Performance Measures and Control Laws for Active and Semi-Active Suspensions

by
Ian Storey

Submitted to
The School of Aerospace, Mechanical & Manufacturing Engineering
in fulfilment of the requirements for the degree of
Doctor of Philosophy
at
The Royal Melbourne Institute of Technology (RMIT)
October 2011

This thesis is my original work and has not been submitted previously, in whole or in part, to qualify for any other academic award at this or any other university. Nor does it contain, to the best of my knowledge and belief, any material published or written by another person, except as acknowledged in the text. The content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Signature of Author_____________________________________________________

25 September 2011
Acknowledgements

First and foremost I would like to thank my supervisors. Doctor Anna Bourmistrova has showed patience, endurance and understanding. Her critique kept the research on track and smoothed the bumpy ride. She has been a superb supervisor and the work could not have been completed without her capable direction. Professor Aleksandar Subic accepted my original proposal, and agreed to act as second supervisor, for which I am extremely grateful. His comments and editing have helped in the evolution of the work and have optimized the results.

David Holinger, Ben Lancini and Doctor Hossein Zadeh helped with the decision to use a microcontroller for proof of concept. Hossein suggested the AVR microprocessor.

Ian Searle helped solve the problem of 50 Hz interference. Professor Ross Smith spurred me on to find proofs. Ian Searle, Doctor Hossein Zadeh, Professor Ross Smith and Frans Koomen all courageously and patienty discussed more than a few sticky issues during the course of the research. Christine Hope provided assistance with research material.

Thanks are due also to my friends David Colechin and Joan Richardson who made very helpful critiques on edits. I would especially like to thank Gergory Plumb who made many useful editing suggestions.

I would like to thank my parents, my brother Phillip and his family, and especially my darling wife, Kate, who have all been supportive and have had to share in the sacrifices needed to complete such a large research undertaking. Kate’s family also helped enormously, and Neville Hard showed great patience in “taking the show on the road”.

Peter Tkatchyk helped with the experimental rig frame and produced an extremely fine result from my rough blueprint (a technician with skills that I envy). Sebastian Naselli provided great assistance with electronic components.
I would like to thank the RMIT Business Department generally, and the Business IT and Logistics School in particular, which provided leave for the thesis completion.

Last but not least, I would like to thank the examiners for their time and attention.

Ian Storey
## Contents

1. Introduction ............................................................................................................. 4
  1.1. Problem Definition ............................................................................................ 11
  1.2. Theory Overview ............................................................................................... 13
  1.3. The Contribution of this Thesis ........................................................................ 18
  1.4. Thesis Chapter Outline ..................................................................................... 21

2. Literature Review .................................................................................................... 23
  2.1. Design Approaches ........................................................................................... 23
  2.2. Dynamics ........................................................................................................... 24
    2.2.1. Quarter-Car Dynamics ............................................................................. 24
    2.2.2. Passive Suspension Design ..................................................................... 26
  2.3. Control Techniques .......................................................................................... 29
    2.3.1. Linear Control ........................................................................................... 31
      2.3.1.1. Modelling Discretised Linear Control ............................................ 35
      2.3.1.2. Kalman Filter .................................................................................... 36
    2.3.2. Nonlinear Control ..................................................................................... 38
      2.3.2.1. Stochastic Input .................................................................................. 41
      2.3.2.2. Performance Measures .................................................................... 42
      2.3.2.3. Minimum-Time Problem ................................................................ 43
      2.3.2.4. Sliding-Mode Control ..................................................................... 48
  2.4. Skyhook ........................................................................................................... 50
  2.5. Groundhook ..................................................................................................... 54
  2.6. Passivity Constraint ......................................................................................... 55
  2.7. Semi-active Skyhook ....................................................................................... 58
  2.8. Road Surfaces .................................................................................................. 59
  2.9. Suspension Design Goals ................................................................................ 62
    2.9.1. Comfort ...................................................................................................... 63
    2.9.2. Vertical Travel .......................................................................................... 65
    2.9.3. Cumulative Absorbed Power ................................................................... 66
    2.9.4. Road Holding ......................................................................................... 66
    2.9.5. Driving “Feel” .......................................................................................... 66
    2.9.6. Safety ........................................................................................................ 67
  2.10. Electronically Controlled Suspensions ............................................................... 70
    2.10.1. Implementations of Controlled Suspension ........................................... 74
    2.10.2. Sensors .................................................................................................... 83
    2.10.3. Control Hardware .................................................................................... 86
    2.10.4. Modelling Controlled Suspensions ......................................................... 88
  2.11. Practical Suspension Control Limitations .......................................................... 91
  2.12. Computer Optimization of Suspension Control .............................................. 93
    2.12.1. Evolutionary Algorithms ...................................................................... 93
    2.12.2. Components of Evolutionary Algorithms ............................................. 95

3. Vehicle Suspension Performance Measures .......................................................... 100
  3.1. Isolation and Comfort ....................................................................................... 101
  3.2. Tracking .......................................................................................................... 111

4. Suspension Control ................................................................................................ 123
  4.1. Introduction ...................................................................................................... 124
  4.2. Energy Dissipation in Semi-Active Suspensions ............................................. 132
  4.3. Road Discontinuity .......................................................................................... 135
  4.4. Overview of Experimental Controls ................................................................. 137
  4.5. Closed-Loop Controls ...................................................................................... 139
6.3.7. Estimate of Damper Inverse Function ........................................... 299
6.4. Overshoot Experiments .................................................................. 304
6.5. Suspension Algorithm Tests ............................................................. 314
  6.5.1. Damper Function and Inverse ...................................................... 314
  6.5.2. Experimental Algorithms ............................................................. 316
  6.5.3. Implementation & Results ............................................................ 323
7. Summary and Conclusions .................................................................. 329
  7.1. Overview ......................................................................................... 329
  7.2. Numerical Modelling Investigation ................................................ 330
  7.3. Physical Experiments ...................................................................... 334
  7.4. Original Contribution ..................................................................... 334
  7.5. Further Investigation ..................................................................... 346
8. Appendix .............................................................................................. 351
  8.1. Basic Nomenclature ......................................................................... 351
    8.1.1. Road and Chassis Height ......................................................... 352
    8.1.2. Stroke, Rattlespace ................................................................. 354
    8.1.3. “Damping Coefficient” and “Damping Constant” ................. 355
    8.1.4. Soft and Stiff Suspensions ...................................................... 356
    8.1.5. Sigmoid Function .................................................................... 357
    8.1.6. Fitness Function ........................................................................ 357
  8.2. Maple Derivation of LQR Coefficients .......................................... 358
  8.3. Maple Derivation using Euler-Lagrange ......................................... 362
  8.4. Transmissibility for Passive and Skyhook Suspensions ............... 364
  8.5. RMS Acceleration in “Elevator Example” ....................................... 367
  8.6. LQ Problem over Jerk with Finite Time ....................................... 368
  8.7. Alternative Proof for Minimum-Time Control .............................. 371
  8.9. Java Code for Time to Landing Surface ........................................ 375
  8.10. Maple Derivation of Jerk for Distance Constraint ....................... 377
  8.11. Java Code for Distance without Overshoot Method ................... 381
  8.12. Exponentially-Weighted Moving Average ................................... 384
  8.13. Independent Test Programs for Complex Algorithms ............... 385
  8.15. Numerical Methods ...................................................................... 393
  8.16. Conjectured Optimality of Skim Control .................................... 397
  8.17. Code for Two-Second Timer Counter ......................................... 401
  8.18. Overview of Test Bed Program Design ....................................... 402
  8.19. Road Surfaces .............................................................................. 410
  8.20. Fitness Measures .......................................................................... 420
  8.21. Genes and Evolutionary processes .............................................. 420
    8.21.1. Genes .................................................................................. 421
    8.21.2. Generations .......................................................................... 423
    8.21.3. Genomes and Conversions of Algorithmic Parameters ....... 424
    8.21.4. The Mechanics of Mutation and Crossover ......................... 429
    8.21.5. Selection ................................................................................ 432
  8.22. Calibration of Digital Accelerometer ............................................ 440
  8.23. Physical Experiment Components and Some Code .................... 444
  8.24. AVR C Code for Crossover Removal ........................................... 448
## Abbreviations and Symbols

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analog-to-Digital Converter</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>BNC</td>
<td>Bayonet Neill-Concelman</td>
</tr>
<tr>
<td>BOB</td>
<td>Bang-Off-Bang</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CAP</td>
<td>Cumulative Absorbed Power</td>
</tr>
<tr>
<td>DE</td>
<td>Differential Equation</td>
</tr>
<tr>
<td>DOF</td>
<td>Degrees of Freedom</td>
</tr>
<tr>
<td>EA</td>
<td>Evolutionary Algorithm</td>
</tr>
<tr>
<td>ER</td>
<td>Electro-Rheological</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HMMWV</td>
<td>High Mobility Multi-Purpose Wheeled Vehicle – also referred to as “Humvee” or “Hummer”</td>
</tr>
<tr>
<td>HRI</td>
<td>Half-Car Roughness Index</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IRI</td>
<td>International Roughness Index</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organization</td>
</tr>
<tr>
<td>ISP</td>
<td>In-System Programming</td>
</tr>
<tr>
<td>LIDAR</td>
<td>Light Detection and Ranging</td>
</tr>
<tr>
<td>LQ</td>
<td>Linear Quadratic</td>
</tr>
<tr>
<td>LQR</td>
<td>Linear Quadratic Regulator</td>
</tr>
<tr>
<td>MCU</td>
<td>Microcontroller Unit (µC is also used)</td>
</tr>
<tr>
<td>MEMS</td>
<td>Micro-Electro-Mechanical Systems</td>
</tr>
<tr>
<td>MR</td>
<td>Magneto-Rheological</td>
</tr>
<tr>
<td>MRF</td>
<td>Magneto-Rheological Fluid</td>
</tr>
<tr>
<td>NHTSA</td>
<td>National Highway Traffic Safety Administration</td>
</tr>
<tr>
<td>OEM</td>
<td>Original Equipment Manufacturers</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PSD</td>
<td>Power Spectral Density</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
</tr>
<tr>
<td>RS-232</td>
<td>Recommended Standard 232</td>
</tr>
<tr>
<td>RTOC</td>
<td>Real-Time Optimal Control</td>
</tr>
<tr>
<td>SSF</td>
<td>Static Stability Factor</td>
</tr>
<tr>
<td>sup</td>
<td>Supremum</td>
</tr>
<tr>
<td>SUV</td>
<td>Sport Utility Vehicle</td>
</tr>
<tr>
<td>TARDEC</td>
<td>Tank Automotive Research, Development Center</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>UTACV</td>
<td>Urban Tracked Air Cushion Vehicle</td>
</tr>
<tr>
<td>XLR</td>
<td>“Cannon X” series electrical connector, with a Latch, and a Rubber compound surrounding the contacts.</td>
</tr>
</tbody>
</table>
Summary

Electronic suspension controls can be optimized for a range of performance goals. There are fundamentally different performance requirements for seismic suspensions, vibrating machinery, passenger vehicles, racing cars and all-terrain vehicles. This thesis concentrates on two competing performance requirements of passenger vehicles: passenger comfort and tracking within a limited suspension stroke. Related problems are also considered. The focus is on real-time feedback controls which can be applied in microprocessors with relatively limited capacity.

Evolutionary algorithms (EAs) are used in this thesis to provide a high-level, general analysis of the problem of balancing comfort and tracking for electronically controlled suspensions with limited stroke. With numerical optimization techniques, such as EAs, the designer is not restricted to the limited range of performance measures that provide analytical solutions for idealized physical problems, such as least squares or minimum-time problems.

Electronic controls can switch force suddenly, delivering uncomfortable jolts and reducing vehicle longevity. Acceleration measures will not necessarily penalize this obvious discomfort. For this and other reasons, jerk (the rate of change of acceleration) is used here as the basis of the suspension comfort performance measure. Ironically, the almost limitless versatility of electronic controls produces a range of technical problems for suspension optimization. In particular, control artefacts can result if ill-chosen performance measures are used in the optimization process. Great care must be taken when selecting performance measures.

Two distinct performance cost functions are used in this thesis to measure relative suspension performance. Jerk is used as the basis of the smoothness/vibration measure, and a novel nonlinear cost function is applied to tracking. A suspension is penalized much more heavily when it approaches close to the edges of the working space of the suspension’s vertical travel.

It seems clear that unnecessarily and repeatedly hitting hard against the edges of the suspension working space should be penalized in a measure of suspension tracking, but
tracking measures generally fail to explicitly refer to the working space width. This matter is analysed below, showing that driver slowdown is a complicating factor. A conservative approach is adopted here using a nonlinear cost function.

Electronic suspensions can be divided into two types: active and semi-active. Active controls can apply appreciable power to produce a desired control force. Semi-active systems use only minimal power. Semi-active suspension control generally takes the form of controllable dampers combined with standard near-linear springs.

The test rig of the physical experiment is of the semi-active type. High performing semi-active controls are generally based on active controls. Thus active controls are also investigated in this thesis.

The linear control that responds directly to chassis velocity, which is arguably the prototypical form of “skyhook” suspension, cannot be implemented by passive control elements. Linear control can be generalized to a linear control over force, or equivalently acceleration (since mass is assumed constant), using evolutionary algorithms to determine linear coefficients. This can be further generalized to a control over jerk.

By “stiffening” the suspension as it moves away from equilibrium it can be made to combine softness over smooth roads with the capacity to react to large bumps when needed. The stiffening algorithms that can be used will generally be impossible to implement using just rubber or neoprene bump stops.

Electronic control opens up the possibility of using on-board, real-time control to determine a smooth chassis trajectory within the possible future limits of the suspension working space. The space within the limits of suspension travel is referred to as the “rattlespace” in this paper. This represents a novel approach to suspension control, to the best of the author’s knowledge. Two general methods are discussed: one that adjusts the suspension “stiffening” according to the current road state, and another that targets edge trajectories within the possible future movements of the rattlespace. Some of these controls performed very well. With further investigation, they may be developed into extremely high performance controls, especially because of their high adaptability to varying conditions.
The problem of avoiding collisions with rattlespace limits suggests that it may help to investigate the simpler problem of using a control to avoid overshoot of a limit distance. It becomes apparent that the residual acceleration at the point of closest approach needs to be limited, otherwise instability results.

This led to the search for controls that attain rest (including smoothly reaching zero acceleration) without overshooting the final rest position. Many initial attempts failed as feedback controls until a discovery was made using the author’s modelling software. The minimum jerk needed for a general minimum-time control that does not overshoot zero displacement is always the control with just one intermediate switch of control, instead of two switches that are generally needed (as shown below).

This was proven to be optimal, and because of its optimality it works consistently when applied as a closed-loop, real-time optimal control. This control deals with the most difficult part of the trajectory: the final, “docking” manoeuvre. It is then possible to use this in combination with other controls that can speed up the time to attain rest, or which can apply other constraints: it is possible to quickly anticipate all the extremes of the control. The control proved to be robust in physical experiments. This control may have application areas such as aerospace, door closing, and lift control.

This control has also been used to describe a general procedure for coming to rest in a stationary rattlespace. Some heuristics have been developed here to account for stochastic movement of the rattlespace edges in suspension controls, and these have proven quite successful in numerical experiments.

Semi-active suspensions have a limit on the forces they can apply (the passivity constraint). It has been known for some time that a “target” control will need to be clipped when used with a semi-active system. Clipping however produces uncomfortable jerk. Novel control methods for removing this jerk are proposed here, based on a theoretical analysis of the physics of the control. One method produces a vast improvement in semi-active controls in the numerical experiments.
1. Introduction

One of the earliest papers on the electronic control of suspensions is Crosby and Karnopp’s article, The Active Damper, dating from the mid-1970s (1973). However, even as late as 1995, Karnopp noted that,

“It is probably no surprise in retrospect that progress on practical active or semi-active vibration control systems has been relatively slow. The design of such systems requires a clear concept, related not only to the mechanics of the system but also to automatic control and system dynamics. Sensors and actuators must be available and their limitations considered and finally cost effective signal processing devices must be available. Only relatively recently has progress in all these aspects come to the point at which practical designs are possible.” (1995, p184)

Since the mid-1990s the landscape has changed dramatically and sensors and actuators have become inexpensive, fast and reliable, to the point that they are now regularly installed in production vehicles. While electronically-controlled suspensions are available mainly in more expensive vehicles, their use is steadily expanding. Indeed, during the course of this research there has been an explosion in the number of types of vehicles with semi-active suspensions using magneto-rheological dampers.

This thesis concentrates on the development of control algorithms for active and semi-active suspensions. Today’s computing power allows complex virtual prototypes to be designed, modelled and tested on a computer, replacing some of the relatively expensive stages of building and testing physical prototypes that were previously needed.

Evolutionary algorithms (EAs) are computer algorithms that can be used to evolve engineering systems, making slight changes to design parameters and selecting improvements based on performance. This is an iterative process that mirrors biological evolution,
employing computer analogues of biological reproduction, cross-breeding, and selection (refer to section 2.12.2). It requires a great deal of computer processing power, but it has proven successful over a wide range of engineering applications.

In this thesis evolutionary algorithms are applied to the development of control laws for “intelligent” vehicle suspensions (Li et al., 2004; Goncalves and Ahmadian, 2002; Ahmadian and Simon, 2002; Deb, 2005; Stembridge et al., 2006; Hyvärinen, 2004). In practice, “intelligent control” refers to electronic control, and the latter term will be used here.

The main focus of this thesis is the design and performance of suspension control algorithms. Evolutionary algorithms are being used as a tool to research the effectiveness of various control algorithms. It is important to clarify that EAs are not being used in the on-board computer control in the suspension systems. EAs are here being used “off-board” (or “off-line”) to research different kinds of electronic suspension controls that potentially can be used in “on-board” (“on-line”) electronic control algorithms. The goal is to develop relatively simple control laws that can be used in real-time microprocessor control.

Suspension control theory in the past rested on a large number of heuristic improvements on linear control: bump stops, bushing, patch dynamics (the “patch” is the section of tyre that meets the road and which supplies the driving and normal force), nonlinearity in dampers and sometimes in springs, and the geometry of the suspension framework including such factors as camber angle, caster angle, toe pattern, roll centre height, scrub radius, scuff and more. (Bastow et al., 2004; Milliken and Milliken, 1995) Such an approach has been highly successful as evidenced by the wide range of high-performance suspension systems available today, and contains essential knowledge for good suspension design.

However, theory that applies specifically to electronically-controlled suspensions is only just emerging and is still very sparse. This thesis aims at a first-order investigation of the theoretical underpinnings of suspension control and the optimization of suspension control with emphasis on electronically controlled suspension.

Many electronic controls in the literature are loose variations on the specific control referred to here as the “purely linear skyhook” (refer to section 2.4). Unfortunately, the term “skyhook” has become almost synonymous with electronic suspension control, although
there should at least be some sense in which a “skyhook” control accounts for absolute chassis height.

The purely linear skyhook control, and “skyhook” controls generally, cannot be realized with purely passive control elements (one can’t actually attach a hook to the “sky”), but they can nonetheless be emulated in an electronically-controlled system. The theoretically superior transmissibility of the purely linear skyhook control (see section 2.4) has been shown to translate into the real-world. Despite the success of the linear skyhook, it is not “optimal” in any mathematical sense and provably optimal systems, such as LQR systems (refer to section 2.3.1), have also been shown to perform well. A large number of controls have proven to be superior in the numerical experiments performed here.

Electronically-controlled suspensions are categorized as either active or semi-active. Active suspensions are generally powered by hydraulics, but other power sources are used, as discussed in section 2.10.1. Semi-active systems, on the other hand, continuously vary the parameters of a suspension element (almost invariably the damper), but they do so without the input of significant amounts of energy; they do “not require either higher-power actuators or a large power supply” (Cho et al., 1999, p667).

A number of physical mechanisms can be used to vary damper stiffness in semi-active controls, but currently the most popular method employs magneto-rheological dampening. By applying an electric current to electromagnets inside the damper, its stiffness can be readily controlled electronically (see section 2.10.1). One of the desirable features of controlled dampers is that they can easily replace a standard passive damper without upsetting the delicate balance of the suspension geometry, which has undergone a century of intensive research and development.

Analytical solutions that might be applied to suspension control are available for only a handful of highly idealized optimization problems. These problems “are typically formulated in the standard LQ manner seeking to minimize sprung mass acceleration, sprung mass jerk, rattlespace requirements, or combinations of these items” (Vaughan, 2004, p9). Amazingly, in 1988, armed with just very basic analytical tools, Redfield and Karnopp were able to perform a quite comprehensive multi-objective analysis using what would now be called Pareto optimization (1988).
Designers are familiar with analytical techniques and these are discussed at length in this thesis, but there are pitfalls and limitations to such techniques. To begin with, only a handful of highly idealized problems have analytical solutions. Of these many do not have corresponding optimal feedback controls. Modern numerical techniques can bypass these limitations, applying vast computing processing power to investigate problems beyond the reach of analysis. In this thesis numerical techniques are applied to an optimization problem which is almost certainly intractable, employing multi-objective, non-quadratic performance measures to models that include complex nonlinear suspension control algorithms.

As with all numerical optimization approaches, there is the caveat that evolutionary algorithms are limited to the specific system that is modelled. They do not produce guaranteed optima; they are said to produce “suboptimal” results. Nonetheless the method is successfully applied to a wide range of engineering problems. While EAs do not necessarily guarantee optimality, the results are found to be generally robust (in the sense explained in section 2.3.2).

Ironically, it is the very flexibility of electronic control that can exploit weaknesses in optimization criteria resulting in undesirable artefacts. One very obvious way in which the extreme flexibility of electronically-controlled suspensions can express unwanted behaviour is by supplying sudden changes in force, even over smooth ground.

This has been observed in experiment, and it can arise for a variety of reasons. It can derive from time optimality considerations, using Pontryagin’s theorem to produce time optimal, bang-bang responses (see section 2.3.2.3), or from optimal power absorption in semi-active systems (see section 4.2). It can arise simply because it is generally easier to switch power on and off than to supply a continuously variable output. It also arises in a natural way from controls that are clipped by some constraint, such as the passivity constraint of semi-active controls (explained in detail in sections 2.6 and 4.8). Sliding-mode suspension control will also exhibit uncomfortable force “chatter” if the artefact is not explicitly removed by somewhat arbitrarily applying a linear control near the sliding surface (as discussed in section 2.3.2.4).
Sudden changes in force produce clear and obvious discomfort, but when root mean square (RMS) acceleration is used as the comfort measure, such sudden changes, between moderate values of acceleration, are completely invisible. And yet least squares acceleration is the most-often used measure of comfort for suspension systems. It could be argued that it has the status of being a standard measure. And it has empirical support. An extensive study in 1978 found in fact that this was a good predictor of perceived ride comfort, “Excellent correlation was found to exist between the subjective ride ratings and simple root mean square acceleration measurements…” (Smith et al., 1978, p34).

Of course, at that time there were no electronically controlled suspensions in production. It should also be noted that in the 1978 paper by Smith, McGehee et al. the rate-of-change of acceleration (jerk) was not investigated as a comfort measure. Certainly, when jerk is used as a comfort measure, sudden changes in force are heavily penalised. Slowly at first, but increasingly in recent decades, jerk (the rate-of-change of acceleration) has been used as a basis for measures of comfort and vibration. Reasons why this might be so are analysed in some depth in section 3.1.

There are a number of extant terms for the rate-of-change of acceleration (refer to section 3.1). The word “jerk”, admittedly not the prettiest term, has broad acceptance and will be used throughout in this thesis.

There are a number of criteria for suspension systems (refer to section 2.9) but the main two objectives are comfort and tracking. These goals compete with each other. If the suspension is too soft it will be comfortable but will not track well. If it is too hard it will be uncomfortable. The goal is to find a suspension control that reaches a compromise between the two and which optimizes both within the constraints of that compromise.

Thus the tracking performance measure is also critical to the optimization process. Again, care is needed to avoid unnecessary artefacts allowed by the performance measure. While the term “tracking” is widely used its precise meaning is complex. This matter is fully analysed in a long discussion on the meaning of tracking in section 3.2. At this point the best, most widely accepted definition of tracking is the ability of a suspension to staying within an acceptable distance of the target, for instance staying within road-height limits that case a suspension to hit hard against the top or bottom of its travel limits. There is the problem that
many tracking performance measures allow travel outside the physical travel limits of the suspension. The range of the suspension’s vertical travel is known by a number of terms, but the expression used throughout this thesis is “rattlespace”. The term “stroke” refers to the suspension displacement, and in this text refers to displacement around the zero equilibrium position. Refer to appendix 8.1 and appendix 8.1.2 for further clarification of these terms.

Free travel within the rattlespace does not of itself adversely affect a suspension. When the wheel is moving over rough, small corrugations, the suspension comfort as well as road normal force are improved if the chassis remains relatively flat. With the chassis remaining flat the wheel must move up and down to match the road corrugations, with corresponding stroke movement. But this suspension movement does not of itself adversely affect comfort, and may well improve road holding.

A suspension should be able to move freely within the rattlespace to optimize comfort, but at the same time minimize the chance of hitting unnecessarily against the rattlespace limits. In the extreme case, one could visualize a suspension travelling over a bumpy road with the chassis remaining completely flat, with considerable wheel movement inside the rattlespace, exactly matching the road height. This will be referred to here as the “flat” control: keep the chassis height perfectly flat. Such a control in fact would be perfect for small road fluctuations. At the time of writing, the Bose website (Bose, 2007) contained some examples of a controlled suspension that could keep the chassis almost perfectly flat over relatively small, but quite rough corrugations. A perfectly flat ride achieves a perfect comfort performance (not to mention the fact that road normal force is kept steady), even though the suspension may be moving heavily in order to exactly match road fluctuations.

However, when the bumps are larger than half the height of the rattlespace, a perfectly flat suspension will show its limitations by unnecessarily and repeatedly hitting up hard against the limits of vertical suspension travel: either “bottoming” or “topping”. Some degree of “bottoming” or “topping” will occur with even the best suspension, and there is a compromise between a too soft suspension that is smooth but which will collide too often with rattlespace limits, and a too rough suspension which tracks well but is uncomfortable. Both “topping” and “bottoming” have an extremely negative impact on suspension performance.
Some practical compromise is sought between maintaining the wheel near equilibrium and allowing free travel to provide comfort and consistent tracking force. How should excessive travel be included in a tracking performance index? It is argued below that large bumps cause drivers to slow down and so perhaps tracking measures need to take account of the speed at which the vehicle can traverse a bump (all else being equal). It should also be born in mind that while slowdown is caused by threatened collisions with rattlespace limits, it is also the result of drivers’ seeking just to avoid the very uncomfortable response of passive suspensions to large bumps, not to mention the instability that might result from oscillations.

Forward vehicle velocity was not used here for a number of reasons, explained in section 3.2. Instead, a novel nonlinear performance indicator was developed, which only penalizes travel near or beyond the rattlespace limits. This then is a compromise, being only a moderate departure from traditional least squares measures, at the same time allowing relatively free travel within the rattlespace constraints.

Controlled dampers in a semi-active suspension can supply force only in one direction at any one time, depending on whether the damper is extending or compressing. When the damper is extending it can only supply a downward force on the chassis; when compressing it can only push upwards. This is known formally as the “passivity constraint” (see section 2.6). When a semi-active control follows a general control law such as, say, an LQR control, the control force will need to be “clipped”. At the point of clipping there is generally a sudden change in force (as explained in section 2.6). This problem is analysed in more depth in section 4.8, and methods for diminishing or removing this effect are discussed in section 4.8.3.

Many control algorithms have been advanced for the numerical experiments performed here; some of these are derived from theoretical considerations, such as linear theory or minimum-time optimization, while others are simply heuristics, such as the “virtual bump stops” which stiffen on approach to a rattlespace limit. These algorithms are developed and tested using evolutionary algorithms, as described in chapter 5. The computer program running the EAs, written in Java, was entirely the author’s work.

A test rig has been constructed and a physical test of a small number of candidate algorithms has been performed (see chapter 6). The physical rig was built to test the feasibility of the suspension controls for real-time feedback control. The rig however demonstrates with very
modest equipment that the controls adapted for the rig are more than capable of running in real time. The rig was also applied to the “landing surface” control showing that this control is a practical real-time feedback control for semi-active systems (refer to section 6.4).

1.1. Problem Definition

In concise terms, the problem that is being analysed here is to provide a practical suspension control algorithm that produces a smooth trajectory for the sprung end of the suspension (the chassis in a vehicle) while maintain an acceptable range of distance from a target (staying within the suspension vertical travel limits for a vehicle). The main target for this research is passenger vehicle suspensions which have a limited travel and in which one of the main suspension performance goals is ride comfort.

The automobile has been subject to an extraordinarily large amount of research, and a great deal of practical knowledge has been amassed (Milliken and Milliken, 1995; Bastow et al., 2004; Barak, 1991; Dixon, 2008). Currently, most vehicles use springs that are generally almost perfectly linear. Originally, most vehicles used nonlinear springs, mainly leaf springs, but these are now virtually non-existent on passenger vehicles. Today, independent, passive hydraulic dampers are still mainly used, although there have been some minor but successful variations such as the Moulton Hydrolastic system (Wang, 2001). It should also be noted that standard passive suspensions are deliberately constructed with nonlinearities in the damper (Milliken and Milliken, 1995; Bastow et al., 2004).

The major focus of this thesis is to investigate the design of real-time controls for electronically-controlled vehicle suspension systems. The goal is to find control algorithms that provide a comfortable ride but which also track the road surface well. These goals conflict: generally a very soft suspension will be smooth and comfortable, but will track badly, while a very hard suspension tracks well, but is less comfortable. So the goal is to find algorithms that provide the best compromise, simultaneously improving both comfort and tracking.
Jerk is the basis for a measure of suspension comfort, and a nonlinear function of stroke displacement (wheel displacement relative to the chassis) is employed for a measure of tracking. This nonlinear measure is entirely new, to the best of the author’s knowledge. The theoretical and practical issues associated with these measures are explored in this thesis.

The control algorithms being sought must be simple enough to be capable of being applied in real time. On modern microprocessors, simple polynomial time computations such as trigonometric calculations, square roots, or the use of the Newton-Raphson method (Kreyszig, 1993, p929) and other simple iterative methods are considered undemanding enough for real-time application. Of course, microprocessor processing speed is continually improving. Nonetheless, on-board electronic suspension controls have processing time limits that cannot be exceeded, and complex numerical algorithms, such as on-board evolutionary computations or numerical solutions to complex differential equations are not here considered appropriate for on-board application.

In this thesis relatively fast and simple control algorithms are sought. Such methods come under the rubric of “real-time computation”. According to Ross,

“In the early years, it was necessary to solve problems analytically (i.e. in terms of elementary functions) so that function evaluations could be done easily. Today, this ease of analytical solutions is defined more fundamentally as real-time computation. Real-time computations are necessary for feedback implementations while analytical solutions are sufficient.” (2009, p26)

Note again that evolutionary optimization is too complex and too slow to be applied in the final on-board system; they are being used off board to evolve the parameters of other, simpler on-board controls, which process real-time computations.

It is important to stress that evolutionary algorithms are not themselves the major focus of the thesis. This thesis addresses the theory of suspension control algorithms. Evolutionary algorithms are being used here as a tool to optimize control performance for particular controls, and also to compare performance between control types, for optimization problems that are intractable analytically.
Each of the controls run in the evolutionary algorithms has a number of real-valued parameters. For example, the passive system, one of the simplest algorithms used, has just two parameters, one representing spring rate and the other representing the damping rate. Other systems have of the order of thirty such parameters. These parameters are adjusted in an evolutionary process. (The final run of EAs, covering some 123 algorithms, took 9 and one half days to complete.)

1.2. Theory Overview

Control over acceleration is sometimes referred to as the “double integrator” (Ross, 2009, p42). By extension, control over jerk can be referred to as the “triple integrator”. Some of the analytically derived double-integrator controls have been extended by the author into triple-integrator controls, such as the minimum-time and LQR controls.

Small changes in an analytical control problem, even just changes in performance measures, can produce dramatically divergent results. For instance, discontinuous forces result from minimum-time problems with constrained acceleration, but smooth controls result from LQR control over acceleration. Sometimes analytical techniques generate obviously absurd and wildly impractical controls. For instance, at the limit, the unconstrained minimum-time problem requires an infinite force over an infinitely small period of time (MacCluer, 2005, p110; Ross, 2009, p61). As MacCluer asserts,

“the application of optimal control to practical problems is an art, requiring the practitioner to perform many analytic and numerical iterations to reach an acceptable (but often not optimal) solution to the original problem” (2005, p113).

An essential part of this process is getting a realistic and workable problem formulation. According to Ross,

“a critical part of designing a practical control system is not only in using a proper computational technique for implementing feedback controls but also getting the problem formation right in the first place!” (2009, p51)

In the case of linear controls, stability, optimal conditions, the most likely state estimates and possible failure conditions (through resonance) are all extremely well understood. There are a
few analytical problems which, like linear controls, have well-defined solutions under general conditions. But the number of such problems is extremely small and any given physical problem is liable to only approximate such idealized problems at best. Analytical techniques are limited to only a handful of idealised problems.

In contrast, numerical methods today can be applied to virtually any physical problem. Due to increased computational capacity, problems can now be approached that would have been impossible a few decades ago. Nonetheless, numerical methods have the drawback that they cannot guarantee generality. For example, the aberrant behaviour of linear systems at resonant frequencies is well understood, but a slight change in a nonlinear system may produce aberrant behaviour. Numerical models may provide theoretical support, but they do not prove generality.

Nonetheless, there is little to lose and potentially much to gain by using numerical methods to optimize and compare various types of control which are intractable analytically. Furthermore, there is a sense in which some evolutionary algorithms can claim a degree of “robustness”, in terms of sensitivity to slight variations in conditions and control parameters (see section 2.3.2).

With numerical techniques the optimization problem can be modified in almost any way that seems desirable to precisely suit the needs of any application. Both the model and the optimization criteria are virtually infinitely flexible. Designing the performance goals then can be an important part of the optimization problem. This is a point which might be overlooked in the highly confined space of analytical techniques, where “least squares” (see section 2.3.1) performance measures are often assumed without discussion.

There is a wide range of possible performance indexes for suspension comfort: least squares chassis vertical acceleration, least squares jerk, maximum jerk, or even higher time derivatives than jerk. Performance measures could be based on frequency, or even subtle combinations of any of the above. Comfort measures could even be based on modes of vibration of rough models of human frames. The relative value of measures using acceleration compared to measures based on higher derivative of motion is a particular focus of this thesis.
There are also a large number of methods for measuring suspension tracking performance. A nonlinear penalty for travel near and outside the rattlespace limits has been used as the performance measure for suspension tracking in the numerical experiments performed for the thesis. This measure has not been used elsewhere to the best of the author’s knowledge. The matter of tracking performance is analysed in depth in section 3.2.

Chapter 4 investigates suspension control theory that is relevant to the controls used in the experiments, including new theoretical elements developed by the author. The chapter begins with two relatively simple, independent theoretical matters that are nonetheless important enough to warrant inclusion in two very short subsections: energy dissipation (section 4.2) and road discontinuity (section 4.3).

It is sometimes claimed that semi-active systems are inherently safe and stable because they dissipate energy. Section 4.2 demonstrates how a semi-active control could be deliberately engineered to increase kinetic energy where possible, even though the control element is technically “dissipative”. A well designed semi-active control will tend to be safer than the passive, and it is relatively easy to characterise the conditions under which such a control will be truly dissipative. The purely linear skyhook for example absorbs energy better than the passive (with the same “spring” and “damper” rates). The “on-off skyhook” absorbs energy at the highest possible rate. Nonetheless, section 4.2 demonstrates that it is not a general physical fact that all semi-active controls are inherently stable.

Section 4.3 proposes a simple first-order categorisation of suspension controls in terms of the capacity of suspensions to travel over road discontinuities. The response of the passive suspension to small sudden changes in road slope is readily observed in the average passenger vehicle. A sharp jolt can be felt by the passenger even over very small discontinuities as the passive damper suddenly changes force. The skyhook suspension, in contrast, has a continuous force over a sudden change in road slope. This characterization in terms of response to road discontinuity is very simple, but it can aid the designer’s intuition.

The remainder of section 4 investigates and develops a number of different suspension control algorithms. These form the basis for the algorithms used in the numerical and physical experiments. Some simple controls have been developed around simple modifications of linear controls. These derive from a number of sources: previous work in the
field, extensions of previous work, heuristic devices that were thought worthy of at least numerical experimentation, and various analytical optimization problems. Some are based on theoretical investigations, such as the time-optimal controls over jerk.

Suspension control is traditionally based on linear theory. There are different types of linear controls described in the literature: ranging from the standard linear suspension, the purely linear skyhook, controls that address the LQ problem (linear control with quadratic performance measures), and controls based on the Kalman filter. Linear control over acceleration is here also extended to linear control over jerk.

In this thesis, control modifications are developed that increase suspension stiffness as the chassis moves further and further away from equilibrium. For the sake of discussion, these have been termed “virtual bump stop” controls.

Evolutionary algorithms are used to “optimize” control parameters, including linear coefficients. The evolutionary algorithms then do all the work of determining the parameters and provide a method for comparing results.

Bang-bang controls and bang-off-bang controls arise naturally from the minimum-time problem (as explained in section 2.3.2.3). Such controls with constrained force (equivalently acceleration) are impractical because of the discomfort caused by sudden changes in force. The minimum-time control over constrained jerk was derived here using Pontryagin’s theorem to be a bang-bang control generally requiring two intermediate switches of control, compared to just one in the case of control over acceleration. Minimum-time control over jerk does not suffer from sudden changes in force.

An RTOC (real-time optimal control) was found that could implement the basic minimum-time control over jerk in closed-loop form, and the control has interesting similarities to sliding-mode control (refer to section 4.6). After a search of the literature, it seems that this feedback method has been applied by Koh et al. in the field of mechatronics (1999). As shown below, the feedback algorithm can be extended to even higher order derivatives of motion than jerk, although it becomes much more complex. Minimum-time controls over jerk and a number of variants developed here have been subject to numerical optimization.
A new theoretical category of controls has been introduced that deals directly with the fact that suspension travel is constrained within the rattlespace limits. The defining trait of such controls is that they in some way affect the physical trajectory of the chassis to explicitly avoid intersecting with the trajectories formed by the edges of the rattlespace (refer to section 4.7.5). These have been termed “rattlespace constraint” controls. These controls need in some way to “anticipate” collisions with rattlespace limits.

After some initial experimentation, it was evident that these kinds of controls need to limit the acceleration that remains once the chassis has made its closest approach to a rattlespace limit. For the sake of discussion this problem is here referred to by the term “rebound” (refer to section 4.7.3). In broad terms, controls need to avoid smashing unnecessarily hard against the rattlespace edge but at the same time not aggravate the situation by increasing energy back in the opposite direction.

For example, if constant force is used to absorb energy then there is a large residual force remaining which needs to be removed slowly. If the forces are too large then either the suspension will have to sacrifice smoothness or potentially become unstable. However, by using controls that smoothly reduce acceleration at the same time as they reach their closest approach to rattlespace limits, the problem can be overcome.

This suggests a related problem: one of bringing a system to rest (including zero acceleration) at a given distance using constrained jerk control without overshooting. Such a problem helps shed light on maintaining a suspension within the rattlespace, but it may also attract independent interest (see section 4.7.3). An initially surprising result was found serendipitously which led to what is here called the “landing-surface” control for reaching a given distance without overshoot.

Semi-active suspensions suffer from a singular control constraint: the damper can only supply force in one direction. As expressed by Ahmadian et al., “at zero crossings of the velocity, conventional skyhook introduces a sharp increase (jump) in damping force, which, in turn, causes a jump in sprung-mass acceleration” (2004, p580). When any general control is clipped to remain with the limits of the controlled damper, an uncomfortable spike in jerk can be produced, and this has been found in experiments in the literature (discussed in section
4.8.1). Controls were developed here that could anticipate an imminent jerk spike and take steps to diminish it.

1.3. The Contribution of this Thesis

A theoretical goal of this thesis is to provide a first-order investigation of the performance measures that are appropriate for highly-flexible electronic suspension controls. Most importantly, reasons are advanced as to why acceleration alone is inadequate as a measure of comfort.

As outlined above and as explained in detail below, there are a number of factors that may make electronic controls prone to the generation of sudden force changes. Nonetheless, the least squares acceleration performance measure fails entirely to penalise sudden changes in acceleration (between moderate values) as being uncomfortable. Using jerk rather than acceleration overcomes this problem.

All suspensions have limits on suspension travel. Vehicles that are intended for rougher terrain tend to have long suspension strokes. Nonetheless all vehicles must find a compromise between comfort and tracking which allows the suspension to remain within its working space. Tracking then becomes, at least in part, a matter of keeping the suspension within the rattlespace without unnecessarily hitting violently against the rattlespace limits. This component of the suspension problem is analysed in depth in section 3.2.

A number of suspension controls have been tested with evolutionary algorithms. Some controls are analytically derived while others are purely heuristic in nature. Evolutionary algorithms have been run for each control, with the same evolutionary schedule and with road data chosen in exactly the same way in each case. The road data is randomly chosen, from one generation to the next.

Evolutionary algorithms are used to determine suboptimal control parameters. By comparing the performance results from different control types, evolutionary algorithms become a tool
for investigating the relative performance of suspension controls. In a sense, the controls “compete” against each other.

Linear controls have been investigated in the numerical experiments, including both the linear passive and the linear skyhook, as well as LQR and general linear controls over both acceleration and jerk, and finally general linear controls over acceleration and jerk that also respond to chassis absolute height, and higher derivatives of chassis height. The linear coefficients in each case are determined by evolutionary algorithms (refer to section 4.5.1).

Furthermore, what are here termed “virtual bump stops” can be applied to modify linear controls in much the same way that actual bump stops are added to passive suspensions. These can be much more flexible however than what can be achieved with rubber or polyurethane bump stops.

Just as LQR control over acceleration can be extended to LQR control over jerk, so minimum-time controls over acceleration can be extended to minimum-time controls over jerk. A practical RTOC for implementing minimum-time control over jerk has been found. (This control can be extended to higher-order derivatives of motion.) This control does not suffer from acceleration chatter.

A general novel category of controls has been developed which targets the chassis trajectory relative to the constraints of the projected rattlespace trajectory. This covers a wide range of possible controls. To provide first-order analysis of this kind of control, simplifying heuristic methods were used to anticipate rattlespace movement, and evolutionary algorithms are used for adjustment of suspension parameters. A number of highly performing controls were found.

The most challenging aspect of this control involves the residual acceleration at the point of closest approach to the rattlespace limit, here termed “rebound”. If this is very large then there is a need to smoothly reduce this acceleration. If this “rebound” is too large it can lead to instability.

This prompted an independent investigation into controls that reach a limit distance without overshoot, with constrained or zero residual acceleration. A number of new methods have
been developed here that have been applied to suspension control, but which may also have other applications. These methods have also been proven theoretically, as well as being modelled numerically. The test rig was used to provide empirical evidence of their effectiveness (refer to section 6.4).

As discussed above, an inherent problem for semi-active suspensions is that they can suffer from sudden changes in force when the damper stroke velocity changes from extension to compression or vice versa. One of the controls in the literature, the no-jerk skyhook (see section 2.6) uses a global control that does not suffer from crossover jerk. During the development of this thesis a number of controls have been developed that remove this jerk.

The contributions of this thesis on the claimed stability and energy-absorbing qualities of semi-active suspensions (see section 4.2) and responses to road discontinuities (discussed in section 4.3) have been outlined above.

The work of this thesis incorporates a number of computer programs written entirely by the author. The main program for processing evolutionary algorithms, SuspensionTest, was developed for the purpose of optimizing and comparing various control algorithms under the same conditions. The resultant optimal algorithms (strictly speaking, suboptimal) were then compared against other control algorithms. The passive and skyhook algorithms are natural benchmarks in these experiments.

The physical test rig was designed in consultation with Peter Tkatchyk, who built the main test rig frame. The electronics for the control circuitry uses a component from Lord Corporation to convert a signal-level input to a current supply for the controllable damper, also supplied by Lord Corporation. The choice of a microcontroller was discussed with Sebastian Naselli. Details can be found below, in section 6.

Throughout this work I have been very ably led and assisted by my supervisors, Anna Bourmistrova and Aleksandar Subic, who helped enormously with suggestions.

Some of the work for this thesis was published during the research for the PhD, and references can be found in the reference section. (These papers are available to examiners.)
1.4. Thesis Chapter Outline

This introduction concludes with a brief outline of the thesis chapters.

Chapter 1) Introduction
This chapter provides an introduction to the suspension problem definition, the theoretical content of the thesis, as well as the contribution made by the thesis.

Chapter 2) Literature Review
Chapter 2 provides a review of the background literature on controlled suspensions. There is also some material on the properties of passive suspensions.

Chapter 3) Vehicle Suspension Performance Measures
This chapter builds on the background literature focusing in particular on the performance goals of comfort and tracking. The rate of change of acceleration, jerk, is compared with acceleration as a comfort measure. Performance measures of tracking have not previously penalized travel near the rattlespace limit. This issue and its relation to driver slowdown is analysed, and is compared theoretically with other tracking measures.

Chapter 4) Suspension Control
Chapter 4 presents the theory used in the algorithms investigated in this thesis. This describes linear controls as well as a number of novel controls and control elements created by the author: linear control over jerk, “virtual bump stops”, and methods for removing jerk created when stroke rate changes vertical direction. It also investigates methods produced by the author for using constrained jerk control (with continuous acceleration) to perform various tasks required by suspensions as well as other applications. A novel theory for suspension control, “rattlespace constraint” is described and two basic approaches are described, as well as specific controls.

Chapter 5) Computer Simulation Environment and Evolutionary Algorithm Coding
Chapter 5 describes the numerical evolutionary experiments performed for this thesis, and the test bed software developed entirely by the author. This chapter focuses on the technical matters of the implementation of the evolutionary algorithms and is somewhat independent of
the rest of the thesis except crucially for the results of the numerical experiment, presented in section 5.7.

Chapter 6) Physical Experiment
This section describes the physical experiment, from design through calibration, including the programming of the microcontroller, to the various experiments performed using the physical test rig. “Overshoot” and suspension control experiments are described and results given.

Chapter 7) Summary and Conclusions
This chapter summarises the thesis in the light of its theoretical developments and experimental results. Conclusions are drawn, and possibilities for further research are suggested.
2. Literature Review

This chapter examines some of the background literature relevant to the thesis topic. It covers the fundamentals of evolutionary algorithms, but the major part is given over to suspension design and suspension control techniques.

2.1. Design Approaches

Barak proposes a general approach to suspension design. He recommends a set of ten “magic numbers”, covering factors such as “bounce resonant frequency of the sprung mass” and “wheel hop resonant frequency and the desired total mass/unsprung masses ratio”.

“Moreover, these numbers will continue to control our design philosophy in the next decade regardless of high technological concepts such as active suspension and four-wheel steer. These magic numbers are timeless. They are the product of the so-called ‘Past Experience’ and/or ‘The Slide-Rule Generation’ in the automobile industry. In this respect, despite innovation in suspension design the reality is that there is no change in these design specifications.” (1991, p1698)

Computer “virtual prototyping” (Hyvärinen, 2004, p45) is the detailed modelling of a physical system for the purpose of analysis and testing. This can greatly assist in the design process, resulting in more sophisticated design with less need for expensive physical prototypes. A virtual prototype can be developed and tested on a computer as a part of the verification and validation process.

