figure 2.25 - Illustrating how interpenetration between rigid bodies is solved using a separating force (F)

In the event of penetration ('Frame X'), an additional force (F) is applied along the contact normal between the two objects that is sufficiently large as to push the objects apart in the next frame (Frame X+1). This interpenetration solution can lead to instability, however; the combination of penetration error and separation force can lead to visible jittering, if the timestep is too large or the solver is insufficiently accurate. [41]

Constraints

The introduction of limits and constraints is often necessary, in order to allow the computer to simulate rigid body behaviour efficiently and reduce error in the simulation. There are different types of constraint that can be applied in a physics engine, two examples of which are discussed below.

- *Physical Constraints (Joints)*

An unconstrained rigid body has 6 degrees of freedom: 3 positional and 3 rotational. A physical constraint is something that enforces a condition between two bodies, limiting one or more of these degrees of freedom. An example of a physical constraint is a joint. In physics engines, joints are connections between objects. For example, a 'fixed joint' holds two objects together rigidly (eliminating the relative degrees of freedom between the objects), while a 'rotational joint' allows relative rotation between the objects in one axis.

- *Error Bound Constraints*

It is also possible to use Error Bound Constraints, which are limits imposed to minimise error in the simulation. For example you might know that the real-life force applied to a particular object will always be within a certain range (i.e. 1,000 to 10,000N) and so you would limit the size of the force acting on that object to within that range. Such a constraint would mean that if an erroneously large force (i.e. 12,000N) is generated - as a result of solver fidelity, penetration issues or floating point errors, for example - then the force that is applied to the object is constrained to be within that range, reducing the detrimental effects that the erroneous force would otherwise have had on the system.

Integration

Rigid Body dynamics and collision response determine the momentum of an object and integration is then used to process the velocity and position of the object over time, using explicit integration as described in Section 2.3.2. The position of each object in each frame of the simulation is based on the position of the object in the previous frame and on the forces acting on it, including any forces applied during collision resolution, integrated based on the time that has passed between frames. The difference, or 'timestep', between the previous state and the current state determines the accuracy and computational complexity involved - increasing the fidelity by decreasing the timestep increases accuracy and reduces error, but also increases computational expense and may compromise performance. This timestep is commonly limited $1/60^{th}$ or $1/30^{th}$ of a second [41], depending on the target framerate of the game/application.

Because PhysX is a commercial engine, the exact implementation details are unknown. However, according to *Boeing and Braunl (2007)* [42], most of the Physics Engines produce similar results to the second-order Euler ('Improved Euler') or the Symplectic Euler method. Symplectic Euler is a semi-implicit variation of the Euler method (which uses v_{n+1} in the equation for x_{n+1}, while the Euler method uses v_n). [21] The results are not identical to those produced by the Euler methods however, suggesting that a bespoke integration method was used.

2.4.3 *PhysX*

NVidia's PhysX Software Development Kit (SDK) is proprietary, multi-platform, real-time physics engine middleware. It supports a number of platforms including Linux, Mac and Windows, as well as so-called 'seventh generation' consoles like the Xbox 360 and the PlayStation 3 [4]. PhysX is one of the most advanced and widely used Physics Engines in the games industry, and the SDK is available free for academic use [4]. Although not open-source, it is a physics engine that is widely used inside and outside the games industry and has been professionally developed and maintained, which makes the PhysX SDK is a good starting point for developing real-time simulations on a desktop computer.

SDK Version

The Locomotion tool uses version 2.8.4 of the PhysX SDK, which was the latest version that was available at the start of the project (in 2009) and is the last version to still be based on the Novodex engine, which is featured in Boeing and Braunl's 'Evaluation of Physics Systems' [42] (which is discussed in Section 2.4.4).

A Black Box Approach

The version of the SDK used to create the Locomotion tool is non-open-source, which means that it is not possible to access or modify the internal workings of the engine. However, using this version of the SDK still allows the evaluation of the 'off-the-shelf system', as would be used in the games industry. This research therefore treats the internal workings of the physics engines as a black box. The development of the simulation was achieved without making any adjustments to the internal workings of the engine and the evaluation of the tool was conducted by studying the simulation produced by the physics engine, rather than analysing the internal mathematics of the physics engine itself. The engine is adjusted and evaluated using the methods and parameters provided by the SDK (discussed in Section 3.4).

Applications in Gaming

PhysX is commonly used in the games industry and is included in popular game engines such as Unreal and Unity. It has been used in over 500 games, including Batman: Arkham Asylum, Mirror's Edge and the Assassin's Creed Series [4]. It has been used for vehicle simulation in racing games such as NASCAR 2011, Blur and the Need for Speed series [4], and was also used in the Train Simulator games [4] (although these games do not truly simulate rail vehicle dynamics in any meaningful way).

Other Applications

It is also used in non-game-related software, including 3D graphics development environments like DX Studio [43] and 3D Studio Max [44], virtual reality platforms such as Active Worlds [45], Microsoft Robotics Developer Studio [46] and other computer graphics and animation suites.

2.4.4 *Related Work*

With the exception of work in the Games Industry, there seems to have been little research into using Physics Engines for engineering or vehicle simulation. However, there are a number of related projects that have provided useful insight into the development of the Locomotion simulation tool, and examples of a few of these projects are discussed in this section.

Evaluation of Physics Systems

Boeing and Braùnl [2007] [42] developed The Physics Abstraction Layer (PAL), a system that allowed them to evaluate a number of physics engines through a unified interface. PAL interfaces with 10 different physics engines, including: Novodex (on which the PhysX was initially based [4]), the Newton SDK [39], Bullet [38] and the Open Dynamics Engine [40] and allows the user to run the same tests using each of the engines, allowing comparisons to be made between them. They investigated the accuracy and computational efficiency of each engine, including their handling of collision detection, material properties and objects that were stacked, joined or otherwise constrained.

This included an evaluation of the efficiency of the integrator component of the engines. One test involved placing a sphere in the scene and allowing it to drop from gravitational forces. The position of the sphere at each frame in each engine is recorded and the following graph (Figure 2.26 - below) shows the accumulated position errors relative to a mathematically calculated ideal and normalised with respect to the Symplectic Euler integration method.

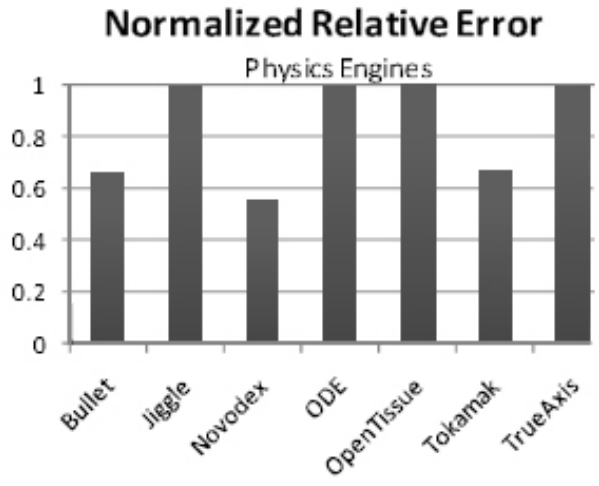


Figure 2.26 - Integrator test data, showing the normalised relative error for each Physics Engine with respect to Symplectic Euler [42]

PhysX/Novodex produced the best results in this test, as it had the lowest normalised relative error.

The integrator was among six 'essential factors' of a physics engine that the authors define as determining its performance. These also included the Simulator Paradigm, which affects the accuracy in resolving constraints, and Object Representation, which contributes to the accuracy and efficiency of collisions. Also tested were Material Properties such as friction and restitution. Another example of the data they collected is shown below in Figure 2.27. Here they were comparing the behaviours of various objects with different 'static friction' values.

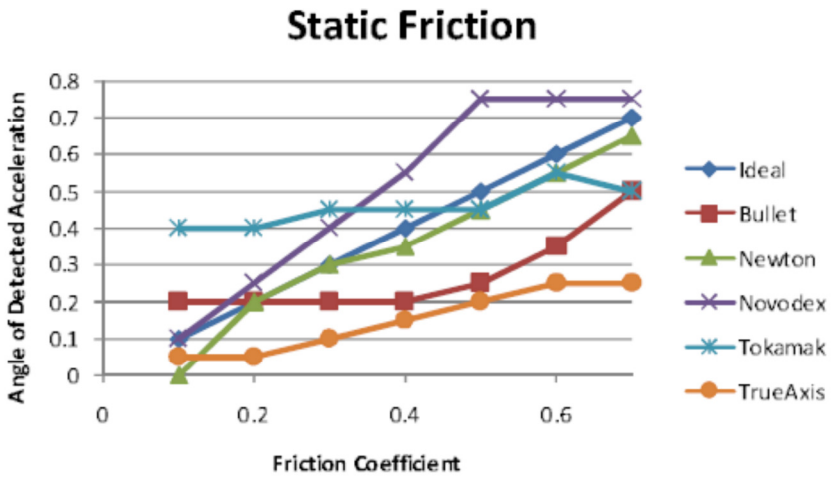


Figure 2.27 - Results from static friction testing in each Physics Engine against the 'Ideal' results [42]

Static Friction is the friction between two objects that are not moving relative to each other, which, for example, prevents an object from sliding down a sloped surface. The angle of detected acceleration for a range of friction coefficient values was recorded and compared to the mathematical 'ideal'. The graph shows that the Newton engine was closest to the 'ideal', but that none of the engines were exactly realistic.

Their conclusion was that all of the engines were suitable for game development, but that no one engine performed best at all tasks. For example, the Newton engine performed well in the static friction test, but produced the worst results in the integrator tests. A different engine performed best in nearly every test and so they concluded that each of the different platforms was better at specific tasks. This makes it difficult for a developer to choose the best engine to use for a specific purpose [42]. However, the fact that Novodex (PhysX) performed well in the integrator test is a promising sign that it might be suitable for use in real-time engineering simulation, as the integrator is the key component in determining the position and velocity of rigid bodies in the scene.

Since PhysX is a commercial engine, the implementation details are unknown; however, as mentioned in the previous section, the authors do state that most of the physics engines that were tested produced results similar to symplectic Euler or 2nd order Euler.

Luo et al

In Luo et al. [47], they showcased a vehicle simulation and debugging environment, built using a commercial physics engine. Their goal was to create a tool to help in the development of intelligent vehicles; small robots that use sensors to look for a predetermined path in the environment and follow it. Figure 2.28 (below) shows the main interface of the simulation environment (left) and one of the intelligent vehicles (right).

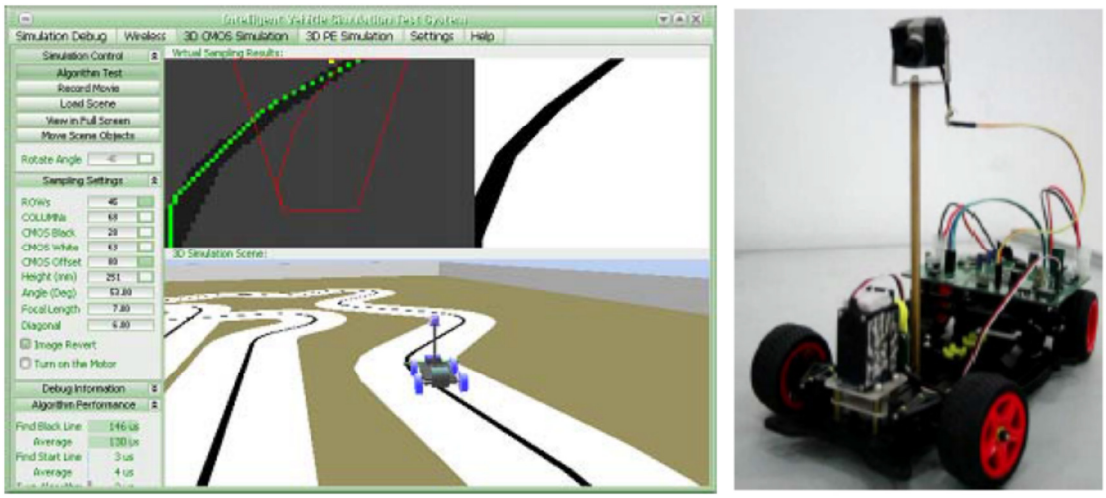


Figure 2.28 - The Main Interface of the CCD/CMOS Vehicle Simulation and Debugging Environment (left) and a photo of an Intelligent Vehicle with CCD/CMOS Sensor (right) [47]

Previously, the development of such vehicles involved 'in-circuit debugging'; downloading the software into a vehicle to test its performance and debug its behaviour. However, the embedded microcontroller software on the vehicle often lacks suitable debugging tools and it is often difficult to analyse the causes of any errors. It is also difficult to develop or debug the software before the associated hardware has been developed.

They created a virtual development environment as an alternative method to develop and debug the software for the vehicles, allowing the software to be developed at a higher speed and lower cost. They used video game technologies to create the development environment, including the OpenGL Graphics Library for 3D rendering and the Open Dynamics Engine for physics simulation [47].

Two robotic vehicles using different types of sensor were successfully constructed using software that was developed and tested in the vehicle simulation and debugging environment. The software was compiled and successfully installed onto the test vehicles, which ran stably on a range of courses at high speeds, showing that (at least on this scale) the physics engine was able to produce suitably realistic results to use for vehicle and software development.

2.4.5 *Summary*

This section has shown how physics engines simulate the dynamics of physical entities, discussed studies into their accuracy and performance, and discussed how they can be used outside the games industry. Game physics may not originally have been designed to produce realistic physical simulations, but physics engines are based on real-world mathematics and may be capable of producing suitably realistic results in certain circumstances. This section has included two papers that evaluated physics engine and shown that they are capable of producing promising results.

2.5 Chapter Summary

This chapter has shown how virtual simulations can be useful to rail engineers and has discussed some of the existing solutions, their limitations and why a real-time simulation could be a useful supplement to these tools. It also discussed how physics engines can provide an alternative approach to the discretisation and modelling of physical objects and how this enables the development of a real-time rail simulation tool.

2.5.1 *The Problem*

Although the mathematics of the wheel/rail interface are fairly well known, the dynamic modelling of this aspect of rail engineering is computationally demanding. Current engineering simulations prioritise accuracy over speed and must solve a large number of complex, bespoke mathematical formulas, which makes the calculations prohibitively expensive for studying the dynamic behaviour of multi-vehicle trains. Rail engineers are aware of these shortcomings and are interested in a more efficient solution that could reduce the time and cost involved in the development of new vehicles and safety features, particularly during the early, more iterative phases of the design process.

Conclusions from Related Research

Many attempts have been made to develop faster versions of existing mathematical techniques used in the rail industry, but none of them approach the simplicity or speed required for use in a real-time solution. The solution presented by Monga et al. [36] allows for real-time simulation, but involves the use of highly specialised and expensive hardware. An alternative solution is therefore needed.

2.5.2 *The Solution*

The solution presented in this thesis, rather than trying to make existing engineering techniques faster, approaches the problem in a different way. The simulation tool presented in this Thesis is based on real-time physics engines from the games industry.

Physics engines are iterative, constraint-based solver systems that are based on Newtonian principles, but use rigid body dynamics and simplified, multi-phase collision detection and response systems as an alternative method for the discretisation and modelling physical objects. The use of real-time physics engines may afford abstracted simulations of entire vehicles and multi-vehicle trains that are reasonably accurate, within a certain error bound, and execute in real-time, and this research represents the first steps in developing and evaluating such a tool for use in the rail industry.

MATLAB was considered as a possible implementation platform, but physics engines were chosen because of their high performance, which should allow the simulation to model more than is currently possible to model using the traditional engineering approach. PhysX was chosen because it is widely used in the games industry and because has been shown to produce reasonably realistic results by studies such as Boeing and Braunl [42]. However, a real-time physics engine introduces a degree of error. The use of explicit integration in the solver is a compromise between accuracy and speed; Rigid Bodies are polygonal, with no true curves, and do not allow for deformation of materials; and collision response does not represent a true contact between two bodies. All of these factors introduce error and this research represents an attempt to evaluate this error, and to constrain it, where possible.

Conclusions from Related Research

Boeing and Braunl [2007] [42] showed that, while different engines performed with differing degrees of success in different testing scenarios, many Physics engines, including a predecessor of PhysX, are capable of producing reasonable approximations of the results predicted by traditional mathematical methods in a range of scenarios, including material properties and rigid body movement. Also, based on Luo et al.'s work [47], it would seem that a physics engine can be used in vehicle development, at least on a small scale. Rail vehicles are much larger and much heavier, but this suggests that physics engines can produce suitably realistic results in certain situations and this research aims to discover if this is also true for rail vehicles.

2.5.3 *Designing the Simulation tool*

The tool will be designed to work in a similar way to most video games, therefore allowing the integration of the PhysX engine (and other game technologies). Additionally, rail engineering formulas from Section 2.2, such as the Gravitational Stiffness Force (GSF) formula, will be used in its development. The research presented in this chapter, and discussions with NewRail staff, have suggested four key areas of rail dynamics and/or possible applications for a real-time simulation tool. Designing Locomotion to achieve these goals and evaluating its ability to perform these functions has therefore been a key focus of this research.

These four goals/applications are:

1. Simulation of the Wheel/Rail Interface (WRI)
2. Simulating Multi Vehicle Trains
3. A Rapid Prototyping Tool
4. A Gauging Tool

In order to achieve these goals/be suitable for these applications, the simulation tool will require:

- Features to evaluate the behaviour WRI - i.e. ways to evaluate the stability, derailment speed and lateral offset of a wheelset
- Features to enable multi-vehicle trains - i.e. design abstractions to allow the simulation of entire rail vehicles and multi-vehicle trains in real-time.
- Flexibility - The tool was designed to be flexible in order to allow the testing in Chapter 5 and 6 to be conducted, as well as to fulfil the intended application as a rapid prototyping tool.
- Gauge testing features - i.e. features designed to detect collisions with bridges/tunnels and other infrastructure l.

The other goal of this research is to maintain the real-time performance of the simulation. The tool was therefore designed to use abstracted models to maximise performance, and the performance (measured as the framerate (frames-per-second) of the tool will need to be measured.

2.5.4 *Evaluating the Simulation Tool*

Limited benchmarking data was available to use in the evaluation of the simulation tool, and so the following methods are used.

Tests were first performed with simple objects that can easily be verified with traditional mathematics (i.e. simple scenarios that allow the normal force between contacting objects to be calculated, and simplified gauge testing scenarios are also used, so that predictions for collisions with infrastructure can be made easily).

The Nadal Limit for Wheel-Climb Derailment is used to evaluate the wheel/rail interface, by predicting the derailment behaviour of a single wheelset or a bogie. Despite the fact that the Nadal Limit is considered to be a conservative estimate, it is widely used in the rail industry and provides a sufficient approximation of derailment behaviour for use in the initial benchmarking of the simulation tool.

[BLANK PAGE]

Chapter 3

Design

This chapter describes the design of 'Locomotion'; a real-time rail vehicle dynamics simulation tool developed using NVidia's PhysX engine. First, the aim and main goals of the project reiterated, and then key elements of the simulation's design are presented, including the vehicle, environment and software design.

3.1 Aim and Goals

In this section, the aim of this research is reiterated and the key goals are discussed in more detail. These goals informed the design of the simulation tool.

3.1.1 *Aim*

The main aim of this research was to design, develop and evaluate a rail dynamics simulation tool based on a real-time physics engine.

3.1.2 *Goals*

The aim can be broken down into a series of distinct goals, which are as follows:

1. To develop a real-time rail dynamics simulation tool based on a physics engine.
2. To determine the error-bound of the simulation in order to appraise its usefulness to engineers.
3. To constrain the error - by introducing constraints, adjusting physics engine parameters or by applying additional forces, where necessary/possible.
4. To produce a tool that is flexible, to enable it to be evaluated easily and to allow it to be used as a rapid prototyping tool.
5. To evaluate the ability of the tool to simulate the wheel/rail interface.
6. To attempt to simulate the dynamic behaviour of multi-vehicle trains.
7. To attempt to simulate other aspects of rail vehicle dynamics, such as gauging
8. To use results from the simulation to discuss whether such a real-time tool would be useful to rail engineers, and for what purpose(s).

3.1.3 *Intended Applications*

In order to properly supplement the existing simulation tools, the simulation design should consider the following potential applications. The simulation tool's suitability for these (and other) applications will be determined by the testing presented in Chapters 5 and 6, and discussed in Chapter 7.

Rapid Feedback for Design Changes

The tool should be flexible, as well as fast, to help to simplify and expedite the process of testing design changes. It should have parameters and settings that can be edited easily, and allow vehicle components to be adjusted easily, without requiring any code recompilation, so that it can provide rapid feedback about the effects of those changes.

A Rapid Prototyping Tool

Similarly, the simulation should allow engineers quickly test a range of scenarios and identify those that are best suited to further investigation, allowing them to make more efficient use of the more sophisticated simulation tools and testing methods. It should therefore have batch testing features and allow a range of parameters to be adjusted and tested automatically, and produce results that show the effect of those adjustments.

A Tool to Analyse Vehicle Stability and Derailment Behaviour

If the tool is to be used to analyse vehicle stability and derailment behaviour, then it should be capable of simulating the dynamic behaviour of rail vehicles during normal operation, at normal operating speeds and on simple rail layouts, as well as during exceptional circumstances.

This includes the following, more specific applications:

- *Simulation of the Wheel/Rail Interface*

The ability to simulate the wheel/rail interface, hunting oscillation and dynamic derailment behaviour would be very useful to rail engineers. This has therefore been a key focus of this research. The aim of this aspect of the research was to see if the simulation is (or can be made) accurate enough to simulate the wheel/rail interface to within an acceptable level of accuracy. The evaluation of the wheel/rail interface will be conducted using the Nadal limit for wheel-climb derailment.

- *Recording flange collisions*

Rail engineers aim to avoid contact between the rails and the wheel flanges, as this causes unnecessary wear and tear, which is potentially dangerous and expensive. The grinding of flange on rail also produces a lot of noise, which is disruptive to people and wildlife. Being able to track flange contacts would allow engineers to identify any areas where these collisions are most likely to occur and the simulation could then be used to find ways to reduce them.

- *Studying the effect of inter-vehicle connections on vehicle stability*

If the simulation is capable of multi-vehicle simulation, then it would allow engineers to study how the stability of adjacent vehicles is affected through the couplings.

- *Determining the maximum safe speed of the vehicle*

Speed limits enforced around the world tend to err on the side of caution and it has been shown that real-world derailments tend to occur at speeds much higher than the posted limits. A real-time tool could be used to show that the vehicles are capable of safely achieving higher speeds, which could lead to limits being increased, improving the efficiency of the railway and increasing its profitability.

A Gauging Tool

Even if the simulation of the wheel rail interface is not sufficiently accurate to study the behaviours described above, the system may be sufficiently accurate to use to study other vehicle behaviours. Another potential application of the tool, if it is capable of simulating entire vehicles, is gauge testing - i.e. determining if the train will collide with other vehicles, bridges, tunnels or other infrastructure.

3.1.4 *Testing Plan (Overview)*

A full testing plan is included in Chapter 4, but an overview of the testing goals of this research is included below. Testing begins with simple scenarios and complexity is iteratively added. Testing with multi-vehicle trains was the ultimate goal, but the focus was on simulating the wheel/rail interface by studying the behaviour of wheelsets and bogies. Each phase of testing is designed to evaluate a particular aspect of the physical simulation and the data collected from these tests should help identify the suitability of the simulation tool for its various intended applications.

Testing Phases

Testing was to be conducted in seven phases, as outlined below.

1. Simple Objects - verify basic object behaviour in PhysX.
2. The Wheel/Rail Interface - evaluate the simulation of the wheel/rail interface - while static and in motion.
3. A Bogie in Motion - test the behaviour of a rail bogie on straight and curved track to evaluate its curving behaviour and operating speed.
4. Whole Vehicles - attempt testing with full vehicles.
5. Multi-Vehicle Testing - if full vehicles can be simulated correctly, test with multi-vehicle trains.
6. Rapid Prototyping - tests to demonstrate how Locomotion may be utilised as a rapid prototyping tool.
7. Gauge Testing - testing to show how Locomotion could be used as gauge testing tool and to evaluate its suitability for that purpose

Testing phases 1 - 3 are presented in Chapter 5. Phases 4 - 7 are presented in Chapter 6.

Additional Objectives

The following additional objectives were to be considered throughout the testing process.

- Identify Error - to attempt to evaluate the error bound. An amount of error is inevitable with a real-time solution, but if the error is small enough, then it may be acceptable.
- Constrain Error/Improve the Results - If any of the behaviours are not realistic or the error is too high, then attempts should be made to improve them, such as by adjusting parameters of the Physics Engine or augmenting the simulation with additional forces.

3.1.5 *Sample Case Studies*

In order to assist in the design and implementation of the tool, the following case studies were considered.

Case Study 1: Curve Derailment Speed

The following case study was designed to evaluate the derailment speed of the vehicle on a particular curve radius.

- Test 1: Start at a Target Speed of 1mph
  [Increment the target speed at the end of each loop until derailment occurs]
- Test 2: Restart at a Target Speed of 1mph
  [Increment the target speed at the end of each loop until derailment occurs]
- Test 3: Restart at a Target Speed of 1mph
  ...
- Test 10: Restart at a Target Speed of 1mph

Ten tests are conducted, allowing the average result and range of results to be collected, which allows an evaluation of the accuracy and consistency of the results.

Case Study 2: Bogie Wheelbase

This case study is designed to test whether Locomotion is suitable for use as a rapid prototyping tool. The target speed is set to 100mph, and the average derailment speed is measured for each bogie wheelbase value, allowing the effect of altering the wheelbase to be evaluated.

- Batch 1: Run 10 Tests using a Wheelbase of 1.5m
- Batch 2: Run 10 Tests using Wheelbase = 1.6m
- Batch 3: Run 10 Tests using Wheelbase = 1.7m
  ...
- Batch 11: Run 10 Tests using Wheelbase = 2.5m

Data Collection and Evaluation

During each test/batch of tests, properties such as the lateral offset of the wheelset, wheelset stability, the number of derailments and the average derailment speed are recorded, along with the performance of the tool, allowing the results to be compared and any changes to be observed.

Summary

These case studies should be indicative of the majority of the tests that will be run during the development and evaluation of the Locomotion simulation tool. The tool was designed to allow batch testing to be carried out automatically, without user supervision.

3.2 Design Overview

This section describes key elements of the design of the Locomotion simulation tool, including the test vehicle and testing environment. These elements were researched and designed before the software architecture (described in Section 3.3) was designed.

3.2.1 *The Test Vehicle*

In this section, the construction and properties of the virtual test vehicle are discussed. The aim when designing the virtual vehicle was to create a vehicle that was as realistic as possible, based on real-world data where this was available. The vehicle is based on a real-world rail vehicle: the Metro de Madrid 5000 Series, a photo of which is shown in Figure 3.1, below).

Figure 3.1 - A photo of the Metro de Madrid 5000 Series Vehicle [provided by NewRail]

This metro vehicle is used on the Metro de Madrid [48] network in Spain and was involved in the bombings in March 2004 [49]. It is being studied by NewRail as part of the SecureMetro project [20], as mentioned in Section 2.3.1.

NewRail have provided access to some of the technical specifications for the 5000 series vehicle, which includes the data in Table 3.1 below:

Total Mass	32,000 kg
Body Length	17.4 m
Body Width	2.8 m
Body Height	2.98 m
Vehicle Spacing	0.52 m
Bogie Spacing	11.1 m
Wheelbase	2.2 m

Table 3.1 - 5000 Series vehicle statistics [Provided by NewRail]

These properties of the vehicle are illustrated below in Figure 3.2.

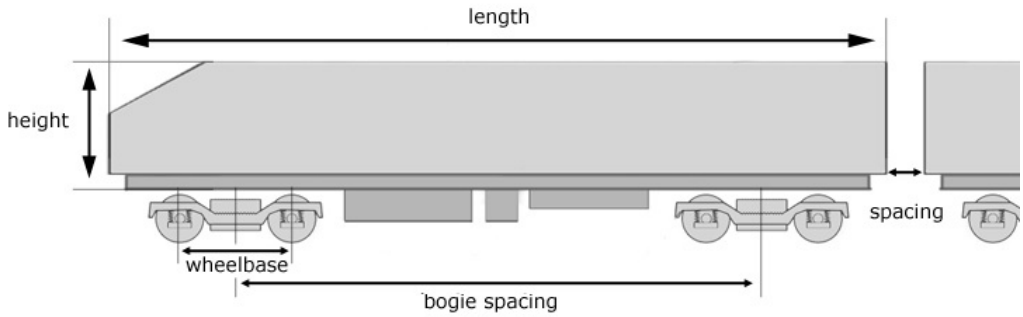


Figure 3.2 - Measurements of the Vehicle

3.2.2 *The Virtual Vehicle*

The virtual vehicle is constructed from a set of components; the body, chassis, bogies and wheelsets. These components, when assembled, produce an abstracted model of the rail vehicle. Figure 3.3 (below) shows these components and the blue arrows show how they are assembled and connected.

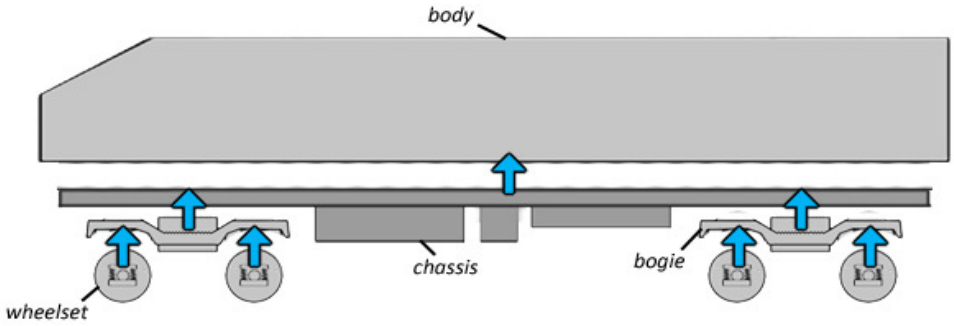


Figure 3.3 - Vehicle Construction in the Locomotion Tool

This use of separate components simplifies and accelerates the file loading process (because each component only has to be loaded into the simulation once) and allows greater control over the way in which the components are assembled. The alternative would be to produce a single model of the whole train, but then the model file would have to be altered (in an external application such as 3D Studio Max) to make any changes (such as adjusting the bogie spacing). By loading the individual components into the simulation, the vehicle can be adjusted more dynamically, making the simulation more flexible. The models are loaded into the application and can then be cloned, positioned and joined together based on parameters that are defined in a text file (and which, by default, are set to the values in Table 3.1), which can be edited without requiring recompilation of the application code.

The Body

The main body of the vehicle is a single rigid body, as shown in Figure 3.4 (below). Its mass and centre of gravity can be altered to simulate different load distributions.

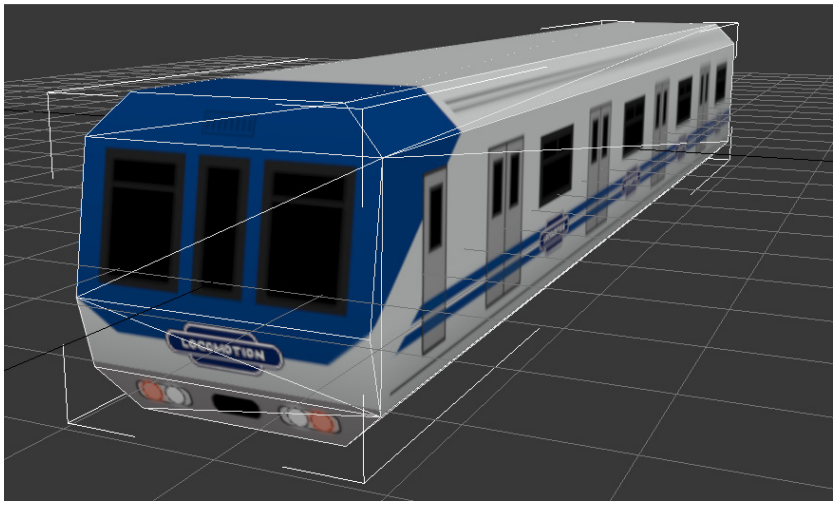


Figure 3.4 - The Locomotive Body Design in 3D Studio Max (3D View)

A texture is applied to create the illusion of windows and doors and make the vehicle look more realistic, but the rigid body is a single shape comprising 21 polygons and 24 vertices. The white lines show the edges of the triangles that make up the polygon.

The Chassis

The chassis of the vehicle represents the underside of the main body, where the fuel, water tank, air conditioning units etc. are attached. The chassis rigid body comprises a number of cuboids, one for the main flatbed shape and four to represent the various components, as illustrated in Figure 3.5 (overleaf). The polygons that make up the chassis have been coloured blue to make them easier to see in the screenshot.

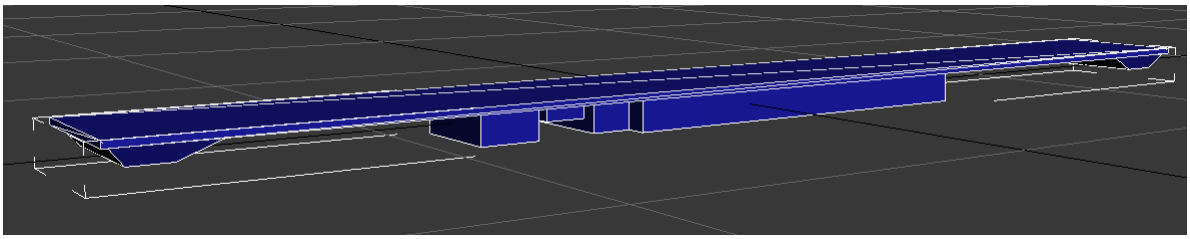


Figure 3.5 - The Chassis Design in 3DS Max (3D View)

The chassis is attached to the main body via a fixed joint. The use of a separate chassis allows for the modelling of different vehicle types by changing only the body. The rigid body for the chassis comprises 46 polygons and 56 vertices.

The Bogie

This simulation uses a fixed bogie setup. The bogie frame is an h-shaped rigid body as illustrated in Figure 3.6, below. The bogie rigid body comprises 96 polygons and 122 vertices.

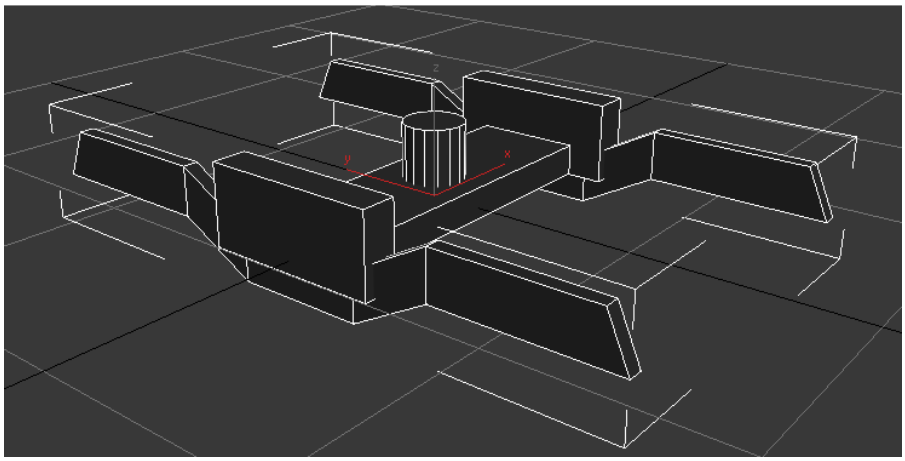


Figure 3.6 - The Bogie Design in 3D Studio Max (3D View)

- *Suspension*

It was decided that suspension would not be modelled as part of the initial implementation of the simulation tool. It may be possible to produce an approximation of the suspension system in real time, but time constraints prevented any implementation during this research (this is discussed in more detail in Future Work (Chapter 7).

The Wheelset

The wheelset includes the wheels, flanges and axle. The axle is a cylindrical rigid body 1.7m in length and with 12 segments. The wheels and flanges are also constructed from cylinders. The wireframe model of one of the wheels is shown in Figure 3.7 (overleaf).

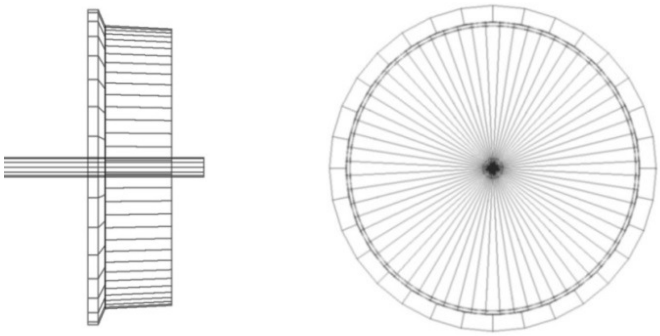


Figure 3.7 - The Wireframe Wheel Profile - Front view (left) and side view (right)

➢ *Wheel Profile*

For the purposes of this simulation, a simplified wheel profile is used that consists of a conical tread and flange (with no chamfer). A simplified profile allows predictions about the behaviour of the wheelset to be calculated more easily. Below are the key properties of the wheels/flanges:

- The wheel consists of a cylindrical shape with a conicity of 1:20.
- The total width of the wheel profile is 135 mm and flange height is 30 mm.

It was intended that more complex profiles would be added to the simulation and tested at a later date, but time constraints prevented their implementation during this research.

➢ *Wheelset Polygon Count*

The most appropriate polygon count for the wheels was not immediately apparent. Three variations were therefore constructed for testing, and are described below in terms of the number of cylinder sections used to create the wheels/flanges and the number of polygons and vertices in the complete wheelset model.

- 32 Segments per wheel/flange - 214 Polygons and 344 Vertices
- 48 Segments per wheel/flange - 310 Polygons and 504 Vertices
- 64 Segments per wheel/flange - 406 Polygons and 664 Vertices

A comparison between the three variations is included in Chapter 5.

➢ *Contact Angle and Conicity*

The contact angle is the angle between the wheel and the ground at the contact point (δ in Figure 2.11 - Forces at the flange contact location Figure 2.11). The contact angle and conicity of the wheels are as follows:

- The wheel has a contact angle of 2.86° (conicity 0.05).

- The flange has a contact angle of 65° (conicity 2.86).

Calculating Conicity

The wheels used in the simulation have simplified conical profiles; the incline does not change across the profile of the wheel. The conicity of each wheel at rest is therefore 1:20. To calculate the effective conicity of the wheelset, consider Figure 3.8 (below).

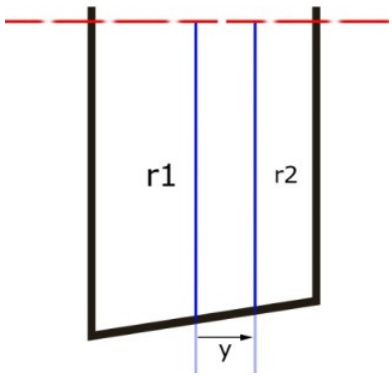


Figure 3.8 - Difference in wheel radius before (r1) and after (r2) a lateral offset of y

Assuming that *r1* is the radius of the wheel at rest and *r2* is the radius when the wheel has experienced a lateral displacement of y, then if *y* = 1 the effective radius of this wheel has decreased and *r2* is 0.05 (1/20) smaller than *r1*

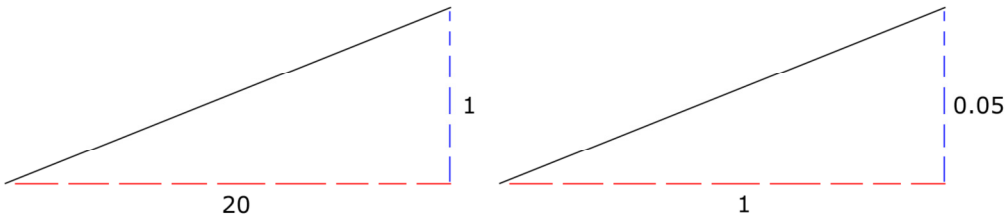


Figure 3.9 - Illustrating the incline of the Wheel

At the same time, the radius of the other wheel will have *increased* by the same amount; 0.05, as illustrated in Figure 3.9 (above). The difference in effective size (known as the Rolling Radius Difference (RRD)) when y = 1 is therefore:

$$\mathrm{RRD}(\mathrm{y} = 1) = \ 0.05 * 2 = 0.1$$

In more general terms, the *RRD* for any lateral offset can be described as:

$$RRD(y) = \ 0.05 * 2\ y$$

The Effective Conicity of the wheelset is calculated using Formula 3.1, below.

$$\lambda(y) = \frac{RRD}{2y}$$

Formula 3.1 - Calculating effective conicity of a wheel from Rolling Radius Difference [50]

However, if the *RRD* is as above, then the multiplication and division by *2y* cancel out, as illustrated in Formula 3.2, below.

$$\lambda(y) = \frac{0.05 * 2y}{2y} = 0.05$$

Formula 3.2 - Calculating the conicity of the simulated wheelset

So the effective conicity of the wheelsets in the simulation is 0.05 and this does not change with lateral offset. Similarly, the effective conicity of the flanges is 2.2. This value can be stored by the simulation during initialisation and need not be calculated when it is needed. However, if a more complex shape was used, this value would have to be calculated as a function of lateral offset.

- *Wheelset Variations*

Two variations in wheelset design were created for use in the simulation.

- *The 'Single Body Wheelset'* or *'SB Wheelset'*- which includes the flanges, wheels and the axle as part of a single rigid body.
- The *'Multi-body Wheelset'* or *'MB Wheelset'*- which comprises separate rigid bodies for the wheels and flanges, which are attached to the axle rigid body via fixed joints.

The Multi-body Wheelset allows the wheel spacing to be adjusted, which allows for different wheel profiles to be used with minimum effort and allows for flange collisions to be detected easily. However, due to the forces acting on the different rigid bodies and the way that the solver that processes the joints between the bodies, the joints may become unstable, making the MB wheelset less stable than the SB wheelset. The two wheelset designs are compared in various tests in Chapter 5.

Conical Wheelset

A third type of wheelset was also constructed for initial testing. The Conical Wheelset is a simple shape consisting of two cones joined at the base, as illustrated in the Figure 3.10 (overleaf). This wheelset is designed to test lateral offset and hunting oscillation behaviour in the simulation by allowing free lateral movement (without impediment by the flanges) in order to evaluate the simulation of the wheel/rail interface.

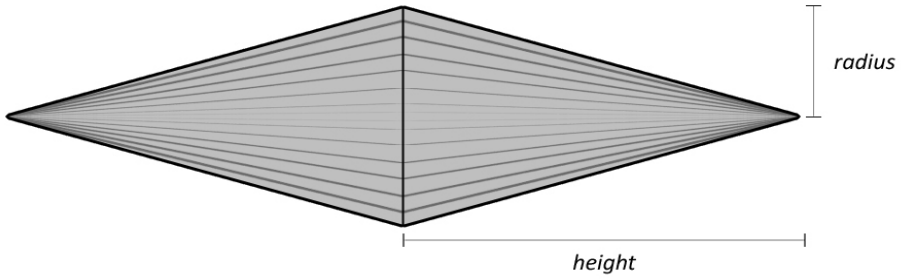


Figure 3.10 - Design for the Conical Wheelset

The cones are created with 64 segments, resulting in a model with 130 polygons and 192 vertices. The radius of the base of the cones is 0.25m and their height is 2.5m. This results in a conicity of 1:10 or 0.1.

Wheelset Propulsion

During tests with a single wheelset, the wheelset is propelled by the application of torque to the rigid body along its local y axis, which causes it to roll forwards.

Bogie Propulsion

In the simulation, the two bogies of the front locomotive are considered to be 'motorised', and the front locomotive pulls the rest of the train - a choice based on common metro vehicle configurations. However, the simulation was designed to enable different configurations by allowing the user to specify whether the bogies are 'motorised' on a per-vehicle basis in the simulation configuration files.

- *PhysX Motor*

PhysX SDK includes a 'Motor' object [41], which can be attached to a revolute joint. A Revolute Joint is created using a Joint Description object, which includes various parameters that allow a motor to be defined and enabled. These parameters are shown in Table 3.2, below, which is taken from the PhysX SDK Documentation [41].

NxReal	**velTarget** The relative velocity the motor is trying to achieve.
NxReal	**maxForce** The maximum force (or torque) the motor can exert.
NxBool	**freeSpin** If true, motor will not brake when it spins faster than velTarget.

Table 3.2 - PhysXMotorDesc (PhysX Motor Description) Attributes [41]

No 'maxForce' is set on the motor, and 'velTarget' is adjusted to control the speed of the vehicle. 'freeSpin' is set to false, so the motor will break to slow the vehicle if necessary.

- *Target Speed*

The user is able to set the 'target speed' of the vehicle and the bogies are designed to accelerate or decelerate (by adjusting the velTarget of the motor on the joint between the bogie and the wheelsets) until the forward velocity of the vehicle reaches the target speed. This target speed parameter can be adjusted dynamically or incremented automatically during batch testing (for more on Batch Testing, see Section 3.3.16).

Train Composition

Trains are created based on a parameter that defines the number of vehicles in the train. If the number of vehicles is set to 1, then a single locomotive is used. If the number is 2, then two locomotives are placed back to back. For any number of 3 vehicles or more, additional carriages are placed between the two locomotives, as illustrated in Figure 3.11, below. This choice was based on common vehicle configurations.

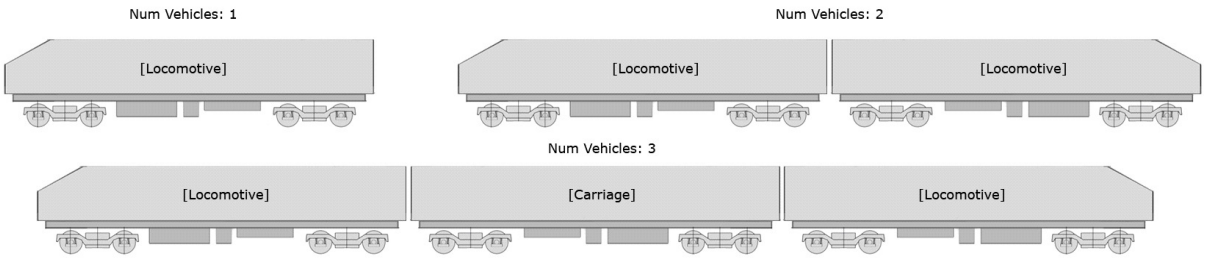


Figure 3.11 - Illustrating Train Compositions (1 Vehicle - top left, 2 Vehicles - top right, 3 Vehicles - bottom)

3.2.3 *Component Masses*

The virtual vehicle is constructed using the dimensions of the 5000 series, and weighs the correct total mass of 32 tonnes. However, the schematics provided by NewRail did not include the masses of the individual components, and so a number of assumptions have been made (these are presented in Tables 3.3. and 3.4, overleaf). These properties are defined in an editable text file.

While these values may not produce a truly accurate representation of the 5000 series vehicle, these values are used in the simulation and the mathematical predictions (see Section 5.5). Since the same values are used in both the simulation and predictions, the results from the simulation should be comparable to the predictions, if the simulation is accurate. If more real-world derailment data was available with which to compare, it would be necessary to adjust these values to make them more realistic.

Locomotive Properties

Table 3.3 (below) shows the values used to construct the locomotive. It shows the mass for each component, as well as the total mass of the vehicle.

Component	Mass (kg)
Wheelset (x4)	500
Bogie (x2)	6,000
Locomotive Chassis	8,000
Locomotive Body	10,000
Total Mass	32,000

Table 3.3 - Estimated Mass of Locomotion Vehicle Components

Carriage Properties

A 'carriage' vehicle was also designed, in order to enable testing of multi-vehicle trains. Table 3.4 (below) shows the values used to construct the carriage. It was assumed, since carriages are usually unpowered, that the bogie and chassis would have lower mass than the equivalent components on a locomotive, but that wheelsets and the vehicle body would weigh approximately the same.

Component	Mass (kg)
Wheelset (x4)	500
Bogie (x2)	5,000
Carriage Chassis	7,000
Carriage Body	10,000
Total Mass	29,000

Table 3.4 - Estimated Mass of Locomotion Vehicle Components

3.2.4 *The Testing Environment*

The environment in which the vehicle was to be tested was designed as follows.

The Scene

The scene in which the track and vehicles exist consists of a few simple elements. There is an infinite ground plane in the x/y axis located at the origin of the scene. The z axis points upwards and gravity is defined as the vector [0, 0, -9.806]. Rails are placed on the ground plane and collisions with this plane are part of the simulation's derailment detection system (described in more detail later in this section). These basic components of the scene are shown in Figure 3.12 (overleaf).

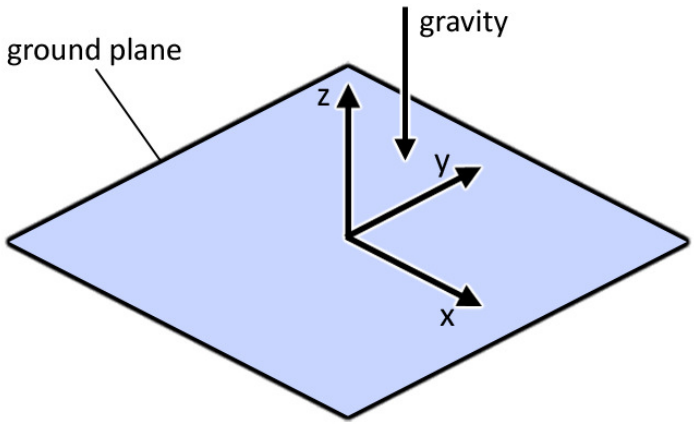


Figure 3.12 - The PhysX Scene

According to the *PhysX* Documentation (which was included with the SDK) [41]: *"The SDK uses unitless numbers to measure three types of basic quantities: mass, length, and time. You can define these quantities to be in any units you want ... The units of derived quantities follow from these basic units."* This simulation uses kilograms (kg) for mass, meters (m) for length and seconds (s) for time. This means that, for example, velocity is measured in meters per second (and converted to miles per hour (mph) where necessary) and forces are measured in Newtons.

Rail Properties

Rails in the simulation are generated based on the profile below. In 3DS Max, this profile is extruded along a spline to create the track geometry. Figure 3.13, below, shows the dimensions of the rail profile.

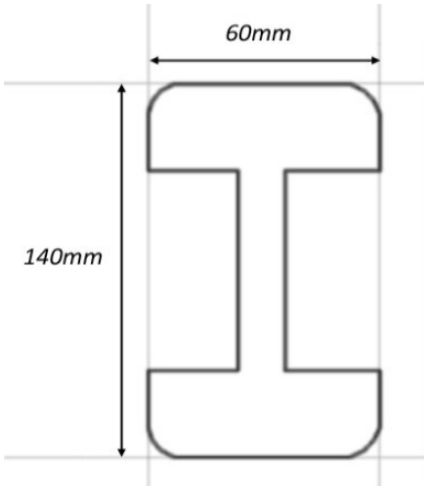


Figure 3.13 - Locomotion Track Profile Dimensions (left)

The track profile has a height of 140mm and a width of 60mm (which, in the absence of more information on the Metro de Madrid track, is based on specifications for a rail profile that is commonly used in the UK) [51].

- *Reversible Rails*

Rather than having a head and a foot, the profile has been designed to be symmetrical (i.e. to have two heads - as shown in the figure above (Figure 3.13)). This allows one model file to be produced for left-curving track and then inverted to create right-curving track. This is a shortcut to optimise the process of constructing the models and loading them into the simulation, by avoiding having to create and load separate models for both left and right bends at each curve radius.

Track Sections

Track is constructed from a set of pre-made 'track sections'. A track section includes the left and right rail, as well as a central spline. Track sections can be assembled in any combination to form a 'track layout', as illustrated in Figure 3.14 (below), a simple straight layout comprised of four straight track sections.

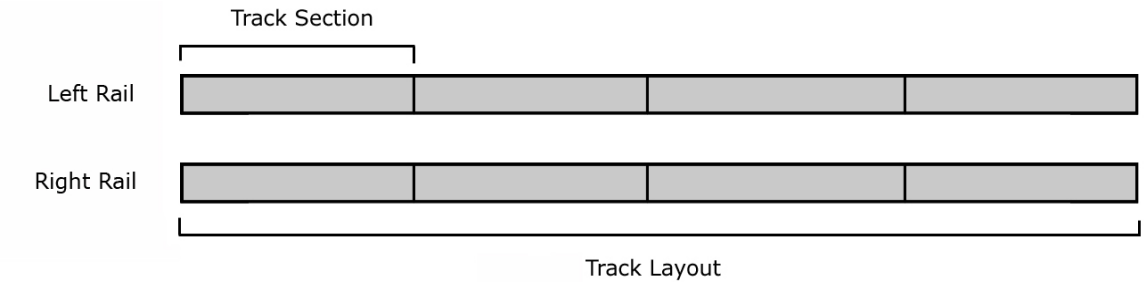


Figure 3.14 - Illustrating Track Sections and Track Layouts

- *Track Gauge*

The gauge of the track used in the simulation is that of the Madrid metro system: which, according to numerous sources including *UrbanRail.net* [52], is 1.445m.

- *Straight Sections*

Straight track sections are 10m in length. A straight rail is constructed by extruding the rail profile polygon (Figure 3.13) along a straight spline in a single segment. This results in a model (including both rails) comprising 48 polygons.

- *Curved Sections*

Curved track is designed so that each section is 1/100th of a circle (i.e. 100 curved sections join together to form a circle, 50 form a semi-circle etc.). This makes it easier to create continuous layouts, such as the loop layout, which is described on Page 64.

Curved track is constructed using splines, as illustrated in Figure 3.15 (below).

Figure 3.15 - Diagram illustrating curved track in the Locomotion Simulation Tool

To construct a track section, the inner rail uses a radius of r minus the half gauge and the outer rail r plus the half gauge.

As mentioned in Chapter 2, in the real world, the radius of a curved track commonly ranges from 500m to 1,000m. For the purposes of the simulation, curved track sections with radii of 500m, 600m, 700m, 800m, 900m and 1,000m have been constructed.

Additionally, in order to enable the derailment testing in Chapter 5, curve sections with the following radii have also been constructed: 100m, 125m, 150m, 175m, 200m, 225m, 250m, 275m, 300m, 325m and 350m. These radii are less common in the real world, but can occur at busy intersections or where there is limited space, such as on inner-city metro routes, so there is some value in simulating them. If, for example, the simulation can be shown to be accurate on these narrower radii, it may be assumed that it will be similarly accurate on the wider radii too.

- *Splines*

Each track section has a spline running along its centre. This is defined as a linear Bézier for straights and a quadratic Bézier for curves, and is based on the straight line/curve shape that was used to generate the track in 3D Studio Max. The spline is used to measure the lateral offset of the wheelset and is used in the calculation of additional centring forces (described in more detail in Section 3.3.14).

Track Layouts

Two track layouts were designed to enable the testing conducted in Chapter 5. Each layout is designed to test different elements of vehicle behaviour.

- *Straight*

The straight is a straight track of variable length, designed to test the acceleration and straight line speed and stability of the vehicle. The length is varied by adjusting a parameter that controls the number of straight track sections created when the scene is initialised. (i.e. if num_straights is 100, then the track is 1km in length).

➢ *Loop*

The loop is designed to test a vehicle's stability and behaviour on curves and consists of two straights and two bends which form an oval shape (as shown in Figure 3.16, below).

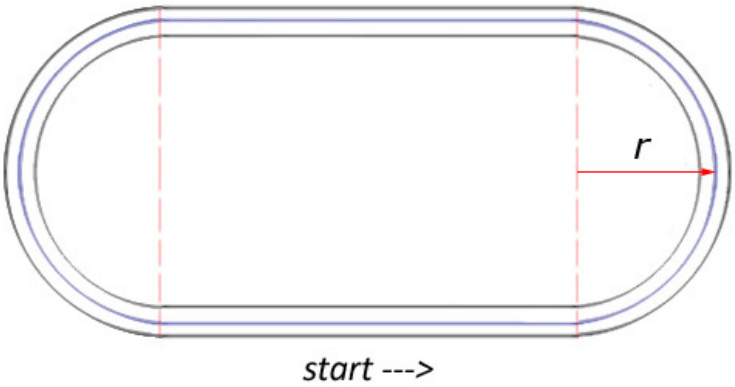


Figure 3.16 - The Loop track layout in Locomotion

The loop can be constructed with curves of different radii. Because of the way the track geometry is created, there is always the same number of track sections (and polygons) in the scene, regardless of the radius that has been selected, and so the performance of the simulation tool should not be affected by changes in the track radius.

➢ *Start & End Sections*

The vehicle starts at a predefined location, on an initial straight section of track that varies in length depending on the type of vehicle and number of carriages. In the case of the looped track, the straight on the opposite side is adjusted to mirror the length of the starting section. Additionally, an extra track section is added to the end of the straight. These start and end sections of track enable the simulation to detect when the vehicle has reached the end of the straight or has completed a loop of the track.

3.2.5 *Summary*

In this section, the design for the vehicle and the testing environment has been presented. Two key design choices have been identified, which will require testing:

- Three wheelset designs with different polygon counts, designed to evaluate the stability and performance of the simulation tool.
- Two wheelset variations (Single Body and Multi Body), designed to test whether it is desirable to have a more flexible wheelset model, but which might be less stable.

It is not clear what the effect of these different design options will be on the simulation, and so they will be evaluated in the testing presented in Chapter 5.

The next section presents the design for the simulation tool's software architecture.

3.3 Software Design

Key elements of the design of the software architecture of the Locomotion tool are presented in this section, including the integration of PhysX and other existing software, the design of the application flow and new features such as the spline-based centring technique described in Section 3.3.14.

3.3.1 *Software and Libraries*

The following software, tools and libraries were used in the development of the Locomotion simulation tool.

3D Studio Max

Autodesk's 3D Studio Max ('3DS Max') 2012 [44] is 3D design software used for modelling, animation and rendering. It was chosen because it is commonly used in the video games industry and has a PhysX plugin available, meaning that it is capable of exporting the 3D models and rigid bodies in a format that can be imported into the simulation tool and used by the PhysX engine (described in more detail below). It is also freely available for academic use. 3DS Max is used to construct the 3D models and physics meshes used in the simulation. 3D Models are exported as .OBJ files, a commonly used format, and a PhysX plug-in enables PhysX rigid bodies to be exported as .XML files.

C++, the Standard Library and the Standard Template Library

The simulation was developed using C++, an object-oriented programming language commonly used by software and game developers. C++ includes access to a number of software libraries, including the Standard Library [53] - a collection of useful classes and functions, including data types, mathematical functions, data/time functions and methods for handling file input/output (some functions of which are used to import the simulation configuration files) - and Standard Template Library (STL) [54] - an expanded C++ software library that provides other useful implementations, including mathematical algorithms and data structures.

Microsoft Visual Studio (2010)

The software was developed in Microsoft's Visual Studio [55], which is an Integrated Developer Environment (IDE) used to write, compile and debug code, and is also commonly used in the games industry. The use of an IDE simplifies the code development and debugging process.

PhysX

The *PhysX SDK* (version 2.8.4) is used to model the physical behaviours of the vehicles and objects in the simulation. The PhysX engine is initialised and updated as part of the simulation's application flow (described in more detail in the next section).

- *NXUStream*

The PhysX SDK includes the *NXUStream* utility, which is a C library that imports the physics XML files (exported from the PhysX plugin in 3DS Max) and manages the creation and initialisation of PhysX objects in the scene from those files. [4]

OpenGL

OpenGL [56] is an open-source platform for rendering 2D and 3D scenes. It has been used in the development of a number of games, from mobile games like Angry Birds to triple-A titles like Portal and the Need for Speed series [56]. OpenGL is used to render 3D models and other data visualisations in the simulation (more on visualisations in Section 4.1.5).

- *GLModel (GLM)*

GLM [57] is a C++ library designed to import and process .OBJ model files so that they can be rendered in OpenGL. It is used to import the 3D models that are generated by 3DS Max and convert them into a format that can be rendered in the simulation. The process creates a 'GLModel' object with a Render function, which can then be rendered along with the rest of the scene (as described later).

Simple DirectMedia Layer (SDL)

SDL [58] is a cross-platform library written in the C programming language that manages the creation of the application window, keyboard and mouse input, audio functions and access to graphics hardware, in this case via OpenGL. SDL is used to simplify the process of initialising the Locomotion application, as well as for handling user input. Version 2.0 of SDL is used in the Simulation.

Summary

These libraries and software packages are used to simplify the process of creating the application, allowing this research to focus on the evaluation of the physics engine.

3.3.2 *Key Classes and Methods of the PhysX SDK*

Initial research into the PhysX SDK Documentation [41] identified the following key classes and methods that are necessary to use the PhysX SDK, including classes that represent rigid bodies. These need to be integrated into the simulation tool.

- *SDK object* - it is necessary to create and initialise an 'SDK object'. This is a factory class used for instancing objects in the PhysX SDK. [41]
- *Scene object* - Once the SDK is initialised, it is then necessary to create a PhysX scene. A scene is a collection of bodies that can interact with each other [41]. A call to the scene's simulate method advances the simulation. The simulate method takes a parameter called 'elapsed time', which advances the simulation by the specified timestep (i.e. 1/60th of a second).

- *NxActor* - Once the scene has been initialised, it is possible to create simple 'Actors' - the PhysX SDK's name for rigid bodies - such as spheres and cubes by passing actor descriptions to the scene object. Alternatively, actors can be loaded into the scene using the NXUStream tools described earlier.

3.3.3 *Application Flow*

In order to integrate PhysX, the simulation has been constructed in a similar way to most video games and has five main phases: the Initialisation, Loading, Setup, Update and Render phases, as illustrated in Figure 3.17 (below).

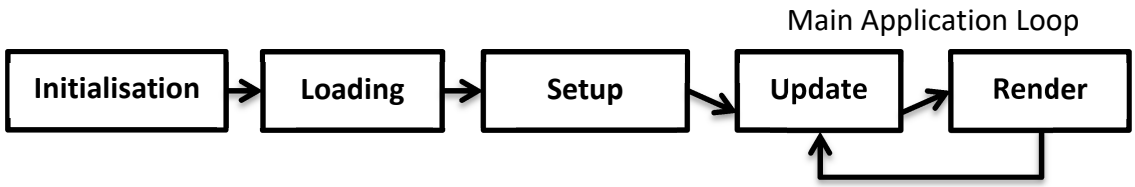


Figure 3.17 - The Five Main Phases of the Locomotion Application Flow

Initialisation Phase

During this phase, the application is initialised, the window is created and the PhysX SDK is initialised. This enables the loading phase to begin.

Loading Phase

The models, physics files and other prerequisite data files - including simulation, vehicle and test parameters defined in configuration files - are loaded into the application.

Setup Phase

The loaded objects are initialised, based on user-defined settings (loaded from the simulation setup files), ready for testing to begin.

Main Application Loop

The main application loop consists of the update and render phases, described below. This loop continues until the user exits the application or until all tests are complete.

- *Update Phase*

The *PhysX* 'simulate' method is called, which advances the physics simulation, and the entities in the scene are updated, forces are measured/applied etc. Most custom behaviours and measurements are processed in this phase in order to minimise the calculations performed during the render phase and maximise simulation performance.

- *Render Phase*

All of the objects are rendered. Other information such as track splines and contact data is also rendered in this phase.

3.3.4 *Wrapper Classes*

While it would have been simpler to develop Locomotion by integrating PhysX directly into the application, the simulation has instead been designed so that the physics engine can be upgraded or replaced with a minimum amount of effort. It was originally intended that this would allow for the PhysX engine to be upgraded if necessary, and to potentially allow for comparisons to be made between different Physics Engines/Engine versions (in a similar way to the PAL interface from Boeing and Braunl (2007) [42]). Unfortunately, time constraints prevented testing with additional engines. This section describes how this was achieved.

Key Physics Wrapper Classes

The physics engine and rigid bodies needed to be encapsulated, which involves creating 'wrapper classes' around key components of the PhysX engine. A wrapper class is an interface that allows interaction with multiple different implementations of an object. Below are examples of two key classes where this approach has been used; Wrapper Classes designed to allow the Locomotion simulation to interact with the PhysX engine in a way that would have allowed the objects to be implemented using an alternative engine to PhysX.

- *Physics Actor*

The Physics Actor was designed as a wrapper around the NxActor class, which represents a rigid body in the PhysX engine, as illustrated in Figure 3.18 (below).

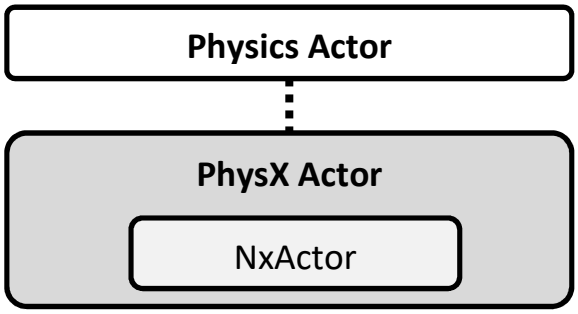


Figure 3.18 - Illustrating the relationship between the base Physics Actor class and the inheriting PhysX Actor class, a wrapper around a PhysX NxActor object.

The generic class Physics Actor (white) is a pure virtual class, with no engine-specific implementation. A PhysX Actor (grey) object is then created, which inherits from Physics Actor and implements those pure virtual functions. This PhysX Actor object is a wrapper around an NxActor, and most of the methods are proxies for the methods of the inner object. For example, when we call the PhysX Actor's "Get Mass" method, this returns the value returned by the NxActor's "Get Mass" method.

Key methods of the Physics Actor object include the following methods, most of which are proxy methods for the NxActor and which are used to access or set various properties and parameters of the object, or to apply forces to the object etc.:

- *GetPosition() / SetPosition(pos)* - used to access and set the position of the actor.
- *GetRotation() / SetRotation(rot)* - used to access and set the rotation of the actor.
- *GetMass() / SetMass(mass)* - used to access and set the mass of the actor.
- *SetMaterialProperties(a, b, c)* - used to set the coefficient of restitution (a), static friction (b) and dynamic friction (c) properties of the actor.
- *ApplyForce(force)* - used to apply a linear force to the actor.
- *ApplyTorque(torque)* - used to apply a rotational force to the actor.

This approach allows for alternative objects to be created that inherits from the generic Physics Actor class, but has a different implementation (i.e. a 'Havok Actor' class). The implementation would be different for the different physics engine, but would still work within the context of the Locomotion simulation because the objects share a common interface, through the base Physics Actor class. The locomotion application classes can then interact with PhysicsActor regardless of whether it is implemented as a PhysXActor or another type of physics actor (such as a HavokActor, for example).

➢ *Physics Engine*

Similarly, the Physics Engine class is a wrapper around key components of the PhysX SDK. It encapsulates all of the key classes and methods of PhysX (as identified in Section 3.3.2). Figure 3.19 (below) illustrates the components of the PhysX2 Engine, and its relationship to the base Physics Engine class.