This thesis aims at an investigation of the appropriateness of various performance measures for suspension systems and their effects on optimality, especially for vehicles with limited or small strokes. It lightly examines suspension system fundamentals that are needed for suspension control design, but the main focus is the literature on suspension control laws.
2.2. Dynamics

The two models shown in figure 2.1 are the typically used models for the quarter-car passive suspension (Milliken and Milliken, 1995, pp235-9). In figure (b) the tyre is represented by a spring. The smaller mass, representing the wheel and the axle components that move with the wheel, is the unsprung mass, denoted here as $M$. The sprung mass, $m$, is the mass of the portion of the vehicle that does not move with the wheel: the chassis, luggage, and passengers. In most models, a constant forward velocity is assumed and it is mathematically convenient to represent road height simply as a function of time, $r(t)$. Half-car and full-car models are discussed below (in section 2.2.2).

Figure 2.1 (a) Single DOF quarter-car suspension model (b) 2-DOF model

2.2.1. Quarter-Car Dynamics
The forces acting on the chassis of the quarter car are the gravitational force, the force from the spring, and the force from the damper. If the equilibrium rest position is taken as zero the force of gravity can be ignored giving the following equations of motion, assuming that the spring and damper are linear,

\[ m\ddot{y} = -c(\dot{y} - \dot{r}) - k(y - r). \]

**Equation 2.1**

Coil springs, as used in most modern suspensions, are close to ideal in practice, except where they are deliberately designed to be nonlinear. The above equation assumes a linear damper, but car dampers in practice are only approximately linear. The nonlinearity of dampers is discussed in section 2.2.2. Even so, linear models are often used in the literature to provide a first-order approximation to real suspension behaviour.

The equations of motion for the 2 DOF linear system are,

\[ m\ddot{y} = -c \frac{d}{dt}(y - y_u) - k(y - y_u), \]
\[ M\ddot{y}_u = c \frac{d}{dt}(y - y_u) + k(y - y_u) - k_t(y_u - r). \]

**Equation 2.2**

The equivalent spring rate for the suspension spring in series with the tyre is found using the formula,

\[ k_e = \frac{k_t k}{k_t + k}. \]

This can be used to calculate the main spring rate when the combined spring rate and the tyre spring rate are known (Milliken and Milliken, 1995, pp239-40).

The tyre is often modelled simply as a spring, as in figure 2.1(b), and it is claimed that “the tire possesses negligible damping” (Miller, 1998, p2048). For more extensive modelling used for late-stage design more extensive tyre models are required. The term “magic formula” has “broad applicability to a wide range of tire curves” (Kasprzak et al., 2006). These models are used for more extensive modelling of tyre dynamics. They often use curves of best fit to empirical data, or even models developed by neural networks (Bastow et al., 2004, p289).
Tyre dynamics are extremely important to a vehicle’s stability and many of the great improvements made in car ride quality in the mid-to-late twentieth century can be attributed to the study of tyre dynamics (Segel, 1993, p7). The force supplied by the tyre lags behind the point where the vertical from the centre of the wheel touches the ground. The tyre meets the ground in a “patch” and this patch supplies the driving force, generally at a small angle away from the direct line of the hub. This results in the effect known as “slip” (Bastow et al., 2004).

A very delicate geometrical placement of the wheel and suspension housing are required to counteract unwanted instabilities. One of the great advantages of the semi-active suspension is that the delicate geometrical balance of the suspension – developed over almost a century of intensive engineering – need not be disturbed, since the only change necessary is the replacement of the passive telescoping damper with the controlled damper.

### 2.2.2. Passive Suspension Design

Parameter values used in various car models found in the literature, whether quarter-car, half-car or full-car are given in table 1 and table 2 below.

The design of suspension parameters is an art as much as a science, but the design process often begins with the targeting of fundamental frequencies (Kim et al., 2001). “For most automobiles, the heave natural frequency of the sprung mass is usually 1.0hz to 2.0hz and the unsprung mass natural frequency is usually 8.0 to 12.0hz.” (Miller, 1998, p2047) Giorgetti, Bemporad et al. targeted 1.5 Hz for the sprung mass natural frequency and 10 Hz for the wheel-hop frequency (2006). Racing cars, on the other hand, with a greater emphasis on road holding, have stiffer suspensions and a higher natural frequency of between 2 and 7 Hz (Woods and Jawad, 1991). The natural frequency, in hertz, is calculated using the formula,

\[ \omega = \frac{1}{2\pi} \sqrt{\frac{k}{m}}. \]

The natural wheel frequency, also known as the “wheel hop frequency”, is affected by the suspension spring and the tyre as if they act in parallel, giving the formula,

\[ \omega_w = \frac{1}{2\pi} \sqrt{\frac{k + k_t}{M}}. \]
The damping ratio, $\zeta$, is the ratio of the actual damping rate divided by the critical damping rate (Milliken and Milliken, 1995, p788). The critical damping rate occurs when the damping rate is,

$$c_{\text{crit}} = 2\sqrt{mk}.$$ 

Therefore,

$$\zeta = \frac{c}{c_{\text{crit}}} = \frac{c}{2\sqrt{mk}}.$$ 

At a damping ratio of $\zeta = 0.25$ the transmissibility at the fundamental frequency is typically about 2.5 and this is regarded as an acceptable compromise for car suspensions (Milliken and Milliken, 1995). Australian Government certification of “road-friendly suspension systems” stipulate that “the mean damping ratio DM must be more than 20% of critical damping,” that is $\zeta = 0.2$ (DOTARS, 2004).

Most modern dampers are designed to have a different damping rate under extension, “rebound”, than under compression, “bounce” or “jounce” (Woods and Jawad, 1991; Rideout, 1998, p8; Isermann, 2001, p98). “The rate in rebound [can be] between two and three times the rate in bump.” (Williams et al., 1996, p45) Various reasons are given for this asymmetry. According to Williams et al., “the asymmetry brings benefits during rapid wheel excursions encountered at pot holes and similar inputs” (1996, p45). According to Milliken and Milliken, “measurements on vehicles have shown that wheel velocities in the upward (bump) direction are generally considerably higher than in the downward (rebound) direction by a factor of about two. The damper is manufactured to have a corresponding asymmetry … thereby keeping forces on the vehicle symmetric” (1995, p800). (Note that upward wheel movement corresponds with suspension compression in this statement.) Bastow et al. see this as a “compromise” between soft ride and “controlling movements of sprung and unsprung masses subjected to periodic disturbances” (2004, p247).

Perhaps the clearest explanation is given by Guglielmino, E., T. Sireteanu, et al. (2008, p7):

“In the occurrence of a bump, vertical upward acceleration can reach several g while if a pothole is encountered, the vertical downward acceleration cannot be larger than 1
g. This is also the reason why hydraulic dampers are designed with non-symmetrical characteristics for bound and rebound strokes.”

It is also suggested, however, that road-holding is improved if the rates are closer together (Milliken and Milliken, 1995, p785).

Other nonlinearities, both wanted and unwanted, exist in damper characteristics: roll-off at high velocities (Milliken and Milliken, 1995, pp801-2), a “small ‘deadband’ … due to the unavoidable presence of some ‘dry’ friction (stiction) in the suspension linkage” (Milliken and Milliken, 1995, p801), and nonlinearities caused by blow-off valves designed to release pressure at high relative velocities (Milliken and Milliken, 1995, p801).

Dry friction is more formally known as Coulomb friction. “Coulomb friction is undesirable ... because it locks the suspension at small forces, and gives a poor ride on smooth surfaces, once known in the USA by the colourful term ‘Boulevard Jerk’”. (Dixon, 2008, p16)

Designers will typically spend a designated period of time, normally five to ten days, tuning dampers. Take-apart dampers are modified in an “iterative process of subjective assessment and valve changing, which continues until the desired characteristics have been approached” (Bastow et al., 2004, p169).

<table>
<thead>
<tr>
<th>Source</th>
<th>Sprung Mass kg</th>
<th>Unsprung Mass kg</th>
<th>Spring Stiffness N/m</th>
<th>Tyre Stiffness N/m</th>
<th>Fund. Freq Hz</th>
<th>Damping Coeff Ns/m</th>
<th>Damping Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tseng and Hendrick (1994, p558)</td>
<td>240</td>
<td>36</td>
<td>16,000</td>
<td>160,000</td>
<td>1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Giua, Seatzu et al. (1999)</td>
<td>288.9</td>
<td>28.58</td>
<td>14,345</td>
<td>155,900</td>
<td>1.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Savaresi, Silani et al. (2003)</td>
<td>400</td>
<td>50</td>
<td>20,000</td>
<td>250,000</td>
<td>1.01</td>
<td>1,300</td>
<td>0.23</td>
</tr>
<tr>
<td>Wang (2001)</td>
<td>250</td>
<td>35</td>
<td></td>
<td>150,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sam (2006)</td>
<td>282</td>
<td>45</td>
<td>17,900</td>
<td>165,790</td>
<td>1.27</td>
<td>1,500</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Table 1 Parameter values for some quarter-car models.
<table>
<thead>
<tr>
<th>Source</th>
<th>Sprung Mass kg</th>
<th>Unsprung Mass kg</th>
<th>Spring Stiffness N/m</th>
<th>Tyre Stiffness N/m</th>
<th>Damper Coeff Ns/m</th>
<th>Radius of Gyration m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Half Car</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ashari (2004, p373)</td>
<td>1500</td>
<td>800</td>
<td>60</td>
<td>38,000</td>
<td>190,000</td>
<td>1.5</td>
</tr>
<tr>
<td>Vaughan (2004) (Heavy Vehicle)</td>
<td>4500</td>
<td>9000</td>
<td>450</td>
<td>250,000</td>
<td>1,500,000</td>
<td>10,000</td>
</tr>
<tr>
<td>Zribi and Karkoub (2004, p510)</td>
<td>1794</td>
<td>150</td>
<td>87.15</td>
<td>16,824.2</td>
<td>1,190</td>
<td></td>
</tr>
<tr>
<td></td>
<td>150 kg load</td>
<td></td>
<td>(front)</td>
<td>(front)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(rear)</td>
<td></td>
<td>140.04 (rear)</td>
<td>18,615 (rear)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full Car</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Youn et al. (2006, p661)</td>
<td>1120</td>
<td>45 (front)</td>
<td>20,310 (front)</td>
<td>157,600</td>
<td>1,050 (front)</td>
<td>875 (rear)</td>
</tr>
<tr>
<td></td>
<td>70 (rear combined)</td>
<td></td>
<td>15,230 (rear)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Caponetto et al. (2003, p789)</td>
<td>1257</td>
<td>49 (front)</td>
<td>29,600 (front)</td>
<td>190,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model of Alfa Romeo 156</td>
<td>39 (rear)</td>
<td></td>
<td>16,800 (rear)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Some parameter values for half- and full-car models.

Half-car models of a vehicle moving over certain bumps at a particular driving speed readily demonstrate that a suspension with harder springs at the back than the front will suffer dramatically less pitch (Milliken and Milliken, 1995, pp795-6; Bastow et al., 2004, p151). This effect is most pronounced at a particular forward velocity and is affected by continuous sinusoidal disturbances, so there is a certain amount of tuning required of the difference between the stiffness of front and back springs.

### 2.3. Control Techniques

This section gives a broad overview of control techniques found in the literature that apply to suspension systems. There are many techniques that derive, in one way or another, from the theory of linear systems. “Early research focused primarily on linear techniques, such as optimal control … and skyhook control” (Dixit and Buckner, 2005, p84). Frequency response is also often a first step in the characterization of nonlinear systems and the natural frequencies of the system ignoring dampers is also useful (Goncalves and Ahmadian, 2002, p3). A step function too has been used as an input road excitation to provide a rough first impression of the transient response of a nonlinear system (Goncalves and Ahmadian, 2002,
Goncalves and Ahmadian settled on peak-to-peak excitations of 0.9 inches (23 mm) for the transient test, which seems appropriate if comfort is being tested, but insufficient if the capacity to stay within the rattlespace is a concern.

Many researchers have used frequency plots to explain the performance of semi-active or active suspensions (Cole, 2001, p230; Lauwerys et al., 2004; Stembridge et al., 2006; Koo et al., 2003, p5; Lu and DePoyster, 2002, p814; McLellan, 1998; Song et al., 2003; Vaughan, 2004, p31; Yi and Song, 1999; Sims and Stanway, 2003, p96; Wang, 2001; Williams and Best, 1994, p341; Yagiz et al., 2000; Yu et al., 2006). It is common that such plots will be compared with passive systems and that the controlled system will show a dampening of the resonance peaks of the passive system (Elbeheiry and Karnopp, 1996, p559; Burton, 1993, p230; Hyvärinen, 2004, p28; Jalili, 2002, p602; Krüger, 2002, p518; Lin and Kanellakopoulos, 1997, p48; Majjad, 1997, p527; Sammier et al., 2000, p981; Yi and Song, 1999, p146; Youn et al., 2006, p669; Hiromatsu et al., 1993, p24; Savaresi et al., 2003, pp2267-8).

While many controls are derived from linear techniques, there are some systems that are nonlinear even at their theoretical basis, such as bang-bang controls derived from Pontryagin’s Principle. Of course, even if a nonlinear system does not have obvious modes of resonance there is no assurance that such a system will not behave erratically under certain conditions. “The primary drawbacks of [nonlinear control techniques] include the ad hoc nature of controller synthesis and absence of robustness guarantees.” (Dixit and Buckner, 2005, p84) In the absence of robustness guarantees, extensive testing is required. Nonlinear systems cannot be guaranteed to fail only under conditions of resonance and frequency plots are insufficient to show that a nonlinear system is stable. There is some reason to believe that semi-active, nonlinear control systems, which cannot input energy, will be more stable than active, nonlinear systems, although even here testing is required. As shown in section 4.2, even semi-active systems can reach highly destructive states, although this is under extremely unlikely conditions.

As shown in the sections immediately below, jerk can be used as a control in both linear and nonlinear systems. It is possible to use jerk in mechanical systems, such as in high-speed cam profiles (Hicks et al., 2006), and in the sculpting of parts for smooth machining (Lee and Lin,
May 1998). Control over jerk has received some attention in the area of industrial robot arm manipulation (Cao et al., 1997; Macfarlane and Croft, 2003) and in lift control (Peters, 1995).

With electronically-controlled systems, both active and semi-active, it is possible to produce a desired level of chassis jerk, by output of the appropriate control. Jerk is not directly output; the control for the required jerk is first calculated in the microprocessor (or other electronic computing system). “Inverse kinematics” (Kyriakopoulos and Saridis, 1988, p364) are employed to produce the control jerk required by the algorithm. Similar remarks apply to control over acceleration, where hydraulic pressure, voltage, damper stiffness or some other directly manipulated control parameter produces the acceleration required of the algorithm.

The term “controls over jerk” is a convenient and useful shorthand terms to distinguish those controls whose control algorithms are based on jerk from those that are based on acceleration, even though the actuators may be the same. “We can think of controls as some ‘internal’ variables that need to be computed to generate the inputs for the plant in much the same way as we view state variables as being internal variables that are used to define the state of the system independent of the output” (Ross, 2009, p9). In the numerical and physical experiments performed for this thesis all such calculations are carried out before the control is applied, and controls over jerk are applied exactly in the same manner as controls over acceleration (see section 8.15).

Jerk as a control and jerk as a performance indicator are quite independent. For example, jerk can be used to measure the performance of a passive system although it is infeasible for it to be used as a control for a passive system.

### 2.3.1. Linear Control

Linear quadratic (LQ) problems involve a linear system and a quadratic cost function. The linear quadratic regulator (LQR) is shown to be the ideal optimum for this problem, derived from Pontryagin’s Principle (Kirk, 1970, pp209-17; MacCluer, 2005, p151; Ross, 2009, p19). “This is also referred to as the Minimum Principle, the Maximum Principle, Pontryagin’s Minimum Principle, and Pontryagin’s Maximum Principle.” (Ross, 2009, p19) While the LQR is the optimal for a linear system, it is itself also a linear control.
The LQR approach is readily applied to suspension systems with weighted sums of quadratic cost measures. For a single variable, the quadratic performance measure is equivalent to the root-mean-square (RMS) measure in the sense that both give the same ranking. (Perhaps it is formally correct to talk about “quadratic measures” or “least squares”, but the term “RMS” is more familiar and will often be used in this thesis.) As well as the usual measures of acceleration and vertical travel, LQR techniques can include rotational movements such as angular acceleration of pitch (Yedavalli and Liu, 1994, p1213).

In matrix form, the LQ problem seeks to minimize the quadratic cost functional,

\[ J = \int_0^T x^T Q x + u^T R u \, dt, \]

Equation 2.3

for a system described by the linear equation,

\[ \dot{x} = Ax + Bu. \]

Equation 2.4

It is assumed also that the matrix, R, is invertible. The matrices in the cost functional, Q and R, are assumed to be symmetric and positive semi-definite (Kirk, 1970, p209; MacCluer, 2005, pp139-40). In these equations x represents the state vector. LQ problems, where cost is measured over a finite time, will admit feedback controls that have time-dependent gains (Ross, 2009, p49; MacCluer, 2005, p146). Where the integral extends to infinity,

\[ J = \int_0^\infty x^T Q x + u^T R u \, dt, \]

the LQR technique with constant gain matrices is optimal. The optimal control, u, is usually found by solving a matrix Riccati equation (Kirk, 1970, p217; Dorf and Bishop, 2005, p693; MacCluer, 2005, p151; Tseng and Hendrick, 1994, p550; Giorgetti et al., 2006; Sammier et al., 2000, p978).

The linear quadratic regulator is often used to approximate a nonlinear system. Takahashi et al. use the LQR approach to find a local weighted-sum optimum (Takahashi et al., 2000). Johnson and Erkus introduced inequalities into constraints of the LQR problem in order to represent the energy dissipative constraint of the semi-active damper. “However, this
constraint is nonlinear, making the problem challenging”, and the problem is solved numerically (2002, p2264).

When modified LQ methods are compared with the skyhook in semi-active applications, the skyhook has been found to have better performance (Wagner and Liu, 2000, p568). In numerical experiments on active systems, LQR methods give remarkable results (Wagner and Liu, 2000).

Because LQR control is later extended to control over jerk, the corresponding control over acceleration is examined here in perhaps a little more detail than would ordinarily be necessary, because the control is very well known. Let \( u \) represent the control over vertical acceleration,

\[
u = \ddot{y}.
\]

In matrix form, as in equation 2.4,

\[
\dot{x} = \frac{d}{dt} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} = Ax + Bu = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u = \begin{bmatrix} \dot{y} \\ u \end{bmatrix}.
\]

(The cost for velocity is not included here because it is being assumed that the performance entails a combination of tracking and comfort, with tracking corresponding to the distance performance measure and comfort corresponds to acceleration.) The cost functional takes the form,

\[
J[x] = \int_0^\infty x^T Q x + u^T R u \ dt = \int_0^\infty \begin{bmatrix} y & \dot{y} \end{bmatrix} \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} + u^2 \ dt = \int_0^\infty \alpha y^2 + \beta \dot{y}^2 \ dt.
\]

Solving the Riccati equation (as shown in section 8.2) gives the linear control,

\[
u = -\sqrt{\alpha} \begin{bmatrix} y \\ \dot{y} \end{bmatrix} = -2\beta^2 y - 2\beta \dot{y}.
\]

Equation 2.5

Here \( \beta = \frac{\sqrt{\alpha}}{\sqrt{2}} \) is a convenient simplification.

This can be found more directly using Euler-Lagrange equations (MacCluer, 2005; Smith, 1998). The problem is to minimize,
\[ J = \int_0^\infty \alpha y^2 + \dot{y}^2 \, dt = \int_0^\infty f(t, y, \dot{y}, \ddot{y}) \, dt, \]

Where the function \( f \) is given as,
\[ f = \alpha y^2 + \dot{y}^2. \]

The Euler-Lagrange equation (the form that includes second-order derivatives) becomes,
\[
\frac{\partial f}{\partial y} - \frac{d}{dt} \frac{\partial f}{\partial \dot{y}} + \frac{d^2}{dt^2} \frac{\partial f}{\partial \ddot{y}} = 2\alpha y + 2\dot{y}^{(4)} = 0.
\]

The roots of the characteristic equation for this linear differential equation are the fourth roots of \( \alpha \). This has stable solutions of the form,
\[ y(t) = e^{-\beta} \left( A \sin(\beta t) + B \cos(\beta t) \right). \]

The following has been used to simplify the expression,
\[ \beta = \frac{\sqrt[4]{\alpha}}{\sqrt{2}}. \]

The parameters \( A \) and \( B \) are determined from initial conditions, \( y(0) = d \) and \( \dot{y}(0) = v \).

These are easily solved to give,
\[ y(t) = e^{-\beta} \left( \frac{d\beta + v}{\beta} \sin(\beta t) + d \cos(\beta t) \right). \]

It remains to show that this satisfies the linear equation 2.5 above. This is easily done by performing the algebra to verify that,
\[ \ddot{y} + 2\beta^2 y + 2\beta \dot{y} = 0. \]

See Maple code to perform these operations in section 8.3. This verifies the result found using Riccati equations.

Note that this linear suspension is simply a passive suspension. It is a slightly underdamped linear passive system with a damping ratio of,
\[ \zeta = \frac{1}{\sqrt{2}} = 0.707. \]

Most cars are much more underdamped than this, with a damping ratio around 2.5 or 3 (see section 2.2.2).

As discussed above, optimal solutions to LQ problems admit constant feedback gains. Such controls are “called proportional-plus-derivative- or simply PD-controllers because the gains
are (linear) functions of ‘errors’ in \textit{x} and its derivative” (Ross, 2009, p49). Proportional-plus-derivative-plus-integral (PID) controllers also include feedback on the integral (Dorf and Bishop, 2005, p391). Some LQ problems require variable gains and “time-dependent gains are known under the heading, gain scheduling, although gain scheduling is frequently not optimal. Much of control theory of the 20th century has been dominated by assuming that control is given in terms of gains and the task is [to] find the right set of gains” (Ross, 2009, p49).

“The Linear Quadratic Regulator (LQR) has been used as one of the main control techniques for dealing with active suspension design” (Camino et al., 1999, p3168). However, linear quadratic controls are not the only form of optimal control, and optimality depends on the optimality problem, including both the system and the performance measure. The LQR will not generally be optimal even for a linear system if a non-quadratic performance measure is used.

2.3.1.1. Modelling Discretised Linear Control

A linear system,

\[
\dot{x} = Ax + Bu,
\]

is known to have the continuous solution,

\[
x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau.
\]

Suppose the system is discretised for step-by-step control, as with control by a microprocessor. Suppose too that the step size is represented as \(h\),

\[
t_{k+1} - t_k = h,
\]

and the control, \(u(t)\), is constant between \(t_k\) and \(t_{k+1}\) and is constant and equal to \(u_k\). Therefore,

\[
x_{k+1} = e^{Ah}x_k + \int_{0}^{h} e^{A(h-\tau)}d\tau Bu_k,
\]

which can then be placed in the form,

\[
x_{k+1} = F_k x_k + G_k u_k,
\]

where \(F_k\) and \(G_k\) are the matrices as defined in the previous equation (Simon, 2006, p27).
2.3.1.2. Kalman Filter

The Kalman filter is a method for state-estimation (Simon, 2006) that is often applied in control systems. It is designed as a method to optimize the accuracy of state estimations. It was applied originally to linear systems, although nonlinear versions now exist. The method discussed here is a discrete-time method appropriate for microprocessor control, although continuous versions exist.

It is a fundamental result of statistics that least squares values are also the maximum likely estimator assuming Gaussian white noise, linearity, independence and equal population variances (Larson, 1982, p482). With a time series of such values a convenient recursive estimator, the Kalman filter may be relatively easily implemented (Simon, 2006, p84). Such a method can be employed as a state estimation phase in a discrete control. The Kalman filter was used for the Apollo space program in the 1960s (Simon, 2006, p487), but the theory of the filter “has its roots in the early 1700s in the least squares work of Roger Cotes” (Simon, 2006, p485).

The discrete-time method is very fast and has been investigated for use in a wide range of applications including controlled suspensions. The Kalman filter has been investigated for use with suspension systems (Lee et al., 2008; Yu et al., 2000; Yoshimura et al., 1987; Best et al., 2007; Sadati et al., 2008).

The actual states (which are hidden from the observer), $x_k$, follow a linear discrete-time model,

$$ x_k = F_{k-1} x_{k-1} + G_{k-1} u_{k-1} + w_{k-1}, $$

while the measurements, $y_k$, are given by,

$$ y_k = H_k x_k + v_k. $$

The terms, $w_k$ and $v_k$, represent zero-mean, stochastic processes. The convergence of the Kalman filter can be calculated directly from covariance matrices, or as in the experiments.
conducted here, with simple time invariant steps, the multipliers can be estimated from previous experiments with a range of values.

Each step of the method involves two stages:
- Finding the \textit{a priori} measurement estimate at time \( k \), \( \hat{x}_k^- \), and,
- Finding the \textit{a posteriori} measurement estimate, \( \hat{x}_k^+ \).

The method proceeds by estimating the process error covariance as well as estimating the process states, which is the only data that needs to be stored between time steps, except for an additional matrix, \( K \), and the constant parameters of the problem. The mathematical processes for the discrete-time Kalman filter are summarised in Simon (2006, pp128–9).

The method statistically minimises least squares errors in estimates, \( \hat{x}_k \), of the true system state, \( x_k \), at times \( k \), given all the previous measurements, \( y_k \), up to time \( k \). It provides the optimal solution if the stochastic processes are zero-mean, uncorrelated, white and Gaussian, but it also provides the optimal linear filter if they are zero-mean, uncorrelated, white but non-Gaussian (Simon, 2006, p130).

When considering control over the first differential, \( u = \dot{x} \), for time invariant systems, it is possible to derive a particularly simple, scalar version of the Kalman filter equations:

\[
\hat{x}_k^+ = \hat{x}_k^- + K(y_k - H\hat{x}_k^-) = Ky_k + (I - KH)\hat{x}_k^- \\
= Ky_k + (I - KH)(F\hat{x}_{k-1}^- + Gu_{k-1}) \\
= Ky_k + (1 - K)(\hat{x}_{k-1}^+ + \Delta tu_{k-1}).
\]

Note that, since the system is time invariant, the model matrices, \( F \) and \( G \), are time invariant, but so are the matrices of the Kalman filter itself, \( H \) and \( K \), and so subscripts are not needed on these matrices. The convergence of the Kalman filter can be calculated directly from covariance matrices, or as in the experiments conducted here, with simple time invariant steps, the multipliers can be estimated from previous experiments with a range of values.

Similarly, an estimate of a derivative, \( \dot{x}_{k-1} \), can provide improved state estimation of a given state scalar by replacing \( u \) in the previous equation:

\[
\hat{x}_k^+ = Ky_k + (1 - K)(\hat{x}_{k-1}^+ + \Delta t\dot{x}_{k-1}).
\]

\textbf{Equation 2.6}
In the physical experiments discussed in section 6 it is more convenient, and equivalent, to use the same scalar equation three times to avoid higher-order matrices. The various Kalman gain values of the experiment were determined by simply experimenting with values that gave an acceptable compromise between good smoothing (low $K$ values) and low time lag (high $K$ values).

### 2.3.2. Nonlinear Control

While linear control is well understood, with a history covering centuries, there are a wide variety of physical systems which do not submit to analytical techniques. With linear systems, system failure is generally accompanied by resonance indicated by large eigenvalues (Strang, 1980, p183). The failure of nonlinear systems is less well known, and even optimal analytical controls can fail in unpredictable ways due to slight variations in parameters or given unanticipated stochastic input. Nonlinear systems require stringent demonstrations of robustness. To test for robustness detailed computer modelling can be used (Krüger, 2002, p500) as well as prototype testing.

The use of evolutionary algorithms during development can help to retain robustness (Fleming, 2001). “The advantage of using an [evolutionary optimization] is that a global robust solution can be obtained and the method can be extended for finding multi-objective reliable solutions easily” (Deb, 2005, p14). “Often in practice, the mathematical optimum solution is not desired, due to its sensitivity to the parameter fluctuations and inaccuracy in the formulation of the problem” (Deb, 2005, p13). “Consider [figure 2.2], in which although the global minimum is at $B$, this solution is very sensitive to parameter fluctuations. A small error in implementing solution $B$ will result in a large deterioration in the function value. On the other hand, solution $A$ is less sensitive and more suitable as the desired solution to the problem. The dashed line is an average of the function around a small region near a solution. If the function shown in dashed line is optimized the robust solution can be achieved.” (Deb, 2005, p13)
Linear control is optimal only in a range of problems. As noted above, the linear LQR technique minimizes a quadratic (almost equivalently RMS) performance measure, for example, equation 2.3, when returning a system asymptotically to rest. If the performance index is changed to a measure of the time spent returning to rest using a constrained control (see section 2.3.2.3), the optimal control becomes nonlinear, and has a completely different character (MacCluer, 2005, p120). Pontryagin’s Principle applies in such cases and the optimal control is not only nonlinear, it is discontinuous, with sudden changes between states. These are known as bang-bang controls (Kirk, 1970, p259; MacCluer, 2005, p116). In textbook examples this kind of control is often applied to the minimization of rocket fuel use, or to economic problems. A change in the problem formulation produces wildly different optimal solutions, even for problems that are superficially similar: for example returning a system to rest.

With bang-bang control over acceleration sudden discontinuities of force are experienced. Such discontinuities cause great discomfort and yet this discomfort remains undetected by the RMS measure of acceleration, as shown below in section 3.1. The discontinuities of bang-bang controls over acceleration are not properly recognized in the performance measure
because they have no effect on it. Force discontinuities also arise in sliding-mode control, in which the control is switched at the “sliding surface”. The pure sliding-mode control can produce rapidly changing accelerations, often called “chattering” (refer to section 2.3.2.4). In the case of the sliding-mode control, chattering is in practice mitigated by using a linear control near the switching surface to soften the discontinuity (see section 2.3.2.4 below). Similarly, the adaptation of an active control for a semi-active suspension, such as the “clipped optimal” control (discussed in section 4.8.1), will generally result in discontinuities as the stroke velocity of the controlled damper changes direction.

Optimal control should not be associated with open-loop solutions to idealized analytical problems. Some optimal controls admit feedback implementations. This is certainly the case with the optimal controls for the LQ problem (as shown in section 2.3.1) that admit passive implementation.

Feedback controls can also be found for other problems beside LQ problems, including minimum-time problems (as shown in section 4.6.1). According to Ross,

“In broad terms, trajectory optimization refers to solving an optimal control problem to a very high accuracy whereas feedback control implies a real-time computation of a control solution. In trajectory optimization, we typically have accurate system models, and hence, seek to find accurate solutions to an optimal control problem. In feedback control, we typically have inaccurate system models, and hence, seek to manage the uncertainty by requiring real-time computation. Thus, the holy grail in optimal control theory is a means to solve problems accurately and in real time.” (2009, p17)

Feedback systems have to contend with inaccurate system models, but also state estimation errors (refer to section 2.3.1.2), stochastic vibration (discussed in section 2.3.2.1) and latency: “a question of great practical and theoretical value is the maximum amount of computational delay a closed-loop control system can withstand before it behaves ‘badly’” (Ross, 2009, p48).
2.3.2.1. Stochastic Input

Linear systems are completely described by their frequency response and their response to transients. For nonlinear systems this is not enough and the designer needs to test with random road profiles that match the statistical properties of actual roads.

It can be useful, in the first instance, to look at the transient response of the unperturbed system. MacCluer used such analyses to derive an insightful comparison of the strengths and weaknesses of simple tracking controls (2005, pp152-6). As discussed above in section 2.3, a number of input forms can be used for at least first-order testing: step functions (Goncalves and Ahmadian, 2002), chirps functions (Goncalves and Ahmadian, 2002; Song et al., 2003), sine waves (Hönlinger and Glauch, 2000; Sims and Stanway, 2003) and randomly generated road profiles (see section 2.8).

As discussed above, analysis shows that the LQR technique returns a system to rest optimizing the quadratic performance measure. The LQR control can then be applied to a perturbed system (stochastic road surfaces) by using the current state as the basis of discrete feedback control. In the same way any method that returns a system to rest can be applied as a suspension for a system that is stochastically perturbed. For instance, the algorithm that returns a system to rest in minimum time can be used as the basis of a suspension (and in fact this becomes a special case of a sliding-mode suspension as explained below in section 2.3.2.4).

If the road changes height on a medium-term basis (over a matter of seconds) or if the road enters a long slope, the suspension must re-zero to the changed conditions. Generally this will require the use of a high-pass filter. Models of maglev suspensions employing skyhook control used by Paddison et al. needed to use a high-pass filter (with a corner frequency of 10 Hz) to correct for long gradients (1994, pp600-1). Such a filter is easily implemented in a digital system using a z-transform (Papoulis, 1980). High-pass filters are also needed to correct for drift in numerical methods (see section 2.10.3).
2.3.2.2. Performance Measures

Various suspension performance measures used in the literature are covered in much greater depth in the next major section, section 3. The various performance indexes are described and compared in that section, and some new measures and theory are introduced.

In the case of the numerical experiments performed for this thesis the performance measure is a weighted sum of the integrals of a nonlinear distance measure and the fourth power of jerk,

\[ J[y] = \int_0^\infty \alpha \varphi(y(t)) + \dddot{y}(t)^4 \, dt, \]

where \( \alpha \) is a weighting factor, \( \varphi(y) \) is the nonlinear rattlespace penalty (see section 3.2) and \( y(t) \) is the height of the chassis. If this problem had a fast analytical solution, this could be used as a feedback control. Unfortunately, problems such as this are generally extremely complex analytically, and are quite often intractable. However, similar problems do have analytical solutions and they can suggest controls, both closed- and open-loop, which may be able to perform better than the linear control.

Even if the nonlinear rattlespace penalty is greatly simplified to a double-step function,

\[
\varphi(y) = \begin{cases} 
0 & \text{if } |y| < R, \\
\ P & \text{if } |y| \geq R,
\end{cases}
\]

Equation 2.7

the problem remains intractable. Note however, that as \( P \) increases, as greater weight is placed on keeping the suspension inside the rattlespace, the optimal control will “try to avoid” hitting the rattlespace limits. This suggests the control problem of maximizing comfort while remaining within the vertical displacement constraints of the rattlespace. This is the inspiration behind the “rattlespace constraint” controls discussed in section 4.7.
2.3.2.3. Minimum-Time Problem

The problem of returning a system to rest in minimum time using a bounded acceleration, \(|\dot{y}(t)| \leq a\), is a common example of a constrained control (MacCluer, 2005, p117; Smith, 1998, p305; Ross, 2009, p62; Kirk, 1970, p249). The optimal solution is a bang-bang control derived according to Pontryagin’s Principle (MacCluer, 2005, p120). The two-stage bang-bang solution can be neatly represented in a phase-space diagram, as in figure 2.3. This is essentially the same figure originally used by Pontryagin et al. as an example of feedback time-optimal control (1986, p26). The phase-plane curve for an application of constant acceleration is obtained by eliminating time, \(t\), from the equations for velocity and displacement,

\[
\dot{y} = \dot{y}_0 + at, \\
y = y_0 + \dot{y}_0 t + \frac{1}{2} at^2.
\]

This produces,

\[
y = y_0 - \frac{\dot{y}_0^2}{2a} + \frac{\dot{y}^2}{2a}.
\]

The acceleration, \(a\), is here either positive or negative. To return to rest, first follow the curve that intersects with the switching curve and then return to zero using the opposite acceleration along the switching curve.

![Figure 2.3 Phase-plane Diagram for Minimum Time using Acceleration](image)

43
The control force that is used in the minimization problem is the largest force available within the constraints of the physical system. Note that if force is unconstrained, the optimal control applies an infinite force returning to rest in an infinitely small time: a clearly absurd, impractical and infeasible solution.

The bang-bang control is “optimal” relative to the conditions for which Pontryagin’s theorem applies, i.e. a linear system with bounded control, but also relative and the performance goal of fastest-time. It is not optimal in the sense of other performance measures, such as minimum RMS acceleration.

Pontryagin’s theorem is applied to prove that a constrained control over acceleration requires at most one “intermediate switch” of control direction to bring a system to rest in minimum time (there are also two other switches to and from zero control at the start and at the end). The mathematical proof is shown here since it is later extended to control over jerk.

What is the minimum time required to return a system to rest using constrained acceleration?

The control time is minimized by minimizing the functional,

\[ Q = \int_0^T 1 \, dt, \]

The absolute value of control acceleration, \( u \), is constrained to be less than a constant value, \( a \):

\[-a \leq u \leq a.\]

The equation of motion in vector form is,

\[ \dot{x} = F(x) = F(x, v) = (v, u). \]

That is to say, \( u = \dot{v} \) and \( v = \dot{x} \). Here \( x \) and \( v \) represent distance and velocity respectively.

The method given in MacCluer, chapter 7, can be applied since the problem is similar to the “rolling cart problem” (2005, p125). The Hamiltonian is,

\[ H = -1 + \lambda^T F = -1 + (\alpha, \beta). (v, u) \]

\[ = -1 + \alpha \dot{v} + \beta u. \]

Maximizing the Hamiltonian produces,
\[ u = \begin{cases} -a & \text{if } \beta < 0, \\ a & \text{if } \beta > 0. \end{cases} \]

**Equation 2.8**

The adjoint differential equation is,

\[
(\dot{\alpha}, \dot{\beta}) = \dot{\lambda} = -H_x = -\left( \frac{\partial H}{\partial \xi}, \frac{\partial H}{\partial \psi} \right) = (0, -\alpha).
\]

Therefore,

\[ \alpha = \alpha_0, \]
\[ \beta = -\alpha_0 t + c, \]

where \( \alpha_0 \) and \( c \) are constants. Since the equation for \( \beta \) is linear, it can cross zero at most once. When \( \beta \) crosses zero, the control \( u \) switches direction, using equation 2.8. Thus there is at most one switching for the control, \( u \), which returns the system to rest in minimum time.

An alternative proof that there is at most one switch for this control under slightly relaxed conditions uses a method discussed in Hermes and LaSalle (1969) and is shown in the appendix in section 8.7.

Given that the final position of the control is zero (zero distance and zero velocity), the final application of acceleration must lie on the switching curve, as shown in figure 2.3. Thus the switch occurs when the control reaches the switching curve. The entire algorithm is elegantly represented by Pontryagin’s diagram, similar to figure 2.3. From any starting position in phase space that it not on the switching curve, the system must first enter the switching curve. It then returns to rest along the switching curve. In a typical application of this algorithm there is one switch in control direction when the system reaches the switching curve, between full acceleration in one direction to full acceleration in the opposite. (There are also switches from zero and to zero at the start and end.)

As noted by Ross a real-time feedback control to implement this control requires a computational procedure to represent the switching curve (2009, p66). If the current point is above the switching curve then implement negative acceleration, otherwise use positive acceleration.
This represents a principle that is used a number of times throughout the thesis. The general concept is known as the Principle of Optimality:

“An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.” (Bellman, 2010, p83; Bellman and Dreyfus, 1962, p15)

This has been used in Dynamic Programming to determine optimal controls (Bellman and Dreyfus, 1962; Kirk, 1970, p54) but it also contains the rationale behind the reason that optimal controls can translate into feedback or Real-Time Optimal Controls (RTOCs). This has been expressed by MacCluer as the principle that, “if [control] $u$ is optimal on $[t_1,t_2]$, it is optimal on every subinterval” (2005, p121). (This is readily proven by contradicting the negation.)

Just as constrained acceleration can bring a particle to rest in minimum time using a bang-bang control over acceleration, so a bang-bang control over jerk can bring a particle to rest in minimum time using constrained jerk. See section 4.6 for an analogous proof that this requires two switches of control direction instead of one.

Bang-bang control over jerk has been studied for the control of industrial robot arms (Muenchhof and Singh, 2003; Koh et al., 1999; Kyriakopoulos and Saridis, 1988). Robot arm movement does not occur in a perturbed environment and most studies of robot arm movement have concentrated on movement beginning and ending with zero velocity and zero acceleration. Similarly a lift rises or lowers from one floor to another with zero velocity and acceleration at either end and lift control using jerk has concentrated on such movement (Peters, 1995). With one degree of freedom this produces the simple symmetric control shown in figure 2.4. If the control takes time, $T$, to bring the system to rest then there are changes in control direction at times, $\frac{T}{4}$ and $\frac{3T}{4}$. Finally at time $T$, distance, velocity and acceleration are simultaneously brought to rest. Koh et al. show a method that can be applied from moment-to-moment, but the only example they give in the paper of 1999 is again the simple symmetrical system. The method is verified by numerical experiments as discussed below in section 4.6. The method was developed in the course of the PhD research by this
A wider literature search later revealed that this algorithm had been advanced by Koh et al. in the area of mechatronics, verifying the technique.

Figure 2.4 Example of Minimum-Time Control over Jerk

Muenchhof and Singh studied controls that were simultaneously jerk limited and acceleration limited. Jerk is applied until acceleration reaches “the saturation level” (Muenchhof and Singh, 2003, p139; Koh et al., 1999), at which time the jerk is turned off. Such controls are called bang-off-bang controls. According to Muenchhof and Singh, “numerical results show that accepting a small increase in the final time, the jerk can be reduced considerably” (2003, p142).

The minimum-time algorithm can be applied on a continuous basis to a randomly perturbed system, as discussed in section 2.3.2.1. After a large disturbance, this control will tend to move the system onto the switching curve and return the system to equilibrium. This can be implemented as a special case of a sliding-mode control (discussed in the following section). Koh et al. proposed the control for robotic arm movements (1999) where robotic arm movement is well-orchestrated and could not be said to be stochastically perturbed except for minor vibrations.

Constrained jerk control has also be applied by Ben-Itzhak and Karniel in the context of robotic arm movement, for the purpose of optimizing RMS acceleration over a given finite time period (2008). This results in a somewhat different control however.
2.3.2.4. Sliding-Mode Control

Sliding-mode control as a suspension control design paradigm has been investigated by a number of researchers (Ashari, 2004; Donahue, 2001; Yagiz, 2005; Dixit and Buckner, 2005; Yokoyama et al., 2001) and the method is claimed to be quite robust, at least in the sense that “the closed-loop response becomes totally insensitive to a particular class of uncertainties and disturbances” (Ashari, 2004, p370). “Sliding mode control was proposed first in the Soviet Union by Emelyanov and Utkin” (Yagiz et al., 2000, p80). Yagiz employs sliding-mode control for an active suspension system, applying Lyapunov Stability Criteria (2000, p82). Zribi and Karkoub apply sliding-mode control to a semi-active suspension using a model of an MR damper, and they show that the system is stable (2004, pp519-20).

Sliding-mode control attempts to place a system in state-space onto a surface, the “sliding surface” (Ashari, 2004, p370) also sometimes called “switching surface” or “sliding manifold”. This is redolent also of the “landing domain” or “landing surface” used for bang-bang control by Koh et al. (1999, p274). Briefly, the method uses available control, usually constrained force, to move the system onto a surface of the phase space, as depicted in figure 2.5. By repeating this process, perhaps using electronic sensors and electronic controls, the control slides back along the manifold to the equilibrium position. Note that the continuous minimum-time solution, discussed in the previous section, is a special case of a sliding-mode control.

Figure 2.5 Phase-Plane Diagram of Sliding-Mode Control (Yagiz et al., 2000)
A major disadvantage of pure sliding-mode control is a phenomenon known as “chattering”, which is “high frequency switching” of the control signal as the system repeatedly crosses the sliding surface (Ashari, 2004, p370). Generally, some method is used to soften the control as it approaches the sliding surface (Ashari, 2004, p371; Dixit and Buckner, 2005, p93; Liu et al., 2005, p1029; Yokoyama et al., 2001, p2654). One way to do this is to use a linear control in the “boundary layer”, a layer close to the switching plane. “Inside the boundary layer, the switching function is approximated by a linear feedback gain” (Young et al., 1999, p329).

The most effective way to do this is to base the sliding surface on a control, say a linear control, and to use the control of the surface itself when “near” the surface. Thus the system returns to rest along the surface rather than switching quickly on either side of the surface as in figure 2.5.

Without this modification, using control over acceleration results in a system that will suffer chattering around the switching surface. Prior to electronic control, “chattering” has been associated with imperfect control because of “limited actuator dynamics” (Lorenz et al., 2003, p381), or because of imperfections in passive components such as loose bushing or hydraulic valving in dampers. In this instance, however, “chattering” in the unmodified control is a consequence of the very control design itself.

The RMS measure of acceleration does not necessarily indicate that chattering is uncomfortable. As discussed below in 3.1, a control system that has quick jumps between relatively small values of acceleration will not penalize those jumps. But these “jumps” are associated with high values of jerk. However, the use of jerk as a comfort measure will immediately penalize the discontinuities as being periods of very high jerk. With electronic control, fast control switching is possible and may even seem desirable: it is generally the easiest control to implement, and Pontryagin’s theorem may indicate that it is optimal. If RMS acceleration alone were used as an indicator of comfort, the pure switched version of sliding-mode control would be hardly different in comfort from the control that returns smoothly along the sliding surface (and switches smoothly onto the sliding surface). This is a practical example where RMS acceleration is deficient as an indicator of suspension comfort, and has been shown to be deficient in the literature (Ashari, 2004).
Even in the case of control systems in which a minimum-time algorithm could seem to be well suited, such as elevator controller optimization, linear analysis is applied, as in Fathy (2003a) which avoids the production of discontinuous forces.

2.4. **Skyhook**

The purely linear skyhook suspension control is ubiquitous in the literature (Karnopp, 1995; Burton, 1993; Paddison et al., 1994; Reichert, 1997; Elbeheiry and Karnopp, 1996; McLellan, 1998; Wagner and Liu, 2000; Goncalves and Ahmadian, 2002; Ahmadian et al., 2004; Donahue, 2001; Song et al., 2003; Song and Ahmadian, 2004; Stembridge et al., 2006; Li et al., 2004; Williams and Best, 1994; Hyvärinen, 2004; Krüger, 2002; Caponetto et al., 2003; Guglielmino et al., 2008, p70). The control is very simple and has been applied with success by many researchers. Karnopp claimed in 1995 to have been associated with the Skyhook concept for 30 years (1995, p177), which makes the skyhook control very old indeed in the field of electronically-controlled suspension systems. Karnopp and Crosby produced a patent in 1974 for a suspension that used feedback of the chassis height for actuator control (1974).

The skyhook is claimed to be “theoretically capable of isolating the body mass from the road input and also reducing the tyre load fluctuations” (Stembridge et al., 2006, p1). It is also claimed that studies “indicate that skyhook control is the optimal control policy in terms of its ability to isolate the suspended mass from the base excitations” (Reichert, 1997, pp12-3). In fact, the skyhook is only “optimal” in a very limited sense. In contrast, the LQR control is mathematically optimal (for quadratic performance measures). Nonetheless, the skyhook has great simplicity and a proven track record that has earned it an iconic status in the literature, and throughout this thesis a number of possible reasons for its success are suggested.
The skyhook can be schematically represented as in figure 2.6. The damper in a skyhook system is attached to the “sky”. Of course such an arrangement is impossible physically. This is a “virtual” control in that this arrangement is physically impossible. Rather, the skyhook is a control law which can be implemented by computer-controlled suspension. The force provided by the suspension mimics the force provided by the arrangement shown in figure 2.6. The damper in this sense is a “virtual damper”. Even the spring in the virtual control may have a different spring rate than any actual spring employed in the suspension.