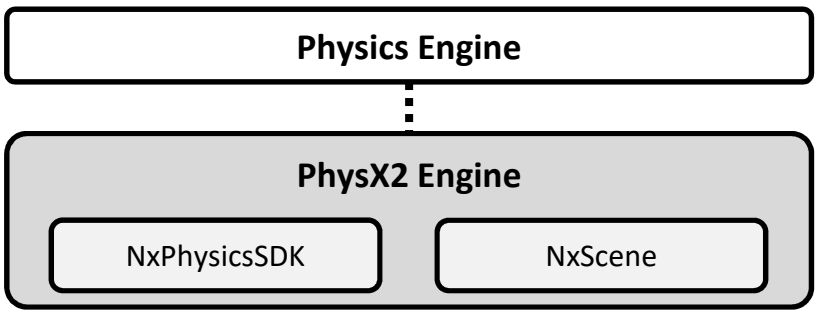


Figure 3.19 - Illustrating the relationship between the base Physics Engine class and the inheriting PhysX2 Engine class, a wrapper around a PhysX SDK and Scene objects.

As with the Physics Actor, the Physics Engine class is a pure virtual base class, which serves as a common interface with the rest of the simulation. The 'PhysX2 Engine' class then implements the virtual methods of the base class using the necessary PhysX objects and methods, including the NxPhysicsSDK and NxScene objects.

The physics engine class includes the following key methods:

- *Init()* - used to initialise the SDK and create the PhysX Scene
- *LoadFile(string filename)* - used to load the specified file into the engine and to initialise it. Stores the created actor so that it can be cloned and placed in the scene.
- *CreatePhysicsActor(string actorName)* - which clones one of the loaded Physics Actor objects and returns a pointer to it.
- *Simulate(float timestep)* - used to call the PhysX Scene's simulate method

Also, as with Physics Actor, this potentially allows for alternative engines (i.e. 'PhysX3 Engine' for a more recent version of the Engine, or 'Havok Engine' for a Havok-based alternative) to be created through a common interface that will work with the Locomotion tool.

3.3.5 *'Entity' Class*

The Entity class is the base class for all entities in the scene, including the track sections, wheelset, bogie and other vehicle components. Each entity comprises two key components; a 'Physics Actor' and a 'GLModel' object, as shown in Figure 3.20 (below).

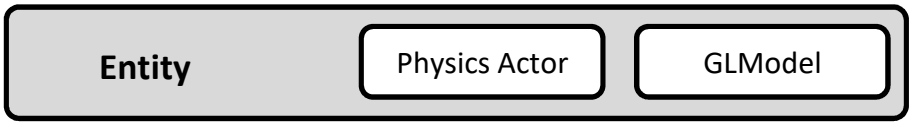


Figure 3.20 - Illustrating the key components of the Entity class

Physics Actor

Physics Actor encapsulates the Entity's physics, as described on the previous page. Properties of the entity, like mass and material properties, are accessed from the Physics Actor. So, for example, the entity has a *GetMass* method, which returns the mass of the object via the Physics Actor's *GetMass* method.

GLModel

The GLModel represents the 3D graphical object, which is loaded from the .OBJ file exported from 3DSMax and imported by GLM. It has a Render function, which is called when the entity's Render function (described below) is called.

Key Methods of the Entity Class

As well as accessor methods for retrieving and setting parameters of the object (such as mass and velocity), applying forces to the object and calculating its speed etc. - most of which are aliases for methods of the Physics Actor object - there are two key methods required to update and render the object.

- *Update method*

The main function of the update method is to retrieve the current position and orientation of the rigid body (from the Physics Actor object) and store it. It also includes any custom behaviour for that object, such as the propulsion method of the bogie entity.

- *Render method*

During the Render method, the stored position and orientation data are used to render the entity's GLModel object in the correct position in the scene.

Example: Wheelset Classes

An example of how entities in the scene are created by inheritance from the Entity base class is the two wheelset designs; the Single Body and Multi Body Wheelset. The SB wheelset is derived from entity, with the MB wheelset inheriting from the SB Wheelset. Figure 3.21 (below) shows how the wheelsets inherit from each other, and how they are represented by their physics actor objects.

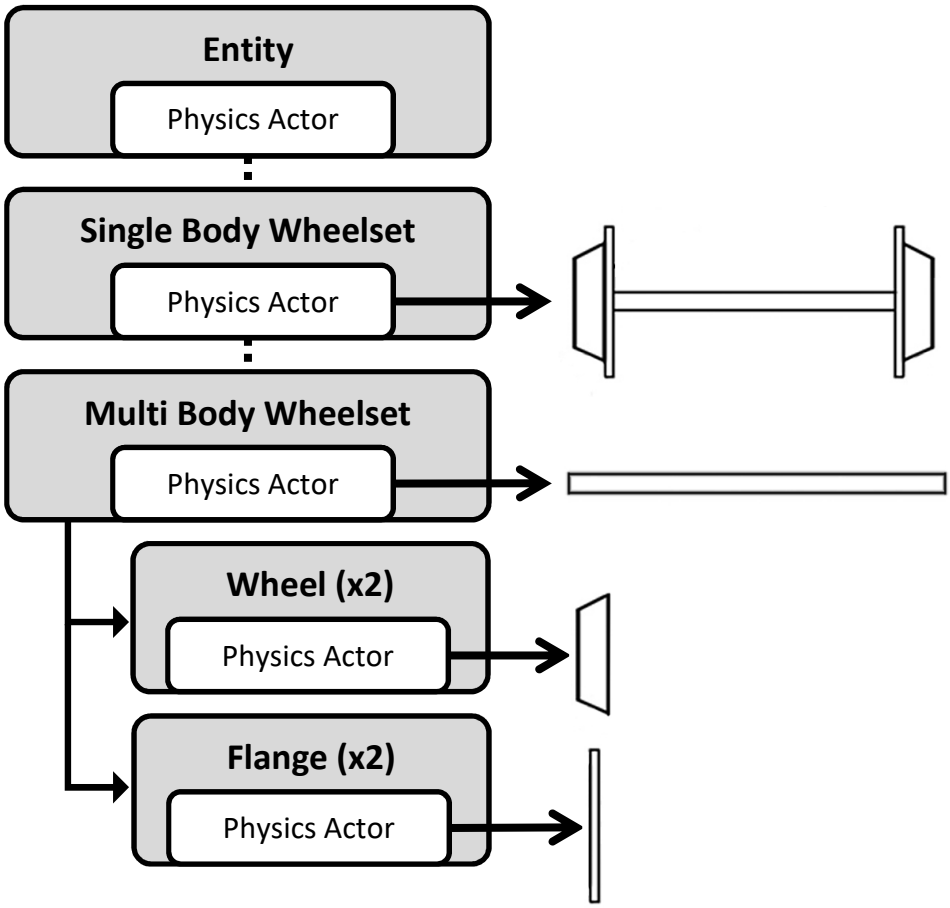


Figure 3.21 - Illustrating the Wheelset/Entity Class Hierarchy

The SB Wheelset is a single Entity, whose Physics Actor is a single rigid body comprising the axle, wheels and flanges. The MB wheelset inherits from the SB Wheelset, with its main Physics Actor being the axle; it then has additional member Entities, each with its own Physics Actor, representing the wheels and flanges, attached to the main actor via fixed joints. Similarly, the Conical Wheelset object inherits from SB Wheelset, but its Physics Actor is replaced with the conical mesh.

Other Entities and their Properties

There are a range of entities in the simulation, including the wheelsets, bogies, track sections and vehicle bodies. As described earlier, many of the physical properties of these objects are accessed via the Rigid Body (Physics Actor), but the simulation also needs to store additional properties which are not. Examples of these are discussed below.

- *Wheelset Properties*

Properties of the wheelset such as conicity and the contact angle will be added as member variables of the wheelset object. These properties will be initialised when the object is created and can then be accessed as needed, as they do not change unless the user changes them manually.

- *Rail Properties*

Parameters such as the gauge and half-gauge of the track will need to be stored in the track section objects, so that they can be used by the code that calculates the centring forces. The half-gauge is stored in order to avoid having to calculate this value whenever it is needed, which is intended to improve the performance of the simulation tool.

3.3.6 *'Simulation' Class*

The Simulation class is the main object of the Locomotion tool. The class has methods for dealing with keyboard and mouse input into the simulation (via SDL methods). Figure 3.22 (below) shows the key components of the Simulation class.

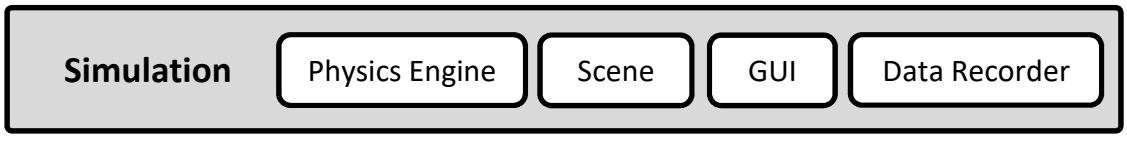


Figure 3.22 - Illustrating the key components of the Simulation class

The key components of the simulation are the Physics Engine, which (as described above) is implemented using PhysX and is used to simulate the physics of the rail vehicle, and the Scene object, which contains all of the 3D models which need to be rendered. Other key components are the Graphical User Interface (GUI), which reports data about the simulation to the user, and the Data Recorder, which tracks data and outputs it at the end of each test. Both of these components are described later in this section.

Key methods of the Simulation Class

The two key methods of the Simulation class are the Update and Render methods, which are described below

- *Update Method*

The first key method of the Simulation class is the Update method. The main steps of the update method are illustrated in Figure 3.23, below.

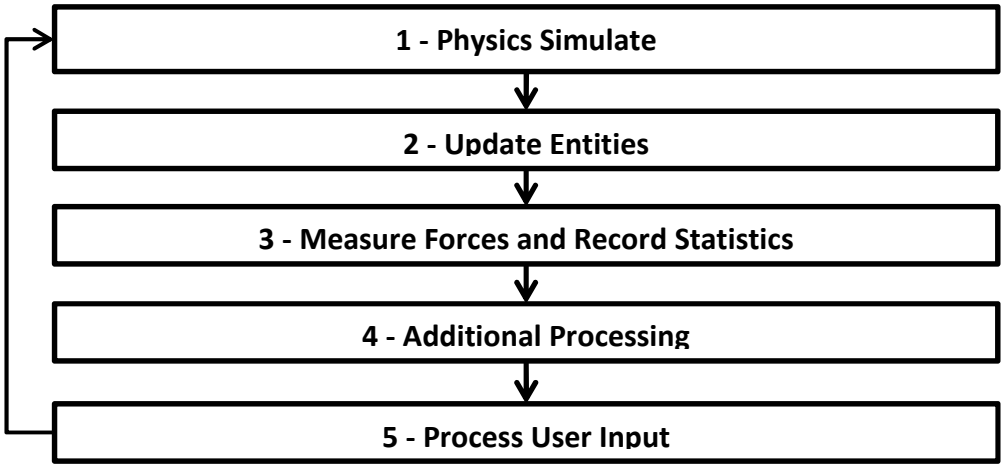


Figure 3.23 - Intended flow of the Locomotion 'Update' Loop

1. *Physics Simulate* - the PhysX Simulate method is called; this triggers the PhysX engine to update the position and orientations of the rigid bodies, processes collisions etc.
2. *Update Entities* - the system updates all of the entities in the scene, executing any custom behaviour and updating the 3D models ready for rendering.
3. *Measure Forces and Record Statistics* - the forces and properties of all the objects in the scene are measured and stored, ready to be output to data files later [1].
4. *Additional Processing* - any additional processing or custom behaviours are performed, such as corrective forces being applied to the objects [2].
5. *Process User Input* - any input from the user - such as camera controls or altering the target speed of the vehicle - are processed, ready for the next frame.

[1] - writing files to the hard drive is a performance cost and so is not done while the simulation is executing.
[2] - this includes the wheelset centring force technique described in Section 3.3.14.

- *Render Method*

The Render method is then called, which iterates through all of the Entities in the scene and calls their render method. A minimum amount of calculations should be performed during the render phase, in order to maximise the performance of the simulation tool.

Graphical User Interface (GUI)

The GUI was designed to provide the user with information about the simulation on screen while it is running. It also allows the user to interact with the simulation. Figure 3.24 (below) shows the initial design for the interface. The figure shows a side view of a wheelset (black) rolling along the track (grey), along with the key areas of the interface where data and controls will be added (white).

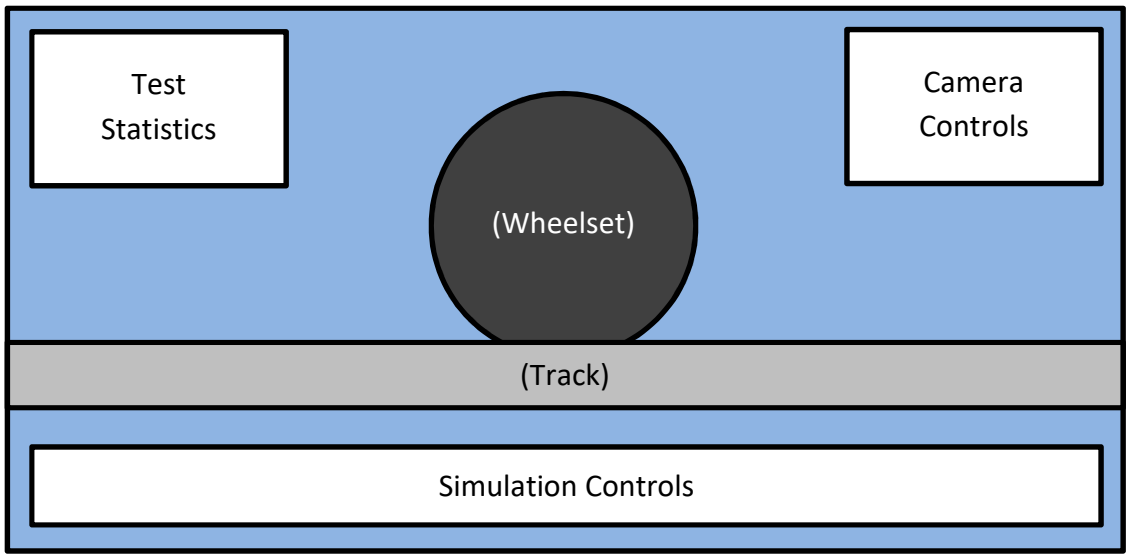


Figure 3.24 - Initial design for Locomotion Graphical User Interface (GUI)

- *Test Statistics (Top Left)*

This part of the interface will show data about the test, such as the position of the vehicle, the elapsed time, the number of tests remaining (if in batch mode) and other statistics such as the number of flange collisions.

- *Camera Controls (Top Right)*

These controls will allow the user to choose from a range of cameras, in order to view the scene from different angles. For example, it will allow the user to select a target wheelset and the direction from which the wheelset is viewed.

- *Simulation Controls (Bottom, Centre)*

These controls will allow the user to adjust key features of the simulation, such as playing and pausing the simulation, and adjusting the target speed of the vehicle.

Data Recorder

Another key component of the simulation is the Data Recorder, which stores key information about each test, such as the parameters used, as well as key data about the vehicle, the testing environment, and the test results, including:

- The Number of Flange Collisions
- The Lateral Offset of Each Wheelset
- The Stability of the Wheelset
- The Speed of the Vehicle
- The Position of the Vehicle
- The average framerate during the test
- The length of each test
- The peak speed of the vehicle during each test
- Whether derailments occurred during the test
- The derailment speed of the vehicle, if derailment occurred

In the case of data such as the lateral offset of the vehicle, an accumulator is used to calculate the average offset. i.e. the accumulator tracks the total recorded lateral offset and a counter records the number of samples that are recorded. The minimum and maximum values are also stored. This allows the simulation to record the average result and range of results, but it avoids storing prohibitively large amounts of data, which can lead to memory leaks and other errors in the application.

- *Additional Data*

If configured to do so, the data recorder is also able to store the following real-time data:

- The lateral offset of the wheelset
- The normal force acting on the wheelset (single body) or each wheel (multi body)

These features require large amounts of data to be collected over the course of the test, and so are not enabled by enabled by default.

3.3.7 *PhysX Callbacks*

The 'PhysX2 Engine' object is configured to receive the contact callbacks from PhysX when collisions are detected between objects. These contacts are processed and then passed to the Simulation object for processing. This data flow is illustrated in Figure 3.25 (below).

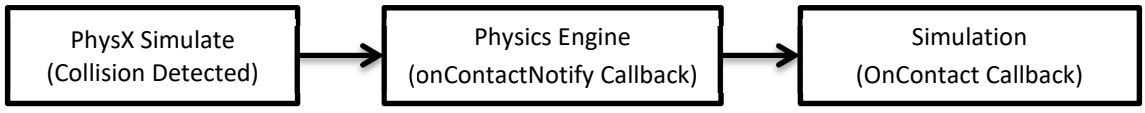


Figure 3.25 - Illustrating the Callback data flow following PhysX Collision Detection

In order to do set up the callback from PhysX, it is necessary to extend a PhysX class called NxUserContact report, as shown in this sample from the PhysX Documentation [41]:

```
class MyContactReport : public NxUserContactReport
{
    void onContactNotify(NxContactPair& pair, NxU32 events)
    {
    //You can read the contact information out of the
    //contact pair data here.
    }
} myReport;
```

This contact report object can then be registered with the PhysX scene using

```
physXScene->setUserContactReport(&myReport);
```

When PhysX detects contacts between rigid bodies, the onContactNotify method is called, which has two parameters. The first is an NxContactPair object, which contains pointers to the two rigid bodies, and an 'events' flag, which contains information about the type of contact event. These events include a range of events, such as:

- NX_NOTIFY_ON_START_TOUCH - which indicates that two objects have come into contact with one another.
- NX_NOTIFY_ON_END_TOUCH - which indicates that two contacting objects are no longer in contact.
- NX_NOTIFY_ON_TOUCH - which occurs during the duration of the contact between two objects.

The callbacks occur during the simulate method of the physics engine and it is recommended that the objects are not altered at this stage [41]. Therefore, any objects that need to be processed are stored and dealt with later, during the rest of the application update loop (described in more detail later, in Section 3.3.6).

Contact Groups

The simulation includes two methods for identifying rigid bodies, the first of which is contact groups. Each rigid body is assigned a contact group, which is represented as an enumerated type. The simulation has collision groups for the ground plane, rails and wheelset (as well as for wheels and flanges in the case of the Multi-body Wheelset). There are also 'super' groups, such as the 'Entity' group, which contains all vehicle components, wheelsets and all sub groups. Collision groups allow for different behaviours to occur following contact between specific component types, such as recording collisions between wheels and the rails.

These collision groups are represented thus:

```
enum Collision Group
{
	CG_IGNORE		= (1<<0),	// binary 0001
	CG_GROUNDPLANE		= (1<<1),	// binary 0010
	CG_ENTITIES		= (1<<2),	// binary 0100
	CG_RAILS		= (1<<3),	// binary 1000
	CG_FLANGES		= (1<<4),
	CG_WHEELS		= (1<<5),
	CG_WHEELSET		= (1<<6)
}
```

The notation "1<<0" denotes binary encoding. This use of 'bitflags' allows bitwise operations to be conducted on the collision group of colliding Entities. It is possible for an entity to be in multiple collision groups, and the simulation should process collisions accordingly. This, for example, allows the collision group for the flanges of the Multi-body Wheelset to be defined as:

```
actor.setGroup( CG_FLANGES | CG_WHEELSET | CG_ENTITIES )
```

This states that the actor is a member of the collision groups for Flanges, Wheelsets and Entities. Then, for example, it is possible to use the code below...

```
if (actor1.group & CG_ENTITIES && actor2.group & CG_GROUNDPLANE)
```

... to check if a member of the Entity collision group has come into contact with the ground plane, which would indicate a derailment, and executing the necessary code to process this. It is then also possible to check for more specific collision scenarios, such as the following example:

```
if (actor1.group & CG_FLANGES && actor2.group & CG_RAILS)
```

This code checks specifically for collisions between flanges and rails, and would process the entities accordingly (i.e. register a flange collision).

Rigid Body Names

The second way of identifying rigid bodies is by their names. Rigid Bodies in the simulation are identified according to which vehicle they belong to, in a hierarchical fashion. This allows data to be collected about which component has become derailed, for example, or to track the lateral offset for a specific wheelset. Each vehicle component has a unique name. For example, the front wheelset of a multiple vehicle train would be named *"FrontLoco_BogieF_Wheelset",* whereas the rear wheelset of the train would be named *"RearLoco_BogieR_WheelsetR".*

3.3.8 *Contact Object*

The contact object represents the contact between a wheelset and the track. When a contact callback from PhysX is registered, the contact data is stored and can then be processed later during the application update loop.

The Contact object consists of:

- Wheelset - a pointer to the wheelset involved in the contact.
- Track 1 - the first track section which the wheelset comes into contact with.
- Track 2 - a second track section pointer, in case two contacts exist.

Two track pointers are necessary for scenarios in which the wheelset moves from one track section to another and may be in contact with both, as illustrated in Figure 3.26, below.

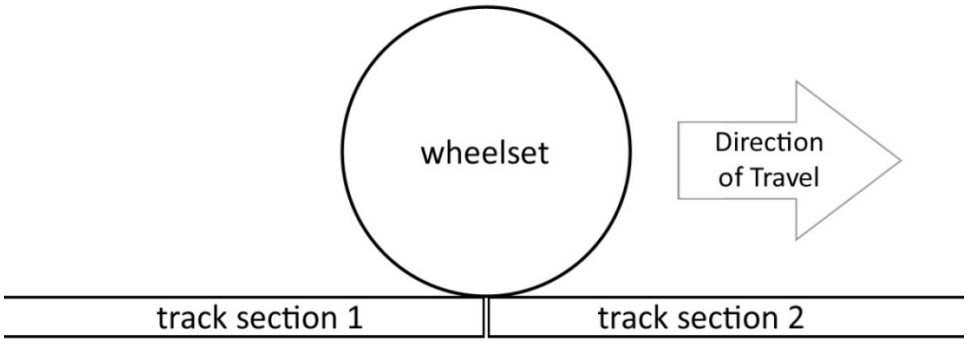


Figure 3.26 - Illustrating the elements of the contact object

One Contact object is created and stored by the simulation for each wheelset in the scene during the callback from PhysX, and is processed in the next simulation update loop.

3.3.9 *Testing Modes*

The simulation will have a range of different testing modes, designed to evaluate different aspects of vehicle behaviour. Each test mode has different testing and data analysis features available, depending on the test vehicle. These are as follows:

- *Wheelset Testing Mode* - to evaluate the behaviour of a single wheelset.
- *Bogie Testing Mode* - For testing with a single bogie
- *Vehicle Testing Mode* - For testing with entire vehicles/multi-vehicle trains.

Wheelset Testing Mode, for example, will have additional features for measuring and visualising the path of the wheelset, whereas Vehicle Testing Mode may have features for describing the stability of each vehicle in the train that are not available in the other testing modes.

Wheelset testing, for example, also provides the user with three further options:

- *Flange Collision Tests* - use the MB Wheelset, track and render flange collisions.
- *Offset tests* - using either wheelset type, track and render the full lateral offset path.
- *Normal force tests* - track and print normal forces acting on the wheelset in real-time.

3.3.10 *Data Accuracy and Output Format*

Data produced by the simulation is accurate to three decimal places, so distance, for example, is measured to a fidelity of 0.001m (1mm). Data from the simulation is output in the commonly used Comma-Separated Values (CSV) data format, also known as comma-delimited data. For example, if the code was tracking the position of the vehicle over time - and if the vehicle was travelling along the y axis and the simulation was recording the position of the vehicle 10 times per second - the output file might look like this.

```
timestamp,x,y,z
0.0,0.000,0.000,0.000
0.1,0.000,2.001,0.000
0.2,0.000,3.024,0.000
0.3,0.000,3.019,0.000
            ...
```

The first row of data represents the column headings. Each subsequent row of text is a row in the data table and commas separate the data columns. By saving the files in the format '<Filename>.csv', the data can quickly be loaded into a range of applications, including Microsoft Excel, for analysis.

3.3.11 *Menu Flow Design Overview*

The user will be able to specify the parameters of the simulation, including vehicle and environment settings, batch testing settings and so on, by editing a number of different configuration files. However, in order to make the simulation easier to use, the user will also be able to change these parameters via a series of interactive graphical menus in the simulation. The full, graphical design for the interface has not been included, but below is the description for how the interface will work.

An initial menu will ask if the user wants to run tests using settings loaded from the setup files, or whether they want to customise the test settings. If they choose to load the settings from the setup files, the testing will begin immediately.

If the user chooses to customise the simulation settings, then the interface should allow the user to make the following selections via a series of menus:

1. Select Testing Mode (Wheelset/Bogie/Vehicle, as described above)
2. (If Vehicle) Select vehicle properties (i.e. number of carriages)
3. Wheelset Settings (Cone Wheelset/SB Wheelset/MB Wheelset)
4. Track Layout (Straight/Loop)
5. Track Properties (i.e. Number of Straights if Straight, Curve Radius if Loop)

Screenshots of this menu flow in the Locomotion Tool are included in Chapter 4.

3.3.12 *Inputs*

As described earlier, the SDL library handles the creation of the application window, as well as handling the user inputs into the application. The simulation was designed to allow the following user input:

- Mouse - mouse input is used for camera control, as well as interacting with the graphical user interface (GUI) to adjust the simulation
- Keyboard - a series of keyboard shortcuts are used for controlling various functions of the simulation, including:
    - Space Bar - used to Play/Pause the Simulation.
    - + / - keys - used to increase/decrease the target speed of the vehicle.
    - Arrow Keys - used for camera control.

3.3.13 *Calculating Lateral Offset*

It is necessary for the simulation to be able to measure the lateral offset of a wheelset. As described earlier, splines have been added to each Track Section to allow the lateral offset to be calculated. The position of the spline of a curved track section relative to the wheelset and rails is illustrated in Figure 3.27 (below).

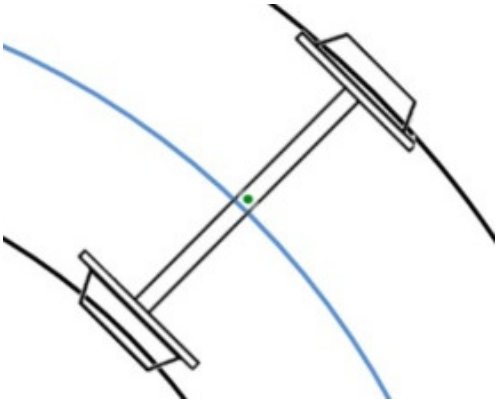


Figure 3.27 - The spline (blue line) between two rails of a curved track section

The figure shows a wheelset that has experienced lateral offset. In this case, the position of the wheel (green dot) has moved away from the central spline (blue line). The lateral offset of the wheelset is measured as the distance from the wheelset's position to the nearest point on the spline (ignoring any vertical difference (in the z axis)).

Bezier Curves

The splines are defined as Linear Beziers for straight track sections and Quadratic Beziers for curves. Bezier curves are parametric curves that describe a straight line (or curve) using a series of control points. A quadratic Bezier curve is illustrated in Figure 3.28 (below).

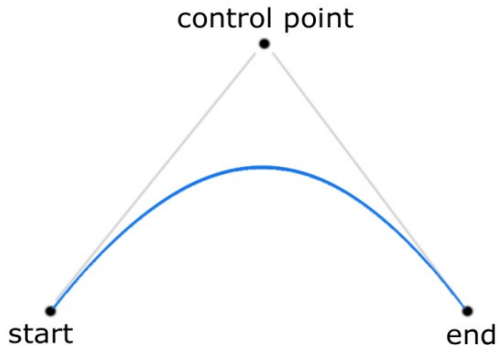


Figure 3.28 - How a Bezier curve (blue) is defined by its control points

It is possible, using the formula below, to interpolate along a Bezier using a value ('t') of between 0.0 (which returns the start point) and 1.0 (which returns the end point). The pseudocode for this calculation is shown below.

```
Point QuadraticBezier::GetPointAt(float t)
{
    float u = 1.0f - t;

    float x = (start.x*u*u) + (control.x*t*t*u) + (end.x*t*t);
    float y = (start.y*u*u) + (control.y*t*t*u) + (end.y*t*t);
    float z = (start.z*u*u) + (control.z*t*t*u) + (end.z*t*t);

    Point result = Point(x,y,z);
    return result;
}
```

Figure 3.29 - Pseudocode for calculating a point on a Bezier curve

Given a value of 't' (between 0 and 1) this pseudocode returns a point on the Bezier spline based relative to the three control points; 'result', 'start' 'control' and 'end' are 'Point' objects that represent positions in 3D space.

PhysX Callbacks

First, collisions between wheels and rails are recorded via callbacks from PhysX (as described earlier) and the wheel/rail pairs are stored in a Contact object for processing later. Then, during the update loop of the simulation, the nearest point on the spline to the position of the wheel is calculated, as described below.

Spline Interpolation

Initially, the nearest point on the track spline was found by interpolating at intervals of 1/10,000 (or approximately once per mm - since the simulation is to be accurate to a fidelity of 1mm, as described in Section 3.3.10) along the Bezier curve until the nearest point was found. The value of 't' and the nearest point are stored and used as a starting point for future interpolation, in order to reduce the calculations required. This process is repeated at every frame during the duration of the contact, and avoids having to interpolate along the entire spline to find the nearest point during every frame. Once the nearest point on the spline has been found, the lateral offset of the wheelset can be calculated. Figure 3.30 (below) illustrates how this is done.

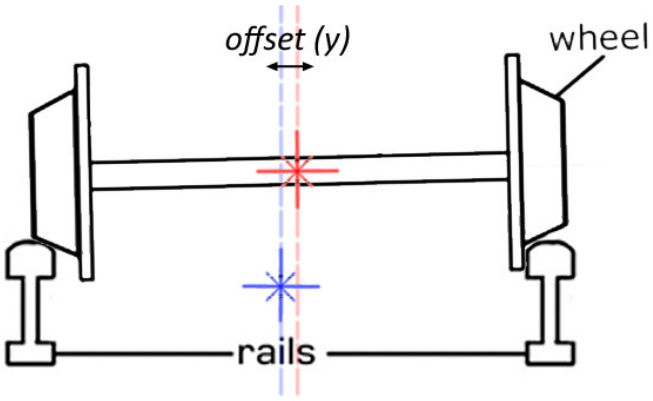


Figure 3.30 - Calculating lateral offset using the nearest point on the spline (blue) and the position of the wheelset (red)

The lateral offset is calculated by using the distance between the nearest point on the spline (blue) and the position of the wheel (red), ignoring any differences in the z-axis. This gives a numerical value that describes the distance (in metres) from the track centre.

Calculating the Direction of Offset

Determining the direction of the offset, i.e. whether the wheel is to the left or right of the central spline, is necessary to detect behaviours such as Hunting Oscillation. It is possible to calculate the offset direction by using the tangent to the spline, calculated as the derivative or direction of the spline at the nearest point. However, the methods below are less computationally expensive and will have a smaller performance impact.

In the simulation, the offset direction is calculated in one of two ways, depending on the track layout, as described below.

- *Straight Track*

As illustrated in Figure 3.31 (below), if the track is straight, then the rails (black) are placed either side of the y axis. This enables the offset to be measured as the x component of the position of the wheelset.

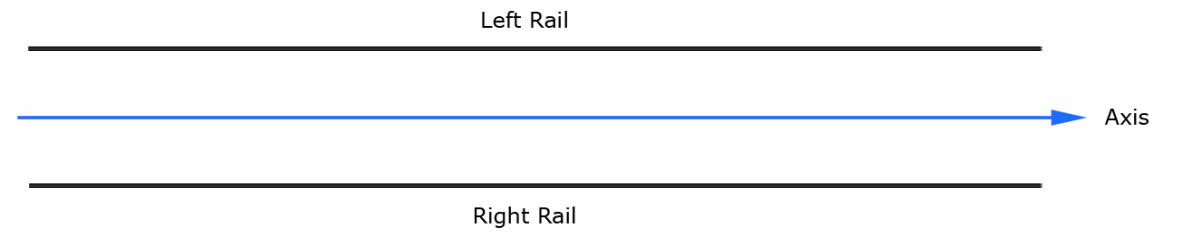


Figure 3.31 - Straight track in relation to the X Axis

- *Looped Track*

The offset direction for the loop layout can also be easily calculated because the loop has been laid out around the origin of the scene (0,0,0), as illustrated in Figure 3.32 (below).

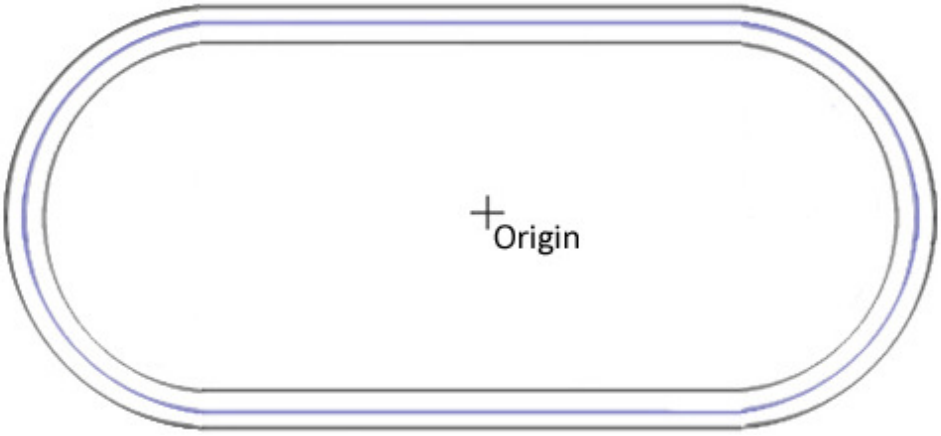


Figure 3.32 - A diagram of the loop layout, showing the origin of the scene

If the wheelset is nearer to the origin than to the nearest point on the spline, it is to the left of the track centre, and if it is further from the origin it is to the right.

The simulation stores the magnitude of the offset vector as a positive value if the wheelset is to the right of the spline and negative if it is to the left.

3.3.14 *Improving the Simulation of the Wheel/Rail Interface*

It was necessary to implement an additional corrective force in order to produce more realistic results from the wheel/rail interface in the simulation. The vehicle was not producing the expected results (as discussed in Section 5.7), and it was assumed that one possible reason for this was that the self-centring effect of the wheelset was not occurring correctly, and so a new technique was developed improve this behaviour. Development of this solution focussed on improving the behaviour of individual wheelsets; to see if augmentations to the system could improve the stability of the simulation and allow the wheelset/bogie to accelerate to higher speeds and derail at speeds closer to the Nadal predictions.

Spline-Based Centring

To replicate the wheelset's self-centring effect, a spline-based approach was used, using the track splines and many of the same calculations that are used to measure the lateral offset of the wheelset. A force is applied to each wheelset to pull it back towards the track centre, based on the distance from the track spline and the properties of the wheelset, as described below.

- *Gravitational Stiffness Force*

The Gravitational Stiffness Force (GSF) equation was used to calculate the size of the centring force. This is Formula 2.3 on Page 19.

- *Lateral Offset*

Section 3.3.14 describes how the lateral displacement of the wheelset is calculated. A Contact object stores the lateral offset distance, along with the nearest point on the track spline and the offset vector. This data can also be used by the simulation to calculate and apply the centring force to the wheelset.

- *Effective Mass of the Wheelset (W)*

A key element of the GSF formula is the Effective Mass of the Wheelset (W). The effective mass of a single wheelset is simply its mass (500kg), but when the wheelset is tested as part of a more complex rail vehicle (a bogie, locomotive etc.), additional calculation is required. When simulating a bogie, for example, the simulation assumes that the mass of the bogie is evenly distributed between the two wheelsets. The effective mass is calculated by dividing the mass of the bogie between the wheelsets, as described below:

$$Effective\ Mass\ =\ Wheelset\ Mass\ (500kg)\ +\ \frac{Bogie\ Mass\ (6{,}000kg)}{2}\ =\ 3{,}500\text{kg}$$

Each bogie/vehicle entity in the simulation defines a function that returns the effective mass of its wheelsets.

This value can be multiplied by gravity to make it equivalent to 'W' in the GSF formula. This value is calculated and stored when the vehicle is constructed and is updated only if the mass of the vehicle changes.

- *Calculating and Applying the Centring Force*

The vector between the position of the wheelset and the nearest point on the spline (ignoring the difference in the z axis) defines the direction of the force, and is stored when the lateral offset is calculated. This vector is normalised to produce a unit vector and the size of the centring force is then calculated using the GSF formula and multiplied together with this direction vector to produce a vector force, which is then is applied to the wheelset. This force is only applied while the wheels are in contact with the rails and it is still possible for the wheelset to derail, as it would in the real world, when the centring forces are unable to counteract vehicle instability or high lateral forces.

The code implementation of this calculation is included in Section 4.1.3.

3.3.15 *Debug Rendering and Visualisation Features*

In order to assist in the development, debugging and evaluation of the simulation tool, some additional debug rendering and visualisation features were implemented. The following features have been developed/enabled as part of the simulation's render loop.

- *PhysX Debug Rendering* - The PhysX API provides methods to render physics properties. This includes the visualisation of rigid bodies and contact points. [41]
- *Locomotion Visualisation Features* - These debug rendering features are mainly to visualise the features of the Locomotion tool. This includes the rendering of the track splines, the nearest point on the spline to each wheelset, flange contacts and more.

These features are discussed in more detail in Section 4.1.5.

3.3.16 *Batch Testing Features*

In order to achieve the goal of developing a rapid prototyping tool, and to enable the testing conducted in Chapter 5, Locomotion was designed to include a batch testing system. This system allows for multiple tests to be conducted automatically, without requiring user supervision.

Tests, Batches and Batch Sets

A 'Test' refers to an individual testing scenario, such as sending the vehicle along a 1km straight. In this example, the test ends either when the vehicle derails or reaches the end of the track. A 'Batch' is a collection of Tests and a 'Batch Set' is a collection of Batches.

This is illustrated in Figure 3.33, below.

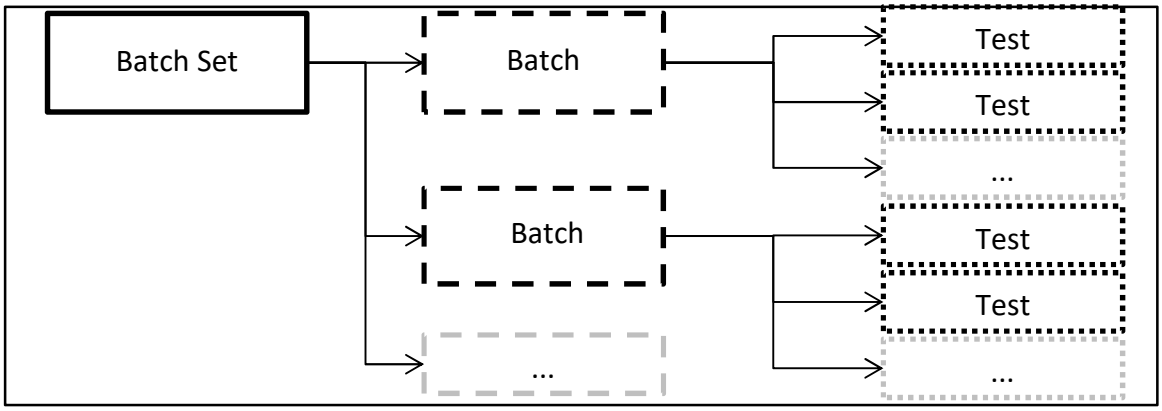


Figure 3.33 - Illustrating Batch Testing Object Hierarchy

Batch Testing

The tool is capable of performing a batch of tests, which automatically reset and restart at the end of each test. Data about each test is output when the test ends. This allows the same test to be performed multiple times, unsupervised, and for averages to be calculated. Summary data about each batch of tests is output when the batch completes.

There are a number of testing options available. For example, if testing in batch mode on a straight track, the simulation will reset when the vehicle reaches the end of the track, or if the vehicle derails. Alternatively, on looped track, the simulation can be configured to end or reset when the vehicle reaches the end of a loop, or to continue until the vehicle derails, or when a certain number of loops have been completed. These features allow for various testing scenarios. For example, it is possible to increase the target speed of the vehicle by a small amount after each loop of the track until the vehicle derails, which is useful for finding out the average speed at which the vehicle derails on each curve radius. Another option allows the user to run a batch of tests, adjust one of the simulation parameters and then repeat that batch of tests multiple times (as per Case Study 2 in Section 3.1.5), which allows comparisons to be made between the results at each parameter value and is one of the simulation's rapid prototyping features.

These batch testing features allow a test to be repeated multiple times, in order to evaluate the consistency and reliability of the tool. It is also possible to calculate the probability of derailment, if the results are not 100% consistent - i.e. if the train derails 6 times during 10 tests with a particular configuration, then derails in 8/10 tests with a different configuration, then it is possible to say that the probability of derailment has been reduced by 20%. The number of tests in each batch is a configurable parameter in simulation setup files.

Multi-Batch Testing (Batch Sets)

The Locomotion tool is also capable of running multiple batches and automatically incrementing a range of simulation parameters and configuration options between batches. This, for example, allows the user to evaluate the stable speed of the vehicle at different parameter values by incrementing the target speed of the vehicle between each batch, and then incrementing the value of the parameter currently being tested between each batch set. The number of batches in each batch set is a configurable parameter in simulation setup files and data about each batch set is output when the tests are completed.

3.3.17 *'LocoDataScan'*

A separate application, known as 'LocoDataScan' was created to process the data files that are produced by simulation after each test has been completed. This application loads the data file from each test and produces a CSV-format spreadsheet file ("results.csv") with summary/average values for all the tests conducted in that batch, allowing graphs to be drawn and data from each test to be compared more easily. The tool then iterates through all of the 'results.csv' files from each batch of tests to produce a series of spreadsheets that summarise the results from the entire batch set. This includes calculating summary data, such as the stable speed of the vehicle, which is defined as the highest speed at which no derailments occur (more details in Section 5.7). This is done by iterating through the batch set data until it finds a batch of tests where the number of derailments is greater than zero. This means that the user does not have to manually process the output data to calculate average derailment speeds, stable speeds, ranges of results, etc. for each batch or batch set. This data processing stage was not included in the main Locomotion application, so that if there is a problem, such as if the application crashes, then most of the data will not be lost and the averages etc. can be calculated afterwards. It also allows the user to combine results from separate batches/batch sets to analyse and compare the results.

3.4 Parameters

This section defines all of the key parameters of the simulation. All of the following parameters are intended to be defined in text files which are loaded into the simulation during initialisation and which the user can modify. These parameters will have to be implemented as variables of the simulation in order to allow them to be changed. Some parameters, such as mass, gravity and material properties, have real world values, while others, such as the physics engine parameters (see Section 3.4.7) do not. Where possible, parameters have been set to real-world values in order to make the simulation as realistic as possible. Parameters that do not have a defined real-world value have to be tested to see which values produce the most realistic results.

3.4.1 *Test Scenario Parameters*

These parameters control the test vehicle that is used in the test.

- Vehicle Type:
  - Wheelset - a single wheelset
  - Bogie - a single bogie (+2 wheelsets)
  - Chassis - a chassis (+2 bogies)
  - Train - a whole vehicle or multi-vehicle Train
- Number of Vehicles (if 'Train' is selected[3]):
  - If 1: a locomotive,
  - if 2: two locomotives
  - 3 or more: carriages are added between the front and rear locomotive.
- Wheelset Type: The type of wheelset to use (as described in Section 3.2.2)
  - Single Body Wheelset
  - Multi Body Wheelset
  - Conical Wheelset
- Test Environment
  - Track Layout
  - Number of Straights / Curve Radius (depending on selected layout)
- Target Speed of the Vehicle

3.4.2 *Vehicle Parameters*

These are the variable properties of the vehicle, which were defined in Table 3.1, including Wheelbase, Bogie Spacing and Carriage Spacing.

3.4.3 *Entity Parameters*

All of the entities in the scene - vehicle bodies, chassis, bogies and wheelsets - have the following properties that can be adjusted.

- *Mass* - the mass of the entity.
- *Centre of Gravity (COG)* - a vector defining the centre of gravity relative to the centre of the entity model (i.e. a value of [0, 0, -1] would lower the COG by 1m).

[3] - as illustrated in Figure 3.11

3.4.4 *Wheel/Rail Interface Material Parameters*

The following material properties can be altered on the wheel, flange and rail rigid bodies. These properties are based on the contact values for dry, steel-on-steel contact, based on multiple reference sources (including *PhysLink.com* [59]).

- Static Friction - 0.74
- Dynamic (Kinematic) Friction - 0.57
- Coefficient of Restitution[4] - 0.55

3.4.5 *Batch Testing Parameters*

The following parameters control how batch testing is conducted in the simulation tool.

- The Number of Batches
- The Number of Tests to be conducted in each Batch
- Reset on Track End - whether to end or continue testing when the vehicle reaches the end of the straight or returns to the beginning of the loop
- Accelerate on Track End - A Boolean flag defining whether the target speed of the vehicle should change at the end of the track/loop
- Acceleration - how much the vehicle should accelerate (or decelerate) by, if the above is true.
- Test Variable - Which variable to adjust between batches/batch sets (if any). This may include the vehicle parameters (described above) and the PhysX engine parameters (described below).
- Test Variable Increment - How much to increment the Test Variable (if selected)

3.4.6 *PhysX Scene Parameters*

The following parameters are properties of the PhysX scene and have therefore been set to their real-world values.

- Gravity - a vector: [0, 0, -9.806][5]

[4] - the coefficient of restitution is the ratio of relative speeds before and after a collision [41]
[5] - equivalent to the real world acceleration due to gravity (9.806m/s^2) at sea level

3.4.7 *PhysX Engine Parameters*

These are parameters of the PhysX engine (or of Rigid Bodies) that were identified from the PhysX Documentation [41] that affect its performance and accuracy, but which have no direct correlation with real-world parameters. One of the goals of this research will be to adjust these parameters in order to find a balance between performance and accuracy. These parameters are:

- Simulation Timing
- Rigid Body Solver Iteration Count
- Skin Width
- Maximum Angular Velocity
- Hardware Acceleration

By adjusting these parameters, the fidelity of the physics simulation can be adjusted without requiring access to the source code or adjusting to the internal workings of the PhysX engine. These parameters are discussed in more detail below.

Simulation Timing

The PhysX scene has a 'SetTiming' method that controls the timestep and maximum number of substeps used by the solver when the physics engine's *simulate* method is called. Below is a description of the *SetTiming* method from the Documentation [41]:

> ***void NxScene::setTiming*** ***(NxReal maxTimestep = 1.0f/60.0f,***
> ***NxU32 maxIter = 8,***
> ***NxTimeStepMethod method = NX_TIMESTEP_FIXED)***
>
> Sets simulation timing parameters used in simulate (elapsedTime). If *method* is NX_TIMESTEP_FIXED, elapsedTime (simulate() parameter) is internally subdivided into up to maxIter substeps no larger than maxTimestep ... The timestep method of TIMESTEP_FIXED is strongly preferred for stable, reproducible simulation.
>
> Parameters:
>
> | | | |
> |---|---|---|
> | [in] | *maxTimestep* | Maximum size of a substep. **Range:** (0,inf) |
> | [in] | *maxIter* | Maximum number of iterations to divide a timestep into. |
> | [in] | *method* | Method to use for timestep (either variable or fixed). |

Also from the documentation: *'The recommended time-stepping method is fixed time-steps where maxTimestep is an exact multiple of elapsedTime*[6] *and elapsedTime is a constant. This way, the user knows the number of sub steps taken, which do not vary.'* [41]

The Locomotion simulation tool uses this recommended method, including a fixed timestep of (1/60th of a second) in the call to *Simulate*.

Rigid Body Solver Iteration Count

Every rigid body has a 'solver iteration count' parameter, which defines the number of solver iterations performed when processing joints and contacts relating to that rigid body. The following description is taken from the PhysX Documentation [41]:

> ***NxReal NxActor:: solverIterationCount***
>
> The number of iterations the solver should perform for this body. The solver iteration count determines how accurately joints and contacts are resolved.
>
> **Range**: [1.0, 255.0] **Default**: 4

Increasing this parameter from its default value of 4 should improve the fidelity of the joints and contacts between the wheels, bogie and rails.

Joint Solver Extrapolation Factor

Revolute Joints, such as the joint between the wheelset/axle and the bogie, have a solver extrapolation factor that defines the rigidity of the joint. The following description is taken from the PhysX Documentation [41]:

> ***NxReal NxJointDesc::solverExtrapolationFactor***
>
> Extrapolation factor for solving joint constraints. This parameter can be used to build stronger joints and increase the solver convergence. Higher values lead to stronger joints.

[6] - 'elapsedTime' is the parameter passed to the PhysX Simulate method

Note: Setting the value too high can decrease the joint stability.
This feature is supported for D6, Revolute and Spherical Joints only.

Range: [0.5, 2.0] **Default:** 1.0

Adjusting this parameter could improve the stability of the joint between the wheelset/axle and the bogie. However, it does not apply to fixed joints and cannot be used to adjust the joint between the separate rigid bodies of the multi body wheelset.

Skin Width

The collision response system in *PhysX* resolves collisions can lead to instability; contacting or penetrating objects can repel each other to the point where they separate and then fall back down on each other in a subsequent time frame, leading to visible jittering [41]. In order to mitigate this problem, *PhysX* allows objects to slightly inter-penetrate each other. The amount of permitted inter-penetration is regulated using a parameter called Skin Width. Figure 3.34 (below) is taken from the PhysX documentation [41] and illustrates the skin width of two contacting cuboid rigid bodies.

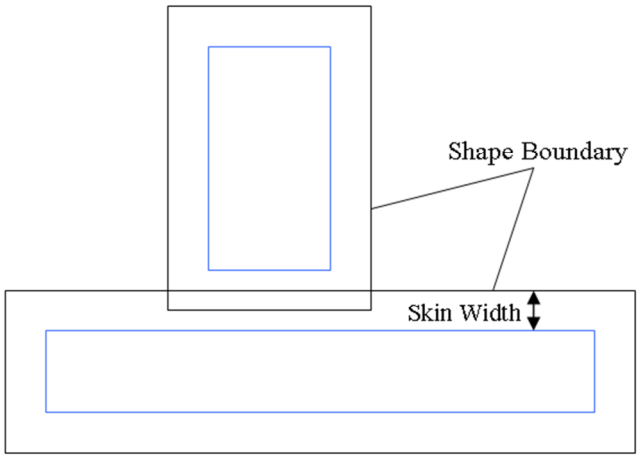


Figure 3.34 - Illustrating Skin Width [41]

The PhysX documentation [41] describes the skin width of a Rigid Body as follows:

NxReal NxShapeDesc::skinWidth

Specifies by how much shapes can interpenetrate. Two shapes will interpenetrate by the sum of their skin widths. This means that their graphical representations should be adjusted so that they just touch when the shapes are interpenetrating.

The default skin width is the NX_SKIN_WIDTH SDK parameter. This is used if the skinWidth member is set to -1(which is the default).

A skin width sum of zero for two bodies is not permitted because it will lead to an unstable simulation. If your simulation jitters because resting bodies occasionally lose contact, increasing the size of your collision volumes and the skin width may improve things.

Range: (0.0, inf) **Default:** -1.0 (use the default specified with NX_SKIN_WIDTH)

And the global parameter of the engine (NX_SKIN_WIDTH):

NX_SKIN_WIDTH

Default value for *NxReal NxShapeDesc::skinWidth*. **Range:** [0.0, inf] **Default:** 0.025

It is possible that adjusting the skin width of the wheels and rails may improve the stability of the simulation and/or prevent penetration issues.

Maximum Angular Velocity

Initially, the forward velocity of the wheelset seemed to be limited to a very low speed. It was discovered that this was due to a parameter called maxAngularVelocity[7]. Angular velocity is typically measured in radians per second or degrees per second and is often represented by the symbol ω [21]. MaxAngularVelocity as a parameter of a rigid body is described as below in the PhysX documentation [41]:

virtual void NxActor::setMaxAngularVelocity (NxReal maxAngVel)

Lets you set the maximum angular velocity permitted for this actor. Because for various internal computations, very quickly rotating actors introduce error into the simulation, which leads to undesired results.

[7] - this parameter was discovered during testing in Section 5.7

With NxPhysicsSDK::setParameter(NX_MAX_ANGULAR_VELOCITY) you can set the default maximum velocity for actors created after the call. Bodies' high angular velocities are clamped to this value.

However, because some actors, such as car wheels, should be able to rotate quickly, you can override the default setting on a per-actor basis. Note that objects such as wheels which are approximated with spherical or other smooth collision primitives can be simulated with stability at a much higher angular velocity than, say, a box that has corners.

The global angular velocity parameter is described in the documentation [41] as below.

NX_MAX_ANGULAR_VELOCITY **Range:** [0, inf] **Default:** 7

The default value for this parameter is 7. Increasing the angular velocity will increase the top speed of the vehicle, but might compromise stability.

Hardware Acceleration

PhysX includes hardware acceleration, which can be used in if a compatible NVidia Graphics card is present on the deployment platform. In order to enable hardware acceleration, the following lines of code have to be executed when the scene is created.

```
1  NxSceneDesc sceneDesc;
2  sceneDesc.simType = NX_SIMULATION_HW;
3
4  g_scene = g_physicsSDK->createScene(sceneDesc);
```

First a Scene Description object is created (line 1) and the Simulation Type is set to Hardware Mode (line 2). Then, when the PhysX scene is created (line 4), it will initialise the scene with hardware acceleration enabled, if support for hardware acceleration is available.

However, as described on the PhysX website: *'GPU hardware acceleration is used mostly for additional physics effects, like particles (fluids, dynamic smoke, debris and chunks from explosions) and cloth (clothing and hair simulation on characters, cloth banners and flags)'* [4] and not for rigid body movement or collision detection, so it may not make a significant difference to the performance of the tool, since the application doesn't make use of any of these features.

3.5 Summary

This chapter has described the design of the Locomotion simulation tool, including the design of the virtual vehicle and the environment in which testing will take place. The software architecture of the simulation has also been described, including the integration of PhysX, as well as the key phases of the application loop. Real-world values, properties and dimensions have been used in the construction of the virtual world/vehicle, where the correct information was available.

Design Choices and PhysX Parameters

Key elements of the design were also identified that will need to be evaluated, including variations in design and parameters of the physics engine. These are as follows:

- *Wheel Polygon Count*

The polygon count is number of polygons that make up the wheelset model. It is not immediately clear what is the best choice for real-time simulation of the wheel/rail interface. Higher polygon counts are expected to produce more realistic results, but potentially at the expense of performance.

- *Wheelset Design Variations*

The vehicle components include two variations on wheelset design; the Single Body and Multi Body Wheelsets. It is expected that the SB Wheelset will produce more stable results, but the MB wheelset would produce a more flexible simulation. If the MB wheelset is (or can be made) suitably stable, then it would be useful to use this variant in order to make the simulation as flexible as possible.

- *PhysX Engine Parameters*

There are five parameters of the PhysX engine that were discovered during the software design that may affect the accuracy and/or performance of the simulation tool, and which do not have corresponding real-world values. These are:

- Simulation Timing
- Rigid Body Solver Iteration Count
- Skin Width
- Maximum Angular Velocity
- Hardware Acceleration

The next chapter describes the implementation of the simulation tool, and includes screenshots of key elements and features, as well as a discussion of implementation issues and any changes made to the initial design as a result of those issues.

[BLANK PAGE]

Chapter 4

Implementation

This chapter describes the implementation of Locomotion, including screenshots of the simulation's key features, as well as details of any issues that were encountered or changes that were made to the design, during the implementation process.

4.1 The Locomotion Simulation Tool

This section includes screenshots of Locomotion that show the vehicle, testing environment and key features of the simulation tool.

4.1.1 *The Vehicle*

The following screenshots show the vehicle components.

Wheelset

One of the wheelsets is shown in the screenshot below (Figure 4.1). The orange lines represent the edges of dynamic physics meshes (i.e. the wheelset) and red lines represent the edges of static physics meshes (i.e. the rails). This is part of the PhysX debug rendering system. The orange dot represents the nearest point on the spline (green line) to the wheelset and is rendered as part of the Locomotion visualisation system. Both of these systems are discussed in more detail later in this chapter.

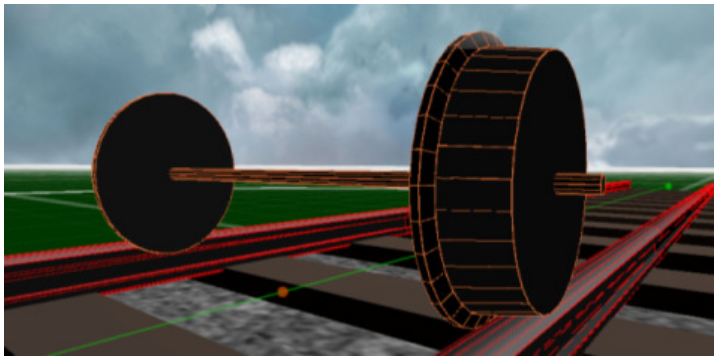


Figure 4.1 - A screenshot showing a Single Body Wheelset in the Locomotion tool

Bogies

A screenshot of a bogie is shown in Figure 4.2 (overleaf). The wheelbase can be adjusted and the wheelset is attached to the bogie via revolute joints. This screenshot shows that the physics mesh (orange) of the bogie is simpler than the 3D model. This was done to maximise performance, while making the graphical model look more realistic.

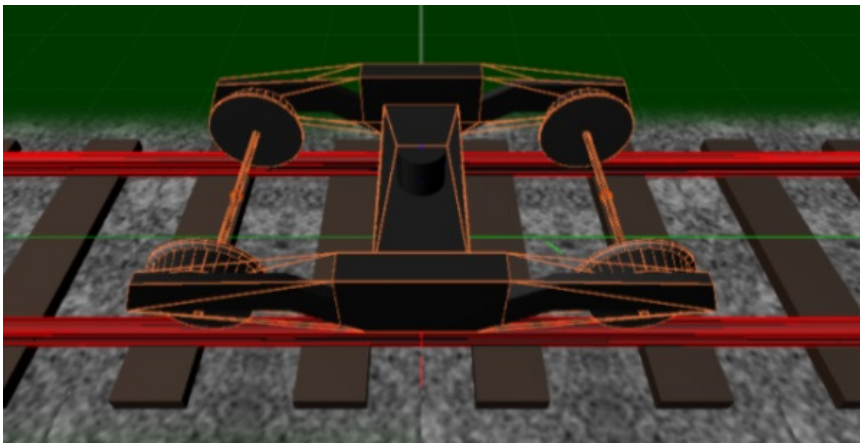


Figure 4.2 - A screenshot showing a Bogie in Locomotion

Chassis

The screenshot below (Figure 4.3) shows the chassis of the vehicle (with two bogies attached). The bogies are attached to the chassis via revolute joints and bogie spacing can be adjusted. The joints between the chassis and the bogies are freely-rotating and do not include any ability to resist yawing motion, as they do in real life (as discussed in Section 2.1.2).

This was intended to be added to the simulation at a later date, but time constraints prevented its implementation.

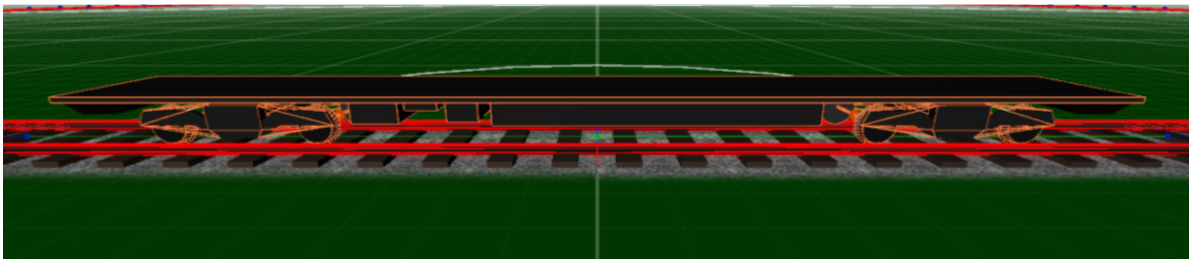


Figure 4.3 - A screenshot showing a vehicle Chassis in the Locomotion tool

- *Couplings*

If there are multiple vehicles in the scene, then the chassis of each vehicle is attached to the chassis of the adjacent vehicle(s) by a Distance Joint, which *"tries to maintain a certain minimum and/or maximum distance between two points attached to a pair of actors"* [41]. Distance joints connect the two bodies, but allow relative rotation (allowing the vehicles to corner) and allow the joint to expand and contract, based on a minimum and maximum distance variable. This was deemed to be a suitable approximation of the workings of the inter-vehicle couplings used on trains for use in this initial evaluation of the simulation tool.

Vehicle Body

Locomotives and carriages have different body objects and different mass properties, defined in the simulation setup files. Figure 4.4 shows a 'locomotive' (top) and a carriage (bottom). The carriage is a different shaped polygon, with a different texture applied to it and constructed using different material properties, but is otherwise very similar to the locomotive. The body is a single rigid body and is joined to the chassis via a fixed joint.

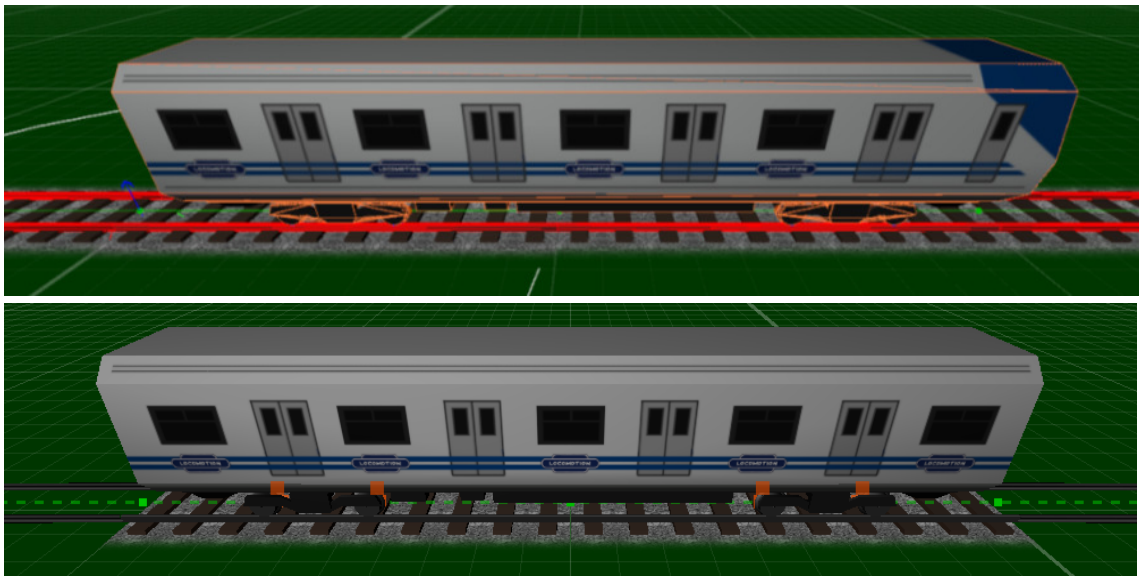


Figure 4.4 - A carriage in the Locomotion Tool

4.1.2 *Track*

The screenshot below (Figure 4.5) shows the end of a straight track section. The red lines show the edges of the static physics mesh. Also shown in the screenshot are the rail sleepers and a texture to represent the gravel ballast beneath the sleepers.

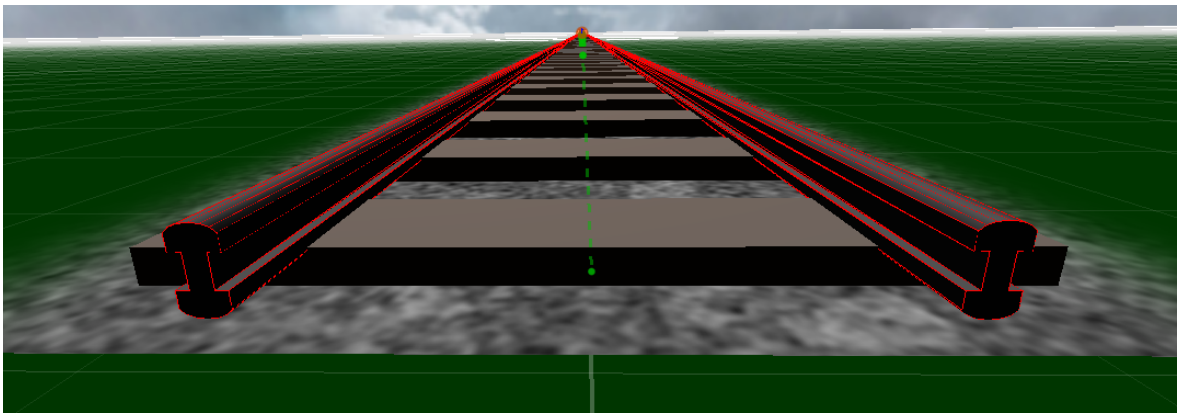


Figure 4.5 - Straight Track in the Locomotion Simulation Tool

The sleepers are not physics objects (as indicated by the lack of a red debug mesh), as this would have a negative impact on the performance of the simulation. The top of the sleepers are level with the ground plane of the scene, so derailments are detected at the point where collision with the sleepers would have occurred.

- *Rail Profile*

The screenshot below (Figure 4.6) is a taken in 3DSMax, and shows the profile of the rail.

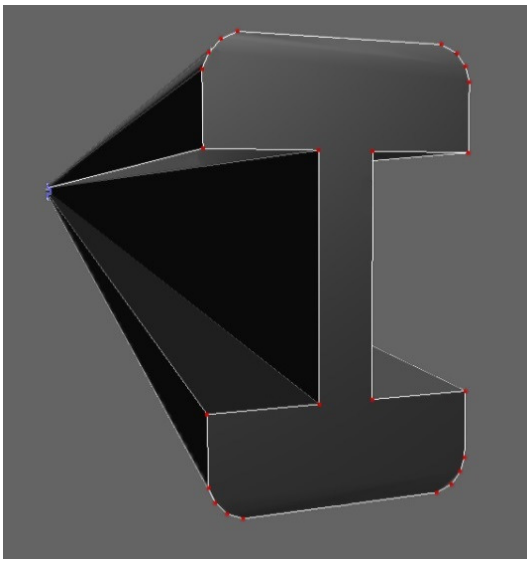


Figure 4.6 - Track Profile in 3D Studio Max

The rail polygon is made up of 24 vertices (red dots), which includes four on each curved edge. This allows an approximation of the shape of the rails that does not compromise the performance of the simulation tool.