The force supplied by the skyhook’s “damper” is proportional to the absolute chassis velocity rather than its velocity relative to the road. If $s$ is used to represent stroke, $s = y - r$, then the damping force supplied by the skyhook is proportional to absolute chassis velocity,

$$F_d = -c\dot{y},$$

rather than the damping force being proportional to relative chassis velocity (stroke) as in the passive suspension,

$$F_d = -c\dot{s}.$$

Stembridge et al. discuss a modified skyhook law (2006) which is simply a system using both a damper attached to the road (or unsprung mass) and a separate damper attached to the “sky”,

$$F_d = -c\dot{s} - c_{sky}\ddot{y}.$$
\[ F = -k_s - c \dot{y}. \]

**Equation 2.9**

This is taken as the prototypical form of the skyhook by Reichert (1997) and this author will do likewise. Nonetheless the linear form of equation 2.9 is not the only control called a skyhook. Generalizing the concept of the skyhook, Paddison et al. assert that, “feedback of the absolute velocity is often referred to as ‘skyhook damping’” (1994, p602).

Semi-active variants of the skyhook are necessarily nonlinear. These can be viewed as approximations to the basic skyhook. In some cases they are very rough approximations indeed, as in the on-off semi-active skyhook (McLellan, 1998, p55). In other cases the closest available control is used, turning the damper off when the required force lies outside the passivity constraint (Guglielmino et al., 2008, p70). Refer to section 2.6 for an explanation of the passivity constraint.

In the model above, the skyhook is applied to heave (vertical displacement), but in half-car and full-car models the skyhook control generalizes to controls over all modes of vibration including rotations, especially pitch (Hyvärinen, 2004, p74). The pure skyhook is linear and it can be analysed in terms of transmissibility. It is useful to compare the skyhook with the purely linear passive control. The equation of motion for the linear passive suspension as shown in figure 2.1 is given as,

\[ m \ddot{y} + c \dot{y} + k y = c \dot{r} + k r. \]

The undamped natural frequency in radians/sec, \( \omega_n \), and the damping factor, \( \zeta \), are defined as,

\[ \omega_n = \frac{k}{\sqrt{m}}, \]

\[ \zeta = \frac{c}{2 \sqrt{km}}. \]

(Stubberud et al., 1994, p48). Thus,

\[ \ddot{y} + 2 \zeta \omega_n \dot{y} + \omega_n^2 y = 2 \zeta \omega_n \dot{r} + \omega_n^2 r. \]

The transmissibility at frequency \( \omega \) is given as,
\[
\frac{|Y|}{R} = \left| \frac{1 + j2\zeta \left( \frac{\omega}{\omega_n} \right)}{1 - \left( \frac{\omega}{\omega_n} \right)^2 + j2\zeta \left( \frac{\omega}{\omega_n} \right)} \right|
\]

(Reichert, 1997, p9).

Graphs of the magnitude of transmissibility for a number of values of damping factor are shown in figure 2.7.

The equation of motion of the skyhook does not have a term containing \( \ddot{r} \),
\[
\ddot{y} + 2\zeta \omega_n \dot{y} + \omega_n^2 y = \omega_n^2 r.
\]

The transmissibility at frequency \( \omega \) is given as,
\[
\frac{|Y|}{R} = \left| \frac{1}{1 - \left( \frac{\omega}{\omega_n} \right)^2 + j2\zeta \left( \frac{\omega}{\omega_n} \right)} \right|
\]

(Reichert, 1997, p12)

Graphs of the transmissibility for the skyhook are shown in figure 2.8.

![Graph of Transmissibility for Passive Suspension](image)

**Figure 2.7 Transmissibility for Passive Suspension**
The skyhook performs dramatically better than the passive suspension at higher frequencies. Passive suspensions used in passenger cars typically have a damping factor less than 0.2. The skyhook reduces transmissibility at the fundamental frequency but drastically reduces transmissibility for higher frequencies at higher values of the damping factor, $\zeta$.

Notice also that below $\sqrt{2}\omega_n$, the passive suspension has transmissibility greater than one, no matter what damping ratio is chosen. For the passive suspension, there is a trade-off; high frequency attenuation is associated with greater transmissibility around the fundamental frequency. With the skyhook, however, the suspension can attenuate all frequencies, with high damping ratios.

### 2.5. Groundhook

The groundhook control is a notional control similar to the skyhook. “Like skyhook, the groundhook damper is supposed to be hooked to a fixed point, in this case the ground” (Guglielmino et al., 2008, p70), as shown in figure 2.9. The groundhook effectively dampens the unsprung mass. Koo et al. investigated a number of forms of the groundhook control law.
(2003). Similar laws are used by Goncalves and Ahmadian (2002) and by Hyvärinen (2004, p42), who looked more closely at the effect of different types of tyres. Looking at the groundhook model, the chassis mass seems to be being used as an “absorber” for the unsprung mass (Jalili, 2002).

![Diagram of Notional Groundhook Suspension]

Figure 2.9 Notional Groundhook Suspension

According to Guglielmino et al., “This logic ... aims at reducing dynamic tyre force, thus improving handling and at the same time reducing road damage (particularly useful in the case of heavy vehicles)” (2008, p70).

## 2.6. Passivity Constraint

The passivity constraint is a major problem for semi-active suspensions. Semi-active systems vary control parameters with minimum input of external energy; semi-active systems do “not require either higher-power actuators or a large power supply” (Cho et al., 1999, p667).

Theoretically, systems that vary spring rate could be called semi-active (Jalili, 2002). There are cases of springs that are adjustable, but springs that are continuously variable at high rates under electronic control are not used. “Semi-active suspension” almost invariably refers to
systems with variable dampers: dampers whose damping rate is variable from one moment to the next under electronic control. This will be the understood meaning of semi-active suspension in this thesis.

Suppose a controllable damper is capable of any damping rate between zero and infinity (at very high damping rates the damper acts like a stiff rod connecting the chassis and the wheel). Such a damper is still limited in the forces that it can apply to the chassis. Suppose the chassis is moving downwards faster than the wheel. (It may help to imagine that the chassis is moving down and the wheel is moving up.) The damper is compressing. The damper can supply any upward force to the chassis between zero and infinity. In the “infinite” case, the damper effectively connects the chassis with the wheel, and there is no “suspension”. However, the damper cannot exert a force downwards on the chassis; it can only exert a force upwards. The forces that the damper can supply are limited to one direction. When the damper is compressing, the force supplied by the damper can only be upwards. When the damper is extending, the force can only be downwards. Here hysteresis effects have been ignored.

Semi-active suspensions, using damper stiffness control, thus suffer from a constraint on the force that they are able to deliver. The term “passivity constraint” has been used to describe this limitation (Yi and Song, 1999, p147; Giorgetti et al., 2006; Sergio M. Savaresi et al., 2003, p2264; Jalili, 2002, p600; Yokoyama et al., 2001). The constraint has been expressed in terms of damping rate, which cannot be negative (Hyvärinen, 2004, pp31-2). The region of forces that can be supplied by the semi-active damper has also been called the “working region” (Stamatov et al., 2008, p1).

An important unwanted side effect of using a semi-active suspension to follow a general control (the skyhook for example) is that there will be a sudden force discontinuity as the stroke velocity crosses zero. According to Ahmadian et al., “at zero crossings of the velocity, conventional skyhook introduces a sharp increase (jump) in damping force, which, in turn, causes a jump in sprung-mass acceleration. This acceleration jump, or jerk, causes a significant reduction in isolation benefits that can be offered by skyhook suspensions” (2004, p580). This artefact of semi-active control is important in this thesis and is here termed “crossover jerk”. It is discussed at length below in section 4.8.
If it is assumed that the damper in a semi-active system is ideal and can be as stiff or as soft as desired, but not negative, then the overall force that can be supplied by the damper is limited in range,

\[ F_d > 0 \text{ if } \dot{s} < 0, \text{ and } F_d < 0 \text{ if } \dot{s} > 0. \]

(Jalili, 2002, pp602-3)

A slightly more accurate model includes finite limits on damping rate, \( c_{\text{min}} \), and \( c_{\text{max}} \), as in figure 2.10 (Koo et al., 2003, p2; Hyvärinen, 2004, p91). The notion that the limits lie between fixed damping rates may also be somewhat unrealistic. Where more accurate models are required the limiting regions may be bounded by curves (Krüger, 2002, p498) that need to be determined experimentally and which may vary with different devices. Further complications are introduced by hysteresis, as noted in section 2.10.4.

![Figure 2.10 Passivity Constraint](image)

The passivity constraint is of particular concern for the development of semi-active suspensions. Because the damper is restricted to providing force in only one direction semi-active systems can produce high jerk when the damper changes direction. A control law that is based on the skyhook suspension but which removes the force discontinuity has been proposed (Reichert, 1997, p63; Ahmadian et al., 2004; Ahmadian and Vahdati, 2006). This law has been called the “no-jerk skyhook” (Ahmadian et al., 2004, p580), and it can be written as,
\[ i = \begin{cases} \gamma \dot{y} \dot{s} & \text{where } \dot{y} \dot{s} > 0, \\ 0 & \text{where } \dot{y} \dot{s} \leq 0. \end{cases} \]

(Reichert, 1997, p63).

2.7. **Semi-active Skyhook**

Almost all semi-active control suspensions in the literature are based on the skyhook. A very simple approximation to the ideal skyhook is the on-off skyhook (Ahmadian et al., 2004, p4; Hashiyama et al., 1995, p168; Savaresi et al., 2003, p2265; Simon and Ahmadian, 2001; McLellan, 1998; Guglielmino et al., 2008, p70). The on-off skyhook simply switches between minimum and maximum damping rates and has the control law of equation 2.10 below. Given the difficulty of engineering smoothly controllable dampers, this is one of the easiest controls to implement in physical experiments (McLellan, 1998; Simon and Ahmadian, 2001). Experiments give mixed results. “The transient test results indicate that the controlled MR dampers appear to yield larger acceleration and displacement peaks, as compared to the stock dampers. The acceleration and displacement RMS values, however, did not show the same increase.” (Simon and Ahmadian, 2001, p374)

In light of the equation for energy absorption (see equation 4.2 of section 4.2), the on-off skyhook will maximize energy dissipation when the damper is capable of it, and will switch off (switch to minimum damping rate) in those conditions where a passive damper would actually be adding energy to the system.

\[ c = \begin{cases} c_{\max} & \text{where } \ddot{y} \dot{s} > 0, \\ c_{\min} & \text{where } \ddot{y} \dot{s} \leq 0. \end{cases} \]

**Equation 2.10**

Savaresi et al. experiment with an interesting variant of this rule which they call the two-state feedback on body acceleration. The control law for this rule responds to car-body acceleration rather than velocity,

\[ c = \begin{cases} c_{\max} & \text{where } \dddot{y} \dot{s} > 0, \\ c_{\min} & \text{where } \dddot{y} \dot{s} \leq 0. \end{cases} \]

(Savaresi et al., 2003, p2266).
Yet another variant on this rule uses stroke displacement, $s$, (sometimes called “relative displacement”, see section 8.1.2). The damper is “on” when $\dot{s} > 0$. Wagner and Liu call this “relative control” (2000, p567).

But the closest approximation that is possible with a semi-active suspension to a skyhook control is to use the same force where possible and to turn “off” the damper when outside the passivity constraint. This requires taking into account the force supplied by the virtual spring. To the best of this author’s knowledge, it is always assumed in the literature that the virtual spring and the real spring have the same rate, but this need not be the case and it is not assumed here. In the literature, the virtual spring is often assumed to be the same as the actual spring, in which case the control over the damper is greatly simplified,

$$ F_d = \begin{cases} c \dot{y} & \text{if } \dot{y} > 0, \\ 0 & \text{if } \dot{y} \leq 0. \end{cases} $$

(Ahmadian et al., 2004, p580; Guglielmino et al., 2008, p70).

Here $c$ represents the damping rate of the virtual damper attached to the sky.

2.8. Road Surfaces

According to Segel (1993, p8), “subsequent to the appearance of the analog computer, the next milestone in advancing of our understanding of the ride and roadholding process was the recognition that the road constitutes a random disturbance function and that ride should be examined as a stochastic process. … The first public recognition that vehicle ride should be considered as a random process seems to have been enunciated by Mitschke” in 1958. Road modelling research has been carried out over quite a long period of time.

Road modelling affects suspension control optimality. While various control methods focus on how to bring a suspension back to equilibrium, the optimality of a suspension is dependent on the statistical properties of the roads that the suspension is likely to encounter. For a particular type of control, whether passive, skyhook, LQR, etc., the optimal parameters depend on the types of roads encountered. Complex road statistical properties are rarely used to analytically derive suspension parameters because of the enormous mathematical
difficulties involved. But road modelling and, of course, testing of suspensions on real and challenging road surfaces is an essential component of practical suspension design in industry.

The methods for collecting road height data vary. Body height is typically measured with an accelerometer (Sayers and Karamihas, 1996, p38). To achieve an “earth-referenced coordinate system”, Schick et al. used a D-GPS (Differential Global Positioning System)” (2006a, p4). Relative height to the road surface can be “measured with a non-contacting sensor, such as a laser or an ultrasonic transducer” (Sayers and Karamihas, 1996, p38). Schick et al. collected data by using a high-speed camera to scan a line projected on the road’s surface (2006a, p4).

The International Roughness Index (IRI), developed in the late 1970s, “has units of slope” (Sayers and Karamihas, 1996, p4). This index shows a remarkably high correlation with psychological tests. A similar index, called the Half-Car Roughness Index, also takes roll into account (Sayers and Karamihas, 1996, p7).

Researchers “have traditionally represented road disturbances as a sum of sine waves whose power spectral density (PSD) matches that of typical road disturbances” (Sims and Stanway, 2003, p94). Roads are often analysed using the measure of Power Spectral Density (PSD). This is an adaptation of techniques used for analysing electrical signals, but it has been widely applied to road surfaces (Sayers and Karamihas, 1996; Elbeheiry and Karnopp, 1996; Kavanagh and Ramanathan, 1982; Burton, 1993, p226; Robson and Dodds, 1970; Yagiz et al., 2000).

Random white noise signals are sometimes used when simulating road surfaces. According to Cole (2001, p347), the “input provided by a typical road surface can be idealised as white noise velocity.” Since the velocity is assumed to contain white noise, a given random white noise input will be integrated to present the road surface. Gaussian white noise is often used for modelling random input signals and is often used for road modelling (Lauwerys et al., 2004, p793; Giorgetti et al., 2006, p41; Karkoub and Zribi, 2006; Majjad, 1997, p523; McLellan, 1998, p22; Vaughan, 2004; Caponetto et al., 2003).
Yagiz claims that a “useful analogy exists between the effects of random signal inputs in the form of integrated white noise and unit step function inputs” (2000, p81). The step function may be useful for some first-order analyses of suspension-optimization problems. In fact MacCluer uses just this problem to nicely compare various mathematical approaches to the optimization of a tracking system (2005, p152).

Chirp signals are also used for experimentation (Lu and DePoyster, 2002; Song et al., 2003). These are signals in which the frequency is slowly changed. Koo et al. use a chirp signal which “sweeps frequency ranges from 0.5 Hz to 10 Hz in 68 seconds. This slow sweeping chirp signal was desired so as to be sure that the low frequency dynamics were not lost” (Koo et al., 2003, p2080). Chirp signals provide a snapshot of the frequency response of the system and can be used to provide an analysis based on frequency (Koo et al., 2003, pp2081-2). The magnitude of the signal can vary with frequency to mimic frequency profiles of actual roads. While a simple frequency analysis is capable of characterizing the response of a linear system, it is insufficient for demonstrating robustness in nonlinear systems.

To simulate road surfaces some researches use simple sine wave tracks, or “corrugated tracks” (Hönlinger and Glauch, 2000, p3). Some have used simple sums of sine waves. Sims and Stanway used a signal comprised of a “sum of sine waves from 0.25 to 15 Hz in steps of 0.25 Hz, each with velocity amplitude of 1 mm/s” (2003, p94). As argued below (in section 3.2), it is important in a nonlinear system to model bumps of a substantial size, approaching the size of speed bumps. Such bumps can be used to augment the results from the study of other road surfaces (Hönlinger and Glauch, 2000, p5), or they can be directly added to the road surface in simulation, as in this author’s experiments.

With two-wheeled vehicles it is often acceptable just to model the wheels encountering the same surface but with a time lag between the wheels (Ashari, 2004, p373). With a four-wheeled, two-track vehicle however there is the problem of the cross-correlation between the tracks (Kavanagh and Ramanathan, 1982, p338).

If computer modelling is used in the optimization process, as in the use of evolutionary algorithms in the experiments performed here, then the road models will affect the optimization. Evolutionary algorithms will adapt their results to the modelled data that is input to them. In this case the input data includes road surfaces. If there are many large
bumps in the training data, then the resulting suspension parameters will be harder than if the training data contains mainly soft bumps (section 5.5.2). For simulation purposes, it may be possible to neglect some factors such “as variation in statistical properties along the length of the road and the effects of hills, banking and holes,” (Kavanagh and Ramanathan, 1982, p337) but a road surface model for the purpose of evolving the parameters of a suspension system should contain a certain proportion of rough road surfaces. These should be mixed with smoother sections to fairly represent normal road conditions and allow a suspension to evolve which is able to handle rough conditions but also be smooth and comfortable under normal operating conditions. A training surface without significantly large bumps will result in a suspension that is unable to handle bumps. On the other hand a training surface with too many bumps will result is a suspension that is too rough over smooth surfaces.

Models of one particular actual road or racing track are obviously realistic, but the use of just that one track for training data may result in a suspension which is only adapted to that track; indeed, random simulated tracks may be better for training data than an actual track’s measurements.

### 2.9. Suspension Design Goals

In 1988 Redfield and Karnopp performed a comprehensive multi-objective analysis of a two-DOF suspension system using just linear models and RMS measures of performance (1988). They looked at both passive and active systems, by the use of what they then called “boundary curves” (1988, p237). Redfield and Karnopp produced an analysis which is equivalent to today’s Pareto optimization. The three performance measures used by them were RMS values of,

1. – Sprung mass acceleration,
2. – Suspension stroke, and
3. – Tyre contact force variation,

(1988, p238).

These measures correspond roughly to 1) isolation or comfort, 2) suspension tracking and 3) road holding or wheel hop. Their analysis showed that the various objectives competed with one another; comfort could be increased at the expense of road holding for example. They
also argued that active suspensions could theoretically provide better performance than passive ones.

Car companies now invest large amounts of research into the “feel” of a car (Harrer et al., 2006; Yamakado and Abe, 2006; Schick et al., 2006b; Toda and Kageyama, 2006). BMW used comprehensive questionnaires of their test drivers to determine which factors seemed to most contribute to the desired feel of their vehicles. These tests were specifically aimed at discerning what the drivers considered to be the feeling of the BMW brand (Harrer et al., 2006). This underscores the fact that at least some of the car performance indicators are of a complex psychological nature, and these factors may not directly correspond to simple objective measures.

Interestingly, passenger cars can provide a greater challenge for designers in terms of performance criteria than commercial vehicles, military vehicles and racing cars, since passenger cars need to fulfil an extremely wide range of performance criteria. On the other hand a Formula One vehicle has one main objective: speed (Bastow et al., 2004), limited, of course, by safety considerations. The following is a brief summary of some of the performance goals of suspension systems with reference to the literature. The performance goals of comfort and vertical travel are central to this thesis and these will be analysed in greater depth in section 3.

### 2.9.1. Comfort

This goal represents the degree of comfort felt by a passenger. It is the extent to which the suspension “softens” the ride of the vehicle. This is a subjective matter and there is no single objective standard by which to measure comfort.

Ride smoothness is associated with “isolation”, meaning vibration isolation. For a vehicle suspension, vibration isolation is the degree to which the chassis is isolated from the road roughness (Yedavalli and Liu, 1994; McLellan, 1998; Karnopp, 1995). According to Barak, “the Human Body is sensitive to certain frequencies such as 0.5-0.7 Hz dizziness and seasickness, 5-7 Hz body resonance and 18-20 Hz head and neck” (1991, p1699).
There are different ways to interpret isolation and the differences are quite subtle. Vibration isolation could be used to:

- reduce the risk of damage to inanimate cargo,
- reduce the bruising of fruits and other foodstuff,
- avoid damage to live cargo,
- reduce health risks such as driver back or spinal damage as a result of long periods of vehicle travel,
- create a sensation of comfort and luxury,
- minimize car sickness,
- create a sense of “fun” from the feel of the suspension,
- create a particular “feel” of responsiveness that a company is aiming for.

Separate goals may correlate with different physical measures; for example damage to inanimate cargo might correlate better to RMS acceleration while passenger comfort may correlate best with RMS jerk. However, in order to retain the distinction between comfort and other goals of vibration isolation, the term “comfort” will be reserved to refer only to the sense of comfort for human passengers.

Redfield and Karnopp used RMS vertical chassis acceleration as the main measure of vibration isolation (1988). Smith et al. analysed a psychological study correlating perceived ride comfort with various metrics and standards of various standards bodies, including the International Standards Organization (ISO) standard, Urban Tracked Air Cushion Vehicle (UTACV) and Absorbed Power metrics for measuring vehicle ride comfort (1978). This research, “supported in part by the United States Department of Transportation” (1978, p39) incorporated an extensive psychological study using two vehicles, 78 different passengers and 18 different roadway sections (1978, p35). Psychological measures and physical measures were calculated and compared. The study looked at both floor board accelerations and seat accelerations, and it concluded that unweighted RMS acceleration in the vertical direction was as good a predictor of ride comfort as any of the other measures used at the time, some of them quite sophisticated.

They also found that there was little difference in using the frequency range 0 to 40 Hz, compared to using the range 0 to 100 Hz, “indicating little additional information relative to the ride quality exists within the 40-100 Hz band” (Smith et al., 1978, p37). McElellan uses
the range 1 Hz to 15 Hz (McLellan, 1998, p36). The frequency range 0.1 Hz to 50 Hz would seem to more than cover the range of frequencies needed to be controlled by an active or semi-active system. This is in the sub-audio frequency range and it is easily dealt with by electronic filters, modern digital processors and analog-to-digital and digital-to-analog converters.

With short-stroke vehicles, the problem of combining comfort with tracking is much more difficult than with long-stroke vehicles. And yet passenger cars, with the greatest demand for comfort, tend to have shorter strokes than say trucks, tanks, HMMWVs (also called “Humvee” or “Hummer”), and other large vehicles.

2.9.2. Vertical Travel

As discussed in section 2.9, RMS suspension stroke has been used as a measure of vertical travel (Redfield and Karnopp, 1988). The smaller that this value is the better the suspension stays near the equilibrium point of the suspension and the better the suspension follows the road surface.

However, the objective with vertical travel is not to minimize the travel, but to constrain the travel within the limits of travel: the rattlespace. This is the objective of stopping the suspension from hitting up hard against the suspension travel limits: either from hitting against the wheel well, “bottoming”, or extending and jerking against the full length of the suspension travel. This is often misinterpreted as the degree of travel of the wheel. The way that vertical travel as a performance measure is interpreted needs a deeper investigation if it is to be addressed in a consistent way. This discussion is continued in a later section (section 3.2).
2.9.3. Cumulative Absorbed Power

Cumulative Absorbed Power (CAP) is an interesting suspension performance measure because it seems to combine a number of performance goals into one. “TARDEC has conducted extensive research on terrain roughness values and how they relate to the human drivable speed for a given road. The maximum drivable speed is the speed that the Cumulative Absorbed Power … reaches 6 Watts.” (Donahue, 2001, p71) Roughly speaking, drivers will slow down when their bodies absorb 6 Watts. Donahue graphs maximum speed versus terrain roughness as a measure of performance. The experiment was performed on a HMMWV. This measure is designed for military vehicles travelling at high speed over rough terrain and so the energy absorption of passengers is a limiting factor, and comfort is a secondary matter. A highway vehicle on the other hand has comfort as a high priority and, with a much shorter stroke, hitting against the suspension travel limits is much more important.

2.9.4. Road Holding

Loss of traction occurs during periods of low tyre contact force. A suspension with lower variation in tyre contact force will, in general, have less periods of low contact force (Hönlinger and Glauch, 2000).

Some studies use tyre normal force as an indicator of road holding (Caponetto et al., 2003, p788; Cole, 2001; Uys et al., 2006, p48). Extensive tyre and road modelling is necessary if this factor is to be modelled accurately (Lee et al., 2006; Lot and Massaro, 2006; Schick et al., 2006a). Tyre deflection can also be used as an indicator of road holding (Youn et al., 2006).

2.9.5. Driving “Feel”

Empirical studies into driving feel indicate that responsiveness to steering is an important indicator of the feel of responsiveness (Uys et al., 2006; Harrer et al., 2006). Yawing under
cornering, and side-to-side accelerations are also important for the feel of steering responsiveness (Harrer et al., 2006, p11; Uys et al., 2006, p44).

2.9.6. Safety

Safety affects the full range of parameters that can be used in a suspension design.

In 1965 Ralph Nader published *Unsafe at any Speed* (1972). Nader criticized the 1960-63 General Motors’ Chevrolet Corvair for the tendency of the suspension to “tuck under”, contributing to vehicle rollover. The ensuing public outcry influenced the US government to legislate standards for car suspensions. In particular, the intent has been to legislate against parameters affecting rollover. Similar legislation now exists throughout the world. The Australian government has legislated for minimum spring rates (DOTARS, 2004). The main design parameter affecting energy storage and affecting rollover onset is the spring constant (Lee et al., 2006).

Wheel hop occurs when the wheel actually leaves the road (Elbeheiry and Karnopp, 1996; Lu and DePoyster, 2002; Lee et al., 2006). The most catastrophic cases lead to car rollover, usually under cornering. Extensive modelling by Lee et al. showed a runaway resonance pattern during rollover. Rollover onset was shown to be sensitive to the stiffness of dampers, springs and bump stops (the bump stop is the buffer, usually rubber, at the ends of the rattlespace travel) (Lee et al., 2006, p8).

The Swedish “elk test” or “moose test” created some controversy “in 1997 when journalists from the Swedish motor magazine Teknikens Värld overturned the new Mercedes-Benz A-Class in the moose test, while a Trabant – a much older and widely mocked car from the former German Democratic Republic – managed it perfectly” (Wikipedia, 2007b). This test uses a swift side-to-side lane change to simulate the avoidance of a road obstacle. SAAB perform the elk test using a physical “model” elk (Wikipedia, 2007b). The US National Highway Traffic Safety Administration (NHTSA) used a rollover test in which vehicles were tilted to see at which angles they tipped and slid (Uys et al., 2006, p45). They also use another test somewhat similar to the “elk test”, called the “dynamic maneuvering test” (NHTSA, 2007).
Larger vehicles and all-terrain vehicles can be more prone to rollover; according to the “car accidents” website (car-accidents.com, 2011):

“There are about 280,000 rollover accidents in the US each year, resulting in about 10,000 fatal accidents. SUVs with a high center of gravity, are more likely to be involved in rollover accidents than typical sedans. About 35 percent of fatal crashes in SUVs resulted from a rollover crash, compared to 16 percent of fatal passenger car accidents resulting from rollovers. This percentage indicates the fact that SUVs are much more likely to rollover and result in a fatality.

Rollover accidents relate to a vehicle’s stability in turns. That stability is determined by the relationship between the center of gravity and distance between the left and right wheels (called track width). In light trucks, Jeeps, Sport Utility Vehicles, high centers of gravity and narrow track width often makes the vehicles unstable in sudden turns or changes of direction. This increases the likelihood that the vehicle will tip over if it skids sideways.”

![Figure 2.11 Rollover (car-accidents.com, 2011)](image)

“In roll, the body rolls out-of-phase with the wheel/tire.” (Brinker et al., 2006) The Metaldyne Suspension System (MSS) (2006) slightly alter the geometry of the control arms to minimize this effect. Improvements to suspension geometry have been applied over almost a century of research and development (Bastow et al., 2004). Semi-active damping has the advantage that such improvements can be retained, since the only physical change
required to the suspension geometry is the replacement of the telescopic damper with a controllable damper.

Catastrophic failure due to modes of resonance of the chassis are rare, but they can occur on large vehicles, as perhaps in the example in figure 2.12. This incident may have occurred during cornering, but the placement of load at each end of the trailer, lowering the frequency of the torsional vibration mode, surely had an impact.

![Figure 2.12 Possible Failure due to Torsional Resonance (UMTRI, 2000, p6)](image)

While electronic control may mitigate resonance and rollover, there is still the problem of a semi-active damper going offline: either the control signal could fail, or the damper could physically fail. MR dampers are at their softest when no current is supplied (Stembridge et al., 2006, p3). MR dampers can be used in such a way that their damping rate when the control signal fails matches that of legal standards, or of original equipment manufacturers. “In the absence of an electric input current to the damper or in the case of a malfunction in the control hardware, the new MRF damper provides the same damping force that is produced by an original equipment manufacturer (OEM) passive viscous damper” (Gordaninejad and Kelso, 2000, p395). Thus “they become passive dampers when the control hardware malfunctions” (Zribi and Karkoub, 2004, p511). Thus MR dampers can be considered as ‘fail-safe’ devices if they still function with an acceptable damping rate when they go offline. MR dampers designed for mountain bicycles have also been designed as fail safe (Breese and Gordaninejad, 2003; Ericksen and Gordaninejad, 2003).
Fail safe comes at the expense of potential comfort however since one of the great potential advantages of a semi-active system is a much smoother ride using a smaller damping rate under smooth road conditions. In the design of Gordaninejad and Kelso (2000, p396), “the MRF damper at passive-off state is designed so that it is slightly softer than the OEM damper”. Another possibility is to have the suspension electronics linked to the engine control and to place the vehicle in limp home mode in the event of a suspension control failure.

Hydraulic suspension systems are potentially unsafe if they do not power down properly. Suddenly switching off control power to a hydraulic system can result in an unexpected release of pressure at some later stage. At power down all forces should be brought to zero smoothly (Donahue, 2001, p19). A semi-active or active suspension should have a similar slow power down, otherwise a sudden jerk could result.

The performance goals of comfort and of vertical suspension travel especially for short-stroke vehicles forms the major focus of this thesis. These are investigated in more detail in section 3.

**2.10. Electronically Controlled Suspensions**

This section looks briefly at suspension control systems with particular emphasis on vehicle suspensions, as opposed to structural suspensions. The term “intelligent” control is sometimes used to refer to control using computers or microprocessors. In practice, intelligent systems are almost always electronically controlled, and the term “electronic controlled” is used here. Electronic suspension systems vary the suspension characteristics on a moment-by-moment basis. These can be divided into active and semi-active suspensions (Stembridge et al., 2006; Goncalves and Ahmadian, 2002; Hyvärinen, 2004; Li et al., 2004; Song et al., 2003). This section gives the definition of active and semi-active suspensions and covers some of the wide range of current implementations. Further details from the literature of suspension control algorithms are discussed in later sections.

In the 1970s researchers were beginning to investigate active suspension. Crosby and Karnopp defined active suspension,
In idealized form, an active suspension element is a controllable force generator powered from an external energy source. Such devices can be programmed to produce forces which are functions of any desired system variable. By sensing system variables, such as, absolute or relative accelerations, velocities, or displacements and appropriately combining them, a command signal is developed for the force generator. With such devices, isolation system performance is limited only by the amount of external power, the designer is willing to expend and system complexity.” (1973, p119)

An active suspension will have a powered actuator that produces the force required by the control, as represented in the left in figure 2.13. Active suspensions can potentially produce any force desired and so are much more flexible than traditional passive controls.

While there was initially some deviation in terminology, definitions for active, semi-active and passive suspensions have now become quite standardised (Karnopp, 1995; Chang et al., 1999, p3276; Ahmadian and Simon, 2002). “A vibration-control system, either as an isolator or an absorber, is said to be active, passive, or semi-active depending on the amount of external power required for the system to perform its function” (Jalili, 2002, p593). A semi-active damper will “draw small amounts of energy ... to adjust the damping level and reduce the amount of energy that is transmitted from the source of vibration energy to the suspended body” (Ahmadian and Simon, 2002, p123) as represented on the right in figure 2.13. The energy used in a semi-active system is only small and could be used only to “operate a valve” in the controllable damper (Ahmadian and Simon, 2002, p123) or to power a small electromagnet to vary the rheological properties of a fluid as in a magneto-rheological damper (Sims and Stanway, 2003).
A passive system has no external power applied, such as the conventional systems comprised of springs and dampers. Note that energy is stored in the springs in a conventional suspension so a passive system can apply stored energy, but the energy is derived from the suspension itself and not an outside power source. Furthermore, a passive system need not be simply the spring and damper system, nor need a control law be mediated by electronic control; a purely mechanical system for semi-active dampers has been proposed by Ivers and Miller (1991, pp337-8). Such systems suffer from the fact that it is cumbersome to base a mechanical control on chassis velocity. Their mechanism was tuned to a very low frequency, 0.1 Hz (Ahmadian et al., 2004, pp337-8).

Hydraulic suspension systems have been in use on production vehicles for some time. Perhaps the simplest such systems use hydraulics to transfer force between the suspensions on different wheels and use only rubber springs and rubber valves for damping, as in “hydrolastic” suspensions used in Minis since 1964 (Longhurst, 2007). Hydrolastic suspensions evolved into hydro-pneumatic suspensions, such as the Moulton Hydrogas suspension which uses nitrogen gas for springing (Longhurst, 2007; Rideout, 1998). The compressibility of gas makes this a more pliant suspension. Although some of these derive some power from the pumps, the pumps simply maintain a given pressure; they cannot properly be called active suspensions.

A fully active system allows power to be applied in any direction and with whatever force the designer desires on a moment-to-moment basis. A semi-active system however requires only a minimum amount of power; it “requires no more power than a headlight” (Ivers and Miller, 1991, p327).

Systems that change their characteristics over a long period of time are termed “adaptive” rather than active or semi-active (Ivers and Miller, 1991, p327). “The distinction between an adaptive and semi-active system can be made by the bandwidth of the system. If the natural frequencies of the suspension system are below the natural frequencies of the vehicle’s natural frequencies, an adaptive system is quoted. The load-levelling system offered by many car manufacturers can be considered to belong to this category.” (Hyvärinen, 2004, pp30-1). Adaptive suspensions can change driving characteristics depending on a driver’s choice of “feel”; they can change characteristics as components age; and they can change characteristics depending on type of terrain encountered, or depending on the load placed on
the vehicle. This requires the electronics to measure suspension properties over time through “on-line system identification” (Song et al., 2003, p219). Adaptive systems are not the focus of this thesis, but adaptive elements that respond to terrain-type variation, component aging and vehicle load can be included in both active and semi-active suspensions.

There has been some research into “active-passive tandem” suspensions. This refers to systems that do not use the road height or the unsuspended mass height as inputs (Hyvärinen, 2004, pp30-1). The term “limited active” control has been used by Elbeheiry and Karnopp to refer to a system that does not measure tyre deflection (1996, p556).

Shi et al. investigated the use of passive components alongside active components in order to save power (1996), and even regenerative systems have been investigated (Okada and Harada, 1996). In a regenerative suspension energy is stored for later use. While passive suspensions store energy in springs, regenerative suspensions store energy that can be returned using electronic control. Okada and Harada’s experiment demonstrated the principle of regeneration but the system was very simple. In one system “An electro-dynamic actuator is used” to regenerate electrical energy when the suspension is moving quickly (1996, p4715). Their work focused on energy usages and the switching used in their control produced a clearly discontinuous response (1996, p4720). A system that regenerates hydraulic pressure has also been proposed (see US patent 6,394,238 B1) (Rogala, 2002).

Jalili uses the term “hybrid” system to refer to systems that switch between passive and active modes. “With the aim of lowering the control effort, relatively small vibrations are reduced in active mode, while passive mode is used for large oscillations” (2002, p599). Such systems are proposed for large-structure seismic stabilization rather than vehicle suspension, where a seismic event such as an earthquake occurs infrequently, hopefully never, and the control switches modes and stays in the new mode for a relatively long period of time. This kind of control allows the system to effectively change its stiffness depending on the urgency of prevailing conditions.

The use of controlled suspension for truck seats “between the vehicle floor and driver’s seating system”, is an attractive alternative to the implementation of electronic control at the wheels, since it requires much less force (Wagner and Liu, 2000, p564). Alternatively, the truck cabin can be independently suspended. Deprez et al. investigated Simulink simulations

Hydraulic systems that use accumulators to store fluids under pressure can have regenerative capabilities (Smith et al., 2006). Whether such a system can be called active depends on the response time and whether or not the force is limited in some way. “Gas springing has been used for many years... The gas is lighter than a metal spring but requires containment.” (Dixon, 2008, p19). This is virtually equivalent to a spring in parallel with the metal spring. But if this is controlled with a fast response time, and if the supplied force is not heavily limited, such systems are very close to the completely general response of a truly active suspension (Williams and Best, 1994).

Vehicle suspensions can also use preview to improve ride. Preview employs information about the approaching road surface to improve the ride. Preview could acquire its data about the up-coming road surface by using a pre-stored database of information, or by using an on-board sensor. Donahue implemented such a system using sensors placed on the front of the chassis (2001, pp19-25). Such systems are still in the experimental stage and are not the subject of this thesis.

2.10.1. Implementations of Controlled Suspension

This section takes a brief overview of the proposed and actual uses of active and semi-active suspensions and some technologies for implementing them. There has been a steady development of such technology over the last few decades. In 1994, Williams and Best claimed that a “high bandwidth active suspension”, capable of dealing with frequencies between 1 to 12 Hz required “the use of aerospace technology” (1994, p338). Today, suspension systems deal with such frequencies routinely and relatively cheaply. In 1993 Burton wrote that a “major obstacle … is linked to the (un)availability of cheap and reliable sensors and actuators which, in addition, have acceptable levels of performance and power consumption” (1993, p225). In fact electronically-controlled suspensions are now used on production vehicles and their price has plummeted, as discussed below.
Active and semi-active suspensions have been studied for application to a wide range of vehicles other than passenger cars: motorcycles (Ivers and Miller, 1991, p330), military tanks (Ivers and Miller, 1991, p330), HMMWVs (Donahue, 2001; Gordaninejad and Kelso, 2000), heavy trucks (Simon and Ahmadian, 2001), articulated vehicles (Palkovics and Fries, 2001), buses (Cole, 2001), trains (Atray and Roschke, 2003), and aircraft landing gear (Krüger, 2002).

Williams et al. experimented with an oleo-pneumatic (hydro-pneumatic) active suspension (1996, p45). “Fluid is supplied to the actuator from a high pressure source (via a control valve).” Most of the springing of the system was due to the gas reservoir. It was found that the nonlinearity of the system needed to be addressed to achieve better control.

There have been a range of dampers that can be adjusted from the driving seat, including a very early dry friction damper, the Hartford Telecontrol damper, operated mechanically via a Bowden cable from the dashboard, shown in figure 2.14. “A common form of adjustable damper has a rotary valve with several positions each having different orifice size. Some form of rotational position control, e.g. a stepper motor, is fixed to the top, controlling the piston valve through a shaft in the hollow rod” (Dixon, 2008, p20). More recently, magneto-rheological dampers have been used as adjustable dampers. Even when employing magneto-rheological dampers the system is not properly semi-active unless it is being adjusted electronically on a moment-to-moment basis.

![Figure 2.14 Hartford Telecontrol Damper (Dixon, 2008, p6)](image-url)
There are a number of systems that implement controlled suspension by passing a fluid through a valve with a controlled aperture, often using just on-off control activated electrically by a solenoid or motor. This is often augmented by gas reservoirs (accumulators) either just to absorb the movement of the suspension or to add to the springing of the suspension. Els et al. experimented with numerical models of a semi-active hydro-pneumatic system operated by solenoid values, with switching times of at most 100 ms (2005). Lawerys et al. experimented with and modelled an active hydraulic system using a controlled solenoid valve, with a switching delay of 6 ms (2004). Their system was powered by a hydraulic pump consuming approximately 500 watts. Wagner and Liu performed numerical experiments using a hydraulic system controlled by a solenoid valve (2000). They also investigated an “active electro-mechanical hydraulic actuator”. They claimed a 22% reduction in RMS acceleration using the semi-active system although the testing was restricted to a single bump. Donahue modelled and implemented an active hydraulic system which moved a valve to “direct high pressure fluid flow” to either side of a piston in parallel with a bypass valve to help regulate the pressure differential (2001, p8).

Solenoid valve systems have inherent force discontinuities and attendant “spikes” in jerk (see section 4.3). Ivers and Miller attributed this problem to pressure build-up in the tyre and they provide an interesting analysis of the problem (1991, p337):

“In many types of primary suspension applications in which on/off semi-active control is used, a noise problem can exist … The noise is described as ‘thumping’ or ‘banging,’ and is correlated with the derivative of acceleration of the vehicle body. It usually occurs only when the valve opens. While the valve is closed in order to resist the vertical motion of the vehicle body, the tire can be compressed (beyond its equilibrium state) by the reaction force. If the valve is then opened, the energy stored in the tire is suddenly released. Since there is very little damping in the system while the valve is open, the axle mass is free to resonate. This is perceived in the vehicle as an undesirable noise.

Methods for solving this problem are fairly simple. One is to increase the damping in the off state, however, this degrades performance. A better solution is to simply not allow the valve to open as long as the tire is compressed. This modification of delaying closed-to-open switches until the relative velocity reaches zero degrades performance
very little because of the small percentage of time involved in the delay… There is an analogous algorithm for the continuously variable semi-active system …”

A number of researchers have experimented with continuously variable hydraulic dampers. Sam and Hudha model a hydraulic system using as a damper for a semi-active system (2006). Gao et al. modelled and numerically tested a damper controlled by a “proportional servovalve” (2006). The control force depends in complex ways on the pressure in accumulators. They demonstrated improved performance using modified skyhook controls. Teixeira et al. experimented with a damper containing a valve controlled by a piezo-electric actuator. The damping force versus velocity is linear in their system (2006, p349). They also produced a physical prototype.

Hydraulically powered systems for implementing active suspensions have disadvantages. “The actuators require hydraulic lines and a hydraulic pump which consumes power from the engine. Other disadvantages include increased weight, maintenance, cost, and reduced reliability,” (Ivers and Miller, 1991, pp328-9) and they are not fail safe.

**Figure 2.15** Comparisons of Suspension Control (Bose Website)

**Figure 2.16** Bose Linear-Motor Strut (Bose Website)
Bose have released an active suspension that uses a linear electromagnetic motor (refer to figure 2.15 and figure 2.16). The motors are driven by power amplifiers based on the highly efficient switch-mode audio power amplifiers. The process is regenerative and Bose claim that the entire system “requires less than a third of the power of a typical vehicle’s air conditioner system” (Bose, 2007). This is an active system as demonstrated when a Bose suspension was programmed to “jump” over a line painted on a test track (Wired, 2005). The system comes in a number of forms but the simplest is designed to have a two-point connection and to replace conventional struts.

Studies into the use of semi-active systems for suspension control, then called “active damping”, date back to the early 1970s with the work of Karnopp and Crosby (1973), and the concept was patented by them in 1974 (1974). “The first documented vehicle installation … of a semi-active system was in 1981 on a Yamaha YZ-250 dirt bike. This consisted of an on/off active damper on the rear wheel and a controller. The controllable valve was located external to the damper” (Ivers and Miller, 1991, p330). “The first vehicle demonstration with experimental evidence to show the benefits of a semi-active suspension was on a military tank … The M551 tank, a tracked vehicle with ten wheels, was selected as a test bed by the U.S. army Tank-Automotive Command (TACOM) for evaluation of the on/off semi-active suspension concept” (Ivers and Miller, 1991, p330).

Krüger modelled the use of semi-active suspensions for aircraft landing gear. Landing gear needs to deal with a “high stroke velocity at landing impact” but it should also provide a “comfortable ground ride” (2002, p497).

There are a number of other ways of implementing a controllable damper besides hydraulic or electrical methods:

“Semiactive dampers can be adjusted by mechanical means or using the rheological properties of the fluid that is used in the damper. The former uses mechanical valves driven by a solenoid or stepper motor to control damper force in a hydraulic damper. The latter category uses the rheological effect of controllable fluids, such as magnetorheological or electrorheological fluids, to provide adjustable damping forces. Although mechanical and rheological control dampers have been researched and developed extensively, the rheological controllable dampers have received much more
attention in the past few years, mainly due to great advances in magnetorheological fluids.” (Ahmadian and Vahdati, 2006, p145)

Fluids that can be controlled in this fashion also go by the generic name of “smart fluids” (Sims and Stanway, 2003, p77). These are fluids that change viscosity in a controllable way. Electro-rheological (ER) fluids change viscosity on application of an electric field, while magneto-rheological (MR) fluids change their viscosity on application of a magnetic field. MR dampers are discussed later in this section.

Friction provides another means of controlling dampers. This “requires only the direct contact of two parts moving relative to each other and it can be incorporated into harsh environments and vacuum environments where the use of elastomeric damping treatments and fluid filled dampers is limited” (Unsal et al., 2004, p60). Because the “static coefficient of friction is noticeably greater than the kinetic coefficient” friction dampers are susceptible to sticking at zero relative velocity: the “stick-slip” phenomenon (Unsal et al., 2004, p60). They are obviously highly nonlinear in this region. Unsal, Niezrecki et al. have used piezo-electric actuators to supply a normal force for the friction damper. They concluded that MR dampers of similar size were capable of providing more force at lower voltage. Guglielmino et al. devote an entire chapter to friction dampers, and they discuss a wide range of friction damper models (2008).

Ways of implementing a semi-active spring, rather than a damper, have been researched (Hyvärinen, 2004, p32; Jalili, 2002, p598). Semi-active springs can be used to maintain a narrow range of spring constant as a spring ages. As noted by Jalili, adaptive springs, that change their properties over longer periods of time, to counteract the effects of aging, or to vary the systems response, are quite feasible.

The most widely used controlled dampers for use in semi-active suspensions are dampers using magneto-rheological (MR) fluids. These are referred to as MR dampers or MRF dampers (Magneto-Rheological Fluid dampers) (Gordaninejad and Kelso, 2000, p395).

Koo et al. studied the response times of the commercially available MR damper used in the “Motion Master Ride Management System” (Lord, 2006; Koo et al., 2004). They “define the response time as the time required to transition from the initial state to 63.2% of the final
state, or one time constant.” Lord claim a response time of 10 ms. Experiment shows that response time does not vary with current (between 0.5 to 2 A) but it does vary exponentially with damper extension velocity. Below 15 mm/sec (0.6 in/sec) the response time is larger than 10 ms.

Koo et al. used an accumulator to compensate for the jerk (“jounce”) of the damper stroke; this was implemented through a nitrogen gas chamber sealed behind a floating piston. Reichart claims that the “accumulator serves two purposes. The first is to provide a volume for the MR fluid to occupy when the shaft is inserted into the damper cylinder. The second is to provide a pressure offset so that the low pressure side of the MR valve is not reduced enough to cause cavitation of the MR fluid” (1997, p18).

McLellan used MR dampers supplied by the Lord Corporation in comparative experiments employing a number of control algorithms (1998, p26). The dampers employed two electromagnetic coils. “One coil is used on the compression stroke, and the other is used for the extension stroke” (1998, p26). “The accumulator, made of closed-cell foam, was necessary to allow for changes in the volume of the fluid as it was heated, and also to allow for the added volume of the piston rod as it enters the damper” (1998, p27). The system “was tested on a Volvo VN series class 8 truck ... Although the cab has three axles, the magneto-rheological semiactive system was installed on only two of the axles” (1998, p30).