- A straight track, when extruded along a straight spline in one segment, results in 24 polygons and 44 vertices per rail.
- A curved track model, extruded along a curved spline in 12 segments, consists of 534 polygons and 660 vertices (including both rails and regardless of curve radius).

Adjusting the number of segments/polygons would result in a more realistic, higher fidelity curve model, but would compromise the performance of the simulation.

4.1.3 *Corrective Force Implementation*

Section 3.3.14 describes how an additional, corrective centring force (based on the Gravitational Stiffness Force (GSF)) is calculated and applied to each wheelset. In this section, the implementation of that algorithm is described, including the initial implementation and an optimised version.

➢ *GSF 1.0*

Below (Figure 4.7) is the initial C++ implementation of the code that calculates and applies the centring force to each wheelset. The calculation of the size of the force (Line 9) is based directly on the Gravitational Stiffness Force formula.

```
1   float  W      = vehicle->GetWheelMass();       // effective weight
2   float  lambda = contact->wheelset->Conicity(); // conicity
3   float  l0     = contact->track->HalfGauge();   // half the gauge
4
5   Vector3 offset = contact->offset;
6   float y = offset.magnitude();              // lateral displacement
7   offset.normalise();                      // direction of the force
8
9   float Lw = (W * lambda * y) / l0 ;
10
11  Vector3 force = offset * Lw;                        // force vector
12  contact->wheelset->ApplyForce(force);
```

Figure 4.7 - Initial implementation of the calculation and application of centring force to the wheelset

First, W is retrieved (line 1) via the vehicle's 'GetWheelMass' function, which, as described in the Design, returns the effective mass of the wheelset multiplied by gravity. The conicity of the wheelset (λ) - stored as a member variable of the wheelset class - is then retrieved (line 2). The half-gauge of the track (l_0) - stored as a member variable of track objects - is retrieved also. The offset vector (between the position of the wheel and the nearest point on the spline, ignoring any difference in the z axis) is retrieved from the contact object (line 5), having been calculated by the code that records the lateral offset, as described in Section 3.3.7. '*y*' (the lateral offset) is calculated as the magnitude of this vector (line 6). The offset vector is then normalised (line 7), resulting in a unit length vector representing the direction of the force. L_w (the magnitude of the force) is then calculated using the GSF formula (line 9). The normalised direction vector is then multiplied by L_w to produce the centring force vector (line 11), which is then applied to the wheelset (line 12).

➢ *Simplifying the GSF Calculation*

The calculation can be simplified. In the version of the code shown overleaf (Figure 4.8), the calculation of y and the normalisation of the offset vector are avoided. The offset vector (which has a magnitude of y) is multiplied by L_w, which is calculated without considering y. This reduces the computational cost of the algorithm, without affecting the outcome - the size and direction of the force are identical in each version of the code.

```
5   Vector3 offset = contact->offset;
6
7   float Lw = (W * lambda) / 10;
8
9   Vector3 force = offset * Lw;
10  contact->wheelset->ApplyForce(force);
```

Figure 4.8 - Simplified calculation and application of centring force to the wheelset

4.1.4 *Simulation Features*

Screenshots from Locomotion are included below (Figure 4.9 and Figure 4.10 (overleaf)), showing the simulation's interface and other key features. The screenshot below shows a single bogie, with debug rendering enabled, including the rendering of recent contact positions (blue).

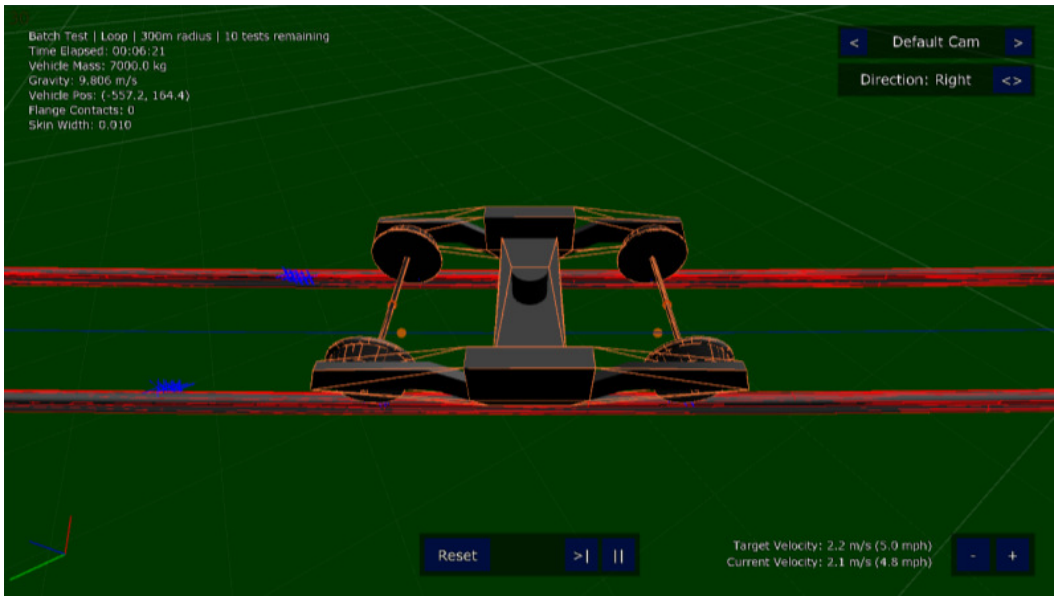


Figure 4.9 - The Locomotion Interface (showing a bogie)

Features Summary

Simulation features include:

- *3D Rendering (using OpenGL)* - In addition to the track and vehicles, rendered items include a floor grid, basic shader lighting and a skybox (these features are optional and can be turned off to improve performance if necessary).
- *PhysX Debug Rendering* - implemented using PhysX API methods, enables the visualisation of the rigid bodies and contact points (e.g. the red, orange lines and blue collision markers in the figure above). These features are not enabled by default as they negatively affect performance, but can be turned on for debugging purposes.

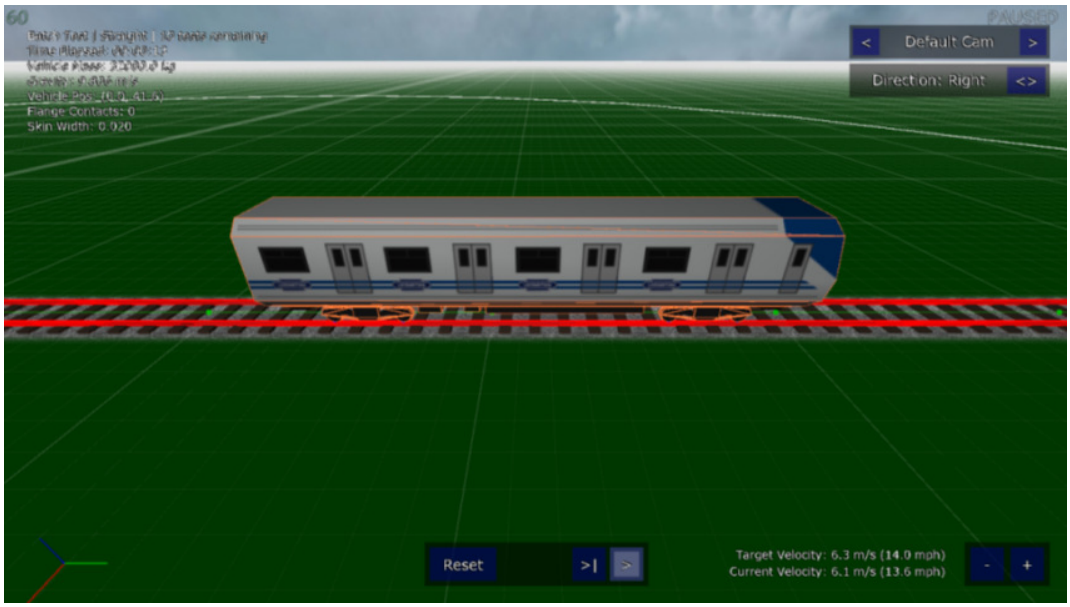


Figure 4.10 - A Screenshot from the Locomotion Tool (showing a Locomotive)

- *Heads-Up Display (HUD)* (top-left) - shows the elapsed time, the current batch number/number of batches, the number of tests remaining, vehicle position, simulation settings, the framerate, vehicle properties and the number of flange collisions, as well as the parameter being studied and its current value, if applicable.
- *Cameras* (top-right) - a range of cameras are available, including a freely-controllable camera and a number of cameras that are anchored to various parts of the vehicle. The user can select a vehicle component as the camera target and the camera will track that component as the vehicle moves. The user is also able to select the direction from which the camera views the scene.
- *Adjustable Target Speed* (bottom-right) - the user can manually adjust the target speed of the vehicle - using the plus/minus buttons - in order to adjust the target speed of the vehicle and/or simulate a driver accelerating and decelerating. This part of the interface also shows the current speed and target speed of the vehicle.
- *Simulation Controls* (bottom-centre) - these controls allow the user to pause, resume and reset the simulation, and to step through the simulation one frame at a time. Resetting the simulation restarts the current test and is intended for when the vehicle becomes stuck (if this is not detected by the derailment detection systems) or experiences other erroneous behaviour, of if the user wants to manually reset the simulation while not running in batch mode.
- *Direction Widget* (bottom-left) - The 'direction widget' is a visualisation of the x, y and z axes of the scene, which changes as the camera angle changes, to help the user orient themselves as the camera moves.

Vehicle Tracking

The simulation tracks the speed and position of the vehicle at regular intervals (based on a variable, currently set to record 3 times per second). At the end of each test, this data is written to a comma-delimited file, which can be imported into Microsoft Excel or other applications for analysis. This data could be used to reconstruct the path of the vehicle and could help to show when the vehicle became unstable in the event of a derailment.

Measuring Simulation Performance and Stability

In order to evaluate the system, the following data-tracking features were implemented:

- *Flange Collision Detection (multi-body wheelset)* - As the flanges on the Multi-body Wheelset are attached separately, the simulation can detect collisions between these rigid bodies and the rails. The simulation stores the Timestamp, Flange ID, Track Section ID, position and current speed of the vehicle for each collision. There is also an (optional) feature to render yellow markers in the scene at the flange contact locations, allowing problem areas to be identified visually (as shown later in Figure 4.12), which is not enabled by default as it affects the performance of the simulation.
- *Derailment Detection* - When the train derails, the simulation records the speed and position of the vehicle, along with the ID of the derailing component. The method of derailment detection is discussed below.
- *Measuring Forces* - PhysX can be made to report some of the forces that are applied to objects in the scene, including the normal force between wheel and rails. These values can be analysed to see if they comply with expectations (more details in Section 5.6.2).
- *Lateral Offset of the Wheelset* - for each wheelset, the distance to the centre of the track is calculated by measuring the wheelset's distance from the track spline (as described in Section 3.3.12)
- *Wheelset Height* - the height of the wheelset (the z component of its position) is used as a measure of its stability (as shown in Section 5.7.6).

This data is also written to a number of CSV output files at the end of each test.

Derailment Detection

A derailment in the simulation is defined as: any component coming into contact with the ground plane; or any wheel coming out of contact with the rails for longer than a certain period of time (a variable currently set to 0.5 seconds). During initial testing it was observed that it was possible for the vehicle to get 'stuck', while not meeting either of the previous derailment conditions. Therefore, if the vehicle was not moving (i.e. its forward velocity was 0 mph) for more than a specified period of time (currently set to 5 seconds), then it was also deemed to have derailed.

➢ *Derailment Speed*

It was necessary to calculate the derailment speed of the vehicle in the simulation, but this presented a problem. The speed at which the vehicle was travelling when the derailment was detected is unreliable because by the time the derailment is detected the vehicle may have decelerated. The 'target speed' is the speed at which the vehicle should have been travelling when the derailment occurred, however it is possible that the vehicle was unable to reach the target speed and so this value is not reliable either.

Instead, the following approach was devised. At regular intervals, the velocity of the vehicle is measured and, if all of wheels of the vehicle are in contact with the rails, then if the current velocity is higher than the currently recorded top speed, then the top speed is overwritten (unless the speed is greater than the target speed of the vehicle, which could suggest that an erroneous behaviour has occurred). This value is assumed to be the most accurate representations of the vehicle's derailment speed that can be determined from the available data, and it is this value that is presented as the 'derailment speed' in Chapter 5.

➢ *Stable Speed*

An alternative measurement of the vehicle's top speed that is used in Chapters 5 and 6 is the 'stable speed'. The vehicle is tested at a range of speeds and the 'stable speed' is defined as the highest speed at which no derailments occurred.

4.1.5 *Simulation Visualisation Features*

The simulation includes a number of features designed to visualise the behaviour of the objects in the simulation.

➢ *Wheelset Contact Visualisation*

As Figure 4.11 (below) shows, the simulation displays a visualisation of the wheelset and properties of its contact with the rails.

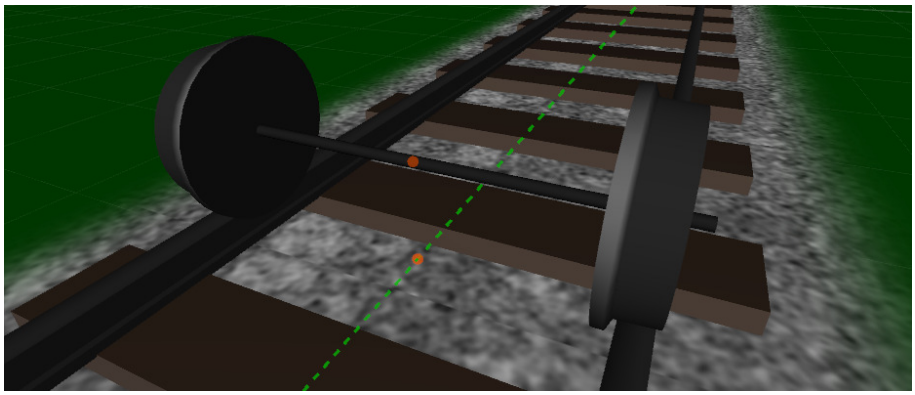


Figure 4.11 - A screenshot of Wheelset Contact Visualisation in Locomotion

The green dotted line represents the track spline. The orange dot on the wheelset represents its position and the orange dot on the spline represents the nearest point on the spline to the wheelset. This helps to visualise the behaviour of the wheelset to and ensure that the nearest point on the spline is being calculated correctly.

- *Flange Collision/Hotspot Detection*

When testing using the Multi Body Wheelset, the simulation records all collisions between the flanges and the rails. The offending flange is highlighted in yellow to show that a collision has occurred/persists, as illustrated in the screenshot below (Figure 4.12), where the two right hand flanges are grinding along the outer rail of a curve in the track.

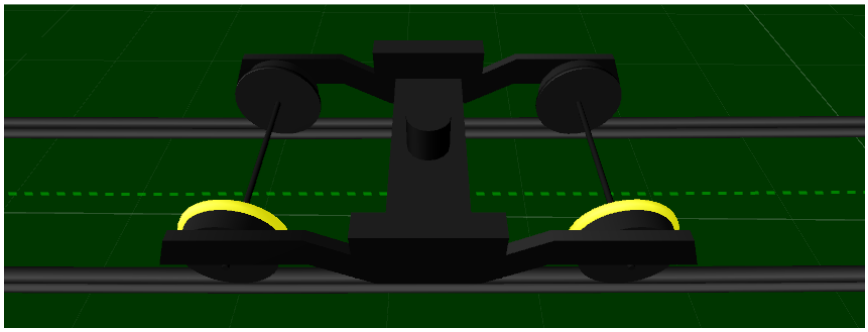


Figure 4.12 - a screenshot showing flange collision visualisation in Locomotion

The position of each flange collisions is stored by the simulation and can be used to render yellow markers in the scene, as illustrated in the screenshot below (Figure 4.13), which shows a number of contacts from the right hand flanges of the wheelset on the outer rail of a track curve.

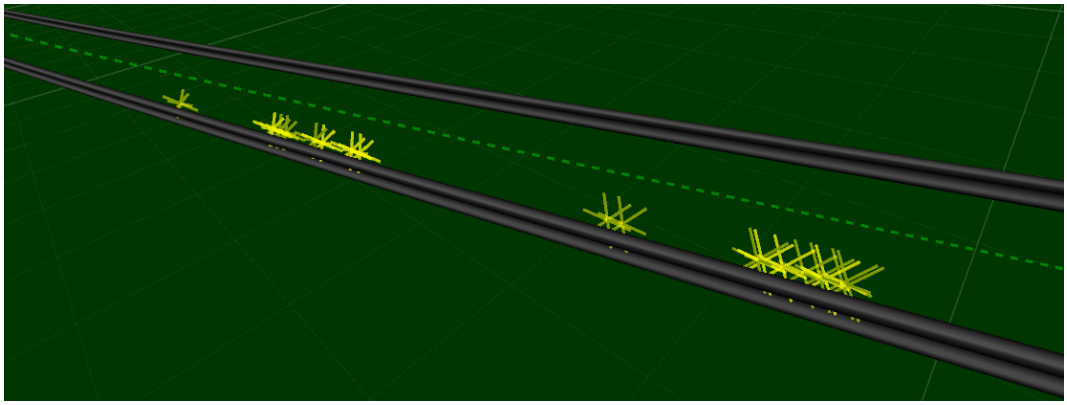


Figure 4.13 - A screenshot showing flange collision visualisation in Locomotion

This enables the user to visually identify hotspots where such collisions regularly occur. This feature is disabled by default as rendering the markers affects the performance of the simulation tool, but can be enabled at any time with a key press.

➢ *Derailment Visualisation*

In the event of a derailment (if the simulation is not running automatic batch tests), the simulation will pause and the component that was detected to have derailed is highlighted in red, as illustrated in the figure below (Figure 4.14).

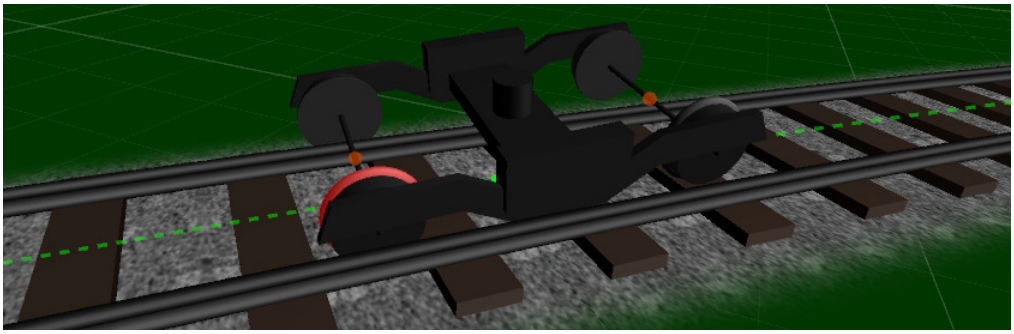


Figure 4.14 - a screenshot showing derailment visualisation in Locomotion

This visualisation allows the user to identify that, in this case, the left flange of the front wheelset derailed first, after the front wheelset climbed the rail and the flange came into contact with the ground plane.

➢ *Wheelset Offset/Hunting Oscillation Visualisation*

When testing with a single wheelset, the simulation is able to visualise the wheelset's path, in order to show the changes in lateral offset over time. This is illustrated in the screenshot below (Figure 4.15); an example of hunting oscillation with the conical wheelset. The orange line shows the path of the wheelset.

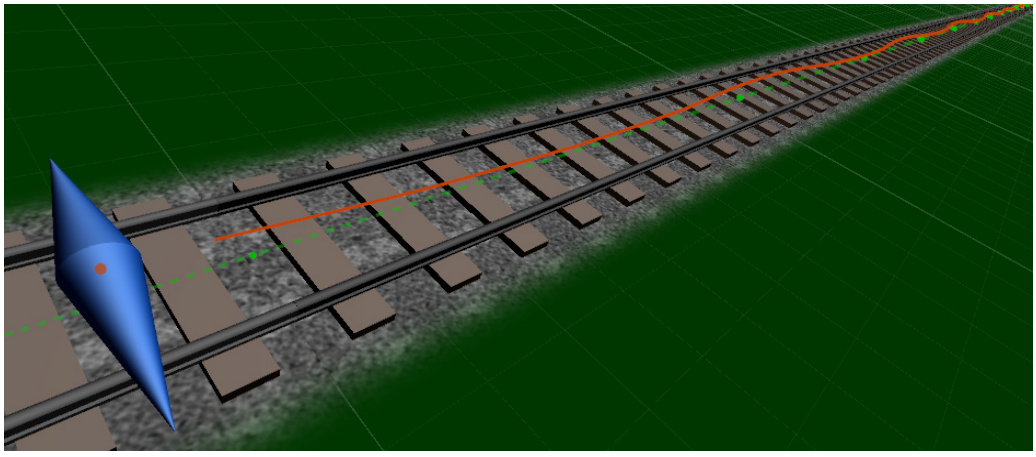


Figure 4.15 - a screenshot showing the recorded path of the wheelset in Locomotion

It is necessary to limit the number of measurements stored, in order to prevent the application from running out of memory. The visualisation is reset when a test ends and a maximum of 1,000 entries are stored.

4.1.6 *Interface and Customisation Options*

This section shows examples of some of customisation menus that were added to the simulation, based on the Menu Flow described in Section 3.3.11. At the start of the simulation, the user is presented with a menu that allows them to choose to either load simulation settings from the setup files or to configure the simulation manually. This initial menu is shown in Figure 4.16 (below).

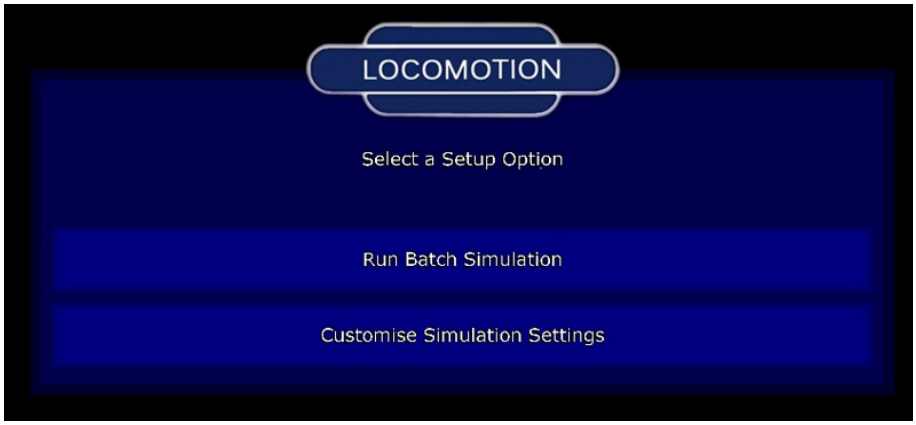


Figure 4.16 - The Locomotion 'Simulation Start-up' Menu

Clicking "Run Batch Simulation" runs the simulation with batch settings loading from the configuration files. Clicking "Customise Simulation Settings" initialises the customisation menu flow, the first menu of which is shown in Figure 4.17, below.

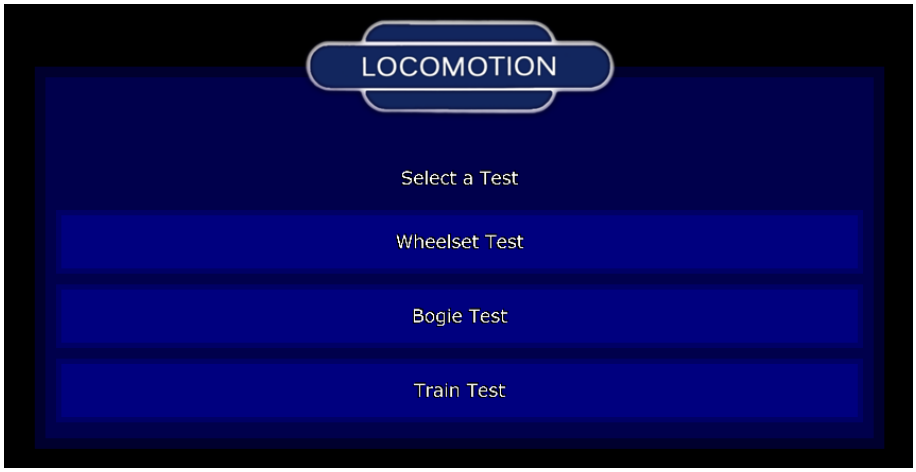


Figure 4.17 - The Locomotion 'Testing Mode' Menu

This menu allows the user to select which type of test is to be conducted (Wheelset, Bogie or Vehicle testing, as described in Section 3.3.9). The user is then presented with additional options, depending on their choices.

Most of the other menus are simple, multiple choice button menus, as in Figure 4.17, but others allow a wider range of options, such as the menu shown in Figure 4.18 (below).

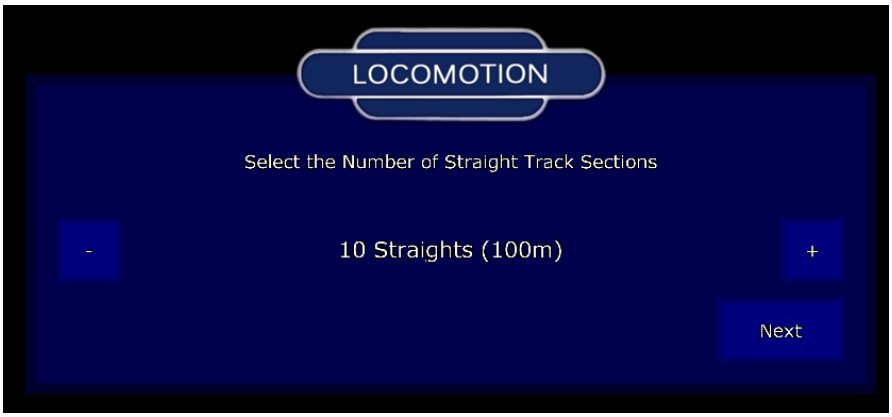


Figure 4.18 - The Locomotion 'Straight Track Layout' Menu

This is the Straight Track Layout menu, which allows the user to select the number of straight track sections that will be used in the scene. The '+' and '-' buttons allow the user to increase or decrease the number of straights, and the menu shows the total track length - in this case, 10 straight sections resulting in a test track 100m in length. A similar menu is used to select the number of carriages and the curve radius of the Loop layout.

4.1.7 *Visual Debugger*

Throughout development and testing of the simulation, the PhysX Visual Debugger (VDB) was used to study the behaviour of the bogie. The VDB records the positions and properties of all the objects in the PhysX scene while the simulation is running, and this data can then be viewed alongside the simulation or replayed once the simulation has terminated. It provides a visualisation of the recorded data and allows the physics simulation to be replayed.

It is possible to move forward and backward through the replay, and to select individual rigid bodies and observe their properties. It was used mainly to debug the simulation tool during development, but a number of tests were replayed in the VDB to study the behaviour of the vehicle.

A screenshot of one of the bogie tests recorded by the VDB is shown in Figure 4.19 (overleaf). This screenshot shows the 3D visualisation (centre) as well as various configuration options (right) and the inspector (left), which allows the user to inspect the properties of rigid bodies. In the VDB, the green objects represent dynamic rigid bodies - such as the wheelsets and bogie - and red objects represent static rigid bodies, like the rails. The ground plane is also visualised.

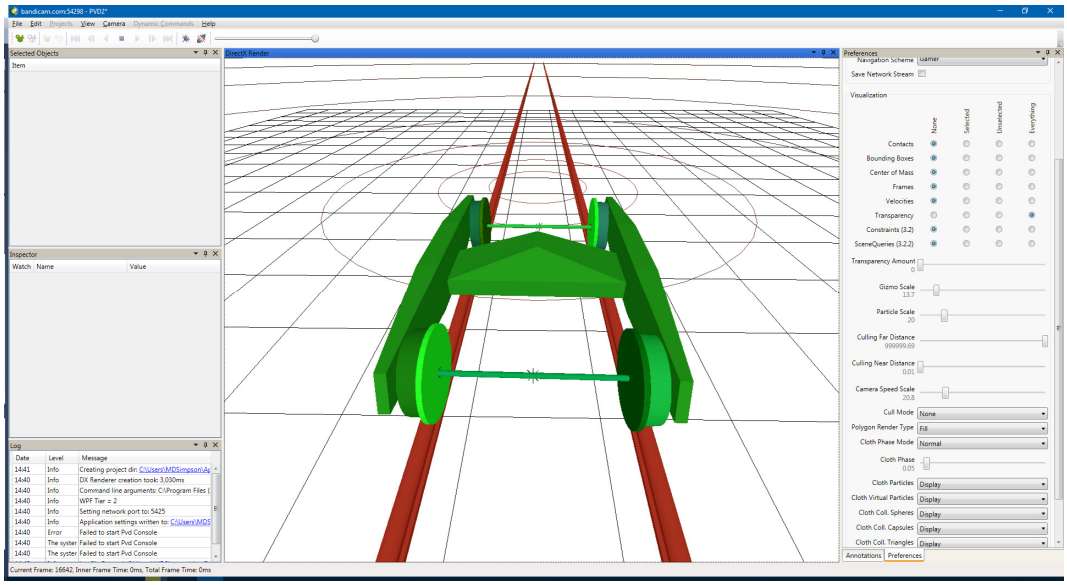


Figure 4.19 - The PhysX Visual Debugger

4.1.8 *Adjusting the Parameters*

In order to enable batch testing, it was necessary to implement features that enable the physics engine parameters to be adjusted between tests while the program is running.

Joint Solver Extrapolation Factor

The Solver Extrapolation Factor (SEF) is a property of a joint and the following code (shown in Figure 4.20) is used to iterate through all of the joints in the scene until it finds the correct joint (with the associated actor names), and then changes the SEF of the joint.

```
void PhysX2Engine::EditJointSolverValue(cstring actorname, float solverValue)
{
    g_scene->resetJointIterator();
    NxJoint* joint = g_scene->getNextJoint();

    while(joint)
    {
        NxActor* actor1, *actor2 = NULL; joint->getActors(&actor1, &actor2);

        if (StringIs(actor1->getName(), actorname)
                                    || StringIs(actor2->getName(), actorname))
        {
            joint->setSolverExtrapolationFactor(solverValue);
        }

        joint = g_scene->getNextJoint();
    }
}
```

Figure 4.20 - Code for Altering the Joint Solver Extrapolation Factor

Per-Rigid Body Parameters

Skin width and Max Angular Velocity are properties the rigid bodies and can be adjusted through the Physics Actor interface.

Timing Multiple

The number of substeps and the timestep used by PhysX when 'simulate' is called are properties of the PhysX scene, and these parameters are set using the scene's 'setTiming' method, which is called during the initialisation of the scene (as described in Section 3.4.7) and can be called between tests during batch testing.

4.2 Implementation Issues

The following issues were discovered during the implementation of the simulation tool and some additional design decisions had to be made.

4.2.1 *Hardware Acceleration*

An attempt was made to initialise PhysX with hardware acceleration. The following code is taken from locomotion and deals with the initialisation of the PhysX scene.

```
1   NxSceneDesc sceneDesc;
2   sceneDesc.simType = NX_SIMULATION_HW;
3
4   g_scene = g_physicsSDK->createScene(sceneDesc);
5
6   if( !g_scene )       // scene init failed
7   {
8         sceneDesc.simType = NX_SIMULATION_SW;
9         g_scene = g_physicsSDK->createScene(sceneDesc);
10  }
```

To enable hardware acceleration, the PhysX scene is initialised with a Scene Description object that has the *NX_SIMULATION_HW* flag set (lines 2 and 4). This initialisation fails if hardware acceleration cannot be enabled, and the code detects the failure (line 6) and the scene can then be reinitialised using Software Simulation (lines 8 and 9).

Initialisation with Hardware Acceleration fails on each of the three computers on which the Locomotion tool was developed and tested. Further investigation suggests that none of these computers has the correct version of graphics card to work with this particular version of the PhysX SDK. However, as mentioned in Section 0, hardware acceleration is used mainly for additional physics effects, rather than for rigid body movement or collision detection [4], so it is unclear whether this would have made a significant difference to the performance of the simulation tool.

4.2.2 *Framerate*

Code was added to the application to track the framerate of the simulation, which allows the performance of the simulation to be evaluated. This is done by calculating how much time has passed since the last call to the render function and calculating how many frames per second the simulation is rendering.

Initially, the framerate of the simulation was limited to a maximum speed of 60 frames per second (FPS). This was discovered to be due to be a setting in OpenGL known as Vertical Synchronisation (VSync), which synchronises the framerate of the application with the refresh rate of the computer's monitor. This feature has been disabled, enabling the application to run at a higher framerate, which in turn allows the effect of altering the various simulation parameters on the performance of the tool to be evaluated. Since the timestep used in the simulation is fixed ($1/60^{th}$ of a Second), this means that the simulation is capable of running at faster than real-time speeds (if it runs at more than 60FPS), but this should not affect the results.

4.2.3 *Stability Issues*

A considerable amount of time was spent integrating PhysX into the Simulation's application flow, as well as fixing bugs in the system that caused the simulation to become unstable. One such issue caused the vehicle to shake violently and to fly off the rails as the simulation struggled to resolve the forces and joints between the vehicle components. After much investigation, this was discovered to be an issue with the mass and inertia properties of the Rigid Bodies. The 'setMass' function of the rigid bodies sets the mass of the object, but does not update the inertia properties of the object - inertia being the resistances of a physical object to any change in its state of motion - so the objects had a large mass, but the inertia properties of a much lighter object. Physics Engines pre-calculate the inertia tensor (which describes the body's mass distribution) of a rigid body using the shapes that make up the rigid body when the actor is created. [41] This was fixed by using the 'updateMassFromShapes' function to update the mass instead of 'setMass', which forces the engine to update the inertia of the object based on the shapes that make up the rigid body and its mass, but discovering the cause of this issue and correcting it cost a considerable amount of development time.

4.2.4 *PhysX Classes*

As discussed in Section 3.3.4, wrapper classes were created around key elements of PhysX, in order to allow the physics engine to interface with the Locomotion tool. In addition to the Physics Engine and Physics Actor classes described earlier, it was also necessary to implement:

Physics Pose

The Physics Pose is wrapper around the 'pose' of an actor, which in PhysX is a 3 x 4 matrix that stores the position and orientation of an object. This enables, for example, the initial position and orientation of an object to be saved and then restored when the simulation is reset.

Static and Dynamic Physics Actors

It was also necessary to create two separate implementations of the Physics Actor class for Static Actors (such as the rails) and for Dynamic Actors (such as the wheels, bogies etc.), because Dynamic and Static Actors in PhysX have different functionality. Both inherit from a base Physics Actor class, which defines all functionality common to both actor types, and then each actor type encapsulates the unique functionality of the associated PhysX Actor.

4.2.5 *Rigid Body User Data*

In order to enable the correct processing of contact between objects within the system, it was necessary to be able to determine the Entity object associated with a PhysX Rigid Body. Each rigid body has a parameter called "userData", a void pointer in which it is possible to store a pointer to the Entity object. Then, when a collision is detected by the PhysX Engine, the Entities associated with each colliding rigid body can be retrieved and passed to the main simulation object, which processes the collision during the next update loop.

This enables the simulation to, for example, retrieve a pointer to the Wheelset entity from its rigid body when PhysX detects a collision with the rails, so that the conicity of the wheelset can be retrieved and used in the calculation of the centring forces (since conicity is a property of the Wheelset entity and not the rigid body).

4.2.6 *Crashes and Memory Leaks*

A considerable amount of time was spent testing the simulation tool before the data presented in this thesis was collected, in particular the batch testing features. In order to collect data during tests conducted across a range of parameter values and at a range of different speeds, it is necessary for the simulation to run for several hours at a time. Over time, data is accumulated in memory and if the simulation runs out of memory it can crash and simulation test data may be lost. Efforts were therefore made to ensure that the data in the simulation was stored efficiently, and was properly deleted once a test was completed and that the data was successfully output to text files when it was no longer required. Data is only output to text files at the end of each test, as doing so during the test would have a significant impact on the performance of the tool.

During one test scenario, for example, the simulation would crash after approximately 7 batch sets when it was asked to complete 10. However, if it was instructed to only attempt 5 batch sets per execution, then it did not fail. In this case, it was necessary to run two separate batch sets and then combine the output files manually so they could be processed by the LocoDataScan utility.

4.3 Chapter Summary

This chapter has discussed the implementation of the Locomotion simulation tool, including some of the issues that were encountered and some of the changes and additions to the simulation's design that were necessary. Also described are additional features that were added to allow the tests conducted in Chapters 5 and 6 to be carried out.

Some of the key simulation features and data tracking functionality are also described, including the vehicle, testing environment, visualisation and data tracking features.

The next chapter describes the testing that was conducted using the simulation tool.

Chapter 5

Wheel/Rail Interface Testing

This chapter presents the evaluation of the simulation of the wheel/rail interface in the Locomotion Simulation tool, as this was deemed to be an important potential application area for a real-time rail dynamics simulation tool. Testing includes verifying the behaviours of basic objects and the wheel/rail interface using traditional mathematical techniques, as well as testing conducted on wheelsets and bogies in motion, using the Nadal Limit as a benchmark. After initial testing showed the limitations of the tool when the default parameters of the PhysX engine were used, tests were conducted to evaluate the effect of adjusting the PhysX parameters identified in Chapter 3 on the stability and performance of the simulation. An evaluation of the new technique that was designed to further improve the real-time simulation of the wheel/rail interface is also presented.

5.1 Computer Specifications

The development and testing of the Locomotion tool was conducted on three different devices, but all of the testing presented in this chapter was conducted on a single desktop PC with the following specifications:

- *Operating System:* Microsoft Windows 7 Enterprise Edition (64bit)
- *Processor (CPU):* Intel Core i7-2600 @ 3.4 GHz
- *Graphics Card (GPU):* NVidia GeForce GTX 590
- *Memory (RAM):* 16.0GB

Several identical tests were conducted on multiple machines, producing similar if not identical results. Sample data from these tests is included in Section 6.3 in Chapter 6.

5.2 Testing Goals

As described in Section 3.1.4, the high-level testing plan is to begin by evaluating simplified scenarios, before adding more complexity and additional features to the simulation, in order to evaluate the ability of the tool to simulate rail vehicle dynamics. This high-level plan can be broken down into a series of objectives, which are summarised below.

5.2.1 *Verify the Behaviours of Simple Objects*

Boeing and Braunl [42] has shown that Novodex engine (on which version 2.8.4. of the PhysX engine is based) can produce realistic behaviours for simple objects and scenarios, as discussed in Section 2.4.4.

Section 5.6 includes tests intended to verify that the Physics Engine is producing correct behaviours for simple objects that can be easily verified mathematically. These tests include an analysis of the forces acting between simple shapes, such as spheres and cubes, as well as a static wheelset.

5.2.2 *Evaluate the Dynamic Behaviour of the Wheel/Rail Interface*

The next step is to evaluate dynamic wheelset behaviour, by measuring the lateral offset and stability of a wheelset in motion, to determine if the simulation of the WRI is realistic, whether behaviours such as hunting oscillation can be observed and whether the wheelset derails at speeds close to the Nadal Limit predictions. This process will include evaluating the effect of adjusting the polygon count of the wheel, as discussed in Section 3.3.2, and a comparison between the Single Body and Multi Body wheelsets, described in Section 3.2.2, to determine which is better suited to the simulation of the wheel/rail interface.

This testing goal includes the following sub-goals:

Evaluate Straight Line Speed

One goal of this testing is to determine (and maximise) the top speed of a vehicle in the simulation; to enable the simulation to work at the normal operating speeds of most trains. The aim is that the vehicle should be able to travel stably at speeds of up to 100mph. It is possible that solver error or instability will limit the top speed, but this may be improved by adjusting physics engine parameters. However, if this target speed is not achievable, the top speed and stability of the vehicle should be optimised (i.e. it is better to have a bogie vehicle stably at 70mph than unstably at 100mph).

- Target Straight Line Speed: 100mph

Evaluate Cornering Behaviour

Using the Nadal Limit, predictions have been made about the derailment speed of a single wheelset and a bogie for a range of curve radii (See Sections 5.5.1 and 5.5.2). The intention is to verify that the wheelset/bogie derails at speeds close to those predicted by the Nadal Limit. In cases where the predicted derailment speed of the vehicle is greater than the target speed of 100mph (or the maximum stable speed of the vehicle), tests will be conducted to verify that the vehicle does not derail on curve radii and at speeds where it should not.

- *Target derailment speeds:* See Nadal Limit Predictions (Section 5.5).

5.2.3 *Attempt to simulate multi vehicle trains*

A key goal for this research is to attempt to simulate multi-vehicle trains. If wheelsets and bogies can be simulated correctly, the next step will be to attempt to simulate entire vehicles, and then multi-vehicle trains. These will initially be abstracted models, with additional features (such as suspension) being added later. The goal will be to evaluate the effect of adding additional vehicles on the stability of the train and the performance of the simulation tool, and to see if it is capable of executing such simulations in real-time (or near-real-time).

5.2.4 *Determine Simulation Suitability for a range of Applications*

To use the data produced by the simulation tool to attempt to assess its strengths and weaknesses and to suggest which areas of rail vehicle dynamics it may be suitable for simulating. Even if the simulation of the WRI is not sufficiently accurate for wheelset behaviour or derailment simulation, the simulation tool may be useful for other purposes, such as Gauge Testing or as a Rapid Prototyping Tool. [8]

5.2.5 *Evaluate Simulation Performance*

Another goal of this testing is to evaluate the performance of the simulation tool. The target framerate of the simulation is 60 Frames-per-Second (FPS), which is real-time with the parameters that are used in the physics engine's simulate method (described in Section 3.3.2), and the aim is that any improvements to accuracy should not compromise this framerate. Vertical sync has been disabled to allow the simulation to run at speeds higher than 60FPS so that any changes in performance can be evaluated. The performance of the simulation tool during each phase of testing is presented at various points throughout this chapter.

5.2.6 *Evaluate System Error and Consistency*

Each test conducted in this chapter will be run multiple times, in order to aid in the evaluation of the error bound in the tool and allow the consistency of the results produced by the simulation to be evaluated. It is important that the simulation is consistent, as well as accurate, if it is to be used by rail engineers in any meaningful way. This will be achieved by measuring the average, range and standard deviation of the results, as well as the difference between the observed results and the predictions.

---

[8] - This discussion is included with the Sample Test Data in Chapter 6

5.3 Parameters and Design Choices

As discussed in Chapter 3, the following are parameters of the physics engine or design choices where it is not a clear which is the best choice for simulating rail dynamics. Evaluating these parameters/design choices is another key focus of testing in this chapter.

5.3.1 *Simulation Parameters*

In the event that PhysX, using its default parameters, was unable to simulate the wheel/rail interface realistically, the following parameters were identified as having the potential to affect the fidelity of the simulation and/or the stability of the wheels. These parameters - unlike mass, centre of gravity or material properties - do not correlate to real-world physics properties and so there is no obvious 'correct' value for them. They have therefore been adjusted to study their effects on the simulation; to see if adjusting them can improve the results and how doing so affects simulation performance. These parameters were described earlier, in Section 3.4.7, and are listed below.

- Maximum Angular Velocity
- Skin Width
- Simulation Timing
- Rigid Body Solver Iteration Count
- Joint Solver Extrapolation Factor

The aim is to find an 'ideal' value for each parameter; where the wheelset/bogie/vehicle is capable of reaching the intended speeds and where the derailment speeds are as close as possible to the Nadal Limit predictions. Increasing the fidelity of the simulation by adjusting parameters such as Simulation Timing and Solver Iteration Count is expected to reduce the performance of the simulation and lower its framerate, and so the other purpose of this testing is to see if the fidelity of the physics simulation can be improved without compromising the real-time performance of the simulation tool.

5.3.2 *Simulation Design Choices*

The following are design choices, discussed in Chapter 3, where there was not an obvious, correct decision. Tests were conducted to attempt to determine which choices are the most suitable for simulating rail dynamics.

Wheelset Polygon Count

Three wheelset variations have been constructed, each with a different number of segments used to create the wheel cylinder. It is expected that higher polygon counts will produce more stable results, but at the cost of simulation performance. It will be necessary to collect data on each wheelset variation in order to determine which design produces the best compromise between stability and performance.

Wheelset Design Variations

There are also two designs of wheelset, featuring different methods of construction from the component parts (axle, wheel, flange). Here is a reminder of the potential strengths and weaknesses of the wheelset designs:

- The *Single Body Wheelset* comprises a single rigid body
- The *Multi-body Wheelset* comprises five separate rigid bodies

The SB wheelset should be the more stable of the two, however it is not as flexible and may provide less data about contact forces and flange contacts. The MB wheelset is more easily adjusted - making the simulation to be more flexible, allowing it to function better as a rapid prototyping tool. It will also enable easier flange collision detection and will provide more detailed contact force reporting (as discussed in Section 5.6.3), but it is expected to be less stable than the SB Wheelset. However, if the stability difference turns out to be minimal, the additional flexibility might make the lower stability acceptable to engineers. It is also possible that adjusting the parameters of the physics engine may alleviate any stability issues.

5.3.3 *New Real-time Wheel/Rail Interface Simulation Technique*

The Gravitational stiffness force, described in Section 2.2.1, was identified as a key component of the behaviour of the wheel/rail interface and a new, spline-based technique, described in Section 3.3.14, was designed to simulate this effect if it was not found to be present or correct in the physics simulation. The effect of this technique on the results produced by the simulation tool, as well as on its performance, is presented at various points throughout this chapter (Testing Phases 2 and 3).

5.4 Testing Plan

A summary testing plan was presented in Chapter 3. The following is a more detailed plan of the phases of testing presented in this chapter and Chapter 6.

5.4.1 *Phase 1 - Test forces between simple, static objects*

As discussed in Section 2.4.4, *Boeing and Braunl* (2007) [42] showed that physics engines generally performed well in tests with simple objects. *PhysX* version 2.8.4 is still based on the *Novodex* engine which featured in these tests, and so this thesis assumes that the observations made in that paper still apply. The tests conducted during this research were designed to confirm these results before testing a wider range of scenarios relating to the simulation of rail vehicles. First, tests are conducted to verify the behaviour of simple objects, to confirm that the Normal force is being correctly applied and to test the objects with mass values closer to those of a rail vehicle to ensure that increasing the mass of the objects does not cause stability or penetration issues.

This phase of testing will also include verifying that the normal forces are being applied correctly to a static wheelset resting on the rails.

- These results are presented in Section 5.6.

5.4.2 *Phase 2 - Dynamic Wheel/Rail Interface Testing*

The next phase of testing will focus on the dynamic simulation of the wheel/rail interface. Testing will involve a wheelset in motion on simple track layouts and, if necessary, adjusting the parameters of the PhysX engine or augmenting the simulation with additional corrective forces, in order to improve the results. These tests include comparisons of wheelset polygon counts and comparisons between the SB vs MB wheelsets.

- Straight Track results are presented in Section 5.7.
- Curved Track results are presented in Section 5.9.

5.4.3 *Phase 3 - Bogie Testing*

The next phase involves testing the dynamic behaviour of a rail bogie. Tests are conducted on straight track to evaluate the speed and stability of the bogie; to evaluate its top speed and attempt to get it up to the target speed of 100mph. Tests are also designed to evaluate the cornering behaviour of a bogie, using a looped track and benchmarked against the Nadal Limit. As with Phase 2, the PhysX engine parameters or augmentations to the simulation may be adjusted to improve the results, if necessary.

- Straight Track Data is presented in Section 5.10.
- Curved Track Data is presented in Section 5.11.

5.4.4 *Phase 4 - Test with full vehicles and multi-vehicle trains*

Phase 4 was to involve testing with whole vehicles and multi-vehicle trains, primarily to evaluate the performance of the tool when simulating these more complex scenarios.

- Time constraints prevented a detailed evaluation of these scenarios, but sample data is included in Chapter 6.

5.4.5 *Phase 5 - Evaluate Suitability for Gauging and Rapid Prototyping*

Phase five was to involve testing to evaluate the ability of the tool to perform some of its intended functions; as a gauge testing environment and rapid prototyping tool.

- Time constraints prevented a detailed evaluation of these scenarios, but sample data is included in Chapter 6, which has allowed an initial analysis of the suitability of the simulation tool for these intended application areas.

5.4.6 *Additional Information*

Specific details about the design and predicted results for each set of tests are presented in the relevant sections, along with the results. Additional observations and extra tests are also described, where necessary. There is a full discussion of the significance of the results at the end of each testing phase.

5.5 Predictions

The following predictions have been made for use in the evaluation of the wheel/rail interface.

5.5.1 *Nadal Derailment Predictions (Wheelset)*

In the absence of real-world or simulated derailment data for the vehicle in question, the main method for evaluating the behaviour of the wheel/rail interface is the following set of predictions made using the Nadal Limit. Predictions are made for a single wheelset and for a single bogie, based on the Nadal Limit, and estimate the approximate speed at which the vehicle should derail (Sections 5.5.1 and 5.5.2). The expected results of adjusting physics engine parameters is also discussed (Section 5.5.3).

The 'Nadal Value'

The simulation and Nadal Limit predictions use the following values:

- Flanges have a maximum contact angle (δ) of 65°.
- The coefficient of (dynamic) friction (μ) is 0.57.[9]

Using these values in the Nadal equation produces the following result:

$$\frac{L}{V} = \frac{tan(65) - 0.57}{1 + 0.57 * tan(65)} = \mathbf{0.709}$$

[9] - Assuming that the wheel and rails are made of Steel, as discussed in Section 3.4.4.

Calculating L and V

To make use of the Nadal value it is necessary to calculate the Lateral (L) and Vertical (V) force acting on the wheelset.

- *Lateral Force (L)*

The lateral force (L) is approximated using the centrifugal force of the wheelset. Centrifugal force (F_c) is calculated using the following formula:

$$F_c = \frac{mv^2}{r}$$

where m is the mass of the object, v is the forward velocity and r is the curve radius. 'm' is the mass of the wheelset (500kg). Therefore:

$$L = \frac{500 * v^2}{r}$$

Using this formula, L can be calculated for a range of velocities (v) and track radii (r).

- *Vertical Force(V)*

The vertical force (V) is calculated using acceleration due to gravity (9.806m/s^2) and the effective mass of the wheelset.

$$f = ma \quad \therefore \quad V = 500 * 9.806 = \mathbf{4,903N}$$

Calculating the Predicted Derailment Speeds

Using the above value of V results in the following equation:

$$\frac{L}{4{,}903} = 0.834$$

So, $$\mathrm{L} = 0.834 * 4{,}903 = 4{,}089.102$$

Since $\mathrm{L} = \frac{\mathrm{mv}^2}{\mathrm{r}}$ the following formula can be used to calculate how changes in velocity (v) and curve radius (r) affect the L/V ratio:

$$\frac{500 * \mathrm{v}^2}{\mathrm{r}} = 4{,}089.102$$

So, to calculate the L/V ratio for a combination of v and r, Formula 5.1 (below) is used:

$$\frac{v^2}{r} = \frac{4{,}089.102}{500} = 8.178$$

Formula 5.1 - The formula used to make predictions based on Nadal

Predictions (100 – 300m)

Solving this formula for a range of velocities and radii (from 100m to 300m) produces the curves in the graph in Figure 5.1, below.

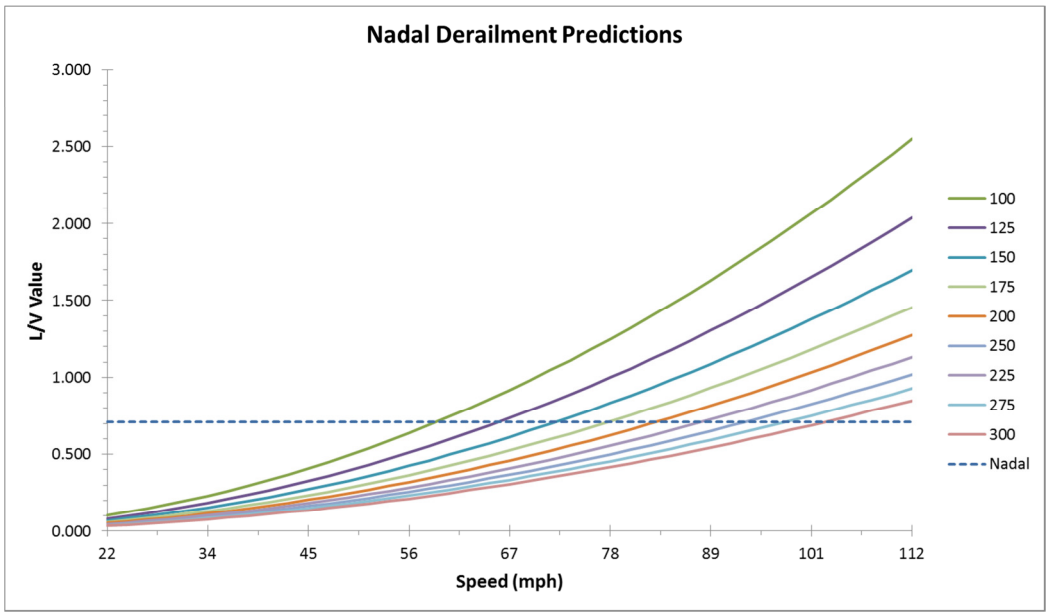


Figure 5.1 - Graph showing L/V ratio for a range of track radii, vs the Nadal Value (100m to 300m)

Each curve on the graph shows how increasing speed affects the L/V ratio for a particular curve radius; the L/V ratio increases as the speed increases and the rate of increase is lower on wider curve radii. The blue dotted line represents the Nadal value and the points where the Nadal value line crosses the other lines indicates the minimum conditions required for wheel-climb derailment to occur, which will serve as a prediction of the approximate speed at which the wheelset should derail in the simulation. Calculating the point where the graphs cross produces the following predictions of the speeds at which the wheelset will derail on each curve radius (shown in Table 5.1, below).

Curve Radius (m)	*100*	*125*	*150*	*175*	*200*	*225*	*250*	*275*	*300*
Speed (m/s)	26.36	29.47	32.28	34.87	37.28	39.54	41.68	43.71	45.65
Speed (mph)	58.96	65.92	72.21	78.00	83.38	88.44	93.23	97.78	102.12

Table 5.1 - Predicted derailment speeds for the test wheelset (100m to 300m)

The derailment speeds for each radius are illustrated in the graph below (Figure 5.2).

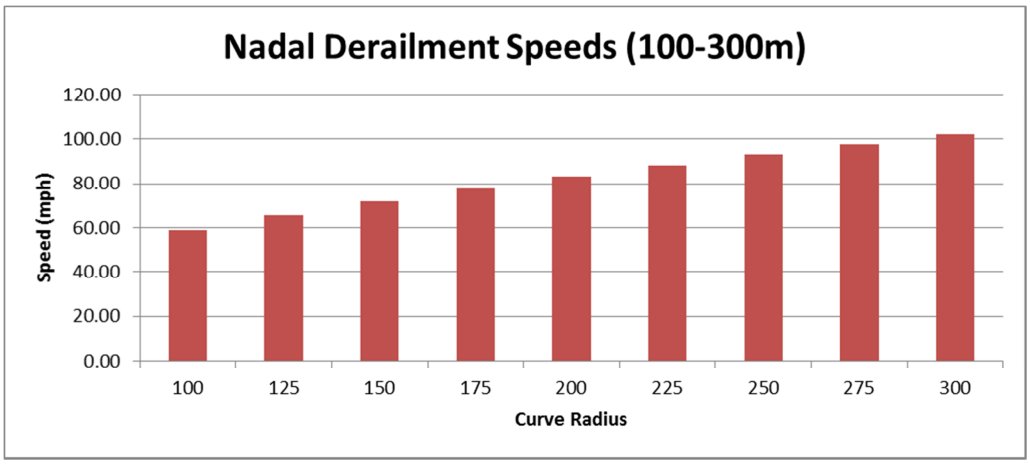


Figure 5.2 - Graph of derailment speeds for a range of curve radii (100m to 300m)

The predicted speed increases with each increase in curve radius, but by a smaller amount each time (the difference between the predicted derailment speeds for radii of 100m and 125m is 9.96mph, whereas the difference between 275m and 300m is 4.34mph). These predictions estimate that the wheelset will derail at just over the vehicle's target speed of 100mph on the 300m radius track, and that the derailment speeds will be higher on higher radii.

Predictions (500 – 1000m)

Because the aim of this research is to test the vehicle at speeds of up to 100mph, it may not be possible to use the Nadal Limit predictions on curve radii wider than 300m. However, the predicted derailment speeds for the higher curve radii are included in Table 5.2, below.

Curve Radius (m)	*500*	*600*	*700*	*800*	*900*	*1000*
Speed (m/s)	58.94	64.56	69.74	74.55	79.07	83.35
Speed (mph)	131.84	144.42	156.00	166.77	176.88	186.45

Table 5.2 - Predicted derailment speeds for the test wheelset (500m to 1,000m)

If the vehicle is capable of achieving sufficiently high speeds (>100mph), then these predictions can be used. Otherwise, tests can still be conducted on these wider curves to make sure that the vehicle does not derail at its intended speed/maximum stable speed.

Summary

While not a truly accurate prediction of the derailment behaviour of the test wheelset, these formulas provide an estimate of the speeds at which it would derail in the real-world, using a formula that is commonly used in the rail industry. These predictions will be used to show that the behaviour of the virtual vehicle is at least logically correct. i.e. Ideally, the wheelset should derail at speeds close to those predicted here, but even if it derails at speeds above or below those predicted by the Nadal Limit, the graphs of the derailment speeds for each curve radius should produce a similar shape to those in Figure 5.2, if the simulation is accurate.

5.5.2 *Nadal Derailment Predictions (Bogie)*

It was also necessary to make predictions about the behaviour of a bogie. As with the wheelset, the Nadal Limit was used to predict the approximate minimum speed at which a bogie would derail on each curve radius.

Lateral Force

Base on the assumption that the lateral forces are even distributed between the two wheelsets, and taking the mass of the bogie into account, the effective mass of the wheelset is calculated using m = the mass of the wheelset (500kg) plus the half the mass of the bogie (6,000kg/2 = 3,000kg) which is 3,500KG. Therefore:

$$L = \frac{3{,}500 * v^{2}}{r}$$

Vertical Force

The vertical force (V) is calculated using acceleration due to gravity (9.806m/s^2) and the effective mass of the wheelset, as above.

$$f = ma \quad \therefore \quad V = 3{,}500 * 9.806 = \mathbf{34{,}321N}$$

Predictions

However, since the lateral and vertical forces have increased by the same amount, this cancels out, meaning that the L/V ratio (and, therefore, the predicted derailment speeds) for the bogie are the same as the derailments speeds for the wheelset.

5.5.3 *Parameter Alteration Predictions*

This section describes the expected results of altering the various physics engine parameters, described in 5.3.1. When the parameters are adjusted, it is estimated that there are four possible outcomes for the changes in results, illustrated by the four graphs in Figure 5.3, below.

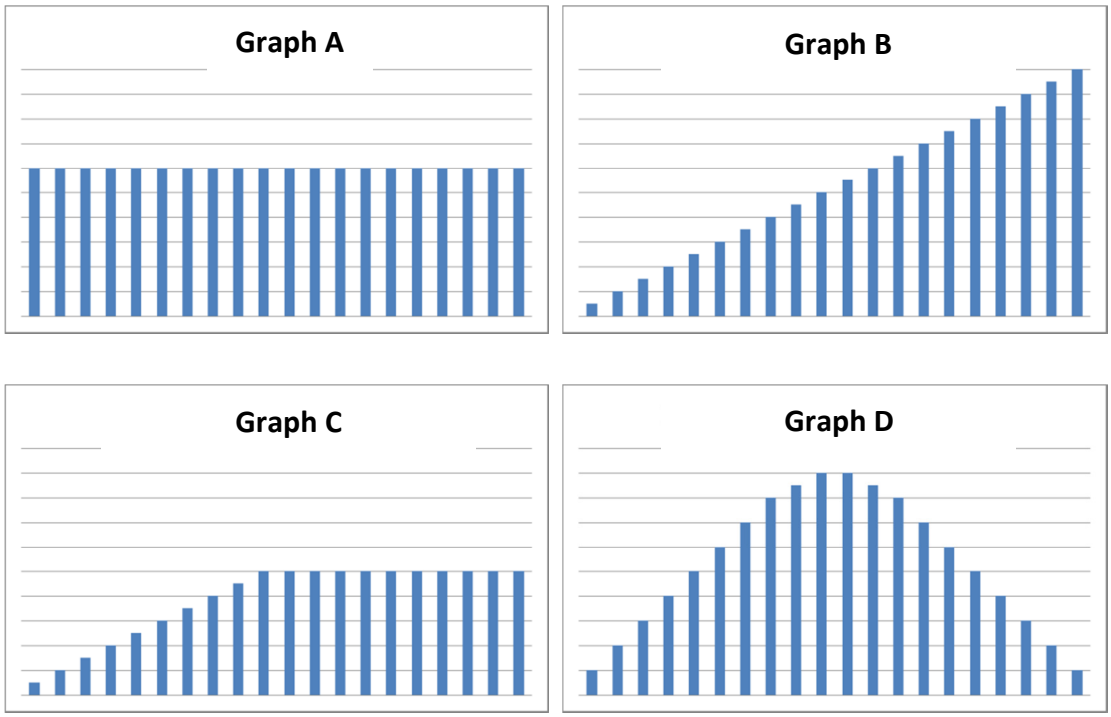


Figure 5.3 - Graphs showing rough predicted outcomes of parameter adjustment testing

These outcomes are as follows:

- *Graph A* - Altering the parameter has little or no effect.
- *Graph B* - Altering the parameter causes the results to improve, and continues to do so across the range of parameter values tested.
- *Graph C* - Altering the parameter has an effect on the results initially, but this change ceases or becomes negligible above a certain value.
- *Graph D* - Altering the parameter improves the results, but above a certain point increasing the parameter has a negative effect on the results.

Predictions Discussion

It is difficult to estimate exactly how adjusting the physics engine parameters will affect the results, but the following predictions were made:

- The simulation is unlikely to achieve the target straight line speed of 100mph, or the predicted curve derailment speeds, using its default parameters. It will be necessary to adjust these parameters and to study what effect they have on the simulation, and whether the simulation can be made to achieve its target speed and produce reasonably realistic derailment results.
- Increasing the value of parameters such as the Simulation Timing parameters and Solver Iteration Count is expected to improve the stability of the vehicle. The results may improve continuously (Graph B) but it is possible that these parameters will reach a value where no further improvement can be made (Graph C) or that above a certain value they will have a negative effect on the stability of the simulation (Graph D). This is hinted at by the PhysX SDK, which describes the Joint Solver Extrapolation Factor with the sentence: *"Setting the value too high can decrease the joint stability"* [41].
- Parameters such as Skin Width and Joint Solver Extrapolation factor are not expected to have any impact on the performance of the simulation tool (Graph A), since they do not alter the fidelity of the solver, just the amount of allowed interpenetration and the flexibility of the rotational joints.

5.5.4 *Summary*

This section has contained predictions about the derailment speed for a wheelset and a bogie on curved track. These predictions, based on the Nadal Limit, are referred to throughout the rest of this thesis. They give a ballpark estimation of the speed at which a single wheelset or bogie should derail, and are used to evaluate the results produced by the Locomotion Simulation Tool.

This section also includes a brief discussion of the possible outcomes of the parameter testing. It is very difficult to predict the precise outcome of these tests, but this section has included a few predictions about the approximate effect of the physics engine parameters on the stability and performance of the simulation tool. More specific predictions for individual test scenarios are included in the relevant sections.

5.6 Testing Phase 1: Static Forces

In these tests, the *PhysX* engine was tested using its default parameters. Initial testing was conducted on simple objects to evaluate the ability of the physics engine to simulate basic rigid body interactions. Tests were also performed on wheelsets and static bogies to determine whether the physics engine was correctly distributing the mass of the object between the wheelsets, and to evaluate the stability of the simulation and the joints between objects. The aim is to verify that the forces acting between objects in PhysX is correct for both smaller mass values and mass values equivalent to those of rail vehicles, before analysing the simulation tool's ability to simulate wheelsets and bogies.

5.6.1 *'SumNormalForce'*

During the callbacks from PhysX, along with the rigid bodies and the contact flags, as described in Section 3.3.7, a parameter called SumNormalForce is returned. This parameter is defined in the PhysX SDK as *'The total contact normal force that was applied for this pair, to maintain non-penetration constraints.'* [41]. If this vector represents the total force applied to the pair, then it is expected to be equal to the normal force acting on the object(s), which is simple to predict and can therefore be evaluated. This force is reported multiple times while the simulation executes, until the wheelset comes to rest, and an average of the reported normal forces measured by the simulation is taken, to see if the results concur with the predicted values.

5.6.2 *Geometric Shape Tests*

This first set of tests focusses on the behaviour of simple geometric shapes. The use of such shapes enables the results to be easily verified mathematically. The aim of these tests is to verify that these simple behaviours are correct, to determine what data is reported by PhysX and to confirm that PhysX is correctly handling the forces acting on the rigid bodies for a range of mass values.

Test Design

A $2m^2$ cubed box is placed in the scene so that it is resting on the ground plane, as illustrated in Figure 5.4, below. The mass of the cube was varied from 1KG to 100,000KG

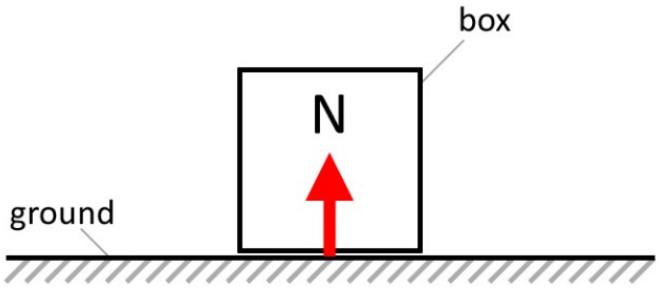


Figure 5.4 - Normal Force between a Box and Ground Plane

Predictions

In these box tests, N should be equal to the gravitational force acting on the box and be perpendicular to the ground plane. Newton's Second Law of motion states that force = mass * acceleration (*F = ma)*. a = acceleration due to Gravity (a vector): [0, 0, -9.806]. So, if the mass (m) of the box is 1kg, then N should be [0, 0, (+) 9.806] and as the mass increases, the normal force should increase accordingly. The predictions for all of the mass values tested are included, along with the results, in Table 5.3, below.

Results

Table 5.3, below, shows the average *sumNormalForce* results for each mass value tested. Include are the predicted normal force ('N'), the average (mean) magnitude of the recorded normal force, the difference between the prediction and the recorded value and the range of magnitudes.