Semi-active systems have been increasingly used in vehicles over the last few decades. According to Goncalves and Ahmadian, “Semi-active suspensions have been shown to offer valuable benefits for vehicle primary suspensions … Available in the 2002 Cadillac Seville STS, the MagneRide semi-active suspension uses a magneto-rheological damper made by Delphi to vary the damping according to the driving conditions. Beyond improvements in ride and handling, Cadillac also claims that the system is very effective at slowing weight transfer, which promotes stability” (Goncalves and Ahmadian, 2002, p1). The MagneRide system employs a number of sensors: “a relative position sensor between each control arm and the body as well as a lateral accelerometer and a steering-wheel angle sensor” (Gehm, 2001, p32). To date semi-active suspensions have mainly been introduced into production vehicles at the “top end” of the market.
The applications of MR dampers do not stop at vehicle suspension control. Research has also been carried out on large scale control systems for seismic stabilization, discussed below, and smaller scale systems for “varying the stiffness of exercise equipment, and for reducing vibrations in truck seats” (Atray and Roschke, 2003, p223; Jolly et al., 1998) as well as for use in clutches (Kavlicoglu et al., 2002). They have been investigated for the vibration damping of helicopter rotors where the extreme temperatures during operation can cause the degradation of the performance of a passive damper (Gandhi et al., 2001).

MR dampers have been used in protecting buildings against seismic shocks. For this purpose they currently have much better properties than electro-rheological (ER) dampers (Soong and Spencer, 2002, p257). Semi-active hydraulic systems have also been developed and used for seismic control in the Kajima Shizuoka Building, Shizuoka Japan (Soong and Spencer, 2002). In 1998, Dyke et al. wrote that, “Extensive research has been done on active structural control … and these systems have been installed in over twenty commercial buildings and more than ten bridges (during construction)”. MR dampers are considered appropriate for building stabilization. They are,
“relatively inexpensive to manufacture because the fluid properties are not sensitive to contaminants. Other attractive features include their small power requirements, reliability, and stability. Requiring only 20–50 watts of power, these devices can operate with a battery, eliminating the need for a large power source or generator. Because the device forces are adjusted by varying the strength of the magnetic field, no mechanical valves are required, making a highly reliable device. Additionally, the fluid itself responds in milliseconds, which allows for the development of devices with a high bandwidth.” (Jansen and Dyke, 2000, pp1-2)

The use of batteries provides a fail-safe since mains-supply power outage can accompany seismic disruptions. Jung, Park et al. have modelled the application of various controls for suspension bridges and propose a hybrid system, using passive as well as controlled dampers.

Seismic applications of suspension control have very different requirements from vehicle suspension. Safety during an earthquake is of extreme importance, and comfort is a minor consideration. The control algorithms appropriate for a commercial building or bridge may be very different from those used in vehicles because the suspension performance goals are different. Electronically-controlled seismic dampers are needed mainly during high winds or earthquakes, and they may be unpowered for much of the time. This may require different physical properties of the MR fluid since the system “cannot rely on dynamic stroking of the damper to provide any mixing action to maintain a well dispersed fluid” (Jolly et al., 1998).

There are a large number of commercially available MR fluids and devices for different applications. Jolly, Bender et al. discuss the physical properties of various kinds of MR fluids and their commercial applications (1998).

In vehicle and seismic suspensions the vehicle or structure is isolated from external vibration. Conversely, suspensions can protect a machine mounting or the floor of a building from the vibration of a machine, a heavy electric generator for example. In another configuration in multi-storey buildings active control systems will move a heavy substructure which acts as an “absorber” so that the building superstructure remains stable (Jalili, 2002, p594).

Thus there are different uses for suspension systems. The performance measures and control algorithms that work best for one application will not necessarily be best for another. For large structural or seismic suspensions, it is energy absorption which is the major concern,
rather than tracking. Even comfort is secondary. This thesis focuses, however, on small-stroke suspensions in which comfort and tracking are important.

2.10.2. Sensors

Electronic suspension control systems require input from the environment. Sensors placed on the car body, on the wheel housing and sometimes in the tyre itself, provide information from the environment. Accelerometers are used most often: one on the car body and another on the unsprung mass (McLellan, 1998, p32). Accelerometers are continually improving in accuracy and decreasing in size. Even as early as 1991, Ivers and Miller claimed that “the cost of an accelerometer has dropped from hundreds of dollars to tens of dollars – as much as a factor of 20. … More than any other single item, the accelerometer cost had prevented semi-active suspension from appearing on production vehicles” (1991, p335).

Accelerometers generally use electro-magnetic effects of various kinds to pick up the movement of a small suspended mass. Until recently, one of the the most common designs for accelerometers has a seismic mass that “rests on a number of piezoelectric discs”. The mass and discs are pre-loaded by a spring and the whole assembly is sealed inside a housing” (Westbrook and Turner, 1994, p157). The piezo-electric effect generates a current which, in a certain frequency range, is proportional to acceleration (Westbrook and Turner, 1994, p159).

The most recent developments in accelerometers use Micro-Electro-Mechanical Systems (MEMS), and they are for all intents and purposes solid state devices (Yoshihiko et al., 2006). Examples of breakout boards containing such devices are shown in figure 2.18. These use piezo-electric effects arising from the very small movement of elements on an etched silicon surface. In recent years the ease of use of MEMS accelerometers, as well as their increasing accuracy and decreasing cost, has made them very attractive for use in production vehicles.

“What makes MEMS important is that it utilizes the economy of batch processing, together with miniaturization and integration of on-chip electronic intelligence … Simply stated, MEMS makes high-performance sensors available for automotive applications, at the same cost as the traditional types of limited-function sensors they
replace. In other words, to provide performance equal to today’s MEMS sensors, but without the benefits of MEMS technology, sensors would have to be several times more expensive if they were still made by traditional electromechanical/discrete electronics approaches.” (Fleming, 2001, p296)

Such accelerometers can include signal conditioning on the chip itself, and can often be bought on circuit boards that simplify the attaching of leads and themselves may contain more signal-conditioning and protection circuitry.

![MEMS Accelerometer](image)

**Figure 2.18 MEMS Accelerometers (Robotshop, 2011)**

Jerk can be calculated by differentiating the acceleration signal, although there has been some investigation into the engineering of specific sensors for measuring jerk. Yamakado and Abe designed a sensor specifically for the measurement of jerk (2006). They were able to develop a dedicated jerk sensor using electromagnetic induction. They found in their studies that drivers tend to select the timing of braking turning “when either longitudinal or lateral motion of a vehicle is momentarily in a stationary state (i.e., jerk zero point) and start to steer or accelerate at this timing” (2006).

Integration on accelerometer output can be performed to estimate velocity and absolute vertical displacement (Ivers and Miller, 1991, p332; McLellan, 1998, p32). Integration can be performed digitally or can be performed by analog components and converted to a digital signal using a D/A (Digital-to-Analog) converter (McLellan, 1998, p32). Low pass filtering is also applied to the accelerometer input to remove noise (Donahue, 2001, p56).
Yi and Song experimented with the errors in measurements and output force in a semi-active suspension using accelerometers alone to measure inputs (Yi and Song, 1999). They found that “all states of a semi-active suspension can be estimated only with acceleration measurements” (1999, p129). Yi and Song showed conclusively that signal processing, to produce an estimate of system states (such as sprung mass velocity) produced improved results. Such estimators are formally known as “observers” (Dorf and Bishop, 2005, p660).

Different types of accelerometers have different frequency responses and accelerometer output may need to be filtered. Furthermore there is the need to remove low frequencies as the vehicle travels up and down hills and long ramps. After integration, Ivers and Miller used a high-pass cut-off frequency of 0.5 Hz (1991, p331).

Sometimes noise can be introduced by the measurement devices themselves. One approach to the problem of noisy observers is to optimize despite the lack of “full state measurement” (Fathy et al., 2003b). In very broad terms, this technique finds the controller input that most likely optimizes the suspension given incomplete information.

Sometimes sensors that directly measure the distance between the chassis and the unsprung mass are used (Williams and Best, 1994, p338; Fathy et al., 2003b). These are attractive because double integration to measure distance introduces potential errors due to drift. Displacement sensors have been used in the MagneRide system, as well as a steering-wheel angle sensor, see section 2.10.1 (Gehm, 2001). Systems for measuring flow inside hydraulic dampers, or some functional equivalent, have also been proposed as a way of measuring damper displacement (Ivers and Miller, 1991, p329).

In researching look-ahead suspension systems, Donahue experimented with two types of preview sensors: a radar sensor and an infrared LED sensor (2001). According to Donahue, “The optical sensor detects small bumps better that the radar... However, since the radar filters small width, low frequency disturbances it may better represent the actual system disturbance” (2001, pp52-3). The generation of road-height data was found to require trigonometric calculation from sensor data plus a “forgetting factor” (2001, p23) to account for noise. According to Donahue, “results from this technique are promising but extracting the bump data from the preview information is not a robust process” (2001, p24).
Mercedes-Benz have produced a research car, the F700, that uses LIDAR (Light Detection and Ranging) sensors in the headlights “to scan the road so that the car’s suspension could firm up or soften its damping and direct the hydraulic shocks to absorb or counter the loads on each wheel.” It is claimed that their sensors can “measure the thickness of the painted lines on the asphalt” (Voelcker, 2008).

Look-ahead suspension systems are not the subject of this thesis. Even without look-ahead sensors, however, it may be possible to implement a modest form of look ahead by using data from the front wheels to anticipate the road encountered by the back wheels. This also was not the topic of the research for this thesis.

2.10.3. Control Hardware

There are a number of elements in the control hardware of a digitally controlled suspension besides sensors and actuators: filters, microprocessors, amplifiers, digital-to-analog, and analog-to-digital converters. This section looks at some of the control hardware used for electronically-controlled suspension. Control software can be processed by either a desktop or laptop computer, or a microprocessor. Desktop and laptop control is highly flexible but it is only appropriate for design purposes. Microprocessors that are easily reprogrammed are also widely available and these can also be used in both design and development work. There is a range of ready-made commercially-available integrated circuits called microcontrollers that can be used for control purposes and which can be easily programmed via computer packages.

There is a range of types of microcontrollers that can be used for electronic control: the 8051 family, the Microchip PIC range (Predko, 2008), Atmel AVRs, the Texas Instruments MSP430 family, and the Rabbit Semiconductor range of MCUs (Edwards, 2005, p16).

Programming of MCUs has now reached the point where they can be programmed in-circuit, which is perfect for experimentation with a number of control algorithms. For instance, Atmel devices can be programmed and reprogrammed via Ethernet (Donahue, 2001, p69), RS-232 or USB (Wikipedia, 2006; Wikipedia, 2007a). Reprogramming can be performed off-
board or it can be done in-circuit. Atmel AVR devices have been used in the automotive industry for “security, safety, powertrain and entertainment systems” (Wikipedia, 2007a).

Standards now also exist for communication between devices in an automobile:

“In Europe the dominant vehicle control network is CAN (Controller Area Network). This protocol was developed by Robert Bosch GmbH in the mid 1980s and was first implemented in a Mercedes Benz S-class car in 1991. CAN has since been adopted by most major European automotive manufacturers and a growing number of US companies are now using CAN. In the USA in 1994 the SAE Truck and Bus Control and Communications subcommittee selected CAN as the basis for the J1939 standard … The IS0 standardised CAN as an automotive networking protocol: ISO 11898 and ISO 11519-2.

Many of the world’s major semiconductor companies now offer CAN implementations. It is estimated that there are already over 140 million CAN nodes installed worldwide.” (Leen et al., 1999, p262)

Robert Bosch GmbH is the name of the company that was started by Robert Bosch in the 1880s. There now exist a number of networking standards that can be used side-by-side in the one vehicle: LIN (Local Interconnect Network), CAN, TTCAN (Time-Triggered CAN), FlexRay and MOST (Media Oriented Systems Transport) (Blijleven, 2010).

Some AVR microcontrollers feature models for automotive use with on-board communication using the CAN protocol. These are easily found by accessing the Atmel website: http://www.atmel.com.

The programming of modern MCUs is performed using a range of computer packages. The developer can program in machine code (McLellan, 1998), use high-level language compilers, or can even use graphical engineering packages such as Simulink to produce code from block diagrams (Donahue, 2001, p60; Atray and Roschke, 2003, p226). Matlab is a popular package for simulation, but it can also be used for programming microprocessors (Donahue, 2001, p60) and for the design of filters (Donahue, 2001, p59).

Dedicated analog filters will generally use op-amps (McLellan, 1998, p38). These amps do not generally need excessive shielding or expensive components as the frequencies of interest are relatively low, in the low end of the audio range 0.1 Hz to 50 Hz (refer to section 2.11).
Although semi-active dampers require much less power than active components, they do require more than just signal-level power. MR dampers supplied by Lord Corporation require a maximum of 2 A current for model number RD-1005-3. (The technical data sheet for this device is contained in the file PhD\Thesis\Appendix\Lord RD 1005 3 Damper.pdf.) Such dampers are highly inductive as the power is being supplied to coils to provide a magnetic field to the rheological fluid. They are also susceptible to overheating and a fuse may be needed to limit overheating (McLellan, 1998, p48).

2.10.4. Modelling Controlled Suspensions

This section presents a brief outline of some of the current research into models of particular controllable dampers for semi-active control. The major focus is the modelling of the MR damper.

Accurate modelling of suspension systems requires such details as the geometry of brackets, the modelling of bushes, hysteresis effects, and “nonlinearities such as backlash and dead zone” (Song et al., 2003, p223). Bushing and bump stops are a crucial part of a suspension. Bushes have been modelled also as a damper in series with the suspension (Lauwerys et al., 2004, p1483). Modelling in great detail is becoming more and more a part of the vehicle design cycle. “Most manufacturers in the world are now investigating the feasibility of zero, or virtual, prototype engineering. A definition of zero prototype engineering is the deletion of hand-built, or ‘soft’ toll, prototypes from a development program” (Bastow et al., 2004, p284). Computer-aided design has greatly reduced the number of physical prototypes required. However, modelling for the purpose of the development of control strategies occurs at earlier stages in the design and need not be as detailed. It is possible to “neglect actuator dynamics to focus on validation of the proposed approach” (Tan et al., 2005, p12). That is to say, simpler models can be used as a proof of concept before larger, more complex models are applied to the problem.

Hydraulic systems have been widely employed for suspension control, as discussed above. Tibaldi and Zattoni developed an algebraic model for a two DOF suspension using a

ER and MR fluids have been studied extensively as methods of damping control, and MR fluids are being used commercially in high-end sports and luxury cars (refer to section 2.10.1). “The discovery of these fluids dates back to the 1940s” (Guglielmino et al., 2008, p166). “ER and MR fluid respond to, respectively, an applied electric or magnetic fields with a dramatic change in their rheological behaviour. The main characteristic of these fluids is their ability to change reversibly from free-flowing, linear viscous liquids, to semi-solids with the yield strength swiftly and continuously controllable (milliseconds scale dynamics) when exposed to either an electric or magnetic field” (Guglielmino et al., 2008, p166).

ER dampers have been studied for a number of years and accurate dynamical models have been developed. ER and MR dampers have a high hysteresis, and there has been some research into the use of feedback to “linearize” the devices (Sims and Stanway, 2003, p83; Stembridge et al., 2006, p4).

“Adequate characterization of an MR damper has shown to be a challenge because of the device’s highly nonlinear dynamic response” (Schurter and Roschke, 2000, p1). MR Damper models used in the literature have varying levels of complexity. The simplest is one in which output force is proportional to current and is regarded as the model of the “ideal” MR damper (Reichert, 1997, p20),

\[ F_{\text{Damper}} = \alpha i \]

where \( \alpha \) is a constant and \( i \) is the current. The direction of the force is opposite to the stroke velocity.

Curves for MR damper force versus stroke velocity have been presented by Goncalves et al. in the context of determining damper response time (2003, p426). Also, in a 2002 paper Goncalves and Ahmadian (2002) present a similar response curve, shown below in figure 2.19. Note the linearity of the response in portions of the graph. It is not expressly stated in either paper how the data was collected.
Figure 2.19 MR Damper Damping Force Envelope (Goncalves and Ahmadian, 2002, p4)

Gordon and Best (1994) employed a less linear model, shown below in figure 2.20.

Figure 2.20 Map of Nonlinear Damper Forces (Gordon and Best, 1994, p333)

A nonlinear mathematical expression based on physical modelling was developed by Yu et al. They derived the following relatively simple expression,

\[ F_{\text{Damper}} = -C_e V + (ai^2 + bi + c) \text{sgn}(V). \]

Here \( V \) is the stroke velocity (2006, pp2-3).

Electronically-controlled devices have latencies caused by delays in the input and output devices as well as the communication interfaces, such as serial buses for USB or CAN communication. Ebau, Giua et al. take into account the time delay in both sampling and updating and in response time of the semi-active damper (2001, p95). They assumed response times of the order of 10 ms.
If greater accuracy is required a more detailed model of damper hysteresis is required. Karkoub and Zribi use a model developed by Dyke et al (Karkoub and Zribi, 2006, p38; Dyke et al., 1996; Dyke et al., 1998; Jansen and Dyke, 2000). It should be noted that the systems studied by Dyke et al. were intended for the control of a building against seismic shocks.

“A significant problem faced in application of [an MR damper] is accurate modelling of its nonlinear hysteretic behavior. Spencer et al. … developed a numerical model of the SD-1000 MR damper model that characterizes its behavior with seven simultaneous differential equations that have 14 parameters. More recently, Schurter et al. … used Adaptive Neuro Fuzzy Inference System (ANFIS) to characterize behavior of the SD-1000 damper. ANFIS is a powerful neuro-fuzzy algorithm that adjusts the membership functions (MFs) of a fuzzy inference system (FIS) such that the FIS can mimic the actual behavior of the MR damper with a high degree of accuracy while reducing the computational time significantly” (Atray and Roschke, 2003, p223).

Various authors have modelled the “viscoplastic behaviour” of the MR fluid directly (Spencer et al., 1997; Breese and Gordaninejad, 2003; Butz and von Stryk, 2002). Hysteresis in MR dampers has also been modelled by Hudha et al. (2005, p235) and Ahn et al. (2009).

These various models cover a diverse range of applications, including vehicle suspension. The model developed by Spencer et al. was intended for the control of “civil engineering structures subjected to strong earthquakes and severe winds” (1997, p1). Schurter and Roschke developed their model for the SD-1000 MR fluid damper which was “put into commercial use as a semi-active suspension system for large on- and off-highway vehicle seats” (Schurter and Roschke, 2000, p1).

### 2.11. Practical Suspension Control Limitations

There are a host of practical factors affecting the function of a suspension system: wear and aging of components, balance and alignment (Scheffer and Girdhar, 2004), unwanted harmonics, bushing (Sung and Jang, 2006), delays in control response, the properties of hydraulic fluids, bump-stop characteristics, tyre types, road types, etc. Component wear and
aging needs to be accounted for. Suspensions should continue to work well even as the components age and weaken. Components should not wear appreciably. As far as possible, steps should be taken so that the control system adjusts for changes in component characteristics.

One important limitation for electronically-controlled components is the time delay between the measurement and the application of the control. “The very use of feedback control introduces these effects” (Bellman, 1966, p5). The road or chassis disturbance needs first to be detected by a transducer, sometimes the data will be conveyed to the microprocessor via a local network inside the car, the control signal then needs to be calculated by the processor, and finally the actuator needs to respond by producing the required change in physical characteristic. Each of these steps takes time. Improving response time may not necessarily be just a matter of improving the processing step time of the microprocessor. However, currently the response times of transducers and actuators, and the speeds of automotive networks and microprocessors are constantly improving.

Since the early 1990s it has been claimed that an overall response time of the order of 15 ms is “necessary to provide good control of the primary suspension” (Ivers and Miller, 1991, p331). Given an algorithm step size of 15 ms the highest road frequency that can be comfortably controlled by the electronic portion of the suspension is of the order 30 Hz. Frequencies higher than this must be dealt with by the passive elements. A vehicle travelling at 100 kph (approximately 60 mph) will cover a distance of 42 cm (approximately 16 inches) in 15 ms. Thus the smallest bump that can be theoretically smoothed by the suspension is 42 cm long. This is clearly unacceptable. Travelling at the same speed, a response time of 1 ms allows the handling of bumps 28 mm (approximately one inch) across.

The tyre is an important passive component of the suspension, although it is not in the controlled portion of the suspension between the wheel and chassis and, in most cases tyre deflection or pressure is not used as a control input. “Usually tire damping … and the speed of tire deformation … are so small that variation of dynamic tire force due to damping is neglected.” (Hyvärinen, 2004, p43)

Sometimes simple filtering of output can produce improvements. For example, Donahue found that, “experimental data depicted considerable model error in the range of 1 Hz to 5
Hz” (2001, p15). To compensate, an “error filter” was used; “system output was attenuated at 1 Hz and amplified at 5 Hz by two”. Further changes were also introduced to “improve tracking near the resonant modes of the suspension” (2001, p16). These changes produced great improvements in the experimental setup.

2.12. Computer Optimization of Suspension Control

In recent years computer techniques have been used for the optimization of engineering structures. Nonlinear systems and intractable optimization problems can be handled using an enormous number of models, each with slightly different characteristics. This section gives some examples of the application of computer-intensive techniques to suspension control design of both seismic and vehicular suspensions. Most attention is given to Evolutionary Algorithms (EAs) as these are the method used in the numerical experiments carried out for this thesis.

2.12.1. Evolutionary Algorithms

In this section somewhat greater emphasis is placed on evolutionary algorithms since these were used in the numerical experiments of this thesis to fine tune the parameters of various suspension control designs. Evolutionary Algorithms (EAs) are used to search a problem space and converge on highly scoring results. In fact, optimality cannot be assured but EAs can find acceptable solutions where analytical techniques are either intractable or impractical (Goldberg, 1989). Values found are said to be “suboptimal”. The processes are analogous to those of biological evolution. In general the fittest genes from one generation are chosen for breeding the next generation. Mutation and crossover ensure that variety is maintained (Goldberg, 1989).

Evolutionary algorithms are sometimes applied to the development of fuzzy controllers, and this technique has been applied to the development of a semi-active suspension control algorithm (Hashiyama et al., 1995; Nawa et al., 1999; Caponetto et al., 2003). According to Hashiyama et al., “setting the performance index is the only procedure for the designer of the
controllers” (1995, p166). There is no need to design a control law since this is discovered by the evolutionary algorithm. This thesis, in contrast, specifically investigates control laws and uses EAs for purposes of comparison.

Yan and Zhou used genetic algorithms to optimize the fuzzy-logic control of a MR damper used for seismic stabilization (2006). Fuzzy controls have been used by Yu et al. to implement skyhook and groundhook schemes (see sections 2.4 and 2.5) for a semi-active suspension based on a quite extensive nonlinear model of an MR damper (2006). They achieved a 7% reduction in RMS acceleration in physical experiments.

Evolutionary algorithms use randomization for mutation and selection, but in many cases the phenomenon being analysed is itself stochastic. “In real life black–box optimization problems, the existence of noise during evaluation is inevitable. Sources of noise can vary from noise in the sensors, actuators, or because of the stochasticity pertaining in some problems such as multi–agent simulations.” (Bui et al., 2005, p779) In the case of vehicle suspensions the stochastic input and source is the road input. Random noise has the effect of slowing the evolution, particularly when Pareto fronts are being sought for multi-objective evolution (Bui et al., 2005). “Noise can mislead the search process considerably” (Dumitrescu et al., 2000, p63).

Tan, Dyke et al. applied genetic algorithms to the development of control algorithms for multi-storey seismic structures (2005). Similar to the method used in this author’s experiments the population genes contain control parameters. They concluded that genetic algorithms were “flexible” and robust.

While EAs cannot guarantee optimality they will generally be robust, in the sense that small changes in parameters will not result in large degradations to performance. “We sometimes try to detect robust solutions. Solutions that are very sensitive to small perturbations of their parameter values may not be useful in certain situations” (Dumitrescu et al., 2000, p63). For engineering design, a robust suboptimal design might even be preferred over a global optimum that is highly sensitive to parameters’ variation (refer to section 2.3.2.).
2.12.2. Components of Evolutionary Algorithms

This section very briefly outlines the theory behind evolutionary algorithms, particularly those aspects that relate to the experiments.

In the early works on genetic algorithms, genes were represented as binary bits (Goldberg, 1989). However, in applications where genes are naturally used to represent floating-point measurements, such as suspension spring stiffness in a semi-active suspension, the genes may be more profitably represented directly as floating-point numbers. “In a classical genetic algorithm, these variables would be encoded as binary genotypes. Here, a conversion process is required to translate between the genotypic representation and phenotypic representation and vice versa. Floating-point representations do not require a conversion process, making them faster to manipulate. Furthermore, they permit greater precision than binary code, constrained only by machine precision.” (Purshouse and Fleming, 2001, p29)

One way to attempt to improve evolutionary algorithms is to induce a high degree of variation at the start and slowly “cool” the evolutionary process. As the EA “cools” there is less diversity, but there is also less “noise” in the evolutionary process. The change from high to low temperature is known as simulated annealing. The temperature can be associated with a number of factors: average extent of a mutation, number of mutations, probability of gene acceptance as a function of fitness, number of crossovers.

The standard deviation of mutations is a possible candidate for a temperature variable. As the genetic algorithm “cools” (or “heats up”) the standard deviation of the mutation can be decreased (or increased).

A rule of thumb for adaptive evolutionary algorithms is the “1/5 success rule”. In broad terms, “the ratio of successful mutations to all mutations should be 1/5” (Dumitrescu et al., 2000, p264). This can be achieved by using a feedback process and changing the standard deviation of the mutation, or by varying the number of mutations. A way to do this is to update the standard deviation “every $n$ generations with learning rates $c$ and $c^{-1}$” (Dumitrescu et al., 2000, p266). The following formula for varying the standard deviation is applied,
Here $p$ is the ratio of successful to unsuccessful mutations in the previous ten generations. The value of $c$ is in the range 0.82 to one; usually it is 0.85 (Dumitrescu et al., 2000, p267).

Mutations are random changes in the genome. Mutation is the primary mechanism for continually re-introducing diversity into a population to allow the exploration of new, possibly fitter solutions. With Boolean-valued genes, mutations take the form of a simple change between one and zero (Goldberg, 1989, p16). With real-valued genes, mutations can be more sophisticated, and an “infinite set of alternatives exists. Gaussian mutation is an attractive method of choosing an alternative. This operator generates a new value based on a normal distribution, centred over the current value. The standard deviation defines the likelihood of generating a value close to the original” (Purshouse and Fleming, 2001, p30).

Genes are selected randomly, but genes with greater fitness have a higher chance of being selected. In the simplest selection algorithm, genes are selected randomly with a probability that is directly proportional to their fitness, referred to as “proportional selection” (Dumitrescu et al., 2000, p46). Suppose the $n$ chromosomes of the current population are represented as,

$$P(t) = \{x^1, x^2, ..., x^n\}.$$  

Each $x^i$ can be a binary value or a vector of floating-point values. The selection probability of chromosome $x^i$ is given as,

$$p_j = \frac{f(x^j)}{\sum_{j=1}^{n} f(x^j)}.$$  

**Equation 2.11**

Here $f$ represents the fitness function (Dumitrescu et al., 2000, p47). Scaling of fitness values will alter the probabilities of selection (Dumitrescu et al., 2000, p59).
A number of mutation schemes exist for floating-point genes. If \( x_i \) is a real-valued gene, out of a vector of such values, then the following could be used to perform the mutation,

\[
x_i' = x_i + \alpha_i N(0, \sigma_i),
\]

where \( \alpha_i \) is a real parameter and \( N(0, \sigma_i) \) is a normally distributed random variable (Dumitrescu et al., 2000, p201). “A multiplicative lognormal perturbation may sometimes be more interesting.” This can be expressed as,

\[
x_i' = x_i e^{\beta N(0, \sigma_i)},
\]

**Equation 2.12**

where \( \beta \) is a real parameter (Dumitrescu et al., 2000, p202). Schemes also exist to stop values from going outside the feasible range (Purshouse and Fleming, 2001, p30). This last method may favour larger values over smaller and may “drift”. Adjustments can be made to counteract this drift.

Crossover (or equivalently “recombination”) in biological evolution involves the splicing and recombining of parent genes. For a particular gene the child may take genetic material from one parent’s gene “before” the splice and from the other parent’s “after” the splice. This provides the child with the chance to share some benefits of the mother’s genes with those of the father’s. Crossover increases diversity but keeps much genetic material intact. The implementation of crossover using binary genes is relatively straightforward, but it is “potentially noncontinuous and depends on the variety, in terms of bits, between the parent chromosomes” (Purshouse and Fleming, 2001, p29).

A simple form of crossover with real-valued genomes is simply to randomly swap random genes. For example, suppose the following genome values are given,

\[
x = (x_1, x_2, x_3, x_4, x_5), \quad \text{and} \quad y = (y_1, y_2, y_3, y_4, y_5).
\]

One form of mutation is to simply swap individual values, for example,

\[
x = (x_1, x_2, y_3, x_4, y_5), \quad \text{and} \quad y = (y_1, y_2, x_3, y_4, x_5).
\]

This is termed discrete crossover (Dumitrescu et al., 2000, p190). Rather than just swapping values, weighted sums of genomes from different sites can be calculated producing continuous crossover (Dumitrescu et al., 2000, p191), as in the following example,
\[ x = \left( x_1, x_2, \frac{(x_3 + y_3)}{2}, x_4, \frac{(x_5 + 2y_5)}{3} \right), \text{ and} \]
\[ y = \left( y_1, y_2, \frac{(x_3 + y_3)}{2}, y_4, \frac{(2x_5 + y_5)}{3} \right). \]

Elitism is the forced retention of a set of the highest-scoring genes into later generations (Goldberg, 1989, p115; Zitzler et al., 2000, p188). In this author’s experiments a certain proportion of the genes of one generation were included into the next. Zitzler et al. found that elitism was a very important factor in the success of the EAs in their experiments.

With multi-objective optimization it may be that one gene scores better in one objective, but the other is more fit in another objective. Which gene is the more fit? One solution is to weight the components. The other solution is to keep both genes; this is referred to as Pareto optimization. For example, if \( J_c \) is the comfort measure and \( J_r \) is the measure of the capacity to stay within the rattlespace, then the scores can be combined into an ordered pair, \( (J_c, J_r) \). In the following pairs,

- \( (200, 300) \) and \( (400, 500) \),

the second gene dominates the first because it scores higher in both components. In the following case however,

- \( (500, 300) \) and \( (200, 400) \),

neither gene dominates the other (Goldberg, 1989, pp198-9). A solution is Pareto optimal if it is not dominated by any other solution (Goldberg, 1989, p198; Fonseca, 1995; Zitzler et al., 2000, p175). The set of such genes defines a “front” of genes that are Pareto optimal. Genes below the Pareto front will not be optimal. Genes that lie along the Pareto front will have the highest chance of passing on their genes to future generations (Goldberg, 1989, pp199-201; Zitzler et al., 2000, p176).

The Pareto front could be determined by applying different weights, plotting the optimal solutions for each weighting, as this author and others have used previously (Bourmistrova, 2005). Means exist however for finding the Pareto front in a single application of an EA. Fonseca and Fleming use the count of the number of times that a gene is dominated as the measure of fitness and this is said to provide better convergence especially under noisy conditions (1993).
With the use of Pareto fronts, the evolutionary process can suffer a loss of diversity (Bui et al., 2005, p783). This is called “niching” (Purshouse and Fleming, 2001, p6), and it is a major difficulty for multi-objective optimization using Pareto fronts (Zitzler et al., 2000, p176). In an effort to circumvent this, fitness can be decreased when genes are too close together. Various techniques are used (Purshouse and Fleming, 2001, pp6-9). Genetic drift generally creates a loss of diversity and “is one of the major factors responsible for the premature convergence of the search process” (Dumitrescu et al., 2000, p98).
3. Vehicle Suspension Performance Measures

Suspension performance measures comprise a central theoretical component of this thesis. The discussion of section 2.9 outlines various suspension performance goals in the literature, and it forms the broader context for this chapter, which focuses on measures applied in later experiments.

In very broad terms, suspension design attempts to reach a compromise between softness and hardness. The softer the suspension the more comfortable it will be. If it is too soft, the wheel will periodically hit violently against the limits of suspension vertical travel, or the vehicle will be uncontrollable over large bumps. At the other extreme, an excessively hard suspension will track well, but it will be unacceptably uncomfortable. There is no single solution to the compromise. For example, racing cars have a harder suspension than touring cars since comfort is of less concern than the ability to corner and handle bumps at speed.

The formulation of the performance measure for an idealized optimization problem can have a critical impact on the optimal solution. For instance, the provably optimal strategy that minimizes quadratic indexes is the smooth, LQR feedback control, discussed in section 2.3.1. By contrast, the analytically proven optimal control that minimizes time in returning to rest is the bang-bang control (see section 2.3.2). Both controls are provably optimal, and yet one is continuous and the other switches discontinuously. The controls could not be more different and yet they are both “optimal”, using different measures of performance.

In contrast to traditional passive suspensions, electronic systems allow a virtually unlimited range of control possibilities. They are capable of greater smoothness and handling, but they are also capable of extreme behaviour. Setting the performance measure is an important part of the formulation of an optimality problem. Caution is needed because the wrong performance measure can mask unwanted effects, as shown perhaps most dramatically in the case of RMS acceleration measures.

A careless formulation of an optimization problem can even result in controls that are absurd and wildly impractical. For instance, the unconstrained version of the minimum-time problem requires an infinite force for an infinitely small period of time (MacCluer, 2005, p110; Ross,
As noted by MacCluer, “the application of optimal control to practical problems is an art, requiring the practitioner to perform many analytic and numerical iterations to reach an acceptable (but often not optimal) solution to the original problem” (2005, p113).

To sum up, caution and perhaps a degree of conservatism are needed when selecting performance measures, especially when using atypical performance measures. This is even more the case where there is much greater flexibility in control, as in electronically-controlled systems. Certainly, comprehensive testing is required, alongside theoretical justification.

### 3.1. Isolation and Comfort

Suspension comfort is typically measured as the RMS of acceleration. This measure has a proud pedigree, and a large number of researchers consider RMS acceleration to be the main if not standard measure of riding comfort (Caponetto et al., 2003; Cole, 2001; Els et al., 2005, p795; Gao et al., 2006; Hiromatsu et al., 1993, p2135; Song et al., 2003; Uys et al., 2006; Vaughan, 2004).

According to Smith et al., RMS acceleration proved to be a good predictor of comfort (refer to section 2.9.1). The various metrics studied by Smith et al. used frequency data (1978). They showed that RMS acceleration correlates well with perceived comfort. It should be noted that none of the metrics used in their comparative study incorporated jerk (1978). In any case, in the late 70s, there were no practical passive mechanisms that directly controlled jerk. Furthermore, passive systems are typically approximated to linear systems, and they only produce force discontinuities under rough road conditions (as discussed later is section 4.3). Thus the problems of jerky control were not initially apparent, or at least were not dealt with analytically.

The following measure has the same ordering as the RMS acceleration measure,

\[
\int_0^T \ddot{y}(t)^2 \, dt.
\]

**Equation 3.1**
Minimizing this produces the least square solution. In matrix form a similar measure is the quadratic performance index (MacCluer, 2005, p139), and equation 3.1 is a special case of this. The least squares measure with linear systems provides analytical methods for both optimization (refer to section 2.3.1) and state estimation (see section 2.3.1.2). This produces the same ordering as the $L^2$-norm and the RMS measure. The least squares measure is convenient as a performance goal, and undoubtedly “a popular optimality criterion among control engineers is the minimization of an $L^2$ cost function” (Ross, 2009, p42).

A similar measure found in the literature uses a heavier weighting for larger values of acceleration by taking the fourth power of acceleration rather than the second. The Vibration Dose Value (VDV), discussed by Deprez et al. uses the fourth power of acceleration,

$$\int_0^T \ddot{y}(t)^4 \, dt.$$  

**Equation 3.2**

(2002, p1498). This produces the same ordering as the $L^4$-norm. At the extreme is the $L^\infty$-norm. For the purposes of this thesis this is equivalent to the “maximumum”,

$$\max_t \ddot{y}(t).$$

One could perhaps argue that comfort is as bad as the worst shock, but it seems reasonable to include an overall gauge using some kind of summation measure.

It should perhaps be made explicitly clear that suspension force on the chassis and acceleration of the chassis are assumed to be directly proportional, since the mass of a vehicle is assumed to be constant. Comfort measures using vertical acceleration have the same ordering as measures using vertical force.

The quadratic measure of acceleration is by no means the only way to gauge isolation and comfort. Measures such as the VDV measure are generally not conducive to analytical means of optimization and typically require numerical techniques, but they have been used. In summary, the least squares measure of acceleration is well understood, has a long history, is mathematically amenable and has been thoroughly accepted by the engineering community. However, its mathematical convenience does nothing to prove that this is the “natural”
measure of comfort. Where numerical optimization techniques make virtually any measure feasible, perhaps higher power measures produce a better performance index.

A number of researchers have investigated the use of jerk (the rate-of-change of acceleration) as a comfort measure (Hrovat and Hubbard, 1981; Hrovat and Hubbard, 1987; Paddison et al., 1994; Hashiyama et al., 1995; Ahmadian and Vahdati, 2003; Ahmadian et al., 2004; Ahmadian and Vahdati, 2006; Yamakado and Abe, 2006). In some cases the term “jerk” seems to mean little more than acceleration discontinuity (Reichert, 1997, p41; Hyvärinen, 2004), and still others maintain that RMS acceleration is a sufficient predictor of ride comfort compared with jerk, as discussed above (section 2.9). In any case, jerk is being used more-and-more widely. Certainly, jerk is being investigated for use in the control of industrial robot arms (Koh et al., 1999; Kyriakopoulos and Saridis, 1988; Ben-Itzhak and Karniel, 2008; Cao et al., 1997; Macfarlane and Croft, 2003).

Jerk is generally estimated in practice from accelerometer measures. While dedicated jerk sensors have been developed (Yamakado and Abe, 2006) they are not readily available. The controls developed here are controls over jerk. They use estimates of displacement, velocity and acceleration, but they do not need estimates of jerk. However, noting the jerk achieved in controls is important in experimentation if not in the on-board system. It is only the chassis jerk that is of concern and so jerk can be estimated from an accelerometer placed on the chassis. This requires differentiation of the accelerometer measures, and this process is very sensitive to high-frequency noise. It will be necessary therefore to filter out high-frequency noise.

Just as with acceleration, there are different possible performance measures involving jerk: RMS jerk (equation 3.1), the “quadratic” measure,

\[ \int_0^T \dddot{x}^2 \, dt, \]

and the integral of the fourth power of jerk,

\[ \int_0^T \dddot{x}^4 \, dt, \]

\textbf{Equation 3.3}
which gives higher weight to greater values of jerk, and is similar to the measure of equation 3.2. This is the measure used in this author’s experiments. Another measure that has been used is simply the maximum absolute value of jerk (Deb and Saxena, 1997, p558).

While the usefulness of acceleration as a measure of comfort has been demonstrated there are a number of reasons to suppose that a suspension optimized for jerk could perform better and feel more comfortable, and perhaps may even have a more natural overall “feel”. Only extensive psychological testing could settle the matter conclusively, but various arguments are presented here to support the contention that jerk is a better measure of suspension comfort than acceleration, and some corroborating empirical evidence is cited.

Humans constantly experience the force of gravity on their bodies without discomfort. A sustained, unvarying acceleration of 0.8 g (roughly 8 m/s²), will not alone cause discomfort, and a constant acceleration of 1.2 g (roughly 12 m/s²) will also not cause great discomfort. However, a sudden change between these accelerations will be very noticeable.

Figure 3.1 shows two acceleration profiles. Suppose that these represent the acceleration profiles of a vehicle traversing a relatively bumpy road. Suppose the profiles represent sizable bumps over a period of 15 seconds or so, that might cause some degree of discomfort to a passenger.

The profile on the right has discontinuities where the acceleration jumps suddenly from one value to another. If the profile on the left is somewhat uncomfortable then it is very clear that the acceleration profile on the right is very much more uncomfortable because of the sudden changes in acceleration. Both profiles have exactly the same magnitude of acceleration at each point; therefore both profiles produce exactly the same measure of RMS acceleration. If trajectories with the same RMS acceleration produce different levels of comfort then the measure of RMS acceleration is insufficient to measure comfort. The discomfort caused by the sudden changes in acceleration is completely masked by the RMS acceleration measure.
Figure 3.1 Acceleration profiles with the same RMS Value – The left profile is continuous but the right profile has discontinuities.

Every discontinuity in the profile on the right represents a spike in jerk. Of course, pure discontinuities do not occur in nature, but they are approximated. In real-world systems where force actuators are suddenly switched on or off, there will nonetheless be a very large, uncomfortable, if not infinite spike in jerk. Nonetheless, the logic of the argument remains the same. Sudden changes between moderate acceleration levels are not penalized by the RMS acceleration measure.

Sudden changes in acceleration are accompanied by large albeit finite spikes in jerk, which cause discomfort and which are penalized by an RMS jerk measure. It is not uncommon in the literature to talk of “force discontinuities” or “acceleration discontinuities” (Reichert, 1997, p69; Ahmadian and Vahdati, 2006, p153; Harris, 2004; Chang et al., 1999, p3276), using this as shorthand for approximations to discontinuities, which have accompanying finite spikes in jerk.

There are a number of reasons why acceleration discontinuities can arise with highly flexible electronic controls:

- Semi-active adaptations of active controls have a natural tendency to produce jerk at the boundary of the passivity constraint (see sections 2.6 and 4.8).
- On-off controls are much easier to engineer electronically.
- On-off dampers can be interpreted as being “optimal” for some purposes: for instance for energy absorption, as explained in section 4.2, and bang-bang controls are optimal in a minimum-time sense, as explained in section 2.3.2.3.
- Pure sliding-mode control, without a linear control near the sliding plane, exhibits “chattering” (see section 2.3.2.4).
Thus, if the RMS measure is used to measure comfort, there may seem to be good theoretical reasons for applying a discontinuous control force. In control problems where Pontryagin’s Principle is applied, for instance, bang-bang is optimal (see section 2.3.2). But a bang-bang control over acceleration is demonstrably highly uncomfortable.

Force discontinuities known as “chattering” also result from the pure sliding-mode control (see section 2.3.2.4) where linear controls have had to be artificially inserted around the switching plane to remove uncomfortable force discontinuities.

With electronically controlled suspension, switching techniques become feasible, and in fact are often the simplest control to implement electronically. Experimental active and semi-active suspension systems have used simple on-off switching for control (Ivers and Miller, 1991; McLellan, 1998). Not surprisingly, experiments reveal uncomfortable force discontinuities.

A semi-active suspension that attempts to approximate an active suspension’s control force will suffer discontinuities because of the passivity constraint, as explained below in section 4.8. As noted by Ahmadian et al., “at zero crossings of the velocity, [a] conventional skyhook introduces a sharp increase (jump) in damping force, which, in turn, causes a jump in sprung-mass acceleration” (2004, p580). Again, the discomfort caused by such discontinuities is completely disregarded if acceleration alone is used as the performance measure. This represents a serious problem for semi-active suspension control and the matter is discussed at length below in section 4.8.

There is a great difference between a discontinuous control over acceleration, and a discontinuous control over jerk. To visualize this, consider a very simple “elevator example”. Suppose bang-bang acceleration control was applied to an elevator, as in figure 3.2 (a), where the elevator car moves from “floor 3” to “floor 0” (Here displacement and acceleration are shown on the same graph for convenience).

Using RMS acceleration as a measure of comfort, this might seem to be a good candidate for elevator control since it brings the system to rest in a finite and minimum amount of time for the given force. Such a control however would contain three sudden changes in acceleration: once at the start, once at the end and a complete reversal of acceleration direction in the
middle. Each of these changes in acceleration represents a sudden change on the force experienced by the passenger in the elevator. There are three “jerks” (actually spikes in jerk) in this control as the acceleration and force felt on the body suddenly changes, at the start, half way, and at the end. It takes little imagination to realize that this would provide an extremely uncomfortable elevator ride.

The control of figure 3.2 (b) is also a bang-bang control, in this case involving four sudden changes in jerk. However, the acceleration is continuous and changes smoothly. A constant jerk smoothly increases the magnitude of acceleration, smoothly reverses the acceleration, and finally eases the elevator to a stop at the end. This is clearly a much more comfortable control, and it is optimal in terms of bringing the elevator car to rest in minimum time using constrained jerk.

Again, RMS acceleration does not indicate that the first control is more uncomfortable than the second. In fact, the second control has a 15% higher level of RMS acceleration (see

![](image)

Figure 3.2 “Elevator Example” using (a) Bang-Bang Acceleration and (b) Bang-Bang Jerk
appendix, section 8.5). On the other hand, an RMS measure of jerk will clearly indicate the control over acceleration is uncomfortable.

Constrained jerk might also be beneficial for delicate machinery or other inanimate cargo as well as passenger comfort. Jerk constraint has been suggested for industrial robotic systems with the aim of increasing the life span of equipment by reducing vibration:

“… a robotic assembly system with high speed motion needs the constraint for maximum jerk to improve the positional accuracy and to prevent the mechanical system from … vibration … The anti-vibration is the key factor for determining the life cycle of the mechanism. In robotic systems, the jerk constrained motion guarantees a smooth and stable motion”. (Koh et al., 1999, p273)

It has been suggested that jerk is a better indicator of discomfort for vehicle suspensions “because drivers’ bodies are sensitive to jerk” (Yamakado and Abe, 2006, p2). The body contains a deformable mass of soft tissue surrounding a rigid skeleton. Under constant acceleration the body deforms due to force, but it reaches equilibrium. Indeed, humans can quite easily sleep under the constant force of gravity. However, in a body experiencing jerk, the deformation does not reach equilibrium. Thus jerk can be seen as a kind of measure of the “jostling” of the human body. Similar arguments could also apply to inanimate objects especially those with liquid or moving parts. Jerk could also contribute to the loosening and wear of machine parts.

Empirical experiment into human arm movement has shown that humans do not use only acceleration in the control of arm movement. Flash and Hogan found that human arm movements were consistent with the minimization of jerk but not with minimization of acceleration (Flash and Hogan, 1985). Actually, the “limited resolution of experimental data” did not allow them “to establish unequivocally which one of the two models, jerk minimization or snap minimization, offers a better fit” (Flash and Hogan, 1985, p1698).

“Snap” is an even higher-order derivative of distance: the rate-of-change of jerk. There are a number of extant terms for higher-order derivatives of distance. “[Jerk] has also been called a ‘jounce,’ a ‘sprite,’ a ‘surge,’” and a “spasm”. The fourth time derivative is referred to as “snap” while the fifth and sixth derivatives are sometimes called “crackle” and “pop” (Sprott, 1997, p538; Ben-Itzhak and Karniel, 2008). Flash and Hogan concluded that the human body
itself prefers jerk minimization, or perhaps even snap minimization, over acceleration minimization. This suggests that a suspension optimized for jerk could feel more “natural” and more pleasing than one optimized for acceleration.

Flash and Hogan offer an interesting hypothesis to explain why humans might use jerk in the control of bodily movements.

“The rationale for jerk minimization in biological trajectory planning does not lend itself to self-evident, casual explanations. Given the fact that the movements under consideration occur at moderate speeds and do not subject the system to undue stress, it is unlikely that such a strategy has evolved to minimize the “wear and tear” on the system. It is possible that the objective is to minimize unwanted, abrupt changes in the forces transmitted to objects carried by the hand. Another possibility is that the objective is to maximize the predictability of the trajectory, which is consistent with minimizing its higher time-derivatives. To discriminate between these and other possibilities will require further work”. (1985, p1698)

Harris has questioned the use of higher-order smoothness than jerk. “Trajectories with higher order discontinuities (i.e. very smooth) require longer times to reach a given state for a given command signal. Keeping the order as low as possible would be beneficial, but there are biomechanical limits” (Harris, 2004, p114). It seems likely that human movement is a trade-off between optimization for one or more of a range of factors, but also for achieving this in an environment in which the “neural control signal is corrupted by noise” (Arechavaleta et al., 2008, p6).

One of the problems in determining the order of smoothness of human movement is that higher-orders of smoothness are difficult to determine from noisy data: “Measurements tend to be noisy, so that successive differentiations of time series of data become rapidly meaningless. Low-pass filtering reduces the noise, but by its nature, smothes out the discontinuities” (Harris, 2004, p100). In any case, whether human movement is smooth at the level of jerk or of higher time derivatives, it is abundantly clear that human movement does not generally involve discontinuities in acceleration.

Another possible argument against RMS acceleration is that it ignores the force of gravity. The human body experiences acceleration as a force, and while the measure of RMS
acceleration includes the up-and-down motion of the body, it does not include the much larger force (generally) due to gravity. If this force were included in the RMS acceleration measure of comfort then a perfectly flat road would produce almost the same measure of discomfort as an undulating one. However, higher order derivatives such as jerk are zero over flat roads and produce measures over corrugations that are more consistent with human discomfort.

Although comparisons have been made between controlling for jerk and for acceleration in purely passive systems (Hrovat and Hubbard, 1987), traditional, passive suspension control produces continuous forces in any case, at least when the road surface is continuous in slope (see section 4.3). The suspension force therefore is at least continuous in the case of passive systems, barring road discontinuities and damper value effects. With electronic suspensions, on the other hand, switched, discontinuous controls are the easiest to implement, and are even optimal for some performance criteria.

As briefly described above, switched force controls may seem an attractive option for the engineer for a number of reasons and such experimental systems have been investigated (McLellan, 1998, p71; Guglielmino et al., 2008; Ivers and Miller, 1991, p336; Crosby and Karnopp, 1973, pp121-2). Discontinuities can also arise as an artefact of a semi-active control following an active control (see section 4.8), or simply as artefacts of power supply switching. Such discontinuities would not necessarily affect an RMS acceleration measure of comfort. The consistent way to resolve the matter when investigating electronically-controlled suspension is to penalize acceleration discontinuities by using jerk as at least a component of a comfort measure.

Despite all this, there may be physical reasons why acceleration should be considered where appreciable power is being absorbed by the human body, especially at higher frequencies. Power absorption is the basis of the CAP measure (Donahue, 2001) (refer to section 2.9). This measure was used for experimentation with a HMMWV. The ability of a military vehicle to pass over rough terrain at high speed is a priority; passenger comfort is secondary. However, these vehicles in such terrain are near the limits of human power absorption, and this is a separate suspension goal relevant for military vehicles, and perhaps racing vehicles.
Different vibration frequencies affect different parts of the body, and the effect depends very much on frequency (Bastow et al., 2004). Higher-frequency vibrations have the greatest effect on the parts in most contact with the seats – back, buttocks and legs – but they also affect the head and neck (Cole, 2001, p331; Donahue, 2001, p26; McLellan, 1998, p15). Perhaps frequencies above say 5 Hz would be perceived by the amount of power that they transfer to the body’s soft tissue. More experimentation is needed to determine if there is a roll-off in jerk-related discomfort to acceleration-related discomfort at higher frequencies. However, even here, a jerk penalty also penalizes acceleration, whereas acceleration can mask discomfort, as explained above.

On balance the arguments of this section indicate that jerk is a superior measure of comfort for suspension systems, and there is some corroborating experimental evidence to support this. In fact, with the advent of electronic control, comfort measures using acceleration are likely to be inadequate.

3.2. Tracking

Suspension control theory has rested heavily on linear theory, and the RMS stroke index, of equation 3.4, is an often used measure of tracking (Song et al., 2003; Vaughan, 2004, p62; Redfield and Karnopp, 1988, p238). This is partly due to the successful application of this measure in variational calculus and statistics. However numerical optimization techniques open up the possibility of using more tailored performance indicators.

\[
J_R = \int_{0}^{T} s^2 \, dt.
\]

Equation 3.4

Imagine a vehicle travelling over a corrugated road, with undulations up to say one fifth of the rattlespace width. Suppose that the suspension control is capable of keeping the chassis perfectly flat, with the wheels made to move up and down to exactly match the road height underneath. Such a control can be theoretically achieved with an active suspension. In fact, the Bose website features a remarkable demonstration of a car using their active Bose®
suspension traversing a “bump course” in which the car body stays almost perfectly level but
the wheels move furiously up and down to exactly match the road corrugations, over middle
to low frequencies (Bose, 2007). As astonishing as this demonstration is, it represents the
very simplest possible control algorithm, which for the sake of discussion will here be called
the “flat” control: \( y(t) \equiv 0 \) for all \( t \). If \( y \) represents chassis height and \( s \) represents stroke (as
in section 2.4) then the stroke movement must match the road exactly:
\[
s(t) = y(t) - r(t) = -r(t).
\]

This is not to suggest that Bose use exactly the flat control for this demonstration. The
demonstration does verify however the feasibility of engineering the inverse dynamics
required for this control, at least for lower frequency vibrations of moderate size. Even so, it
is the theoretical possibility of the flat control that is at issue here.

The important theoretical point is that there is no inherent negative effect of wheel
displacement inside the rattlespace. The flat control over small bumps has perfect comfort;
the comfort measure is zero by any reasonable measure. Furthermore, road normal force is
perfectly constant (at least the component of road normal force contributed by the heavy
chassis, excluding the wheel) and so there is little or no change of traction force. The flat
control algorithm dramatically demonstrates that it is not stroke travel which has a negative
effect on tracking, since the wheels are moving up-and-down within the rattlespace to match
the road corrugations, and yet the control has perfect comfort and theoretically perfect
traction (ignoring tyre distortions).

The extreme case of the flat control demonstrates the general theoretical point that the wheel
should move within the rattlespace to produce as smooth a chassis trajectory as possible. It is
important to make good use of the rattlespace travel available in order to provide as smooth a
ride as possible, and to remove unnecessary chassis movement. Unnecessary chassis
movement creates discomfort, degrades tracking and contributes to instability.

Of course, the problem with the extreme case of the flat control quickly becomes obvious as
soon as it encounters a bump that is large enough to cause the chassis to hit against the
vertical travel limits of the rattlespace. And the closer the chassis approaches the vertical
travel limits, the more likely it is that future movement will cause a collision with the
rattlespace limits. Nonetheless, it is the potential collision with the vertical travel limits that is the ultimate performance problem, not rattlespace movement itself. By penalising relatively small movement within the rattlespace, the quadratic measure of equation 3.4 does not properly represent the tracking performance goal.

Let us make a first attempt to remedy the situation by applying no penalty at all for travel inside the rattlespace. Rather than weight travel in proportion to the square of stroke, \( s^2 \), it may then be better to weight travel inside the rattlespace as zero and to give a high weighting outside (the theoretical consequences of considering motion outside the rattlespace limits are discussed below). Suppose a weighting of \( P \) is assigned for travel outside the rattlespace, but no weighting for travel inside, as in equation 3.5 below.

\[
\phi(s) = \begin{cases} 
0 & \text{if } |s| < m, \\
0 & \text{if } |s| \geq m.
\end{cases}
\]

**Equation 3.5**

The cost function for measuring the capacity of the suspension to stay within the vertical travel limits could then be found by integrating this penalty function over time:

\[
J_R = \int_0^T \phi(s(t)) dt.
\]

**Equation 3.6**

This measure is then in proportion to the time spent outside the rattlespace; minimizing this measure is exactly the same as minimizing the time that is spent outside the vertical travel limits.

Minor modifications to equation 3.5 assist in the optimization process. In the numerical experiments performed for this thesis, the penalty was modified by adding a small cost as the suspension approaches the vertical travel limits. Extra weighting has also been added in proportion to the distance travelled beyond the rattlespace. This is suggested by analogous experiments with linear programming problems, showing that weightings on the distance from feasible solutions help to improve convergence (Gen and Cheng, 1996). The modified penalty function used in the experiments is given by equation 3.7.
\[ \varphi(s) = \begin{cases} 
0 & \text{if } |s| \leq m_1, \\
\left( \frac{|s| - m_1}{m_2 - m_1} \right)^3 & \text{if } m_1 < |s| \leq m_2, \\
P_2 + (|s| - m_2)T & \text{if } |s| > m_2. 
\end{cases} \]

Equation 3.7

A graph of an example of this function is shown in figure 3.3. A very similar function, called a “nonlinear filter” function, was used by Lin and Kanellakopoulos (1997, p51), although in their experiments the function was used for the control of a band-pass filter rather than for measuring performance. This penalty is integrated exactly as in equation 3.6, and this is the performance measure used in this author’s numerical experiments.

![Figure 3.3 Example of Cost Function Penalty for Suspension Travel](image)

If the road surfaces that are used in evolutionary algorithms are too smooth, there is a danger that the evolved suspension will be unable to deal with large bumps. An evolutionary algorithm will quickly converge on a suspension which is too soft to be of practical use. The suspension can remain perfectly flat, as with the flat control, and achieve a perfect comfort score. Thus, EA training data should contain road bumps which can test the suspension’s capacity to avoid hitting against the rattlespace limits: “topping” or “bottoming” (Lord, 2006). To do this, some of the bumps used in the process of evolution should be at least half the rattlespace width in height.
To date there are not many modellers who have used real road profiles in such a design process (Yu et al., 2006, p1) but even real road profiles, gathered from physical experimentation, section 2.8, should include such large bumps. Of course, at the other extreme, too many large bumps in the training data will produce extremely hard suspension systems. A compromise is needed. The training data used in the numerical experiments for this thesis erred perhaps on the side of rough roads for training data, in order to more thoroughly test the capacity of a system to be capable of being smooth while at the same time handling rough terrain.

All the performance measures of tracking discussed above use the general form of an integral of a penalty function, including the quadratic measure which uses the square as the penalty. In all cases the penalty function includes a weighting for movement outside the rattlespace.

But real suspensions typically do not crash through the rattlespace travel limits, at least not without causing fatal physical damage to the vehicle. Viewed in this way, the entire basis of the model seems unrealistic, and all the above performance measures seem to be dependent on an unrealistic model.

This problem arose during the PhD research, and it was not immediately clear to this author how the problem should be resolved. The unreality of models that smash through the vertical travel limits would seem to indicate that they should be abandoned in favour of realistic models that include bump stops: especially when the major benefit of numerical optimization is that it easily includes more realistic modelling. Furthermore, there seems to be the potential for a more elegant, “unified” performance measure that includes both comfort and tracking, as explained below. The pivotal step in resolving the issue requires taking a fresh look at the “realism” of the model, especially the matter of how real drivers behave when approaching large bumps.

To begin with, suppose an attempt is made to employ a model that includes very stiff bump stops near the ends of the suspension travel. As discussed, this could appear to solve the problem of travel outside the rattlespace. Travel is limited to the rattlespace and severe bumps with the rattlespace ends are penalised heavily because of the discomfort they cause. These bumps apparently make the model more realistic, avoiding the embarrassing matter of modelling “infeasible” suspension travel beyond the rattlespace limits. Furthermore,
optimization would seem to avoid such collisions since they cause extreme discomfort, albeit for extremely short periods of time.

This can seem to be a very tidy “unified” solution since the tracking problem is included in the goal of comfort: the one measure of discomfort doubles as a travel measure since it includes the great discomfort that results when the chassis hits up hard against the travel limits. The discomfort of hitting the rattlespace limits then also acts also as a performance indicator for suspension vertical travel. To reiterate, this seems like an elegant solution that does not require unrealistic movement outside the rattlespace.

However, when using this method, optimization depends critically on how hard the suspension hits against the vertical travel limits, creating instability in the optimization process. The exact trajectory of the rattlespace edge collisions has a critical effect on optimization, as does the method of measuring discomfort. Even if the collisions are very violent and uncomfortable, they may be so quick that they will not create a large impact when integrated over time. With only brief albeit very hard collisions with the vertical travel limits a very soft suspension that very frequently collides with the vertical travel limits may predominate. On the other hand, if discomfort is measured as maximum jerk then optimization will avoid any topping or bottoming at all, and the suspension that is optimal will be very hard. The compromise between comfort and tracking is not a true unified measure and the effect on optimization will almost certainly be skewed.

Most importantly, however, the premise behind the “unified” measure is entirely flawed; the model is still unrealistic in that it assumes that real drivers attack large bumps at constant speed. What seems like a more realistic model is highly unrealistic, in ways that make the method unworkable.

In the real world, when drivers approach a section of road that is liable to cause rattlespace collisions they will generally slow down. Real drivers are inconvenienced or frustrated by suspensions that are likely to hit unnecessarily and violently against rattlespace travel limits, and it is this inconvenience which is the major negative component of the tracking performance.
Other factors affect slowdown, such as momentary changes in traction, especially when accelerating, braking or cornering. Note that traction has little effect at constant straight-line velocity. Unpredictable movements that may lead to loss of control in other ways, such as rollover, also cause drivers to slow down. Furthermore, just the threat of severe jolts from road bumps can cause a driver to slow down, even if the bumps are not large enough to threaten rattlespace collisions.

Thus real drivers generally slow down when approaching large bumps, changing the road height versus time profile encountered by the vehicle. The assumption of constant forward vehicle speed in the model is unrealistic. Driver slowdown is likely to occur when the driver judges either that the bump will produce an extreme and sudden change in suspension force or, even worse, that an impact with a rattlespace limit is likely or imminent. Resonance with the tyre over rutted roads can be complex, but generally it also produces slowdown.

This kind of driver behaviour can be directly observed at speed bumps. Drivers will usually slow down when approaching a speed bump. Of course, this is the very purpose of speed bumps. All previously discussed models of suspension systems do not model the driver’s slowing down for large bumps, and in this sense the models are unrealistic.

Most speed bumps are roughly an inverted circular section in profile, producing an enormous spike in jerk at the points of “slope discontinuity” (refer to section 4.3). Undoubtedly, discomfort at sudden jerks on the chassis, as well as the potential mechanical damage they threaten, are a contributing factor to driver slowdown. Nonetheless, whether in the form of a deliberately designed speed bump or not, large bumps, and especially large bumps which threaten rattlespace collision will certainly cause a careful driver to slow down.

Driver slowdown alters the road profile as a function of time, reducing the rate of change of road height (and other derivatives of road height) in proportion to the vehicle’s forward speed. A slower change in road height allows a suspension more time to avoid hitting rattlespace limits, and it generally improves stability. The resulting change in road height profile constitutes a considerable change to the model.

The less the driver needs to slow down for bumps the better is the suspension tracking, all else being equal. If driver slowdown were included in the model, the goal of maintaining the
suspension within the rattlespace limits could be measured as the amount that the driver is forced to slow down. Suppose the preferred speed of the car is $V$, but the actual speed with driver slowdown is $v(t)$, (which can be assumed to be lower than $V$). The tracking objective function could then use a least squares measure taking into account forward velocity, as shown in equation 3.8.

$$J_R = \int_0^T (V - v(t))^2 dt.$$ 

Equation 3.8

This measure assumes that the target speed for the vehicles forward velocity is $V$ and that the driver does not travel above this speed. The measure of equation 3.8 therefore penalise the amount that the driver has to slow down below the target speed $V$ in order to maintain a smooth ride. The less the driver has to slow down, the less the diver’s frustration at the suspension’s inability to track the surface.

Can driver slowdown then be faithfully modelled and employed in computer optimization? Different drivers attack speed bumps at different speeds, and some seem to have more regard for the health of their vehicles than others. At first sight, modelling driver slowdown would seem to require complex and extensive empirical psychological testing.

Nonetheless, it may be possible to produce a simple numerical model that reasonably approximates driver slow-down behaviour. Suppose a model is altered to slow down whenever the rattlespace limit is crossed (or even when instability or loss of traction arises). It should be possible to develop an iterative algorithm that would slow down to just the point that the suspension avoids hitting the rattlespace edges, or the forces or jerks generated reach a given limit. Drivers learn by experience how to avoid hitting against suspension limits over large bumps, and they develop a good instinctive knowledge of their vehicles, and it seems likely that optimal slow-down behaviour will be approximated by a good driver who is familiar with their vehicle. With this assumption it becomes possible to model at least some aspects of driver slowdown. In fact, to help drivers to acquire a sense of the vehicle’s travel limits it might be helpful if a warning light could be included in the dash to warn the driver that they are travelling dangerously close to the rattlespace limits. The assumption that the driver knows the limits of the vehicle could also be used to model slowdown as a means of
avoiding large jerks to the chassis, by again using slowdown to limit jerk to a maximum value.

Unfortunately, the use of this method requires a great deal of experimentation of its own. Given that this thesis already deals in novel ways with jerk as a comfort measure, the author decided to stay within the bounds of a relatively conservative movement away from standard measures. What is more, the development of the method would have greatly delayed the research. Thus driver slowdown is not included in the numerical experiments described below.

Nonetheless, driver slowdown is extremely important theoretically. Most importantly, it explains how the use of unrealistic models that incorporate travel beyond the rattlespace must be balanced against the unreality of models that have only constant forward velocity.

It should be said in this context that any reasonably smooth suspension, no matter how high performing, will collide with rattlespace limits if driven fast enough over a rough road. The only way to completely eliminate rattlespace collision is to have an extremely stiff and extremely uncomfortable suspension: virtually the equivalent of no suspension at all. Thus, without driver slowdown, some high-performing suspensions will collide with vertical travel limits. Without driver slowdown, some excursion beyond the rattlespace limits, with performance penalties, will produce more realistic optimization results than simply including models of excessive jolts against the travel limits.

While driver slowdown has not been modelled or used as a performance measure in this thesis, it is needed in order to comprehensively describe the alternatives. It is needed as theoretical background to explain the seeming contradictions described above, to provide a theoretical background for the relative benefits of alternative tracking performance measures, and to explain the compromise measure used in this thesis: a nonlinear measure with high penalty for travel outside the rattlespace.
Table 3 Summary of Possible Techniques for Measuring Tracking

Table 3 summarises the various alternatives for tracking performance measures that have been alluded to in the above discussion. There are three ways of visualizing the tracking performance:

1) Minimize displacement around the equilibrium by using various weighting functions, including RMS and non-quadratic measures that severely penalize travel outside the rattlespace.

2) Model the suspension hitting hard against bump stops at the vertical suspension travel limits, and use comfort as the single performance measure.

3) Minimize frustration by the driver slowing down to avoid hitting hard against the rattlespace limits and combine with a comfort measure.

Note that 1) covers the first two rows of the table, 2) corresponds with the third row, and 3) covers the final row. The first row represents a special case of the second, but it is included because of its special place in engineering practice.

RMS or least squares measures produce tractable solutions for both LQR control and for least square estimators, when applied to linear systems. And they have a proven track record. They
have proven successful in LQR systems and least square estimators. These measures employ a penalty in proportion to the square of the difference.

“Nonlinear” penalties (more properly non-quadratic measures) can also be used. Thus suspension travel very close to or beyond the rattlespace could be penalized very heavily, while travel within say the middle third of the rattlespace might not be penalized at all. As explained above, a nonlinear penalty that heavily penalises travel outside the rattlespace can be readily adapted for numerical techniques, as used here.

The method of modelling bump stop collisions, option 2) above, can seem at first attractive. Tracking is penalized by the discomfort of hitting hard against the vertical travel limits, and providing one combined measure for both comfort and tracking. With such a model a “rattlespace collision” is a very hard collision with a very stiff bump stop at one end of the rattlespace. There is a “collision” but there is not an “intersection” with the rattlespace: the chassis does not smash through the ends of the suspension travel, destroying the suspension. This method seems more realistic, and it also seems to offer the possibility of a more elegant, “unified” measure of comfort and tracking.

Ultimately however, the logic that the model is more realistic is fundamentally flawed because of driver slowdown. It is not necessary to perform the delicate balancing act of measuring the discomfort of such rattlespace collisions because they rarely occur in the real world, and the method would seem to produce more instability in optimisation than a method that merely penalises travel near the vertical suspension travel limits. It is a reasonable assumption that a careful driver will usually avoid discomfort and vehicle damage by slowing down for large bumps. (Collisions with the rattlespace limits do occur, but they are the exception that proves the rule.)

Models of collisions with stiff bump stops are just as unrealistic as models in which the chassis travels outside the rattlespace, because of driver slowdown. The modelling of severe bump stop collisions is ultimately unrealistic, except perhaps for reckless driving or driving in emergencies.

HMMWV experimentation employed a measure very like the combined measure anticipated here (Donahue, 2001): the Cumulative Absorbed Power (see section 2.9.3). For a military
vehicle travelling quickly over rough terrain, where comfort is very much secondary; the Cumulative Absorbed Power measure has some justification. For delicate optimization processes for passenger vehicles balancing comfort with tracking, the measure is inappropriate. The “unified” measure is unrealistic and will be biased towards either very soft or very rough suspensions, depending on the discomfort caused by large jolts over very short periods of time. In the end, if realistic bump stops are to be included in the model, then realistic models of driver slowdown are also necessary.

The nonlinear, non-quadratic performance measure for suspension stroke using equation 3.7 was used as the tracking performance measure for this thesis. It represents a compromise. It departs minimally from proven engineering tradition. As a compromise for the lack of a model of driver slowdown, it allows travel outside these limits. The measure did not produce noticeable artefacts or instability in the optimization process. Driver slow-down may be superior theoretically, but the method remains unproven and may have its own artefacts or instabilities depending on how driver slowdown is modelled. However, a more cautious approach was used in this thesis. Given that the use of jerk as a comfort measure is somewhat nonstandard (although it becoming more widely used), and given the extra experimentation needed for modelling slowdown, the tracking measure used in this thesis simply applied a minor variation on the quadratic measure by heavily penalizing travel outside the rattlespace: the second row in table 3. In optimization this has the effects of finding the most comfortable method possible that at least heavily penalizes travel time spent outside the rattlespace.
4. Suspension Control

It is not an objective of this Chapter to present the complete overview of suspension control theory. The focus will be only on the areas relevant to the scope of this research, particularly the theory applied in experiments (numerical or physical) and especially those aspects of theory which are original in this thesis. Many elements of suspension control theory have been covered in the literature review (see chapter 2), and the corresponding sections of the literature review will be referred to as needed.

(Various pieces of Java computer program code have been developed using the Eclipse development environment which is freely available on the web. The workspace used is PhD\Eclipse RSpace Constraint\Edge Overshoot in the PhD directory. In Eclipse, press File|Switch Workspace and browse to find the workspace.) A number of demonstration programs are included in this code. The ExampleFileFilter class has been copyrighted by Sun (refer to code for details). All remaining code here has been written by this author, including all the graphing software in the FunctionGraph package, and code for maintaining persistent parameters using text fields and sliders in the SliderGroupControl package.

Note that the SuspensionTest program is not run in Eclipse (for reasons that are made apparent in chapter 5). This program was developed using a text editor and it was compiled and run using instructions issued by hand in a command or “DOS window”. This code is explained in detail in chapter 5.

In many of the examples in this section, for instance in figure 4.1 and figure 4.2, dimensions of length are not indicated. Meters or feet, or any desired dimensions could be used. This is often the practice in the literature when discussing control theory, as opposed to actual experimental results. On the other hand, seconds are assumed for the time dimension unless otherwise stated. This is simply for convenience.
4.1. Introduction

Knothe and Bohm refer to experiments performed in 1931 into vibrations in steering systems (1999, p311) as well as many experiments in the 1950s and 60s into railway and road vehicle suspension stability. Very early theoretical investigations into road vehicle suspension systems were also carried out by aeronautical engineers (Knothe and Bohm, 1999, p305). Of course, racing car suspension development too has had an enormous impact on road vehicle suspension theory (Milliken and Milliken, 1995).

As discussed in section 2.10, active suspensions can potentially produce any force desired and so are much more flexible than traditional passive controls. This great flexibility creates new problems not observed in passive suspensions. Indeed force discontinuities can be observed in the Crosby and Karnopp’s model of the “active damper” control (1973) as shown in figure 4.1 (Crosby and Karnopp, 1973, p125).

![Figure 4.1 Force Discontinuities in the “Active Damper”](Crosby and Karnopp, 1973, p125)

In 1978, Smith et al. published the results of a quite extensive study of performance indexes of riding comfort involving 18 roads and 78 passengers (1978). Smith et al. recommended RMS acceleration of chassis movements as a predictor of ride comfort. In 1988, Redfield and Karnopp analysed suspension performance using three separate indexes, producing what today would be recognized as a form of Pareto optimisation (1988). In their analysis,
acceleration was used as the performance measure of comfort, and indeed is still the most applied measure of passenger comfort.

Despite the distinguished pedigree of RMS acceleration, for reasons argued earlier in this thesis, especially in section 3.1, acceleration alone is not a viable measure of passenger comfort. In earlier times, when all suspension systems were passive, suspension force discontinuities resulted mainly from road surface discontinuities. Passive suspensions do not generally generate sudden force changes whereas active controls can produce any desired force at any time, and semi-active systems can suddenly change damper stiffness at high stroke velocity. Indeed, such sudden force changes have been observed in both numerical and physical experiments into both active and semi-active suspensions (Ivers and Miller, 1991; McLellan, 1998; Crosby and Karnopp, 1973; Ahmadian et al., 2004).

Section 3.1 covers the reasons in detail that jerk is used in the experiments here as the measure of comfort. Note that if jerk is kept low then, necessarily, so is acceleration, but low accelerations can be accompanied by very high jerks, as demonstrated in section 3.1.

With the advent of active and semi-active control comes the possibility of the use of a wide variety of controls, and a large number of basic control methods have received at least some attention in the literature. This includes methods based on the linear quadratic resonator, sliding-mode control and skyhook control, all discussed in the literature review in chapter 2. As explained below, the LQR control is widely used. The linear “skyhook” and a number of variations that are referred to as “skyhook” controls are also widely used (Karnopp, 1995; Burton, 1993; Paddison et al., 1994; Reichert, 1997; Elbeheiry and Karnopp, 1996; McLellan, 1998; Wagner and Liu, 2000; Goncalves and Ahmadian, 2002; Ahmadian et al., 2004; Donahue, 2001; Song et al., 2003; Song and Ahmadian, 2004; Stembridge et al., 2006; Li et al., 2004; Williams and Best, 1994; Hyvärinen, 2004; Krüger, 2002; Caponetto et al., 2003; Guglielmino et al., 2008, p70). The term “skyhook control” has become quite broad, covering almost any control that uses absolute chassis height or chassis height velocity as a control parameter. (In this thesis, in order to clarify the distinction, the term “linear skyhook” is used to describe the linear skyhook with a linear “virtual” damper attached to the “sky” as described in section 2.4.)
It should be noted that the linear skyhook is not provably optimal in any sense although it has very pleasing transmissibility, as shown in section 2.4, and the “damper” can be shown to absorb chassis kinetic energy (in the vertical direction) at all times as discussed below in section 4.2.

The LQR is a popular basis for control which is truly optimal, but it is optimal for a very specific performance measure: quadratic measure (or nearly equivalently, RMS measure). Above (in section 2.3.1) the single DOF control has been optimized for RMS displacement and velocity, producing a linear control, with a damping coefficient of approximately 0.7, which is much greater than actual damping rates used in practice in modern vehicles of about 0.25 (Milliken and Milliken, 1995). In section 4.5.1 below, the LQR control is adapted to the minimisation of jerk to produce a linear control over jerk.

Only a very limited range of highly idealized physical problems are truly optimized by analytical, mathematical techniques. For example, as discussed in section 2.3.1, it is stated that “the Linear Quadratic Regulator (LQR) has been used as one of the main control techniques for dealing with active suspension design” (Camino et al., 1999, p3168). The LQR technique addresses the problem of optimizing a quadratic performance for a linear system. But real-world systems are not linear. Furthermore, quadratic performance measures may be convenient, and they may have a respectable pedigree, but this is no guarantee that they represent the “true” performance goal.

For any given real-world problem, it should be asked if the idealized solution to the idealized problem is sufficient for the task at hand. Analytical optimisation techniques are important and useful, but they only work for a very small set of idealised optimised criteria. As MacCluer nicely demonstrates, optimisation using different performance costs can produce wildly different controls, and he concludes that,

“The application of optimal control to practical problems is an art, requiring the practitioner to perform many analytic and numerical iterations to reach an acceptable (but often not optimal) solution to the original problem.” (2005, p113)

When more flexible numerical techniques become available it is possible to employ more flexible models and more realistic performance measures, even though these do not admit of a neat mathematical solution.
As discussed above in chapter 3 the performance criteria used throughout the experiments here are relatively non-standard. While quite a few researchers have used jerk as a performance goal of one form or another (Hrovat and Hubbard, 1981; Hrovat and Hubbard, 1987; Paddison et al., 1994; Hashiyama et al., 1995; Ahmadian and Vahdati, 2003; Ahmadian et al., 2004; Ahmadian and Vahdati, 2006; Yamakado and Abe, 2006) acceleration is still preferred as a measure of ride comfort. Furthermore, to the best of this author’s knowledge there have been no researchers into suspension who have looked at the rattlespace as a constraint, as done below in section 4.7, or who have even used nonlinear performance measures of vertical travel dependent on vertical suspension travel limits.

Over recent decades, computational capacity has increased enormously and various techniques have arisen for performing large-scale numerical analyses to attempt to improve the performance of engineered systems: fuzzy logic, neural networks, various forms of evolutionary algorithms and others. These techniques are proving immensely useful to the art of engineering design.

In this paper numerical techniques are being applied only to first-order suspension controls, to circumvent the inadequacies of analytical techniques. Evolutionary algorithms in particular are used to compare the performance of very simple, first-order suspension controls. It is important to stress the difference between this and the application of computing power to develop a specific engineering design. In industrial use, the engineer typically produces a very detailed model in an attempt to iteratively improve a detailed design. This is known as “virtual prototyping” (Hyvärinen, 2004, p45) and it is extremely important for modern research and development.

Here, however, numerical techniques are simply used to compare the effectiveness of various high-level first-order algorithms such as linear controls, virtual bump stops, skyhook controls, various controls over jerk and an entirely new kind of control here termed rattlespace constraint controls. Another reason for a first-order investigation is that performance measures are to some extent the focus of this thesis. Jerk is used as a comfort measure while the majority of researchers are using acceleration, and this thesis also uses a nonlinear measure for suspension tracking which has never been applied before.
The use of flexible controls, including controls over jerk, is made possible by the fact that electronic controls can switch quickly between different levels of force, and yet such force discontinuities are not penalised by the more often used RMS acceleration measure of comfort, as discussed in section 3.1. There are also a number of reasons why electronic controls may be prone to such extreme jerkiness (discussed in section 3.1) compared to passive suspension systems.

Numerical techniques open up the possibility of testing a much broader range of control algorithms. Instead of finding an analytical optimum, the numerical method finds a suboptimal value. This is done while running the simulations over stochastic input (random road surfaces) and any desired performance criteria can be applied. The numerical method is not restricted to a very limited range of tractable idealistic problems that cope very badly with stochastic input.

In this thesis the use of fuzzy sets has not been included. While fuzzy sets have been widely used in the context of optimization with EAs, experiments with simple linear sets in one dimension revealed little beyond the tendency of a property to strengthen or weaken with the independent parameter. A simpler way to judge such changes is to use a simple sigmoid function, as explained below in section 4.5.2.1. Sigmoid functions evolve much more quickly than fuzzy functions. Furthermore, the approach of using fuzzy sets is extremely general. “Setting the performance index is the only procedure for the designer of the controllers” (Hashiyama et al., 1995, p166). Fuzzy sets are so general that they can hide the details of how they work. It was felt that in this thesis, which aims at a theoretical understanding, the contribution of fuzzy sets is questionable.

There are two small subsections at the beginning of this chapter, sections 4.2 and 4.3 that deal briefly with a couple of particular matters needing qualification and clarification. The first is energy dissipation in the semi-active suspension. It has been claimed that “since the semiactive damper does not add any energy into the system, the system is stable” (Song et al., 2003, p227) but it is shown in section 4.2 that this notion at least needs qualification. These qualifications, to the best of this author’s knowledge are original. Section 4.3 deals with discontinuities in road surfaces, in particular sections of road where road height or road slope changes suddenly. Some simple observations regarding a suspension’s response to such surfaces provide useful insights, for the engineer, into the differences between various
suspension controls and these are noteworthy enough to be included in this thesis. The work of that section is entirely original.

The sections following section 4.3 deal in more detail with the controls used in numerical or physical experiments (not all controls used in the numerical experiments were applied to the physical test rig). These sections deal with a variety of algorithms that can be used wholly, or in part, in a suspension control system.

The controls advanced in this chapter come from a variety of sources. Some are derived directly from linear theory, while others are simple heuristics that derive from modifications of linear controls. This includes “virtual bump stops” that “stiffen” the suspension when it is approaching the limits of the suspension travel. The use of virtual bump stops and some of the other modifications are original, as far as the author is aware.

A new category of controls, entirely this author’s original work, called “rattlespace constraint” controls are introduced here in section 4.7. A search has not revealed any controls of this type in the literature. These controls look at the suspension tracking problem as one of constraint (refer to figure 4.17 below). It seems almost trivial to say that a suspension is constrained within the rattlespace, and that it should be free to move smoothly within this constraint, and yet the suspension system problem is never presented this way in the literature.

The simplest idealised mathematical form of the rattlespace constraint problem that would be a constraint problem:

\[
\text{Maximise ride smoothness (comfort),} \\
\text{where suspension stroke, } s, \text{ is constrained by the rattlespace,} \\
|s| < R, \\
\text{and } R \text{ is defined as half the width of the rattlespace.}
\]

The use of half the width of the rattlespace is mathematically convenient because of symmetry but it contains a hidden assumption: that the equilibrium position of the suspension will be in the centre of the rattlespace. This cannot of course be assumed and in fact, the
equilibrium position of a fully loaded vehicle will generally be different from an unloaded
one. This matter is discussed below in the Further Investigations, in section 7.5.

Rattlespace constraint controls address this problem directly by targeting the ends of the
rattlespace and remaining within it. It is worth repeating that, “the application of optimal
control to practical problems is an art” (MacCluer, 2005, p113), and it is important not to
naively leap to the conclusion that idealized solutions to this or any other mathematical
problem will immediately be practical. Experiments, as always, are required. However, this
method is quite intuitive and the techniques developed in this chapter should at least form a
solid base for possible further investigation.

Some techniques used below are derived from the consideration of the minimum-time
problem, discussed in the literature review in section 2.3.2.3. As discussed there, minimum-
time problems are generally resolved using Pontryagin’s Principle (MacCluer, 2005; Smith,
1998; Hermes and LaSalle, 1969; Kirk, 1970). To the best of this author’s knowledge,
minimum-time controls have never in any form been proposed before for suspension control
and the work here is entirely original. Minimum-time control over acceleration is a popular
text-book example of control as discussed in the literature review (section 2.3.2.3) but it
would be extremely uncomfortable if applied to suspension control. The minimum-time
control over jerk, on the other hand, is very subtle. The minimum-time control with a given
initial displacement and zero velocity and acceleration is quite simple, but finding the control
with any initial displacement, velocity and acceleration proved quite elusive. The author
spent some time searching for this control. Finding the control conclusively and the proof of
the control occurred almost simultaneously.

The optimality of this control is proved by a relatively simple extension of the mathematical
proof of the minimum-time control over acceleration as found in MacCluer (2005). This
proof is given below in section 4.6. A search uncovered experiments using minimum-time
control for application to the movement of industrial robot arms by Koh et al. (1999) and
similar experiments by others applied in robotics (Macfarlane and Croft, 2003; Ben-Itzhak
and Karniel, 2008), however, the proof given below has not been presented before to the best
of this author’s knowledge and neither has the alternative method given in the appendix,
section 8.7, based on a slightly different but equivalent approach from Hermes and LaSalle
In the development of some algorithms for rattlespace constraint controls, there arose a need for a continuous control over jerk which could smoothly reach equilibrium without overshoot (refer to figure 4.2). An algorithm has been found to achieve this with a continuous acceleration, using constrained jerk. Furthermore, an iterative algorithm has been found to achieve this. This method could have independent interest in other applications. All this is original work to the best of this author’s knowledge.

![Figure 4.2 Achieving Rest – Left: with Overshoot, Right: without Overshoot](image)

A theory and a method were developed for removing jerk (force discontinuity) from a semi-active suspension (see section 4.8). Semi-active suspensions are prone to force discontinuities when they attempt to follow a given active control, because of the passivity constraint, as explained below. This work follows from the work of Ahmadian et al. who developed a global control for the removal of jerk (2004). In papers written in the course of this thesis, methods have been proposed for locally dealing with the passivity constraint (Storey et al., 2006; Storey et al., 2008), and the discussion of section 4.8 represents a maturation of this approach. This work, insofar as it diverges from the method of Ahmadian et al. is unique.
4.2. Energy Dissipation in Semi-Active Suspensions

In introducing active and semi-active systems, many researchers will often make statements about the relationship between energy dissipation and a suspension’s stability. This brief section takes a careful look at energy dissipation in the semi-active suspension.

While the worst responses of linear systems are generally due to resonance, a nonlinear system might exhibit aberrant or dangerous behaviour for unknown reasons. Active systems thus may have “inherent stability problems” (Elbeheiry and Karnopp, 1996, p548). During development, they may exhibit idiosyncratic side effects, such as the effect observed by Williams and Best in which their oleo-pneumatic active system maintained movement with no disturbance; the “open loop roll response proved to be quite interesting, as both measured and theoretical results show resonances at 3.5 Hz and 4.5 Hz when the vehicle is stationary” (1994, p342).

Semi-active systems however are often regarded as inherently stable; “since the semiactive damper does not add any energy into the system, the system is stable” (Song et al., 2003, p227). Dyke et al. claim that, “according to presently accepted definitions, a semi-active control device is one which cannot input energy into the controlled system” (1996, p565). It is also claimed that “semiactive dampers do not add any energy to the system – they only dissipate energy (the same as a passive damper)” (Ahmadian and Simon, 2002, p123).

Such statements can be misleading and at least need qualification. Under some conditions the semi-active damper can add to the kinetic energy of a vehicle’s vertical motion. This is true too of a passive system, but an extremely bad semi-active suspension control could conceivably perform much worse than the passive, as explained below.

A simple analysis of the energy dissipation in a linear system is sufficient to show this. The energy comprises principally of gravitational potential energy, chassis kinetic energy and spring potential energy,

\[ E = mgr + \frac{1}{2} my^2 + \frac{1}{2} ks^2. \]

Equation 4.1
(Refer to section 2.2 for a description of the variables used. The kinetic energy of the forward motion of the vehicle is ignored here as this is assumed constant.)

Power is derived by differentiating this equation with respect to time. After inserting the formula for acceleration (equation 2.1) the power is given as,

\[
P = \dot{E} = mg\dot{r} + k\dot{s} + m\ddot{y}
\]

\[
= mg\dot{r} + k\dot{s} + m\ddot{y} - \frac{c\dot{s} - ks}{m}
\]

\[
= mg\dot{r} + k\dot{s}(\dot{s} - \dot{y}) - c\ddot{y}
\]

**Equation 4.2**

The damping rate is positive, \( c > 0 \), and so the term involving the damping rate, \(-c\ddot{y}\), contributes to energy absorption only when the stroke velocity and the chassis velocity are in the same direction, \( \dot{s}\dot{y} > 0 \).

**Equation 4.3**

However, when they are in opposite directions, as will occur periodically over rough terrain, the damper in a passive system actually contributes energy to the system. It seems counterintuitive that a dissipative element is actually increasing system energy, but because the damper is not tethered to a stationary point, there are conditions under which the damper force increases the chassis velocity. When the damping force, \(-c\dot{s}\), acts in the same direction as the chassis vertical velocity, \( \dot{y} \), the damper actually pulls the chassis with it, thus increasing the kinetic energy of the chassis.

In fact a semi-active system could be deliberately designed to switch the damper on during only these periods, making it much more dangerous than a conventional passive system. Roughly speaking, a deliberately very badly designed semi-active suspension could be made to “ratchet kinetic energy upwards”, as the suspension uses road disturbances to increase the energy in the system. Such a system would hardly be a candidate for a viable suspension control and this is an artificial scenario, but the very possibility is a counter example disproving the claim that semi-active systems act only to dissipate energy in the system. Of
course, as a broad rule of thumb any reasonably viable semi-active system will almost certainly be safer even than the passive. The point is that the damper in a semi-active system can add energy to the system, and energy absorption cannot be said to be an inherent physical property of semi-active systems. In fact, as a general principle, it is misleading to say that untethered dampers are “dissipative”.

The calculation shown above can be made much simpler. The power absorption of the damper could be more readily calculated using the simple formula of force times velocity (Meirovitch, 1985, p78),

\[ P = F_d \dot{y} \]
\[ = -c\dot{y}^2. \]

In this expression, damper power is sometimes positive as explained above, putting energy into the system. Nonetheless, the slightly more comprehensive analysis is perhaps justified given the often made assumption that the damper is entirely dissipative.

Energy dissipation has been used specifically as a key component of suspension design logic by Johnson and Erkus. “If the dissipativity constraint can be imposed in the design of the primary controller, it will produce predominantly dissipative forces, making semiactive systems more efficient. As a result, semiactive control strategies will be applicable for a wider range of problems.” (2002, p2463). This is perhaps a promising line of enquiring for suspensions in which energy dissipation is a much greater priority than comfort, as in seismic applications, but their approach has not yet directly produced any noteworthy consequences for vehicle suspensions.

The purely linear, active skyhook has interesting energy absorption properties. In the skyhook suspension the damper’s force is proportional to chassis height velocity, and the force acting on the chassis is,

\[ F = -ks - cy. \]

Thus the acceleration of the chassis is,

\[ y = \frac{-ks - cy}{m}. \]

Performing a similar differentiation to the one in equation 4.2 the equation for power in a skyhook system is found to be,
Thus the skyhook has the pleasing property that the damper always acts to decrease energy in the system, since the term involving the damping rate is always negative, $-cy^2$.

This should not be interpreted to mean, however, that the skyhook is therefore optimal in absorbing energy, as discussed in the following paragraph. The pure skyhook is a virtual control, and the “virtual spring” (see section 2.7) produces power in some parts of the suspension’s swing. Nonetheless, this may be another factor which helps to explain the enduring qualities of the skyhook algorithm.

The on-off skyhook control (Ahmadian et al., 2004, p4; Hashiyama et al., 1995, p168; Savaresi et al., 2003, p2265; Simon and Ahmadian, 2001; McLellan, 1998; Guglielmino et al., 2008, p70) given by equation 2.10, is the optimal control for removing vertical kinetic energy from a semi-active suspension when the spring is the actual spring, because it turns the damper off in precisely the condition that it would otherwise contribute energy to the chassis, when equation 4.3 does not hold, and it is full on otherwise. It uses the damper to absorb power at its maximum possible rate. The problem is that the control is very jerky and highly uncomfortable (as discussed in section 2.7). Nonetheless, it may form the foundation for the control of a seismic suspension under extreme conditions where energy absorption is the primary goal.

### 4.3. Road Discontinuity

When a passive suspension traverses a step discontinuity in the road surface, as in figure 4.3, it produces an extremely high force for a very brief period of time. In the “step discontinuity” on the left in figure 4.3 the spring experiences a sudden change in force, and an accompanying high jerk. The damper force also has an extremely high spike in force.

Of course, a discontinuity is a mathematical abstraction; road surfaces are not truly discontinuous. In reality, an infinitesimal spike, or more technically a Dirac delta function (Meirovitch, 1985), is approximated by a large but finite spike. Nonetheless, even quite small
bumps that approximate a discontinuity produce great discomfort in road vehicles, easily verified in the average passenger car. Despite the mathematical artefacts, the theoretical response of a suspension control to such “discontinuities” is highly revealing and is worthy of a brief discussion.

Figure 4.3 Left: Step Discontinuity – Right: Slope Discontinuity

In a purely linear passive suspension the spring does not experience a sudden change in force when traversing a discontinuity in road slope, as in the diagram on the right in figure 4.3, but the damper does. The damper produces a sudden change in acceleration due to the sudden change in extension velocity. Again, the damper contributes most to the worst part of the suspension response. Perhaps this is another reason that practical passive suspensions have such a low damping ratio (as discussed in section 2.3.1). Certainly, this helps to explain why the skyhook may be superior to the passive suspension.

Note again that the RMS acceleration measure of ride discomfort is unaffected by a force discontinuity between two moderate forces, as explained in section 3.1, and yet abrupt force changes are demonstrably uncomfortable. On the other hand, a jerk measure reveals the problem. The sudden change between two moderate acceleration values produces a “spike” in jerk which is read by a discomfort measure employing jerk.

In contrast to the passive suspension, the damper of the purely linear skyhook suspension does not experience a jerk spike over a slope discontinuity (because the skyhook damper responds to chassis movement and not stroke). The skyhook’s damper does not produce either a spike in either acceleration or jerk over either the step or slope discontinuities.

Table 4 summarises the responses in terms of acceleration discontinuity (jerk spike). The passive suspension has a discontinuous force over both a step discontinuity and a slope discontinuity (due to the damper) while a skyhook suspension produces a continuous force
traversing a slope discontinuity. The skyhook does not cause great discomfort over the slope discontinuity.

In this way, a first-order characterization of suspension systems can be developed that depends on whether discontinuous forces are produced by road discontinuities. The FlatLinearJerk01 control, which is a linear control over jerk, is an example of a control that theoretically has a continuous force response over both kinds of road disturbances.

<table>
<thead>
<tr>
<th>Discontinuity Type</th>
<th>Passive</th>
<th>Skyhook</th>
<th>FlatLinearJerk01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step Discontinuity</td>
<td>Discontinuous</td>
<td>Discontinuous</td>
<td>Continuous</td>
</tr>
<tr>
<td>Slope Discontinuity</td>
<td>Discontinuous</td>
<td>Continuous</td>
<td>Continuous</td>
</tr>
</tbody>
</table>

Table 4 Acceleration Continuity for Various Suspensions

This may help to explain the superiority of some controls, but it also points out another very important factor for electronically controlled suspension: latency. Electronic control is capable of a smooth force response even with road discontinuities, but the control must be able to respond very quickly. On real road surfaces, if an approximation to a discontinuity occurs over a greater period of time than the response time of the system there can be a noticeable improvement in response in comparison to passive systems.

This section has investigated a very simple non-numeric categorization for comparing suspension control algorithms. For one thing, it clearly indicates one advantage of the skyhook control over the passive. This is only a first-order characterization, but it is revealing in ways which more complex analyses may not be. This analysis should not be seen as a replacement for transmissibility and more technical analyses, but simply as an intuitive aid for the designer.

4.4. Overview of Experimental Controls

The numerical and physical experiments in the following chapters will compare a variety of suspension control algorithms. These experiments employ the performance measures discussed in section 3 to test the suspension performance, specifically the performance
measures of equation 3.3, equation 3.6 and equation 3.7. The remainder of this chapter explains the theory behind the various control algorithms used in the experiments.

In retrospect there appears to be a rough correspondence between three broad categories of control strategies discussed here and analytical solutions to three ideal optimization problems. Linear controls derive from LQ optimization, sliding-mode controls relate to minimum-time problems, and rattlespace constraint controls derive from problems of constrained suspension travel. This theoretical correlation however was not the origin of the division, rather the categories were chosen because they seemed to correspond to different design approaches.