- In each test, the Normal force was reported 23 times before the box became 'at rest'

Mass	Prediction (N)	Average Magnitude	Difference	Range
1KG	9.806	9.80596	0.00004	0.002
10KG	98.06	98.05952	0.00048	0.018
100KG	980.6	980.59535	0.00465	0.177
1,000KG	9806	9805.95287	0.04713	1.77
10,000KG	98060	98059.53948	0.46052	17.703
100,000 KG	980600	980595.39696	4.60304	177.063

Table 5.3 - Box/Ground Test Results (1KG to 100,000KG)

In each test, the results were, on average, just 0.00047% lower than the predicted results. The difference between the prediction and the results, as well as the range of results, increases relative to the mass of the box (i.e. increasing the mass by a factor of 10 causes the range and difference to increase by a factor of 10). The range of results is always 0.18% of the mass of the object. The standard deviation of the magnitudes is, on average, 0.007% of the mass of the box across all tests.

- *Consistency Tests*

In order to evaluate the consistency of the physics simulation, each of the previous tests was repeated five times and the results from each test were compared. Table 5.4 (overleaf), for example, shows the five results from the 10,000KG tests. The results were identical each time, as were the results for the other mass values.

Test	Mass	Average Magnitude	Difference	Range
1	10,000KG	98059.53948	0.46052	17.703
2	10,000KG	98059.53948	0.46052	17.703
3	10,000KG	98059.53948	0.46052	17.703
4	10,000KG	98059.53948	0.46052	17.703
5	10,000KG	98059.53948	0.46052	17.703

Table 5.4 - Box/Ground Test Results (10,000KG x 5)

Initial Observations

A real-time system was always going to introduce error. The error here is less than 0.001% and, the range of results is, on average, less than 0.2% of the mass of the object (and the standard deviation is less than 0.01% of the mass). This is believed to be an error bound that is sufficiently small as to be acceptable to engineers in a real-time tool. Additionally, data from multiple tests shows that the simulation is capable of producing data that is consistent.

5.6.3 *Wheelset Tests*

In this section, the normal force between the wheels and rails in the simulation is studied using a static wheelset. Both designs of wheelset (MB and SB Wheelsets) are tested to verify that the forces are being applied correctly and to see if either design produces more realistic and/or more stable results.

Test Design

In these tests, the wheelset is placed onto a single (10m) section of straight track. The Normal force acting on the wheelset is then measured. The range and standard deviation of the values is also studied as a measure of the stability and consistency of the simulation. Figure 5.5, below, shows a screenshot captured during one of these tests.

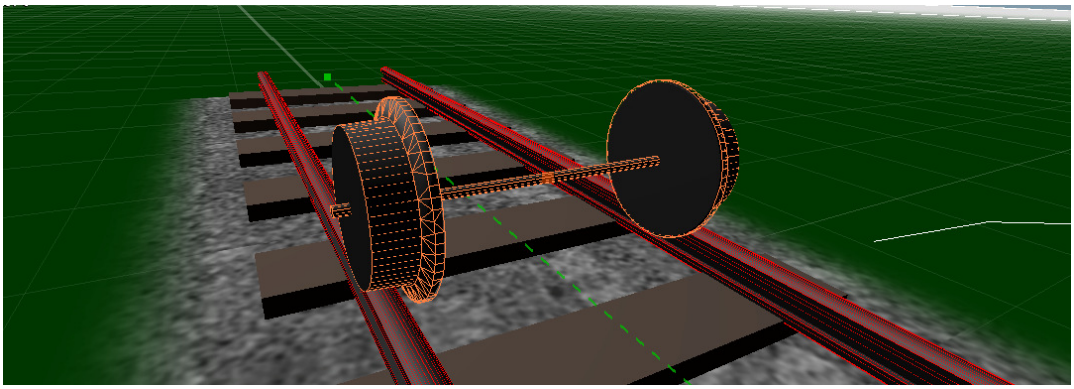


Figure 5.5 - Screenshot from Static Wheelset Testing

➢ *Properties*

All PhysX parameters and properties are set to their defaults, with the exception of the dynamic and static friction values of the wheels and rails, which have been set to 0.74 and 0.57, respectively (the coefficient of friction of steel-on-steel - as described in Section 3.4.4). The wheelset was constructed using the properties described in Table 3.3, i.e. a mass of 500kg.[10]

Predictions

If the wheelset is placed exactly between the rails, it is assumed that there will be even distribution of mass between the two wheels. The rails are placed at the origin of the scene, either side of the y axis. A diagram of the wheels and their normal forces (relative to the x axis) is shown in Figure 5.6, below (wheel conicity and the angle of the normal forces (red arrows) have been exaggerated).

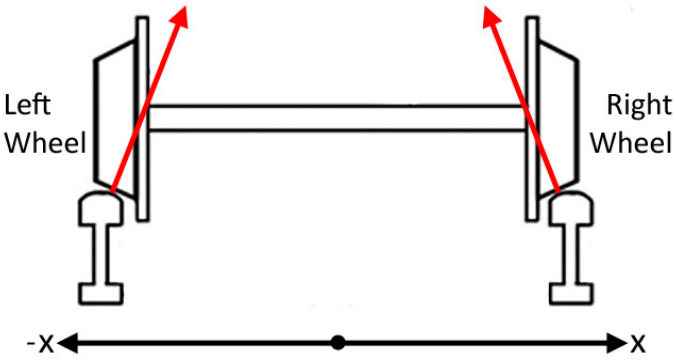


Figure 5.6 - Wheel Normal Forces (red arrows), relative to the X Axis of the scene

The equation for the magnitude of a normal force on an inclined plane is $N = m\ g\ \cos\ (\delta)$, where m is the mass of the object, g is acceleration due to gravity and δ is the angle of the plane.

➢ *Multi Body Wheelset*

Because the Multi-body Wheelset is made up of separate rigid bodies and the callbacks from PhysX that report the normal forces are per rigid body, it was expected that there will be one callback per wheel, reporting the normal force acting on that wheel (the red arrows in the diagram above).

[10] - Results from tests with other mass values were consistent with those described in this section.

So, given that the wheel contact angle at rest (2.86°) is equivalent to the angle of an inclined plane, and assuming even mass distribution of the 500kg mass of the wheelset between the two wheels, the magnitude of the normal force should be:

$$N = \left(\frac{500}{2}\right) * 9.806 * \cos(2.86) = \mathbf{2,448.45\ N}$$

The (normalised) Normal to the flat ground plane is the vector: [0, 0, 1]. Because the rails run in the direction of the y axis, this normal can be rotated about the y axis by +2.86° to find the direction of the normal force for the left wheel and -2.86° for the right wheel. This produces the 'Normal (Normalised)' vectors in the table below. This direction vector can then be multiplied by the predicted magnitude of N (above) to produce the 'Prediction' vectors in Table 5.5, below.

	Normal (Normalized)			**Prediction**		
Wheel	**x**	**y**	**z**	**x**	**y**	**z**
Left	0.0499	0.0	0.9988	122.167	0.0	2445.4
Right	-0.0499	0.0	0.9988	-122.167	0.0	2445.4

Table 5.5 - Predicted Normal Forces for an MB Wheelset on Rails

➢ *Single Body Wheelset*

Because the Single Body is constructed from a single rigid body, and the normal force reported by PhysX is a per-rigid-body force, it is expected that the magnitude of *sumNormalForce* should be one of two values:

- 'Prediction A' - The first possibility is that PhysX detects the contact points between each wheel shape and each rail shape separately, and applies the forces at each contact point of the wheelset (in the same way that forces are applied to the individual rigid bodies of the Multi-body Wheelset). The reported force would then be the sum of those forces, and should be equal to the sum of the two vectors in the Multi-body Wheelset prediction above; a vector: [0.00, 0.00, 4890.80].
- 'Prediction B' - Alternatively, if the wheelset is treated as a single rigid body, then the force could be an upward force, with the magnitude calculated based on the mass of the wheelset as a whole, multiplied by acceleration due to gravity: $500 * 9.806 = \mathbf{4,903N}$ and should be pointing directly upwards: [0.00, 0.00, 4,903.00].

The results from the simulation should be close to one of these values, and will give some insight into how the forces are calculated and applied by the physics engine.

Results

The results from the tests conducted on both wheelset types are shown below. Each test was repeated three times to confirm that the results were consistent.

➢ *Multi Body Wheelset*

Below (in Table 5.6) are the average x, y and z components of the force acting on the wheels of the Multi-body Wheelset, as reported by PhysX. Also included are: the average magnitude, the difference between the prediction and the results, and the range of results.

- 115 results were collected per wheel during these tests.

	Force					
Wheel	**X**	**Y**	**Z**	**Magnitude**	**Difference**	**Range**
Left	122.430	-0.018	2451.128	2454.184	-5.74	1.57
Right	-122.268	-0.023	2449.899	2452.948	-4.50	1.75

Table 5.6 - Wheelset forces (Wheelset/Rails)

These results are very promising. The x, y and z components of the force, as well as its magnitude, were close to the predicted values, with the magnitude of the force being, on average, 0.2% above the prediction. The range of magnitudes is also very small, only 1.57/1.75N; less than 1,000th of a percent of the magnitude of the force. There is a small difference of 0.02N between the two wheels. One possible reason for this could be that the wheelset did not come to rest exactly at the midpoint between the rails, possibly as a result of error in the way the solver handles the collisions between the objects in the simulation. Or it could be that a floating point error, penetration error or other error introduced by the way that the solver handles the joints between the rigid bodies.

➢ *Single Body Wheelset*

The results in the table below are from the SB Wheelset test. Table 5.7 shows the average force vector, its magnitude, the difference between that magnitude and the second prediction (which was the nearest) and the range of magnitudes.

- 59 results were collected in total

Force				
X	**Y**	**Z**	**Magnitude**	**Range**
1.076	0.004	4913.424	4913.425	199.66

Table 5.7 - Single Body Wheelset/Rails Results

Table 5.8, below, shows the difference between the two predictions and the results:

Prediction	Predicted Value	Magnitude	Difference
Prediction A	4890.80	4913.425	-22.625
Prediction B	4,903.00	4913.425	-10.43

Table 5.8 - Single Body Wheelset/Rails Results

This data suggests that the forces are being applied to the rigid bodies as a whole, since the magnitude of the observed forces is closest to that in Prediction B.

The average force is 10.43 above the prediction. The reason for this discrepancy could be some initial differences in the size of the force, as demonstrated in the graph below (Figure 5.7), which shows the magnitude of the normal force during the first 50 samples.

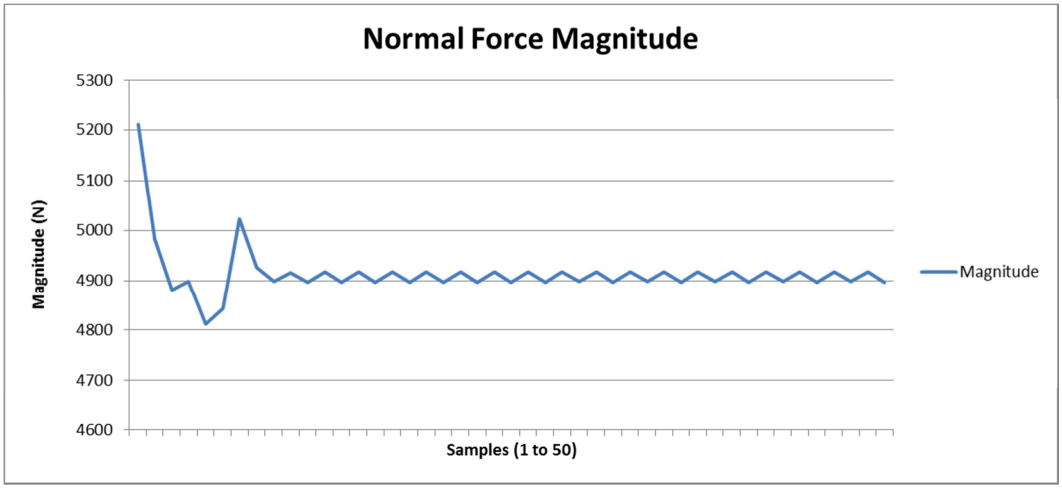


Figure 5.7 - Magnitude of the Normal Force (Single Body Wheelset - First 50 Samples)

The initial magnitude is much higher than the average value. After the first 10 samples, the result oscillated but is closer to the predicted value. The standard deviation of the results is 50.61N, meaning that the majority of the results are within 1.03% of the mean value. Initial 'peak' value is likely to be responsible for the discrepancy between the average result and the prediction, as well as the high range of results.

The magnitude fluctuates over the first 10 samples, which is likely to be the result of penetration error or other issues internal to the physics engine as the initial contact is resolved. Different amounts of force are applied in order to correct for penetration error and/or for the amount of applied force being too large or too small in a previous iteration. This suggests some instability, but PhysX does eventually resolve this and declares the object to be at rest. These initial values may also explain the range of results.

5.6.4 *Bogie Tests*

The tests in this section are designed to evaluate whether the mass of the bogie is being correctly distributed between the wheelsets.

Test Design

A bogie is placed on a section of straight track and allowed to come to rest, as illustrated in the screenshot below (Figure 5.8). The forces reported by PhysX until the bogie comes to rest are recorded, and an average is taken.

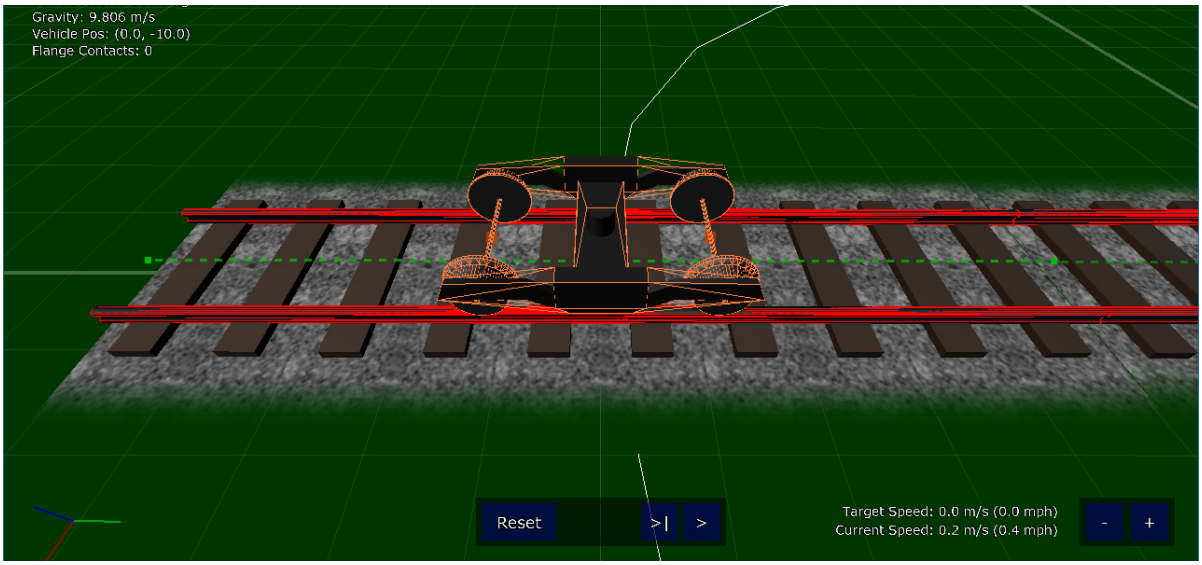


Figure 5.8 - Screenshot from Static Bogie Testing

Multi Body Wheelset

The following tests were conducted using a bogie with multi-body wheelsets attached.

➢ *Predictions*

The effective mass of each wheelset is calculated as follows (based on the assumption that the 6 tonne mass of the bogie is evenly distributed between the two wheelsets, and then between the two wheels):

$$Mass = \frac{\left(\frac{6{,}000}{2} + 500\right)}{2} = 1{,}750kg$$

The normal force acting on the wheelset can then be calculated as follows:

$$N = 1{,}750 * 9.806 * \cos(2.86) = \mathbf{17{,}139.1\ N}$$

The prediction in Table 5.9 (below) is calculated using the normalised Normal force direction vector from earlier (Table 5.5) multiplied by the predicted magnitude above.

	Normal (x Magnitude)		
Wheel	**x**	**y**	**z**
Left	855.169	0.0	17,117.791
Right	-855.169	0.0	17,117.791

Table 5.9 - Predicted Normal Forces for a Multi-body Wheelset of a Bogie on Rails

➢ *Results*

Below (in Table 5.10) is the data each of the four wheels. The table contains the average x, y and z components of the force, along with the average magnitude, the difference between the prediction and recorded magnitude, and the range of magnitudes.

- 111 results were recorded per wheel.

	Force					
Wheel	**X**	**Y**	**Z**	**Magnitude**	**Difference**	**Range**
Front-Left	858.224	-2.192	17179.511	17200.935	-60.385	20175.194
Front-Right	-856.044	-2.402	17150.365	17171.716	-28.805	23469.879
Rear-Left	857.735	-2.207	17172.968	17194.376	-54.011	20313.381
Rear-Right	-856.252	-2.410	17148.318	17169.682	-27.774	23608.398

Table 5.10 - Multi-body Wheelset forces (Bogie/Ground)

There is a difference of 31.3N between the highest and lowest average magnitude, but on average the magnitude of the force was just 0.249% higher than the predictions. The standard deviation of the results is 2303.59, which is quite high, 13.46% of the average magnitude. The range of magnitudes is high because of 'spikes' in the results, as illustrated in the graph below (Figure 5.9), which shows the normal force acting on the left and right wheels of the front wheelset of the bogie.

These discrepancies are likely to be the result of penetration errors. A higher than necessary force is applied when the penetration error occurs, resulting in a lower than average result in the following frame, and then the results stabilise. The results also correlate directly with an opposite spike in the opposite wheel's results, as illustrated in Figure 5.9 (overleaf). The impact on stability caused by these discrepancies is minimal however, since each of these callbacks represents one iteration of the physics engine and there are eight iterations by default for each frame (call to simulate), which happens 60 times per second (so the data represents 13 frames, a fraction of a second).

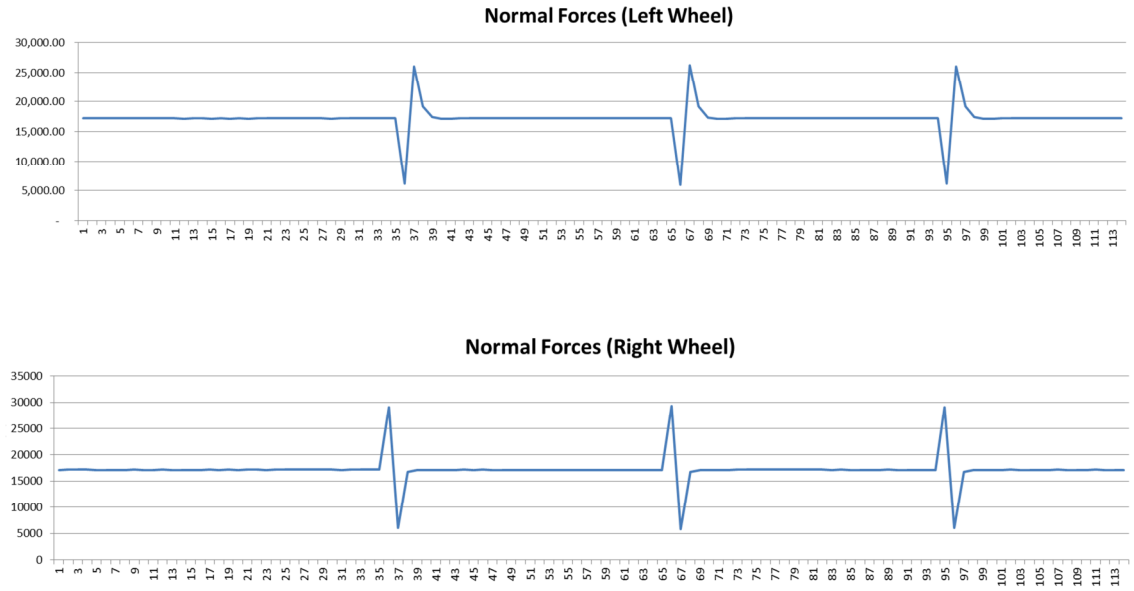


Figure 5.9 - Normal Force acting on the Front, Right Wheel

Single Body Wheelset

Next, the tests from the previous section were repeated with a bogie that had a pair of SB Wheelsets attached to it.

➢ *Predictions*

Because the SB Wheelset consists of a single rigid body, and based on the test results in Section 5.6.3, the expectation is that the force should be directly upward and not distributed between the wheels, with a magnitude of **34,321N.**

➢ *Results*

The table below shows results of the Single Body bogie testing.

- 43 results were recorded per wheelset.

	Force					
Wheelset	**X**	**Y**	**Z**	**Magnitude**	**Diff**	**Range**
Front	-33.736	-7.152	31814.876	31819.673	2501.327	1673.361
Rear	53.453	-11.783	31803.431	31808.262	2512.738	1726.429

Table 5.11 - Single Body Wheelset Forces (Bogie/Ground)

There were just 43 results instead of the 59 recorded in the previous Single Body Wheelset tests. There are also fewer results and a lower range of values than the equivalent Multi-body Wheelset results, suggesting improved stability.

In these tests, the magnitude of the normal force acting on the SB Wheelset is lower than the expected values. However, if the effective mass of the wheelset is calculated differently, it is possible to produce these results. If, instead of calculating the mass as

$$\frac{Bogie\ Mass}{2} + Wheelset\ Mass$$

... it is calculated as:

$$\frac{Bogie\ Mass + Wheelset\ Mass}{2}$$

Then the predicted magnitude of the normal force is 31,869.5m and the results are then, on average, only 0.174% above the prediction. It is unclear why this behaviour is different for the SB Wheelset than it is for the MB one.

Once again, there is a high range of magnitudes, but a closer inspection reveals the following graph (Figure 5.10):

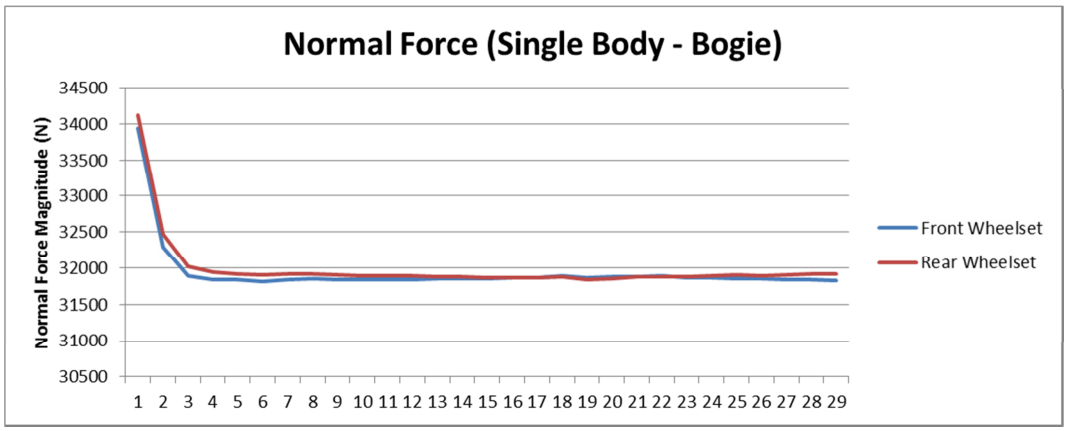


Figure 5.10 - Normal Force acting on the Front and Rear Wheelset (Single Body Bogie)

There is initial variance in the magnitude of the force, presumably as the solver resolves the initial contact/penetration issues and the joints between the wheelsets and the bogie. After that, the results are more stable and consistent. The standard deviation of the results was 441.42N for the front wheelset and 477.02N for the rear wheelset, so the results were, on average, within 1.4% of the average.

Observations

In the Multi Body tests, the bogie rolled backward and forward for a few seconds before it stopped moving. In the single body tests, the bogie did not appear to move at all. Results are identical if the test is repeated.

This data suggests that the forces being applied to it are correct, within an error bound of approximately 1%, caused by initial instability while the contact is resolved. The data also suggests that, as expected, the SB Wheelset is more stable, as it generated fewer results and a smaller range/standard deviation of results.

5.6.5 *Summary*

This section has shown that the Physics Engine is capable of producing promising results for simple objects and the wheel/rail interface when tested with a static wheelset/bogie.

Box Tests

In the box testing, there is a discrepancy of only 0.006% on average between the expected normal forces and those reported by PhysX across all mass values, including values up to and including 100 tonnes. The range of results is always 0.18% of the mass of the box. This consistency is promising for engineering use.

Wheelset Testing

The results for a wheelset at rest are also promising. The reported normal forces for the Multi-body Wheelset on were off by less than 0.25%. There are some irregularities between the wheels, which is believed to be the result of the wheelset not coming to rest exactly between the rails, or minor errors in the way that the solver handles collisions and joints between the objects. The data from the Single Body Wheelset also suggests that the normal forces are being correctly applied. The number of results collected, which was lower in SB testing than in MB testing, also suggests that the SB wheelset is more stable.

Bogie Testing

During bogie testing there were some minor discrepancies between the wheelsets and high range of range of results, which are believed to be caused by the way in which the PhysX solver handles the collisions and joints between rigid bodies. However, it is also believed that the impact of these discrepancies on the simulation is minimal, as the simulation compensates for these erroneous results over the following frames.

Performance

SB tests ran at an average of 221.7 frames per second and MB tests at 118.9 FPS, so can be said to be running in real-time. The MB wheelset, as expected, has a lower performance than the SB Wheelset due to the additional rigid bodies.

Conclusions

The results are promising. The consistency and the error bound of the results are good, and may improve further with adjustments to the Physics Engine parameters.

5.7 Testing Phase 2: Wheelset in Motion (Straight Track)

The tests in this section were conducted using wheelsets in motion, in order to conduct an initial evaluation of the simulation of the wheel/rail interface in the Locomotion tool.

5.7.1 *Initial Cone Wheelset Testing*

The first set of tests involved a conical wheelset in motion on a section of straight track.

Test Design

These tests are conducted using the conical wheelset and the default parameters of the physics engine. The conical wheelset was tested on a 100m straight section of track at a range of speeds. The use of the conical wheelset allows free lateral movement of the wheelset (unrestricted by flanges). These tests were intended to evaluate the behaviour of the wheelset and to look for indications of the self-centring mechanism or hunting oscillation behaviours. A screenshot from one of the conical wheelset tests is shown in Figure 5.11, below.

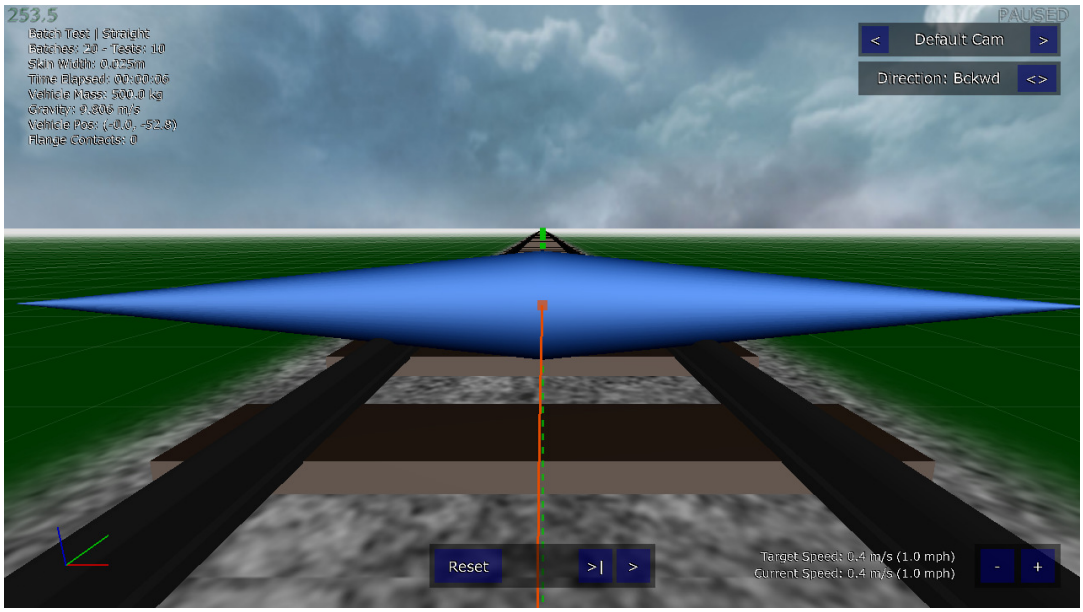


Figure 5.11 - A screenshot of Cone Wheelset Testing in the Locomotion Tool

The screenshot shows the conical wheelset (blue) travelling along the rails. It also shows the path of the wheelset (orange) and the centre spline of the track (green dotted line).

Predictions

Since the track is straight, the average lateral offset of the wheelset should be zero. It is expected that there will be lateral offset in a sinusoidal motion as the wheelset moves from side to side and the effective radius of the wheel at the point of contact with the rails changes (as illustrated in Figure 2.12). It is possible for hunting oscillation to occur.

Results

The results in this section represent a sample of the tests performed with the conical wheelset.

- *Initial Testing*

Figure 5.12 shows a test that produced clear visual results; the path of the wheelset (orange) shows that the wheelset experienced lateral movement in a sinusoidal motion.

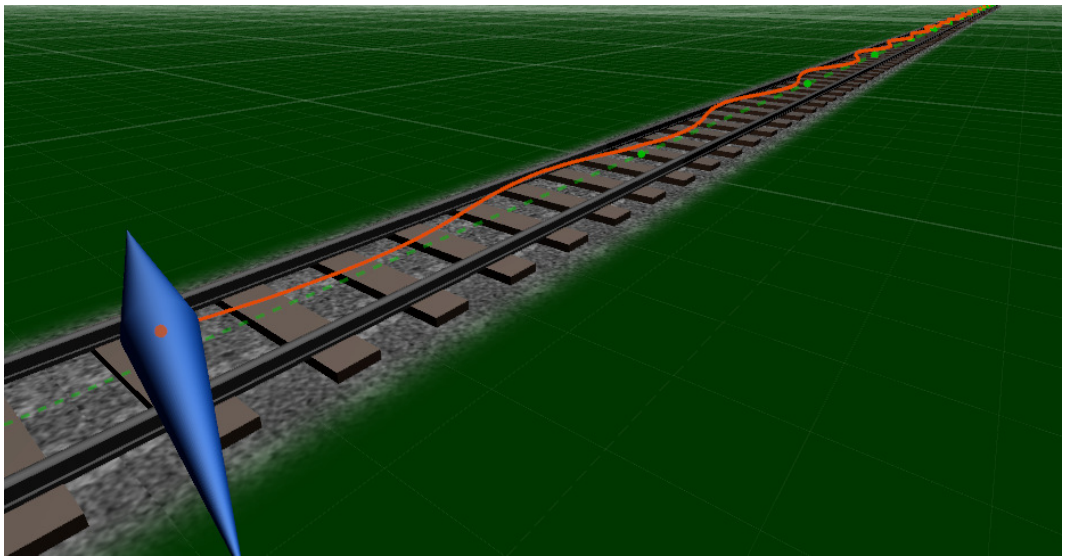


Figure 5.12 - A cone wheelset test showing hunting oscillation (orange line)

The graph in Figure 5.13 (below) shows the path of the wheelset in tests on a 100m straight track at 1mph. The lateral offset was recorded multiple times per second throughout the test.

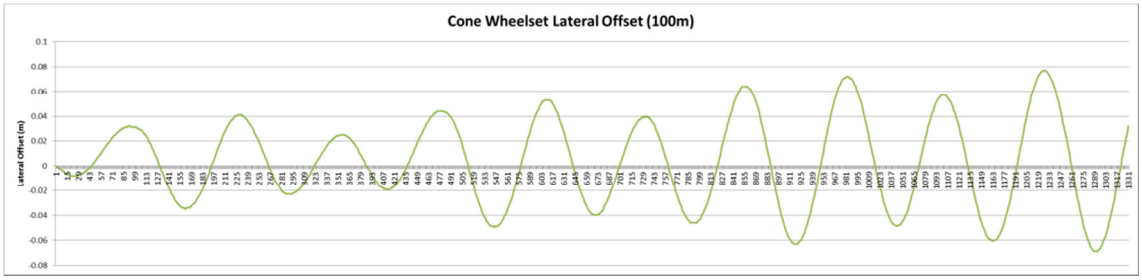


Figure 5.13 - Sinusoidal motion of a wheelset along a 100m straight track

The graph shows that the motion of the wheelset is sinusoidal, as the wheelset moves from side-to-side, as was predicted. The graph also shows that the amount of offset in each direction appears to increase over time. This would seem to suggest that the centring action of the conical wheelset is occurring to some degree in the physics engine, and that it may also be capable of simulating the effects of hunting oscillation.

➢ *Multiple Tests*

The test was repeated 10 times, under the same conditions, to study any variance in the results. The graph below (Figure 5.14) shows the average (blue), maximum (left - green) and minimum (right - red) lateral offset in each of the 10 tests (at 1mph).

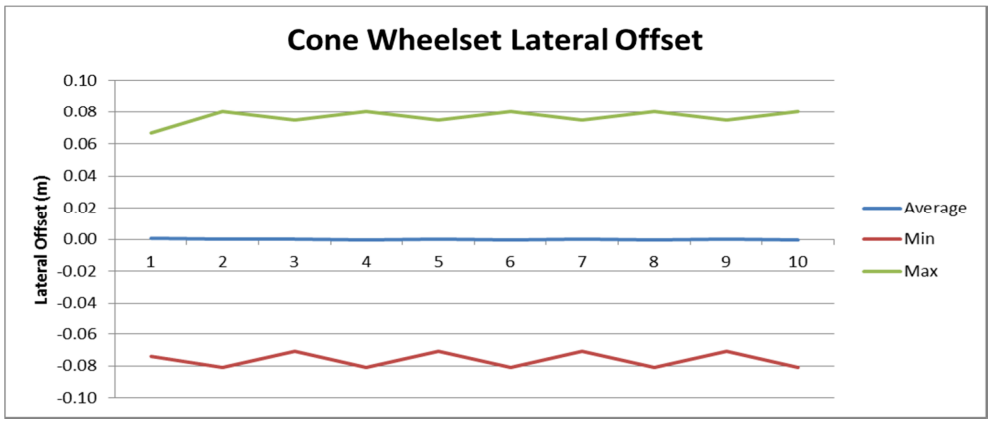


Figure 5.14 - Average, Min and Max Lateral Offset for the Cone Wheelset (1mph, 100m Straight)

As the graph demonstrates, the average lateral offset is close to 0.0m and the left and right lateral movement is roughly symmetrical, as expected. There is some variance in maximum offset between 0.07m and 0.081 meters in each direction. The table below (Table 5.12) shows the results, as well as the range of results, for the average, minimum and maximum offset across all 10 tests.

	Average	Maximum (Left)	Minimum (Right)
Average Offset	0.001	0.076	-0.076
Range of Results	0.002	0.013	0.010

Table 5.12 - Average and Range of Results for the Cone Wheelset Offset Tests

The average offset was 1mm to the left, just 1mm away from the predicted result. The average offset is also symmetrical for left and right moment of the wheelset. The range of results for the max and min offsets was approximately 0.01 meters (1 centimetre), though the range of results for movement to the left is slightly higher (0.003m or 3mm larger than the movement to the right).

If the path of the wheelset during all 10 tests is all plotted on the same graph, it produces the results shown in Figure 5.15, below.

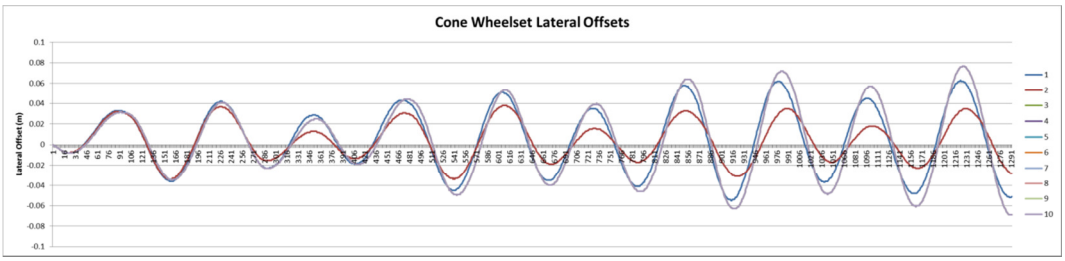


Figure 5.15 - Graph showing the path of the wheelset over 100m (single body wheelset at 1mph)

This graph shows just three distinct paths taken by the wheelset across the 10 tests, suggesting that, while there is some variance in the results, there is also some consistency.

5.7.2 *Initial Testing at Higher Speeds (Cone Wheelset)*

The next set of tests was conducted at a range of speeds from 1mph to 10mph.

Initial Testing

During these initial tests at high speeds, the wheel was unable to accelerate above approximately 2.5 mph. In all tests where the target speed was set to 3mph or above, the peak speed was between 2.50 and 2.64 mph.

Max Angular Velocity

After some investigation, this limited top speed was found to be the result of the wheelset's Maximum Angular Velocity (MAV) parameter. This parameter is described in Section 3.4.7. In order to try and determine what exactly this value represents and how it affects the speed of the wheelset, the following tests were conducted. The MAV value was increase and the peak speed of the wheelset during each test was recorded. The graph below (Figure 5.16) shows the results.

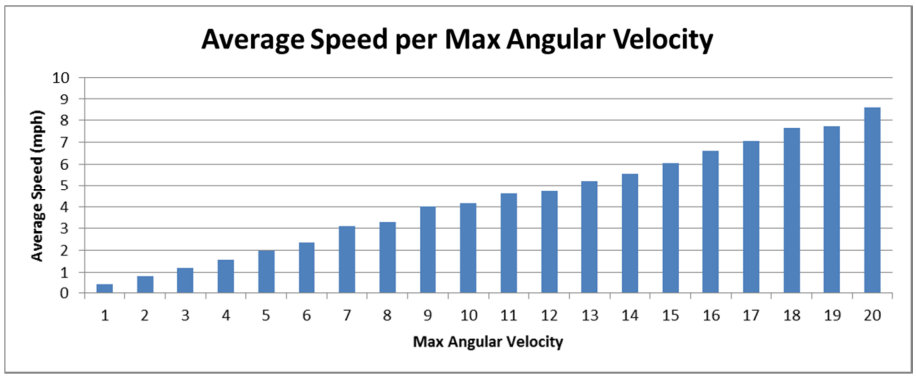


Figure 5.16 - Graph of Average Speed per Angular Velocity (Conical Wheelset)

As the graph shows, the speed of the wheelset increased as the MAV parameter was increased. The ratio of angular velocity to speed in these tests was, on average, 2.44. This suggests that, in order to allow the conical wheelset to travel at a speed of 100mph, the MAV should be 244, assuming that this trend continues.

The graph below (Fig. 5.17) shows the number of derailments in each test at each speed.

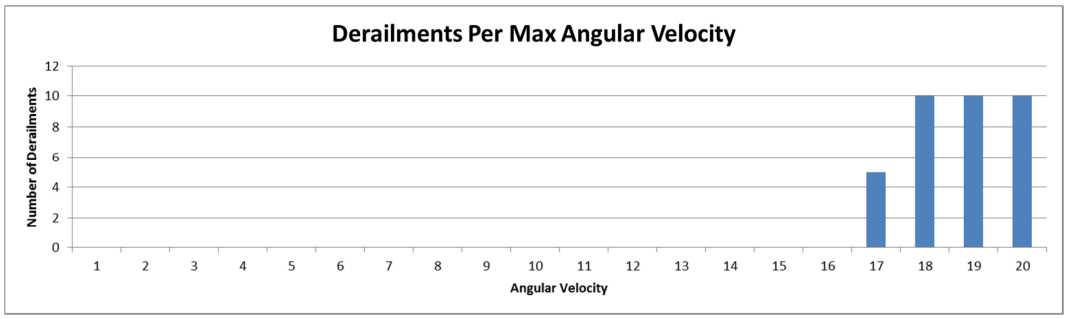


Figure 5.17 - Derailments per Angular Velocity (Conical Wheelset)

These results show that increasing the angular velocity increases the top speed of the wheelset. However, the wheelset derails in all tests above a certain speed (approximately 8mph), and so further testing of this parameters is not possible at this stage.

Further MAV Testing is conducted with the SB Wheelset in Section 5.7.4, in order to determine a suitable value for use with the actual train wheelsets (which have a larger radius at the point of contact with the rails, and so will travel forwards faster if the angular velocity is the same).

5.7.3 *A range of Speeds (Conical Wheelset)*

After the maximum angular velocity of the wheelset was increased to 20, the conical wheelset was tested again at a range of speeds.

Test Design

The conical wheelset was tested at a range of speeds on a 100m straight track.

Predictions

The average offset should be zero, the min and max offset should be symmetrical, as with the tests in the previous section. It is also expected that the average left and right offsets should increase with the forward velocity of the wheelset.

Results

The following graph (Figure 5.18 - overleaf) shows the average lateral offset (blue) as well as the maximum offset (left of the track - green) and minimum offset (right of the track centre - red).

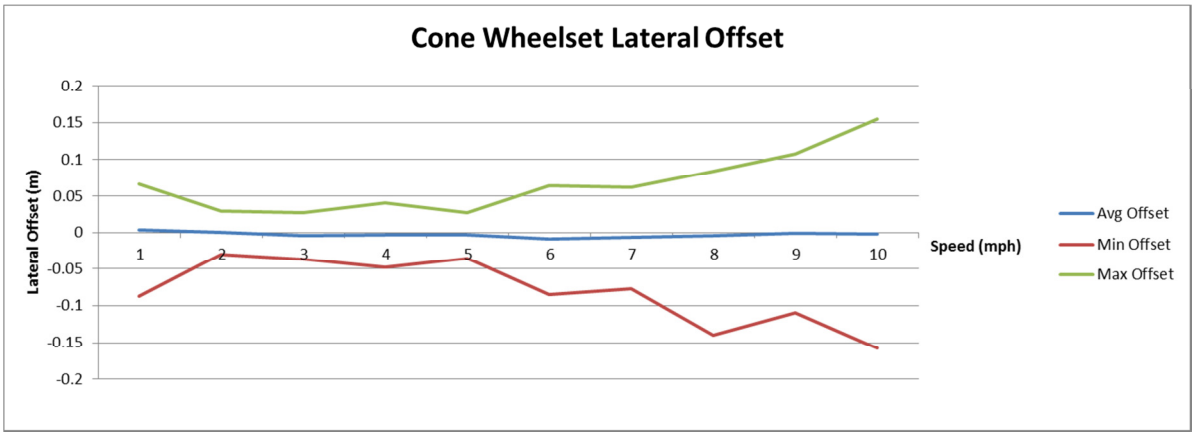


Figure 5.18 - Graph showing the change in the offset of the single body wheelset at a range of speeds

The average offset is close to 0.0m, as expected. The range of results is less than 1mm. The minimum and maximum lateral offsets increase with speed (as expected), with the exception of the 1mph tests, which are higher than the 2mph tests. This seems to suggest that the lateral motion is occurring in a logically correct way, but that the vehicle may be less stable at very low speeds.

The results are less consistent at higher speeds (9/10mph), as indicated by the min offset (red line), but this this is likely to be due to the wheelset derailing, which it did in the 8mph, 9mph and 10mph tests (but not at 7mph).

Performance

The average framerate during these tests was 205.8fps at all speeds.

Summary

The average lateral offset in all tests was close to zero, with the largest difference being just 1mm away. The motion is sinusoidal, as expected, though less regular and possibly larger than expected. There is some variance in the results between each individual test, but the results in the example in Figure 5.14, for example, are all within a relatively small range. When ten identical tests are conducted, the range of average results is approximately 2mm and the range for the min and max values is 1cm. the wheelset took just three distinct paths along the straight track. It is possible that adjusting the parameters of the physics engine will produce more consistent results. It is also possible that this is the result of error in the physics engine and that some variance is inevitable. This could suggest that the simulation is not suitable for running individual tests, but could be used to run multiple tests, produce averages and calculate the range/standard deviation of the results.

All future tests are conducted using at least 10 tests per batch to evaluate the range and standard deviation across the range of tests and so to evaluate the consistency of the simulation tool, as well as the derailment speed, lateral offset and other results. These results will also show if the average results for each batch of tests are consistent and correlate with predictions, even if the range of results is high.

While the results are logically correct, without being provided with the correct data or formulas to use to evaluate the results of these tests it is difficult to discuss whether these results are truly accurate. If such data could be obtained, further testing could be done using this wheelset.

Testing in the next section will focus on testing a single wheelset on a straight track and on using the Nadal Limit to evaluate the derailment speed of a wheelset during cornering.

5.7.4 *SB and MB Wheelsets - Initial Results*

The next phase of testing involved an initial evaluation of the Single Body and Multi-Body Wheelsets, using the default parameters of the physics engine.

Angular Velocity Testing

The following tests were conducted to verify the results of the angular velocity testing conducted on the Conical Wheelset in the previous section. The aim of these tests is to determine a suitable value for Max Angular Velocity (MAV) (until further testing can be conducted). These tests need to be re-run because the single and multi-body wheelsets have a different radius to the conical wheelset at the point of contact with the rails, and the forward velocity of the wheelset will be different as it is related to the angular velocity and radius of the wheels. The graph below (Fig. 5.19) shows the results from tests conducted on a 100m straight with the SB wheelset, with a range of MAV from 1 to 20.

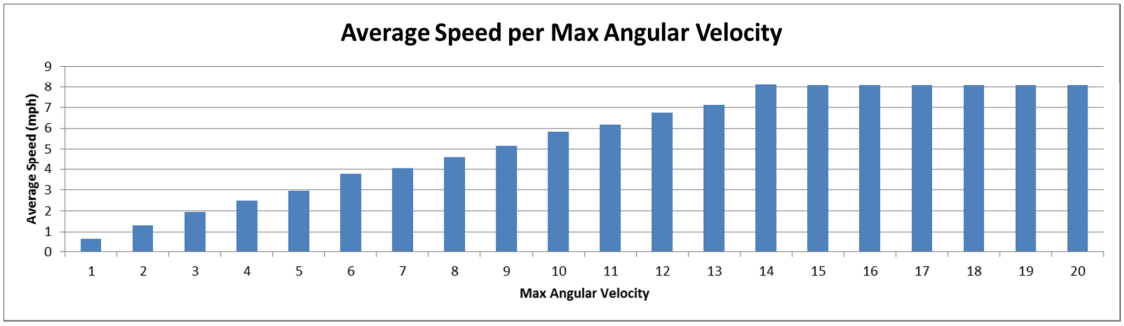


Figure 5.19 - Average Speed per Max Angular Velocity

The results for an angular velocity below 14 show a similar trend to those in the previous conical wheelset tests. Above 14, however, the results no longer increase. In these tests, the wheelset derailed in every test at all speeds.

The results for MAV values of 14 and above suggest that the wheelset is not capable of travelling above these speeds without derailing. The graph below (Fig. 5.20) shows the number of derailments in each batch of tests.

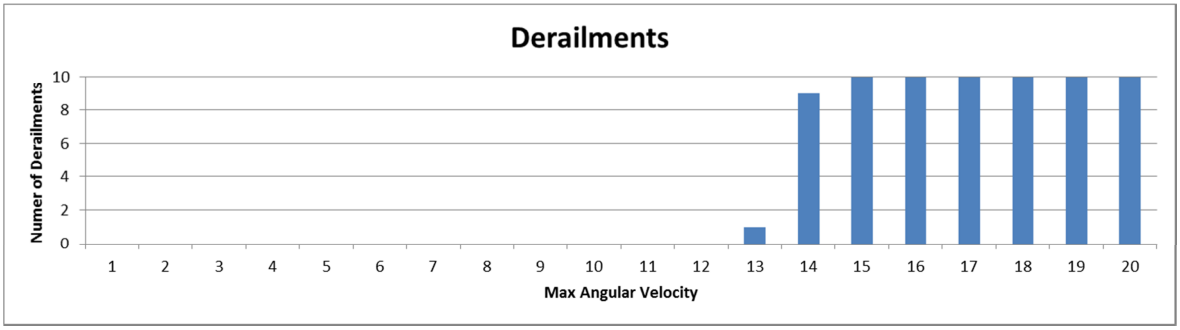


Figure 5.20 - Number of Derailments per Max Angular Velocity value

The wheelset derailed once with an MAV value of 13, in 9/10 tests at an MAV of 14 and in all tests at an MAV of 15 and above, which corresponds with the peak speeds achieved in Figure 5.19.

The SB Wheelset achieved a higher speed at each Angular Velocity value than the conical wheelset, because the SB Wheelset has a higher radius at the point of contact with the rails. It is rotating at the same speed, but because the radius is larger, it is travelling forward faster. The ratio of angular velocity to peak speed is approximately 1.68 (measured as an average across the results from MAV of 1 to 14). MAV has therefore been set to 170 to enable the wheelset to accelerate to over 100mph.

➢ *Performance*

It was not expected that changing Max Angular Velocity would have any effect on the performance of the simulation tool, but to confirm this, the performance during the previous tests was measured and is shown in the graph in Figure 5.21, below.

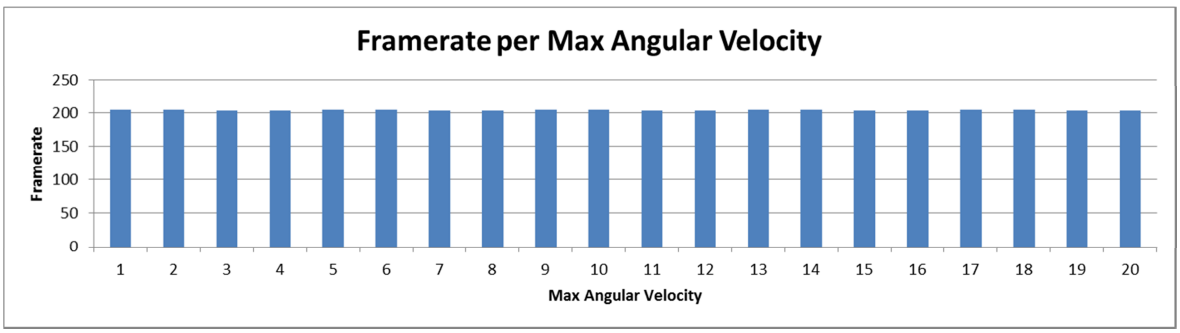


Figure 5.21 - Performance per Max Angular Velocity (Single Body Wheelset)

The performance varies between a maximum framerate of 205.1 FPS and a minimum framerate of 202.9 FPS but not in any consistent way that correlated with the changes in MAV. This data supports the hypothesis that altering MAV does not impact the performance of the simulation.

Target Speed Testing

With Max Angular Velocity increased, the next step was to evaluate the wheelset's stability at a range of speeds. Table 5.13, below, shows the number of derailments per target speed for the Single Body Wheelset on the 100m straight track.

Speed	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Derailments	10	10	10	10	10	10	10	10	10	10

Table 5.13 - Derailments per Target Speed (Initial Single Body Wheelset Testing)

There were derailments in every test conducted at each target speed value. Further investigation revealed that these derailments were being caused either by the inner wheel coming into contact with the ground plane or because part of the wheelset was penetrating the rails and becoming 'stuck', as shown in the screenshots below (Fig. 5.22).

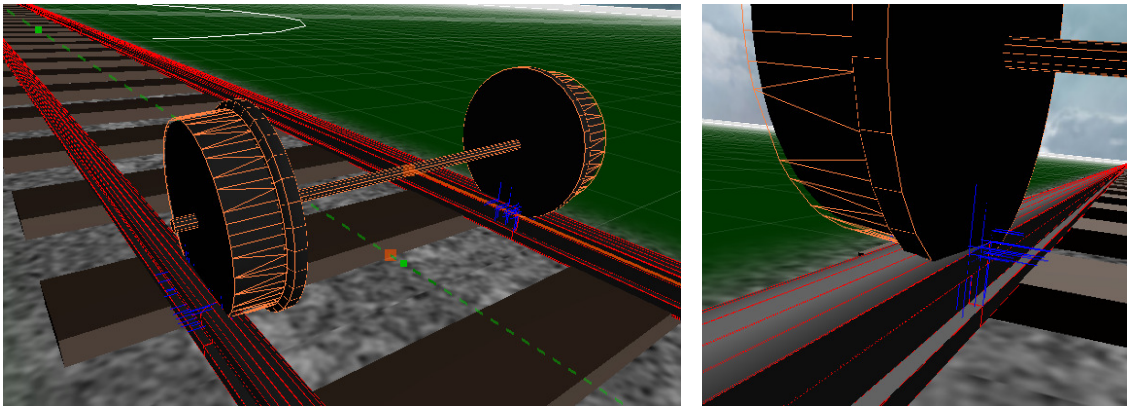


Figure 5.22 - Screenshots illustrating a wheelset stuck in the rails as a result of interpenetration

These screenshots includes the blue markers, which are part of the PhysX debug rendering system and show the contact points between rigid bodies. These contacts only seem to occur where edges of the bodies (red and orange lines) meet, suggesting that this is how PhysX detects contacts between bodies.

Given the dimensions of the wheelset and the rails, it should not be possible for the wheelset to derail in this way. These issues were thought to be the result of penetration error, which was assumed to be the result of the rigid bodies' Skin Width parameter, which by default is set to 0.025m (or 2.5cm). A series of tests were therefore conducted to evaluate the effects of altering skin width.

5.7.5 *Skin Width Testing*

The following tests were designed to determine whether the derailment problems described in the previous section could be eliminated by adjusting the skin width parameter of the wheelset (wheels and flanges of the MB wheelset) and the rails. Skin Width is a parameter of the rigid bodies that controls how much interpenetration between entities is allowed, as described in section 3.4.7.

Test Design

Skin Width of the wheelset (wheel and flanges in the case of the MB wheelset) and of the rails is adjusted in order to evaluate its effect on the lateral offset of the wheelset and the number of derailments that were occurring at low speeds in the previous tests.

Determining the Maximum Skin Width Value

First, tests were conducted in an attempt to determine the maximum skin width value that should be used in the simulation. The skin width was reduced from 0.025m (the default) to 0.01 in increments of 0.001 (1mm) and Table 5.14 (below) shows the number of recorded derailments for each skin width value from 0.025 to 0.017. These tests were conducted at 1mph.

Skin Width	*0.025*	*0.024*	*0.023*	*0.022*	*0.021*	*0.02*	*0.019*	*0.018*	*0.017*
Derailments	10	10	10	10	10	0	0	0	0

Table 5.14 - Derailments per Skin Width (0.025 - 0.017m)

As the table shows, the wheelset derailed in 10/10 tests where the skin width was above 0.02m, but did not derail in any tests at 0.02m or below.

The graph below (Fig. 5.23) shows changes in the average lateral offset of the wheelset as skin width changes.

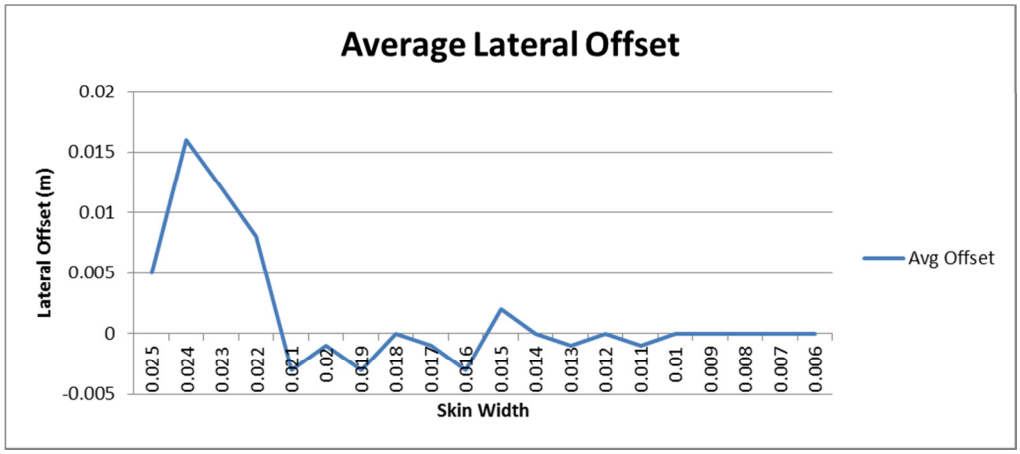


Figure 5.23 - A graph showing how the average lateral offset of the wheelset changes with Skin Width

As the graph illustrates, the offset of the wheelset is high and/or unstable during the tests at 0.025 to 0.021m, where derailments occurred. There is also some instability during tests with a Skin Width of between 0.02 and 0.01, but the average offset is then consistently 0.0 at tests at 0.01m and below. It was expected that the average value should be zero, even if some oscillation occurred, and so data suggests that a Skin Width of 0.01 or below may be the optimum choice and that a Skin Width of 0.02 should be considered to be the maximum value.

Skin Width 0.0

Logically, in order to make the simulation as realistic as possible, a skin width of 0.0 should be used, since the wheels and rails have been modelled on their real-world counterparts as closely as possible. However, the PhysX documentation recommends against setting the Skin Width of two contacting objects to 0.0, as *'it will lead to an unstable simulation'* [4]. To confirm that a skin width of 0.0 produces stability issues, the normal force tests (using a cube placed on a ground plane as previously shown in 5.6.2) were repeated. The skin width of the cube and the ground plane was set to 0.0 and the graph shows the magnitude of the normal force between the two objects at each sample. The data in the graph below (Figure 5.24) was collected during tests with a 1KG cube.

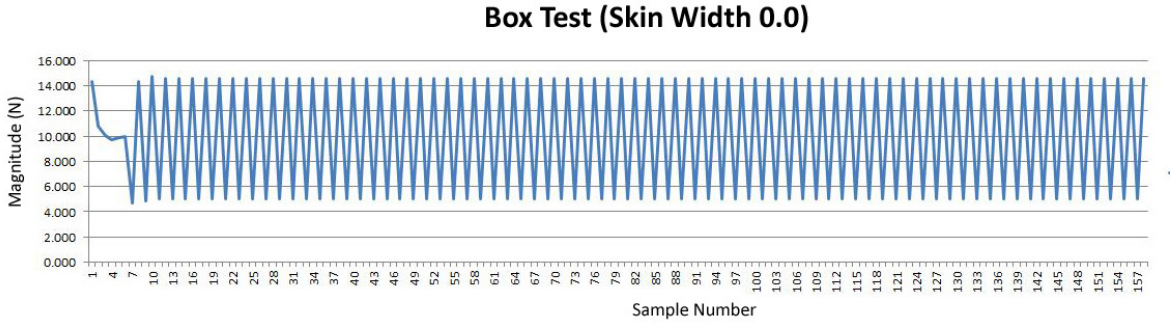


Figure 5.24 - Normal Force Samples at a skin width of 0.0

The average size of the force was 9.81, and so is still very close to the correct magnitude (9.806). However, there were more results than in the previous test (158 callbacks as opposed to the 23 callbacks in the initial test) and the magnitude of the force oscillated within a range of nearly 10N throughout the test. PhysX was able to resolve this instability and declare the box 'at rest' despite this, but such a variance in forces is not desirable for an accurate engineering simulation.

This variance did not occur at a Skin Width of 0.001, as illustrated in the graph in Figure 5.25 (below).

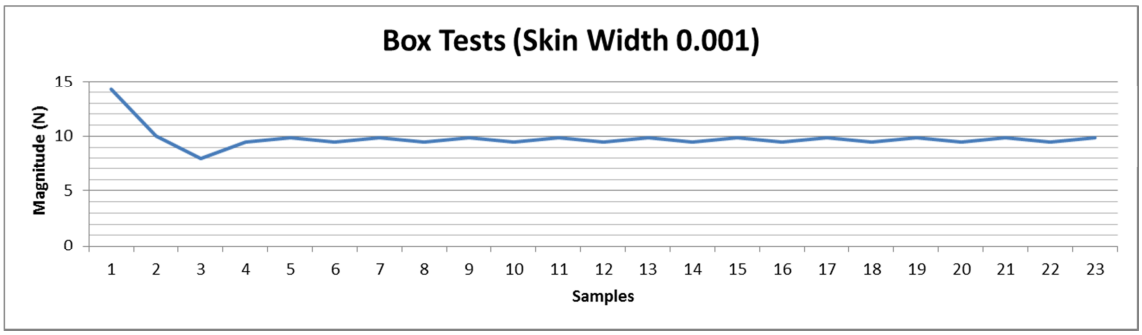


Figure 5.25 - Normal Force Samples at a skin width of 0.001

The graph shows similar results to the wheelset tests (Figure 5.10). There is a large initial magnitude as the objects make contact, which falls below the target value at sample at sample 3, then there is a very small amount of oscillation between the remaining tests, and the results are much closer to the target value of 9.806. The average result is 9.807, just above the prediction, probably due to the erroneously high initial force and the standard deviation is just 1.04N. This confirms that a skin width of 0.0, while arguably the most realistic, produces undesirable behaviours and should not be used.

Step 3: Determining the Minimum Skin Width Value

During initial testing with a skin width of 0.001, it was observed that the rails were flickering. The rails in the simulation are highlighted to indicate when a track section is in contact with one of the wheelsets, and this flickering was the result of wheelsets repeatedly entering and exiting contact with each track section. This contact issue was also causing the nearest point on the track spline to be updated incorrectly, as this can only be calculated when the wheelset and rails are in contact. A number of tests were therefore conducted to determine how low Skin Width values affect the generation of contact callback events from PhysX, and to determine the minimum value of skin width that should be used in further testing.

- *Test Design*

The bogie was tested on the 100m straight track at a range of skin width values between 0.001 and 0.010. Each test was repeated 10 times and the average number of 'start touch' events per track section for each wheelset was recorded. If there are no contact issues, then there should only be 1 of these events per wheelset/track pair.

➢ *Results*

A sample of the data collected during these tests is shown in Table 5.15 (below). The data in question is taken from a test at 1mph using the SB Wheelset, and shows the average number of contact events per track section for the wheelset.

Skin Width	*0.001*	*0.002*	*0.003*	*0.004*	*0.005*	*0.006*	…	*0.010*
'Start Touch' Events	19	23	21	23	1	1	…	1

Table 5.15 - Number of 'start touch' events at varying Skin Widths

This data shows that a Skin Width of below 0.005 causes multiple 'start touch' events, but that multiple contact events did not occur when Skin Width was greater than or equal to 0.005. Based on this data, a Skin Width of 0.005 should be considered to be the minimum value that should be used during any further testing, with 0.02 being the maximum as this is the point above which erroneous derailments begin to occur.

Performance

It was not expected that the skin width would affect performance, but Figure 5.26, below, shows the average framerate from each of the Skin Width tests between 0.005 and 0.02.

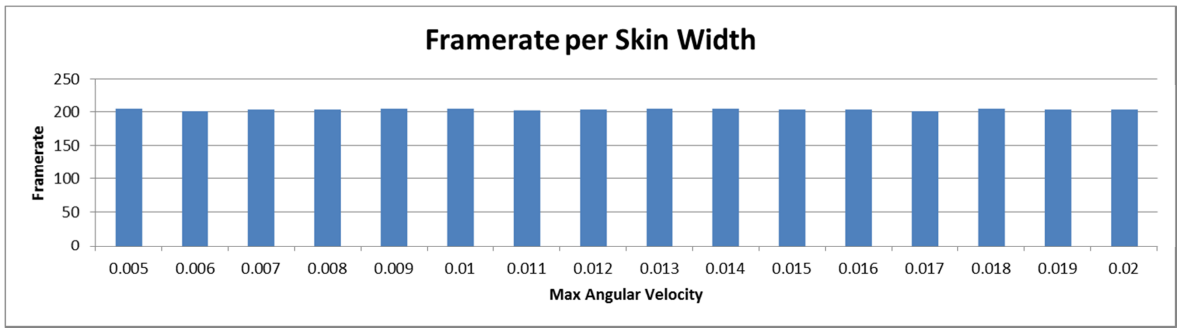


Figure 5.26 - Graph of Performance results from Skin Width Testing

The performance varies between a maximum framerate of 205.1 FPS and a minimum framerate of 201.1 FPS but not in any consistent way that correlated with the changes in Skin Width. This makes sense, as it is not adjusting the fidelity of the simulation, but only the amount of interpenetration allowed between objects.

Conclusions

The data presented in this section suggests that 0.02m should be considered the maximum value of Skin Width allowed, to avoid these unrealistic derailment behaviours in future. The minimum value of Skin Width can be considered to be 0.005, since the results show that the simulation is unstable and multiple contact events are registered if the Skin Width parameter is set below this value.

The data collected during these tests suggests that a value of 0.01 would seem to be the ideal, as this produced the expected lateral offset of zero and there was no further improvement to decreasing the skin width below this value. A value of 0.01 is used in the following tests and, if any future testing of skin width behaviour is to be attempted, the value should be incremented between 0.005 and 0.02. Adjusting the Skin Width had no discernible effect on the performance of the tool.

5.7.6 *Wheelset Polygon Count Testing*

Three variations of wheelset were constructed for testing, as described in Section 3.2.2. The wheels and flanges of these wheelsets were constructed from cylinders in 32, 48 and 64 segments, which allows the stability and performance of the tool to be tested with different polygon counts.

Test Design

In the following tests, the Single Body Wheelset was tested at a target speed of 1mph on a 100m straight track, with each variation of wheelset. Here is a reminder of the three wheelset variations:

- 32 Segments per wheel/flange: 214 Polygons and 344 Vertices
- 48 Segments per wheel/flange: 310 Polygons and 504 Vertices
- 64 Segments per wheel/flange: 406 Polygons and 664 Vertices

The height of the wheelset was measured during each frame of the simulation, to evaluate the stability of the wheelset. Simulation performance is also measured.

Predictions

There is expected to be a higher variation in the height of the wheelset when the polygon count of the rolling surface of the wheel is lower. To explain, consider Figure 5.27 (below), where the wheel is represented as a hexagon.

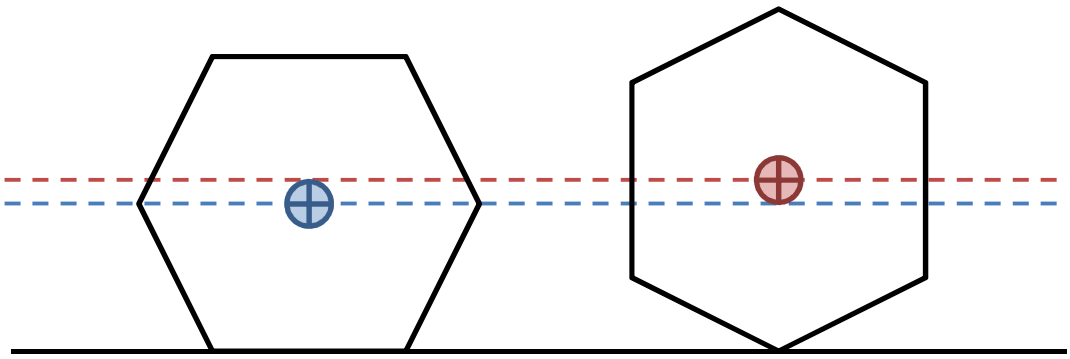


Figure 5.27 - Illustrating changes in the height of a hexagonal wheel.

When the wheel is resting on the track (Figure 5.27 - left) the position of the wheel is at a certain height, illustrated by the blue dotted line. As the wheel rolls forwards (Figure 5.27 - right), the position of the wheel moves higher, as illustrated by the red dotted line. This is an exaggerated example of the effect that occurs with the wheelsets. The 'smoother' the wheel shape, the smaller this difference should be; i.e. increasing the polygon count should improve the instability of the wheelset.

There should be a decrease in performance as a result of the increased number of polygons in the scene. Because the number of polygons that make up the wheelset have almost doubled, this may have a considerable impact on the framerate of the simulation.

Results

Below are the results of testing with the different polygon count wheelsets.

- *32 Segment Wheelset*

Figure 5.28 (below) shows the height (z component of the position) of the 32 segment wheelset (the lowest fidelity model) over the first 150 samples collected during the test.

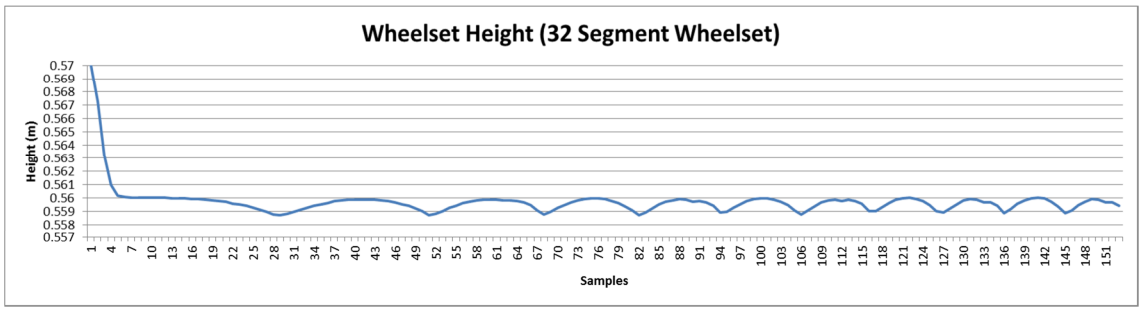


Figure 5.28 - Graph of wheelset height over time

From its initial height of 0.57 metres, it drops approximately 1cm (which is likely to be due to Skin Width) to 0.56m, until it is resting on the rails. As the wheelset rolls along the track, the height varies and the wheel appears to 'bounce' along the track, creating a pattern consistent with the rolling of a polygonal wheel as illustrated in Figure 5.27.

- *Full Results*

Table 5.16 (below) shows the range and standard deviation of the results for each of the three wheelset variations.

Wheel Sections	*32*	*48*	*64*
Range (m)	0.01130	0.01060	0.01040
Standard Deviation (m)	0.00048	0.00038	0.00036

Table 5.16 - Range and Standard Deviation of Wheelset height per Number of Wheel Segments

There is a slight decrease in range, which is expected to be similar in each test as a result of the initial height change/Skin Width. However, there is a significant decrease the standard deviation of the results between the tests, which suggests changes in the wheelsets' stability. Figure 5.29 (below) shows the standard deviation results visually.

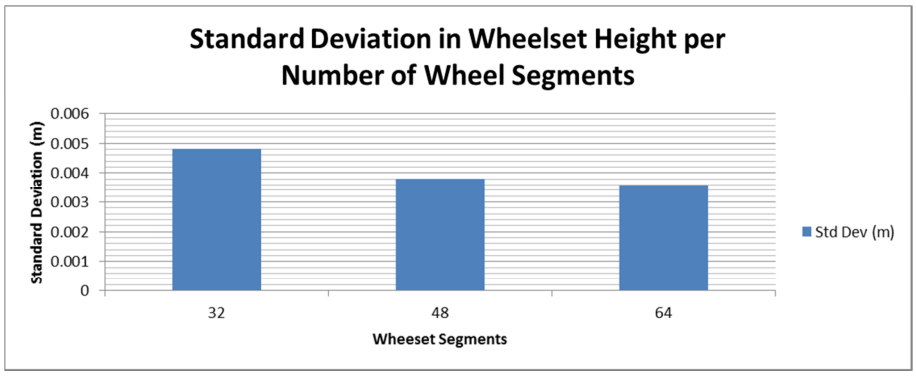


Figure 5.29 - Graph showing the Standard Deviation in Wheelset Height per Number of Wheel Segments

Here, the large decrease between 32 and 48 (0.001m or 1mm) is clearly visible, compared to the smaller difference between 48 and 64 (0.0002m or 0.2mm).

- *Performance*

The table below (Table 5.17) shows the average framerate during each test.

Wheel Sections	*32*	*48*	*64*
Framerate (FPS)	187.999	187.647	187.012

Table 5.17 - Framerate per Number of Wheel Segments

There was not a dramatic change in framerate, but the performance does decrease slightly as polygon count increases. The performance of the 64 segment wheelset was approximately 1 frame per second (0.987 FPS) lower than the 32 segment wheelset, despite the 64 segment wheelset having nearly twice as many polygons.

Conclusions

There are notable stability benefits from increasing the polygon count of the wheelset, but increasing the segments from 48 to 64 produce a smaller improvement in the standard deviation of the results than the increase from 32 to 48, so it is reasonable to assume that further increases are not likely to have any further significant effect. There was also not a significant performance drop; the 34 and 64 section wheelsets only changed the framerate by <1FPS. It is possible that this effect will be multiplied as the number of wheelsets in the scene increases, but the decrease is not as high as expected.

The 64 segment wheelset will therefore be used in all future testing.

5.7.7 *Initial Multi Body Wheelset Testing*

During the initial testing it was noticed that the Multi Body Wheelset was unstable, with apparently weak joints between the axle and the flanges/wheels); the wheels and flanges appeared to be moving separately from the axle, as illustrated in the screenshot below (Figure 5.30 - a rather extreme example of the problem).

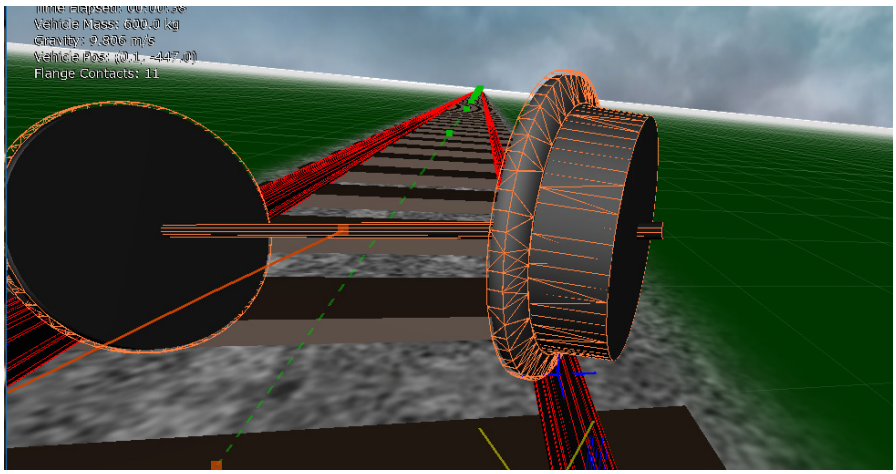


Figure 5.30 - A screenshot showing how the wheel and flange move relative to the axle

Initially the wheelset was constructed by attaching the wheels and the flanges to the axle with fixed joints. It was theorised that if each flange rigid body was also attached to the corresponding wheel rigid body, then the stability of the wheelset may improve.

Test Design

The MB wheelset is tested at 1mph on a 100m straight track, first with the flanges and wheels joined only to the axle ('unattached') and then with the wheels and flanges also joined to each other ('attached'), as illustrated in Figure 5.31, below. The blue arrows represent joints between the rigid bodies.

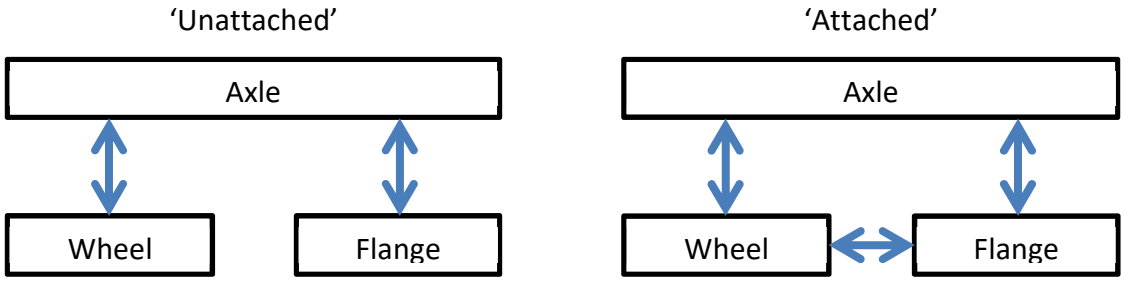


Figure 5.31 - Illustrating the difference between the Unattached (left) and Attached (right) multi-body wheelsets

Predictions

It is possible that introducing additional joints will make the simulation less stable and may cause issues for the solver, or that doing so may strengthen the joints and make the wheelset more stable. It is hoped that the latter will be true.

Results

The following results (Table 5.18) were collected during multiple tests with the MB wheelset at 1mph on a 100m length of straight track. The 'unattached' wheelset derailed in all tests and the 'attached' wheelset did not derail in any of its tests. This is the first indicator that the 'attached' wheelset is more stable.

Wheelset	*Unattached*	*Attached*
Derailments	10	0

Table 5.18 - Derailments per Multi Body Wheelset design variation

The graph in Figure 5.32 (below) shows the lateral offset of the wheelset variations during each test. The blue results are for the 'unattached' wheelset and the red results are from the 'attached' wheelset.

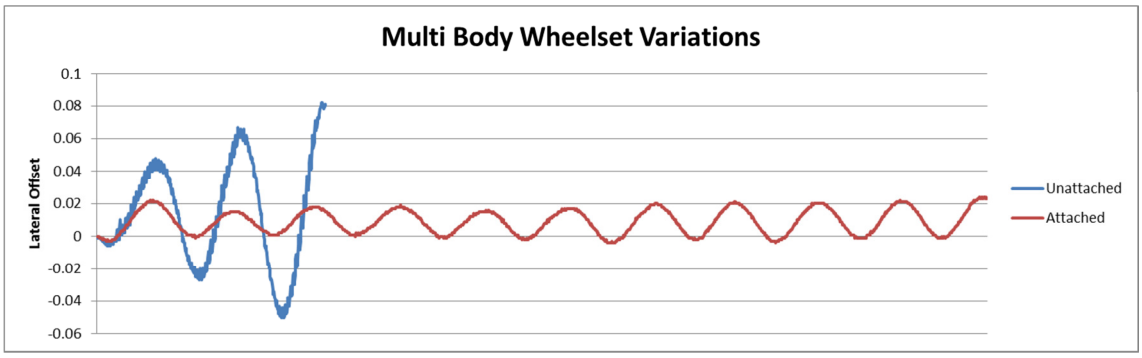


Figure 5.32 - Graph comparing lateral offset for the two variants on the multi body wheelset

This graph shows that the unattached wheelset oscillated wildly before derailing in less than 20m, whereas the attached wheelset made it all the way to the end of the track with a logically correct sinusoidal motion (albeit with preference of 1mm to the left of the track centre).

A closer look at the data reveals the following graph (Figure 5.33 - overleaf)) (up to the point where the unattached wheelset derailed). This graph shows that, not only is the lateral movement of the unattached wheelset higher, but that but that it was highly unstable and oscillated wildly throughout the test.

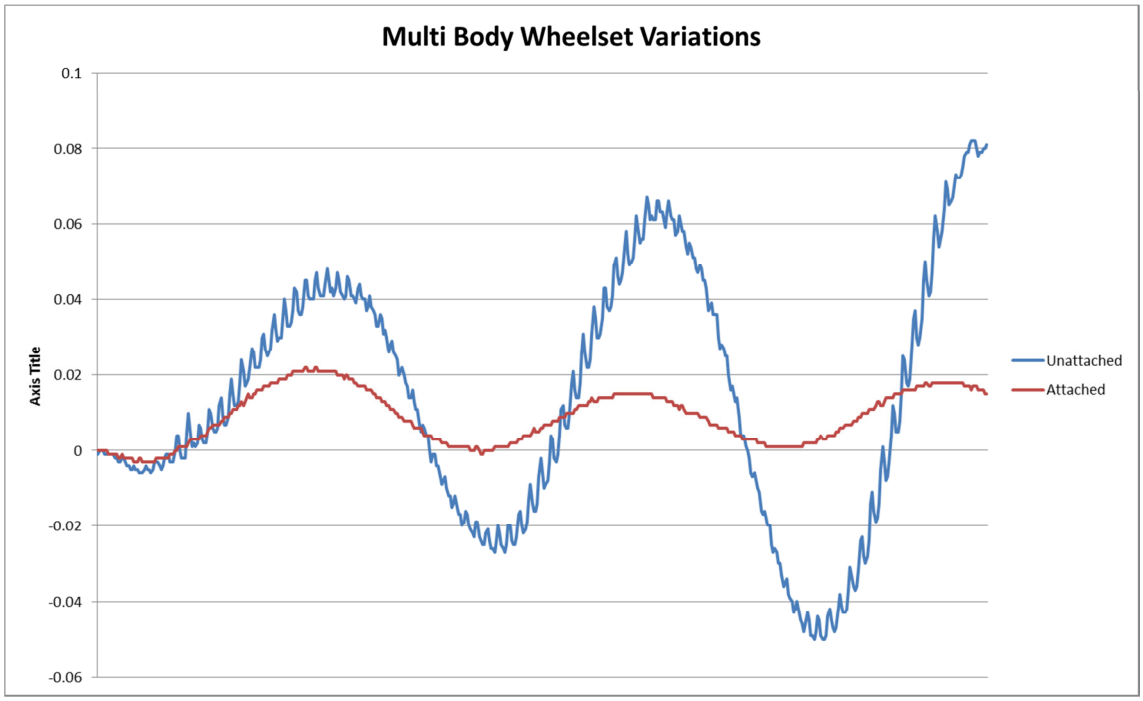


Figure 5.33 - More detailed view of previous graph showing oscillation more clearly

Conclusions

The conclusion from this initial testing of the Multi Body wheelset is that the results are more stable if the two bodies are joined together, but that the joints do seem to introduce an amount of instability to the wheelset. This is undesirable, but if these effects can be mitigated, then the adjustability of the wheelset may outweigh the instability.

The 'attached' version of the Multi Body wheelset is used in all future tests.

5.7.8 *Initial Stable Speed Tests*

The following tests were conducted using the Single Body and Multi Body Wheelsets on a straight track at a range of speeds.

Design

The SB and MB wheelsets are placed on the straight track layout. The track was extended to 1km to ensure that there is plenty of space for the wheelset to accelerate to its target speed and maintain it for a longer distance before reaching the end of the track.

The target speed is initially set to 1mph and each wheelset is tested 10 times. Once that first batch of 10 tests has been completed, the target speed is increased to 2mph and another batch of 10 tests is conducted. After another 10 tests, the target speed is incremented to 3mph and so on. Both wheelsets were tested at speeds of up to 50mph.

Predictions

The target speed for the vehicle is 100mph, so ideally the wheelsets will be able to reach that speed, but they are not expected to do so using the default PhysX engine parameters. It is expected that, the MB wheelset may be less stable than the SB Wheelset, due to the joints between the axle and the wheels/flanges.

Single Body Wheelset Results

The following data was collected during testing with the Single Bogy Wheelset. There were no derailments below 20mph, but the graph in Figure 5.34 (below) shows the number of derailments between 10 and 30mph.

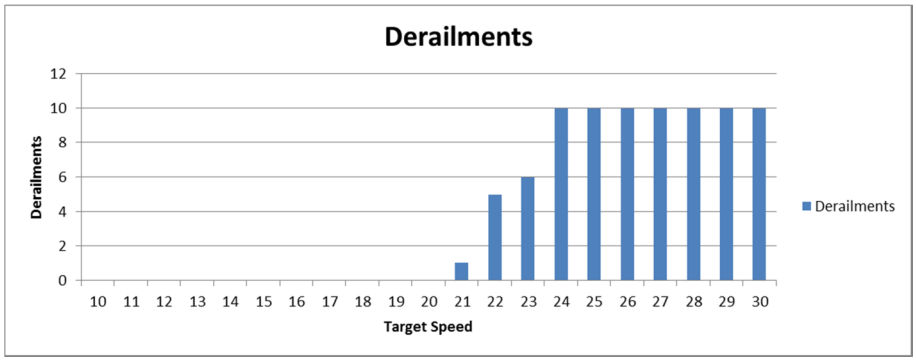


Figure 5.34 - Graph of Derailments per Target Speed (Single Body Wheelset)

This data suggests that the wheelset became unstable at speeds of approximately 20mph, as it derailed in at least some of the tests in all batches above that speed. By the time the target speed of the vehicle reaches 24 mph (and above) the vehicle derailed in every test. 20mph is the highest speed at which no derailments occurred. This is known as the 'stable speed', as defined earlier (Section 4.1.4) and used in the rest of the thesis.

The graph below (Figure 5.35) shows the peak speed recorded during each test (blue), as well as the range of recorded speeds (red).

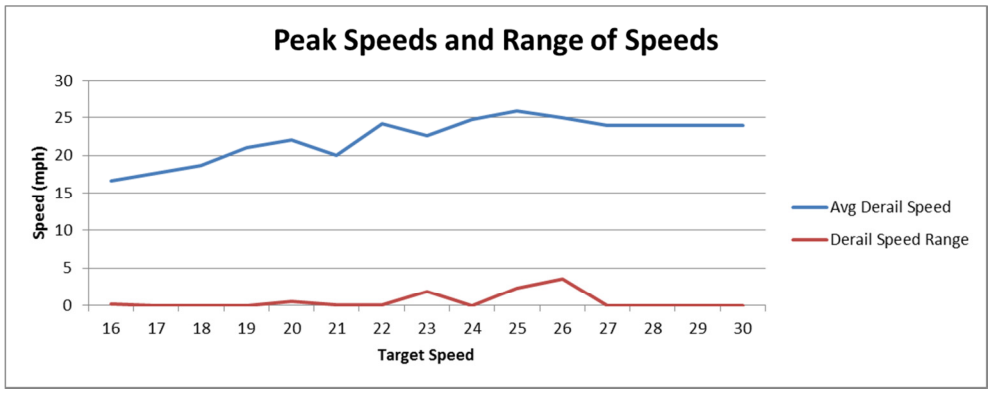


Figure 5.35 - Average and range of derailment speeds for a range of target speeds (BB Wheelset)

The top speed of the vehicle increases with the target speed, indicating that the vehicle achieved its target speed during the tests in which it didn't derail (20mph and below). There are some fluctuations in the peak speeds between 21 and 26 mph, possibly indicative of the instability of the wheelset that caused the derailments. At 27mph and above the vehicle derails at, on average, 23.993 mph in all tests. The fact that the range of results in these tests is 0 (or close to 0) suggests that it is derailing in a consistent manner.

The derailment speed data for a range of target speed values shown in more detail in Table 5.19, below.

Target Speed (mph)	*21*	*22*	*23*	*24*	*25*	*26*	*27*
Average	22.115	22.586	24.197	24.770	25.858	24.946	24.000
Standard Deviation	0.146	0.729	0.058	0.000	0.713	1.539	0.000
Range	0.471	1.922	0.110	0.000	2.304	3.560	0.000

Table 5.19 - Results for a range of Target Speeds (21 - 27mph) (Single Body Wheelset)

Although the derailment speed of the vehicle is low, the small range and standard deviation of the results is promising. A real-time system that could estimate the derailment speed of a vehicle to within 3.56 mph could be considered sufficiently accurate as to be useful to engineers.

Multi Body Wheelset

The tests were repeated with the Multi Body Wheelset. The wheelset was noticeably less stable than the Single body Wheelset, as previously indicated in Section 5.6.4. The screenshot below (Figure 5.36) shows the flange climbing the rail, but the wheel and flange have rotated separately from the axle. This is more stable than in the multi body tests in the previous section (as illustrated in Figure 5.30), but still perceptible. Here the flange is highlighted in yellow to indicate contact with the rails.

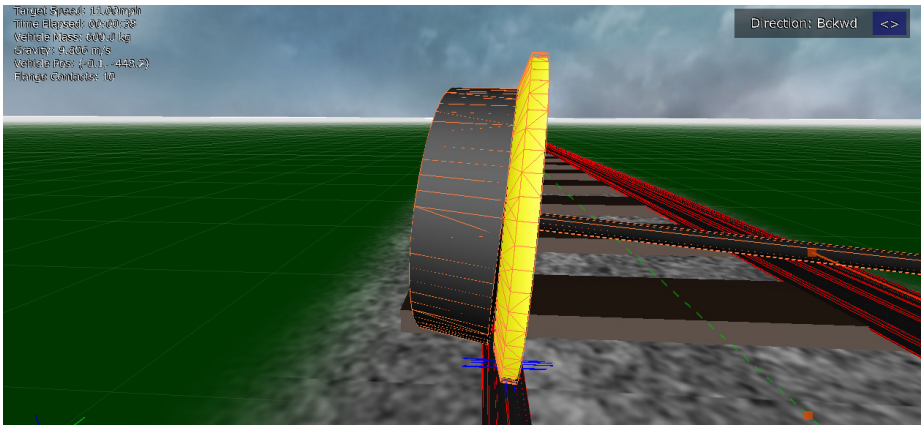


Figure 5.36 - Wheel and Flange moving separately to the axle in Multi Body Wheelset Testing

The graph in Figure 5.37 (below) shows the number of derailments for the different target speeds of the wheelset.

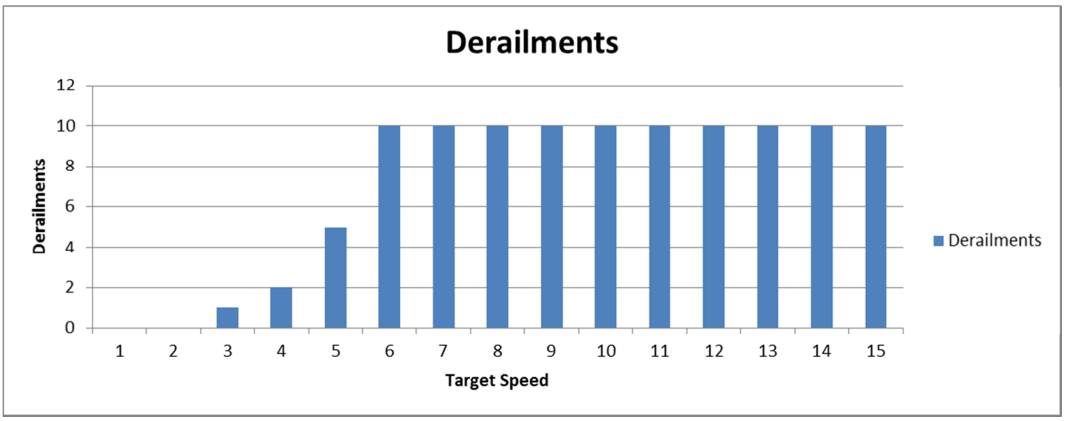


Figure 5.37 - Derailments per target speed - Initial Testing (multi Body wheelset)

The wheelset did not derail in tests at 1 or 2 mph, but became unstable at 3 mph and derailed in all tests above 6mph. 2mph can therefore be considered to be the stable speed of the multi body wheelset. This is considerably lower than the 20mph stable speed of the single body wheelset.

The graph below (Fig. 5.38) shows the average derail speed (blue) (or the peak speed in tests that it did not derail) and range of derail speeds (red).

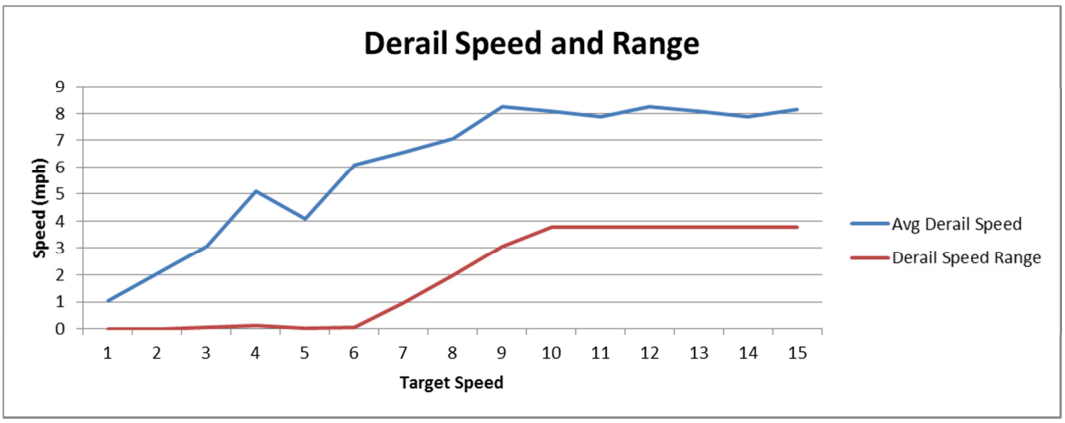


Figure 5.38 - Average Derail Speed and Derail Speed Range (Multi Body Wheelset)

The average derailment speed increases, but so does the range. The range is higher than the equivalent results for the SB Wheelset, suggesting the MB Wheelset is less stable.

Conclusions

The low top speed (approximately 25 mph for the SB Wheelset and 4 mph for the MB Wheelset) is a concern, as is the high range of results. This suggests a lack of stability and consistency in the simulation, both of which mean that the simulation would not be very useful to engineers. The MB wheelset is significantly less stable than the SB Wheelset in these tests, derailing at lower speed and with a higher range of results, but it is possible that adjusting parameters of the physics engine, identified in Section 3.4.7, might improve the fidelity of the simulation and so improve the top speed and consistency of results. If this is the case, the flexibility of the MB wheelset is worth pursuing further.

5.7.9 *Summary*

These tests have identified the maximum speed that each wheelset can achieve using the simulation's default parameters, as well as the speed at which the wheelset is able to travel along the track without derailing, herein referred to as the 'Stable Speed'.

These tests have shown that:

- Increasing the polygon count of the wheelset improves stability, but further increase above 64 segments is unlikely to produce any further significant improvements.
- Joining the wheels to the flanges of the multi body wheelset produces more stable results, but that the multi body wheelset appears to be significantly less stable than the single body wheelset, and produces less consistent results.

Table 5.20 (below) shows the highest speed at which the wheelsets did not derail during each test, herein referred to as the "stable speed", along with the top speed:

Wheelset	Stable Speed (mph)	Top Speed (mph)
Single Body	20	26.14
Multi Body	2	8.41

Table 5.20 - Stable Speeds (Initial Testing)

As expected, the Multi Body Wheelset is less stable than the Single body Wheelset. The derailment speeds of the wheelsets on straight track are disappointingly slow. It is possible that these results can be improved with adjustments to the physics engine parameters.

These results, conducted using the Physics engine's default parameters, will be used as a baseline, against which any modifications to the parameters of the physics engine will be measured.

Max Angular Velocity

During initial testing at higher speeds, the top speed of the wheelset was found to be limited by a parameter called Max Angular Velocity (MAV).

- The value for MAV in Future Tests is (unless specified otherwise): 170.
- Increasing/decreasing MAV does not affect the performance of the simulation tool.

Skin Width

Tests in this section have shown that Skin Width should be between 0.02 and 0.005, otherwise stability issues or unrealistic derailment events occur.

- The tests in this section suggest an ideal value for skin with of: 0.01.
- Increasing/decreasing skin with does not affect the performance of the simulation.

Performance

The average framerate for the wheelsets (on a 1km track at a speed of 1mph over the course of 10 tests) is as follows:

- The Single Body Wheelset: 253.50 FPS.
- The Multi Body Wheelset: 237.56 FPS.

This is a very high framerate, approximately 4 times faster than the target speed of 60 FPS (and therefore running at a speed of 4 times real-time). This framerate is expected to decrease as the fidelity of the simulation (and accuracy of the results) increases. There is a difference of 15.94 FPS between the SB wheelset and the MB wheelset, suggesting, as expected, that constructing the wheelset from separate rigid bodies comes at a performance cost.

5.8 Increasing Wheelset Speed

The tests in this section were designed to evaluate the effect of altering various physics engine parameters on the speed and stability of the wheelset. In order to be useful, and in order to utilise the Nadal formula to evaluate wheelset/bogie derailment, the top speed of the vehicle needs to be increased (ideally to the target speed of 100mph). The physics engine parameters tested in this section, previously identified as having the potential to improve the fidelity of the simulation (as described in Section 3.4.7), are:

- Simulation Timing
- Rigid Body Solver Iteration Count

These tests also include an evaluation of the improved centring technique described in Section 3.3.13.

5.8.1 *Simulation Timing*

The first attempt to improve the fidelity of the simulation, and so increase the top speed of the wheelset, was to adjust the simulation's timing parameters. As described in Section 3.4.7, the simulation has an iteration count and maximum substep size that is used during the call to PhysX Simulate.

- The default substep size is $1/60^{th}$ of a second
- The default number of substeps is 8.

Timing Multiple (TM)

A multiple, herein referred to as the 'Timing Multiple' (or 'TM') was added to these values to adjust them from their default values. A Timing Multiple of 1 represents the default values, a value of 2 doubles the number of substeps and halves the size of the maximum time step, and so on. This increase in the substeps and decrease in the size of the timesteps is expected to improve the stability of the simulation and so increase the top speed/stable speed of the wheelset.

Test Design

Tests were initially conducted under identical conditions to the tests in the previous section. The tests are repeated, with the TM being incremented by 1 between tests. These tests are intended to test two values, to allow the results to be compared with the results from the previous section:

- *Top Speed* - the top speed of the vehicle achieved before derailing. This is tested by setting the target speed of the vehicle to 100mph and measuring the peak speed measured while the wheelsets are in contact with the rails.
- *Stable Speed* - this is defined as the highest speed at which the wheelset can travel without any derailments in any of the 10 tests. This is measured by incrementally increasing the target speed of the vehicle between each batch of 10 tests.

Predictions

When testing the top speed of the vehicle, it is intended that the vehicle should derail in every test and it is expected that the wheelset's derailment speed will increase as the TM is increased. It is also expected that the stable speed with also increase with the TM. It is also expected that there may be a value of TM, after which no further improvements are made (or the results will begin to deteriorate).

The performance of the tool is expected to decrease as the Timing Multiple is increased.

Initial Results (Single Body Wheelset - 1Km Straight)

The graph below (Fig. 5.39) shows the results for the Single Body Wheelset for Timing Multiples of 1 to 20. Figure 5.39 shows the change in the peak speed of the vehicle.

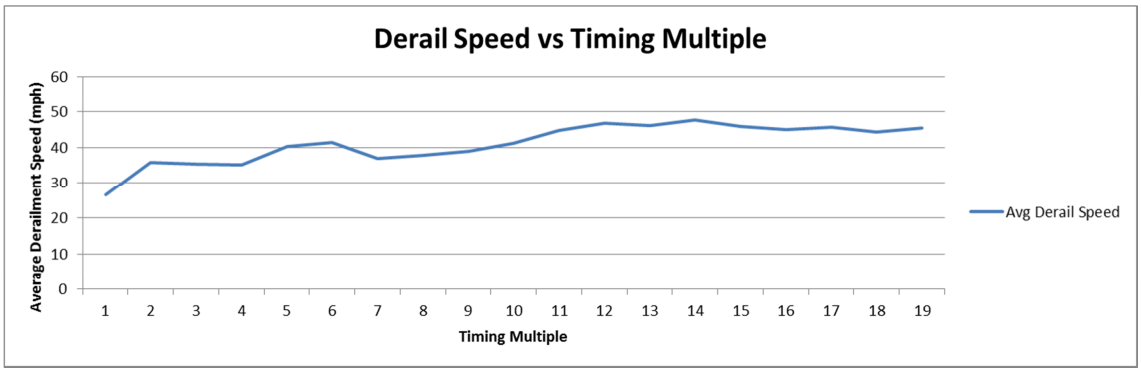


Figure 5.39 - Derailment Speed per Timing Multiple (Single Body Wheelset)

From the original peak speed of approximately 25 mph, the vehicle was able to reach an improved peak speed of around 45 mph.

Figure 5.40, below, shows the number of derailments in each test.

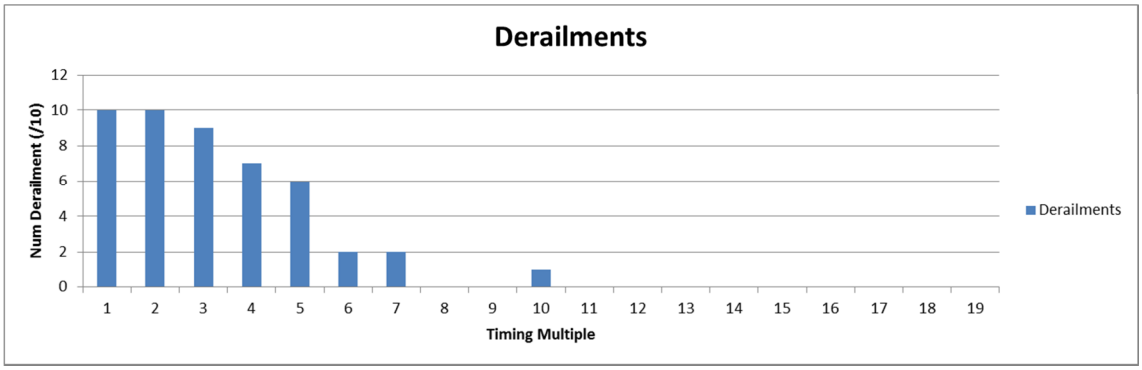


Figure 5.40 - Graphs of Average derailment speed (top) and Number of Derailments (bottom)

While the results in Figure 5.39 shows that the top speed of the vehicle does increase with TM, the results in Figure 5.40 showed that the vehicle was reaching the end of the track before it could accelerate any further and not derailing, suggesting that it was reaching the top speed that it was capable of achieving on that track length.

Extending the Track and Increasing Wheelset Acceleration

In order to allow the vehicle to accelerate to higher speeds and derail before reaching the end of the track, the track was extended to 2km and the tests were conducted again.

➢ *2km Straight Tests*

During these tests, the vehicle reached a similar 'plateau' to the one in Figure 5.39, at a speed of around 68mph, suggesting that stability is increasing, but without derailing, implying that there was still not time for the wheelset to accelerate to full speed.

➢ *Increasing Acceleration*

The torque applied to the wheelset is updated in the Wheelset Entity's 'update' method. This method is passed the timestep between frames (1/60) as a parameter. The torque is initially set to a magnitude of zero, and increased by the timestep during each frame. In order to make the wheelset accelerate more quickly, the timestep was multiplied by two, allowing the wheelset to accelerate to a higher speed over a shorter distance.

Single Body Wheelset (2km Straight)

With the rate of acceleration increased and the track length extended to 2km, the single body wheelset was retested at a range of timing multiples.

➢ *Peak Speed Tests*

These tests were designed to determine the peak speed of the vehicle at each timing multiple. The target speed was set to 100mph and the derailment results are as follows: The wheelset derailed in every test, as intended, and the graph below shows the average derail speed (blue) and the range of derailment speeds (red) for each timing multiple between 1 and 50. The grey dotted line shows the approximate trend in the results

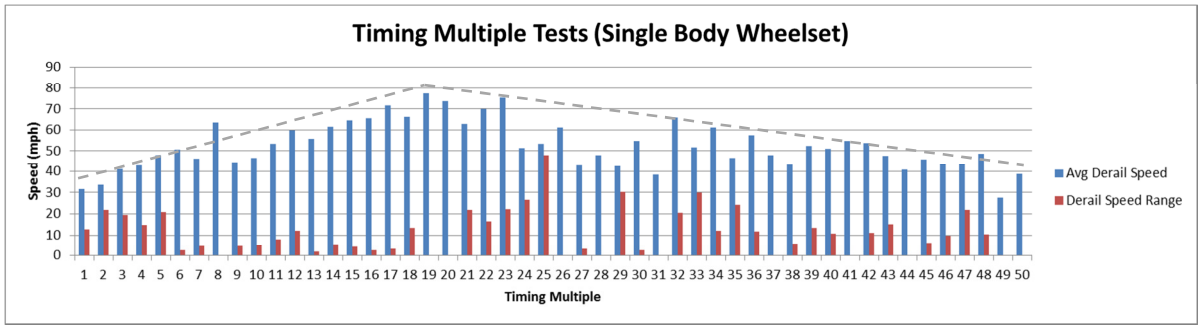


Figure 5.41 - Timing Multiple Test Results (Single Body Wheelset)

The average derailment speed of the vehicle increases with Timing Multiples of up to 19, after which it decreases again. A multiple of 19 produces the best results (the highest average derailment speed (77.45mph) and the lowest range of derailment speeds (0).

➢ *Stable Speed Tests*

These tests were designed to determine the stable speed of the vehicle at each TM. The Stable Speed in the previous tests was 20mph. The graph in Figure 5.42, below, shows the results from testing at a Timing Multiple of x2.

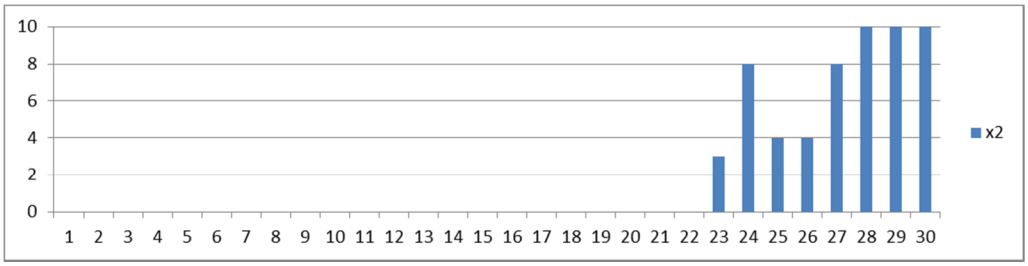


Figure 5.42 - Stability Tests - Single Body Wheelset at a Timing Multiple of x2

This shows an improvement over the default (x1) tests; the stable speed has increased from 20 to 22mph. This trend continues, as illustrated by the following examples from tests at TM x4 and x8 (Figure 5.43 - top and Figure 5.43 - bottom, respectively).

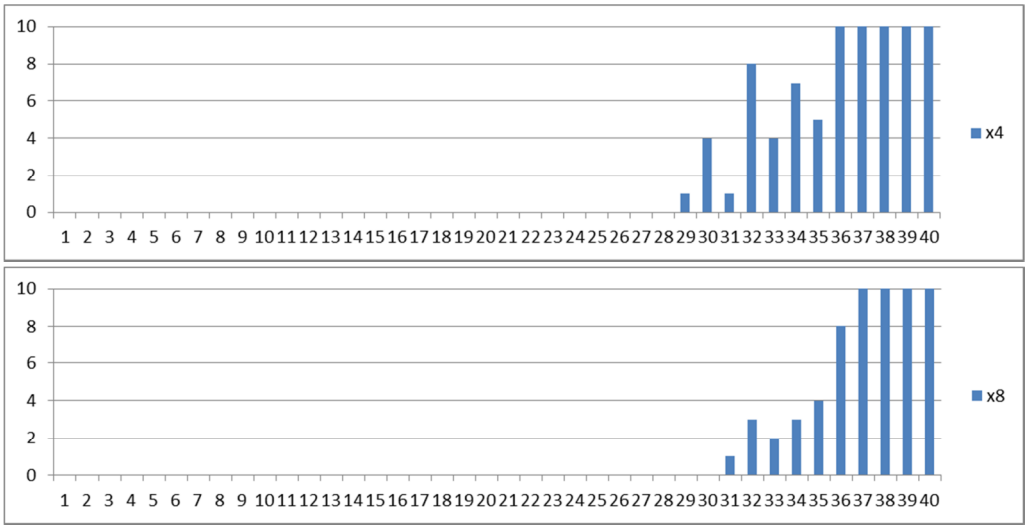


Figure 5.43 - Stability Tests - Single Body Wheelset at Timing Multiples x4 (top) and x8 (bottom)

By x8 (bottom) the wheelset is capable of reaching speeds of 30mph before becoming unstable. To determine if this trend continues, further testing was conducted with timing multiples from x14 to x32, in order to determine which value produced the best results. In these tests, there were no derailments below 30 mph, and so the x axis of the graphs ranges from 30mph up to 70mph. Data from tests at a TM of x14 are included in Figure 5.44 (below) and tests from x16 and x18 are included in Figure 5.45 (overleaf).

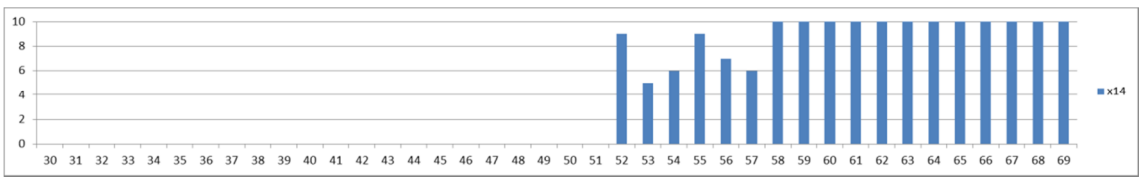


Figure 5.44 - Stable Speed Tests (x14)

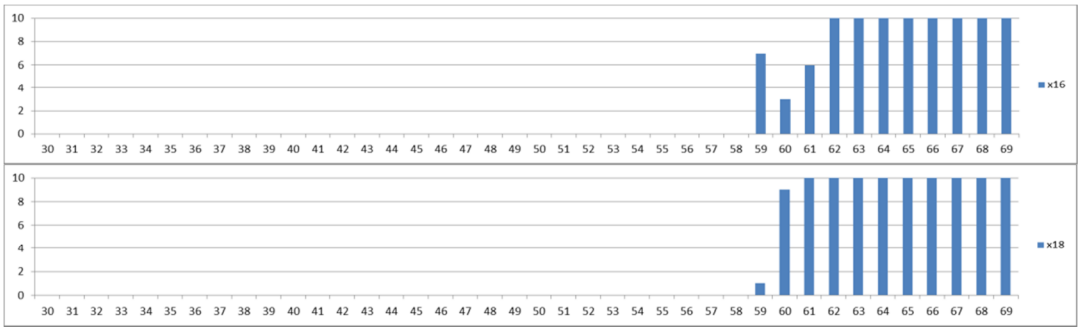


Figure 5.45 - Stable Speed Tests - x16 (top) and x18 (bottom)

These results show a continued increase in the stable speed of the vehicle. Figure 5.46 (below) shows the results from testing at a TM of x20:

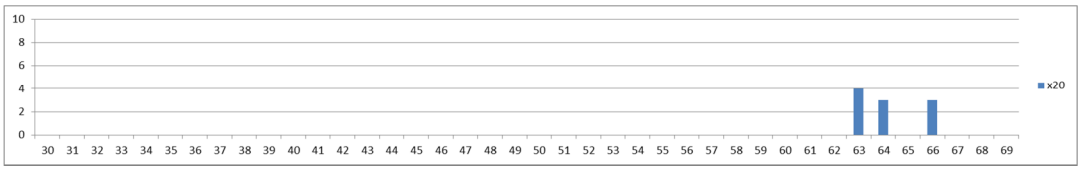


Figure 5.46 - Stability Tests (Single Body Wheelset - Timing Multiple x20)

In this test, the stable speed of the wheelset reached 62 mph, and shows a continued improvement over the previous results. Here, again, despite the fact that there were no derailments at 65, 67, 68 and 69mph, the stable speed is defined as the highest speed at which no derailments occurred, and so 62 mph is the result. This data offered promise that the wheelset may be able to achieve higher stable speeds if the parameters were adjusted further. However, this trend does not continue, as illustrated in the graphs below (Figure 5.47), showing the results from testing at a TM of x22, x24 and x26.

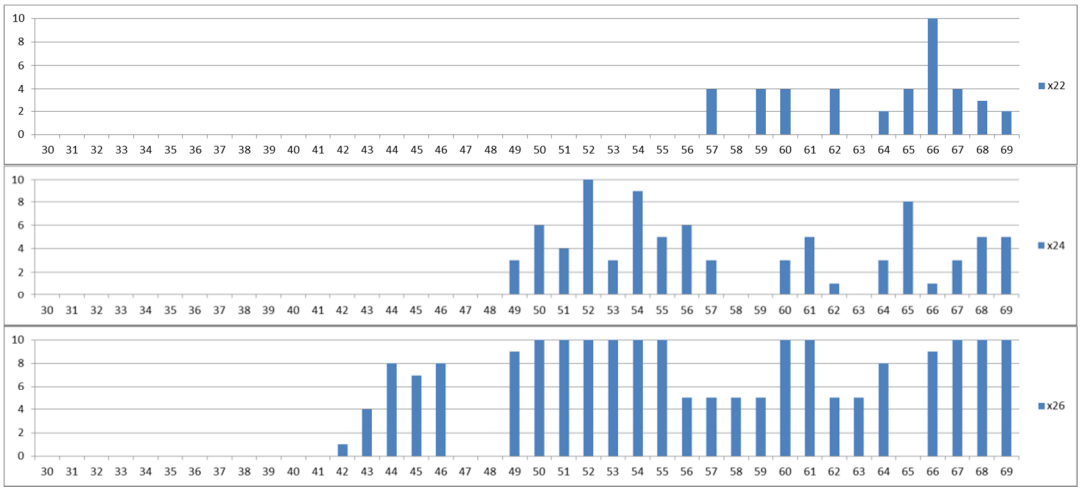


Figure 5.47 - Stability Tests (Single Body Wheelset x22 (top) x24 (mid) and x26 (bottom).

In these tests, the stable speed of the vehicle decreases, to 56 in the x22 tests (top) to 48 in the x24 tests (mid) and to 41mph in the x26 tests (bottom). This supports previous assertions, and the PhysX documentation, which estimated that increasing the parameters beyond a certain point can cause the simulation to become less stable. This shows the stable speed of the vehicle increases up to a TM of 20, and then decreases above 20, with a TM of 20 producing the best results.

Figure 5.48 (below) shows the stable speed achieved in each test between x14 and x30. The grey dotted line shows the approximate trend in the results.

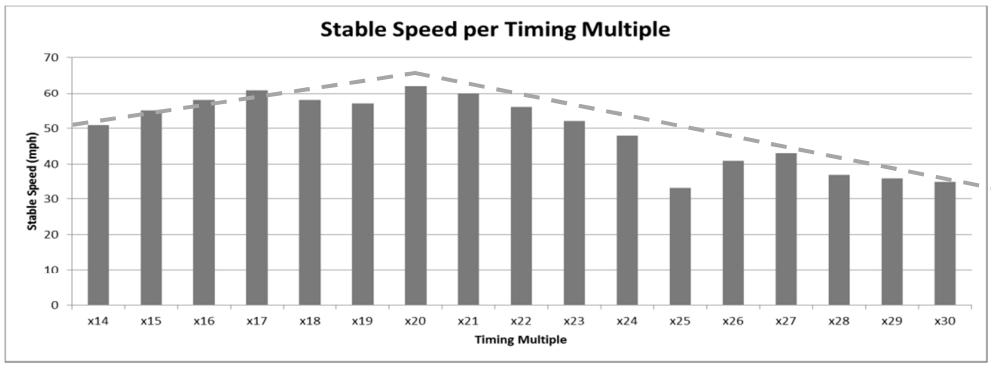


Figure 5.48 - Stable Speeds of the Single Body Wheelset (Timing Multiple x14 to x30)

These results show that the stable speed of the wheelset increase with the TM until the multiple reaches 20, and then the stability of the vehicle begins to decrease again. This suggests that 20 is the best result to use for the Single Body Wheelset.

Multi Body Wheelset (2km Straight)

The results for the multi body wheelset are as follows.

- *Peak Speeds*

The wheelset derailed in every test and Figure 5.49 (below) shows the average derail speed (blue) and the range of derailment speeds (red) for each TM between 1 and 50.

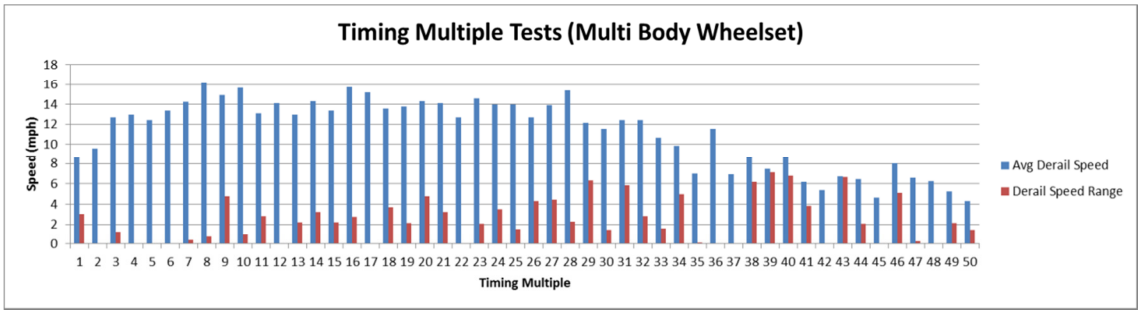


Figure 5.49 - Peak Speed Tests (Multi-body Wheelset - Timing Multiples)

As was the case in the Single Body tests, the average derail speed initially increases with TM, but above a certain value it decreases again, supporting the hypothesis that increasing the value above a certain point has a detrimental effect on the stability of the simulation. The highest derailment speed was 16.16mph, recorded with a TM of 8. The range of results in that test was 0.725mph. Another promising result occurs at a TM of 17, where the average derail speed was 15.17mph and the range of results was 0.

➢ *Stable Speeds*

The stable speed tests were repeated with incremental increases in Timing Multiple. The graphs in Figure 5.50 (below) show the results for tests with a TM of 1 to 8. The x axis represents the target speed of the vehicle; the y axis is the number of derailments in each batch of tests.

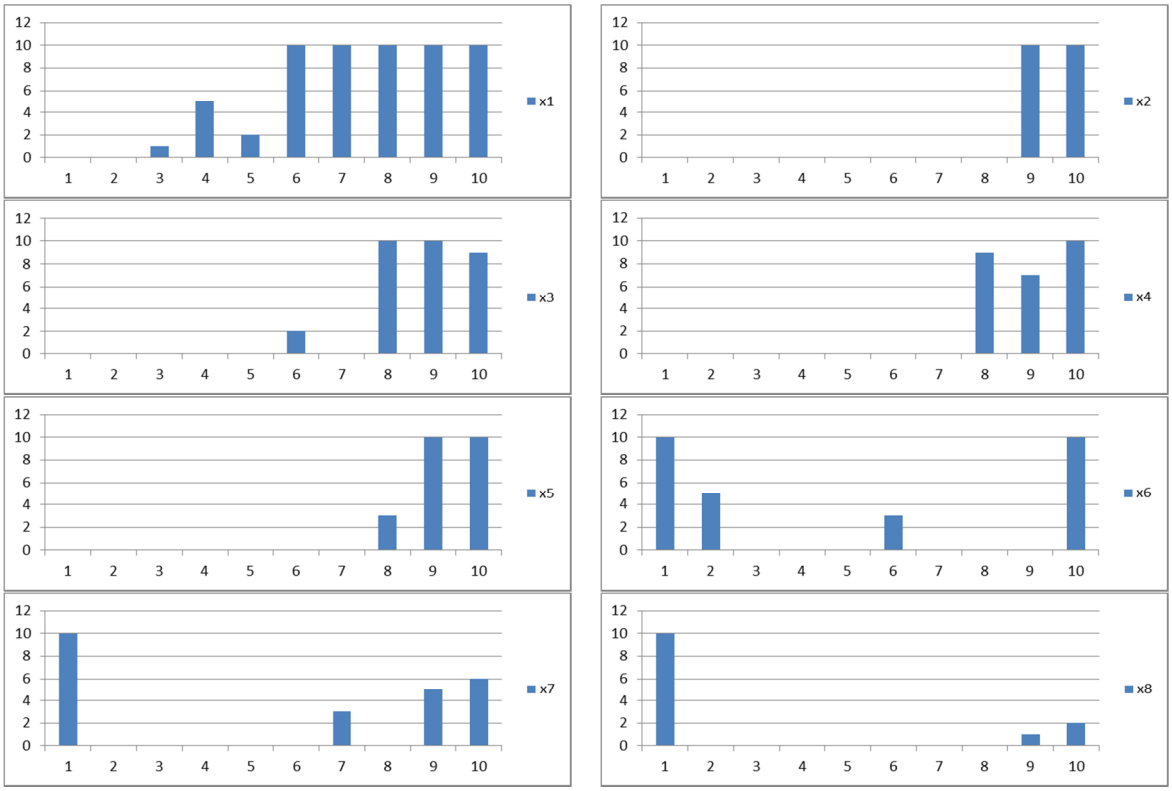


Figure 5.50 - Derailments per Target Speed for Timing Multiples 1 - 8

Changing the TM does have an effect on the results, but the differences are less consistent than the Single Body Wheelset tests. For example, between x1 and x2, the stable speed increases from 2mph to 8mph, but at TM values of x3 and x4, the results are less stable. The results improve in the x5 test, but are not better than the x2 results. By the x7 and x8 tests, the wheel doesn't derail in all tests at 10mph, but does derail in all tests at 1mph.

The graphs in Figure 5.51 (below) show further testing between a TM of x9 and x16.

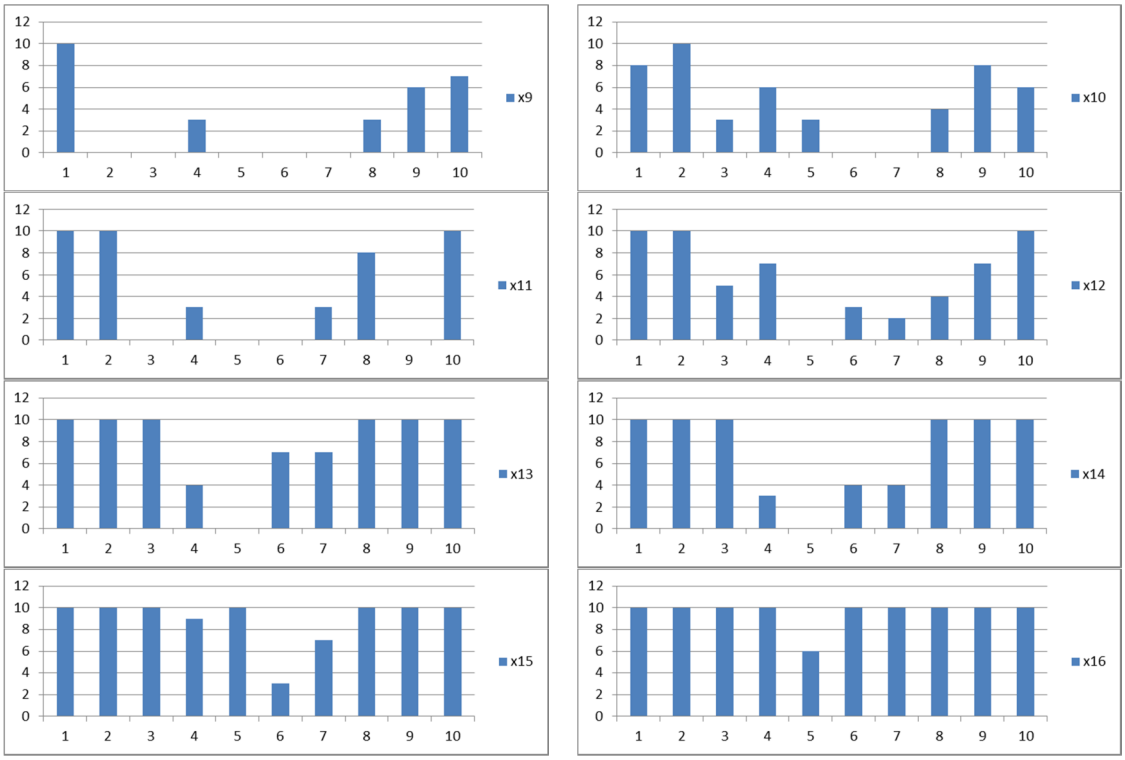


Figure 5.51 - Derailments per Target Speed for Timing Multiples 9 (top left) - 16 (bottom right)

There is some variance in the results, and there are derailments in most (or all) of the tests conducted at 1mph. There are tests, such as 7mph in the x9 test and 9mph in the x11 test, where the vehicle does not derail at higher speeds, but this research is interested in the stable speed, as it is necessary to be able to test the vehicle reliably across the widest range of speeds possible. In all tests with a TM above 17 (up to 25) the wheel derails in all tests at all speeds and TM values.

Further testing was conducted using TM values between 1 and 10, since the previous data suggests that the wheelset may be more stable at higher speeds, even though derailments occur in all tests at 1mph. In these tests, the vehicle was tested at a higher range of speeds between 1 and 20 mph with Timing Multiples of between 1 and 10. The results of these tests are shown in Figure 5.52, below, and Figure 5.53, overleaf.

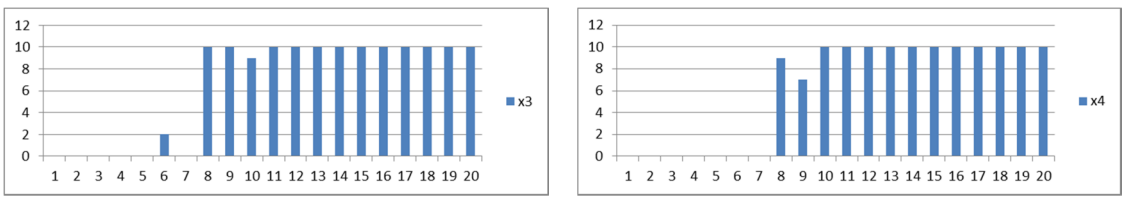


Figure 5.52 - Derailments per Target Speed for Timing Multiples of 3 and 4 (1 - 20mph)

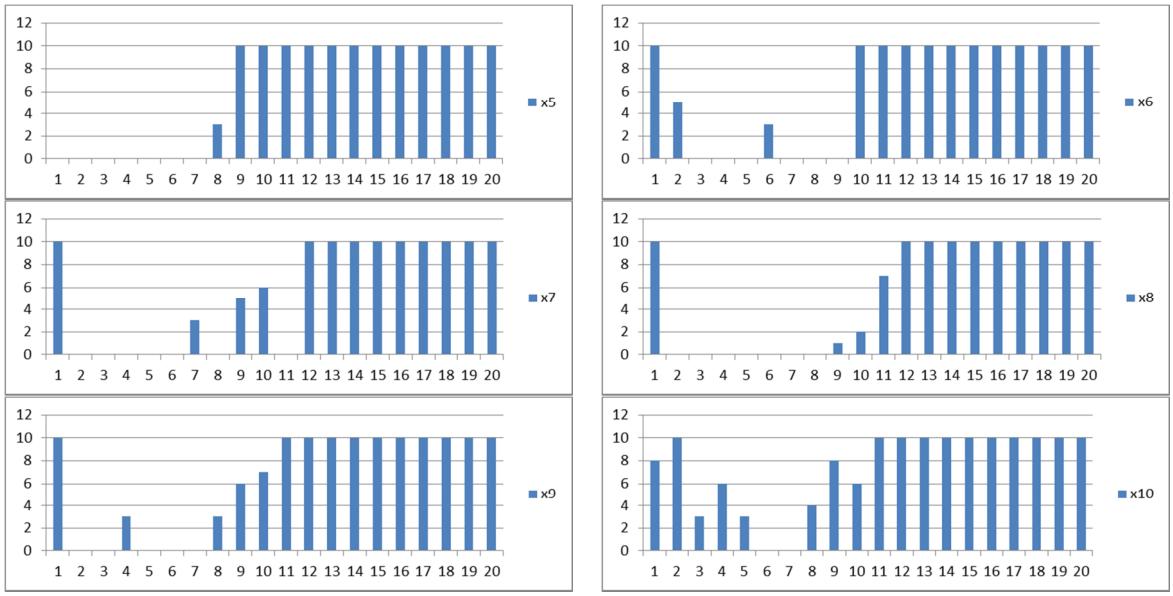


Figure 5.53 - Derailments per Target Speed for Timing Multiples 5 - 10 (1 - 20mph)

There are a few tests in which the wheelset did not derail at higher speeds, for example it does not derail at 11 mph in tests at x7, even though it derailed at lower speeds. However, the wheelset derails in some or all tests at 12mph and above in all batch tests.

The graph in Figure 5.54 (below) summarises the results of the TM tests on the MB Wheelset, showing the stable speed for each batch of tests at each TM value.

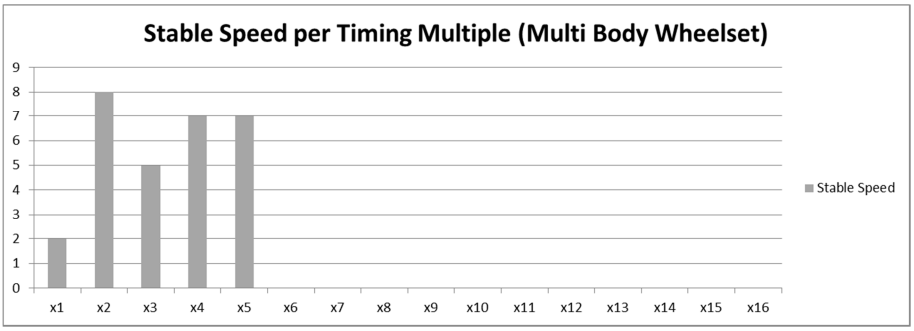


Figure 5.54 - Stable Speed per Timing Multiple (Multi Body Wheelset)

From the initial stable speed of 2mph at a TM of x1, there is an increase to 8mph at x2, then 5mph at x3 and 7mph at x4 and x5, at x6 and above, the stable speed is zero, because the vehicle derailed in many (or all) of the tests at 1mph in each batch of tests, despite the wheelset not derailing at higher speeds in some of the tests.

This data suggests that a TM value of x2 produces the best result for the MB Wheelset.

Performance

Figure 5.55 below shows the average framerate recorded in each test with both wheelset designs. The lines show the change in framerate over the course of testing for the SB Wheelset (red) and MB Wheelset (blue). Both tests were conducted under the same conditions as the previous performance tests (a 1km track at 1mph).

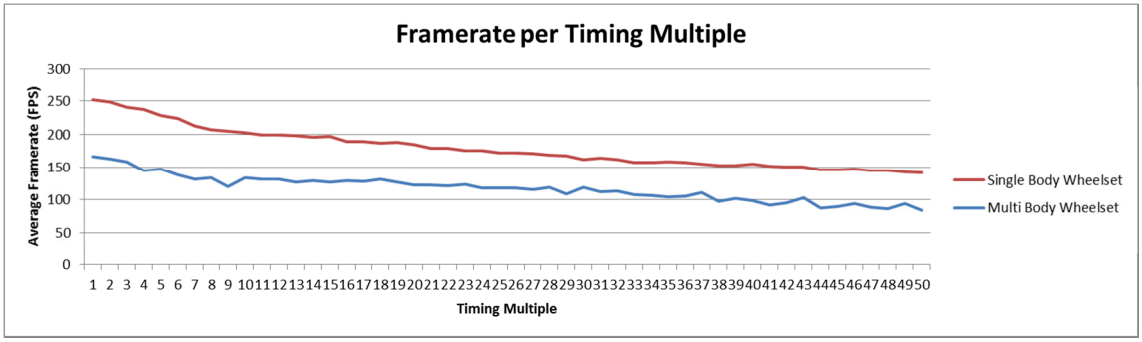


Figure 5.55 - Framerate per Timing Multiple (Single Body and Multi Body Wheelset (x20))

There is a decrease in framerate as the TM increases, as expected, but it is still higher than the target framerate of 60FPS for both wheelsets. The SB Wheelset decreased from a maximum of 178.8 to a minimum of 142.2 and the MB Wheelset decreased from a maximum of 166.2 to a minimum of 84.4 FPS. The Multi Body Wheelset's performance is lower than the Single Body wheelset in all tests, which is to be expected as the physics engine is processing multiple bodies and the joints between them, rather than a single rigid body.

Conclusions

Both wheelsets showed an improvement in top speed and stable speed as a result of the increase in Timing Multiple, as expected, but the Multi Body Wheelset is still derailing at speeds significantly below the Single Body Wheelset.

In some tests, the vehicle was able to travel along the straight without derailing at speeds above the identified 'stable speed', however the aim of this research is to produce reliable results across as wide a range of speeds as possible, in order to enable engineers to test the widest possible range of scenarios.

- The best result for the SB wheelset was achieved with a TM of 20.
- The best result for the MB wheelset was achieved with a TM of 2.

Table 5.21 (below) shows the best results for the stable speed of the wheelsets (using the parameters described above), compared to the values attained using the physics engine's default parameters.

Wheelset	Default Parameters	TM Best
Single Body	20 mph	62 mph
Multi Body	2 mph	8 mph

Table 5.21 - Stable Speeds (Timing Multiple Testing vs Default Parameters)

There is an improvement in the stable speed of both wheelsets, but the Multi Body wheelset is still considerably less stable than the Single Body wheelset; the MB Wheelset reaching a stable speed of 8 while the SB wheelset achieved a stable speed of 62.

5.8.2 *Solver Iteration Count*

The next parameter to be tested is the Solver Iteration Count (SIC), which, as described in Section 3.4.7, is a per-rigid-body parameter that controls the number of solver iterations performed when processing joints and contacts relating to the body.

Test Design

Tests are conducted across the range of SIC values.

- The default value for SIC, used in initial testing, is 4.
- The range of values for this parameter is from 1 to 255

Tests are conducted in intervals of 5, from the default of 4 up to the maximum value of 255, allowing the full range of values to be tested in approximately 51 batches (of 10 tests) - requiring only 510 tests instead of the 2,550 tests it would take to tests the whole range of values. If there is a trend in the results, as there was in the previous section, it should still be noticeable.

Predictions

It was hoped that increasing the value of the SIC parameter would produce some improvement to the results for the Single Body Wheelset. It was also hoped that a more significant improvement might be achieved with the Multi Body Wheelset, since the SIC parameter effects contacts and joints between bodies, and the wheel, flange and axle are connected by fixed joints.

The performance of the simulation is expected to decrease as SIC increases.

Single Body Wheelset

Using a Timing Multiple of 20, the SB wheelset was tested with a range of SIC values.

➢ *Peak Speeds*

Figure 5.56 (below) shows the peak speed of the vehicle as SIC is incremented. The wheelset derailed in every test and the results below represent the average speed recorded across each batch of 10 tests.