There is also a control method developed in this chapter for reducing what is here called “crossover jerk”. This is the sudden jerk that is caused when a semi-active control is clipped by the physical limit of the passivity constraint (Yi and Song, 1999, p147; Giorgetti et al., 2006; Sergio M. Savaresi et al., 2003, p2264; Jalili, 2002, p600; Yokoyama et al., 2001; Hyvärinen, 2004, pp31-2). The passivity constraint has been discussed in section 2.6. The methods developed in this chapter are “local”. That is, they attempt to reduce crossover jerk when crossover is imminent, leaving the control open at other times for some high-performance control. Previous attempts to reduce or remove crossover jerk have applied “globally”, at all times (Reichert, 1997, p63; Ahmadian et al., 2004; Ahmadian and Vahdati, 2006). Thus while controls to remove crossover jerk are not a separate category, they can be applied to a semi-active system in combination with any other control, allowing high-performance active controls to be applied to a semi-active system without too great an impact from crossover jerk. This matter is discussed below in section 4.8.3.

A slight ambiguity can result over the use of the term “variable” in the context of evolutionary algorithms. EAs have been used to test a large number of suspensions of the one type with different properties. To give a simple example, a large number of passive suspensions with different spring rates and damper rates will be run in simulation. There are a lot of different passive suspensions tested with different spring rates. The spring rate for any one suspension is constant, but the spring rate varies between instances of passive suspensions. A passive suspension’s spring rate is constant and not “variable” for any given suspension instance, but it does “vary” between suspensions. In the context of this thesis the term “parameter” or “suspension parameter” will tend to be reserved for properties, like the spring rate, which are constant from one suspension instance to another. The term “variable”
can be safely applied to those properties that vary over time for even one particular suspension instance, such as vertical chassis height, chassis velocity, or road height. It is the parameter values which may be subject to evolutionary change.

The basic method with almost all suspensions discussed in this thesis, is to iteratively calculate a value for the acceleration (or jerk) to be applied at any instant. In the cases of a semi-active system the damper stiffness is calculated based on desired damping force. Evolutionary algorithms can be applied to set the suspension parameters to potentially produce at least a robust suboptimal performance.

### 4.5. Closed-Loop Controls

A “feedback control” is one that produces output by comparing a desired reference input against a feedback signal (Meirovitch, 1985, p302). Generally the control will become “harder” or “stiffer” as it moves “further” from equilibrium. Suspension stiffness has generally referred to force, and so for instance the damper in a passive suspension will tend to supply a larger force with larger extension velocity. In this thesis however, jerk as well as force can be controlled using feedback. These are also referred to as “closed-loop controls” or “feedback controls” as they are often modelled in block diagrams with closed loops (Dorf and Bishop, 2005, p193).

#### 4.5.1. Linear Feedback Control

The LQR technique applied to the problem of minimizing quadratic acceleration, with weighted quadratic displacement, yields the controls of equation 2.5 in section 2.3.1. In a similar way, it is possible to develop an optimal control over the third rate of change of distance (jerk),

\[ u = \ddot{y}. \]

where the cost functional now takes the form,
The equations of motion in matrix form are,

\[ \dot{x} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix} = Ax + Bu = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u = \begin{bmatrix} \dot{y} \\ \ddot{y} \\ \dddot{y} \end{bmatrix}. \]

The cost functional takes the form,

\[ J = \int_{0}^{\infty} x^T Q x + u^T R u + \int_{0}^{\infty} c y^2 + \dddot{y}^2 \ dt. \]

Solving the time invariant Riccati equation (MacCluer, 2005) as shown in more detail in section 8.2, produces the linear control,

\[ u = \left[ -\sqrt{\alpha} \ 2\sqrt[3]{\alpha} \ 2\sqrt[5]{\alpha} \right] \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} = -\beta y - 2\beta^2 \dot{y} - 2\beta \ddot{y}. \]

**Equation 4.4**

The expression has been simplified slightly by using \( \beta = \sqrt[3]{\alpha}. \)

Equivalently, using Euler-Lagrange equations (MacCluer, 2005; Smith, 1998) this problem can readily be solved directly. The problem is to minimize,

\[ J = \int_{0}^{\infty} c y^2 + \dddot{y}^2 \ dt = \int_{0}^{\infty} f(t, y, \dot{y}, \ddot{y}, \dddot{y}) \ dt, \]

where,

\[ f = c y^2 + \dddot{y}^2. \]

Solving this requires the form of the Euler-Lagrange equation that deals with second- and third-order derivatives. The Euler-Lagrange equation becomes,

\[ \frac{\partial f}{\partial y} - \frac{d}{dt} \frac{\partial f}{\partial \dot{y}} + \frac{d^2}{dt^2} \frac{\partial f}{\partial \ddot{y}} - \frac{d^3}{dt^3} \frac{\partial f}{\partial \dddot{y}} = 2c \dot{y} - 2\dddot{y} = 0. \]

Solving this directly produces the same linear control as represented in equation 4.4.
Since a critically-damped suspension in the second-order linear system occurs when the characteristic equation has multiple roots, the natural analogy in the third-order case is one in which the characteristic equation is a pure cubic,

\[ \ddot{y} = -\beta^1 y - 3\beta^2 \dot{y} - 3\beta^3. \]

**Equation 4.5**

A “critically damped” control over jerk will have coefficients in the ratios shown in equation 4.5 whereas an LQ control will have coefficients in the ratios given in equation 4.4. The LQ optimal control is somewhat “underdamped” in the sense that the coefficients of speed and acceleration are lower, given the same coefficient of distance, which is similar to what was found for the second-order case. In the example shown in figure 4.4 the system swings back beyond zero in the manner characteristic of an underdamped system (dependent on initial conditions).

![Figure 4.4 Example of an Optimum Trajectory using LQR](image)

LQR controls are optimized only with respect to the quadratic performance measure. Again, the simple fact should be stressed that they will almost certainly not be optimal using non-quadratic performance measures. When using linear control in evolutionary algorithms the linear coefficients can be determined purely by the evolutionary process itself, without appeal to the LQR, critical damping or any other theoretical derivation of parameters. The completely general version of the second-order linear control represented by equation 2.5 then becomes,

\[ u = \ddot{y} = -\beta_1 s - \beta_2 \dot{s}. \]
This is a passive system of indeterminate stiffness, where $\beta_1$ and $\beta_2$ represent the spring rate and damping rate respectively. During evolution, the spring and damper are slowly altered until an acceptable sub-optimal value is found (sub-optimal in the sense that the numerical method cannot guarantee optimality, as discussed in section 2.12). However, given that this suspension has only two parameters, the optimization process should be extremely efficient. Indeed, if other, more complex suspension controls outperform the passive suspension in the same environment, then they are certainly deserving of attention, even if the results are suboptimal. Thus the passive system makes a very good benchmark against which to compare other controls.

As shown in section 2.4, the passive suspension only responds to stroke properties, $s$ and $\dot{s}$, while the skyhook responds to the velocity of chassis height (the “sky”), $\dot{y}$:

$$u = \ddot{y} = -ks - c\dot{y}.$$  

Like the passive suspension, this has only two parameters and is a second good benchmark in experiments. It is also well represented in the literature (Karnopp, 1995; Burton, 1993; Paddison et al., 1994; Reichert, 1997; Elbeheiry and Karnopp, 1996; McLellan, 1998; Wagner and Liu, 2000; Goncalves and Ahmadian, 2002; Ahmadian et al., 2004; Donahue, 2001; Song et al., 2003; Song and Ahmadian, 2004; Stembridge et al., 2006; Li et al., 2004; Williams and Best, 1994; Hyvärinen, 2004; Krüger, 2002; Caponetto et al., 2003; Guglielmino et al., 2008, p70).

The superior transmissibility of the skyhook compared to the passive suspension has been discussed by Reichart (1997) as explained in detail in section 2.4, but evolutionary algorithms can be used to determine a suboptimal mix of linear parameters for the stroke and chassis height, $s$ and $y$, as well as their derivatives, $\dot{s}$ and $\dot{y}$. A numerical approach then is to employ general linear coefficients,

$$u = \ddot{y} = -\beta_1 y - \beta_2 \dot{y} - \beta_3 s - \beta_4 \dot{s}.$$  

**Equation 4.6**

where all coefficients, $\beta_i$, are simply determined by an evolutionary algorithm. Note the inclusion of chassis height parameters, $y$, and $\dot{y}$, as well as the passive linear parameters, $s$, and $\dot{s}$. If evolution favours a “skyhook-like” control, then it could be expected that EAs would produce relatively small values for the parameters $\beta_1$ and $\beta_2$.  

142
In the same way that equation 2.5 has been generalised to equation 4.6, so equation 4.4 can be expanded to a general linear control over jerk, \( \dddot{y} \),

\[
u = \dddot{y} = -\beta_1 y - \beta_2 \dot{y} - \beta_3 \dddot{y} - \beta_4 s - \beta_5 \dddot{s} - \beta_6 \dddot{s}.
\]

**Equation 4.7**

Again, all the coefficients are determined by evolutionary algorithms, but here the resulting control is applied to control over jerk.

Note that the LQ problem using the cost function over a finite time,

\[
J = \int_0^T x^T Q x + u^T R u = \int_0^T \alpha y^2 + \dddot{y}^2 \ dt,
\]

can also be solved to give feedback control, but with variable gains as in the case of the analogous problem with control over acceleration, as described in section 2.3.1. Using Pontryagin’s theorem it can be shown that the control solution is a quadratic control function of time, rather than a linear function as in the case of control over acceleration (refer to appendix 8.6).

### 4.5.2. Nonlinear Feedback Control

Even though suspensions of production cars are based on linear models they are far from being truly linear. Production shock absorbers (dampers) are designed to have two different rates, under bump and rebound, as outlined in section 2.2.2. Sometimes variable-rate springs are used, generally as after-market add-ons, and rubber or polyurethane bump stops also contribute to suspension nonlinearity. The nonlinear feedback controls discussed in this section vary “stiffness” rates in a nonlinear way. Thus instead of multiplying by a parameter, as in equation 4.6, stiffness will be a nonlinear function.

In this thesis nonlinear functions of a single variable will be used, which can be summed together as represented in general form in equation 4.8 below:

\[
u = \dddot{y} = -\beta_1 (y,s) - \beta_2 (\dot{y},s) - \beta_3 (s,s) - \beta_4 (\dddot{s},s)
\]

**Equation 4.8**
The functions $\beta_i$ are not necessarily linear. It could be expected, for instance, that the suspension may become stiffer with increasing travel away from equilibrium, especially as the suspension approaches the rattlespace limits. Similarly, controls over jerk may have the form,

$$u = \ddot{y} = -\beta_1(y,s) - \beta_2(\dot{y},s) - \beta_3(\ddot{y},s) - \beta_4(s,s) - \beta_5(\dot{s},s) - \beta_6(\ddot{s},s)$$

Equation 4.9

Possible nonlinear functions are discussed in the following sections.

### 4.5.2.1. Sigmoid Functions and Fuzzy Sets

Truly fuzzy control was not employed in this thesis although fuzzy set implementations were developed in some early experiments. Fuzzy control with a large number of parameters was found to produce very slow evolution even though the fuzzy sets were developed independently for each parameter. Fuzzy control with a large number of member sets was found to deliver very little advantage here. The purpose of this thesis was to derive a better theoretical understanding of controlled suspension, and using just a few “member sets” is much more easily interpreted and analysed than a complex fuzzy control. Fuzzy controls can have the effect of concealing the theory behind computer intensive numerical methods. Fuzzy controls can be effectively developed almost completely outside any design theory or strategy; “setting the performance index is the only procedure for the designer of the controllers” (Hashiyama et al., 1995, p166). The reverse engineering of a comprehensive fuzzy control may give interesting results but it was thought that this approach would have severely impeded the overall development of the research for this thesis.

Sigmoid functions (see section 8.1.5) can be thought of as functions with just two “member sets”. These have a profile similar to that shown in figure 4.5. Sigmoid functions, as in equation 4.8 and equation 4.9, can be employed in EAs to perhaps evolve stiffness parameters that become larger as they travel away from equilibrium in a nonlinear way.

The sigmoid function employed to separate the “member sets” uses the sine function,
\[
y = \begin{cases} 
  h_1 & \text{if } x \leq c - w, \\
  h_2 & \text{if } x \geq c + w, \\
  \frac{h_2 + h_1}{2} + \frac{h_2 - h_1}{2} \sin \left( \frac{\pi(x-c)}{2w} \right) & \text{otherwise.}
\end{cases}
\]

Equation 4.10

The values \( c, w, h_1 \) and \( h_2 \) are the centre, width, height at the lower end and height at the upper end respectively. An example is shown in figure 4.5. The parameters can be determined by an evolutionary algorithm. Sigmoid functions can be used to vary all or any of the parameters of equation 4.8 or equation 4.9.

Figure 4.5 Sigmoid Function

4.5.2.2. Virtual Bump Stops

Real-world passive suspensions have bump stops at the end of the suspension travel, which dramatically increase the restorative force of the suspension when they are depressed. In an electronically-controlled suspension this effect can be mimicked (or enhanced if actual bump stops are also used) by supplying extra force as the suspension approaches the vertical suspension travel limits.
A suspension should stiffen when approaching the rattlespace limits (stroke travel limits) to prevent or decrease the risk of the wheel hitting hard against the rattlespace limits. This is represented schematically in figure 4.6. Furthermore, if the wheel is close to the vertical suspension travel limit and moving towards it, then the suspension should become stiffer than if the wheel is moving away from the chassis.

![Stiffness Graph](image)

**Figure 4.6 Asymmetrical Multiplying Factor**

With an electric control, one simple way to do this is to modify the parameters of a linear function, as shown in equation 4.8 and equation 4.9. One way to think of this is to use coefficients that increase near the rattlespace limit. Such a modified linear control could be represented as,

\[ u = \ddot{y} = -\beta_1(s)y - \beta_2(s)\dot{y} - \beta_3(s)s - \beta_4(s)\dot{s}. \]

The functions \( \beta_1 \) to \( \beta_4 \) could have the form represented in figure 4.7, which has four parameters for each coefficient, \( s_1, c_1, s_2 \) and \( c_2 \). Alternatively, a sigmoid function such as that represented in figure 4.5 could be used. The parameters of the skew function could be determined by an evolutionary algorithm.
A simple refinement on this is to only increase the coefficient if the parameter is moving in the direction of the rattlespace edge. For example, if the stroke velocity is such that the chassis is approaching the rattlespace limit, then increase the coefficient, but if it is moving away from the rattlespace do not increase the coefficient (use the smaller value, for example $c_1$ in figure 4.7). Similar modifications could be made to linear control over jerk.

Many different formulas could be used to implement an increasing in stiffness on approach to the rattlespace limit. Such a formula may involve dividing by the distance to the closest vertical suspension travel limit,

$$\frac{1}{R - s},$$

where $R$ is the limit of the value for stroke, $s$. This is an asymptotic function, increasing to infinity as the distance approaches zero, which creates instability for the numerical methods. Of course a real suspension physically disintegrates before it actually reaches this condition. This matter is discussed further below, in section 4.9.

More complex measures of “closeness” are of course possible. The above measures have been applied heuristically to see if they produce improvement in numerical experiments. Will a control that is “soft in the center” and which becomes stiffer on approach to the rattlespace limits provide a better compromise between softness over relatively smooth roads with the ability to avoid rattlespace collisions in bumpy condition? In any case, it is clear that the
controls discussed above include controls that are impossible to implement even with the most sophisticated passive bump stops.

For the sake of discussion, suspension controls that use a nonlinear function to stiffen on approach to the rattlespace limits have been termed by this author “virtual bump stop” controls. “Virtual bump stop” methods have not been investigated before, to the best of the author’s knowledge.

### 4.6. Minimum-Time Control

As shown in section 2.3.2.3, the optimal control to return to rest in minimum time using constrained acceleration is a bang-bang control. The proof of optimality appeals to Pontryagin’s theorem. Bang-bang control can be extended to control over jerk. With acceleration control, distance and velocity are simultaneously brought to rest. But with jerk control, distance, velocity and acceleration all come to zero simultaneously. Acceleration is continuous, changes smoothly, and is brought to rest simultaneously with distance and velocity. Thus acceleration is not suddenly dropped to zero. In this section Pontryagin’s theorem is used to prove the optimality of the bang-bang control over jerk (and to derive the switching points).

A literature search has shown that variants on bang-bang controls have been researched in the area of mechatronics (Koh et al., 1999). These have tended to deal with predetermined symmetric trajectories, or with different optimization criteria. The simple discrete control based on determining the phase-space point in relation to the landing surface had been developed in the course of the research, but it was later discovered that this method has been investigated by Koh, Aum et al. This method will be explained in detail below. This method has been published by the author during the course of the PhD research (Storey et al., 2009).

Interestingly, the discrete bang-bang method could be characterised as a special case of sliding-mode control (see section 2.3.2.4) over jerk. Sliding-mode control over acceleration
has been widely researched (Ashari, 2004; Donahue, 2001; Yagiz, 2005; Dixit and Buckner, 2005; Yokoyama et al., 2001), and it has been found to have the problem of chattering, which has been resolved in the research by using a linear control near the switching manifold. With control over jerk however, “chatter” results in only very small, smooth changes in force, rather than large, fast swings in force. The control proposed here is a sliding-mode control (over jerk) which does not need to be artificially softened near the sliding surface.

Similar to the way in which at most one intermediate switch of acceleration direction is needed for minimum-time control over acceleration, at most two switches of control direction are needed for minimum-time control over jerk. The proof is discussed below and parallels the proof using Pontryagin’s Principle shown in section 2.3.2.3. The major difference is that a solution to the adjoint equation is linear in the case of control over acceleration, and so crosses zero at most once, while the corresponding equation in the case of control over jerk is quadratic as a function of time and generally crosses zero at most twice.

Consider then the following optimization problem: what is the minimum time required to return a system to rest (zero distance, velocity and acceleration) using constrained jerk? The proof below uses Pontryagin’s Principle as applied in MacCluer (2005, pp120-9). A different proof, taken from the method in Hermes and LaSalle (1969) can be found in section 8.7, validating the result.

To minimize time, minimize the functional,

\[ Q = \int_{0}^{T} dt, \]

where the control jerk, \( u(t) \), is constrained,

\[ -j \leq u(t) \leq j. \]

The equation of motion in vector form is,

\[ \dot{x} = F(x) = F(x, v, a) = (v, a, u). \]

That is to say, \( u = \dot{a}, \ a = \dot{v}, \) and \( v = \dot{x} \). Here \( x, v \) and \( a \) represent distance, velocity and acceleration respectively. The Hamiltonian is,

\[ H = -1 + \lambda.F = -1 + (\alpha, \beta, \gamma).(v, a, u) \]
\[ = -1 + \alpha v + \beta a + \gamma u. \]

The control, \( u(t) \), which minimises the Hamiltonian with control is given as,
\[ u(t) = \begin{cases} -j & \text{if } \gamma(t) < 0 \\ j & \text{if } \gamma(t) > 0 \end{cases} \]

**Equation 4.11**

The adjoint equation is,

\[
(\dot{\alpha}, \dot{\beta}, \dot{\gamma}) = \dot{\lambda} = -H_x = -\left( \partial H / \partial x, \partial H / \partial v, \partial H / \partial a \right) = (0, -\alpha, -\beta).
\]

Solving this system of differential equations produces,

\[
\begin{align*}
\alpha &= \alpha_0, \\
\beta &= -\alpha_0 t + c_1, \\
\gamma &= \frac{1}{2} \alpha_0 t^2 - c_1 t + c_2,
\end{align*}
\]

**Equation 4.12**

where \( \alpha_0, c_1 \) and \( c_2 \) are constants. The equation for \( \gamma \) is quadratic and changes sign at most twice, and together with equation 4.11, this implies that there are at most two switches for the control, \( u \). That is to say, there are at most two points at which the jerk switches between \( j \) and \( -j \), or vice versa.

As in section 2.3.2.3, to find the optimal trajectory, the control is deduced from end conditions. It is useful to define a two-dimensional “switching surface” in the three-dimensional phase space comprising displacement, velocity and acceleration: \( (y, \dot{y}, \ddot{y}) \). This surface contains all the points that can reach the origin, including acceleration, \((0,0,0)\) using just one switch. In sliding-mode control this surface could be called the “switching surface” or “switching manifold”, but it has also been called the “landing surface” in the literature on time-optimal control for robotic manipulators (Koh et al., 1999, p273).

On the landing surface, velocity and acceleration are each brought to zero simultaneously using jerk in exactly the same way that distance and velocity are brought to zero using acceleration, elegantly depicted in the interpretation of Pontryagin’s diagram (Pontryagin et al., 1986, p26) shown earlier in figure 2.3. The control of the switching surface can be projected onto the plane \( y = 0 \), shown below in figure 4.8, producing exactly the same schema as in figure 2.3 except with \( y \) replaced by \( \dot{y} \), and \( \ddot{y} \) replaced by \( \ddot{y} \). Figure 4.8 thus represents a projection of the switching surface onto the velocity and acceleration axes; the displacement axis, \( y \), is out of the page.
A three-dimensional plot of the landing surface is shown below in figure 4.9. This has been generated in Matlab and the code can be found in the appendix, in section 0. Again, this is the surface from which the system can return to rest (including zero acceleration) with just one intermediate switch of jerk. The system will generally be initially either above or below this surface, and will switch when reaching this surface, and then finally switch once on this surface, at the final switching curve. The final switching curve is shown as a thick line on the diagram.

Figure 4.8 Switching-Surface in Phase-Space – projected onto the velocity and acceleration axes
Figure 4.9 3-D Plot of Landing Surface

It has been shown that the control to return to rest can consist of at most 2 switches. If the system begins from a point “above” the switching surface then the initial jerk must be negative to bring the system onto the switching surface. At the switching surface the direction of jerk changes and there is one more change of direction of jerk on the switching surface. Similar remarks apply if the system’s initial point is below the switching surface (with the initial jerk in the opposite direction).

The calculation required to deduce the direction of jerk at any point outside the landing surface is quite simple. Find out if the current point in phase space is above or below the switching surface by calculating the distance on the surface with the same velocity and acceleration as the current point. This is almost exactly the same mathematically as the bang-bang control over acceleration.
The control over acceleration and velocity once on the landing surface is easily calculated. This determines jerk while on the landing surface and so movement on the landing surface is easily calculated; once on the landing surface simple cubic polynomials give the trajectory, and the timing of the jerk switch is readily calculated (Java code for this calculation is given in the appendix in section 8.8).

Thus, once on the landing surface the trajectory is easily calculated, and it is virtually the exact same algebra as control over acceleration. The problem is that the point at which the system enters the landing surface is not so easily calculated. While the height above the switching surface is easily calculated, the optimal trajectory does not generally intersect the switching surface at the point directly “beneath” the initial point. Nonetheless, it is a simple matter to determine if any given point is above or below the landing surface, and this is sufficient to determine the jerk direction required. Thus a feedback control can be very easily implemented simply by calculating if the current point is above or below the switching surface, in exactly the same way that a closed-loop control can be implemented using acceleration which determines if the initial point is above or below the switching curve.

Suppose that the system is at the initial position in three-dimensional phase space, \((y_0, v_0, a_0)\), where \(y_0, v_0\) and \(a_0\) are the initial displacement, velocity and acceleration respectively. To determine if the point is above or below the switching curve simply determine the displacement for the switching curve directly “beneath” the intial point; that is, the point with the same velocity and acceleration values. Suppose that the distance moved on the landing surface is \(d\) (calculated using cubic polynomials). This point on the landing surface is \((d, v_0, a_0)\) and is not generally the same as the current \((y_0, v_0, a_0)\). However, it is an easy matter to determine if the current point is above or below the landing surface just by comparing \(d\) and \(y_0\). If \(y_0 > d\) then apply negative jerk, \(-j\), and if \(y_0 < d\) apply positive jerk \(j\). As expressed by Koh et al., this method can calculate “the distance from the current point ... to the landing surface ... The jerk is determined to make the point approach the landing surface” (1999, p275).

It should be re-emphasized that this does not determine the entire trajectory. In particular it does not determine the point that the trajectory first enters the landing surface. It determines
only whether the initial point is above or below the landing surface and this determines the
direction of jerk at the initial point. The system does not enter the landing surface at the point
\((d, v_0, a_0)\) nor does it generally pass through this point. This method calculates only the jerk
at the current instant. For an iterated control this is all that is needed, and it has been
investigated as a possible robotic arm movement control. “The proposed motion planning
method is implemented as [an] iterative algorithm which determines velocity, acceleration
and position commands at every control instant” (Koh et al., 1999, p275). The calculation
involves only a few mathematical steps, the most costly operation being a single square root.
The control is quite straight-forward and easy to implement.

In some of the algorithms discussed in this thesis, the maximum and minimum distance
travelled in the trajectory need to be computed and the algorithm just outlined is insufficient.
However, all aspects of the trajectory are easily computed once the time taken to reach the
landing surface is known, as discussed at length in subsection 4.6.3.

Minimum-time bang-bang controls over jerk can be extended to bang-bang controls over
higher-order derivatives, in the same way that the one-switch bang-bang control over
acceleration has been extended to the two-switch control over jerk. Note that control over
jerk moves first onto a two-dimensional switching surface, while the control over
acceleration discussed earlier moved first onto a one-dimensional switching curve. Thus
control over jerk can be transformed to a three-switch control over snap (the rate-of-change
of jerk), but with much greater computational difficulty. To control snap, the three-
dimensional switching manifold in four-dimensional phase space is made up of the control
over jerk. The distance “above” this switching manifold is calculated, which in turn
determines the direction of snap that needs to be applied at a given moment. Note that this
requires the full trajectory for the control over jerk. Control over snap or further higher-order
derivatives have not been implemented in this thesis.

An example application of minimum-time control over jerk is shown in figure 4.10. In this
example the initial values for displacement, velocity and acceleration are \(-1.25, 2\) and \(-0.2\)
respectively. The graphs of jerk, displacement, velocity and acceleration are shown. The
control jerk value is plus or minus 2. Displacement, velocity and acceleration all reach zero
simultaneously.
Initially, constant negative jerk pulls back on the acceleration (dashed line), then switches to increasing the acceleration to a positive value, and finally brings the acceleration back to zero. Note that acceleration is continuous, and hence force is continuous, although jerk is discontinuous.

This graph was produced by a computer program (written by this author) in which the various parameters could be varied and simultaneous zero displacement, velocity and acceleration were achieved in all observed cases, confirming the algorithm. Code to implement this
algorithm is discussed in the appendix, in section 8.8. (Select the button “Min-Time Jerk” in the program OvershootDemo.java. Sliders for “road Parameters” allow initial conditions to be changed, and the “COMFORT_JERK” parameter in the “Jerk Profile” group of sliders allows the control jerk to be changed.)

Graphs such as the above consume a lot of space. In figure 4.11 below, the graphs of acceleration velocity and distance are superimposed. The jerk control can be inferred from the straight slopes in the acceleration profile. Excluding the jerk for the superposition seems to make the graphs clearer. This kind of graph is used here to save space. Other authors have used similar superimposed graphs to illustrate controls (Ross, 2009, pp46-7; Kirk, 1970, p266).

![Combined Graph](image)

**Figure 4.11 Combined Graph**

### 4.6.1. Feedback Minimum-Time Control over Jerk

An iterative, discretised version of the minimum-time control over jerk is very easy to implement. The current state in phase space is compared with the landing surface, as explained in section 4.6. If the iterative, “discrete” method is used with a reasonably small time period, the numerical result is very close to the analytical result, except that there is some jerk “chattering” (see section 2.3.2.4) around zero. Compare figure 4.11 with figure
4.12 below. Note again that the jerk graph in figure 4.12 has not been included because it can be easily inferred from the constant slopes of the graph of acceleration. The optimal, minimum-time control is achieved as a discretised feedback control via real-time optimal control (RTOC) (Ross, 2009, p48).

Chattering is in part due to accumulated numerical errors, but mainly to the fact that the jerk is never actually sent to zero, even when the system is at or near rest. (The code for this algorithm is also given in section 8.8.) It should be noted however that jerk chattering need not be uncomfortable and in fact may be entirely unnoticeable. Acceleration chatter, on the other hand, results in sudden, extreme changes in force, which are certainly uncomfortable.

There is an obvious similarity with the sliding-mode control. This feedback version of the control can be viewed as a special case of a sliding-mode control over jerk: jerk changes direction depending on whether the system is above or below the landing surface in phase space. In fact, “chattering” of a kind is produced here for the same reason that it develops in sliding mode control over acceleration, although jerk chatter does not result in the discomfort produced by acceleration chatter.

![Figure 4.12 Discretised Minimum-Time Control Example](image)

After a literature search it was found that this control was proposed by Koh et al. for robotic arm control. Koh et al. propose the control for its anti-vibration benefits, “The anti-vibration
is the key factor for determining the life cycle of the mechanism” (Koh et al., 1999, p273). Low jerk results in low wear and tear on physical components, but it is also claimed to result in improved tracking; “Jerk limitation is important in industrial robot applications, since it results in improved path tracking and reduced wear on the robot” (Macfarlane and Croft, 2003, p42).

4.6.2. Bang-Off-Bang Control

“Bang-off-bang” (BOB) controls have also been studied for use with robotic arm movements, as discussed in section 2.3.2.3. By sacrificing time, such controls can produce softer results overall (Muenchhof and Singh, 2003, p142). Bang-off-bang controls are often the result of secondary constraints. Koh et al., for example, “propose a minimum time motion planning algorithm considering jerk and acceleration constraints” (1999, p273). Note the inclusion of both jerk and acceleration constraints.

A bang-off-bang control is much more easily visualized in the case of a control over acceleration rather than jerk. Consider the two controls for returning to rest depicted in figure 4.13. The control, A to B to D and then to rest, is the bang-bang control of return to rest using constrained acceleration as described above in section 2.3.2.3. The force is constant and negative from A to B, and switches to a constant positive force from D to rest. After reaching the rest position, the force is zero. The second control, A to B to C, and then to rest, is a bang-off-bang control; there is no acceleration between points B and C. The bang-off-bang control takes longer than the bang-bang control, but it travels the same maximum distance, and uses less overall acceleration in the process (it here uses the same acceleration for less time, but it also uses therefore less total RMS acceleration).
The phase plane diagram corresponding to figure 4.13 is shown below in figure 4.14. The first control is a trajectory from the phase-space diagram of figure 2.3. The dotted line shows the BOB control moving from $B$ to $C$. The BOB control sacrifices time but travels with a smaller velocity.

Figure 4.13 Return to Rest comparing Bang-Bang with Bang-Off-Bang

Figure 4.14 Phase-Plane Diagram of Bang-Bang and Bang-Off-Bang
A phase-plane plot of a bang-off-bang control over acceleration is shown below in figure 4.15, but variations are possible. Kirk has a similar variation resulting from weighted time and fuel performance (1970, p283). The relationship between this diagram and the phase plot for the minimum-time control in figure 2.3 should be clear.

Figure 4.15 Acceleration Control with Constrained Velocity

One way then to implement BOB control over jerk is to use the control over acceleration depicted in figure 4.15 as a landing surface for a control over jerk, as was done for the minimum-time control. Using this method produces trajectories such as the one shown in figure 4.16. As with control over acceleration, time is sacrificed, and overall RMS jerk is decreased.
In figure 4.16 the acceleration is constrained on part of the control. Thus both jerk and acceleration are limited. BOB controls result naturally when there are multiple constraints such as this. BOB controls over acceleration result for instance when there are velocity constraints. Another BOB algorithm using jerk is developed later in section 4.7.3.7, and this also derives from constraint over both acceleration and jerk.

### 4.6.3. Time to Landing Surface and Optimal Displacements

In later sections, the maximum and minimum distances travelled by the minimum-time control over jerk need to be computed, as well as other parameters. As shown above, the feedback control is quite simple and does not require the calculation of the time taken to reach the landing surface. However, to fully describe the trajectory of the minimum-time control the time to reach the landing surface needs to be computed.

The entire trajectory for the minimum-time control over jerk is simply made up of three cubic polynomials and all the characteristics of the curve, such as the optimal displacements and the times at which they occur, can be easily determined by simple calculus once the time
taken to reach the landing surface is determined. However, no simple closed-form formula for this value was found.

A simple, reasonably fast iterative numerical method has been devised. As discussed above, it is easy to determine whether a given point is above or below the landing plane; this is the basis of the iterative method.

The numerical method begins by finding a time period which results in a point on the opposite side of the landing plane to the initial point, by repeatedly multiplying an initial time guess by two and stopping when a point on the opposite side of the landing plane is reached. This produces two points, one on either side of the landing surface. Iteratively take the midpoint between these two times. If the midpoint time is too low it is used as the new low estimate, if it is too high it replaces the new high estimate. In this way the difference between these estimates decreases exponentially. Once the difference becomes acceptably small, one of the values is chosen as the value of the time to the landing surface. Once this time is found, the rest of the algebra involves simple cubic polynomials.

The entire code for this algorithm, including finding maxima and minima, and the times at which they occur is to be found in the Eclipse Java package in the Overshoot package in the class SwitchingPlaneX.java. Test code can be run from the OvershootDem0.java class in the OvershootControl package. (Click the “Min-Time Jerk” and “Cts. Min-Time Jerk” buttons and vary parameters d, v, a and COMFORT_JERK.)

The Java code for finding the time to reach the landing surface is also given in the appendix in 8.9. This method uses an initial guess of the time to reach the landing surface. If the distance travelled in this time results in a point on the same side of the landing plane as the initial point, then try a time twice as large as the original, otherwise halve. Continue this process until two times are found, one on either side of the landing plane. Now, dissect these two times and use the new time as an upper or lower limit, depending on whether it results in a point on the opposite or same side of the landing plane as the initial point. For instance, if the new point is on the opposite side then this becomes the new upper limit time. Keep dissecting in this manner until an acceptably small “tolerance” level is reached. This is a “binary” method which has a log order magnitude of algorithmic complexity.
4.7. **Rattlespace Constraint Controls**

Figure 4.17 below shows a graph of a quarter car suspension as it travels over a road surface. The chassis movement is physically limited to remain within the suspension’s rattlespace. Hitting the rattlespace limits results in great discomfort and perhaps even vehicle damage. Any movement within the rattlespace itself has no inherent negative effect on suspension goals. Even if the suspension is very close to a vertical suspension travel limit there is no negative effect. Of course, if the chassis is close to a limit there may be more chance of hitting the limit in the near future, but there is no inherent negative tracking performance impact as long as it does not hit the limit.

![Figure 4.17 Rattlespace Limits](image)

The goal of a limited-stroke suspension control is to guide the chassis smoothly within the rattlespace without hitting up hard against the ends of the rattlespace, more precisely, to find a trajectory through the rattlespace that achieves an optimal compromise between handling and comfort. Suspension movement is constrained by the rattlespace and suspension tracking performance should be measured by a suspension’s capacity to remain within the rattlespace.
This is an obvious point and yet, as discussed in chapter 3.2, there are no experiments or theories in the surveyed literature dealing with tracking in this way.

This section of the thesis investigates how controls might be devised specifically to stay within the rattlespace. For the want of a better term, such controls have been referred to by this author as “rattlespace constraint” controls; their defining trait is that control is in some way based on a model of the trajectory of the chassis within the anticipated rattlespace. Control is exerted to restrain the chassis from crossing a rattlespace limit.

Two distinct general approaches are possible, as explained in a later section. One is “variable hardness”, and the other “edge constraint”. Specific implementations of these methods are developed and are applied in numerical experiments.

Of course, real suspension controls have limits. Given a sufficiently bumpy road (traversed at a sufficiently high speed) any suspension will hit up hard against the vertical travel limits. The reasons why numerical models may allow travel outside the rattlespace (discussed at length in section 3.2) and the numerical considerations of section 4.9 below, still apply to rattlespace constraint controls. Models may therefore travel outside the rattlespace. However, heavy penalties are applied to travel outside the rattlespace, and evolution should avoid this where possible, consistent with the goal of maintaining a smooth ride.

All the controls and theory presented in this section (4.7) are original work unless otherwise stated, to the best of the author’s knowledge, and were conceived during the research for this thesis.

4.7.1. Theoretical Basis of Rattlespace Constraint

In order to maintain the trajectory of the chassis within the future limits of the rattlespace, it is necessary to anticipate the future movements of the rattlespace limits. This thesis does not investigate look-ahead systems and it is assumed that the road surface ahead of the vehicle is uncertain, and so the best models of future rattlespace limits must be statistical in nature.
Thus the future wheel movement can be heuristically represented as a statistical “probability cloud”, as in figure 4.18.

In this diagram, zero time represents the current “instant” (the small time period of the current calculation in an iterated method). Probability densities in this diagram could represent probabilities that a given height at a given time is within the rattlespace. The darker regions in figure 4.18 represent higher probability that a given height at a given time will be within the rattlespace. The vertical suspension travel limits before this are known with certainty, because this period has just elapsed, and probabilities are either zero or one. Future rattlespace limits however are unknown, and become less and less certain with increasing time.

Perhaps a probability density of trajectories is more accurate, but there is little benefit from this complication in this context because only heuristics are applied in this thesis and the graphical representation of figure 4.18 greatly assists the explanation. Even so, given a statistical description of road types, the probability of a point on the graph being within the rattlespace is mathematically well-defined.
It is intuitively obvious that the “probability cloud” becomes more and more uncertain with increasing time above zero, and the probability density is roughly highest at the current height and decreases with further distance from that height. The exact mathematical nature of the probabilities, however, is not needed for this thesis. Heuristic methods are employed as discussed in section 4.7.3 in order to approximate future rattlespace limits, and evolutionary algorithms have been used to optimize and test performance. The model has a theoretical purpose at this point.

The challenge for suspension design is to find a path through the probability cloud that is smooth but which minimizes the likelihood of hitting against the rattlespace limits, at least in the “near to short term”. Time periods that are closer to the current instant are more urgent because later times can be subject to further adjustments.

A rattlespace control should have the property that it can prioritize comfort when there is little immediate danger of hitting against the rattlespace limits, but become stiffer if the threat of a rattlespace collision is imminent, as depicted in figure 4.19. Here the chassis is pulled down more tightly than in figure 4.18, in order to avoid the more highly likely imminent edge collision. When collision is not imminent, a rattlespace constraint control should be able to prioritize comfort, even if the edges of the rattlespace are undergoing rough but small corrugations. In fact, distinguishing small corrugations from large bumps is perhaps the main difficulty for a rattlespace constraint control.

![Figure 4.19 Rattlespace Constraint with Imminent Collision](image)

Figure 4.19 Rattlespace Constraint with Imminent Collision
4.7.2. Nominated Target Rattlespace Limits

There are a range of possible future positions for the rattlespace edge. It may be possible then to designate a certain particular line through the probability cloud as a target for current chassis trajectory control. That is to say, at each future time particular heights are nominated as possible rattlespace limit targets. In figure 4.20 a particular line, after the current instant, represents the upper rattlespace edge. The goal then is to keep the chassis within these explicit targets. Clearly, this is a heuristic to avoid complexity.

![Figure 4.20 Nominated Target Rattlespace Limits](image)

This should not be thought of as a single possible future rattlespace trajectory. Rather, the target limits should produce a control which can distinguish those moments when rattlespace collision is imminent from those moments where collision is not imminent, and comfort can be prioritized. When collision is imminent, the target lines should form a limit that the suspension control can aim for and have some chance of recovery in the case that the rattlespace edge moves within the expected limit. Thus the target line should be more restrictive in the near term, and spread out in the long term. Evolutionary algorithms can be
used to determine the parameters of the target rattlespace limits. The goal of the control then is to find, on a moment-to-moment basis a control that can “steer” the chassis through the nominated target limits and apply the control for that trajectory in that time cycle.

Suppose the actual rattlespace limits (past and future) are represented as \( R_T = r(t) + R \) and \( R_B = r(t) - R \), where \( R \) is half the rattlespace width and \( R_T \) and \( R_B \) are the top and bottom rattlespace limits. Simply flat target limits could be employed, as depicted in figure 4.21(a) below, or more complex parabolic limits could be nominated using the equations,

\[
R_T = r(0) + R + D_A t + D_A t^2 + P_T \ddot{r}(0)t + \frac{1}{2} P_T \dddot{r}(0)t^2 \\
R_B = r(0) - R - D_A t - D_A t^2 + P_T \ddot{r}(0)t + \frac{1}{2} P_T \dddot{r}(0)t^2,
\]

depicted in figure 4.21(b).

![Rattlespace Projections](image)

**Figure 4.21 Rattlespace Projections**

Figure 4.22 shows a road with a mixture of large bumps and corrugations. Distinguishing corrugations from large bumps using only past data is perhaps the greatest challenge for a rattlespace constraint control. On the one hand, a soft response to a large bump leads to
rattlespace collision, or at least high jerk responses near the rattlespace edge. On the other hand, a hard response over-reacts to small edge corrugations and produces an uncomfortable ride.

Figure 4.22 Large Bump with Corrugations

A more sophisticated statistical analysis of current and past road conditions (or at least wheel conditions) may be needed to yield reliable predictions for the onset of large bumps. In this thesis, only current instantaneous road conditions are applied, although filters using moving averages as described in 8.12 may be applied, and these could be regarded as a rudimentary form of road “memory” that are used in order to distinguish rough from smooth sections of road.

When the road conditions are quite soft and the rattlespace edge is moving only say within 5% of the rattlespace edge, there is no reason why the chassis should move appreciably. A passive suspension, and even the skyhook, will move somewhat under such conditions, but a rattlespace constraint control might barely move at all. It is only when there is an imminent likelihood of rattlespace collision that the suspension should move or stiffen to avoid edge collisions. The problem is to distinguish which of the two modes, call then “soft” and “hard”, should take priority.

Accordingly, a rattlespace constraint control may need to switch between a “hard” mode that avoids collision with the rattlespace limits and a “soft” mode that seeks to reach equilibrium comfortably. As will be explained in later sections, the main problem for the hard mode is to avoid edge collision without creating an unstable rebound away from the edge.

Another kind of control results from varying the “hardness” of the system. For example, if maximal jerk is the measure of “hardness”, the hardness is set to the level such that the
maximum displacement is not outside the rattlespace. There are a number of ways of interpreting suspension “hardness” as explained in detail later in section 4.7.5 so there are a number of ways of implementing this general idea. This is simpler to implement than the edge constraint method just outlined, and such methods have been implemented with some success in the numerical experiments.

Fast and simple control algorithms are the object of this thesis, and rattlespace constraint techniques tend to be relatively complex, at least compared to linear or sliding-mode controls. Nonetheless, microprocessor power is becoming very cheap, as are transducers for measuring movement. It is becoming possible to use more sophisticated control algorithms in an ever wider range of applications.

4.7.3. General Displacement Constraint Problem

When approaching the bottom, say, of the rattlespace, control (either jerk or acceleration) may be needed to pull the chassis back up. The suspension displacement is constrained to lie above the lower rattlespace limit, and below the upper limit. A defining component of a rattlespace constraint control is that it moves the chassis to avoid hitting either the top or the bottom of the rattlespace.

This in turn suggests the general problem of finding a control that is constrained by distance in just one direction. The problem examined below is to come to rest as quickly and as smoothly as possible without overshooting the rest position. This problem could be easily extended to the case of smooth approach to an accelerating target distance, by translating into a frame accelerating with the target. This kind of “normalisation” is sometimes used in control theory (Ross, 2009, p32).

Control methods to address this problem may find application in other contexts where it is desirable to reach a rest position without overshoot, such as robot arm movement (Constantinescu and Croft, 2000), parking of hard drive heads (Chang and Hori, 2006), heavy door closing, elevator approach to a floor, satellite rotation control (Zadeh, 2004) or possibly even with airplane landing gear (Krüger, 2002).
With a robotic arm the goal of a distance constraint is to move the end of an arm to rest at a certain point, without collision with another object beyond that point. Industrial robotic arms should move from point to point as quickly and as smoothly as possible, and cubic polynomial trajectories have been investigated for this purpose, in an effort to avoid the jerky end movement that so typifies robot movement (Ben-Itzhak and Karniel, 2008; Constantinescu and Croft, 2000; Hicks et al., 2006; Macfarlane and Croft, 2003). Smoothness can help to minimize wear and tear (Koh et al., 1999; Cao et al., 1997) and perhaps even decrease the cost of robotic rigs. Cubic splines have been used in the literatures for pre-planned movement, but the methods described here allow jerk to be used with feedback controls.

4.7.3.1. Rebound

It is helpful to illustrate the problem of rebound in the context of relatively simple examples. Without loss of generality, and to simplify some of the algebra, suppose that the target rest position is set as zero. The displacement constraint problem is one of smoothly bringing a system to zero displacement with zero velocity, so that the system does not overshoot zero distance. Visualize an industrial robot arm coming to rest near a delicate piece of machinery without bumping into it.

Unless otherwise stated, \( d, v, \) and \( a \) represent initial distance, velocity and acceleration respectively,

\[
d = y(0), \quad v = \dot{y}(0), \quad a = \ddot{y}(0).
\]

Without loss of generality, the initial distance is assumed positive, \( d > 0 \), unless otherwise stated.

Let us begin with the simple method of using constant force. This problem is then readily analysed in terms of energy and work. To bring the system to rest requires just enough force to absorb the initial kinetic energy.

\[
\text{Work Done} = \text{Initial Kinetic Energy},
\]
\[ Fd = mad = \frac{1}{2}mv^2, \]
\[ a = \frac{v^2}{2d}. \]

**Equation 4.13**

Constant acceleration will bring the system to zero distance with zero velocity, as depicted in figure 4.23.

**Figure 4.23** Constant Acceleration to Rest: \( d=4, v=-4 \) and \( a=2 \)

With this very simple method the force is constant, from beginning to end. However, the system will rebound upon reaching zero unless the acceleration is suddenly dropped to zero, with an associated spike in jerk. To actually bring the system to “rest” including zero acceleration requires then a large jerk as the acceleration is suddenly dropped to zero. If the trajectory is to avoid discontinuous force (and extremely high jerk) then the problem remains of reducing the rebound acceleration.

In the case of a suspension system, rebound away from one of the rattlespace limits can cause it to move back too quickly towards the opposite side of the rattlespace. Rebound in this case is a source of instability.

This shows that simple energy considerations alone do not solve the problem of finding a smooth trajectory that avoids collision at a given distance. Because of the added requirement of smoothness, the force that remains at the end cannot be suddenly dropped to zero, and if
the residual force is too large then either there must be a sudden uncomfortable change in force, or the trajectory must rebound a large distance, moving back towards the opposite rattlespace limit and affecting stability.

Exponential decay does not have this problem of rebound. Consider the example shown in figure 4.24.

![Figure 4.24 Exponential Decay](image)

Suppose that the equation of motion is,

\[ y = de^{-\beta} \]

If the initial velocity is \( v \), then,

\[ v = \dot{y}(0) = -\beta d. \]

Solving for \( \beta \) gives, \( \beta = -v/d \), and the initial value of acceleration is,

\[ a = \ddot{y}(0) = -2\beta v - \beta^2 d \]

\[ = \frac{v^2}{d} \]

**Equation 4.14**

Now there is no residual acceleration or threat of rebound.

Interestingly, the force at time zero is now exactly twice the force needed in the previous case using constant acceleration, given by equation 4.13. It may seem counterintuitive that this is a “smoother” system since it requires greater initial force. However, by absorbing energy at a faster rate at the start, less force is required later, and there is no rebound at the end. The price for removing rebound is a higher initial force. On the other hand, it does seem intuitively
clear that a stiff suspension is less prone to wild movement between rattlespace limits than a very soft one.

In the case of exponential decay all derivatives of motion are brought to zero, which may be more than needed. A slightly more general result can be obtained, using a constant \( k^{th} \) derivative of motion to bring a system to rest with all lower derivatives coming to zero.

Suppose that a constant \( k^{th} \) derivative of motion, equal to \( -K \), is employed in order to bring all lower derivatives to zero simultaneously. The algebra becomes slightly easier to represent if the timing is reversed and the system starts at zero, with all the lower-order derivatives set to zero at time zero. Therefore,

\[
y(0) = y'(0) = ... = y^{(k-1)}(0) = 0 \quad \text{and,} \quad y^{(k)}(t) = K.
\]

Furthermore,

\[
y^{(k-1)}(t) = Kt,
y^{(k-2)}(t) = \frac{1}{2} Kt^2,
\]

\[
\vdots
\]

\[
y^{(k-r)}(t) = \frac{1}{r!} Kt^r,
\]

\[
\vdots
\]

\[
y'' = y^{(k-(k-2))}(t) = \frac{1}{(k-2)!} Kt^{k-2},
y' = y^{(k-(k-1))}(t) = \frac{1}{(k-1)!} Kt^{k-1},
y = y^{(k-k)}(t) = \frac{1}{k!} Kt^k.
\]

Finally this produces,

\[
\frac{v^2}{ad} = \frac{y'}{y''} = \frac{(k-2)!k!K^2t^{2k-2}}{(k-1)!(k-1)!K^{2k-2}t^k} = \frac{k}{k-1}.
\]

**Equation 4.15**

(The answer here is the same sign when the directions are reversed.) Note that this general result includes the control using constant acceleration above as a special case, with \( k = 2 \) giving equation 4.13, and the use of exponential decay can be seen as the limiting case with \( k \) approaching infinity,

\[
\frac{v^2}{ad} = \lim_{k \to \infty} \frac{k}{k-1} = 1,
\]

as in equation 4.14.
If constant jerk is used to bring a system to rest, $k = 3$, then,

$$\frac{v^2}{ad} = \frac{3}{2}.$$  

**Equation 4.16**

This is exactly half way between the initial force required for constant acceleration and exponential decay.

The mathematics has been useful to show how larger force is needed to reduce rebound, and the formulas give some idea of the relative force increases that might be needed. There is the problematic matter however of the initial point. It has been assumed in this section that the initial force can be selected arbitrarily. This creates a problem which is examined in following sections. The important point of this section however is the theoretical recognition that there is a trade-off between smoothness and rebound, and the price for removing rebound (and potential instability) seems to be a larger control force. In the case of a discrete control this translates to higher control levels than may seem to be needed from energy considerations alone.

In summary, while the problem of rebound may be solved, there remains the problem of the “initial instant” (or the current time in a discrete feedback control). The problem with all the controls above is that no matter how smooth they are at later times, the control force at time zero is not generally continuous; the initial acceleration of the control does not necessarily match the initial acceleration of the given system. At the start there will be a spike in jerk as the acceleration changes instantaneously to match the acceleration required of the control.

### 4.7.3.2. Simple Two-Stage Controls

In the previous section the initial acceleration was not taken into consideration. In each of the controls in the last section the acceleration will change instantaneously initially, to correspond to the initial acceleration required of the control. A solution to this problem is to apply a two-stage control, with the first stage setting up conditions for smooth decay in the second stage, but with continuous acceleration at all times.
The approaches developed in this section are only very briefly outlined because much superior methods were discovered later in the research, and are covered in later sections. The controls developed here illustrate weaknesses that are solved in later sections.

Suppose constant jerk is applied in the first stage so that exponential decay can be applied in the second and final stage, as depicted in figure 4.25. Acceleration rises linearly in the first stage, while distance velocity and acceleration decay to zero in the second. In this example, constant jerk is first applied to create the conditions under which equation 4.14 holds, and exponential decay is then applied to approach zero without overshoot. Note that acceleration can be made continuous at all times, including the initial instant.

![Figure 4.25 Control with Exponential Decay Jerk](image)

(The graphs above have been generated using Java code. The software can be found in the Eclipse workspace, PhD\Eclipse RSpace Constraint\Edge Overshoot, discussed previously. The JerkControlDemo class contains the main routine and the above code can be run by clicking the button labelled “Specific Solution Exp”. Different initial values for displacement, velocity and acceleration, \(d\), \(v\), and \(a\), can be input using sliders.)
In order to implement a two-stage method, a time period for the first stage of control must be nominated. The time period for the first stage of the control will here be designated as “$T$”. In the example in figure 4.25, the first stage time was calculated as the time it takes for the system to cross zero displacement with no change in acceleration (that is, if acceleration were simply held constant at the initial value).

Given a value for $T$, the values of distance, velocity and acceleration at the end of the first stage must be such that they satisfy equation 4.14, allowing exponential decay in the second stage. If the initial distance, velocity and acceleration are $d$, $v$ and $a$ respectively (at the start of the first stage), and a constant jerk $j$ is applied, then the values of distance, velocity and acceleration after time $T$ are easily calculated by successively differentiating the formula for distance,

$$y(t) = d + vt + \frac{1}{2} at^2 + \frac{1}{6} jt^3.$$  

This can be solved simultaneously with equation 4.14. This has an algebraic (closed-form) solution: the jerk required to satisfy equation 4.14 in time $T$ is,

$$j = -\frac{2aT^2 + 6d \pm \sqrt{36d^2 - 12adT^2 - 12v^2T^2 - 12avT^3 - 2a^2T^4}}{T^3}.$$  

Equation 4.17

The derivation of this equation (using the algebraic software package, Maple 7) is shown in the appendix, in section 8.10.

A similar technique uses constant jerk for the second stage, rather than exponential decay. Figure 4.26 gives an example of this algorithm, with the same initial conditions as the example shown above in figure 4.25. (The example was generated by clicking the button labelled “Specific Solution” in the JerkControlDemo class.) At the end of the first stage, at time $T$, the values of displacement, velocity and acceleration must now satisfy equation 4.16. This was also solved (using Maple, as shown in section 8.10) and gives the analogous result to equation 4.17,

$$j = \frac{1}{2} \frac{4v^2 + 2avT + a^2T^2 - 6ad}{T(vT + 3d)}.$$  

Equation 4.18
Once the time period of the initial stage is selected this method is algebraically extremely simple.

![Graph showing displacement, acceleration, and jerk over time with labels for acceleration continuous throughout and jerk constant in second phase.](image)

**Figure 4.26 Return to Zero without Overshoot**

The above control gives an acceptable response, but it fails when used as a feedback control. A discrete control is “iterated” in the sense that the control value is recalculated from one time step to the next in a microprocessor (Simon, 2006). (This should not be confused with “numerical iteration” such as used in Newton’s method, which might be employed by a microprocessor within a single time step.) The control is discretised by calculating the jerk on a moment-to-moment basis, using the current conditions as the initial conditions, as was done successfully with the minimum-time method in section 4.6.1.

Figure 4.27 shows a comparison of the non-iterated control with the iterated control for the same starting conditions. (The iterative example was generated by clicking the button labelled “Cont. Pure Jerk” in the JerkControlDemo class.) Figure (a) shows the kind of trajectory that results from a control that is planned at the start, but figure (b) shows the trajectory that results when the control is implemented as a feedback control. The discretised version fails to bring acceleration to zero. This produces again the problem of residual rebound acceleration. This problem is caused by the fact that the estimate of the first stage time is not consistent from one step to the next.
This author’s first attempts to resolve this problem focused on the ratio, \( \frac{v^2}{ad} \), which was used above to distinguish the various methods of returning to zero smoothly, in section 4.7.3.1. Evolving from this idea, the control became more and more complex in order to deal with exceptions. Some experimentation produced an algorithm which gave acceptable results over a wide range of initial conditions. An example of the control with the same initial conditions as in figure 4.27 is shown below in figure 4.28. (Press the button labelled “Cont. Specific Soln.” in JerkControlDemo.) While the control is quite ungainly, it does have the property that acceleration decays to zero.
4.7.3.3. Landing-Surface Method

A much more elegant method was found, discovered by the author after experimenting with simulations. In a simple simulation program, control jerk values and initial conditions could be varied smoothly with a slider, while the minimum-time method was used to return to rest. (The code can be executed by clicking the button labelled “Min-Time Jerk” in the OvershootDemo class in the OvershootControl package.)

Recall that the minimum-time control over jerk generally has two intermediate switches of jerk value, as shown in figure 4.11 earlier. When the software was used to vary the jerk value being applied, it always happened that the least jerk value which did not produce overshoot occurred exactly at the point where there was only one intermediate switch of jerk, as in figure 4.29 (b) below. Note that in figure 4.29, the jerk profile should be inferred from the constant slope of the acceleration, as discussed in section 4.6, this is done to save space and to make the graphs less cluttered and complex.

When the absolute value of jerk is a little too large, as in figure (a), there is a jerk switch near the initial point. In this example the initial values of distance, velocity and acceleration are
0.4, 1 and -1.5 respectively. When the jerk is slightly too small, as in figure (c), overshoot occurs with a switch in jerk near the end. At the “Goldilocks point”, when the jerk is just large enough to avoid overshoot, as in figure (b), there is only one switch in control, and the system is on the landing surface (the landing surface is explained in section 4.6). After experimenting with a large number of different initial conditions the result was always the same. The least jerk required to avoid overshoot occurred exactly when only one intermediate switch of jerk was required: too much jerk and a switch appeared at one end, too little and a switch appeared at the other end. This result was entirely unexpected.
The method deriving from this result has been dubbed here the “landing-surface” method, because the initial point is on the phase-plane landing surface for the minimum-time control over jerk (refer to figure 4.9 in section 4.6). The landing surface is the surface in phase-space made up of the points that can reach zero with one switch of jerk in the middle. (The landing-surface control should not be confused with the iterative minimum-time control discussed above in section 4.6.1. The landing surface is being used here in a completely different way to find a control with no distance overshoot.)

A proof of this result was eventually found by this author, and it will be shown below that:

*If the initial conditions produce overshoot with no change in acceleration, then the minimum-time control that avoids overshoot and which employs the least jerk magnitude is the landing-surface control.*

That is to say, the minimum-time control that has the least jerk and which does not overshoot is the landing-surface method. The proof is discussed below in section 4.7.3.4.

To apply the landing-surface method it is necessary to find the jerk value such that the initial point is on the landing surface. This problem has been solved by the author in a number of ways, each method more efficient than the previous. The first method involved calculations of distances onto the landing surface given various values of jerk, until the initial point converged onto the landing surface. Another method involved varying the timing of the first
stage, rather than varying jerk. This method amounts to finding the intersection of two curves such as those shown in figure 4.30. The Java code for this method can be found in the appendix, in section 8.11.

An advantage of this method is that it can be modified easily so that the jerk of the final approach is lower than that used in the first stage. The method is slow to converge, however, and requires the calculation of a square root at every step.

Equation 4.16 can be used to estimate the jerk required in the final stages of decay. From equation 4.16, and from the fact that velocity and acceleration have opposite sign, it can be shown that,

\[ v_{est} = -\text{sgn}(a) \sqrt{\frac{3ad}{2}}. \]

This estimate can then be applied in the formula for the calculation of jerk,

\[ jerk = \frac{a^2}{2v_{est}}. \]

An example of this control is shown in figure 4.31, which is quite close to the ideal of figure 4.29 (b). The reason that this control works as an iterative control is explained below. (The graphs in this section were produced using the JerkControlDemo.java program in the JerkControl package.)
The numerical method developed here lends itself to finding a softer jerk in the second stage by altering the convergence condition. In the above, the condition of convergence applies when the jerk of the first and second stages are equal in magnitude,

\[ \text{Jerk}_1 = -\text{Jerk}_2. \]

By a simple change in this condition,

\[ \text{ReverseJerkFactor} \times \text{Jerk}_1 = -\text{Jerk}_2, \]

where \(0 \leq \text{ReverseJerkFactor} \leq 1\), the second stage jerk can be made softer than the first, generally at the expense of a small increase in the size of the initial jerk, as shown in figure 4.32. Here the magnitude of the second stage jerk is 0.3 times the magnitude of the first. Note that this also produces a quite flat jerk profile in an iterative method. (To demonstrate this algorithm, change the slider value of ReverseJerkFactor to a value between zero and one in the program OvershootDemo.java.)
4.7.3.4. Proof of Landing-Surface Method

The above methods are quite slow computationally. A faster method became possible after a proof had been developed by this author showing that the landing-surface control is minimal in the sense discussed above. The Newton-Raphson method can be applied (Kreyszig, 1993, p929) to equations derived in the proof. The proof is outlined in this subsection, with the details transferred to an appendix (in section 8.14).

Figure 4.33 Switching Times
Refer to figure 4.33 above. Suppose that the initial conditions are on the landing surface (there is only one intermediate switch of jerk). The system reaches rest (zero distance, velocity and acceleration) at time $t_E$ and the intermediary switch occurs at time $t_I$. The initial distance, velocity and acceleration are represented as $d$, $v$ and $a$ respectively. Without loss of generality the initial distance is assumed positive, $d > 0$.

The magnitude of the jerk used in the method is represented as $j$. Thus, in the example depicted in figure 4.33, the jerk from time zero to time $t_I$, is constant at $j$, and the jerk of the final stage, from $t_I$ to $t_E$, is $-j$. The case where initial jerk is negative and the second is positive can be easily dismissed since this implies that the displacement is negative in the second stage and thus overshoots. Hence $j$ is a positive parameter of the method.

Since the initial point in phase-space, $(d, v, a)$, must lie on a landing surface, there are restrictions on $d$, $v$, and $a$ for a given $j$. For the proof of the optimality of the landing-surface method, and in the numerical method for applying it, it is useful to define a function $j_L(d, v, a)$, which represents the jerk of the landing surface method as a function of initial distance, $d$, velocity, $v$, and acceleration, $a$. It can be shown that $j_L$ is well defined, at least when the initial point results in overshoot with zero control jerk (otherwise there is an easy no overshoot control, using zero jerk). (If there were more than one jerk value the supremum could be selected, but it becomes clear later that the jerk value for the point to be on the landing surface is unique.)

The following result can also be shown:

*The function $j_L(d, v, a)$ is continuous, monotonic decreasing as a function in $d$, and also,

$$\lim_{d \to \infty} j_L = 0.$$*
require advanced mathematics and the details of the proof are shown in the appendix, in section 8.14. This also shows that the value of $j_L$ is uniquely defined.

The period from $t_1$ to $t_E$ is easy to characterize. At time $t_E$ the final values of acceleration, velocity and distance are zero simultaneously. Also in this period jerk is constant, so in the time between $t_1$ and $t_E$, the values of distance and acceleration must be opposite in sign to velocity and jerk. Assuming distance is positive, jerk and velocity are negative. This also shows that the opposite jerk in the initial period, up to $t_1$, must be positive.

It is shown next by contradiction that the displacement on the landing surface cannot drop below zero prior to the time $t_1$. Suppose then that distance is negative at some time between the initial point (at time zero, $d=y(0)>0$ by assumption). There is therefore a local minimum between time zero and $t_E$. Because distance is positive at time $t_1$, $y(t_1)>0$, and the slope at time $t_1$ is negative, $y$ must have a local maximum some time after the local minimum and before $t_1$. This implies that the jerk before $t_1$ is negative (the only way for a third-order polynomial to have a minimum followed by a maximum with negative final velocity is for the third-order coefficient to be negative). But it has been shown in the previous paragraph that jerk is positive in the first phase, so this is impossible. Thus the assumption of this paragraph, that the distance can be negative before time $t_1$, is false. Thus it has been shown that the displacement is positive before $t_1$ and positive between $t_1$ and $t_E$, therefore the displacement is positive at all times: there is no overshoot on the landing surface.

The proof for the result given in section 4.7.3.3 can be now shown. The result is restated as:

*If the initial conditions produce overshoot with no change in acceleration, then the minimum-time control that avoids overshoot and which employs the least jerk magnitude is the landing-surface control.*

Using proof by contradiction, suppose first that the required control does not begin on a landing surface. The initial point is either above or below the landing surface. If it is below
the landing surface then the first jerk of the minimum-time method is positive (as discussed in section 4.6.1), the second jerk is negative, and the final jerk is positive. In this case, in the final stage of the minimum-time method jerk is positive, acceleration is negative, velocity is positive, and distance is negative, as seen in the final stage in figure 4.29 (c). Negative distance implies overshoot, contradicting the assumption that the initial point is below the landing surface.

Suppose then that the initial distance, \( d \), is above the landing surface. Let \( j_m \) be the supposed minimal jerk of this assumption. Suppose that the landing surface distance directly “below” the initial point \((d, v, a)\) is \((d_m, v, a)\) with \( d_m < d \). Thus the jerk used on the landing surface of the minimal method can be written as,

\[
j_m \overset{\text{def}}{=} j_L(d_m, v, a).
\]

The landing surface method at the initial point would use a jerk given by,

\[
j_0 \overset{\text{def}}{=} j_L(d, v, a).
\]

However, from the above lemma, \( j_L \) is a decreasing function of \( d \), and \( d_m < d \), giving,

\[
j_0 = j_L(d, v, a) < j_L(d_m, v, a) = j_m.
\]

Now the landing-surface method using \( j_0 \) is itself a method reaching zero without overshoot, since it has been shown above that the landing surface does not overshoot. Furthermore it uses lower jerk than \( j_m \), contradicting the assumption that \( j_m \) is minimal. This then contradicts the possibility that the initial point is above the landing surface.

The initial point in phase space is neither above nor below the landing surface so it is on the landing surface. In fact, the function \( j_L(d, v, a) \) can be used to find the jerk needed for the initial point to be on the landing surface (refer to appendix 8.14).

Because the landing-surface method is also a minimum-time method, it must also be a minimum-time method which avoids overshoot (using the jerk value of the landing surface) since it already has the property that it avoids overshoot. All minimum-time controls using larger jerk than the landing surface control also avoid overshoot, but at the cost of smoothness.
Optimality explains the observed consistency when used as a feedback control. “If [control] $u$ is optimal on $[t_1, t_2]$, it is optimal on every subinterval” (MacCluer, 2005, p121). As noted in section 2.3.2.3, this is, in fact, an expression of Bellman’s “principle of optimality”. Because of this principle, the optimal control translates to a consistent RTOC.

### 4.7.3.5. Skim Methods

The landing-surface method uses the least jerk of all minimum-time controls that do not overshoot, but is it minimal of all controls? Is it the control which avoids overshoot using the least maximal jerk?

In fact, it is not minimal in this sense and the actual minimal control is quite easily determined. Suppose a constant positive jerk $j_s$ is applied to keep distance positive. This gives the following values for acceleration, velocity and distance:

$$
\ddot{y}(t) = a + j_s t,
$$

$$
\dot{y}(t) = v + at + \frac{1}{2} j_s t^2,
$$

$$
y(t) = d + vt + \frac{1}{2} at^2 + \frac{1}{6} j_s t^3.
$$

**Equation 4.19**

Set both distance and velocity simultaneously to zero at some time $T$: $y(T) = 0$ and $\dot{y}(T) = 0$. (Note that acceleration is not set to zero.) Now $j_s$ can be eliminated from the equations for $\dot{y}$ and $y$ giving a quadratic in $T$ as below,

$$
\frac{1}{2} a T^2 + 2vT + 3d = 0.
$$

Solving the quadratic for $T$ and choosing the least positive solution gives,

$$
T = \frac{-2v \pm \sqrt{4v^2 - 6ad}}{a},
$$

It can be shown that, in fact, in the case that $d > 0$, the negative always applies:

$$
T = \frac{-2v - \sqrt{4v^2 - 6ad}}{a}.
$$

Using this, $j_s$ is derived from equation 4.19 by setting $\dot{y}(T) = 0$ and solving to give,
\[ j_S = -\frac{2(v + aT)}{T^2} = \frac{2a^3v}{(2v + \sqrt{4v^2 - 6ad})^3}. \]

At the point that the trajectory skims the time axis, both distance and velocity are at zero, but there will generally be some remaining rebound acceleration, \( \dot{y}(T) \neq 0 \). This remaining acceleration will be dealt with below.

At this point, it is easy to show that the truly minimal control for avoiding overshoot (but not coming to rest with zero acceleration) must use jerk \( j_S \) up to the time of skimming the time axis, \( T \). Suppose that there is a jerk control, \( j_m(t) \) which does not overshoot but which has a smaller jerk than \( j_S \) at all times before time \( T \):

\[ \sup_{0 \leq t \leq T} j_m(t) < j_S. \]

Suppose that \( j_m(t) \) is integrable, then,

\[
y_m(T) = d + vT + \frac{1}{2}aT^2 + \int_0^T \int_0^\alpha \int_0^\beta j_m(\tau) d\tau d\beta d\alpha
\]

\[
< d + vT + \frac{1}{2}aT^2 + \int_0^T \int_0^\alpha j_S d\tau d\beta d\alpha = d + vT + \frac{1}{2}aT^2 + \frac{1}{6}j_S T^3 = y(T) = 0.
\]

And so the method overshoots, contradicting the assumption. Thus \( j_S \) is the minimum jerk that avoids overshooting the time axis. Furthermore, it is clear that \( T \) is the minimum time to intersect the time axis without overshoot, using jerk no larger than \( j_S \).

When overshoot does not occur with zero control, a negative value of \( j_S \) will also produce a skim. However, the case that \( j_S \) is negative, controls for a minimum-time control with lower values of control jerk than \( |j_S| \) will not produce overshoot, but larger values may. In fact the minimum-time method (with no distance constraint) will then overshoot for jerk values between \( |j_S| \) and \( j_L \).
Figure 4.34 shows an example of a skim compared to a landing-surface control. Here the initial values are $d = 1$, $v = -1.5$ and $a = 0.3$. For these values the value of $j_S$ is approximately $j_S \approx 2.1233$.

**Figure 4.34 Landing-Surface compared to Skim**

The problem for the skim method though is that some rebound acceleration generally remains at the point that the trajectory skims the time axis, similar to the rebound problem discussed previously in section 4.7.3.1. The quickest way to deal with this remaining acceleration is to employ a minimum-time control over jerk to bring the system back to zero, as in figure 4.35. This control can be added to control the rebound, immediately after the skimming of the time axis.

**Figure 4.35 Removing Remaining Acceleration**
Any value of jerk can be used for this control without producing overshoot, even infinitesimally small values. However, the time taken using such a control is given as,

\[ T_R = \frac{(1 + 2z)a_R}{j_R} \approx \frac{4.39031a_R}{j_R}, \]

**Equation 4.20**

where \( a_R \) is the acceleration remaining after applying \( j_S \), \( j_R \) is the jerk applied, and \( T_R \) is the time taken to reach zero in this portion of the control. (Click on “J vs T” in Overshoot Control Demo.) Here \( z \) is the positive root of \( 12z^4 - 24z^2 - 16z - 3 \),

\[ z = 1.69515635. \]

(Refer to Maple file PhD/Maple Experiments/MinTimeZeroInitialDistAndVel 02.mws.) Furthermore, the maximum rebound distance is found to be,

\[ d_{max} \approx 0.66665443 \frac{a_R^3}{j_R^2} \approx \frac{2a_R^3}{3j_R^2}, \]

and occurs at time,

\[ T_{max} \approx 2.0000247 \frac{a_R}{j_R} \approx \frac{2a_R}{j_R}. \]

**Equation 4.21**

From equation 4.20, the jerk value which returns the system to zero in minimum time with jerk not larger than \( j_S \) must use jerk value \( j_R = j_S \). Figure 4.36 shows the full skim control including this control of rebound after the initial skim.
Figure 4.36 Skim showing Rebound, compared with Landing-Plane

In the example shown above the values of jerk for the landing-surface method, \( j_L \), and for the minimum-jerk skim method, \( j_S \), are,

\[
j_L \approx 2.50835 \quad \text{and} \quad j_S \approx 2.12331
\]

The jerk for the skim method is lower, but the rebound takes a very long time and, in this example, even travels back beyond the initial point. Depending on the application this could induce instability.

There are some initial conditions for which skim jerk can be somewhat lower than landing-surface jerk, \( j_L \). With initial conditions \( d = 1.8, \ v = -1.8 \) and \( a = 0.85 \), the jerk values are somewhat further apart:

\[
\begin{align*}
\ j_L & \approx 0.488759, \quad \text{and} \quad j_S \approx 0.0760144.
\end{align*}
\]

However, the rebound time and rebound distance for the skim method are extremely large in this case.

There is a continuous range of controls with jerk values between the landing surface control, \( j_L \), and the minimum-jerk skim control, \( j_S \). For a given value of jerk, \( J \), that lies between \( j_L \) and \( j_S \) a two-stage control can be applied, using a constant positive jerk, \( J \), until such time as the conditions require a constant negative jerk of \( -J \) to just skim the time axis. This
is clearly simple to apply as a feedback control and numerical experiments verify that this works as a closed-loop iterative method. Figure 4.37 shows a comparison using three values of control jerk. The initial conditions are $d = 0.4$, $v = -0.4$ and $a = 0.1$. The three controls are the landing surface control, the minimum-jerk skim control, and an example of a skim control with a control jerk value intermediate between the two extremes. (Click on “Newton vs. Skim” in Overshoot Control Demo.)

![Figure 4.37 Various Skim Methods Compared](image)

The full range of such controls will here be considered the definition of a “skim” control. Note there are three stages in the general skim control: stage one is a period of increasing acceleration, using nominal jerk $J$; stage two switches from positive jerk to negative, $-J$, at exactly the point when this is required to just skim the time axis; and stage three occurs just after skimming the time axis and comprises the rebound, using the standard minimum-time control with jerk magnitude $J$, removing the acceleration at rebound.

For lower jerk values, the skim controls have larger rebound, given the same initial condition. At the lowest jerk limit, the skim control that has no initial stage becomes the minimum-jerk skim control. At the other extreme, with the landing-surface control, there is no rebound at all. The controls can be extended for jerk limits greater than $J_L$ simply by including the standard minimum-time method, which do not overshoot for these jerk values.

This complete range of “skim” methods then is conjectured to be the full range of minimum-time methods that have distance and jerk constraints that return to rest without overshoot at zero distance. That is, they are conjectured to be the full range of solutions to the following problem.
What is the minimum-time control that returns to rest (zero distance, velocity and acceleration) with constrained jerk,

\[ |\dot{y}(t)| \leq J, \]

and which does not overshoot zero displacement, that is,

\[ y(t) \geq 0, \]

for all \( t \geq 0 \)?

The two extremes, the landing-surface control and the minimum-jerk skim control, have been proven to be minimum time here, and of course, the standard minimum-time controls are minimum time. A proof of the wider claim, including the controls between the minimum-jerk skim and the landing-surface control, is more difficult. What follows in the next few paragraphs is a condensed version of a more detailed discussion given in the appendix, in section 8.16.

The general “no overshoot” problem is constrained in the state-space: \( y(t) \geq 0 \). Pontryagin’s method does not apply to problems with state-space constraints, although the performance criteria can be modified so that solutions only occur within the state-space constraints (Kirk, 1970, pp237-8). This method may be useful numerically, but it does little to assist with an analytical solution. The approach taken by this author is to divide the problem into two portions, with the point of skim forming the division. It then remains to show that any small admissible perturbation in the initial section (one that does not cause overshoot) produces only an overall increase in the total time.

The first stage of control must be equivalent to a minimum-time control, unconstrained by displacement, which ends in zero distance and velocity, but with a non-zero “rebound” acceleration. Since this problem has the same adjoint equation as equation 4.12, the same conclusion results: there are at most two switches in jerk.

A small perturbation then can only take the form of the introduction of a switch a small time after the start, or a small time before the end (with concomitant changes in the other switching times, constrained by the fact that distance and velocity must be zero at the end). Such perturbations, with finite differences in control value but small timing differences, are
allowed as perturbations in the proof of Pontryagin’s minimisation principle (Mesterton-Gibbons, 2009, p169; Pontryagin et al., 1986, p87).

Suppose the control is perturbed by switching a small time after time zero, $\delta t_0$, from $-J$ initially to $J$. When the change in overall time is graphed (such as shown in figure 8.14) it is apparent, in at least all cases examined by the author, that there is an increase in overall time. This of course does not constitute a proof. It needs to be shown algebraically that the rate of change of overall time with respect $\delta t_0$ is positive, and this also needs to be shown for small perturbations “at the end”.

If this is shown then the minimum-jerk skim and the landing surface methods, which have been proven independently, are special cases of skim controls. Thus there is a compromise. Larger jerk values decrease comfort, but they produce smaller rebound, and quicker overall time to zero. Thus there is a trade-off between smoothness, rebound, and overall time.

The extent to which rebound represents a problem depends on the application. For example, a small amount of rebound might not be a problem in a suspension avoiding a rattlespace limit, since a suspension needs to move to the centre of the rattlespace in any case. On the other hand, a landing surface may be exactly what is needed in other contexts, such as a robotic arm movement up to a delicate piece of machinery. Skim methods with very low jerk control values might also prove useful near rest to deal with small state-estimation errors, numerical round-off errors or small vibrations. On the other hand, some physical implementations may be incapable of producing rebound, such as semi-active controls in a configuration that is limited by the passivity constraint.

Consider now a system which is constrained on two sides (such as in a suspension, or when considering error corrections). Suppose that a control is applied to skim from one side to the other until an acceptably smooth standard minimum-time method can be used within the rattlespace, as depicted in figure 4.38. In this method, different values of jerk magnitude, $\dot{J}_1$ to $\dot{J}_n$, are used with skim methods applied in stages between each edge. (The arrows in the diagram indicate that the controls are opposite in direction; there are jerk switches in each region.)
If the minimum-jerk skim is used to target each opposite edge, then it is easy to show that the system will oscillate forever from one edge to the other with jerks equal in magnitude, $j_1 = j_2 = j_3 = \ldots$, and with the acceleration at each edge being the same in magnitude but opposite in direction. At the other extreme, if the landing-surface jerk is used, the system will come to an abrupt stop at the first edge encounter. Using skim methods with jerk values between the landing-surface and the minimum-jerk skim, the system can be made to decay in a controlled way with $j_1 > j_2 > \ldots > j_n$. Acceleration at the opposite edge decreases with each traversal. Finally, a minimum-time method brings the system to rest with an acceptable jerk value. The kind of decay used will depend on the needs of the application, specifically on the relative importance of smoothness compared to settling time: a smoother control will take longer to reach zero.

It takes little imagination to see that the method can be adapted to coming to rest without actually skimming rattlespace limits, but turning at convenient distances within the rattlespace.

What this shows is that jerk control at the level of minimum-jerk skim will oscillate forever within the rattlespace. The landing-surface jerk on the other hand is too large and brings the system to rest at the rattlespace edge. We conjecture that a suspension control using jerk will therefore use jerk values in this range, or perhaps slightly larger. If larger values are needed they might be skim jerk values that apply to a smaller rattlespace, say one third the wider width.
4.7.3.6. Applying the Landing-Surface Method

To apply the landing-surface method numerically in a discrete control, it is necessary to solve a fourth-order equation. Refer to the appendix, 8.14. In the case of positive initial acceleration, case 1, the following equation holds,

\[
\frac{ad}{v^2} = \delta_i \left( \sqrt{\beta} \right)
\]

Equation 4.22

where the function \( \delta_i \) is defined as,

\[
\delta_i(x) = \frac{2(4 - 6x^2 + 3x^3)}{3(2 - x^2)^2},
\]

Equation 4.23

and \( \beta(j) \) is defined as,

\[
\beta(j) = 2 - \frac{4vj}{a^2}.
\]

Equation 4.24

To find the jerk for the landing-surface method requires first solving the fourth-order polynomial given by equation 4.22 and equation 4.23. Closed-form solutions can be calculated for fourth-order equations, as shown by Dixon (2008, pp385-91), but the solution is quite cumbersome. “In practice, numerical iteration methods may be as good in this case, although the analytic method ... is more predictable in computation time (Dixon, 2008, p391).” The numerical Newton-Raphson method has been used in this thesis. The iterative method has some scope for improvement in efficiency. For instance, the equation has really only one varying parameter, \( ad / v^2 \), so there is some scope for the use of look-up tables to provide initial estimates. There may also be the possibility of using previous values in initial estimates in a real-time feedback control. These refinements were not investigated here.

Using the Newton-Raphson method, equation 4.22 can be solved for a value of \( x \) that solves \( \delta_i(x) = ad / v^2 \). Each step requires the following calculation,
\[
x_{n+1} = x_n - \frac{4a}{2v^2} \frac{\delta_1(x_n) - \frac{ad}{v^2}}{\delta_1'(x_n)} = x_n - (2-x_n^2) \frac{2(4-6x_n^2+3x_n^3) - 3\frac{ad}{v^2}(2-x_n^2)^2}{2x_n(-8+18x_n - 12x_n^2 + 3x_n^3)}.
\]

Once \(x\) is found, this is squared to give \(\beta\) (defined in the appendix in section 8.14) and the jerk is derived from the equation for \(\beta\),

\[
\dot{\beta} = \frac{a^2}{4v}(2-\beta).
\]

The most computationally complex operation in each step of Newton’s method is a single division. This is a considerable improvement over the numerical methods developed previously in section 4.7.3.3 which required a square root in each step, and which converge much more slowly.

In the case of negative acceleration, case 2, the iteration step requires the calculation,

\[
x_{n+1} = x_n - \frac{4a}{2v^2} \frac{\delta_2(x_n) - \frac{ad}{v^2}}{\delta_2'(x_n)} = x_n - (2-x_n^2) \frac{2(4-6x_n^2-3x_n^3) - 3\frac{ad}{v^2}(2-x_n^2)^2}{-2x_n(-8+18x_n + 12x_n^2 + 3x_n^3)},
\]

where,

\[
\delta_2(x) \overset{\text{def}}{=} \frac{2(4-6x^2-3x^3)}{3(2-x^2)^2}.
\]

Equation 4.25

Asymptotes exist near points where \(\beta = 2\) causing convergence problems, but it is a simple matter to trap when these problems occur and move closer to a region of convergence. This is helped by the fact that the signs of initial acceleration and velocity determine whether \(\beta < 2\) or \(\beta > 2\). The method used is to calculate a new value of \(\sqrt{\beta}\) closer to \(\sqrt{2}\) using a formula of the form,

\[
\sqrt{\beta}_{n+1} = \frac{2k + \sqrt{\beta}_n}{k+1},
\]

where \(k\) is some number larger than or equal to 1 (10 was used in the author’s code).

The cases \(a = 0\) and \(v = 0\) are also easily dealt with by applying the results for these special cases when \(a\) or \(v\) is near zero (see section 8.14).
When $\frac{ad}{v^2}$ is very large, convergence is slow but very close approximations can be found.

For example, when acceleration is positive and $x$ is large, the following is a close approximation,

$$\delta_1(x) \approx \frac{2}{x}.$$  

(The Java code for the entire method can be found in the procedure “jerkForLandingSurface” in the `SwitchPlane01.java` class definition in the Overshoot package.)

An alternative method for finding the control parameter is of less value numerically, but it is perhaps worth mentioning. It is simpler algebraically, but it has a slower convergence. Recall from section 4.7.3.2 that equation 4.18 was derived by assuming that the end conditions of the first stage were such that constant jerk could be applied to bring the system to rest, up to zero acceleration. Solving for first-stage jerk in terms of initial conditions produces the following variant of equation 4.18,

$$j_1 = \frac{1}{2} \frac{4v^2 + 2avt_1 + a^2t_1^2 - 6ad}{t_1(vt_1 + 3d)}.$$  

Recall that the second stage jerk could be different from the first. Now set just the final velocity and acceleration to zero and solve for jerk magnitude in the final stage,

$$j_2 = \frac{-\ddot{y}(t_1)^2}{2\dot{y}(t_1)} = \frac{(2v + at_1)^3}{2(3d + vt_1)(at_1^2 + 4vt + 6d)}.$$  

Equating $j_1$ and $j_2$ gives the value of $t_1$ as the root of the equation,

$$t_1(2v + at_1)^3 + (4v^2 - 6ad)(3d + vt_1) = 0.$$  

**Equation 4.26**

This equation can also be solved using Newton’s method. This method was not used here (although the solution is calculated in the method `jerkForLandingSurfaceGamma` and can be seen running by pressing the button labelled “Newton” in `OvershootDemo.java`). Also the former method uses functions $\delta_1$ (equation 4.23) and $\delta_2$ (equation 4.25) that have only one parameter and which thus may allow further optimization using look-up tables and other methods.
Applying the landing surface method as a real-time feedback control produces results similar to those shown in section 4.7.3.3. Figure 4.39 shows an example of an application of the method, with initial conditions, \( d = 1.5, \; v = -1 \) and \( a = -0.5 \). Distance has been colour-coded in the diagram according to the stage control being applied.

![Iterative Landing-Plane Method Diagram](image)

**Figure 4.39 Iterative Landing-Plane Method**

The jerk of the second stage of the iterated landing-surface method needs to be estimated when used in a feedback control. Suppose \( d_c, \; v_c \) and \( a_c \) are the current distance, velocity and acceleration, at a time after the first stage. Suppose that jerk \( j_2 \) is used to bring velocity and acceleration to zero at time \( t_2 \) (after the current instant). Solving for \( t_2 \) and \( j_2 \) gives,

\[
\begin{align*}
  t_2 &= -\frac{v_c}{a_c}, \\
  j_2 &= -\frac{a_c}{t_2}.
\end{align*}
\]

This method when iterated produces small round-off errors at the end of the final stage, shown in the figure above. Even so the second-stage jerk is very flat. It is vastly superior to the iterated methods used in sections 4.7.3.2 and 4.7.3.3 which failed completely to work as iterative methods, or were cumbersome and slow.

To deal with the round-off effects the distance remaining when the velocity and acceleration reach zero can be estimated as,
\[ d_R = d_c + v_c t_2 + \frac{1}{2} a_c t_2^2 + \frac{1}{6} j_2 t_2^3. \]

A small jerk can be added to \( j_2 \) in the opposite direction to \( d_R \) using,

\[ j_2 = j_2 - \alpha \frac{d_R}{t_2^2}. \]

For small errors only a small value for the proportionality factor, \( \alpha \), is needed. The method can be iterated improving the distance estimate, but since the value should be small this is only done twice in the control demonstrated below. The result is a smoother response near zero, as seen in figure 4.40, although there are still some round-off artefacts to deal with. Problems resulting from small values should be dealt with by using code to identify when round-off produces division by near-zero values, or methods that “capture” the final state in a very small “rattlespace” could be used.

Figure 4.40 Example of Feedback Landing-Surface with Distance Error Correction

At any rate, the small errors of the final stage are not a concern in suspension applications or some semi-active systems. With a suspension, the control should seek the centre before reaching a rattlespace edge, and a semi-active system that reaches zero cannot supply rebound forces.
4.7.3.7. Control Variants for Increased Speed

The control outlined in the previous section can be used for the delicate final “docking” approach. Because the method above can determine the jerk value required for the control in real-time, it can be applied only when the jerk required reaches a nominated value. Thus any other control can be used initially to speed up the approach trajectory. If the jerk required is already larger than the nominated value then the landing surface method will give as smooth an approach as possible, in terms of jerk magnitude, without rebound. Because the jerk level of the landing-surface control is known, and because other parameters, especially maximum acceleration, are easily calculated, the landing-surface method can be used to complete what might otherwise be a very difficult manoeuvre.

Specifically, suppose the jerk magnitude, $J_B$, is nominated as being acceptably soft for comfort and for protection against damage due to vibration in the final approach. Suppose also that at the initial point, the landing surface method requires less jerk than $J_B$, as will almost certainly be the case when initialising a point-to-point movement. Then any other control could be used, until the point that $J_B$ is needed for the landing method. When this point is reached, the landing-surface method takes over for the complex final approach. This combination allows both fast and acceptably smooth movement.

In a real-time feedback control landing surface jerk required is calculated at each step. If the jerk is less than $J_B$, then apply the alternative method, otherwise use the landing-surface control. Any control prior to this could be used: linear jerk control, or even just constant negative jerk, $-J_B$. Different applications may need different constraints resulting in subtly different controls.

In the literature, jerk controls over robotic arm movement exploit control symmetries (Muenchhof and Singh, 2003; Ben-Itzhak and Karniel, 2008; Peters, 1995). Muenchhof and Singh restrict their attention to controls that are “point-symmetric about the mid-maneuver time” (Muenchhof and Singh, 2003, p140). The inclusion of the landing-surface control, however, handles the problem using feedback control and deals with the most difficult part of the manoeuvre, the final “docking” approach.
An example of a comparison between the pure landing-surface method (press “Cts. Land Test” in Overshoot Control Demo) and a modified method (press “BOB2”) is shown in figure 4.41. The initial conditions of this example are, \( d = 2, \ v = -1, \) and \( a = 0.1, \) and the nominated value of \( \mathcal{J}_B \) is 1. The system returns to rest more quickly; in approximately 3 seconds as opposed to 4.5 seconds.

(a)

(b)

**Figure 4.41 Comparison:** (a) Landing-Surface, (b) BOB with Improved Time

In a similar way the maximum acceleration of the landing-surface method is easily calculated and can be used to implement a constraint on acceleration. The maximum value of the landing-surface acceleration is given as,

\[
j_L(t_E - t_1)
\]
Thus the method applies the landing surface method only when the system reaches a state where the maximum acceleration required by the method reaches or exceeds some nominal value, say $A_B$. Similarly, when using $-J_B$ the jerk should stop when acceleration reaches $-A_B$. Figure 4.42 shows the use of an acceleration limit. (The examples are run using button “BOB2”.) The initial conditions are $d = 1$, $v = 0$ and $a = 0.35$. The jerk limit is $J_B = 1.5$, and the acceleration limit is $A_B = 0.35$. Figure (a) has a jerk limit only and figure (b) shows that a smoother response is yet again achieved at the sacrifice of speed.

Figure 4.42 Figure (b) has an Acceleration Limit
Recall from the previous section (section 4.7.3.6) that the final stage of the iterative control uses an estimate of the time taken and jerk required for the remainder of the very final stage:

\[ t_2 = -2 \frac{v_c}{a_c}, \]
\[ j_2 = -\frac{a_c}{t_2}. \]

Here substantially the same calculations are performed as in the previous section except that if \( t_2 \leq 0 \), then the control is not in danger of overshoot and jerk back to zero can be applied, \(-J_B\). Also when the control has a large residual distance (\( d_R \) in the previous section) or if it takes a long time to reach zero, then return jerk is controlled to remain lower in magnitude than \( J_B \).

4.7.4. Edge Constraint

The above theory can now be applied to suspension systems. Firstly, in this section two general approaches developed by this author are explained which, for the sake of discussion, are here termed “edge constraint” and “variable hardness”. These methods have developed relatively late in the thesis research, but there are a few numerical algorithms applied in the numerical experiments using these methods.

The basic idea of edge constraint is to apply a control of some kind to avoid an imminent collision with one of the rattlespace limits. Suppose that the chassis height trajectory is controllable, while the road height as a function of time is not. The idea is that the future trajectory of the chassis should not overshoot the approaching rattlespace limit, which matches road movement (ignoring the tyre) and is stochastic. The problem discussed in the previous section, of reaching a given distance with no overshoot, was inspired by the need to control a suspension chassis as it approaches a rattlespace limit, either top or bottom. As mentioned earlier (in section 4.7.1) one major difficulty is trying to predict the future movement of the rattlespace, since it is stochastic.
The edge constraint should be mixed with an acceptably soft return to equilibrium when there is little danger of a collision with a rattlespace edge. It has been found that if this is not included the control will tend to wallow between the edges without reaching equilibrium. Thus, the control should be able to prioritize whether to avoid an edge collision or softly return to equilibrium. This decision is complex, and among other things it depends on the control strength needed at future times. Ultimately, it also depends on the statistical properties of the road, as noted in section 4.7.1.

Inside the rattlespace, the jerk needed to avoid a collision with the rattlespace edge can be calculated for both the top and the bottom edges. Consider the simple method of using the value of jerk with the largest magnitude, on the assumption that the greater the danger of rattlespace collision, the greater the jerk value. Figure 4.43 shows an example of this control under relatively smooth conditions. Clearly, the simple method stays within the rattlespace but, as might be expected, it does not seek the centre.

![Figure 4.43 Edge Avoidance](image)

When the edges have rough corrugations, as in figure 4.44, it may be helpful to target smooth edges that lie inside the rattlespace, thus avoiding over-reacting to small corrugations. Momentarily high edge velocities do not necessarily indicate an impending large bump. This
is a statistical matter, but a rough approximation can be made by smoothing the edge. A two parameter method was used to give the results shown in figure 4.44. The method employed a simple formula to find a given proportion of the distance between the chassis and the rattlespace edge.

Using the example of the top rattlespace edge, the height of this point is given as,

$$A = y + \text{PROP\_REL}((r + R) - y),$$

where $y$ is the chassis height, $r$ is the height of the centre of the rattlespace, $R$ is half the rattlespace width and PROP\_REL is the proportion factor, between zero and one. Another proportional factor, PROP\_ABS, is the absolute proportion between the rattlespace centre and the edge and is a limit on PROP\_REL. The code for performing these calculations can be found in,

`Overshoot\RattlespaceTargetX.java`.

(A demonstration of this code can be run by pressing the “Continuous Control button” in the OvershootDemo program.)

Employing this algorithm produces the results shown in figure 4.44 (using “RCE” as the “baseAlgorithm” parameter. Here the parameter values are, PROP\_ABS=0.5 and PROP\_REL=0.4.)
Figure 4.44   Edge Avoidance with Edge Targets

Further refinements can be made to take account of edge velocity. In the first edge collision in figure 4.44, for example, the rattlespace edge is clearly moving quickly towards the chassis. When a large bump does occur, the suspension should respond accordingly. The best approach would analyse edge statistics to determine how best to distinguish between edge corrugations and large bumps. Again a simple heuristic was adapted here.

The method used here employs only one parameter COLLISION_TIME. Firstly, an estimate of the current edge velocity is found. It is then easy to estimate the time to edge collision using current chassis height and rattlespace edge height and velocity. Suppose $t$ represents this estimated time to collision, the interval between zero and COLLISION_TIME, then the following formula is used to provide an edge velocity measure for the edge avoidance algorithm that discounts when collision is not imminent:

$$v_{\text{use}} = \left(\frac{\text{COLLISION\_TIME} - t}{\text{COLLISION\_TIME}}\right) \frac{v_E}{\text{COLLISION\_TIME}}.$$  

Here $v_E$ is the estimate of the current edge velocity, and $v_{\text{use}}$ is the velocity estimate. COLLISION_TIME is the single parameter of the method. With this formula, the closer the likelihood of edge collision, the closer the edge velocity estimate is to the true edge velocity. (The code for this algorithm can also be found in, Java\TwoJerksEdgeX\RattlespaceTargetX.java.) This produces a vast improvement on the capacity to avoid edge collisions as can be seen in figure 4.45, where a COLLISION_TIME value of 0.5 seconds was used.
There is a trade-off. With small values of COLLISION_TIME, large bumps are not recognized until very late. With large values, the suspension can over-react to small corrugations, mistaking them for large bumps. Again, the optimal value depends on road statistics, but robust, acceptable values might be determined by an evolutionary algorithm.

The control for return to zero could be any soft control such as linear control over jerk or a minimum-time control over jerk (see section 4.6). In its simplest form, this needs just one parameter value, the absolute value of jerk. This parameter will be represented as COMFORT_JERK. In order to determine if this is too soft, and is likely to hit the rattlespace edge, it is necessary to be able to determine the maximal and minimal travel of the minimum-time method. This might not be a trivial calculation in the case of the minimum-time method, but modern microprocessors should be more than equal to the task.