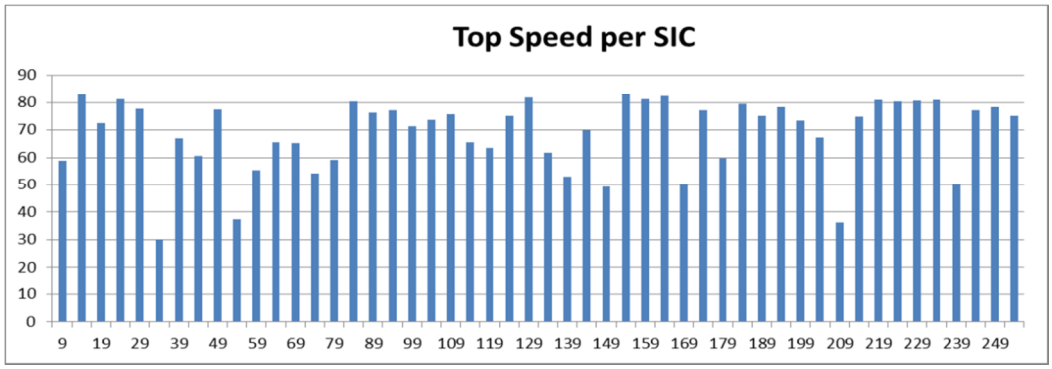


Figure 5.56 - Graph of Peak Speed per SIC (Single Body Wheelset

Speeds range from a lowest speed of 46.15 mph to a highest speed of 83.19 mph. There is no clear, consistent pattern, however this result has not necessarily been a particularly useful indicator in the past, as the average derailment speed maximum has not always occurred using the same parameters that achieved the maximum stable speed. The range of derailment speeds in all tests was 0, suggesting that the vehicle is always derailing in a consistent way (an improvement over initial testing).

➢ *Stable Speeds*

Figure 5.57 (below) shows the stable speed of the vehicle at each SIC value.

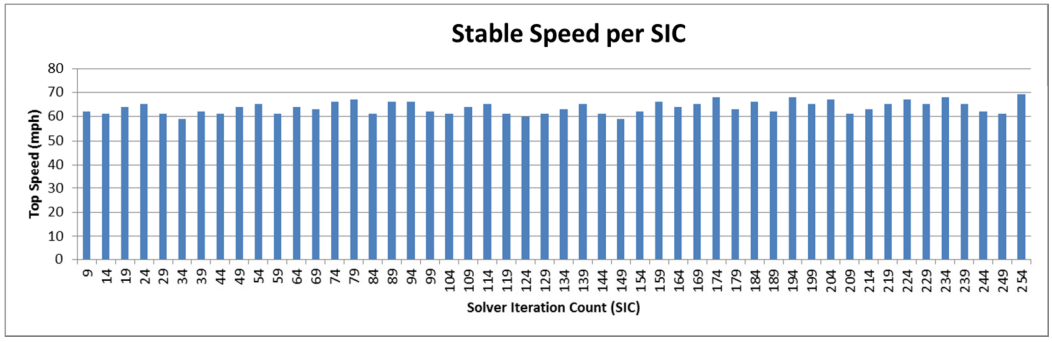


Figure 5.57 - Stable Speed per SIC (Single Body Wheelset)

There is not a significant increase in the stable speed of the wheelset, which reached a speed of 62 mph in the previous Timing Multiple tests. The best results were achieved in the SIC 184 tests, which produced a stable speed of 68mph, an increase of 6mph.

Multi Body Wheelset Results

The following tests were conducted with the Multi Body wheelset

- *Top Speeds*

Figure 5.58 (below) shows the top speed achieved, on average, in each batch of tests.

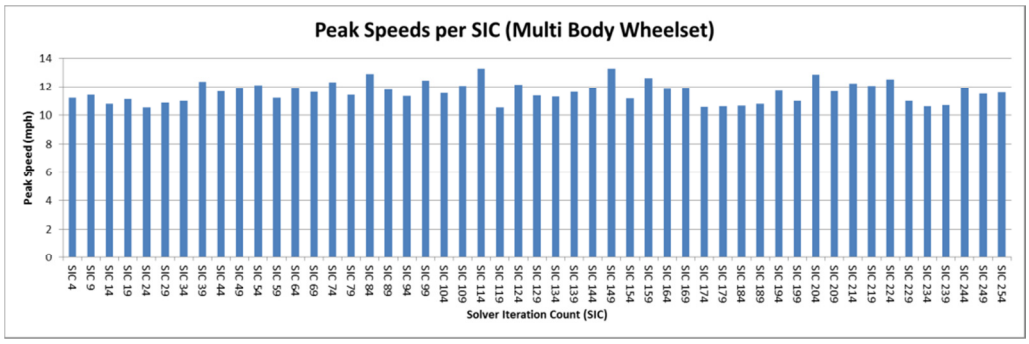


Figure 5.58 - Peak Speeds per SIC Value (Multi Body Wheelset)

As in the SB Wheelset tests, there is some variation in the results from 10.52mph to 13.28mph, but not in any consistent way. The best results were achieved at an SIC value of 114.

- *Stable Speeds*

In the previous tests, the maximum stable speed of the vehicle was achieved at a Timing Multiple of 2 (8mph).Figure 5.59 (below) shows the stable speed for each increment of SIC across the range of parameter values tested.

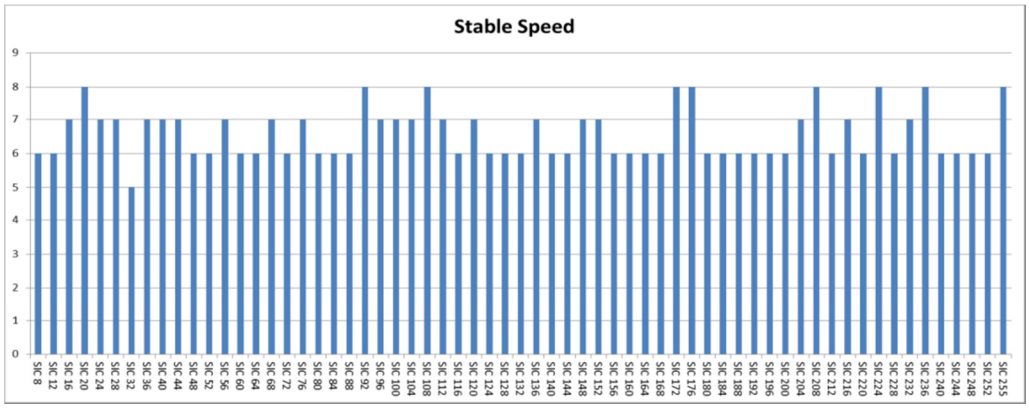


Figure 5.59 - Stable Speed per SIC (Multi Body Wheelset)

There is some variation in the results, but no improvement in the stable speed of the wheelset; the best results (SIC 20, 92, 108, 172, 176, 208, 224, 236 and 225) produced a stable speed of 8mph, the same achieved with the default value (SIC 4) in the previous tests.

Performance

Figure 5.60 (below) shows how the framerate of the simulation changed across the range of testing. Both of these tests were conducted under the same conditions; a Timing Multiple of 20 and increasing SIC values

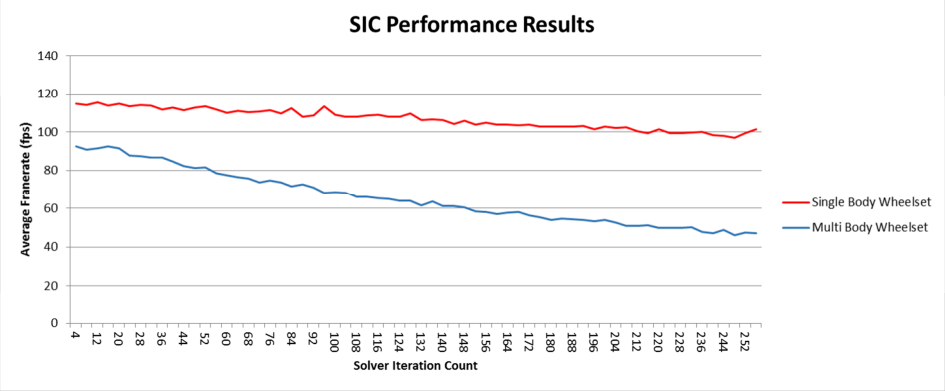


Figure 5.60 - Solver Iteration Count Testing - Performance Results

These results show a steady decline in framerate over the course of the test, as expected. They also show that the Multi Body Wheelset not only produces a lower framerate in the first batch of tests (SIC 4) but that the framerate for the MB wheelset also decreases at a faster rate than the SB Wheelset as the SIC value increases. The MB wheelset drops below the target framerate of 60fps at around SIC 140.

Summary

The best results were achieved SIC values of:

- Single Body Wheelset: 184
- Multi Body Wheelset: 4 (no improvement)

Table 5.22 (below) shows the best stable speeds achieved by each wheelset, compared to previous results.

Wheelset	Default Parameters	Timing Multiple Best	SIC Best
Single Body	20 mph	62 mph	68 mph
Multi Body	2 mph	8 mph	8 mph

Table 5.22 - Stable Speeds (Solver Iteration Count Testing)

The SB wheelset showed an improvement of 6mph, whereas the MB Wheelset showed no improvement. In the case of the SB Wheelset, SIC values of 194, 234 and 254 also produced a stable speed of 68 mph, but it was decided to use the SIC value that produces not only the best results, but also the best performance.

5.8.3 *Centring Forces*

It was hypothesised that the reason for the wheel derailing at such low speeds (and for the high amount of lateral offset and the presence of hunting oscillation in the conical wheelset tests in Section 5.7.1) was that the centring forces of the wheelset, while shown to exist in the conical wheelset tests, may not be as high as they should be. So, it was decided that an artificial centring force would be added to the wheelset, as described in Section 3.3.14, in an attempt to improve the results further.

Test Design

Due to the lack of stability and performance of the Multi Body Wheelset, it was excluded from these tests. The results below are from tests conducted with the Single Body Wheelset. The centring force was enabled and the wheelset was tested on the 2km straight.

Results

Table 5.23 (below) shows the results, which are included along with the initial test results for the Single Body Wheelset and the results without the centring force applied.

Test	Stable Speed (mph)	Top Speed (mph)
Initial Results	20	26.14
Best Results without Centring Force	68	74.10
Best Results with Centring Force	76	79.65

Table 5.23 - Peak and Stable Speeds of the Single Body Wheelset (initial testing, best results without centring force, centring force)

The application of the centring force has resulted in an increase in the stable speed and derailment speed of the vehicle, to 76mph and 79.65mph respectively.

5.8.4 *Summary*

The adjustment of PhysX engine parameters and the implementation of the additional centring force have been shown to successfully increase the stable speed of the wheelset, as shown in Table 5.24 (below).

Wheelset	Default Parameters	TM Best	SIC Best	Centring Force
Single Body	20 mph	62 mph	68 mph	76 mph
Multi Body	2 mph	8 mph	8 mph	*N/A*

Table 5.24 - Stable Speeds (Centring Force Testing)

The wheelset was unable to reach the target speed of 100mph, instead reaching a maximum stable speed of 68mph. Application of the Gravitational Stiffness Force improved the results, the SB Wheelset achieving a stable speed of 76mph.

PhysX Parameters

The tests in Section 5.8.1 and Section 5.8.2 show that adjusting the simulating timing parameters and SIC can increase the top speed of the wheelset. The following parameters have been identified as producing the best results for the SB Wheelset. The single body wheelset was able to reach its highest stable speed of 76mph.

- TM = 20
- SIC = 184

Wheelset Polygon Counts

Increasing the polygon count of the wheelset was shown to increase the stability of the rolling motion, as discussed in Section 5.7.6. The increase in stability between 48 and 64 suggests that increasing the polygon count beyond that of the 64 segment wheelset will not significantly improve the stability of the wheelset further. The drop in framerate was also negligible, despite the increase in polygon count.

- The 64 Segment wheelset is therefore used in all future testing.

Wheelset Designs

The MB Wheelset is significantly less stable than the SB Wheelset, and its stability cannot be significantly improved. It achieved lower peak and stable speeds in initial testing, and showed only negligible improvements as the simulation parameters were adjusted. This suggests that the SB wheelset is the better choice for engineering simulation because, despite not being as flexible as the MB Wheelset, it is significantly more stable. There are performance considerations too, as the MB wheelset produced a lower framerate than the equivalent SB tests, and in many tests was below the target framerate of 60FPS.

- The Single Body Wheelset is therefore used in all future tests.

Performance

The performance tests from the previous section (1mph on a 1km straight track) were repeated for the SB Wheelset using the parameters above, and the results were:

- Single Body (No Force): 173.61 FPS (78.89FPS lower than initial testing)
- Single Body (With Force): 166.90 FPS (86.6 FPS lower than initial testing)

There is a difference of 7.71 FPS between the two tests, which suggests that calculating and applying the force to the wheelset has a small impact on the performance of the simulation. With the centring force applied, this is a reduction in performance of approximately 86 FPS over the results achieved using the default parameters of the simulation, but 167 FPS is still more than twice the real-time target of 60fps.

5.9 Wheelset in Motion (Curved Track)

The tests in this section were conducted on the loop track layout, using a Single Body Wheelset.

5.9.1 *Test Design*

As with the previous tests, the wheelset is tested at a range of speeds, this time using a looped track layout. The wheel is tested on both left and right turning track to check for consistency. From the tests in the previous section, using the following settings:

- Timing Multiple: 20
- SIC: 184
- Gravitational Stiffness Force applied

5.9.2 *Predictions*

The wheelset has a maximum stable speed of 76mph. At this speed there should be no derailments on curves above a radius of 150m. At radii of 150m and below, the wheelset should derail at speeds close to those predicted by the Nadal formula, which are reiterated in Table 5.25, below.

Curve Radius (m)	*100*	*125*	*150*
Speed (mph)	58.96	65.92	72.21

Table 5.25 - Predicted Derailment speeds for a wheelset on 100 to 150m radius curves

5.9.3 *Results*

The wheelset was first tested on the 1,000m radius curve, the widest that has been constructed for testing, at speeds ranging from 1mph to 10mph. Table 5.26 (below) shows the number of derailments recorded during each batch of 10 tests for each target speed value.

Target Speed	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Derailments	10	10	10	10	10	10	10	10	10	10

Table 5.26 - Derailments Per Target Speed (Single Body Wheelset - 1,000m radius curve)

The wheelset derailed at all speeds in all tests conducted. The results were the same for all curve radii tested. It would appear from these results that the wheelset is incapable of traversing any track curve at any speed.

The screenshot below (Fig. 5.61) shows the wheelset derailing in one of the tests.

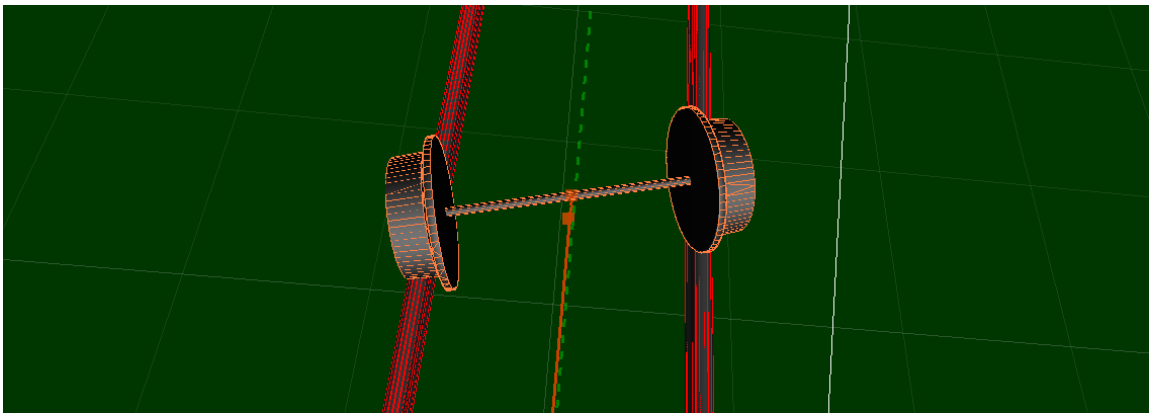


Figure 5.61 - a single wheelset becoming 'stuck' on a curved track

In the majority of the observed tests, a derailment was detected because the wheelset had rotated away from the track centre, as illustrated in the figure above, and was unable to roll forward. The derailment detection code includes a check for the wheelset becoming immobile for 5 seconds, and this criterion that was triggered in the majority of these tests. In other tests, derailment was detected because the wheel had rotated so far as to drop off the rails and come into contact with the ground plane.

Tests were repeated with and without the centring force applied, but the results were the same.

5.9.4 *Performance*

The simulation ran at an average of 98.46 fps on the 1,000m radius track. This is significantly lower than the 1km straight track (166.9fps), but this is to be expected due to the number of rigid bodies in the scene, and the complexities of those bodies. This is still above the target of 60 FPS.

5.9.5 *Conclusions*

A single SB wheelset is unable to traverse the looped track at any speed on any curve radius. It could be argued that simulating a single wheelset in this way is of limited use to rail engineers, but it would have been useful to be able to apply the Nadal predictions to evaluate its cornering behaviour. However, this has not been possible.

In Section 5.11, testing is conducted with a bogie, which for various reasons that will be discussed, may produce more stable results and is closer to the target application of the tool; the simulation of complete rail vehicles.

5.10 Cone Wheelset Curve Testing

An additional set of tests was designed at this stage in the testing process, which not part of the original plan but was added in response to the results from the previous section. The following results were collected using the conical wheelset on the loop track layout, in order to determine if the centring behaviour was logically correct or was occurring at all on curved track

5.10.1 *Test Design*

The wheelset is tested on left and right curving loops at speeds of 1mph to 10mph. The wheelset is first tested without the centring force, and then with the centring force applied to determine what effect this has on the results. The lateral offset of the wheelset and the number of derailments in each batch of tests is recorded.

5.10.2 *Predictions*

Due to the size and shape of the conical wheelset, the wheelset should not derail in the same fashion as the single body wheelset in the previous tests. The wheelset may or may not traverse the curves at these speeds, depending if the centring behaviour is correct. The behaviour and lateral offset of the wheelset should be the same on left and right curving track. The average lateral offset of the wheelset should be higher at higher speeds and higher on smaller curve radii, as the centrifugal force acting on the wheelset is higher.

5.10.3 *Results (No Centring Force)*

The following tests were conducted without the application of centring forces to the wheelset. Tests were conducted on a bend of 1,000m in radius at a range of target speeds. The wheelset derailed in all tests and did not complete any loops of the track, as shown in Table 5.27, below.

Speed (mph)	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Derailments (Left)	10	10	10	10	10	10	10	10	10	10
Derailments (Right)	10	10	10	10	10	10	10	10	10	10

Table 5.27 - Derailments Per Target Speed (Cone Wheelset, 1000m radius, no force)

5.10.4 *Results (With Centring Force)*

The following test results (Table 5.28, below) were collected on a bend of 1000m in radius at a range of target speeds, with the centring force applied.

Speed (mph)	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Derailments (Left)	10	10	10	10	5	0	0	0	0	0
Derailments (Right)	10	10	10	10	6	0	0	0	0	0

Table 5.28 - Derailments per Target Speed (1) (Cone Wheelset, 1000m radius, with centring force)

The wheelset derailed at low speeds (below 5mph) but did not derail in tests above 5mph. Further testing was therefore conducted at speeds of 11 to 20mph, and these results are shown in Table 5.29, below.

Speed (mph)	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*
Derailments (Left)	0	0	0	0	4	10	10	10	10	10
Derailments (Right)	0	0	0	0	5	10	10	10	10	10

Table 5.29 - Derailments per Target Speed (2) (Cone Wheelset, 1000m radius, with centring force)

The wheelset did not derail in tests between 11 and 14 mph, derailed in several tests at a speed of 15mph and derailed in all tests at 16mph and above.

The results for left and right hand bends are very similar, though there were more derailments on the right hand bend at 5mph and 15mph than there had been on the left hand bend. The results are identical if the tests are repeated. Although the difference between the curve directions is small, this suggests that the wheelset was somehow less stable on right curving track than left.

A closer look at the lateral offset data (for the tests where no derailments occurred) reveals the graph below (Figure 5.62).

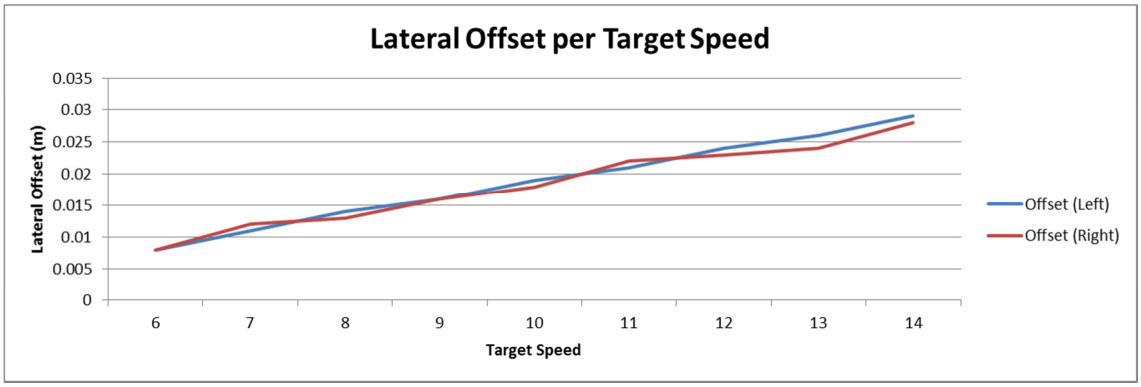


Figure 5.62 - Lateral Offset per Target Speed

The offset values for the left (blue) and right (red) movement for the wheelset are very similar. This is logically correct, as the offset increases with speed and is very similar on both curve directions.

The results were repeated across a range of curve radii from 100m to 1,000m. In all tests below 500m, the wheelset derailed in all tests at all speeds. Other results for 500 to 900m produced similar results to the 1,000m tests, derailing in some of the low speed tests (below approximately 5mph) and then at higher speeds (above approximately 15mph).

To compare the results across the range of curve radii tested, the results from 9mph and 10mph on each curve radius have been plotted onto the graphs below. These are the only speeds at which no derailments occurred in any tests on these curve radii. The graph below (Fig. 5.63) shows the average lateral offset of the wheelset at 9 mph.

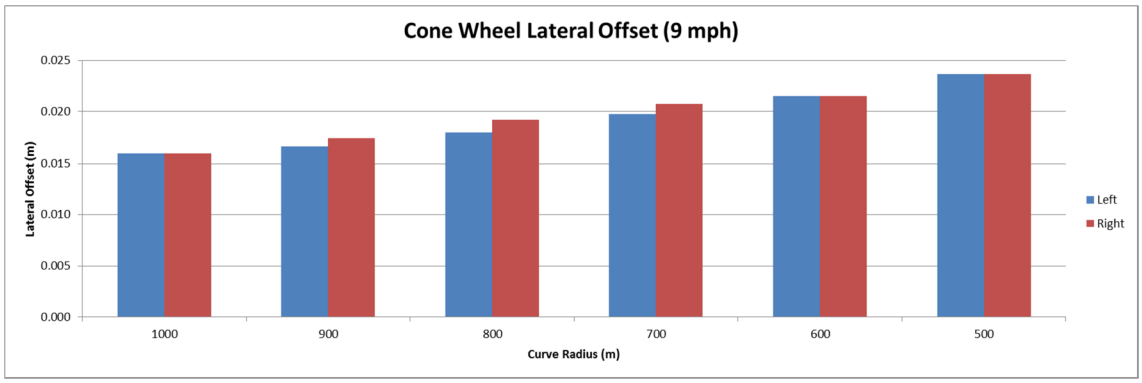


Figure 5.63 - Cone Wheelset Lateral Offset (Left and Right) per Curve Radius (9mph)

The graph below (Fig 5.64) shows the results for a target speed of 10 mph.

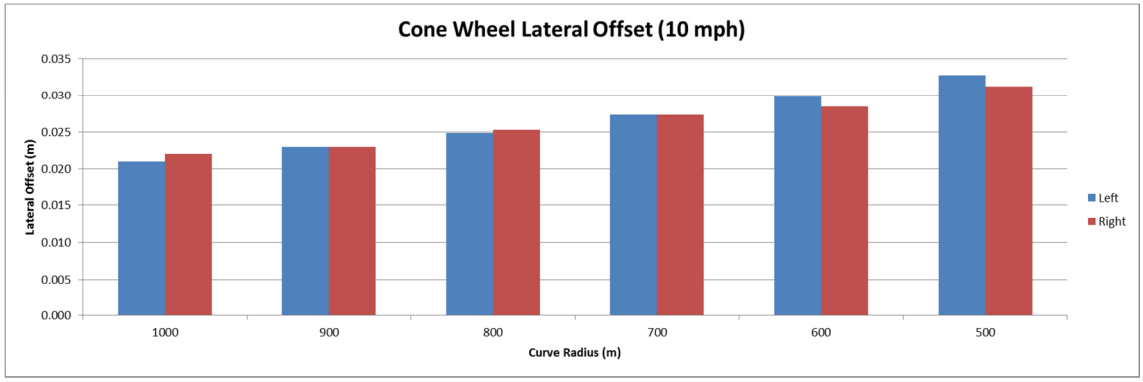


Figure 5.64 - Cone Wheelset Lateral Offset (Left and Right) per Curve Radius (10mph)

This data shows a steady increase in the left (blue) and right (red) offset of the wheelset as the curve radius decreases, as expected. In these tests, the right offset of the wheelset (red) was higher in some tests, but only by 1mm, which is the sensitivity of the simulation (as described in Section 3.3.10).

In order to compare the two sets of tests, the trend in the leftward movement of the wheelset at 9mph and 10mph is shown in the graph below (Figure 5.65 - overleaf). As the data shows, the increase in the leftward offset of the wheelset is similar in both cases, and is higher for the 10mph test, which was conducted at a higher speed than the 9mph tests.

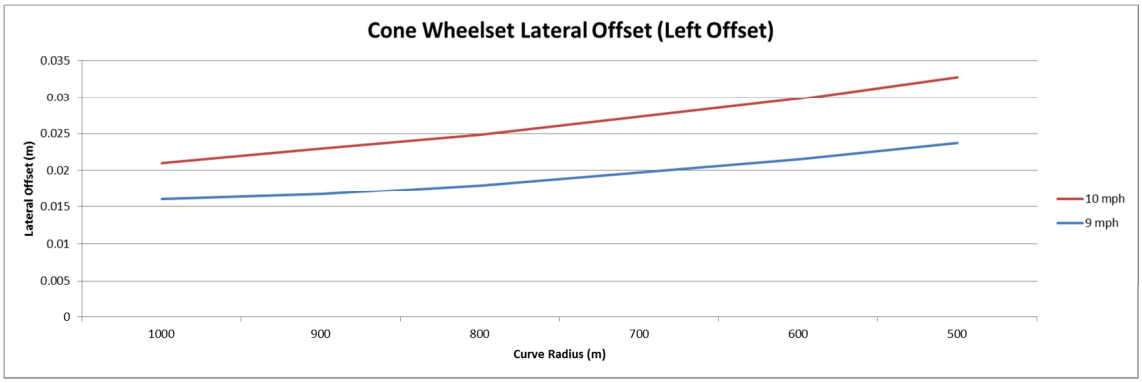


Figure 5.65 - Changes in Leftward Lateral Movement for the Conical Wheelset

5.10.5 *Performance*

The average framerate during these tests was 71.58 FPS. This decrease is due to the increase number of polygons in the scene, as a result of the track sections that make up the looped track. There is some variance in performance between the different curve radii tested, but never by more than +/- 2.4 FPS and not in any consistent way.

5.10.6 *Conclusions*

Many of these results show logically correct behaviours; an increase in lateral offset on narrower curves/at lower speeds. The conical wheelset derails in all tests with the centring force disabled, but does not derail at a range of speeds when the centring force is enabled. This suggests that the centring force does improve the results..

Curiously, the wheelset derails in tests at very low speeds (at 5mph and below on 1,000m radius track) but not and higher speeds. This suggests that the simulation is unstable at very low speeds, supporting results from previous tests, but that this instability is less of an issue when the wheelset is travelling at higher speeds.

5.11 Conclusions from Phase 2 Testing

Tests in this phase were designed to evaluate the ability of the simulation to model the wheel/rail interface. Tests were conducted with individual wheelsets on straight and curved track. The results suggest that the simulation is not capable of simulating the behaviour of a single wheelset correctly. It has therefore not been possible to conduct the curve behaviour tests using the Nadal limit, as was intended.

Stable Speed

A single, SB Wheelset is capable of achieving speeds of 76mph on straight track, if the physics engine parameters are adjusted and the centring force is applied to it, but the wheelset does not corner correctly, even on the widest radius tested (1,000m).

Wheelset Designs

The Multi Body Wheelset is too unstable and cannot be made stable enough to compete with the results for the Single Body Wheelset. The SB Wheelset is used in all future tests.

Polygon Count

The 64 segment wheelset was chosen as it produced the most stable results with only a small decrease in performance. The 64 Segment Wheelset is used in all future tests.

PhysX Parameters

The best results were achieved with the following physics engine parameters:

- Timing Multiple (TM): 20
- Solver Iteration Count (SIC): 184
- Max Angular Velocity (MAV): 170
- Skin Width: 0.01

These parameters are used in all future tests, unless specified otherwise.

Conical Wheelset Tests

Conical wheelset testing on looped track show that the behaviour is logically correct - sinusoidal movement, with higher offset at higher speeds and on narrower curve radii, as predicted. This shows that the behaviour of the wheelset is logically correct in the simulation, but may not be truly realistic. The data from the conical wheelset confirms that the cornering behaviour is improved by the addition of the gravitation stiffness force.

Other Observations

Data from the conical wheelset tests again suggests that the wheelset is less stable at low speeds (below 5mph), but becomes more stable above these speeds, supporting other results from previous tests.

5.12 Testing Phase 3: Bogie in Motion Tests (Straight Track)

The tests in this section were designed to evaluate the behaviour of a bogie in motion. The tests in the previous section managed to achieve a straight line speed of 76, but a single wheelset was highly unstable on curved track. It was hoped that the additional mass of the bogie, coupled with the way that the two wheelsets are joined together by the rigid h-shaped bogie frame, which limits each wheelset's degrees of freedom, might improve the stability of the vehicle.

The following tests were conducted with a bogie accelerating along a straight track to test its stability and top speed. The physics engine parameters were also adjusted to determine whether it was possible to further improve the results.

5.12.1 *Aim*

The aim of these tests is to study the behaviour of a rail bogie in the simulation tool and to maximise its top speed and stability. If the bogie not able to achieve a sufficiently high speed or is too unstable, then the previously identified physics engine parameters will be adjusted in an attempt to improve the results.

5.12.2 *Test Design*

The bogie was tested at a range of speeds on a 2km straight track to determine both its derailment speed and stable speed.

5.12.3 *Initial Testing*

First, tests are conducted using the physics engine parameters that produced the best results in the previous tests. (TM 20, SIC 184). Tests are conducted with and without the application of the centring force.

Results - Without Centring Force

The first test is conducted without the centring force, and Figure 5.66 (below) shows the derailments per batch for each target speed. Here, the stable speed was 77mph.

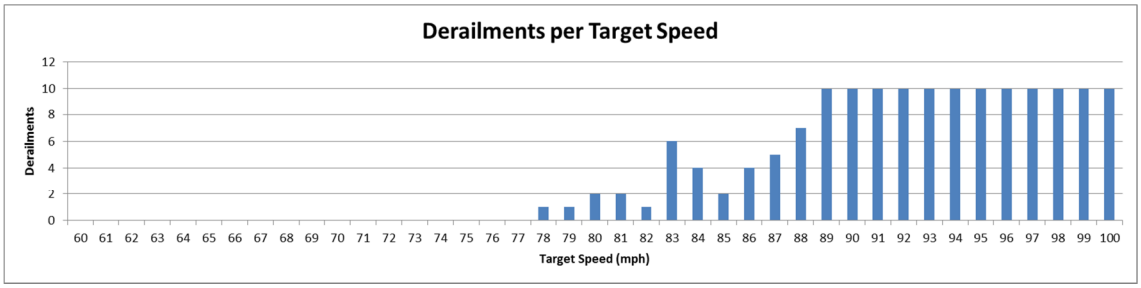


Figure 5.66 - Initial Stable Speed Test (Bogie, no centring force)

Results - With Centring Force

The test was then repeated with the addition of the centring force. Here the stable speed increased to 88mph, as shown in the graph below (Figure 5.67).

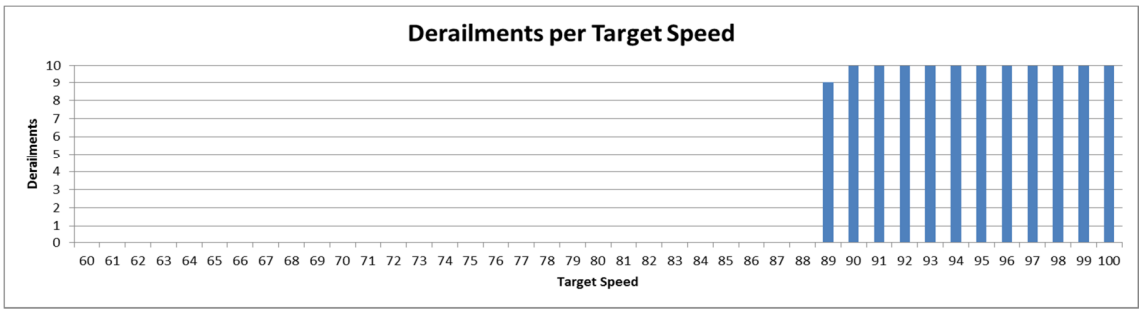


Figure 5.67 - Initial Stable Speed Tests (Bogie, with Centring Force).

Performance

Below is a comparison of the average framerate between the two test batches.

- Without Centring Force: 92.44 FPS
- With Centring Force: 88.29 FPS

A difference of 4.15 FPS, again suggesting that the calculation and application of the centring force does have a negative impact on the performance of the simulation.

5.12.4 *Additional Physics Engine Parameter Testing*

Further testing was conducted to see if the top speed of the bogie or performance of the simulation could be further improved by adjusting the parameters of the physics engine. Tests were conducted across a range of Timing Multiple and SIC values. Sample results from a range of tests are included below (Figures 5.68, 5.69 and 5.70).

TM: 18 - SIC: 64

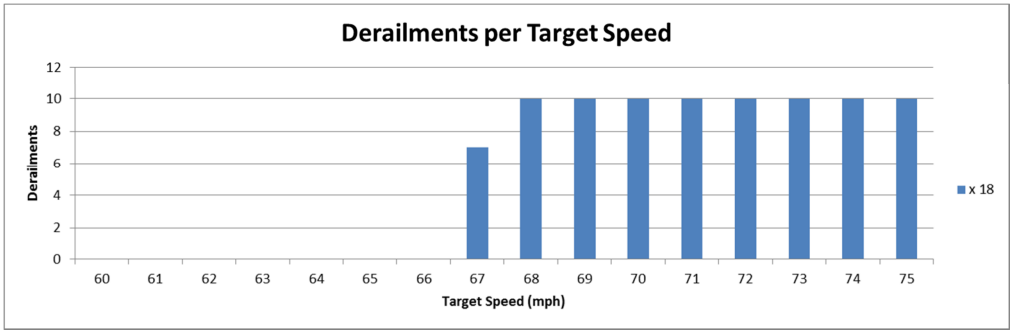


Figure 5.68 - Graph of Derailments per Target Speed (TM18 SIC64)

- Stable speed: 66 mph
- Performance: 106.20 FPS

TM: 20 - SIC: 124

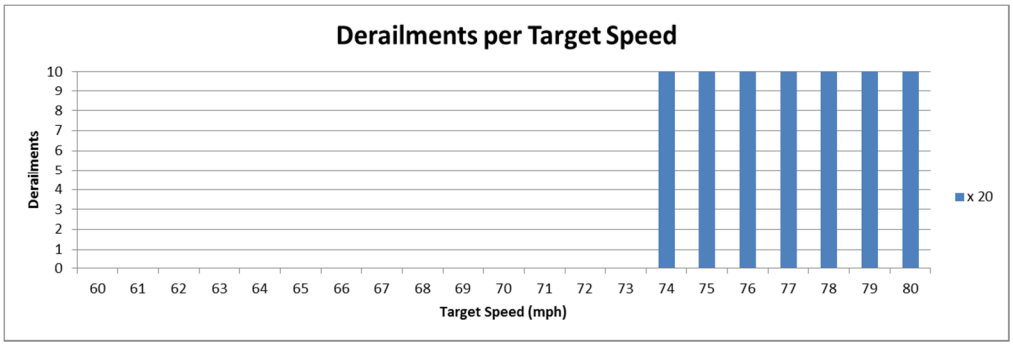


Figure 5.69 - Graph of Derailments per Target Speed (TM20 SIC14)

- Stable speed: 73 mph
- Performance: 100.75 FPS

TM: 22 - SIC: 249

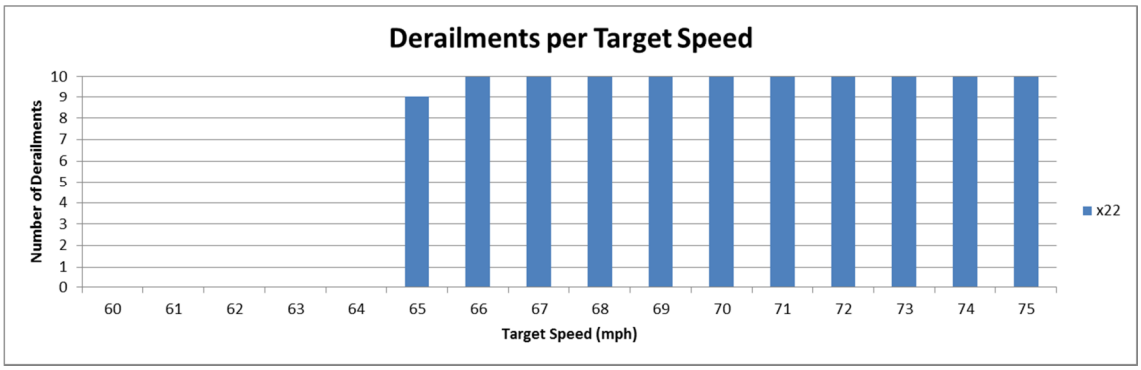


Figure 5.70 - Derailments per Target Speed (TM22 SIC249)

- Stable Speed: 64 mph
- Performance: 76.33

Further Results

Below (Figure 5.71) are the stable speed graphs for the full range of SIC values at Timing Multiple values of 20, which produced the best results in the wheelset tests, as well as TM values of 19 and 21 (one above and one below the previous best results).

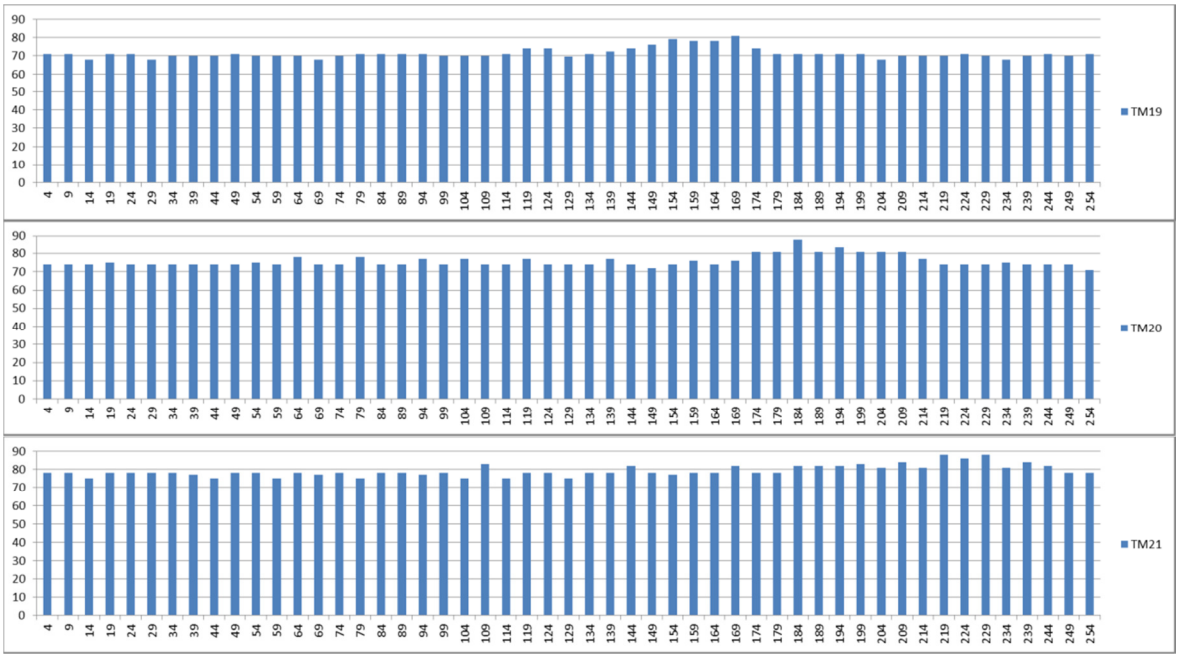


Figure 5.71 - Stable Speeds per SIC for TM Values of 19 (top) 20 (mid) and 21 (bottom)

These results are a lot more consistent (less varied) than the wheelset tests, with a much smaller range of results.

Table 5.30, below, shows a summary of the results; the min and max stable speeds achieved for each value of timing multiple.

	Stable Speeds		
TM	**Min**	**Max**	**Range**
19	68	81	13
20	71	88	17
21	75	88	13

Table 5.30 - Min, Max and Range of Stable Speeds for SIC Testing at TM of 19, 20 and 21

There are two results, at TM21 SIC219 and TM21 SIC229, that produce the same maximum stable speed of 88 mph, but no results are higher. The framerate of the TM21 SIC219 tests was 84.64, which was lower than the 88.29 FPS achieved with TM20 SIC184. Other values tested (TM 18 and below, TM 22 and above) do not produce the same peak speed of 88mph achieved with the parameters described above.

Since the other values that achieve the same top speed produce a drop in framerate without an actual, noticeable improvement in results, the value of TM 20 and SIC 184 will continue to be used in future tests.

Conclusions

It is possible that further adjustments to these physics engine parameters might improve the results, but the data collected during the tests presented in this section suggests that the previously suggested parameters continue to produce the best results. These parameters are:

- Timing Multiple (TM): 20
- Solver Iteration Count (SIC): 184

These produced a stable speed for a bogie using single body wheelsets of 88mph, and a peak speed of 91.03mph. These values are used in all tests in the rest of this Thesis.

5.12.5 *Joint Solver Extrapolation Factor*

Revolute Joints, such as the joint between the wheelset and the bogie, have a Solver Extrapolation Factor (SEF) that defines the rigidity of the joint. This parameter is discussed in more detail in Section 3.4.7. The following tests are conducted to discover the effect of altering this parameter on the stability of the bogie.

Test Design

The solver extrapolation factor was incremented from its minimum value (0.5) to its maximum value (2.0). The tests are conducted with a bogie on a straight track.

Predictions

It was theorised that adjusting this parameter might affect the stability of the bogie - as it affects the joints between the bogie and the wheelsets, possibly improving the results and allowing the bogie to reach a higher stable speed. Adjusting the SEF is not expected to affect the performance of the simulation.

Stable Speed Tests

The following tests were conducted to determine the stable speed of the vehicle at each SEF value, in order to determine what effect the rigidity of the joints has on the stability of the vehicle. The results are shown in Table 5.31, below.

Joint SEF Value	0.5	0.6	0.7	0.8	0.9	1.0	...
Stable Speed	0	0	0	0	0	88	...

Table 5.31 - Stable Speed per Joint SEF Value (Bogie - Straight)

The stable speed during each test at an SEF of 0.9 and below was reported as 0, because the wheelset derailed in all tests at 1mph. The graphs below (Fig. 5.72) show the derailments per target speed in tests the between a SEF of 0.5 and 1.0,k at speeds between 1 and 10 mph.

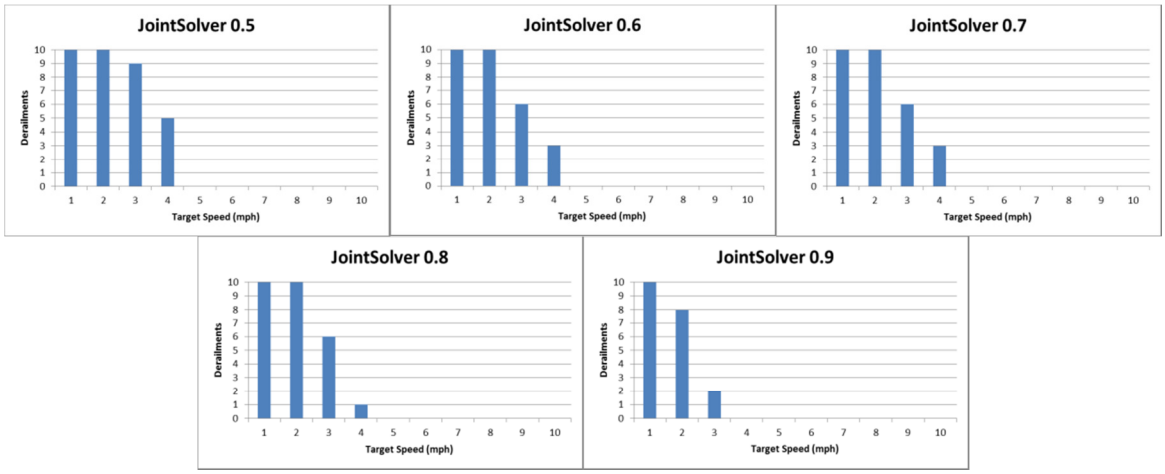


Figure 5.72 - Graphs of low speed derailments at a range of Joint Solver Extrapolation Factor Values

Although the bogie derails at lower speeds (below approximately 4mph), it does not derail in higher speed tests (5 mph and over). This data would seem to corroborate previous test data suggesting that the vehicle is capable of accelerating up to higher speeds in a stable fashion, but is unstable when forced to maintain lower speeds. This behaviour is undesirable, as the aim is to ensure consistent, stable behaviour across as high a range of speeds as possible. In these tests there were derailments at low speeds < 5mph, but no derailments at higher speeds.

There are more derailments as the Joint Solver Value decreases too, as shown in Table 5.32, below.

Solver Value	*0.9*	*0.8*	*0.7*	*0.6*	*0.5*
Total Derailments	20	27	29	29	34

Table 5.32 - Total Derailments per Solver Value

This data suggests that decreasing the SEF reduces the stability of the vehicle. These low speed derailments do not occur with a Solver value of 1.0 or above. Table 5.33 (below) shows the stable speeds for an SEF of 1.0 and 2.0.

Joint Solver Value	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
Stable Speed	88	88	88	88	88	88	88	88	88	88	88

Table 5.33 - Stable Speed per Joint Solver Extrapolation Factor Value

In these tests, the low speed derailments do not occur, but increasing the SEF value does not appear have any positive effect on the stable speed of the vehicle.

Lateral Offset

The graph below (Fig. 5.73) shows how the lateral offset of the front wheelset of the bogie changes as the SEF value is adjusted.

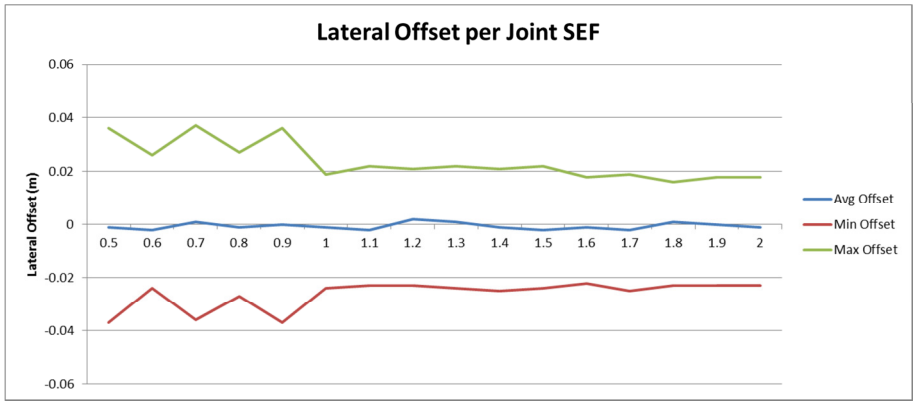


Figure 5.73 - Graph of Lateral Offset per Joint Solver Extrapolation Factor (Bogie 10mph)

From the default value of 1.0 up to the maximum value of 2.0, the results are fairly consistent, with no considerable change in the min, max or average lateral offset of the wheelset. Similar results were collected for the rear wheelset also.

However, with an SEF value of below 1.0, the results become less consistent, and the min and max offset become larger. This data suggests that the wheelset is less stable below the default value of 1.0, which concurs with the derailment results.

This would seem to suggest that there is little value to increasing the value of this parameter, but that it should not be decreased below its default value, as this makes the bogie less stable. The SEF will therefore be kept at its default value of 1.0.

Performance

It was not expected that altering the SEF value would affect simulation performance. The graph below (Fig. 5.74) shows the change in framerate over the course of the tests.

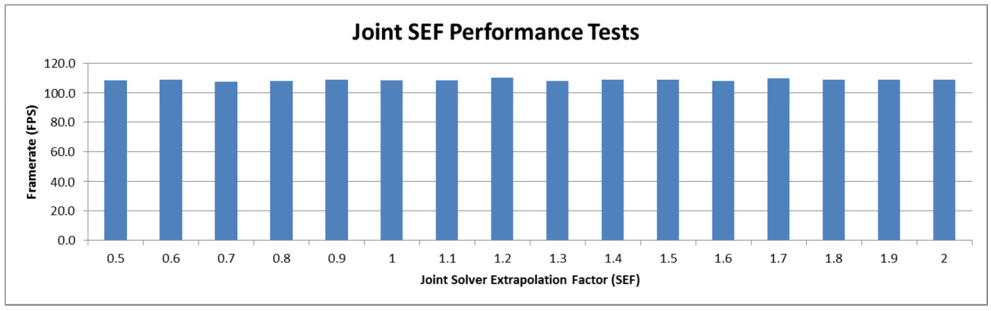


Figure 5.74 - Graph of results from performance testing (Bogie, SEF)

The framerate varied between 107.8 and 110.2 fps, but not in any consistent way, suggesting that changing the SEF had little or no effect on the performance of the simulation tool. The best results, attained with the parameters above, produced the following performance results (with centring force applied).

- Bogie: 109.47 FPS

This is 57.43 FPS lower than tests with a single wheelset using the same settings, but the decrease in performance is expected given that there are more rigid bodies in the scene. The performance is still well above the real-time target of 60 FPS.

5.12.6 *Conclusions*

The results presented in this section confirm the hypothesis that a bogie, where two wheelsets are joined by a rigid frame, is more stable than a single wheelset. The results are more consistent and the stable speed of the bogie is higher than that of a single wheelset. Adjusting the Joint Solver Extrapolation Factor parameter had no effect on the performance of the tool, nor did it produce any improvement to the stable speed.

The highest speed achieved by the bogie was 88 mph, using the following parameters.

- TM: 20
- SIC: 184
- SEF: 1.0

Altering these parameters produces no further benefit to the stability of the simulation.

5.13 Bogie in Motion Tests (Curved Track)

The tests in this section were designed to evaluate the cornering behaviour of the bogie.

5.13.1 *Test Design*

The following tests were conducted on a looped track with a range of track radii from 100m to 1,000m. The tests are conducted using the physics engine parameters identified in Section 5.12, with the spline-based centring force applied.

Curve Radii

The following curve radii are tested, though the limited top speed of the bogie limits the potential testing possibilities. Testing is divided into two parts:

- *Smaller Curve Radii*

Results collected using the following curve radii can be evaluated against the Nadal derailment predictions.

- 100m, 125m, 150m, 175m, 200m

- *Larger Curve Radii*

The other radii that were constructed for use in the simulation were also tested, but it is not possible to use the Nadal Limit predictions due to the limited maximum stable speed of the bogie. These are as follows.

- 300m, 400m, 500m, 600m, 700m, 800m, 900m, 1000m

Derailment Speed Tests

In these tests, the vehicle starts at a speed of 1mph. When it reaches each straight section of the looped track, its target speed is increased by 1mph. If it reaches 88mph and completes a loop of the track without derailing, then the test ends with no derailments. Otherwise the derailment speed is recorded, as in previous tests.

Stable Speed Tests

The stable speed tests work in the same way as the stable speed tests in previous sections. The bogie is tested up to (and including) its maximum stable speed on straight track; 88mph. The highest speed at which no derailments occur is the ‘stable speed’.

5.13.2 *Predictions*

For smaller radii, the predicted derailment speeds for these curve radii is below the maximum stable speed of the bogie (88mph). In these tests, the bogie should derail at speeds close to those predicted by the Nadal Limit.

These predicted speeds are shown in Table 5.34, below.

Curve Radius	*100*	*125*	*150*	*175*	*200*
Predicted Derailment Speed (mph)	58.96	65.92	72.21	78.00	83.38

Table 5.34 - Predicted Derailments for Curve Radii 100 - 300m (from Section 5.5)

The predicted derailment speeds for the larger curve radii (see Section 5.5) are above 88mph, which is the maximum stable speed achieved by the bogie in the previous tests. On these loops, the bogie should be able to accelerate up to its maximum stable speed of 88mph, without derailing, if the simulation is sufficiently stable and accurate.

5.13.3 *Initial Testing*

This section presents results from initial bogie testing.

Without Centring Force

Tests were first conducted with the bogie without the application of additional centring forces. In these tests, the bogie derailed in all tests, at all speeds on all curve radii. The remaining tests in this section were conducted with the centring force applied.

Wider Radii

The following tests were conducted on radii of 300 to 1,000m. It was expected that the bogie would be able to accelerate to its previously observed stable speed of 88mph on these tracks without any derailments occurring.

- *1,000m and 900m*

The following results (shown in Table 5.35, below) were taken from 1,00m and 900m testing, the widest radii tested:

Curve Radius	*900*	*1,000*
Number of Derailments	0	0

Table 5.35 - Derailments per Curve Radius (at 88mph)

The bogie successfully accelerated up to 88mph without derailing on both curve radii. This was the expected result. These results show that the application of the centring force has a positive effect on the cornering behaviour of the bogie, which derailed in all tests when the force was not applied.

- *300m to 800m*

The graph in Figure 5.75 (overleaf) shows the derailment speeds for curve radii of 300m to 800m. The graph shows the predicted derailment speed (blue), the peak/derailment speed (red) and the maximum stable speed (green).

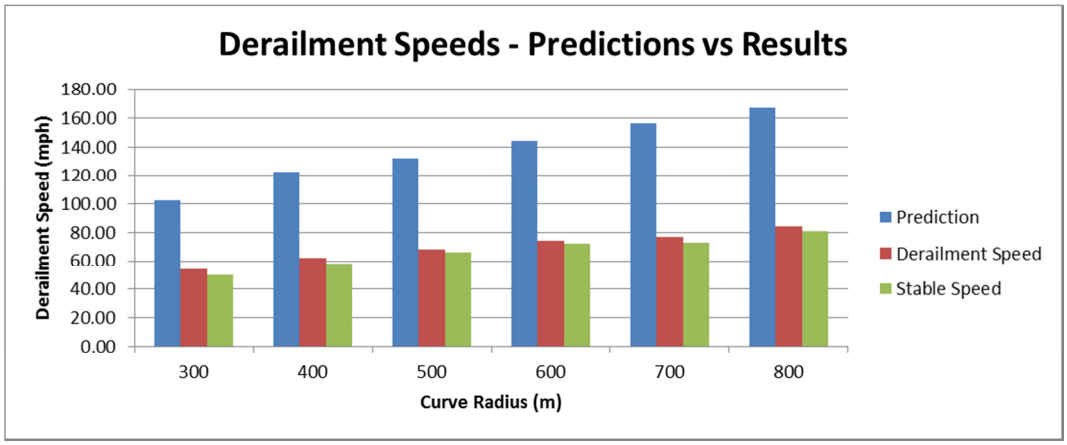


Figure 5.75 - Derailment Speeds - Prediction vs Results (300m to 800m radius) Bogie Initial Testing

These results are also shown in Table 5.36, below, along with the predictions and difference between prediction and results.

Curve Radius (m)	*300*	*400*	*500*	*600*	*700*	*800*
Prediction (mph)	102.12	122.27	131.84	144.42	156.00	166.77
Derail Speed (mph)	53.80	62.25	68.29	74.16	77.06	84.01
Difference (%)	47.3%	49.1%	48.2%	48.6%	50.6%	49.6%
Stable Speed (mph)	50	58	66	72	73	81
Difference (%)	51.0%	52.6%	49.9%	50.1%	53.2%	51.4%

Table 5.36 - Derailment Speed per Curve Radius (Bogie, 300m to 800m)

The results were between 49.4% and 52.7% below the prediction for each of these curve radii. The stable speed in each test was between 2.16 and 4.25mph below the derailment speed.

If the average is taken across the whole range of results:

- The derailment speeds were, on average, 48.9% below the predictions.
- The stable speeds were, on average, 51.4% below the predictions.

Smaller Radii

The graph overleaf (Figure 5.76) shows the results of tests conducted on radii of 100 to 200m. These are the speeds at which it is expected that Nadal limit predictions can be applied. The derailment speeds on each curve radius were significantly lower than the predictions. The stable speeds were lower still.

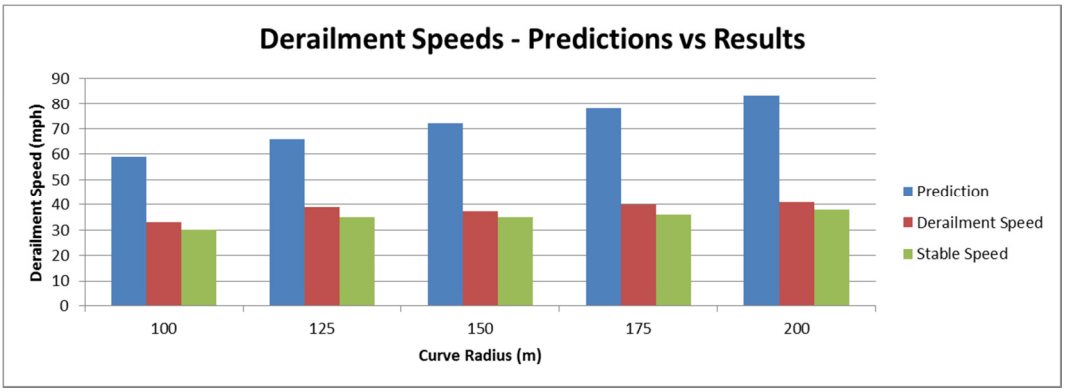


Figure 5.76 - Derailment Speeds - Prediction vs Results (Bogie, 100 - 200m radius)

The average derailment speed for each radius is shown in Table 5.37, below, alongside the Nadal prediction.

Radius	*100*	*125*	*150*	*175*	*200*
Prediction (mph)	58.96	65.92	72.21	78	83.38
Derail Speed (mph)	33.15	38.86	37.26	39.77	40.97
Difference (%)	43.8%	41.0%	48.4%	49.0%	50.9%
Stable Speeds (mph)	30.00	35.00	35.00	36.00	38.00
Difference (%)	49.1%	46.9%	51.5%	53.8%	54.4%

Table 5.37 - Derailment Speeds versus Nadal Predictions

The results are higher on wider radii, as expected, but are still lower than the Nadal Predictions. The best result was 59% of the predicted speed and the worst result was 49.1% of the predicted speed.

- The derailment speeds were, on average, 46.6% below the predictions.
- The stable speeds were, on average, 51.2% below the predictions

Performance

The following tests were conducted on the 1,000m radius track. The bogie was tested with and without the application of the centring force. Without the centring force, the bogie derailed in all tests. The average framerate during each batch is as follows:

- Without Centring Force applied: 70.03 FPS
- With centring force applied: 65.02 FPS

There is a difference of approximately 5 FPS, suggesting that the calculation and application of the centring force to the wheelsets comes at a performance cost.

Summary

The centring force has improved the cornering ability of the bogie, but the bogie still does not approach the derailment speeds predicted by the Nadal Limit.

5.13.4 *Multiplied Force Results*

The GSF significantly improved the cornering behaviour of the bogie, but was not enough to produce results close to the Nadal predictions. It was theorised that increasing the size of this force would lead to improved results, and so the following tests were conducted to investigate this possibility. The implications of this are discussed at the end of this section.

Force Multiple

In order to test this hypothesis, a 'force multiple' was added to the code at the point where the centring force is calculated and applied. The value of Y_w (line 7 in Figure 4.8) was multiplied by *force multiple* before it was used to calculated the force vector, as illustrated in Figure 5.77, below.

```
7   float Lw = ((W * lambda) / l0) * forceMultiple;
8
9   Vector3 force = offset * Lw;
10  contact->wheelset->ApplyForce(force);
```

Figure 5.77 - Centring code with Force Multiple

Test Design

The force multiple will be increased from its initial value of 1 in increments of 1 up to a maximum of 30. The stable speed and derailment speeds of the vehicle, as described earlier, are recorded and compared to the previous results.

Predictions

It is estimated that increasing the size of the force will increase the derailment speed/stable speed of the bogie. However, it is possible that increasing it above a certain value will make the simulation unstable.

Initial Force Multiple Testing

In order to evaluate the effect of increasing the magnitude of the centring force, the value of *force multiple* was incremented between 1 and 30.

- *Force Multiple Testing (1 to 30)*

The bogie was tested with a range of force multiples across the range of track radii (100m to 200m). In each test, the percentage difference between the predicted results and the average derailment speed of the vehicle was measured, and the average percentage difference at each multiple is shown in Figure 5.78 (overleaf).

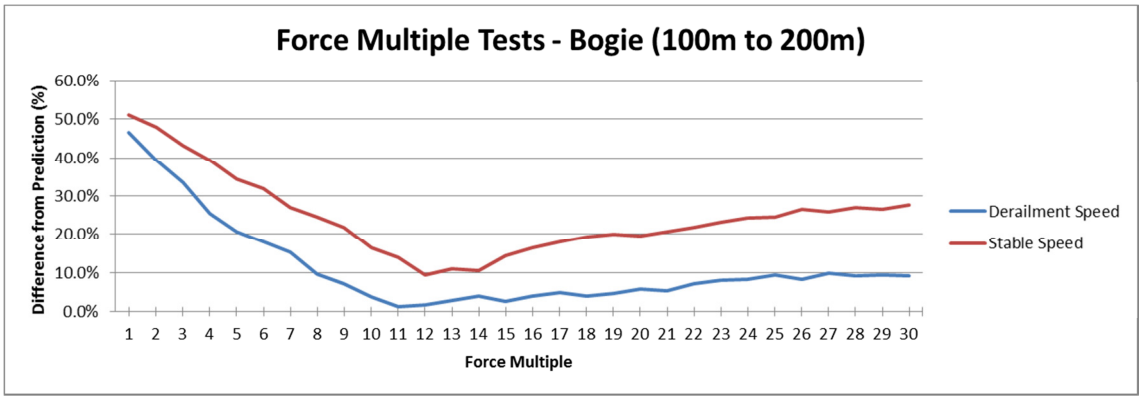


Figure 5.78 - Average difference between predicted and recorded derailment speeds

The average difference between each set of results and the predictions decreases as the test multiple increases, up until a point at which it becomes less stable again.

- The best result for the derailment speed tests was achieved with a force multiple of 11, where the average speed was, on average, 1.2% away from the predictions.
- The best result for the stable speed tests was achieved with a force multiple of 12, where the stable speed was, on average, 9.26% away from the predictions.

A *force multiple* of 11 or 12 would appear to be the best results because the derailment speeds are, on average, closest to the predicted derailment speeds.

➢ *Force Multiple 11*

The derailment and stable speed tests were repeated using a force multiple of 11. The results of the derailment and stable speed tests compared to the Nadal predictions are shown in Figure 5.79, below.

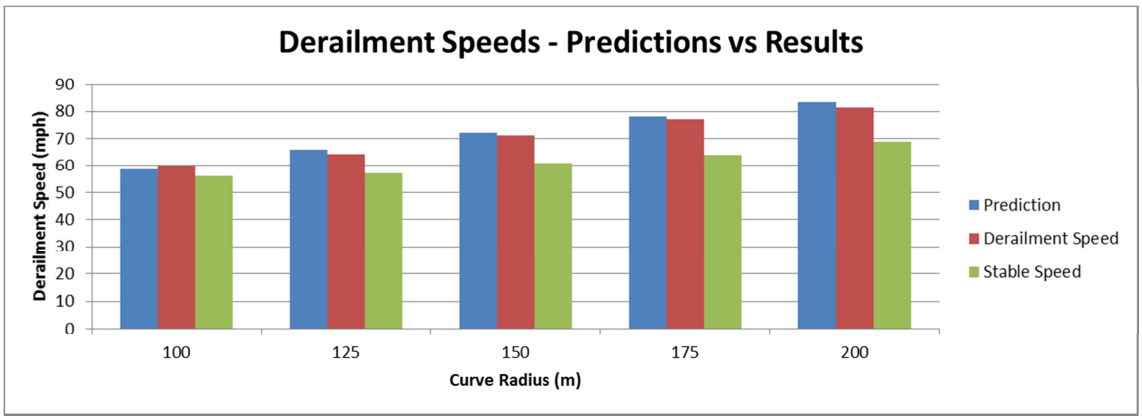


Figure 5.79 - Predicted Derailment Speeds versus recorded results

The graph shows that the derailment speeds (red) are close to the predicted speeds (blue) on each curve radius. The stable speed results (green) are lower than the derailment speeds across all curve radii. Both the derailment speeds and stable speeds increase with curve radius, which is logically correct.

Table 5.38, below, shows the predicted derailment speed, the recorded derailment speed results and the differences for each curve radius.

Curve Radius	*100m*	*125m*	*150m*	*175m*	*200m*
Prediction	58.96	65.92	72.21	78	83.38
Derailment Speed	60.1	64.2	71.22	76.95	81.22
Difference	-1.14	1.72	0.99	1.05	2.16
%	-1.9%	2.6%	1.4%	1.3%	2.6%
Range	14.35	15.65	19.30	16.91	17.23
Standard Deviation	2.64	4.18	3.72	3.12	3.27

Table 5.38 - Differences between Prediction and Derailment Speed Results - Force Multiple 11

All of the derailments were within 2.16mph of the Nadal prediction across this range of radii and were, on average, off by 2.2%. The range of results is quite high, but the standard deviation of the results is, on average, 3.39mph.

Table 5.39, below, shows the prediction, along with the stable speed results and differences for each curve radius.

Curve Radius	*100m*	*125m*	*150m*	*175m*	*200m*
Prediction	58.96	65.92	72.21	78	83.38
Stable Speed	56.00	57.00	61.00	64.00	69.00
Difference	2.96	8.92	11.21	14.00	14.38
%	5.0%	13.5%	15.5%	17.9%	17.2%

Table 5.39 - Differences between Prediction and Stable Speed Results - Force Multiple 11

The stable speed was, on average, 13.85% below the predicted results, and as much as 17.2% below the prediction.

- *Force Multiple 12*

The tests were repeated 10 times on each radius using a force multiple of 12. The results of the derailment and stable speed tests compared to the Nadal predictions are shown in Figure 5.80 (overleaf).

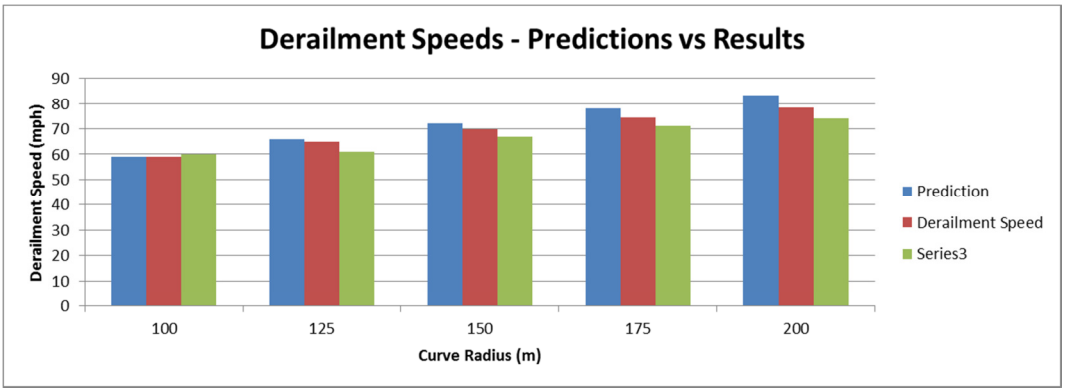


Figure 5.80 - Predicted Derailment Speeds versus recorded results

Table 5.40 (below) shows the prediction, derailment speed results and differences for each curve radius.

Curve Radius	*100m*	*125m*	*150m*	*175m*	*200m*
Prediction	58.96	65.92	72.21	78	83.38
Average Results	61.1	65.2	70.22	74.95	79.22
Difference	-2.14	0.72	1.99	3.05	4.16
%	-3.6%	1.1%	2.8%	3.9%	5.0%
Range	14.35	15.65	19.30	16.91	17.23
Standard Deviation	2.68	4.28	5.75	3.12	3.46

Table 5.40 - Differences between Prediction and Results

All of the results are within 4.16mph of the Nadal prediction across this range of radii and on average were off by 1.84%. The average standard deviation is 3.86mph.

Table 5.41 (below) shows the prediction, stable speed results and differences for each curve radius.

	100m	*125m*	*150m*	*175m*	*200m*
Prediction	58.96	65.92	72.21	78	83.38
Stable Speed	57.00	60.00	65.00	69.00	73.00
Difference	1.96	5.92	7.21	9.00	10.38
%	3.3%	9.0%	10.0%	11.5%	12.4%

Table 5.41 - Differences between Prediction and Stable Speed Results - Force Multiple 11

The stable speed was, on average, 9.26% below the predicted results, and as much as 12.4% below the prediction.

➢ *Wider Radii*

The bogie was tested on curve radii from 300m to 1,000m using a Force Multiple of 11. The expectation was that the bogie would not derail at speeds up to and including 88mph on any of these radii. The results are shown in Table 5.42 (below).

Curve Radius (m)	*300*	*400*	*500*	*600*	*700*	*800*	*900*	*1,000*
Stable Speed (mph)	88	88	88	88	88	88	88	88

Table 5.42 - Stable Speed for each Curve Radius (300m - 1,000m)

The results were the same for a Force Multiple of 12. This was the expected result for these curve radii; a stable speed equal to the maximum stable speed of the bogie.

➢ *Straight Track*

The following tests were conducted on the 2km straight track. Figure 5.81 (below) shows the average derailment speeds from testing across a range of force multiples.

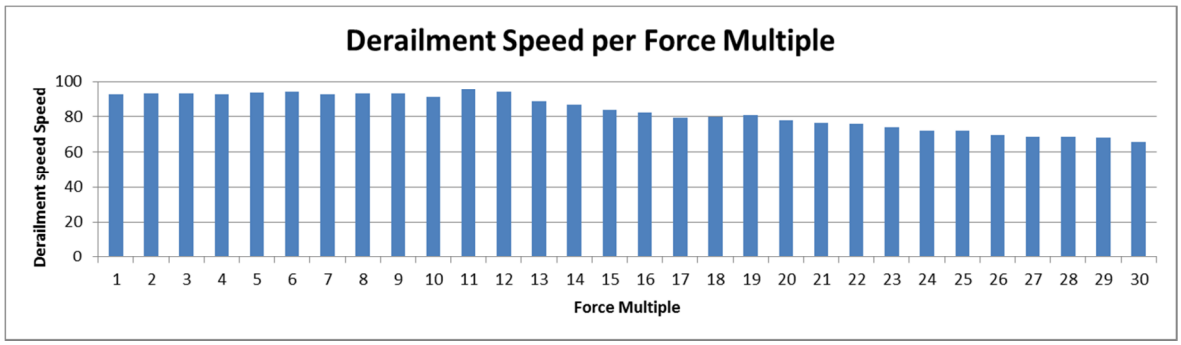


Figure 5.81 - Average derailment speeds for the bogie across a range of force multiples

Figure 5.82 (below) shows the stable speed of the bogie.

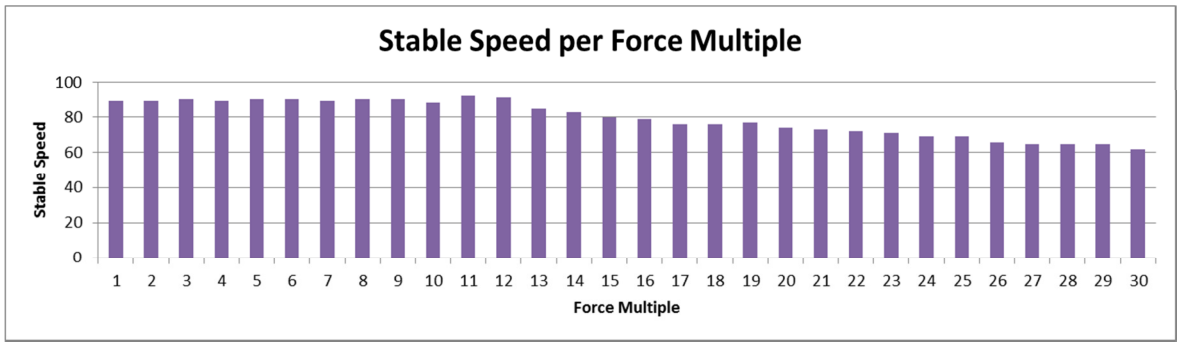


Figure 5.82 - Stable Speeds for the bogie a range of force multiples

The best results from this data compared to the best un-multiplied results are shown below:

- The maximum derailment speed of the vehicle was 95.23mph, achieved with a force multiple of 11, an increase of 4.2mph.
- The maximum stable speed of the vehicle was 91 mph, achieved with a force multiple of 11 and 12, an increase of 3mph.

This data suggests that it was not just the lack of a correct centring wheelset centring behaviour that was limiting the top speed of the bogie. Another issue (or issues) may be causing the instability that prevents the vehicle reaching any higher speeds, perhaps vibrations caused by the polygonal shape of the wheels, or errors the way in which collisions and joints between rigid bodies are handled by the solver.

Performance

There is some variation in performance between tests on different curve radii, but not in any way consistent with the radius currently being tested. The average framerate during these bogie tests was 66.03 FPS, which is just above the real-time target of 60 FPS.

5.13.5 *Conclusions*

The results in this section show that the spline-based technique can be made to produce results that approach the Nadal predictions across this range of curve radii, if a multiplied centring force is applied to the wheels. Using the Gravitational Stiffness Formula, but multiplying the resultant force by a factor of 11, produced the best results across a range of track curves, where the predicted derailment speed is less than the 88mph top stable speed of the vehicle.

These tests also show that it was not just the lack of a realistic self-centring effect of the wheelsets that was preventing them from achieving high speeds, and that other factors are also contributing by causing instability in the wheelsets.

The results could be improved by taking the radius of the curve into account, but this is unlikely to produce a universal solution. A more sophisticated calculation, perhaps considering additional rail industry formulas such as creep forces, lateral velocity of the wheelset etc. might produce better results, and allow this arbitrary multiple value to be removed from the force calculation. Unfortunately, time constraints prevented further evaluation of this hypothesis within the scope of this research project.

5.14 Chapter Summary

In this chapter has shown how the simulation of the wheel/rail interface in the Locomotion tool was evaluated, and how the simulation was iteratively adjusted in order to improve the results and produce more realistic data.

Key Results

Conical wheelset testing showed some logical behaviours; sinusoidal lateral motion, lower offset on higher curve radii and higher offset at higher speeds. This shows that the simulation is capable of simulating some aspects of wheelset centring behaviour, even if it is not producing truly accurate results. The vehicle was unable to reach the target straight line speed of 100mph, instead achieving a stable speed of 88mph. During curved track testing, designed to evaluate the behaviour of the bogie against the Nadal Limit predictions, the results for radii of 100 to 200m were accurate to within 2% of the predictions, with an average standard deviation of less than 4mph. These error bounds should be considered suitably small by engineers for use in a real-time simulation tool.

Design Choices

The Single Body wheelset, with 64 segment cylindrical wheels, was shown to be the best design choice, as it was significantly more stable than the Multi Body wheelset. A single wheelset cannot traverse a curve successfully, which meant that it was not possible to use the Nadal Limit benchmarks to evaluate wheelset cornering behaviour as intended.

Improving the Simulation of the Wheel/Rail Interface

It is necessary to multiply the centring force by 11 to produce the most realistic results across the range of curve radii tested. This suggests that there is error in the centring behaviour of the wheelset in the PhysX engine. It is possible that this Force Multiple can be eliminated by fine-tuning the centring force, taking additional properties of the wheels and rails, or additional forces into account. This is discussed in more detail in Chapter 7.

Physics Engine Parameters

The best results were achieved with the following physics engine parameters:

- Timing Multiple: 20
- Solver Iteration Count: 184
- Joint Solver Extrapolation Factor: 1.0
- Skin Width: 0.01

Performance

During the final set of tests, the performance of the simulation tool was 66.03 FPS, which is above the real-time target of 60 FPS.

Chapter 6

Additional Testing and Sample Data

The data presented in this chapter is intended to illustrate the capabilities of the Locomotion simulation tool; the kinds of tests that can be conducted and the data that it is capable of producing. This includes sample data from stability and derailment tests on full vehicles and multi-vehicle trains, as well as testing designed to evaluate the suitability of the tool for use in rapid prototyping and as a gauging tool.

It has not been possible to fully validate the results presented in this chapter, but it is presented based on the assumption that vehicle behaviour is reasonably realistic, based on the results from the previous chapter.

6.1 Overview

This testing presented in this chapter includes the following.

- Section 6.3 presents the results of tests conducted on multiple deployment platforms, designed to evaluate the consistency of the simulation tool.
- Section 6.4 presents the results of tests conducted with a full vehicle.
- Section 6.5 presents the results of tests conducted with multi-vehicle trains.
- Section 6.6 presents the results of initial testing designed to evaluate the suitability of the simulation tool for rapid prototype testing.
- Section 6.7 presents the results of initial testing designed to evaluate the suitability of the tool for gauge testing.

6.2 Parameters

The tests were conducted using the same parameters as the previous tests:

- Timing Multiple: 20
- Rigid Body Solver Iteration Count: 184
- Skin Width: 0.01
- Joint Solver Value: 1.0
- Force multiple: 11

The Single Body Wheelset was used in these tests and the vehicles were tested on straight track and a range of curve radii.

6.3 Consistency Tests

The following tests were conducted on multiple machines, in order to confirm that the simulation produces consistent results on multiple deployment platforms.

6.3.1 *Computer Specifications*

Below are descriptions of the three computers on which the tests were conducted. Included are their device name and their hardware specifications.

'Zerg09'

This is the name of the machine on which all of the tests Chapter 5 were conducted (for specifications of this machine, see section 5.1.).

'Mal'

A desktop PC:

- *Operating System:* Windows 10 Professional (64-bit)
- *Processor:* INTEL Core i7 CPU 920 @ 2.67GHz
- *Ram:* 12 GB
- *Graphics Card:* NVIDIA GeForce GTX 760

'Kaylee'

A laptop:

- *Operating System:* Windows 10 Home (64-bit)
- *Processor:* INTEL Core i3-4000M CPU @ 2.40GHz
- *RAM:* 8.0 GB
- *Graphics Card:* NVIDIA GeForce GTX 765M

The three machines are similar, in that they all have Windows operating systems, INTEL processors and NVIDIA graphics cards, but there is some variance in the specifications.

6.3.2 *Test Design*

The Locomotion source code was downloaded to and compiled on each of these machines, and then identical tests were conducted on each machine using the Visual Studio IDE to determine whether the simulation produces the same results on each deployment platform, and how the performance of the simulation differed between them.

The stable speed tests were conducted using a rail bogie on a 2km straight track, using the 'ideal' parameters described at the start of this chapter and with the centring force applied to the wheelsets.

6.3.3 *Predictions*

It is hoped that the simulation results (stable speed, derailment speed, range and standard deviation of results) will be very similar across each machine. It is possible that there will be some minor variation due to the error introduced by a real-time system, but ideally the results will be identical.

It is expected that Zerg09, the most powerful machine, will produce the highest framerate and that Kaylee, the least powerful, will produce the lowest framerate. It is not expected that the differences in framerate will correlate with any variance in the results.

6.3.4 *Results*

Table 6.1 (below) shows the stable speed, derailment speed, range of (derailment speed) results and the framerate recording during stable speed testing on each machine.

Computer	Stable Speed	89mph Derailments	Derailment Speed	Range of Results	Standard Deviation	Framerate
Zerg09	88	9	92.271	13.74	3.77	66.0
Mal	88	9	91.033	14.57	4.67	51.6
Kaylee	88	9	91.033	14.57	4.67	30.8

Table 6.1 - Consistency Test results from the three machines tested

The stable speeds were the same, and there were the same number of derailments at 89mph, on each machine. The derailment speeds and range of results for Mal and Kaylee were identical, while the derailment speed for Zerg09 was 1.2mph higher and the range of results 0.83mph lower. The biggest difference is in framerate, with Zerg09 executing 14.4 FPS faster than Mal and 35.2 FPS faster than Kaylee, as expected

6.3.5 *Conclusions*

There is a small amount of variance in a few of the results, but the results are mostly consistent across all three machines. The stable speed results are identical, and there are variations of approximately 1mph between the derailment speed results on Zerg09 and the other two machines. The performance of the simulation varies depending on which computer it is executing on, but this does not seem to have any affect the physical results; the results on Kaylee and Zerg09 were the same, despite the framerate of the tests conducted on Kaylee being less than half that of those conducted on Zerg09.

The main difference between Mal/Kaylee and Zerg09 are the operating systems and the generation of graphics cards. The data collected here suggests that one of these factors may be the reason for the minor variation in results, but further testing would be necessary to confirm this hypothesis.

6.4 Locomotive Tests

The following data is from tests on a locomotive. The vehicle has been constructed based on the Metro de Madrid 5000 series vehicle, using the properties defined in Table 3.3.

6.4.1 *Static Tests*

The following tests were conducted to determine if the mass of the vehicle was being correctly distributed, and the normal forces were being correctly applied, to each of the wheelsets.

Test Design

As with the bogie testing in Chapter 5, Phase 3, the Locomotive is placed on a straight track, its target speed is set to 0mph and the forces acting on each wheelset of each bogie of the vehicle are measured, until the physics engine puts the objects to rest.

Predictions

The effective mass of the locomotive acting on each wheelset is 8,000kg, and so the predicted magnitude of the normal force is calculated thus:

$$8{,}000 * 9.806 = \mathbf{78{,}448}\boldsymbol{N}$$

Results

The following data (Table 6.2) was collected from tests on a vehicle built using Single Body Wheelsets.

- 87 samples were collected per wheelset.

Wheelset	Magnitude	%	Range
Front Bogie - Front	80,228.07	-2.27%	6068.43
Front Bogie - Rear	76,652.62	2.29%	7699.871
Rear Bogie - Front	79,931.23	-1.89%	7268.68
Rear-Bogie - Rear	77,180.18	1.62%	5437.531
Average	78,498.03	-0.06%	6,618.63

Table 6.2 - Single Body Wheelset Locomotive Rails Tests

There is a variance of 3,047.90 between the front and rear wheelset of the front bogie and 3,278.61 between the front and rear wheelset of the rear bogie. The average result across all four wheels was just 0.06% above the predicted value. The average range of results was 8.44% of the magnitude of the force and the average standard deviation of the results was just 1.33%.

6.4.2 *Locomotive in Motion (Straight Track)*

The following tests were conducted with a single Locomotive on straight track.

Design

A single locomotive is tested on a 2km straight track layout. Its derailment speed and stable speed are measured and compared to the predictions and results for a single bogie.

Predictions

The additional complexity of the vehicle and the additional weight of the vehicle body may cause the vehicle to be less stable than a single bogie and to derail at lower speeds.

The framerate of the simulation is expected to be lower than the bogie tests, as there are more bodies and joints in the scene.

Results

The results in Table 6.3 show the derailment speed and stable speed of the vehicle.

	Derailment Speed	**Stable Speed**
Locomotive - Straight	77.34	74

Table 6.3 - Derailment Speed and Stable Speed of a Locomotive on Straight Track

Both the derailment speed and stable speed are lower than the equivalent bogie results.

6.4.3 *Locomotive in Motion (Curved Track)*

The following tests were conducted with a single locomotive on the loop layout.

Design

A single locomotive is tested on the loop layout at a range of curve radii. Its derailment speed and stable speed are measured and compared to the predictions and results for a single bogie.

Predictions

Again, because of the additional complexity and weight of the vehicle, and based on the results from the previous tests, the vehicle is expected to derail at lower speeds than the bogie, and the framerate of the simulation is expected to be lower than the bogie tests.

Results

The following results show the derailment speed of the vehicle, compared to the predictions made for a single bogie. Figure 6.1 (overleaf) shows that the derailment speed of the locomotive is considerably lower than the derailment speed of a single bogie.

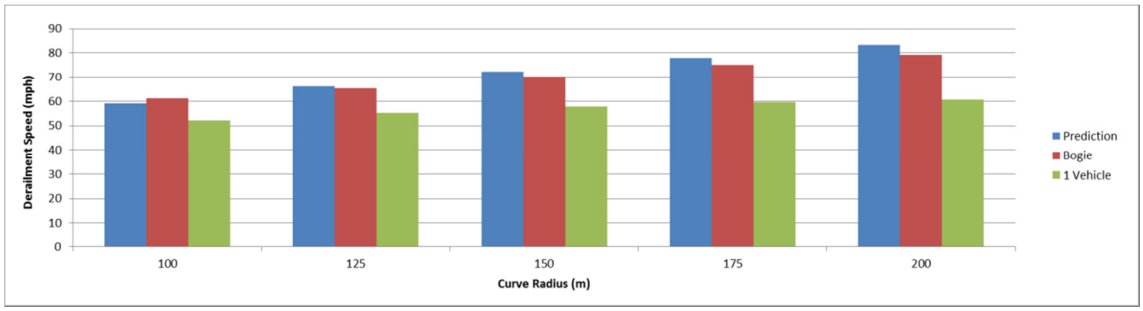


Figure 6.1 - Derailment Speeds for a single vehicle, compared to predictions are results for a single bogie

6.4.4 *Performance*

The average framerate of the simulation during the stable speed tests was 51.16 FPS.

6.4.5 *Conclusions*

The derailment speeds and stable speeds of the locomotive on straight and curved track are significantly lower than the results for a single bogie, as expected. There is a reduction in the derailment and stable speeds of approximately 10mph, and a similar trend in the curve derailment speed results. The derailment speeds do increase with curve radius, which is logically correct. The performance is also significantly lower, reaching values that are just below the real-time target of 60 FPS, but can still be described as 'near-real-time'.

6.5 Multi-Vehicle Trains

The following tests were performed under the same conditions as the bogie tests, with trains comprising different numbers of vehicles.

6.5.1 *Results*

Figure 6.2 (below) shows the average derailment speed for each train on each curve radius, along with the Nadal Prediction and Bogie derailment speeds for comparison purposes.

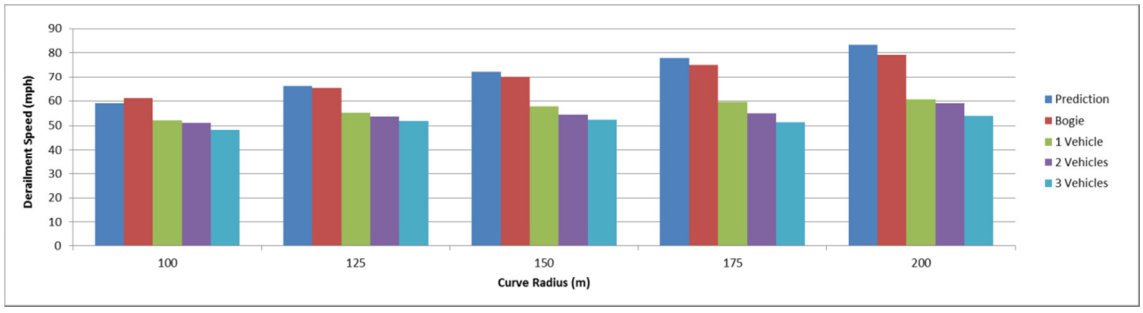


Figure 6.2 - Derailment speeds for a number of different vehicles

This data shows that adding additional vehicles causes the train to become less stable, since the derailment speeds at each curve radius decrease as more vehicles are added.

6.5.2 *Vehicle Derailments*

When a derailment is detected, the offending component's name is recorded. The name includes which bogie and which vehicle of the train the component belongs to, which allows the following results to be ascertained from the data. The following data is from the multi-vehicle tests and shows which of the vehicles in the train derailed the most across all of the tests on all radii. The charts in Figure 6.3 below represent the results of the two vehicle tests (left) and three vehicle tests (right).

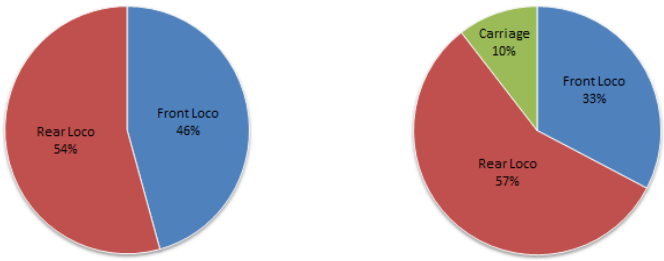


Figure 6.3 - Percentage of vehicle derailments (left) 2 Vehicles and (right) 3 Vehicles

During the two vehicle tests, the rear locomotive derailed in a higher number of tests than the front locomotive. In the three vehicle tests of tests, the rear locomotive also derailed in the majority of tests, while the carriage derailed in only 10% of the tests.

6.5.3 *Bogie Derailments*

The tool is also capable of producing the following data, which shows whether the front or rear bogie of the vehicle derailed the most.

Two Vehicles

The following charts (Fig. 6.4) show which bogie derailed in the two vehicle tests.

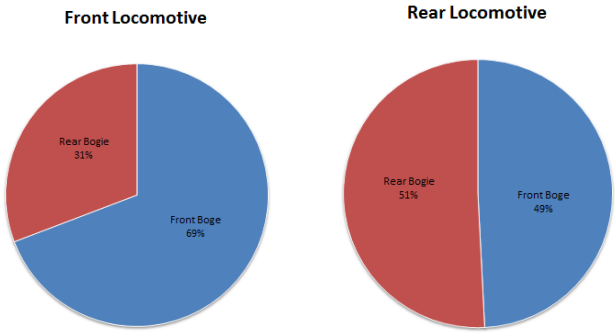


Figure 6.4 - Bogie derailments per vehicle (2 vehicles)

This data shows that, in instances where the front locomotive derailed, it was the front bogie that derailed in the majority of tests. Whereas when the rear locomotive derailed, the front bogie and rear bogie derailed in roughly equal amounts.

Three Vehicles

The data in the charts below (Figure 6.5) is from the three-vehicle tests.

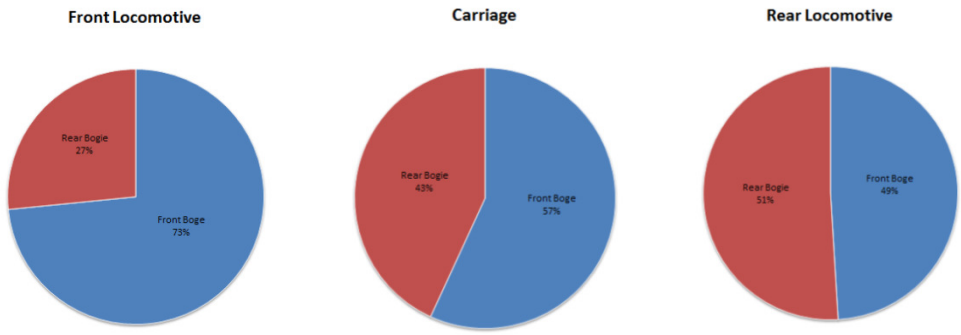


Figure 6.5 - Bogie Derailments per vehicle (3 Vehicles)

The data for the front and rear locomotive is very similar to the previous tests; the front bogie of the front locomotive derails in the majority of tests, while the bogies of the rear locomotive derail in roughly equal amounts. It also shows that the carriage's front bogie derails in more tests than its rear bogie.

6.5.4 *Wheelset Derailments*

It is also possible to identify how often individual components of the vehicle become derailed, as demonstrated by the wheelset derailment data below.

Two Vehicles

Table 6.4 (below) shows the number of derailments per wheelset recorded during the two vehicle tests.

Component	Derailments
FrontLoco_BogieF_WheelsetF	108
RearLoco_BogieR_WheelsetR	97
RearLoco_BogieR_WheelsetR	97
RearLoco_BogieF_WheelsetF	88
FrontLoco_BogieF_WheelsetR	69
RearLoco_BogieR_WheelsetF	63
RearLoco_BogieF_WheelsetR	52
FrontLoco_BogieR_WheelsetR	49
FrontLoco_BogieR_WheelsetF	29

Table 6.4 - Number of Derailments per Wheelset (Two Vehicles)

The chart in Figure 6.6 shows the derailments per wheelset in the 2 Vehicle tests.

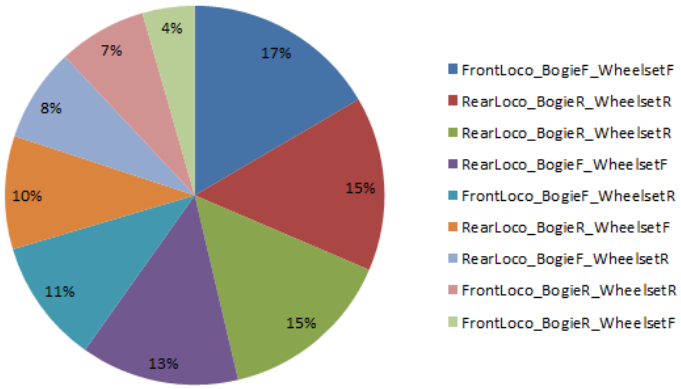


Figure 6.6 - Derailments per Wheelset (2 Vehicles)

This data shows that the front wheel of the front wheelset of the front locomotive was the most common wheelset to become derailed, while the front wheel of the rear bogie of the front locomotive was the least common.

Three Vehicles

The number of derailments per wheelset in the three vehicle tests is shown in Table 6.5, below.

Component	Derailments
FrontLoco_BogieF_WheelsetF	99
RearLoco_BogieR_WheelsetR	97
RearLoco_BogieR_WheelsetR	97
RearLoco_BogieF_WheelsetF	88
RearLoco_BogieR_WheelsetF	67
RearLoco_BogieF_WheelsetR	56
FrontLoco_BogieF_WheelsetR	41
FrontLoco_BogieR_WheelsetR	32
Carriage1_BogieF_WheelsetF	25
FrontLoco_BogieR_WheelsetF	18
Carriage1_BogieR_WheelsetF	12
Carriage1_BogieR_WheelsetR	11
Carriage1_BogieF_WheelsetR	9

Table 6.5 - Number of Derailments per wheelset (3 Vehicles)

The following chart (Fig. 6.7) shows the data from the three vehicle tests visually.

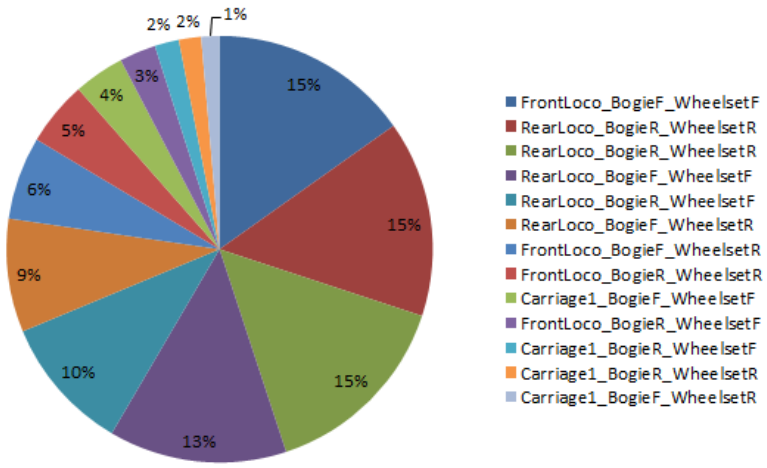


Figure 6.7 Derailments per Wheelset (3 Vehicles)

Again, the front wheelset of the train derails the most, but in these tests it is the carriage wheelsets that derail the least.

6.5.5 *Performance Data*

Table 6.6 (below) shows the average framerate recorded in each test on each curve radius during each of tests presented in this section.

	Average Framerate (FPS)					
Radius	*100*	*125*	*150*	*175*	*200*	*Average*
Bogie	64.9	66.3	61.6	67.1	64.5	66.0
1 Vehicle	51.3	48.9	48.7	54.2	52.7	51.16
2 Vehicles	43.3	44.3	43.9	47.9	46.5	45.18
3 Vehicles	37.9	40.9	40.8	40.4	41.5	40.3

Table 6.6 - Performance figures for a range of trains and curve radii

Increasing the number of vehicles leads to a decrease in framerate, which is to be expected. Increasing the radius of the curves does not affect performance. A single bogie runs at 65.4 FPS and is therefore running in real-time.

A single vehicle runs at 51.15 FPS and introducing additional vehicles decreases the average framerate by approximately 5fps. This level of performance is below the real-time target of 60 FPS, but can still be described as ‘near-real-time’.

6.6 Sample Rapid Prototype Testing

In this section are samples of the results collected during tests designed to test the ability of Locomotion to act as a rapid prototyping tool. The following tests were conducted with a single bogie or locomotive on a 2km straight and a 200m radius looped track.

The tests in Chapter 5 have already demonstrated how the batch testing features can be used to evaluate a range of different simulation settings automatically, without requiring user input or supervision after the initial setup. However, the tests in Chapter 5 involved using the vehicle properties specified in Table 3.1 and varying the parameters of the Physics Engine. The tests in this section will use the Physics Engine parameters that produced the best results during the testing in Chapter 5, and will adjust the properties of the vehicle itself, as an engineer might do during rapid prototype testing.

In previous tests on the straight:

- The stable speed of the bogie was: 88mph.
- The derailment speed of the bogie was: 92.27 mph
- The stable speed of the locomotive was: 74 mph
- The derailment speed of the locomotive was: 77.34 mph

In previous tests on the 200m radius loop:

- The bogie derailed at: 79.22 mph.
- While it was predicted by the Nadal Formula to derail at: 83.38 mph.
- The locomotive derailed at: 60.53 mph.

6.6.1 *Test Design*

Three parameters of the vehicle have been selected for testing in this section. These parameters, along with their default values, are as follows:

- Wheelbase: 2.2 m
- Bogie Spacing: 11.1m
- Locomotive Centre of Gravity: -1.0

Each of these parameters will be tested using Locomotion's batch testing features, in order to determine the effect of adjust them on the stability, derailment speed and stable speed of the bogie/vehicle.

The vehicle is tested on two layouts:

- 2km Straight
- 200m Loop

This enables an evaluation of its top speed on the straight track and of its derailment speed on the curved track (200m was selected as it is one of the layouts on which the Nadal Limit predictions can be used).

6.6.2 *Bogie Spacing Tests*

In these tests, the spacing between the bogies is altered to determine if it has an effect on the stability and derailment speed of the vehicle.

Test Design

The locomotive is tested on straight track, and on the 200m looped track. The bogie spacing is adjusted from the default of 11.1m to 15.1m, as shown in the screenshots below (Figure 6.8). This range of bogie spacing is the widest range possible which avoids contact with the chassis elements, while ensuring that the front/back wheels are do not extend in front of/behind the vehicle.

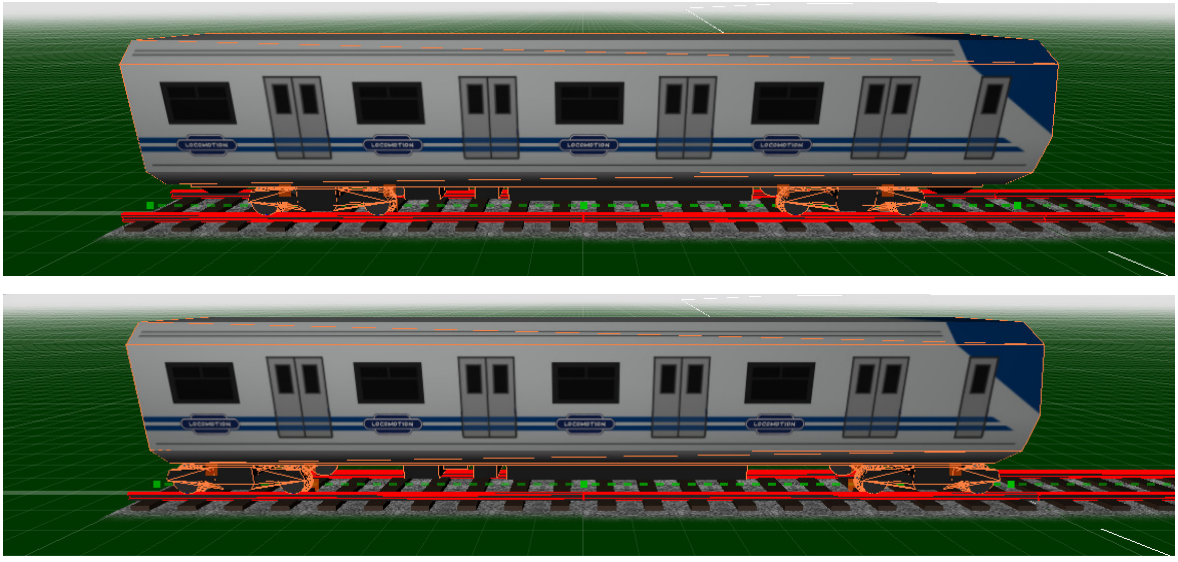


Figure 6.8 - Locomotive with Bogie Spacing of 11.4m (top) and 15m (bottom)

Predictions

It is expected that a significant reduction to the bogie spacing would reduce the stability of the vehicle, but the range of spacing values tested are not expected to make a significant difference.

Straight Results

Figure 6.9 (below) shows the results of testing on the 2km straight track.

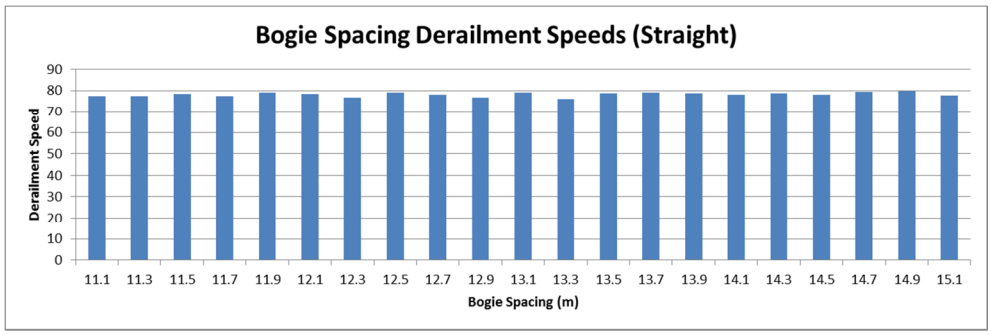


Figure 6.9 - Derailment Speeds per Bogie Spacing (Straight)

There is no significant trend in the results. The results vary between a minimum of 75.88 mph and 79.744 mph, but not in any way consistent with the changes in bogie spacing.

Loop Results

Figure 6.10 (below) shows the results of testing on the 150m radius loop track.

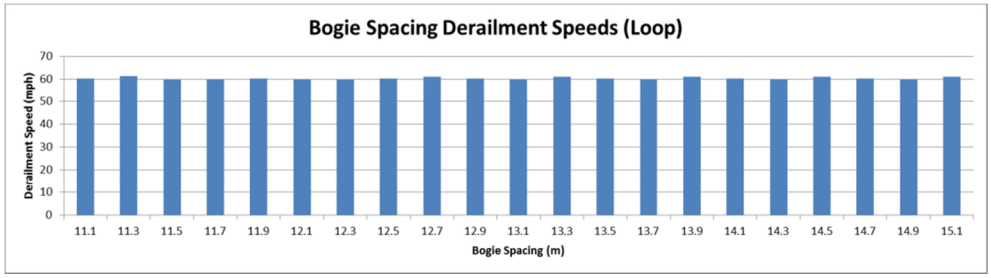


Figure 6.10 - Derailment Speeds per Bogie Spacing (Loop)

As with the straight tests, there is some variance in derailment speed of between 59.96 mph and 61.49 mph, but it does not change in a way that is consistent with the changes in the bogie spacing.

Stable Speeds

The stable speed of the locomotive was 74mph in all tests, regardless of bogie spacing, which is the same as the results in Section 6.4.

Conclusions

Adjusting the bogie spacing appears to have no significant effect on the stability of the Locomotive. However, the tests in this section have demonstrated that it is possible to iteratively adjust the dimensions of the vehicle using the batch testing feature in the Locomotion tool.

6.6.3 *Bogie Wheelbase Tests*

In these tests, the spacing between the wheelsets of the bogie is altered to see if it has an effect on the stable speed of the bogie, or on its derailment speed on the 200m radius curve. The default value for the bogie wheelbase, based on Table 3.1, is 2.3m.

Test Design

The bogie is tested on straight track, and on the 200m looped track. Its derailment speed and stable speed on each layout is recorded. The wheelbase of the bogie is adjusted from 1.5 to 3.0m. The screenshots below (Fig. 6.11) show the extremes of the values tested.

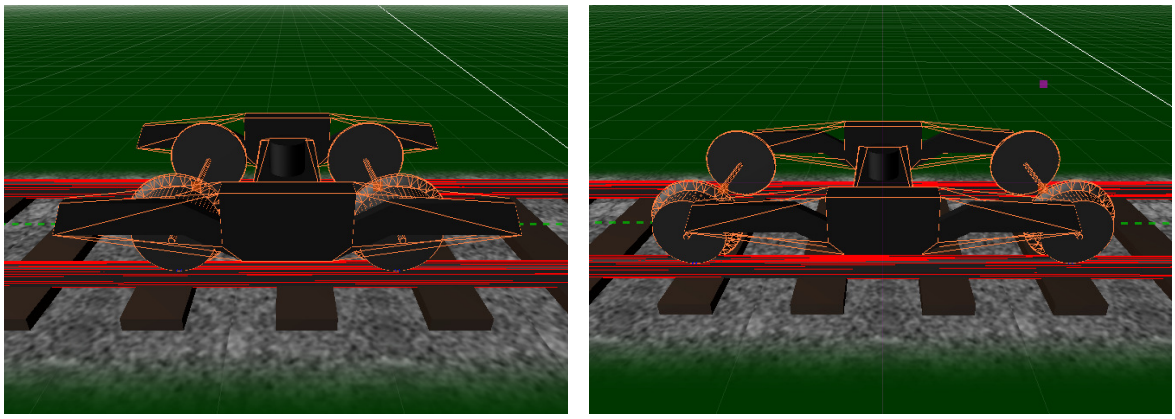


Figure 6.11 - Bogies, with wheelbase of 1.5m (left) and 3.0m (right)

Predictions

It is expected that increasing the spacing between the wheels of the bogie will produce more stable results (and that reducing the spacing below the default value will make the bogie less stable).

Straight Track Results

The graphs below (and overleaf) show the results of testing on the 2km straight track. Figure 6.12 shows the derailment speed of the bogie in each test.

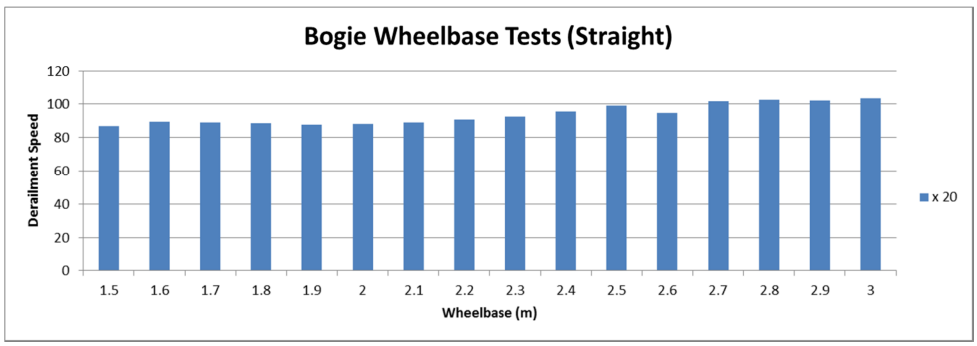


Figure 6.12 - Bogie Wheelbase Derailment Speeds (Straight)

Figure 6.13 shows the number of derailments in each batch of 10 tests.

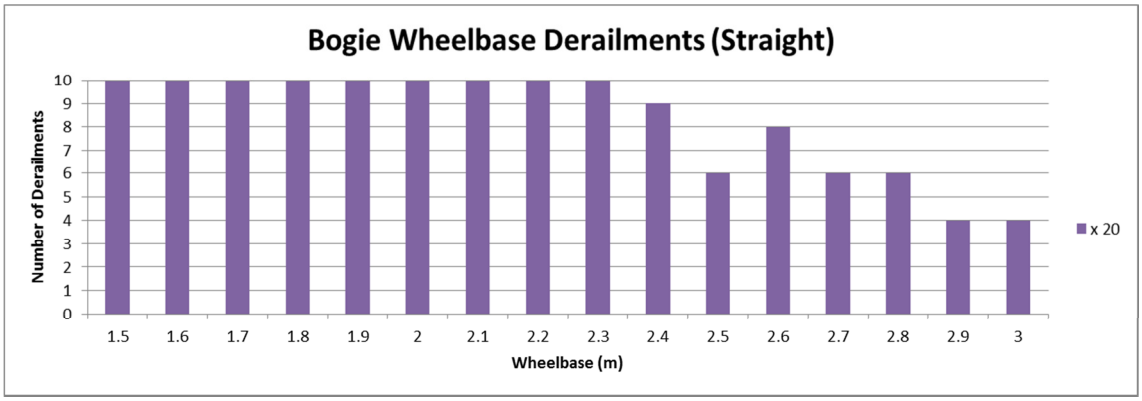


Figure 6.13 - Derailments per Bogie Wheelbase (Straight)

The results show an increase in the derailment speed of the vehicle, and a decrease in the number of derailments, as the wheelbase of the bogie was increased. This means the bogie was successfully reaching the end of the track in at least some of the tests at a wheelbase of 2.4m and above. In tests with a Wheelbase of 3m, the bogie only derailed in 40% of the tests and reached a peak speed of 103.223 mph, an increase of approximately 10mph over the results achieved with the default wheelbase of 2.3m.

Stable Speed Tests (Wheelbase = 3.0m)

Further testing was conducted to determine how increasing the wheelbase affects the stable speed of the bogie. The graph in Figure 6.14 (below) shows the derailments per target speed for a wheelbase of 3.0m on the 2km straight track.

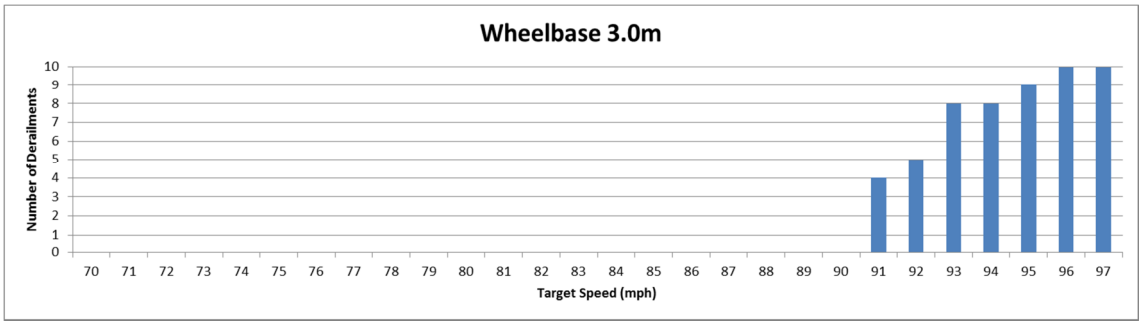


Figure 6.14 - Derailments per Target Speed (Bogie, Wheelbase 3.0m)

In stable speed tests at a wheelbase of 3.0m, the bogie was able to reach a stable speed of 90mph, an increase of 2mph over the previous best results.

Loop Results

Figure 6.15 (below) shows the results of testing on the 200m radius loop track. During these tests, the bogie derailed in all tests with all wheelbase values.

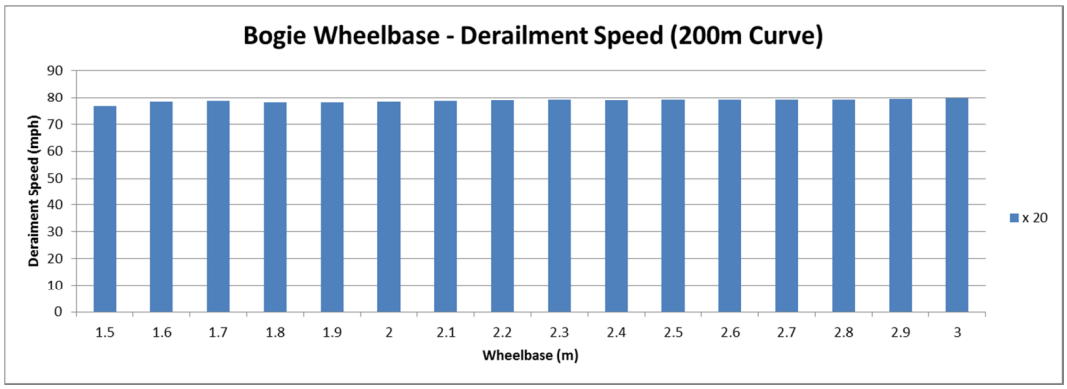


Figure 6.15 - Derailment Speeds per Bogie Wheelbase (200m Loop)

There is, in general, an increase in the derailment speed of the bogie as wheelbase increases. It increases from a minimum of 76.70 mph at 1.5m to 79.77mph at 3.0m. However, this increase is minimal and it could be argued that the results do not show a conclusive improvement.

Conclusions

Both tests of tests results show an increase in the derailment speed of the bogie. This suggests, as expected, that increasing the wheelbase of the bogie makes it more stable.

6.6.4 *Centre of Gravity Tests*

The following tests show the effect of adjusting the centre of gravity of a single locomotive on its cornering behaviour.

Test Design

The Locomotive is tested on straight track, and on the 200m looped track. Its derailment speed and stable speed on each layout is recorded.

The centre of gravity (COG) of the vehicle entity is a vector, which defines the displacement of the centre of gravity from its default value, which for a rigid body in PhysX is defined (in local coordinates) as [0, 0 ,0]. In these tests, the z offset of the centre of gravity is adjusted from -1.0m to (+)1.0m.

This is illustrated in Figure 6.16 (overleaf). The grey box represents the locomotive body, the black cross represents the default centre of gravity of the body ([0,0,0]) and the arrows show how the COG is adjusted in the z axis.

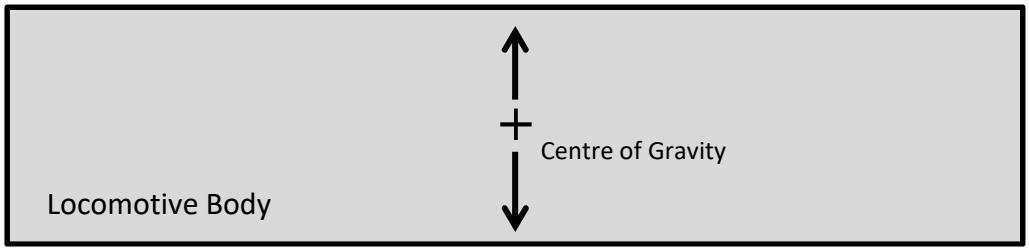


Figure 6.16 - Illustrating the Centre of Gravity of the Locomotive Body and how it is adjusted

Predictions

It is expected that raising the centre of gravity of the locomotive will reduce the derailment speed of the vehicle, particularly on the curved track.

Straight Results

The graph below (Fig 6.17) shows the results of testing on the 2km straight track.

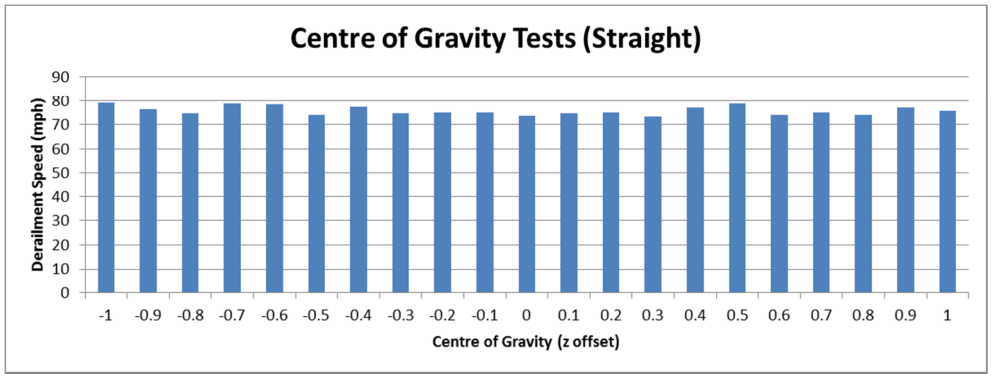


Figure 6.17 - Derailment Speed per Centre of Gravity (Straight Track)

The graph in Figure 6.18 (below) is a closer look at the upper part of the graph (speeds above 70mph), showing the variance of results in more detail.

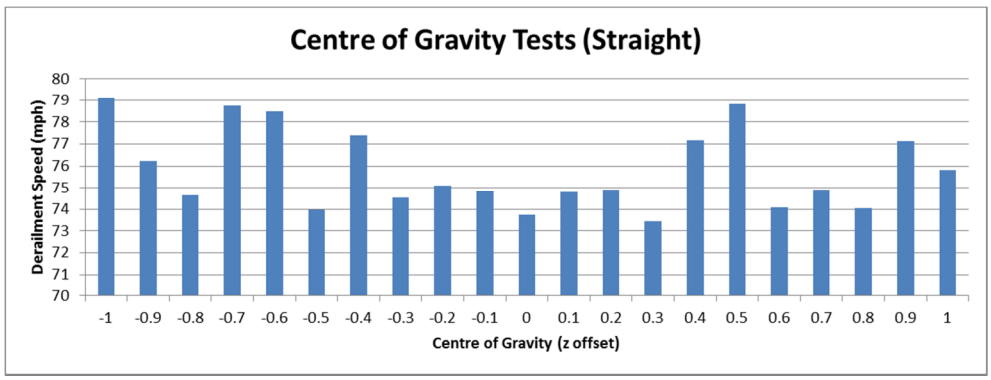


Figure 6.18 - Derailment Speed per Centre of Gravity (Straight Track - 70mph to 80mph)

There is some variance in the derailment speed between 73.44 mph and 79.14 mph. It could be argued that there is a slight downward trend, but the values fluctuate, so it is difficult to be certain of the effect of altering the COG from this data.

Loop Results

Figure 6.19 (below) shows the results of testing on the 200m radius loop track.

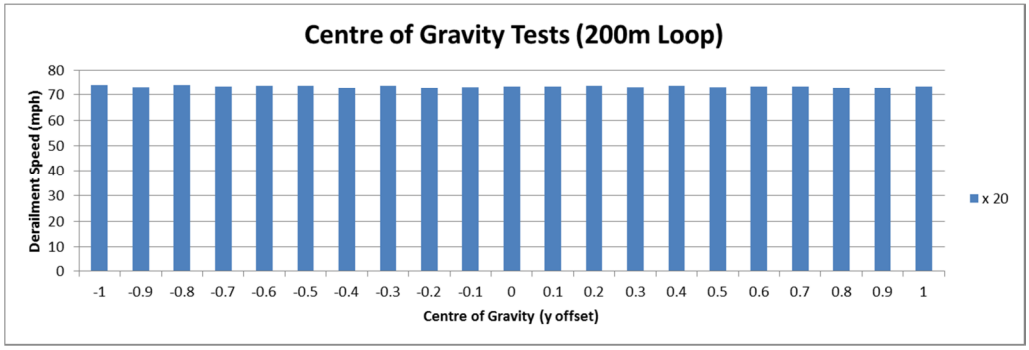


Figure 6.19 - Derailment Speeds per Centre of Gravity (200m loop)

The graph below (Figure 6.20) is a closer look at the upper part of the graph (speeds above 70mph), showing the variance of results in more detail.

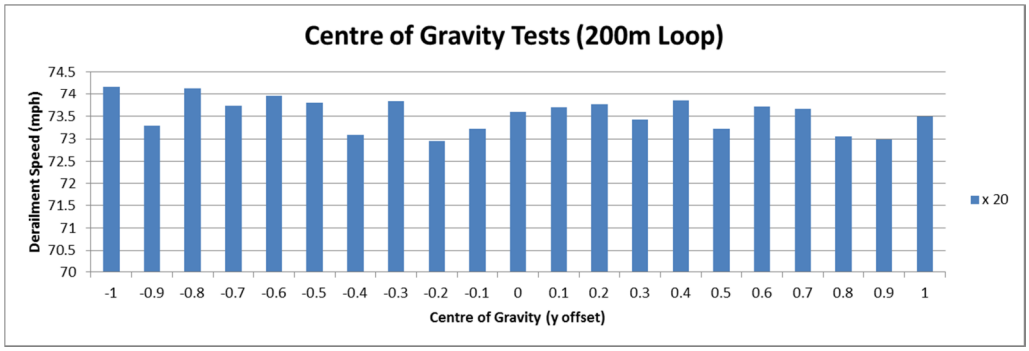


Figure 6.20 - Derailment Speeds per Centre of Gravity (200m Loop - 70mph to 75mph)

There is some variance in the derailment speed between 73.44 mph and 79.14 mph. Again, it could be argued that there is a downward trend, but the values fluctuate, so it is difficult to be certain.

Conclusions

The data collected in this section shows that adjusting the centre of gravity of the body shell of the locomotive may have a small effect on the stability of the vehicle, reducing the derailment speed on straight and curved track by a small amount as the centre of gravity is raised. The results also show that it is possible to vary this vehicle property as part of the simulation tool's batch testing features.

6.6.5 *Conclusions*

The Testing in Section 6.6.2 showed that adjusting the spacing of the bogies did not appear to have any effect on the derailment speeds of the locomotive on straight or curved track. Adjusting the wheelbase of the bogie showed that moving the wheelsets further apart seemed to increase the stability of the bogie, causing it to derail at higher speeds on both straight and curved track. Adjusting the locomotive's centre of gravity had a negligible effect, but raising it could arguably be said to decrease the stability of the vehicle and increase its derailment speed, though there is a lot of variation in the data.

These results have not been verified with any mathematical predictions, simulated results or real-world testing, but are included to demonstrate the capabilities of the tool. Whatever the results, the testing in this section has shown that the tool is capable of being used to quickly evaluate the effect of adjusting the properties and dimensions of the vehicle and its components, and shows how Locomotion could be useful as a rapid-prototyping tool.

6.7 Sample Gauging Tests

The following test scenario was designed to test the gauging of the vehicle. This would allow engineers to avoid gauging errors (like the one mentioned in Section 2.3.2).

6.7.1 *Test Design*

A set of 'gauge gates' have been built, as illustrated in Figure 6.21, below. Each gate is made up of two vertical cuboids and a cuboidal 'cross-piece'. The gate is constructed based on two parameters; height and width. The polygons are created using the PhysX SDK, which has a method to create a box actor given a vector that defines its dimensions. The cuboids are half a metre square, with their height being defined according to the height and width properties of the gate. Figure 6.21 (left) shows a train passing through a gate, and (right) shows the height and width properties of the gate.

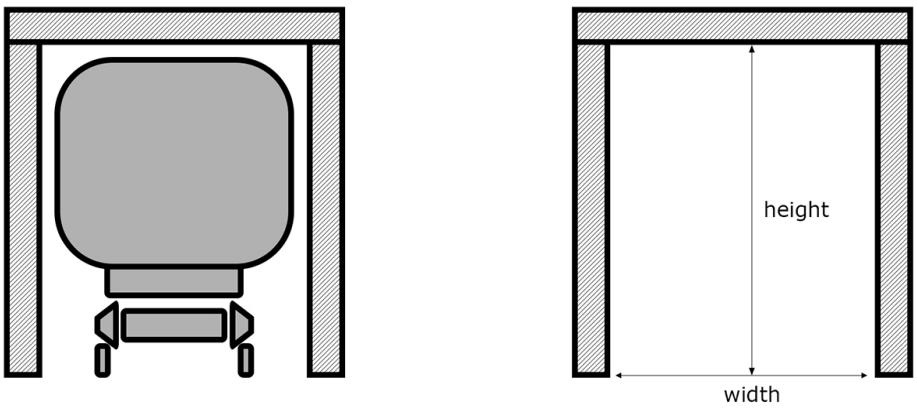


Figure 6.21 - Diagram showing the gates used in the gauging tests

The gates have collision detection enabled (and so will notify the application when a collision has occurred) but do not have collision resolution enabled, so the physics engine will not process the collision beyond reporting it. This means that if the vehicle collides with gate 5, the test will not end and it will go on to collide with gates 6+ as well. Gates are colour coded green initially and change to red to indicate that a collision has occurred.

100 gates are placed at 1 metre intervals along a 100m track. First, the height of the gate is set to a value above the height of the train, and the height is reduced by 1cm between each gate. The tests are then repeated, but with the width value decreasing by 1cm between each gate.

6.7.2 *Predictions*

The results should be reasonably accurate, but Skin Width will have to be taken into account. Skin Width of the gates and the body of the locomotive is set to its default value of 0.025m.

- The width of the vehicle body rigid body is 2.8 m, so the vehicle should collide with gates of that width
- The height of the body is 2.98 m, and the full height of the assembled vehicle (including track, wheels, bogie and chassis) is 4.04m , so the vehicle should collide with gates of that height

6.7.3 *Screenshots*

The screenshot below (Figure 6.22) shows the test in action.

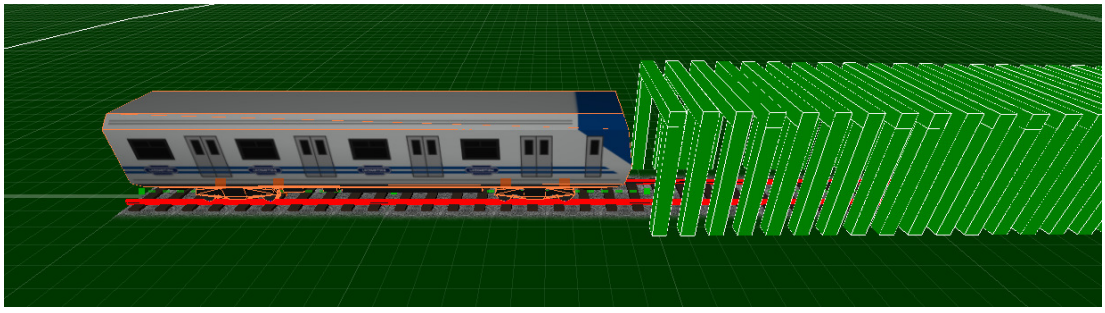


Figure 6.22 - Screenshot from Gauge Gate testing, showing gates highlighted in Green (no collision)

The vehicle starts outside the ‘tunnel’ of gates, and the gates start with their ‘collision count’ set to zero, and their colour set to green. The screenshot overleaf (Figure 6.23) shows the train at a later stage, where collisions have occurred.

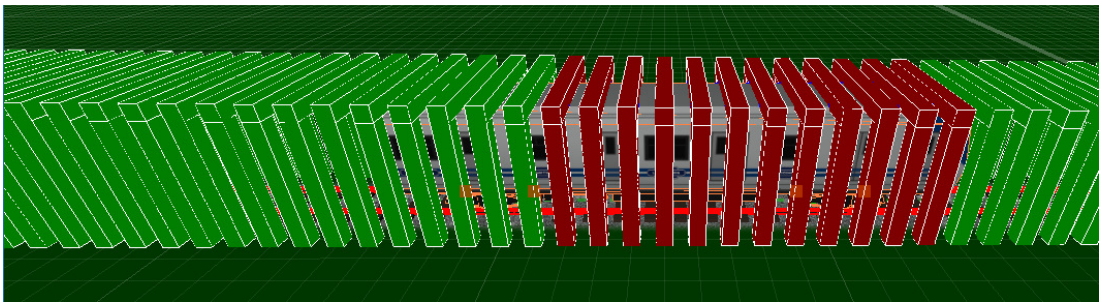


Figure 6.23 - Screenshot from Gauge Gate testing, showing gates highlighted in Red (collision)

The figure shows that the locomotive has collided with a series of gates, and how the colour of the gates has changed, making it easy to identify the point where the first collision occurred. Each time a collision with a gate is detected, the 'collision count' of the gate is incremented.

6.7.4 *Results*

The results of the gauge tests with a locomotive and arrange of gate height and width parameters were as follows.

Height Tests

In these tests, the height of the first gate was set to 4.2m, which is 0.16 metres above the height of the vehicle, and each subsequent gate was set to be 1cm lower than the previous gate. There were 100 gates with heights ranging from 4.2m to 3.21m. The test was repeated 10 times. The graph in Figure 6.24 (below) shows the number of collisions for gates with heights of between 4.2m and 3.75m, during tests conducted at 1mph.

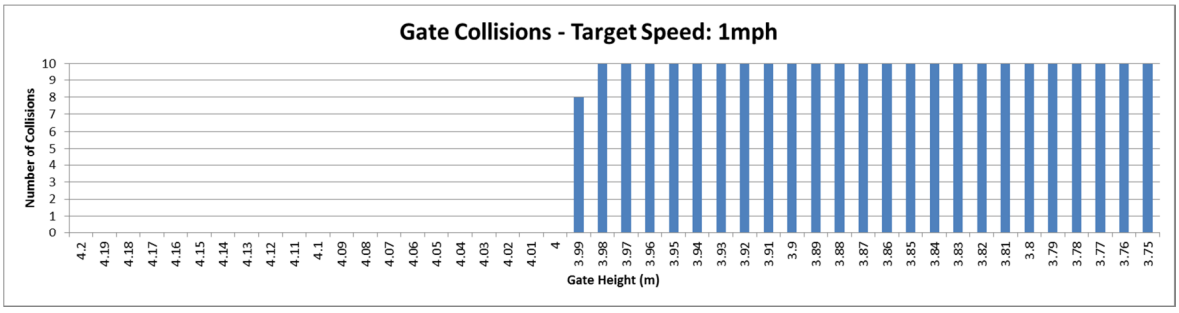


Figure 6.24 - Graph of Collisions per Gauge Gate Height

As the graph shows, there were no collisions for a gate height of above 4m, but there are 8/10 collisions at a height of 3.99m, and 10 collisions per gate at a height of 3.98m and below.

The tests were also repeated from 1 to 10mph, and a subset of the results is shown in Table 6.7 (below).

	Collisions								
Gate Height (m)	*...*	*4.03*	*4.02*	*4.01*	*4*	*3.99*	*3.98*	*3.97*	*...*
1 mph	...	0	0	0	0	8	10	10	...
2 mph	...	0	0	0	0	0	10	10	...
3 mph	...	0	0	0	0	0	10	10	...
4 mph	...	0	0	0	0	0	10	10	...
5 mph	...	0	0	0	0	0	10	10	...
6 mph	...	0	0	0	0	0	10	10	...
7 mph	...	0	0	0	0	0	10	10	...
8 mph	...	0	0	0	0	0	10	10	...
9 mph	...	0	0	0	0	0	10	10	...
10 mph	...	0	0	0	0	0	10	10	...

Table 6.7 - Gauging Height Test Results

The vehicle hit the 3.98 height gate 10 times in each of the 10 batches, and did not hit any of the previous gates. It is only in the tests at 1mph that the vehicle hits the gate at 3.99m. This suggests that the vehicle is somehow less stable when travelling at 1mph. The height of the full vehicle model is 4.04, and the results are 6cm out. However, the gates and the locomotive body both have a Skin Width of 0.025m, and the wheels and rails have a Skin Width of 0.01m each, which is a likely explanation for this discrepancy.

Width Tests

In these tests, 100 gates were placed along a 1km track. The width of the first gate was set to 3m, which is wider than the width of the vehicle, and each subsequent gate was 1cm narrower than the previous gate. The test was repeated 10 times and the results are shown in Figure 6.25 (below). In these tests, the collision counts are as high as 20 for each gate, indicating that collisions occurred with both the left and right post (which are separate rigid bodies, so there is one collision callback for each).

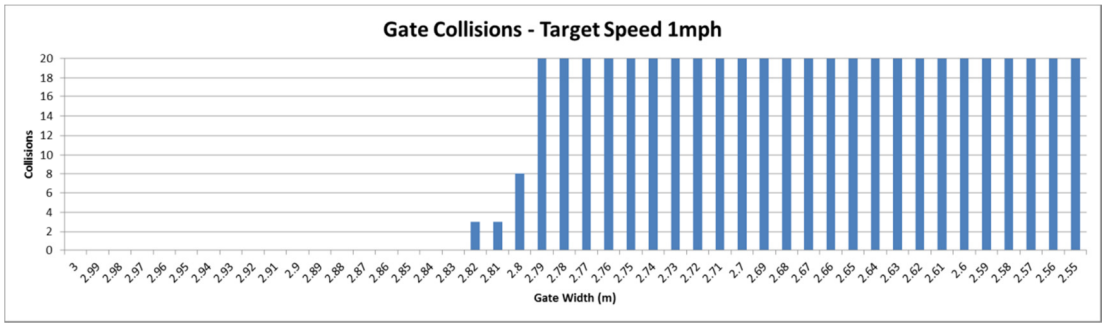


Figure 6.25 - Gate Collisions - Target Speed 1mph (Width Tests)

At a width of 2.82 and 2.81m, there are 3 collisions, which suggests that there was some lateral movement of the vehicle and that it possibly only hit one of the side posts in some of the tests. There are 8 collisions at a width of 2.80m and then collisions in all tests at a width of 2.79 and below. This is just 1cm below the actual width of the vehicle body, but it should be narrower as this does not take Skin Width into account. The collisions should not consistently occur until the gate width has decreased to 2.75m.

The tests were also repeated from 1 to 10mph, and a subsection of the results are shown in Table 6.8 (below).

	Collisions												
Gate Width (m)	*...*	*2.83*	*2.82*	*2.81*	*2.8*	*2.79*	*2.78*	*2.77*	*2.76*	*2.75*	*2.74*	*2.73*	*...*
1 mph	...	0	3	3	8	20	20	20	20	20	20	20	...
2 mph	...	0	0	0	0	11	20	20	20	20	20	20	...
3 mph	...	0	0	0	0	0	5	10	20	20	20	20	...
4 mph	...	0	0	0	0	0	0	0	0	20	20	20	...
5 mph	...	0	0	0	0	0	0	0	0	20	20	20	...
6 mph	...	0	0	0	0	0	0	0	0	20	20	20	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 6.8 - Gauging Height Test Results

The results show that at speeds of 3mph and below, there are collisions at wider gates (2.76m and above) but these collisions stop in the tests at 4mph and above, which consistently collide at widths of 2.75. This is correct with respect to the predictions, as the model is 2.8m wide, but the Skin Width of the gate posts and vehicle body is 0.025m (a total difference of 5cm). This concurs with previous data suggesting the vehicle is unstable when it is forced to maintain very low speeds.

6.7.5 *Conclusions*

The results in most of the gauge tests were logically correct, once the Skin Width of the rigid bodies is taken into account. In some tests at low speeds (1mph in height tests and below 4mph in the width tests) there were additional collisions that should not have occurred. This may indicate some low speed instability in both scenarios, and possible lateral movement of the vehicle in the width tests.

These are simplified scenarios, but they demonstrate that the Locomotion tool could be used as a gauging tool. It may be necessary to adjust the models or the code to account for Skin Width, but the results were within, at worst, 4cm of the predicted results.

A similar test could be constructed to create a tunnel on curved or sloped track, but time constraints prevented the full implementation and evaluation of these scenarios. More details on how this feature could be extended are included in Future Work (Section 7.4).

6.8 Chapter Summary

The data in this section is presented as an example of the sort of data that the simulation tool can produce.

Whole Vehicles and Multi-Vehicle Trains

This data includes the derailment speeds for two and three vehicle trains. This data could be used to analyse derailment speed for particular trains on particular curve radii, something that is currently prohibitively expensive, or is beyond the capabilities of existing rail simulation tools. The results also include data about which vehicles/components derailed the most, which. could allow engineers to identify the components that are causing the problems and target them with design improvements.

A Rapid Prototyping Tool

These tests, as well as the testing conducted in Chapter 5, shows how the batch testing features and flexibility of the Locomotion tool allows a range of simulation/vehicle parameters to be adjusted and for any changes in the results to be analysed. This shows that the simulation tool could be suitable for use as a rapid prototyping tool.

A Gauging Tool

The gauge testing results were promising. There were some collisions with the gauge gates at low speed that should not have occurred, but the worst tests were within 4cm of the predicted values, and higher speed values (>3mph) were exactly as predicted. This shows that the tool has great potential for use in gauge testing, if a method can be found to take the Skin Width of the models into account.

A (Near) Real-Time Simulation

In terms of performance, on the looped track a bogie runs at 66 FPS; a single locomotive runs at approximately 50 FPS; two locomotives approximately 45 FPS and three locomotives at 40 FPS. Framerate varies with the complexity of the layout, as well as the number of vehicles in the train. Although not truly real-time, this performance can still be described as ‘near real-time’ and is a considerable increase in speed over traditional tools.

The tests took between 9 and 12 hours, depending on the curve radius, derailment speed and framerate of the simulation. These speeds could be reduced if the simulation could be optimised and its performance increased. However, the fact that the tests are running in near real-time means that hundreds of tests can be conducted and average results collected in the same time (or less time) than a single test may potentially take in the more sophisticated engineering simulation tools.

Chapter 7

Conclusions and Future Work

This chapter contains the conclusions of the research presented in this thesis, along with suggestions for future research and development that could be conducted using Locomotion, or a similar real-time rail dynamics simulation tool.

7.1 Thesis Overview

Chapter 2 presented background material and related work in the fields of rail dynamics, rail simulation and real-time physics engines. This chapter identified the problem area, introduced key terminology and application areas within rail dynamics, and discussed how a real-time simulation tool could supplement the existing tools that are currently used in the rail industry. It also discussed real-time physics engines, their applications inside and outside the games industry and how they might be the solution to creating a real-time engineering simulation tool.

Chapter 3 presented the design of the Locomotion tool, including the design of the virtual vehicle and testing environment. A number of design variations were identified that would require testing in order to determine which was the most suitable for the simulation of rail dynamics. This chapter also included the software design, including the integration of the PhysX engine into the simulation and the identification of key physics engine parameters that controlled the fidelity of the physical simulation and which did not correspond to real-world values and so would needed to be evaluated.

Chapter 4 presented details of the implementation of the Locomotion tool, including a summary of the issues that were encountered (some of which caused significant delays) and some of the changes that were made to the design, such as disabling Vertical Synchronisation in order to study the effect of any changes to the simulation on its performance.

Chapter 5 presented an evaluation of the Wheel/Rail interface simulation, including tests conducted with the default physics engine parameters and how the results were iteratively improved by adjusting the parameters defined in Chapter 3. The results show that the simulation can be configured and augmented to produce results that are within 2% of the Nadal Limit predictions, with a standard deviation of less than 4 mph.

Chapter 6 presented sample data from multi-vehicle testing, as well as an evaluation of the gauge testing and rapid prototyping features of the Locomotion tool.

7.2 Aim and Goals

The following discussion is based on the aims, goals and intended applications that were described in Section 3.1.

7.2.1 *Aim*

The main aim of this research was to design, develop and evaluate a rail dynamics simulation tool based on a real-time physics engine. This thesis has presented details of the design, implementation and evaluation of the Locomotion simulation tool, which was developed using the PhysX engine and has produced some promising results, which will be discussed in more detail later in this chapter.

7.2.2 *Goals*

A discussion of this this research with reference to the goals (presented in Section 3.1.2) is included below:

1. *To develop a real-time rail dynamics simulation tool based on a physics engine.*

The Simulation Tool presented in this Thesis is capable of simulating the dynamics of a rail bogie in real-time. This is thanks in part to the integration of the PhysX engine into the application, as described in the Simulation Design (Chapter 3).

2. *To determine the error-bound of the simulation in order to appraise its usefulness to engineers.*

As discussed in Chapter 5, initial testing with simple objects, as well as with more complex rigid bodies, revealed very consistent results with margins for error that were within hundredths of a percent. Testing on bogie derailment with adjusted parameters and the addition of the spline-based centring force produced results that were within 2% of the target derailment speed, with a standard deviation of less than 4 mph. This error bound should be sufficiently small as to be useful to engineers, in scenarios where a real-time tool would be used instead of, or as a supplement to, the more accurate simulation tools.

3. *To constrain the error - by introducing constraints, adjusting physics engine parameters or by applying additional forces, where necessary/possible.*

The initial results were improved by adjusting a range of physics engine parameters and by the addition of the spline-based centring code, as described in the testing presented in Chapter 5.

4. *To produce a tool that is flexible, to enable it to be evaluated easily and to allow it to be used as a rapid prototyping tool.*

As many parameters of the simulation, vehicle components and testing environment as possible are defined in configuration files, which are loaded into the simulation and do not require any recompilation of the application code to edit. Many of the parameters are also capable of being automatically incremented during batch testing, as shown throughout Chapters 5 and 6.

5. *To evaluate the ability of the tool to simulate the wheel/rail interface.*

Evaluation with the conical wheelset has shown sinusoidal lateral motion of the wheelset which is logically correct and shows that the centring behaviour of the wheelset is occurring to at least some degree in the simulation. This research has also included testing of rail bogies, which are shown to derail at speeds close to those predicted by the Nadal Limit, if the simulation is adjusted and augmented appropriately.

6. *To attempt to simulate the dynamic behaviour of multi-vehicle trains.*

Multi-vehicle trains have been simulated using abstracted models, and although their performance is below the target real-time framerate of 60 FPS, it can still be described as ‘near-real-time’. Time constraints prevented further evaluation, but sample data was presented in Chapter 6.

7. *To attempt to simulate other aspects of rail vehicle dynamics, such as gauging*

Gauge testing does not necessarily require the simulation of the wheel/rail interface to be truly accurate, nor does it necessarily require the vehicle to be capable of travelling at very high speeds. An evaluation of the tool’s suitability for gauge testing is included in Chapter 6 and, though limited time was available to fully evaluate these features, the initial testing produce promising results.

8. *To use results from the simulation to discuss whether such a real-time tool would be useful to rail engineers, and for what purpose(s).*

A more detailed discussion of the suitability of the simulation tool for its intended applications is included in the next section; however the results in many of the tests were promising. The simulation of the WRI was reasonably accurate, within a certain error bound, but it need not necessarily be so in order to enable other testing applications, such as gauge testing.

7.2.3 *Intended Applications*

Based on the research presented in Chapter 2, a number of potential applications for a real-time rail dynamics simulation tool were discussed in Section 3.1.3. In this section, a discussion of the locomotion tool's suitability for each of these intended applications is presented, based on the data collected in Chapters 5 and 6.

Rapid Feedback for Design Changes

The tool can be easily customised by adjusting the various parameters, which control vehicle properties (such as mass and material properties), test settings (such as track layout and vehicle type) and component settings (such as the choice between Single Body and Multi Body wheelsets). As many of these parameters as possible are defined in text files that are loaded into the simulation and so adjusting them does not require any recompilation of the application code, reducing the turnaround between tests. Feedback from the simulation is also rapid. The longest tests, which involved multiple batch sets, consisting of hundreds of tests across a range of parameters and which were conducted on the widest curve radii, were completed in around 12 hours. This is considerably shorter than the amount of time that traditional engineering simulation tools could take to process certain scenarios; multiple tests have been conducted using Locomotion quicker than a single test may take in traditional tools.

A Rapid Prototyping Tool

In addition to the features described above, the tool also has batch testing features, which allows a range of simulation settings, vehicle parameters and vehicle components to be automatically tested without user supervision. The tests conducted in Chapters 5 and 6 show that the batch testing features (and the additional LocoDataScan utility) mean that Locomotion would be very useful as a rapid prototyping tool, as it allows the user to quickly see the effect of changing vehicle components and parameters, with averages and comparisons between derailments, lateral offset, wheelset stability, etc. calculated automatically and data exported in a format that can be loaded into other applications such as Microsoft Excel for rapid analysis.

A Tool to Analyse Vehicle Stability and Derailment Behaviour

Time constraints prevented the full implementation of features such as suspension, yaw resistive motion on the bogie pivots etc., however sample data from successful multi vehicle tests was included in Chapter 6.

- *Determining the maximum safe speed of the vehicle*

Locomotion has been shown to produce derailment results for a single bogie close to the Nadal derailment predictions. It could conceivably be used to test the derailment speed for real-world track layouts, rather than the simplified layouts used in this research.

However, further evaluation is required to ensure that the simulation of the wheel/rail interface is sufficiently accurate, and further development of the virtual vehicles (with features such as suspension), is necessary to make such assertions. Additionally, in its current state the simulation is limited to tests at or below 88mph, which limits the speeds and layouts that can be tested using the tool.

- *Recording flange collisions*

The Multi Body Wheelset can used to detect flange collisions, as shown in Section 4.1.5 and Section 5.7. However the MB wheelset was considerably less stable than the single body wheelset and was only capable of achieving a stable speed of 8mph. However, if an alternative method of detecting flange collisions was developed to enable this behaviour to be studied, then the simulation tool would be eminently suitable for such testing.

- *Studying the effect of inter-vehicle connections on vehicle stability*

Time constraints prevented a full implementation and evaluation of these behaviours. However, the tool is capable of simulating multi-vehicle trains in near-real-time, and the results presented in Chapter 6 show how the addition of extra vehicles to the train affects its stability. It is conceivable that the tool could be extended to include extra features to study these behaviours, which would likely compromise the simulation's performance but may still be capable of near-real time execution. This is discussed in more detail in Section 7.4.

Wheel/Rail Interface Simulation

Tests presented in Chapter 5 with the Conical Wheelset (Section 5.7.3 and Section 5.10) have shown sinusoidal motion of the wheelset, and possible hunting oscillation, and that this behaviour is logically correct (i.e. increases with speed, decreases with curve radius). Other testing has shown that a bogie can be made to derail at speeds close to the Nadal Limit predictions on a range of curve radii, at speeds up to the maximum stable speed of 88mph, and does not derail on wider curve radii. Despite this, however, it is difficult to say with certainty that the simulation of the wheel/rail interface is truly accurate in the current version of the simulation tool. Additional data would have to be collected to make this assertion with more certainty. It would have been useful, for example, to test additional wheelsets - more complex profiles, different conicities etc. - and this was part of the original plan for the project, but it took too long to develop, evaluate and refine the simulation features presented in this Thesis. It was also not possible to conduct derailment tests on looped track with a single wheelset, as described in Section 5.9. The data presented in this Thesis suggests that this might not be the best application area for such a simulation tool, but that it may be more suitable to simulating bogies and whole vehicles. As discussed in Section 5.12, one reason for this is the fact that the bogie frame limits the wheelsets' degrees of freedom, resulting in a more stable simulation.

A Gauging Tool

Although limited time was available for gauge tests to be carried out, the tests presented in Chapter 6 (Section 6.7) have shown that the tool may be suitable for use as a gauge testing tool. The scenarios presented in this Thesis were simplified, but the results were very promising, corresponding well with the predictions, and should go on to work with more complex gauge testing scenarios, such as tunnels on sloped and curved track layouts. The Skin Width of the rigid bodies has to be taken into account. Perhaps the rigid bodies could be constructed so that they are slightly larger, so the shapes are closer to their true size when the Skin Width is applied. Alternatively, the tool could easily be programmed to take Skin Width into account when producing and outputting its data.

7.3 Conclusions

A rail dynamics simulation tool has been implemented using *NVidia's PhysX* Engine, and it has been shown to be capable of producing results that approach those predicted by other means. The following are general comments on the key results of this research.

7.3.1 *Straight Line Speed*

One of the goals of this research was to get the vehicle to travel at speeds of up to 100mph, but the vehicle was unable to reach this speed. Instead the best stable speed result, achieved with a single bogie, with Single Body wheelsets, modified simulation parameters and the application of additional centring forces to the wheels, was 88mph.

7.3.2 *Cornering Behaviour*

Although it was not possible to conduct testing on a single wheelset, testing with a single bogie on a looped track showed that the vehicle can be made to derail at speeds close to the predictions, within a small margin of error and a small standard deviation of results.

7.3.3 *Simulation Design Decisions*

A number design decisions were identified in Chapter 3 and evaluated in Chapter 5. These were evaluated, and the collected data was used to suggest the most suitably choices for simulating rail vehicle dynamics.

Single Body vs Multi Body Wheelsets

The first of these was the difference between the Single Body and Multi-Body wheelsets. The MB wheelset was more flexible, allowed for flange collision detection and provided more information about the forces acting on each wheel/flange of the wheelset. However, testing (in Sections 5.7 and 5.8) has shown that the MB Wheelset is significantly less stable than the SB wheelset, achieving a stable speed of 8mph compared to the SB Wheelset's 76mph (when tested under similar conditions).

Wheelset Polygon Count

Three variations of wheelset were designed with different polygon counts. The data collected in Section 5.7.6 showed that a higher polygon count produced more stable results, with only a minor reduction in the performance/framerate of the simulation. However, the data also suggests that increasing polygon count of the wheelset further was unlikely to produce significantly more stable results. The optimum results identified in these tests was a wheelset whose wheels/flanges were constructed from cylinders in 64 segments.

7.3.4 *Physics Engine Parameters*

The testing presented in Chapter 5 identified the following combination of physics engine parameters that produced the best results (highest straight line speed and most realistic cornering behaviour), according to the collected test data. These parameter values are:

- *Simulation Timing Multiple:* 20, which produces:
    - *Number of Substeps:* 160
    - *Timestep:* 0.000833̇
- *Rigid Body Solver Iteration Count (SIC):* 184
- *Joint Solver Extrapolation Factor (SEF):* 1.0
- *Skin Width:* 0.01

It is possible that different configurations are better suited to different applications. Gauge testing, for example, is less dependent on the accuracy of the wheel/rail interface simulation and does not necessarily require the vehicle to be able to travel at high speeds, and so the performance of the simulation during gauge testing could be improved by reducing these parameters.

7.3.5 *Performance*

The target framerate of the simulation was 60fps. By disabling Vertical Synchronisation (VSync), the simulation was capable of running at higher framerates (initially in excess of 220 FPS), and the effect of improving simulation fidelity on its performance could then be measured. On a 1,000m radius looped track, a single bogie executes at approximately 66 FPS, which is faster than real-time. A locomotive executes at 51 FPS (85% of real-time), two vehicles at 45 FPS (75% of real-time) and three vehicles at 40 FPS (67%). The tool is not capable of simulating whole vehicles in real-time, but their performance may still be considered ‘near-real-time’ (and represents a significant performance increase over the existing rail simulation tools).

7.3.6 *Setbacks and Delays*

Due to the issues described in Section 5.9 that prevented testing on curved track with a single wheelset, the Nadal limit predictions for a single wheelset were unable to be used. These would have produced more realistic predictions and results than the bogie tests.

The development and debugging of the simulation tool took significantly longer than expected, due to issues such as those described in Section 4.2. It also took longer than expected to refine the features necessary for the testing conducted in Chapters 5 and 6, and to carry out these tests and analyse the results. These delays prevented a detailed analysis of the results presented in Chapter 6. This was intended to be a more thorough examination of multi-vehicle trains, gauging and rapid prototyping features, but it was decided that the focus of the remaining time should be on the simulation of the wheel/rail interface.

7.4 Future Work

This section describes potential areas of research and development that could be conducted based on the simulation tool presented in this thesis.

7.4.1 *Further Research*

Further evaluation of the Locomotion tool may be possible, with only minor additional features and modifications.

Collecting Additional Data

Additional data could be collected from the current system. The tests from Chapter 5, for example, could be repeated on a larger range of curve radii, or using different track gauges and wheel profiles to see if the data is consistent with additional, revised predictions made using the Nadal Limit. It is also possible that there is additional data to be extracted from the PhysX engine, such as more detailed information about the contact points and forces, which not discovered during this research.

Additional Validation Data

It may also be possible to develop new tests of other behaviours of the wheelset/bogie/vehicles using the current version of the tool that can be validated using real-world data, data from the more advanced simulation tools or other mathematical predictions, if such data can be obtained. One potential example is described below.

- *Conical Wheelset Testing*

For example, if the necessary formulas/data/calculations could be attained, it might be possible to use the conical wheelset, as featured in the testing in Section 5.7, to further study the simulation of the wheel/rail interface in the Locomotion Tool.

The simulation could then be used to fine tune the spline-based centring technique to make the results as accurate and realistic as possible. During this research, the relevant formulas could not be obtained and there was no access to other simulation tools that would have allowed a more detailed evaluation of this behaviour.

Further Refinement of Parameters

It may also be possible to fine tune the results with further adjustments to the physics engine parameters. For example, another method - other than the timing multiple - could be used to adjust the simulation timing parameters. Or the Solver Iteration Count could be incremented in smaller steps (i.e. steps of 1 rather than 4 or 5 at a time) between batches in order to find the optimum results. Furthermore, the target speed of the vehicle could be incremented in smaller steps (i.e. 0.1mph between tests instead of 1mph), in order to more accurately determine the true stable speed and derailment speed of the vehicle. Time constraints prevented more extensive testing such as this during this research, as doing so would have required a significant number of additional tests to be conducted and for the results to be collected/analysed etc.

Improving the Spline-based Centring Technique

It is possible that further refinement of the spline-based centring technique could produce more consistent or more stable results. It would be necessary to conduct a more detailed study into the forces acting on the wheelset and how these compare to expected results, in order to calculate exactly the simulation differs from reality. Then, refinements could be made to the centring force, which may be able to produce more realistic results in a wider range of scenarios. This could include using the Conical Wheelset, as described above.

7.4.2 *Further Development*

After consultation with *NewRail,* a number of potential applications for a real-time rail dynamics simulation tool have been suggested, in addition to those discussed earlier. Some of these applications would require the implementation of additional features, and those discussed in this section are those that it is believed will be well-suited to the use of Physics Engines and game development techniques.

More Dynamic Vehicle Adjustment

The tool could be adjusted to allow for additional modifications to be made to the simulation/vehicle/testing environment in real-time. This would require further parametrisation of the simulation, as well as additional interface features. This would allow the user to make changes to more aspects of the simulation without having to exit the application, edit the configuration files and restart the application.

Extra Features

As mentioned in Section 2.1, bogie joints are designed to resist yawing motion, but this feature has currently not been included in the Locomotion tool. Additionally, some passenger trains also often have flexible walkways between the vehicles, which allow passengers to move from one carriage to another, while still allowing the necessary relative motion to enable proper cornering behaviour. It should be possible to add such features to the simulation with negligible impact on its performance.

- *Suspension*

A major feature currently missing from the simulation is suspension, which could be modelled in a number of ways, such as the use of more complex joint objects and/or springs (a common feature of physics engines [60]). Abstractions would have to be made to produce a suitable approximation of the system that will not compromise the real-time performance of the simulation, but a reasonable approximation should be possible, if the properties of the suspension system can be abstracted appropriately. This would increase the realism of whole vehicle simulation, as well as the simulation of the WRI.

Vehicle Gauging

It should be possible to extend the gauge testing features from Section 6.7 to add more complex models of infrastructure (tunnels, bridges, platforms, points etc.), without a significant impact on the performance of the simulation. It may also be possible to simulate other vehicles in order to ensure that collisions are avoided, though this is likely to impact the performance of the simulation significantly.

Canting

Canting has not been included in this simulation, but could be added to improve the realism of the simulation and to model real-world rail layouts. Generating the track geometry would be a more complex process, but simulating the track once it has been generated would not affect the performance of the simulation, if the track geometry does not have a significantly increased polygon count over the current track layouts.

AI Vehicle Control

It should be possible to write an Artificial Intelligence ('AI') system to control the train, in order to produce a more realistic approximation of dynamic vehicle behaviour in a complex rail network. For example, an AI 'driver' could be programmed to accelerate and decelerate for corners or stop in stations, just as a real train driver would. There could be a series of parameters to control this behaviour that the user could adjust in real time, or as part of the simulation's batch testing features, in order to, for example, determine the safest driver behaviour on a section of track that is prone to instability or derailment.

Buffers and Couplers

There is currently no simulation of buffers and a distance joint currently approximates the coupling between the vehicles. Additional entities would have to be added, possibly using springs, to approximate the buffers, but this could be done with only limited impact on simulation performance. Alternatively, for vehicles that do not use buffers, the damping effects of inter-carriage coupler systems, such as the walkways mentioned earlier in this section, could likely be approximated in an abstract way with only minimal impact on the performance of the simulation.

Decoupling Vehicles

The couplings between vehicles could conceivably break, for example in the aftermath of an explosion, and engineers might be interested in studying the effect of decoupling vehicles at speed. This could be done by adding a maximum force to the joints between the components, allowing them to break when the threshold is exceeded, or by allowing engineers to detach these components manually, or at specific points on a rail layout.

Articulated Vehicles and other Vehicle Types

Some vehicles, such as the 'Metrocar' vehicles used on the Tyne and Wear Metro system [61], use articulated vehicles. It should be relatively simple to simulate such vehicles without compromising the real-time performance of the simulation tool, if similar abstractions used in the development of the existing Locomotion virtual vehicle are used.

- *Freight and Passenger Vehicles*

The simulation already includes the ability to select from multiple vehicle types. The 'freight' locomotive and wagon have different dimensions, mass and centre of gravity properties from the 'passenger' locomotive and carriage that were used in earlier testing. A screenshot of a three-vehicle freight train, comprised of two freight locomotives and a freight wagon, is shown in Figure 7.1 (below).

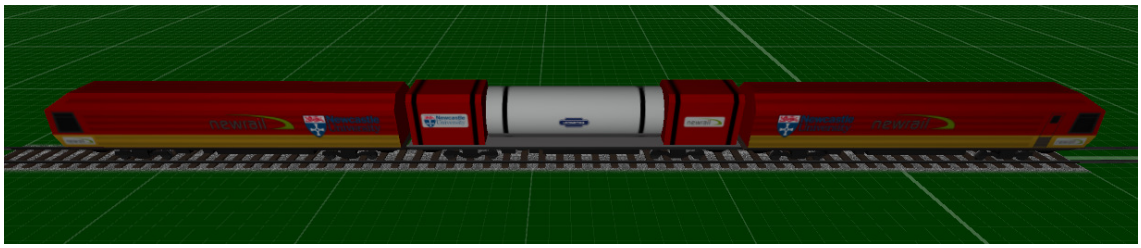


Figure 7.1 - Screenshot of Freight Vehicles in Locomotion

This feature could be extended to allow for a wider range of train/vehicle types to be simulated in the Locomotion tool.

More complex wheel profiles

A range of different and more complex wheel profiles could be constructed and used in the simulation, allowing more data to be collected and for comparisons to be made. However, more complex rigid bodies would reduce simulation performance. Alternatively, it may be possible to approximate more complex wheel profiles by using simplified physical models and adjusting the formulas used to calculate relative conicity and centring forces, allowing the effect of changing wheel profiles to be simulated in a way that will have less impact on the performance of the simulation.

Replay System

It should be possible to store the position of all of the objects in the system, which would allow the last few seconds off the simulation to be replayed. In the event of a derailment, for example, the simulation could allow engineers to step backward through the last few seconds of the simulation to study the instability and attempt to identify the cause of the derailment. It may even be possible to store the data in such a way that the entire test can be viewed at a later date, similar to the visualisations produced by VAMPIRE (see Section 2.3.3) or the PhysX Visual Debugger (Section 4.1.7).

Soft Body Dynamics

Some physics engines, including PhysX, include the ability to simulate material deformation via the use of 'Soft Bodies'. Soft Bodies are simplified, deformable mesh objects [21] and could be used to model flexible materials and add more realism to the simulation. The most useful application of this feature in the context of rail vehicles and multi-vehicle trains would be the addition of flexibility to the vehicle bodies in the current system, since this flexibility can have an effect on the vehicle's stability [8], particularly in the event of hunting oscillation or explosive blasts. It is likely that this would compromise the real-time performance of the simulation, but physics engine methods are designed to simulate such material deformation as efficiently as possible. An evaluation of the realism and performance of Soft Body material simulation in the physics engine would be required before it could be used in an engineering tool.

Comparing Physics Engines

It was originally intended that the simulation would be implemented using multiple physics engines, in order to make a comparison between them. Time constraints prevented this, but it should be possible to use an interface such as the *Physics Abstraction Layer* (as described in Section 2.4.4) to compare different physics engines to see which is best suited to modelling rail vehicle dynamics. It has already been shown that each engine has advantages and disadvantages in different circumstances [42] and a comparison between the different physics engines might be useful to determine which is best suited to the task of simulating rail vehicle dynamics and/or the wheel/rail interface.

- *PhysX Version 3.X*

It is possible that a new version of the PhysX engine might produce better results. PhysX version 3 was released in 2011 and is a major reworking of the physics engine which has '*undergone architecture and API improvements, code was cleaned and refactored, considerable legacy cluster was removed, new features were added*' [4] and so could be capable of producing different or more realistic results to version 2.8.4.

Access to Physics Engine Source Code

The use of an open-source physics engine, such as *Bullet*, *Newton* or the *Open Dynamics Engine*, would enable adjustments to be made to the internal workings of the engine, allowing it to be customised to suit the needs of rail engineers, rather than treating the internal workings as a black box and dealing only with the parameters, methods and callbacks that the physics engine API provides. Also, an open-source engine would provide access to more data about wheel/rail contacts and the forces applied to each wheel etc., which are not accessible through the 'off-the-shelf', closed-source version of *PhysX*.

- \

In 2015, shortly before the conclusion of this research, NVIDIA released the source code for the PhysX engine [62] (version 3). If the source code were to be obtained, then a more detailed study of the internal workings of the physics engine could be conducted, as described above (including determining the exact integration method used - information that is not available at time of writing).

Use of Specialist Physics Engines

As well as the general-purpose physics solutions like *Havok* and *PhysX*, there are a number of other systems such as *Digital Molecular Matter* (*DMM*) [63] and the *Euphoria Engine* [64] that are used in game development. *DMM* is designed to allow realistic looking materials, such as glass shattering, wood splintering and metal bending. Euphoria animates characters by *'generating motion on the fly – by simulating the character's motor nervous system, body and muscles'*. Game developers at *Lucasarts* were able to combine *Havok*, *DMM* and *Euphoria* in their game *Star Wars: The Force Unleashed*[11] without compromising the real-time performance of the game (though true realism in the physical behaviour of characters and objects was not the goal of this development).

[11] - this was based on information on the Lucasarts website in 2011, but an up-to-date reference on this game could not be found, since Lucasarts ceased trading in 2013

It may be possible to similarly integrate these engines into an engineering simulation. For example, if blast tests were added to the simulation then *DMM* could be used to show how the glass in the train's windows might shatter and be thrown from the vehicle, indicating the chance of injury to bystanders if the blast occurred in a station. *Euphoria* could be used to model virtual passengers sitting on seats (or moving about the carriage), to study the effect of events - such as derailments - or design changes on their safety and comfort.

Hardware Acceleration

It should be noted that, although *PhysX* supports hardware acceleration, the simulation currently does not (see discussion in Section 4.2.1). The simulation attempts to initialise *PhysX* with hardware acceleration and falls back on software simulation if that initialisation fails. However, this version of *PhysX* does not support the graphics processor that is installed on any of the systems on which it was developed and tested. However, also as discussed in Section 4.2.1, it is unclear whether this would improve the performance of the rigid body simulation, as, according to the PhysX documentation, hardware acceleration is mainly used for additional features of the physics engine.

7.4.3 *Supplementary Software Tools*

The following are suggestions for additional tools that could be developed for use in conjunction with a tool such as Locomotion. As with any simulation package, there are a number of input files required by the Locomotion tool, and generating these files is a time-consuming part of the process. These separate tools are used to generate these input files, as supplements to the main simulation.

Track Builder Tool

As described in Section3.2.4, track models have been generated by extruding the rail profile along splines (in 3DS Max) and it is conceivable that a tool for generating track geometry could be developed, allowing a greater range of layouts (which include hills, canting etc.) to be designed and optimised. The user would be able to select from a range of rail profiles and the geometry could be automatically, based on a number of configurable parameters such as curve radius and track gauge, and imported into the Locomotion tool. Such a system could, for example, allow the user to edit track layouts by adjusting the spline control points using a graphical interface.

Vehicle Builder Tool

VAMPIRE includes an Interactive Vehicle Builder, which provides a 3D graphical environment in which to construct vehicle models [29]. Such an application would provide an alternative to generating or modifying the models in 3D Studio Max or editing vehicle properties in a text file. It would provide a more intuitive, bespoke environment for rail engineers to quickly make adjustments to the vehicle settings.

It should be possible to automatically generate the geometry of the vehicle from various properties. The vehicles are comprised of relatively simple rigid bodies and so it should be possible to procedurally generate vehicles based on properties such as length, width and height, mass and centre of gravity, and wheelsets could be generated based on wheel spacing, conicity and material parameters.

Cloud-based Data Analytics

Another possible extension to the tool could make use of a cloud-based data analytics platform, such as e-Science Central [65]. The simulation tool could be configured to upload test data to such a platform, which could automate the process of analysing the data and producing many of the summary results that are currently produced by the LocoDataScan tool. This data could be stored in a database, or cloud data store, and reports or visualisations could be produced from that data. It would also be possible for the data visualisations to be displayed on a dynamic, interactive web page, eliminating the need for the user to install, or learn to use, specialist software.

7.4.4 *Other Potential Applications*

Based on the experience of developing Locomotion, the results presented in this thesis and discussion with NewRail engineers, the following additional potential applications of a real-time tool were discussed. These would require extensions to the locomotion tool, but are potentially useful application areas within rail dynamics that rail engineers may wish to study using a real-time simulation tool.

Accident Investigation

The tool could be used to reconstruct an accident in order to determine a possible cause. It would be necessary to extend the tool to allow data from the site of the accident to be used to reconstruct the environment and vehicles involved in the incident.

Post Derailment Behaviour

The simulation tool could be used - in additional to finding the point of derailment and study the conditions that led to it - to study what happens to the vehicle after derailment has occurred, and, for example, what damage might be caused to surrounding infrastructure.

Route design and optimisation

A real-time tool could help with the design and the optimisation of track routes. The tool would have to be extended with features to construct track layouts from real-world data, and to allow changes to be made to the track layout, which would require more sophisticated track layout creation and editing tools, such as by the use of the track builder tool described in the previous section.

Studying the Effects of Load Distribution

The way in which cargo is loaded onto a freight train can affect its stability. It is currently possible to simulate different load distributions in the Locomotion Tool by allowing the user to adjust the properties of the vehicle (such as mass and centre of gravity) and to show how doing so affects vehicle stability. An alternative system might be to create a virtual flatbed vehicle and cargo containers of different sizes and mass properties, and allowing the user to load the virtual cargo onto the vehicle in different ways. This would allow users to alter the load distribution of a vehicle in the simulation in a more intuitive way, possibly via a 3D graphical interface, and would allow the user to optimise the loading efficiency of a vehicle while also ensuring safety.

Studying Dynamic Loads / Fluids

It may also be possible to implement some basic fluid dynamic simulation to evaluate the effect of liquid cargo on the stability of the vehicle. It would also be possible to adjust properties such as the centre of gravity to simulate the movement of passengers or other dynamic cargo about the vehicle in an abstract way.

Effects of Gust Loading or Extreme Winds

It should be possible to add a basic approximation of wind and/or air resistance into the simulation, for example by applying forces to the vehicle based on the direction and strength of the wind. This would enable engineers to study, for example, how strong winds affect vehicle stability while the vehicle is in motion.

Occupant Simulation

Ragdoll Physics, a common feature of Physics Engines [21], could be used to simulate virtual passengers on to the train. Engineers would then be able to analyse how passengers are affected during explosions, derailments or crash events. It may even be possible to have animated passengers moving about the cabin (without requiring an additional, specialist physics engine like Euphoria - discussed in Section 7.4.2). However, doing so is likely to comprise the real-time performance of the simulation.

Fire & Smoke Modelling

Fire and smoke in video games are often simulated using particle effects [21]. A particle system, if constructed using real-world properties, could conceivably be used to simulate the spread of fire and smoke through a carriage in an emergency situation (including while the vehicle is in motion and in high winds), conditions that are difficult to evaluate in real-world safety tests, such as those described in Section 2.3.1.

Explosion Modelling

Hunting and Wheel-Climb Derailment can occur in the aftermath of an explosive blast or other event, if the event introduces a lateral component into the vehicle's motion. Additionally, there is another form of derailment, known as 'Impact derailment', which is caused by collision or explosion that causes the wheel to jump onto the railhead or derail completely [8]. Point load forces could be applied to the vehicle to simulate the effect of a blast in different parts of the vehicle. A basic implementation has already been attempted. A more-detailed model was constructed with a hollow frame and detachable doors and windows, show in Figure 7.2 (below).

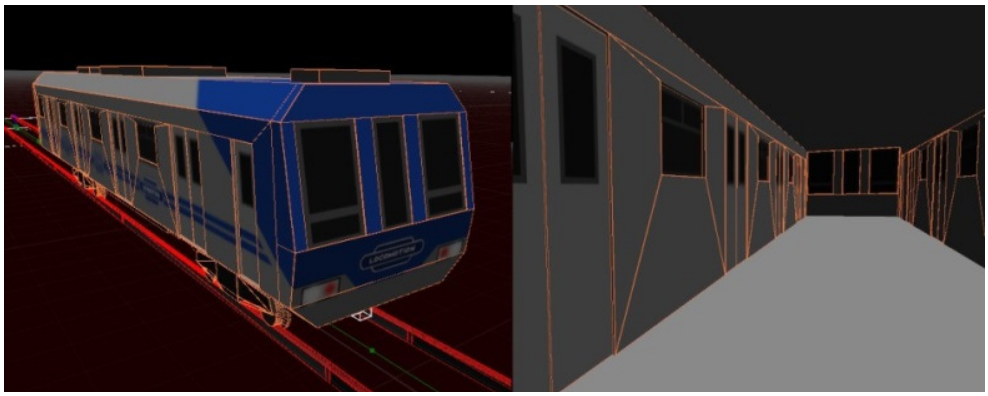


Figure 7.2 - Exterior (left) and Interior (right) of more complex carriage model

The shockwave (represented by the yellow area in Figure 7.3, below) is then simulated as an expanding sphere and forces are applied to the vehicle and its components based on the position of the objects relative to the epicentre of the blast.

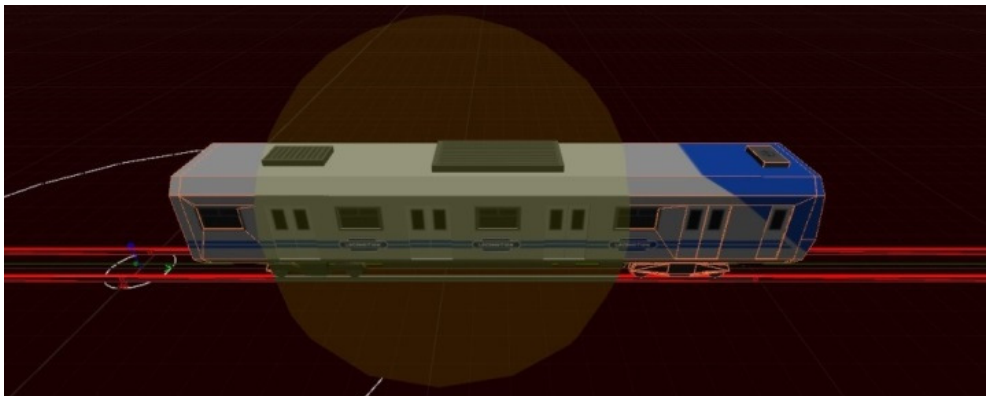


Figure 7.3 - A prototype Explosion in Locomotion

Such features would allow engineers to study the effect on vehicle stability and probability of derailment while the vehicle is in motion, and to adjust the design of the vehicle to compensate.

7.5 Contributions

The contributions of this research were outlined in Section 1.7. Below is a discussion of these contributions in relation to the results presented in this thesis.

7.5.1 *Development of a Real-time Rail Dynamics Simulation Tool*

This thesis has described the development of Locomotion, a real-time rail dynamics simulation tool based on the PhysX Engine. The simulation is based on real world vehicles and properties. The simulation is capable of performing 100 tests with a single vehicle on a looped track layout in 9 to 12 hours (depending on factors such as the curve radius). This is shorter than an individual test could potentially take in the more sophisticated simulation tools and represents a significant increase in speed over traditional methods.

7.5.2 *An Evaluation of a Physics Engine-based Tool for Engineering Use*

As discussed in Chapter 5, PhysX, with its default parameters, was not able to produce realistic results, but these results were improved, as discussed in the next Section. The tool was able to produce logically correct sinusoidal lateral motion with the conical wheelset (Section 5.7.3), achieve a peak stable speed of 88mph using adjusted parameters (Section 5.12) and derail at speeds close to those predicted by the Nadal Limit in bogie testing on curved track with the addition of the spline-based centring forces (Section 5.13). The range of results in these final tests was quite high, as much as 19.3 mph, but the standard deviation was, on average, just less than 6mph (4%).

The data suggests that the simulation is perhaps not as well suited to interactive, real-time simulation of the WRI as was originally hoped. In its current state, the simulation is not suited to high speed testing, but is capable of reaching speeds of approximately 88mph. The simulation may be suited to lower speed testing <80mph, but may not be suited to extreme low speed testing (<5mph) as this has also been shown to be unstable in several tests.

The range of results that it can produce, and the variance in results between tests, means that any individual test run in the simulation may not produce accurate results. However, the results show that if an average is taken over a large number of tests, then the average is close to the intended results. Running multiple tests and collecting averages helps to minimise the negative impact of any individual erroneous results that may be produced by the simulation, and the real-time performance of the tool allows large numbers of tests to be completed rapidly.

It is worth noting that the Nadal Limit is often considered to be a conservative estimate and so this simulation can be considered to be conservative as well, as most of the results were slightly below the Nadal predictions.

7.5.3 *Improving the Simulation of the Wheel/Rail Interface*

Key parameters of the engine were adjusted in an attempt to improve this behaviour, but while there was some improvement, these adjustments were ultimately unable to produce derailment speeds close to those predicted using the Nadal Limit.

It was necessary to develop a new real-time simulation technique; the spline-based wheelset centring method (described in Section 3.3.14), which was designed to replicate the gravitational stiffness force. Testing has shown that this spline-based method of manually applying the GSF to each wheelset improved the results considerably, but was still not producing realistic results. The size of the centring force had to be multiplied to produce results closer to the predictions, but it is based on a real-world phenomenon and published rail engineering formulas. For track curves of between 100 and 200m in radius; the averages over 10 tests were within 5% of the predictions, off by approximately 2% on average, and no derailments at its maximum stable speed on wider radii curves.

Further development of the centring technique is recommended. It is possible that the solution is capable of producing more realistic results in a wider range of scenarios, if this calculation can be refined correctly. It will be necessary to develop a more sophisticated method for calculating the size of the centring force, without using the 'force multiple'. It may be that, for example, other forces (such as creepages) or other properties of the wheelset and/or rails can be used to eliminate the multiple from the calculation and improve the results. It is also possible that the application of additional, non-real-world forces is necessary in order to compensate for errors introduced by the physics engine, and so a more detailed evaluation of this error would be necessary.

A wheelset was tested on straight track and allowed the simulation to be fine-tuned to increase the top speed of the wheelset/vehicle; however, cornering behaviour was unable to be tested, meaning that the Nadal Limit predictions for a single wheelset could not be used. Although it may be of limited use for engineers to simulate an individual wheelset in this way, it would have been useful for this research to be able to simulate the behaviour of a single wheelset in order to confirm that this relatively simple (and easy to predict) behaviour was realistic with respect to the Nadal Limit predictions.

7.5.4 *A Simulation of Multi-Vehicle Train Behaviour*

Further data is needed to validate the results of tests on whole vehicles, but although not validated, this Thesis demonstrates examples of the sort of useful data that the tool can generate (Chapter 6), including data from the simulation of two and three vehicle trains. The application has been tested with up to 9 vehicles, though simulating this many vehicles has a significant impact on the performance of the application. Time constraints prevented a more detailed evaluation of these behaviours during this research.

7.5.5 *Alternative Applications*

The testing conducted in this Thesis has allowed a discussion of the potential suitability of a real-time simulation tool for a number of potential applications within rail vehicle dynamics. In addition to the simulation of the wheel/rail interface and multi-vehicle trains, which have already been discussed, these application areas include:

- A Gauging Tool
- A Rapid Prototyping Tool

As discussed in Section 7.3, the suitability for these application areas has been discussed based on the data presented in Chapters 5 and 6. The simulation of the WRI in the evaluated circumstances was considered suitably accurate, but even if this was not the case, the tools suitability for simulating multi-vehicle trains in certain circumstances, such as rapid prototyping and gauge testing, is promising.

7.5.6 *Discussion of Possible Applications of a Real-time Simulation Tool*

Based on the data collected in Chapters 5 and 6 and a discussion with NewRail engineers, a number of possible applications of a real-time simulation tool, in addition to those listed above, are suggested in Section 7.4, including: explosion modelling, the addition of soft body dynamics, passenger simulation and the study of cargo distribution and dynamic loads. Many of these applications would be suitable for implementation using the features of physics engines, or other techniques used in the games industry, as discussed in Section 7.4.

7.6 Thesis Summary

This thesis has presented details of the design, implementation and evaluation of a rail dynamics simulation tool based on a real-time physics engine. It has shown that the simulation can produce accurate results for the wheel/rail interface within what rail engineers might consider to be an acceptable margin of error for real-time testing.

This Thesis has shown how the physics engine can be integrated into the simulation, and has shown how a range of physics engine parameters can be configured to produce the best results, as well as which design variations are most suitable for use in the simulation of the wheel/rail interface. This Thesis has also shown the sort of data that the simulation tool/physics engine is capable of producing, including its potential usefulness as a gauging tool and a rapid prototyping tool.

The error introduced by the physics engine means Locomotion may not produce reliable results from an individual test, but the real-time performance of the tool allows multiple tests to be executed, allowing averages to be calculated quickly.

The variations in the results are assumed to be a side effect of the way that the solver handles collisions and joints between rigid bodies, and it is possible that further adjustments to simulation parameters, physics models or properties of the rigid bodies would improve the results further and would make the wheelset more stable, allowing it to accelerate to higher speeds and producing more realistic cornering behaviour in a wider range of scenarios.

From the results presented in this Thesis, it is possible to conclude the following. It is possible to use Physics Engines to develop real-time rail vehicle simulations that are capable of producing reasonably realistic results that approach those predicted using traditional mathematical methods for certain application areas within rail dynamics. This suggests that there is significant potential for further research and development in this area. These results show that physics engines have the potential to produce results that are accurate to within a small error bound in certain scenarios and test conditions, if properly configured and augmented. It is not currently suited for testing at high speeds, or for derailment testing on wide curve radii (larger than 200m), as it cannot exceed 88mph due to instability, but could be used for testing at lower speeds, or for a number of other functions.

Validation has been one of the biggest challenges in this project and it has been very difficult to find or derive appropriate mathematical prediction formulas, or to attain test data from existing simulations or real-world tests, to use in the evaluation of the tool. However, test scenarios have been benchmarked against mathematical predictions, including the Nadal Limit; a widely used benchmark in the rail industry. Suggestions for additional development and evaluation of the simulation tool have been put forward, that could expand upon the results of this research, if the appropriate benchmarking data can be obtained. This includes further evaluation and refinement of the spline-based centring technique, which may be capable of producing realistic results in a wider range of circumstances.

Further research, development and validation are necessary if the simulation tool is to become a viable engineering product. It will be necessary to see if the simulation can be further refined and if the results produced by the simulation can be verified across a wider range of testing scenarios, but this initial research shows that this application area has considerable potential.

References

[1] NewRail, “NewRail,” NewRail, [Online]. Available: http://www.ncl.ac.uk/newrail/. [Accessed August 2016].

[2] NewRail, “NewRail Rail Vehicles Group,” NewRail , [Online]. Available: http://www.ncl.ac.uk/newrail/research/msdg/. [Accessed August 2016].

[3] J. Carruthers, *Real-time Simulation of Complex Engineering Scenarios (Appendix 1),* 2011.

[4] NVIDIA, “NVIDIA PhysX,” NVIDIA, [Online]. Available: http://geforce.com/hardware/technology/physx. [Accessed August 2016].

[5] JR-Central, “About the Shinkansen,” Japan Railways Group, [Online]. Available: http://english.jr-central.co.jp/about/outline.html. [Accessed August 2016].

[6] Voith, “The Scharfenberg Operating Principle,” Voith, [Online]. Available: http://www.voith.com/en/products-services/power-transmission/scharfenberg-couplers/the-scharfenberg-operating-principle-13784.html. [Accessed August 2016].

[7] A. Orlova and Y. Boronenko, “The Anatomy of Railway Vehicle Running Gear,” in *The Handbook of Railway Vehicle Dynamics (Chapter 3)*, Taylor & Francis, 2006, pp. 39-84.

[8] S. Iwnicki, Handbook of Railway Vehicle Dynamics, Taylor & Francis Group, 2006.

[9] Tata Steel, “Rail Technical Guide,” 2014. [Online]. Available: http://www.tatasteelrail.com/. [Accessed August 2016].

[10] J.-B. Ayasse and H. Chollet, “Wheel-Rail Contact,” in *Handbook Of Railway Vehicle Dynamics, Chapter 4*, Taylor & Francis, 2006, pp. 85-120.

[11] Encyclopædia Britannica, “Encyclopædia Britannica- Standard gauge,” Encyclopædia Britannica, Inc., [Online]. Available: http://www.britannica.com/technology/standard-gauge. [Accessed August 2016].

[12] S. Iwnicki, “Simulation of Wheel-Rail Contact Forces,” *Fatigue & Fracture of Engineering Materials & Structures,* vol. 26, no. 10, p. 1031–1029, 2003.

[13] H. Wu and N. Wilson, “Railway Vehicle Derailment and Prevention,” in *Handbook of Railway Vehicle Dynamics (Chapter 8)*, Taylor & Francis, 2006, pp. 209-238.

[14] J. Evans and M. Berg, “Challenges in the simulation of rail vehicle dynamics,” *Vehicle System Dynamics,* vol. 47, no. 8, pp. 1023-1048, 2009.

[15] W. Yan and F. Fischer, “Applicability of the Hertz contact theory to rail-wheel contact problems,” *Archive of Applied Mechanics,* vol. 70, no. 4, pp. 255-268, 2000.

[16] J. Kalker, “Wheel-rail rolling contact theory,” *Wear,* vol. 144, no. 1-2, pp. 243-261, 1991.

[17] J. J. Kalker, “A Fast Algorithm for the Simplified Theory of Rolling Contact,” *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility,* vol. 11, no. 1, pp. 1-13, 1982.

[18] J. Nadal, “Theories de la Stabilite des Locomotives,” *Annales des Mines,* vol. 10, no. 232, 1896.

[19] B. Marquis and R. Greif, “Application of Nadal Limit in the Prediction of Wheel Climb Derailment,” *2011 Joint Rail Conference,* pp. 273-280, 2011.

[20] SECUREMETRO, “SECUREMETRO,” SECUREMETRO Partners, 2013. [Online]. Available: http://securemetro.inrets.fr/. [Accessed August 2016].

[21] D. M. Bourg, Physics for Game Developers, O'Reilly Media, 2002.

[22] D. Eberly, Game Physics (2nd Edition), Taylor & Francis, 2010.

[23] The MathWorks, Inc., “MatLab,” The MathWorks, Inc., [Online]. Available: http://www.mathworks.co.uk/products/matlab/. [Accessed August 2016].

[24] J. W. Eaton, “GNU Octave,” GNU, [Online]. Available: https://www.gnu.org/software/octave/. [Accessed August 2016].

[25] Altair Hyperworks, “RADIOSS,” Altair Hyperworks, [Online]. Available: http://www.altairhyperworks.com/Product,51,RADIOSS.aspx. [Accessed August 2016].

[26] Livermore Software Technology Corp., “LS-DYNA,” Livermore Software Technology Corp., [Online]. Available: http://www.lstc.com/products/ls-dyna. [Accessed August 2016].

[27] ESI Group, “PAM-CRASH,” [Online]. Available: http://www.esi-group.com/products/crash-impact-safety/pam-crash.

[28] TASS International, “MADYMO,” TASS International, [Online]. Available: http://www.tass-safe.com/en/products/madymo. [Accessed August 2016].

[29] DeltaRail Group Ltd., “VAMPIRE,” DeltaRail Group Ltd., [Online]. Available: http://vampire-dynamics.com/. [Accessed August 2016].

[30] Rail Accident Investigation Branch, “Derailment of a passenger train near Liverpool Central station, 26 October 2005,” Rail Accident Investigation Branch, [Online]. Available: https://www.gov.uk/raib-reports/derailment-near-liverpool-central-underground-station. [Accessed August 2016].

[31] O. Polach, “A Fast Wheel-Rail Force Calculation Computer Code,” *Vehicle System Dynamics 33,* pp. 728-739, 1999.

[32] MSC Software, “Adams/Rail,” MSC Software, [Online]. Available: http://www.mscsoftware.com/product/adams. [Accessed August 2016].

[33] W. Wang and G. Li, “Development of a high-speed railway vehicle derailment simulation - Part 1: A new wheel/rail contact method using the vehicle/rail coupled model,” *Engineering Failure Analysis,* vol. 24, pp. 79-92, 2012.

[34] W. Wang and G.-x. Li, “Development of high-speed railway vehicle derailment simulation - Part II: Exploring the derailment mechanism,” *Engineering Failure Analysis,* vol. 24, pp. 93-111, 2012.

[35] A. Anyakwo, C. Pislaru and A. Ball, “A New Method for Modelling and Simulation of the Dynamic Behaviour of the Wheel-rail contact,” *International Journal of Automation and Computing,* vol. 9, no. 3, pp. 237-247, 2012.

[36] M. e. a. Monga, “Real-time Simulation of Dynamic Vehicle Models using a High-performance Reconfigurable Platform,” *Proceedings of the International Conference on Computational Science (ICCS),* 2012.

[37] Havok, “Havok Physics,” Havok, [Online]. Available: http://www.havok.com/. [Accessed August 2016].

[38] e. a. Erwin Coumans, “Bullet Physics,” [Online]. Available: http://bulletphysics.org/. [Accessed August 2016].

[39] J. Jerez, A. Suero and e. al., “Newton Physics SDK,” Newton Dynamics, [Online]. Available: http://newtondynamics.com/. [Accessed August 2016].

[40] R. Smith, “Open Dynamics Engine,” 2007. [Online]. Available: http://www.ode.org/. [Accessed August 2016].

[41] NVIDIA, “PhysX v2.8 Documentation (Installed with the SDK),” 2011.

[42] Boeing and Braunl, “Evaluation of real-time physics simulation systems,” *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia,* pp. 281-288, 2007.

[43] World Weaver, “DX Studio,” World Weaver, [Online]. Available: http://www.dxstudio.com/. [Accessed August 2016].

[44] Autodesk, “3DS Max Overview,” Autodesk, [Online]. Available: http://www.autodesk.co.uk/products/3ds-max/overview. [Accessed August 2016].

[45] ActiveWorlds, Inc, “Active Worlds,” ActiveWorlds, Inc, [Online]. Available: https://www.activeworlds.com/web/index.php. [Accessed August 2016].

[46] Microsoft, “Microsoft Robotics Developer Studio,” Microsoft Corporation, [Online]. Available: https://msdn.microsoft.com/en-us/library/bb483024.aspx. [Accessed August 2016].

[47] F. e. a. Luo, “Intelligent Vehicle Simulation and Debugging Environment based on Physics Engine,” *2009 International Asia Conference on Informatics in Control, Automation and Robotics,* pp. 329 - 333, 2009.

[48] Metro de Madrid, “Metro de Madrid,” Metro de Madrid, [Online]. Available: https://www.metromadrid.es/en/. [Accessed August 2016].

[49] BBC News, “Madrid Train Attacks,” BBC, [Online]. Available: http://news.bbc.co.uk/1/shared/spl/hi/guides/457000/457031/html/. [Accessed August 2016].

[50] A. Anyakwo, C. Pislaru, A. Ball and F. Gu, “Modelling the Dynamic Behaviour of the Wheel Rail Interface Using a Novel 3D Wheel-Rail Contact Model,” 2011. [Online]. Available: http://eprints.hud.ac.uk/13466/. [Accessed May 2012].

[51] RailwayTechnical.com, “RailwayTechnical.com,” Railway Technical Web Pages, [Online]. Available: http://www.railway-technical.com/. [Accessed August 2016].

[52] UrbanRail.net, “UrbanRail.net,” UrbanRail.net, [Online]. Available: http://www.urbanrail.net/eu/es/mad/madrid.htm. [Accessed August 2016].

[53] cplusplus.com, “C++ Standard Library,” cplusplus.com, [Online]. Available: http://www.cplusplus.com/reference/. [Accessed August 2016].

[54] cplusplus.com, “C++ Standard Template Library (STL) Reference,” cplusplus.com, [Online]. Available: http://www.cplusplus.com/reference/stl/. [Accessed August 2016].

[55] Microsoft, “Visual Studio,” Microsoft, [Online]. Available: https://www.visualstudio.com/. [Accessed August 2016].

[56] SGI, “OpenGL,” SGI, [Online]. Available: https://www.opengl.org/.

[57] F. Devernay, “GLM: an Alias Wavefront OBJ file library,” [Online]. Available: http://devernay.free.fr/hacks/glm/. [Accessed August 2016].

[58] SDL, “SDL,” SDL Community, [Online]. Available: https://www.libsdl.org/. [Accessed August 2016].

[59] PhysLink.com, “Friction Coefficients - PhysLink.com,” PhysLink.com, [Online]. Available: http://physlink.com/Reference/FrictionCoefficients.cfm. [Accessed August 2016].

[60] G. Palmer, Physics for Game Programmers, Apress, 2005.

[61] Nexus, “Tyne and Wear Metro - Train Fleet Refurbishment,” Nexus, [Online]. Available: http://www.nexus.org.uk/news/item/metro-train-fleet-refurbishment-completed. [Accessed August 2016].

[62] NVIDIA, “PhysX Source Code,” NVIDIA, [Online]. Available: https://developer.nvidia.com/content/latest-physx-source-code-now-available-free-github. [Accessed August 2016].

[63] Pixelux Entertainment, “Digital Molecular Matter,” Pixelux Entertainment, [Online]. Available: http://www.pixelux.com/DMMengine.html. [Accessed August 2016].

[64] NaturalMotion, “Euphoria Engine,” NaturalMotion, [Online]. Available: http://www.naturalmotion.com/middleware/euphoria/. [Accessed August 2016].

[65] Digital Institute, Newcastle University, “e-Science Central,” Digital Institute, Newcastle University, [Online]. Available: http://www.esciencecentral.co.uk/. [Accessed August 2016].

[66] J. Carruthers, *Real-time Simulation and Visualisation of Rail Vehicle Safety and Security (Appendix 2),* 2011.

[67] K. Johnson, Contact Mechanics, Cambridge University Press, 1985.

[68] Bombardier, “Bombardier,” Bombardier, [Online]. Available: http://uk.bombardier.com/en/transportation.html. [Accessed August 2016].

[69] EA Games, “Need for Speed: Shift,” EA Games, [Online]. Available: http://www.needforspeed.com/en_GB/shift. [Accessed August 2016].

[70] RRUK, “Rail Research UK Association,” RRUK, [Online]. Available: http://rruka.org.uk/. [Accessed August 2016].

[71] Transport Research and Innovation Portal, “Securemetro,” Transport Research and Innovation Portal, [Online]. Available: http://www.transport-research.info/web/projects/project_details.cfm?ID=41640. [Accessed August 2016].

[72] A. Boeing, “Physics Abstraction Layer (PAL),” [Online]. Available: http://www.adrianboeing.com/pal/index.html. [Accessed August 2016].

[73] Klingel, “Uber den Lauf der Eisenbahnwagen auf gerader Bahn (On the running of railway vehicles on straight track),” *Organ für die Fortschritte des Eisenbahnwesens XX,* 1883.

[74] U. Olofsson and R. Lewis, “Tribology of the Wheel-rail Contact,” in *Handbook of Railway Vehicle Dynamics (Chapter 5)*, Taylor & Francis, 2006, pp. 121-142.

[75] O. Polach, M. Berg and S. Iwnicki, “Simulation,” in *Handbook of Railway Vehicle Dynamics (Chapter 12)*, Taylor & Francis, 2006, pp. 359-422.

[76] Altair Engineering, “Visual Solutions,” Altair Engineering, [Online]. Available: http://www.vissim.com/. [Accessed August 2016].

[All links correct as of August 2015]

Appendix 1

The following document was provided by NewRail engineer Dr Joe Carruthers at the start of the project in 2011.

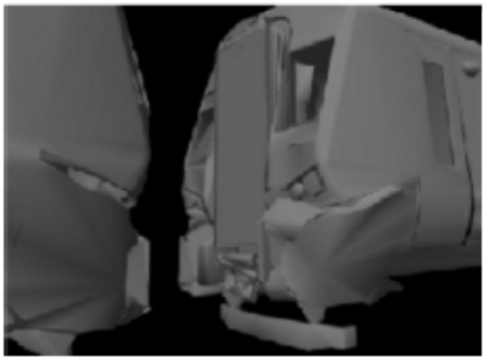
(J. Carruthers, “Real-time Simulation of Complex Engineering Scenarios,” 2011.) [3]

Newcastle University

NewRail

Project:

Real time simulation of complex engineering scenarios



Project Leader(s): Dr. Joe Carruthers
Contact: joe.carruthers@ncl.ac.uk

One of the frustrations with conventional numerically-based tools for engineering simulation, such as finite element analysis, is the length of time that it can take such models to solve. This is particularly true when modelling very complex scenarios such as vehicle collisions, which are inherently transient and non-linear by their very nature. With finite element crash simulations typically taking hours or even days to solve, iterative design optimisation can become a slow process.

Therefore, as a complement to existing engineering analysis software, NewRail is currently developing alternative simulation technologies that are capable of providing ballpark estimations in real time or near real time. These facilitate the very rapid evaluation of design concepts, particularly for the early stages of a design process in which the rapid approximate evaluation of a wide range of design options is often more important than absolute precision.

Staff

Dr Joe Carruthers **(http://www.ncl.ac.uk/newrail/people/profile/joe.carruthers)** Rail Vehicles Group Manager	Email: joe.carruthers@ncl.ac.uk (mailto:joe.carruthers@ncl.ac.uk) Telephone: +44 (0)1246 281634

NewRail - Newcastle Centre for Railway Research
Faculty of Science, Agriculture and Engineering, Newcastle University, Newcastle upon Tyne
NE1 7RU, United Kingdom.
Email Webmaster (mailto:karen.mctigue@ncl.ac.uk)

Appendix 2

The following document was provided by NewRail engineer Dr Joe Carruthers at the start of the project in 2011.

(J. Carruthers, “” 2011.) [66]

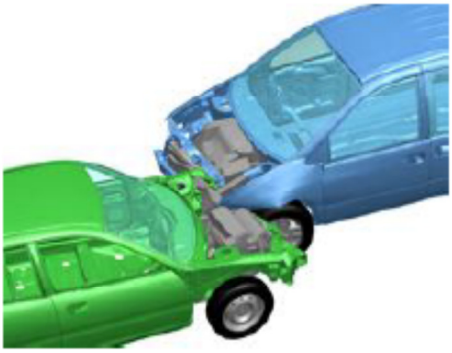
Working title

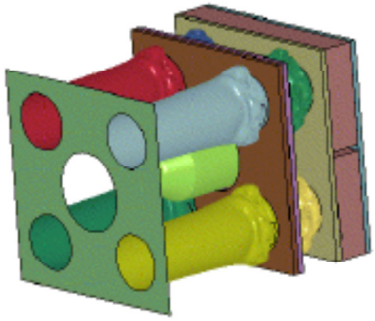
"Real-time simulation and visualisation of rail vehicle safety and security"

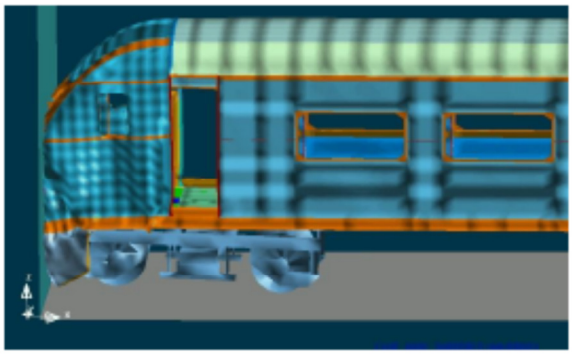
Overview

I'd like us to work on the development of (near) real-time alternatives to existing types of engineering analysis. To include some (if not necessarily all) of the following:

1. ***Transient, non-linear analysis for the modelling of large structural deformations, as in crash scenarios.***
    - Typical engineering software packages: LS-DYNA, PAM-CRASH, RADIOSS.

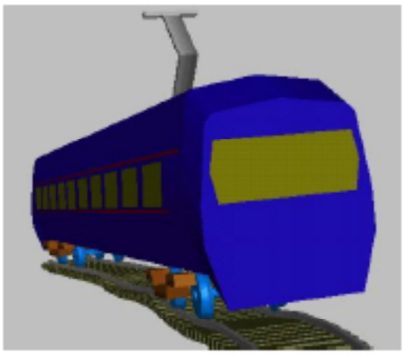






- Response MUST be related to the underlying material properties and structural geometry, at least in ballpark terms (that's what makes it an engineering PhD!) – density, stiffness, strength, etc. I'm not saying that we should be able to differentiate between different grades of steel, but it would be nice to be able to differentiate between a metal and a polymer, and it would be fantastic if we could differentiate between steel and aluminium (for example). Suggest we start with ductile materials (e.g. steel, polypropylene), before moving on to more brittle ones that will introduce complexities such as fracture.

2. *Physics-based model of the dynamics of a rail vehicle running on a track*
   - Typical engineering software packages: VAMPIRE

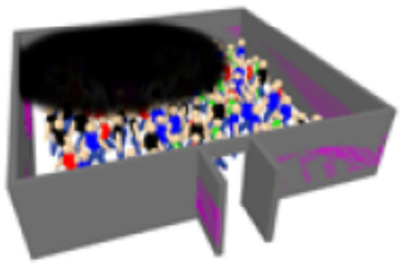


   - Representative wheel profile; representative rail profile; traction delivered through the wheel/rail interface; steering delivered through the wheel/rail interface; adhesion/wheel slip; derailment; representative suspension behaviour (so that body movement can be simulated for advanced gauging analysis).
   - My thinking is that the simulation-end of motor racing games (Live for Speed, iRacing, rfactor, etc.) have pretty much nailed a lot of this in automotive terms, and compared to us they have the added complexities of highly non-linear temperature/wear dependent tyre models, and much less certainty of vehicle positioning, angle of attack through corners, etc. If anything, a physics-based rail vehicle model ought to be more straightforward / less computationally intensive than an automotive one.
   - Simulating derailment would be a major long term objective in this area.

3. *Occupant simulation under crash conditions*
   - Typical engineering software packages: MADYMO.

- Continuation of student final year project work.
- There appears to be real interest from Bombardier (train builder) in this work.
- We could do some more extensive calibration of the gaming tech against the engineering software and/or crash dummy tests.
- Combining such occupant analyses with activity 1 (vehicle crash simulation) would be a fantastic achievement.

4. *Fire modelling*
    - Typical engineering software packages: Fire Dynamics Simulator (FDS).



- I need to think this one through a bit more, but the basic idea might be to try and control something like the particle emitters in Unreal Engine 3 (or whatever) using physics-based input parameters. NewRail is very strong in the area of fire modelling, and I think there could be some mileage in this.

5. *Real-time calculation and visualisation of vehicle gauging.*
    - i.e. predicting the clearance between a moving rail vehicle and trackside objects (tunnels, bridges, etc.).
    - We have already made a start in this area.
    - In the longer term, it would be nice to couple it to activity 2 (vehicle dynamics model).