The determination of which algorithm to employ, either the “hard outer” controls or the “soft inner” control, can be based on the projected trajectory of the soft inner control. The algorithm will switch to the hard outer control if the soft control is likely to collide with the rattlespace limits. Furthermore, if both upper and lower limits lie outside the rattlespace, it is
the limit which occurs first which determines which edge avoidance is applied. A simulation showing the application this algorithm is given in figure 4.46. Periods of the inner control are shown in red on this graph.

Just as corrugations can cause over reactions in the outer control, so they can disrupt what should be a smooth inner control. Smoothing of the target can be performed using a low-pass filter (which is easily implemented in a discrete control using a moving average as in section 8.12). Such smoothing is shown in figure 4.46. (The algorithm for road smoothing can be found in file,  

Java/TwoJerksEdgeX/RoadFilter.java.

The code for the version of the dual-mode algorithm as outlined here can be found in the class definition file,  

Java/TwoJerksEdgeX/EdgeTargetWithMinTimeX.java.)

In the example, the value of comfort jerk is 1.25. The decay rate for the rattlespace centre is 1.05 and the decay rate for rattlespace centre velocity is 0.1.

![Figure 4.46 Dual-Mode Control](image-url)
This variant has been implemented for numerical experiments. More sophisticated variants are possible and more research is needed.

### 4.7.5. Variable Hardness

An alternative approach for rattlespace constraint is to vary a control’s “hardness” so that it stays within the rattlespace limits. So for example a linear control could momentarily increase its linear coefficients to ensure travel remains in the rattlespace. The iterated minimum-time bang-bang control could also be used. The jerk parameter of the method could be varied to ensure that the control remains within the rattlespace.

Suppose that the nominal future rattlespace limits are assumed to be constant, as in figure 4.21 (a), and as in figure 4.47. Here the initial distance, velocity and acceleration are given. Without loss of generality, suppose the centre of the rattlespace is zero height. The minimum control jerk that is required to just glance against the edge of the rattlespace can be calculated, producing the chassis trajectory shown in figure 4.47.

![Figure 4.47 Maximum Displacement at Rattlespace Limit](image)

Suppose first that the maximum and minimum distance travelled for any given magnitude of control jerk, $j$, can be calculated. Let the maximum distance travelled vertically be
represented as max, and the minimum as min. These values are easily calculated for any
given jerk, \( j \), once the time to reach the landing surface has been calculated (see section 8.8).
Since the separate parts of the trajectory are cubic polynomials, the maximum and minimum
points are easily found using calculus.

The pseudo code in figure 4.48 provides a numerical method to find a jerk value that will
maintain the system within the rattlespace. Let \( R \) represent half the rattlespace width. Firstly
the maximum and minimum heights are calculated for the initial jerk value, JInit, using the
current displacement, velocity and acceleration of the chassis. If the initial estimate leads to
either max or min outside the rattlespace,

\[
\text{max} > R \quad \text{or} \quad \text{min} < -R,
\]
then the initial jerk value is too small. The suspension is not “hard” enough, and so the
algorithm repeats, doubling the jerk value. If it is inside the rattlespace, the jerk value is not
yet “soft” enough and so the step is repeated with half the jerk control value. Eventually two
jerk values are derived: a “hard” value, JIn, which maintains the system inside the rattlespace,
and a “soft” value, JOut, which produces a trajectory outside the rattlespace. This algorithm
can be forced to stop if it reaches a predetermined large, maximum value for jerk, J_MAX,
which circumvents the need to mathematically prove that the algorithm stops, although it
allows the method to stray outside the rattlespace under extreme conditions.

After this, the interval between JIn and JOut is repeatedly dissected, modifying these values
so that they become closer and closer, stopping when the difference between them reaches a
predetermined very small tolerance value, JERK_TOLERANCE.
********* Find initial values for JIn and JOut, using doubling or halving

Find min and max with jerk set to JInit
If outside rattlespace (i.e. max>R or min<-R) Then
  JOut=JInit
  JIn=2*JOut
  Find min and max with jerk set to JIn
  While outside rattlespace (max>R or min<-R) and JIn< J_MAX
    JOut=JIn
    JIn=2*JIn
  End While
Else
  JIn=JInit
  JOut=JIn/2
  Find min and max with jerk set to JOut
  While inside rattlespace (i.e. max≤R or min≥-R) and JIn>JERK_TOLERANCE Then
    JIn=JOut
    JOut=JOut/2
  End While
End If
********* Keep dissecting till tolerance level reached
While JIn-JOut>JERK_TOLERANCE and JOut< J_MAX
  Jerk=(JIn+JOut)/2;
  Find min and max with jerk set to Jerk
  If outside rattlespace (i.e. max>R or min<-R)
    JOut=Jerk
  Else
    JIn=Jerk
  End If
End While
Return JIn or J_MAX, whichever is smallest

Figure 4.48 Pseudo code for Variable Hardness

The value found can effectively keep the suspension within the rattlespace, if the rattlespace does not move discontinuously, and if a jerk less than J_MAX is sufficient to achieve this goal. The use of this value not only ensures that the method stops, it is also included for numerical stability in evolutionary algorithms (see section 4.9). There is also a limit to the hardness of a suspension that a passenger should endure and which should cause a reasonable driver to slow down.
It has been found that control is improved if there is some decay in control strength. If a hard suspension is required momentarily, it is best not to decrease the control strength too quickly. A moving average (see section 8.12) can be adapted to this purpose.

4.8. Force Discontinuity in Semi-Active Suspension

Because of the passivity constraint (Yi and Song, 1999, p147; Giorgetti et al., 2006; Sergio M. Savaresi et al., 2003, p2264; Jalili, 2002, p600; Yokoyama et al., 2001; Hyvärinen, 2004, pp31-2) semi-active controls are constrained in the force they can supply, as explained in section 2.6. Semi-active controls are often adapted from other successful controls, frequently using “clipping” as explained below. For example, there are a number of “semi-active skyhook” controls in the literature which follow the skyhook algorithm (see section 2.4) within the constraints of the semi-active system.

4.8.1. Clipped Semi-Active Control

The simplest way to adapt a control algorithm for a semi-active system is to match the target system’s control forces where possible, and to use the closest value where this is not possible (Gordon and Best, 1994, p332; Johnson and Erkus, 2002, p2463; Jalili, 2002, p603; Giorgetti et al., 2006, p524; Dyke et al., 1996). The target control could be any desired control: an LQR control, a purely linear skyhook control, minimum-time control, sliding mode control, etc. The unconstrained target control before clipping will be referred to generically here as the “target control”.

It will be assumed that the damping rate limits are $c_{\text{min}}$ and $c_{\text{max}}$, where $0 \leq c_{\text{min}} < c_{\text{max}}$. (More accurate damper models are available, as discussed in section 2.6.) It is convenient to apply a “saturation operator” (Savaresi et al., 2003, p2266; Giorgetti et al., 2006, p524; Tseng and Hendrick, 1994, p549), which clips the input to the extremes. This is defined as,
Saturation is often referred to as “clipping” by engineers, and the term has also been used for semi-active controls (Krüger, 2002, p498; Jalili, 2002, p603; Giorgetti et al., 2006; Johnson and Erkus, 2002, p2463; Spencer et al., 1997, p14). Clipping the target suspension control results in the following damping rate,

\[
|c_{\text{min}}, c_{\text{max}}| = \begin{cases} 
  a & \text{if } x < a, \\
  b & \text{if } x > b, \\
  x & \text{otherwise.}
\end{cases}
\]

Equation 4.27

In this equation \(F_T\) is the force required by the target control. As in previous sections, \(k\) is the spring rate, \(s\) is the stroke and \(c\) is the damping rate. Note that as \(\dot{s}\) approaches zero, the damping rate saturates at its maximum value.

Refer to figure 4.49 below. Here a semi-active system starts out by being able to supply the target force, but as the stroke rate drops to zero the target force can no longer be maintained because of the passivity constraint. The point at which the damper loses the ability to reach the target force occurs when the stroke rate crosses from one direction to another (assuming that the target force does not change sign at exactly the same time). As the stroke rate approaches zero the damping force is clipped by the maximum damping rate, \(c_{\text{max}}\). At the point at which clipping starts, the damping force drops as stroke velocity approaches zero with force given by,

\[F_d = c_{\text{max}} \dot{s}.\]

The rate-of-change of damping force is,

\[\frac{d}{dt} F_d = c_{\text{max}} \ddot{s}.\]

The damping force drops to zero, and accompanying the sudden change in force is a spike in jerk. (The force after the stroke changes direction may be non-zero because the damper has a minimum damping rate \(c_{\text{min}}\).)
The greater the upper limit on damping rate, the higher the spike in jerk. Controllable dampers can become extremely stiff and they are capable of producing an immense spike in jerk. Thus a high jerk is caused by the clipping of a target control algorithm, and it occurs when the stroke velocity, \( \dot{s} \), changes sign. Furthermore, it produces a spike in jerk without necessarily encountering a road surface discontinuity (as discussed in section 4.3).

Gordon and Best claimed that clipping provides an acceptable approximation to their target control, but fitness in their experiments was measured using acceleration and force (1994, p334) rather than jerk. Sudden changes between moderate acceleration values can have almost no effect on RMS acceleration, as explained in section 3.1. Clipped controls can seem to perform well using RMS acceleration because this measure ignores entirely this artificially created, highly uncomfortable spike in jerk.

The clipped linear quadratic regulator has been referred to as the “clipped-optimal” control, and it has received a great deal of attention in the literature (Krüger, 2002, p498; Giorgetti et al., 2006; Johnson and Erkus, 2002; Tseng and Hendrick, 1994, p546). Giorgetti, Bemporad et al. refer to this also by the more specific term, “clipped-LQR” (2006, p523). Similarly, a clipped version of the pure skyhook control will be termed here the “clipped skyhook” control.
This author briefly investigated the possibility of a system that might allow two controlled dampers “back-to-back” to produce a control force in either direction. This is almost certainly impossible, given that the damper always seems to produce a resultant force that opposes the motion of the chassis relative to the wheel, and vice versa. Even if there were some system of pulleys or levers that made this possible, it would no doubt require a fundamental and somewhat inconvenient change to suspension geometry. Furthermore, control force would still drop to zero when the stroke rate reaches zero, and the “crossover” problem substantially remains. This possibility will not be considered further in this thesis.

Another variant on a clipped control law, put forward by Savaresi et al., is called the “linear skyhook”. This mixes groundhook and skyhook in a generalized way. The control law is,

\[ c = \begin{cases} \text{sat} \left( \frac{\alpha \dot{s} + (1 - \alpha) \dot{c}_{\text{max}} \dot{\gamma}}{\dot{s}} \right) & \text{if } \dot{\gamma} \dot{s} \geq 0, \\ 0 & \text{if } \dot{\gamma} \dot{s} \geq 0. \end{cases} \]

The parameter, \( \alpha \), determines the amount of mixing and the “typical value for \( \alpha \) is 0.5” according to Savaresi et al. (2003, pp2265-6).

### 4.8.2. No-Jerk Skyhook

This short section examines a control law from the literature, the “no-jerk skyhook” (Ahmadian et al., 2004; Reichert, 1997), which has been devised as a global control law to remove crossover jerk. Crossover jerk occurs when the stroke velocity, \( \dot{s} \), changes sign. To counter this effect, the damping rate can be reduced in a controlled way as the stroke velocity changes sign. Similar methods have been developed and published as part of this PhD research (Storey et al., 2006).

Recall from section 2.6 that the no-jerk skyhook control law can be written as,

\[ i = \begin{cases} \gamma \dot{s} & \text{where } \dot{\gamma} \dot{s} > 0, \\ 0 & \text{where } \dot{\gamma} \dot{s} \leq 0, \end{cases} \]

(Reichert, 1997, p63). In this equation, \( i \) represents the control force and \( \gamma \) is a parameter.

This control was tested in a physical experiment with a heavy-truck seat suspension (Ahmadian et al., 2004, p581; Reichert, 1997). Ahmadian claimed that the physical analysis
“clearly showed the elimination of jerk”. It has energy damping properties similar to the on-off skyhook (discussed in section 4.2) but it doesn’t suffer the jerk spike of the on-off skyhook. This control law is a global law applied for only a local effect, but it shows the effectiveness of the control as stroke rate crosses zero.

### 4.8.3. Semi-Active Jerk Reduction

A variant of the no-jerk skyhook was developed during the PhD research and was termed the “lo-jerk skyhook” (Storey et al., 2006). The crucial element of this control was the approach to zero stroke rate: damping control reduced to a minimum with zero stroke rate. A “lo-jerk” control was defined generically as any control in which the damper approaches minimum stiffness as the stroke rate approaches zero. In fact a fuzzy control version of the lo-jerk skyhook was the highest performing semi-active control, according to numerical experiments represented in this paper.

A semi-active control was developed which switches from any general high-performing control, say skyhook, but reduces the damper stiffness to minimum when stroke rate seems to be decreasing to zero. The method to reduce control force “cuts in” when stroke velocity seems to be approaching zero, but it allows another control to be used at other times (Storey et al., 2008). Thus a high performance control can be used when there is no danger of jerk caused by clipping, but the reduction to zero damper force is smooth when clipping seems imminent.

While later working on the theory below it was discovered that a paper had been published by Stamatov et al. in April 2008 (2008), as this author’s paper was in submission. Stamatov et al. also have a two-mode control, but the algorithms used are different and there is no analysis of jerk. The differences are discussed in the conclusions, in section 7.4.

For the purposes of this discussion the high-performing suspension control will be referred to as the “target control”. This could be any control that otherwise would need to be clipped for a semi-active suspension: skyhook, linear with bump stops, LQR, etc. When the stroke rate,
\( \dot{s} \), drops toward zero, the control switches to a mode that is designed to smoothly reduce the damper force to zero. Thus the “crossover reduction” method will activate when it is determined that the target control is about to be clipped, and will determine the damper force in order to provide a smooth transition to minimum control. A heuristic method for doing this was published during the course of the PhD research (Storey et al., 2008). The method presented below is developed in the context of the physical model.

In figure 4.49 above, crossover jerk can be seen to occur when the stroke velocity approaches zero. A change in direction of the damper’s stroke velocity moves the system “across” the boundary of the passivity constraint (refer to figure 2.10). The removal of jerk becomes urgent at the point that the stroke velocity, \( \dot{s} \), changes sign. For the purposes of this discussion, the point at which the stroke rate crosses zero, \( \dot{s} = 0 \), can be referred to as “crossover”.

Without loss of generality, as in figure 4.49 above, the target control force is assumed positive before the stroke rate changes sign. The diagram shows damper force only (the spring force component has been subtracted). Up to the point that stroke velocity reaches zero, the maximum force that can be applied by the damper is,

\[
F_{d, \text{max}} = -c_{\text{max}} \dot{s}.
\]

To remove the spike in jerk before clipping, it is necessary to reduce the damping force smoothly to zero when a change in the direction of stroke rate is imminent.

![Figure 4.50 Crossover Jerk Reduction](image)

Figure 4.50 Crossover Jerk Reduction
Consider figure 4.50. From time $t_0$ to $T_X$ the damping force is dropped gradually to zero with constant jerk. Without loss of generality, suppose that the time $t_0$ is zero: $t_0 = 0$. The stroke velocity and acceleration are functions of time, $\dot{s}(t)$ and $\ddot{s}(t)$.

Suppose also that stroke velocity reaches zero at time $T_C$,

$$\dot{s}(T_c) = 0.$$  

**Equation 4.28**

Let us designate the stroke velocity and acceleration at time zero as follows:

$$\dot{s}_0 = \dot{s}(0) \quad \text{and} \quad \ddot{s}_0 = \ddot{s}(0).$$

If these are roughly constant after time $t_0$ up to the point of crossover then,

$$\dot{s}(t) \approx \dot{s}_0 + \ddot{s}_0 t.$$  

After combining with equation 4.28 this gives,

$$T_C \approx \frac{-\dot{s}_0}{\ddot{s}_0}.$$  

**Equation 4.29**

This is one, simple method for estimating $T_C$. Other methods based on more complex statistical techniques may provide better estimates, but these are not examined here.

In order to be conservative with the time estimate, a parameter $\alpha$, is introduced to reduce damping force more quickly than otherwise might be the case. The following conservative time estimate then could be used,

$$T_X = \frac{T_C}{\alpha} \approx -\frac{\dot{s}_0}{\alpha \ddot{s}_0},$$  

**Equation 4.30**

where say $1 \leq \alpha \leq 2$. There are heuristic reasons for using the parameter $\alpha$. Using a conservative estimate for crossover time, $T_X < T_c$, may help remove anomalies produced by state estimation errors (especially in the acceleration estimation), and it may help with
stochastic fluctuations in road trajectory predictions. In numerical experiments, the best value for the parameter can be simply determined by evolution. Again, other methods are possible. It is possible to ignore this step, simply by putting \( T_x = T_c \).

Now a constant jerk is needed to smoothly reduce the damping force down to zero during the period \( t_0 \leq t \leq T_x \). Suppose that a value of jerk, \( J_x \), is nominated for this task. This is judged to be a “comfortable” jerk level, yet sufficient for the task. Suppose \( J_x \) is the value of jerk used in figure 4.50 (in that case \( J_x \) is negative). During the period \( t_0 \leq t \leq T_x \) the damping force will approximate the linear function,

\[
F_d(t) = F_0 + mJ_x t,
\]

Equation 4.31

where \( m \) is the chassis mass and \( F_d(t) \) represents the damping force. Suppose the target is to achieve a zero force, \( F_d(t) = 0 \), when \( t = T_x \). Then equation 4.31 gives,

\[
F_0 = m\ddot{s}_0 = -mJ_x T_x.
\]

Equation 4.32

Solving for \( J_x \), this produces,

\[
J_x = \frac{-\ddot{s}_0}{T_x}.
\]

Equation 4.33

Substituting \( F_0 \) from equation 4.32 back into equation 4.31 produces,

\[
F_d(t) = F_0 + mJ_x t = -mJ_x T_x + mJ_x t = mJ_x (t - T_x).
\]

Here, \( T_x \) is determined from equation 4.30 and \( J_x \) is determined from equation 4.33. If the step size of a discrete control is \( h \), the desired damping force at the end of the time step is,

\[
F_d(h) = mJ_x (h - T_x).
\]

Equation 4.34

These equations then provide a method for determining control force required for crossover: equation 4.30, equation 4.33, and equation 4.34. Equation 4.30 and equation 4.33 are the
simplest possible methods for estimating crossover time and required jerk, and are capable of refinement.

It remains to determine if crossover is needed or if the target control should be applied. A very simple method is to use just the crossover option whenever the crossover force magnitude is lower than the target control force magnitude (and of the same sign).

Figure 4.51 shows a simulation using this method. (Select the button “Crossover Demo” in the program OvershootDemo.java.) The “target law” here is simply to use jerk to maintain acceleration at a constant. This is used to allow the jerk reduction strategy to be clearly differentiated from the target.

Figure 4.51 Crossover Jerk Removal
In figure (a), no crossover strategy is applied, and there is a sudden drop in damper acceleration at crossover, associated with an uncomfortable, enormous spike in jerk. Figure (b) shows the slow reduction to zero acceleration using crossover jerk removal. Because the predicted movement cannot be exact, in a stochastic environment, jerk changes direction before crossover, however, small jerk chatter is barely noticeable compared to acceleration chatter, or indeed to sudden large changes in acceleration. Again, the jerk chatter should be small enough to produce barely noticeable changes in force (refer to section 2.3.2.4 for a discussion on jerk chatter). On the downward slope crossover removal is being applied, as given in equation 4.33.

In figure (c) a simple strategy is used to reduce chatter. If crossover reduction has been applied in the previous step, but is not required in the current step, then a jerk value in between the crossover jerk, $J_x$, and the target value is required. In the simulation the jerk applied is given as,

$$\left( J_T + 3J_x \right)/4,$$

Where $J_T$ is the target damper jerk. This step is unnecessary, since small jerk chatter is not a great problem, but it does seem to help to slightly improve prediction of crossover time.

The above analysis investigates the case when the target force is moving out of the controllable region. The case that the target force is moving into the controllable region is much less problematic. In the case of control over acceleration, a limit jerk can be applied for the rate of increase of acceleration up to the target value, as in the simulation in figure 4.51 (b). In the case of a control over jerk, however, there is no problem at all because the jerk limits of the control itself take over.

Up to this point the jerk contributed by the suspension’s spring has been ignored. However, assuming a linear spring, it can be shown that the spring does not contribute jerk when the stroke rate changes sign. Jerk is the derivative of force, and the condition that the damper is reaching crossover, $\dot{s} = 0$, is precisely the condition that the jerk component contributed by the spring is zero;

$$\text{Spring Jerk} = \frac{d}{dt} \left( -k\dot{s} \right) = -\frac{k}{m}\ddot{s} = 0.$$
Given that springs are highly linear, the simplification of ignoring the spring force is justified.

A small technical problem for simulations is that the system acceleration needs to be initialized to the spring acceleration rather than zero. This is because the damper will attempt to “correct” for a false zero, producing a spike in jerk on the first control step.

The method described here is not computationally complex, using the simple equations, equation 4.30, equation 4.33, and equation 4.34. The first two of these could be refined, as noted above. Other possible refinements include a minimum damping rate, and a minimum jerk level. It is also possible to make the method “stiffer” near the rattlespace limits, allowing the system to be more responsive when there is a danger of rattlespace collision.

### 4.8.4. Semi-Active Suspension and Rattlespace Constraint

The passivity constraint restricts the forces that can be applied by a semi-active system. Damper forces are limited in direction, as explained in section 2.6. However, the controlled damper in a semi-active suspension can always resist stoke movement away from equilibrium. Therefore, at precisely the moments that a control (such as a rattlespace constraint control) senses movement towards a rattlespace limit, either nearly topping or bottoming, the damper stiffness can increase, resisting the movement towards the edge of the rattlespace. At precisely those moments, the passivity constraint does not apply. This seems to be one of those pleasant cases in which physics favours the engineer. It seems likely therefore that semi-active controls are well suited to the implementation of rattlespace constraint controls.

### 4.9. Numerical Parameters and Numerical Constraints

The parameters of the numerical experiments are based on data contained in table 2, which has been derived from models of actual vehicles. Also wheel stroke was set to correspond to the stroke of a passenger vehicle. Deprez et al. (2002) provide data on RMS accelerations
attained under various conditions (see table 5) for tractors. Their graph of road accelerations under rough conditions shows the acceleration peaking at around 1 g, corresponding to the driver being weightless under some conditions. Such conditions, of course, are rarely attained with passenger vehicles. Nonetheless, values of close to 1 g are sometimes experienced over rough conditions, with above 1 g being highly unusual.

<table>
<thead>
<tr>
<th>Input profile</th>
<th>Suspension</th>
<th>RMS Acceleration m/s²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field (4 km/h)</td>
<td>No Suspension</td>
<td>1.009</td>
</tr>
<tr>
<td></td>
<td>Passive</td>
<td>0.058</td>
</tr>
<tr>
<td></td>
<td>Semi-Active</td>
<td>0.025</td>
</tr>
<tr>
<td>Unpaved road (11 km/h)</td>
<td>No Suspension</td>
<td>0.686</td>
</tr>
<tr>
<td></td>
<td>Passive</td>
<td>0.094</td>
</tr>
<tr>
<td></td>
<td>Semi-Active</td>
<td>0.054</td>
</tr>
<tr>
<td>Paved Road (28 km/h)</td>
<td>No Suspension</td>
<td>1.502</td>
</tr>
<tr>
<td></td>
<td>Passive</td>
<td>0.352</td>
</tr>
<tr>
<td></td>
<td>Semi-Active</td>
<td>0.063</td>
</tr>
</tbody>
</table>

Table 5 Calculated RMS Acceleration Under Various Conditions

Some of the control functions developed above increase force asymptotically as the stroke approaches the rattlespace limit. Numerical instability can be controlled by placing limits on experimental controls that output unrealistic control forces.

Let $d_{\text{max}}$ represent the maximum distance that a control can move in a period of one second.

If the maximum jerk is $j_{\text{max}}$ then,

$$d_{\text{max}} = \frac{1}{6} j_{\text{max}} t^3 = \frac{j_{\text{max}}}{6}.$$  

Equation 4.36

A different analysis yields a slightly different result. Suppose that the maximum time period of a time step is $h$, and suppose the distance travelled in one time step is limited to $d_{\text{max}} / h$.

In this case,

$$\frac{1}{6} j_{\text{max}} h^3 = \frac{d_{\text{max}}}{h},$$

$$j_{\text{max}} = \frac{6d_{\text{max}}}{h^3}.$$  

Equation 4.37

This is a somewhat larger value (given that $h$ is small). For acceleration a similar analysis yields,
It is important to distinguish between constraints that are part of the control and constraints that are used for numerical stability. A constraint that is capable of being set by an evolutionary process for example, may become quite low during evolution, becoming a true part of the control. This can affect the control interpretation, and it can slow down evolution.

\[ a_{\text{max}} = \frac{2d_{\text{max}}}{h^3}. \]

4.10. Performance Measures Applied in Experiments

Suspension controls developed in this thesis was first tested using computer models applied in evolutionary algorithms. A test bed program, described in the next chapter, was developed to simulate a number of different kinds of suspensions and to compute the comfort experienced by a passenger using the fourth power of jerk as in equation 3.3,

\[ J_c(s, y) = \frac{1}{T} \int_0^T \dddot{y}(t)^4 dt. \]

and to separately compute a factor designed to indicate how well the suspension tracks the road surface using the non-quadratic weighting, \( \varphi \), of equation 3.7,

\[ J_p(s, y) = \frac{1}{T} \int_0^T \varphi(s(t)) dt. \]

The fourth power of jerk was used for the comfort measure in order to penalize large jerk values more severely than the least squares measure. This is because the least squares measure may mask some conditions where there is a high jerk for very short periods of time. In order to overcome this possibility the next highest even power was used. It was felt that very high-power norms, or even the \( L^\infty \)-norm (maximum magnitude of jerk) would too severely penalize momentary discomfort. This is very much a compromise and itself needs further investigation. The reasons for the tracking measure are explained in detail in section 3.2.

A large number of suspension control algorithms were tested. Some systems were related to each other. For instance, there were a number of systems using the basic skyhook control
with minor refinements. Each of these suspension controls has a number of parameters. For example, the passive suspension has just two parameters, one for the spring rate and one for the damping rate. Other suspensions may have as many as 20 parameters. The parameters of each of the controls were submitted to an evolutionary algorithm.

When referring to a particular suspension control type the term “control” or perhaps even “control algorithm” will be favoured over the term “algorithm”. This is to avoid ambiguity with evolutionary algorithms.

Each type of suspension system runs through an evolutionary algorithm with different parameter values in competition over many generations. The schedules for the evolutionary algorithms are explained in the next chapter. Randomly chosen roads are used as the input. The mass of the vehicle was constant throughout; otherwise the evolutionary algorithm would tend towards lower and lower vehicle weight. Suboptimal control parameters are derived by selecting the highest scoring suspension from the final generation.

The major benchmark controls, the passive and the skyhook controls, use only a small number of genes (two real number genes each). With such a small search space, and with such simple controls, evolution is very quick. As discussed in section 4.5.1, the passive and skyhook controls make good benchmarks against which to judge other controls. These controls also admit linear modelling and their frequency response characteristics are well understood; see section 2.4. Any controls that perform convincingly better than these benchmarks deserves further investigation, in possibly more detailed simulation or in physical experimentation.

Details of the test bed program running the evolutionary algorithms and the suspension simulations are discussed in the following chapter.
5. Computer Simulation Environment and Evolutionary Algorithm Coding

This chapter describes the numerical evolutionary experiments performed for this thesis, and the test bed software. The numerical test bed program is the main focus of this chapter, as well as a small number of ancillary programs. This program was written entirely by this author in Java.

The graphs used throughout this thesis were produced by software written by the author for the purpose of the thesis. Some standard drawing routines were applied, but the placing of ticks and axes, and routines for drawing the graphs are all the work of the author. The code for these routines can be found in Java/GraphX, in the class definition file FunctionGraph.java, and a demonstration can be run from the main() routine in Demo.java.

5.1. Objectives

As described above, this research is not aimed at a comprehensive model of any one particular suspension control, but at an overview of a various candidate algorithms for electronically controlled suspensions. Thus the purpose of the computer test bed program was to compare a large number of separate basic suspension control algorithms by means of numerical suspension simulations and relevant evolutionary algorithms.

The suspension problem investigated here is only slightly more complex than the simple LQR system and yet it is intractable analytically and should be approached numerically. The sheer versatility of electronic control makes the use of more realistic performance measures crucial. In this research performance measurements are comfort and rattlespace, which are recognised to be intractable analytically. Also, as has been shown in section 3, some controls that perform well using classical analytical techniques prove extremely jerky in practice.
Thus EAs are used for testing simple control algorithms where analytical techniques prove intractable. EAs are virtually limitless in the number of performance measures they can use.

The method then was to submit a range of suspension control algorithms to EAs and find a “suboptimal” control for each algorithm. The EAs were run separately for each suspension type. The median score of the final generation of the EA for a given suspension type becomes the “score” for that suspension control. The median score was used because excessively bad road conditions produce outliers that greatly skew the mean. On the other hand, in the final generation, large numbers of roads are used to produce a stringent test of suspension capabilities.

For the purposes of comparison, each suspension algorithm should be run under the same conditions as the others. All systems were tested with the same chassis weight and exactly the same road conditions. Furthermore, all semi-active systems were tested with the control output being the only difference between them. Similarly, for all active algorithms in the same numerical model with control output being the only difference.

The problem with using EAs is that the control that is judged superior will depend on the particulars of the numerical experiment, chassis mass and road perturbations. To this end, a number of candidate algorithms are chosen here as candidates for further investigation of suspension control theory. A number of different controls were also selected for physical experimentation. Despite this, EAs are quite robust (see section 2.3.2) and so the numerical experiments performed here can provide a fair performance comparison.

5.2. Platform

The process of running the various suspensions can be very time consuming, taking days and even weeks to complete. Speed of processing was a major consideration in this project and the writing of dedicated code seemed to offer greater speed than if the experiment had been set up in Matlab or Simulink. An advantage of using Java as the programming language is
that the control algorithms run in Java use code that is similar to the C languages programming code used in the microprocessor.

All code used for this thesis was written by the author during the course of research for the PhD, including the graphing software. Standard Java libraries for user interfacing, file access, etc., were used as needed. Development was performed using the Java SE Development Kit (JDK), Version 6, downloaded from the Sun website. All code was written using the text editor, TextPad.

Java code for various purposes such as graphing and running EAs is freely available on the Internet, and this is one of the reasons that Java was selected as the programming language. However, the code snippets found on the Internet for processing genetic algorithms were slow and awkward to use. In the end, third-party software was not used for any purpose, only the standard Java libraries including the Java Swing libraries.

There is insufficient space here for an in-depth explanation of the design process or a line-by-line explanation of the programming code. Some programming code details are supplied where these are considered to be useful in replicating or verifying algorithms, or in explaining the implementation of a programming technique.

The code for the test bed program is contained within the subdirectory Java/SuspensionTestX, where “X” stands for the version number. For example, the version 3.02 is contained in the subdirectory Java/SuspensionTest302. (Not all code edits resulted in new version numbers.) The program source files are all the files with the extension .java contained in this directory. The test bed program can be compiled and run using the following statements (in a “DOS” console):

```plaintext
D: \ Apps \ Java \ jdk1_6 \ bin \ javac  *.java
D: \ Apps \ Java \ jdk1_6 \ bin \ java  SuspensionTest
```

(The directory structure is dependent on the machine implementation of Java, and will almost certainly be different from that shown here.)

Every effort has been taken to minimize the use of Java code in the thesis, but in some cases more detail is needed. It is important to verify the way in which the SuspensionTest program converts between the phenotype (the code that represents the suspension control algorithm in
the on-board microprocessor) and the genotype (a collection of floating-point values representing the suspension parameters). The goal of this procedure is to keep the algorithmic code in the numerical experiments as close as possible to the target microprocessor code used in electronic control. Because the process is non-standard and somewhat complex there is a need to go into some depth in the explanation of it. Nonetheless, the intent of the conversion should be obvious to someone who skims section 8.21.3, and this section can be skimmed without affecting the rest of the thesis content.

Throughout the code a pseudorandom number generator has been used that is supplied in the standard Java libraries, Math.random(). This is claimed by Sun to produce pseudorandom floating-point numbers that mimic a uniform random variable between zero and one. The term “random” will be used freely below, although the more awkward term “pseudorandom” is perhaps more accurate.

5.3. Overview of Test Bed Program Functionality

A screen shot of the main user interface of the suspension test bed program, SuspensionTest, is shown in figure 5.1. The main goal of the computer program was to provide a flexible test bed for the design of various active and semi-active suspension control algorithms and to use evolutionary algorithms as a testing tool.

Figure 5.1 User Interface for the Test bed Program
During the processing of an evolutionary algorithm, the parameters of a suspension control are altered. For example, in the case of the purely linear passive suspension there are just two control parameters: the spring rate and the damping rate. The evolutionary process will experiment with a large number of such suspensions each with different spring and damping rates. The aim of the evolutionary process is to evolve robust suspensions of a number of different suspension control types with as high a performance as possible for each type of suspension.

At the completion of the evolutionary process, after sufficient “cooling”, the median performance measure of the final generation is nominated as the performance measure for that control. This is in fact a suboptimal value. Many control algorithms are “optimized” in this way and compared for performance. The performance of one system over another is dependent on road surfaces. The random road surfaces of the numerical experiments are discussed in detail below.

Two fundamental types of suspension are catered for: active and semi-active. Crucial to the numerical experiment is the fact that algorithms are run in exactly the same manner in the numerical model, with the application of control being the only difference. All active controls are processed in the same numerical model, with only acceleration as the output. Similarly, each semi-active algorithm is processed in the same numerical model with damping rate as the output. The numerical methods, their test bed programs and their validation are discussed in the appendix, section 8.15.

Some of the suspension control algorithms are quite complex, and in some cases separate test bed programs for these algorithms or a set of related algorithms have been developed. Appendix section 8.13 discusses these programs in a little more detail.

The main test bed program design is based on object-oriented design principles. The design also used data-driven principles and persistent data. The design is explained in more detail in the appendix, section 8.18.
5.4. **Simulation Models**

The simulations run in the EAs require physical modelling of road surfaces and suspension responses. This section explains the physical modelling in more detail.

5.4.1. **Road Surfaces**

The test road surfaces were designed to have repeated bumps of a mixture of frequencies and to contain single bumps, to test the capacity of algorithms to handle substantial and sustained height changes. The details of the generation of the road surfaces is contained in the appendix in section 8.19.

An example of a road surface generated by this algorithm is shown in figure 5.2.

![Figure 5.2 Example Road Surface](image)

5.5. **Fitness Measures**

The code for calculating the fitness measures is contained in the class Fitness. The fitness measures are calculated according to Equation 3.3 and Equation 3.6 using the penalty of Equation 3.7. The parameters of these functions can be set at runtime by the user. Refer to the appendix for details, in section 8.20.

5.5.1. **Comfort**

The comfort factor is calculated using the Simpson’s method approximation (Kreyszig, 1993, p961) to the integral,
\[ J_C = \int_0^T \dddot{y}(t)^4 \, dt. \]

(See section 3.1.) The value of jerk was approximated from sampled acceleration values,
\[ \dddot{y}(t_n) \approx (\dddot{y}(t_{n+1}) - \dddot{y}(t_n))/h. \]

The overall score out of 1000 was then calculated using
\[ C = 1000 - cJ_C, \]
where \( c \) is the scaling factor for comfort used for the weighted performance index. This factor is contained in the constant \texttt{JERK_FACT}, defined in the \texttt{FitnessData} class in the file \texttt{Fitness.java}.

### 5.5.2. Rattlespace Tracking

The rattlespace tracking factor is calculated from the Simpson’s method approximation to the integral,
\[ J_R = \int_0^T \phi(s(t)) \, dt. \]

The weighting function \( \phi \) is calculated using the \texttt{extensionBadness()} function of the \texttt{ExtensionFunction} class. (All functions of this class are static.) The equation used for \( \phi \) here is given above in equation 3.7, and a graph of the weighting function \( \phi \) is shown above in figure 3.3. The values used for the calculation of the rattlespace penalty function are contained in the class \texttt{RattlespaceParameters}, in the file \texttt{Parameters.java}. The correspondence between the variables in this file and the variables in equation 3.7 is:

- \texttt{mediumExtensionBadness} \( \equiv P_1 \),
- \texttt{maxExtensionBadness} \( \equiv P_2 \),
- \texttt{maxGoodExtension} \( \equiv m_1 \),
- \texttt{absoluteMaxExtension} \( \equiv m_2 \),
- \texttt{tailRate} \( \equiv T \).

The overall score out of 1000 was then calculated using
\[ R = 1000 - rJ_R, \]
where $r$ is the scaling factor for rattlespace tracking. The value used for $r$ is contained in the constant `RATTLE_FACT`, defined in the `FitnessData` class in the file `Fitness.java`.

5.6. Genes and Evolutionary Processes

“Genes” represent suspension control parameters and these are tested in simulation over a large number of roads. Different gene types are responsible for representing the different suspension control algorithms; each gene class represents a different control algorithm. The gene class thus contains the crucial logic of the control algorithm. Furthermore, the code in the gene class is similar to the control logic in the microprocessor that would control the suspension.

The representation of genes and generations, and the processes of mutation, crossover and selection have all been programmed by the author, and the details are explained in the appendix (see sections 8.21.1 to 8.21.5). Note that some of the evolutionary process involved genes that were spread along a Pareto front, as shown graphically in figure 8.35.

5.7. Results of EA Experiments

It should be kept in mind that a poor result for some of the numerical controls could be due to programming bugs. Every effort has been taken to ensure that the routines work properly, and separate testing routines have been used, especially in the case of more complex control algorithms. But, as a programming lecturer of more than decade’s experience and as one who has worked in the field, the author knows better than to guarantee that all the routines coded here are entirely bug free.

On the other hand the general modelling environment has been very thoroughly tested (refer to section 5.3 and appendix section 8.15), and the author is highly confident that any bugs in at least the numerical model are limited to the “plug-in” routines defined by the gene classes (in Gene.java) and not to the physical models.
There are other important qualifying factors to be kept in mind when interpreting the results below, such as the fact that EAs produce suboptimal and not optimal results. These qualifications will be addressed in the concluding chapter, especially in section 7.2.

In order to save on doubling-up of naming, the various controls will be given the name of the Java programming class in which the control logic is coded. For example, the standard passive control is encoded in the Java class “Passive” and so will be referred to in the results as the “Passive” control. The various classes have evolved over a long period of time (longer than 5 years). Some of the class names may appear a little inscrutable, such as “SkyhookOnOffFilteredRoadGene” for instance. The names distinguish the control, but the name alone might not exactly define or indicate the control function.

The two benchmark controls are the Passive and the ActivePureSkyhookGene (the linear skyhook examined in section 2.4). Both of these are purely linear and they are perhaps the simplest controls used (they use just two real-numbered gene components each). Certainly, they are the simplest controls that produce reasonable results.

Some of the control algorithms are explained here, but not all. In the PhD appended material are HTML pages that describe a major proportion of the control functions. (These can be referenced from the page,

PhD\Java\SuspensionTestX\Help\Help.html)

These pages contain a brief overview of a large number of genes (click on the “Gene Classes” link). The complete control logic of every gene in the form of Java code is contained in a class defined in a single Java source file (in the file “Gene.java” in the folder,

PhD\Java\SuspensionTestX.)

The final overview statistics for the performance of the genes in the final run of the evolutionary algorithms can be obtained (by clicking on the link, “Comparative stats for all genes”). (This page is generated by pressing the “All Stats Summary” button in the “Statistics” frame. A history of previous generations’ results can be found in the folder,

PhD\Java\SuspensionTestX\stats\)
While not all the controls are explained in detail the author is of the opinion that showing the results for all genes developed gives a fair overview of the relative success of the controls. The controls that performed on a par with the passive control or better are the primary focus of the discussion below.

In figure 5.3 below, samples are shown of the final, and hence highest performing generation of three different suspension types: the passive (Passive), the linear skyhook (ActivePureSkyhookGene) and a general linear control over acceleration with a virtual bump stop (FlatLinearAcceleration01Skew). In these examples, the green line represents a road (randomly generated as described in section 5.4.1) and the red line represents the chassis. The superiority of the tracking performance of the skyhook over the passive is evident even from a cursory glance at these graphs.

Figure 5.3 Examples of Suspension Performance
In contrast, figure 5.4 shows an example trajectory of a poorly performing control: SkyhookPassiveSpikeRemovalGene with a score of -3899. Note that the control seems smooth, but there are a number of rattlespace collisions (as seen when the graphs cross in the case of the bottom rattlespace limit). The evolutionary algorithm has been unable to find a high performing compromise between smoothness and tracking using this control.

![Figure 5.4 Example of Poor Performance](image)

For the experiments the tracking performance weighting function (as described in section 5.5.2) is shown in figure 5.5. The rattlespace width is effectively 0.6 m. That is to say, the chassis will hit the rattlespace edges when the suspension moves 0.3 m from the rattlespace centre (which is assumed to be the equilibrium point).

![Figure 5.5 Rattlespace Penalty Function](image)
The results presented below are derived from a near continuous run on a laptop which ran from 4:31 PM 20th November 2010 to 10:41 AM on the 30th. The run was not entirely continuous because the EA would be stopped from time-to-time to save the current state (the facility to do this took some time to program, but it resulted in saved time over the course of the research by allowing evolutionary algorithms to be stopped, saved and re-started instead of losing data when programs were halted half-way). Furthermore, there was one point, after perhaps a few hours where a bug in one gene had stopped the evolutionary algorithm. This gene was dropped, and the evolutionary algorithm had to be re-started.

All-in-all the evolutionary process took approximately nine and a half days. Some 123 Controls were tested with an average time of roughly 2 hours for each algorithm. The data shown below is derived from an automatically generated HTML file that summarizes the performance of the various control algorithms. (The HTML file can be found at, PhD\Java\SuspensionTestX\stats\allStats.htm)

The benchmark controls are shown highlighted.

<table>
<thead>
<tr>
<th>Gene Type</th>
<th>Median</th>
<th>Mean</th>
<th>Median Jerk/Smoothness</th>
<th>Median Rattle-Space/Tracking</th>
<th>Suspension Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlatLinearJerk01Skew</td>
<td>982.1</td>
<td>980.4</td>
<td>978.1</td>
<td>986.2</td>
<td>Active</td>
</tr>
<tr>
<td>FlatLinearAcceleration01Skew</td>
<td>978.0</td>
<td>975.6</td>
<td>976.9</td>
<td>979.1</td>
<td>Active</td>
</tr>
<tr>
<td>RCollisionAvoidJerk03</td>
<td>977.5</td>
<td>974.1</td>
<td>975.5</td>
<td>979.5</td>
<td>Active</td>
</tr>
<tr>
<td>FlatLinearAcceleration01</td>
<td>976.4</td>
<td>973.6</td>
<td>970.5</td>
<td>982.3</td>
<td>Active</td>
</tr>
<tr>
<td>SigmoidLinearAcceleration01</td>
<td>974.7</td>
<td>971.9</td>
<td>965.2</td>
<td>984.2</td>
<td>Active</td>
</tr>
<tr>
<td>ActiveAdaptiveJerkFilterGene</td>
<td>974.4</td>
<td>969.7</td>
<td>971.8</td>
<td>977.0</td>
<td>Active</td>
</tr>
<tr>
<td>ActivePassiveSkyhookGene</td>
<td>973.9</td>
<td>971.3</td>
<td>968.0</td>
<td>979.7</td>
<td>Active</td>
</tr>
<tr>
<td>FilterWithStiffening01</td>
<td>971.7</td>
<td>960.5</td>
<td>958.9</td>
<td>984.6</td>
<td>Active</td>
</tr>
<tr>
<td>ActiveAdaptiveJerkFuzzyFilterGene</td>
<td>968.2</td>
<td>-9.2E9</td>
<td>941.2</td>
<td>995.3</td>
<td>Active</td>
</tr>
<tr>
<td>BackMomentumActive01Gene</td>
<td>966.6</td>
<td>964.6</td>
<td>963.0</td>
<td>970.1</td>
<td>Active</td>
</tr>
<tr>
<td>FlatLinearJerk01</td>
<td>965.0</td>
<td>960.5</td>
<td>985.0</td>
<td>945.0</td>
<td>Active</td>
</tr>
<tr>
<td>ActiveAdaptiveJerkFilter01Gene</td>
<td>963.9</td>
<td>962.6</td>
<td>944.9</td>
<td>982.9</td>
<td>Active</td>
</tr>
<tr>
<td>ActivePureSkyhookGene</td>
<td>958.3</td>
<td>955.9</td>
<td>973.5</td>
<td>943.1</td>
<td>Active</td>
</tr>
<tr>
<td>AnticipatedTravelActive01Gene</td>
<td>951.3</td>
<td>5.6</td>
<td>982.2</td>
<td>920.4</td>
<td>Active</td>
</tr>
<tr>
<td>PassiveGeneDoubleSpring</td>
<td>950.4</td>
<td>948.3</td>
<td>939.8</td>
<td>961.0</td>
<td>DoubleSpring</td>
</tr>
<tr>
<td>EdgeJerk01</td>
<td>948.0</td>
<td>945.1</td>
<td>924.8</td>
<td>971.2</td>
<td>Active</td>
</tr>
<tr>
<td>FullFilter01</td>
<td>931.8</td>
<td>929.7</td>
<td>909.7</td>
<td>954.0</td>
<td>Active</td>
</tr>
<tr>
<td>RattlespaceCollisionActive03Gene</td>
<td>928.6</td>
<td>926.2</td>
<td>909.6</td>
<td>947.5</td>
<td>Active</td>
</tr>
<tr>
<td>EulerLagrange02</td>
<td>923.2</td>
<td>920.5</td>
<td>942.7</td>
<td>903.8</td>
<td>Active</td>
</tr>
<tr>
<td>Function</td>
<td>Value1</td>
<td>Value2</td>
<td>Value3</td>
<td>Value4</td>
<td>Status</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>EulerLagrange01</td>
<td>920.6</td>
<td>917.9</td>
<td>902.5</td>
<td>938.8</td>
<td>Active</td>
</tr>
<tr>
<td>RCollisionAvoidJerk02</td>
<td>918.8</td>
<td>-8.1E04</td>
<td>862.4</td>
<td>975.2</td>
<td>Active</td>
</tr>
<tr>
<td>ActiveAdaptiveFilter02Gene</td>
<td>896.7</td>
<td>894.5</td>
<td>885.3</td>
<td>908.1</td>
<td>Active</td>
</tr>
<tr>
<td>ActiveAdaptiveFilter01Gene</td>
<td>889.9</td>
<td>885.7</td>
<td>882.5</td>
<td>897.3</td>
<td>Active</td>
</tr>
<tr>
<td>MinJerkActive01Gene</td>
<td>888.0</td>
<td>883.7</td>
<td>859.1</td>
<td>916.8</td>
<td>Active</td>
</tr>
<tr>
<td>EulerLagrange03</td>
<td>887.4</td>
<td>884.0</td>
<td>912.0</td>
<td>862.7</td>
<td>Active</td>
</tr>
<tr>
<td>EulerLagrange04</td>
<td>874.0</td>
<td>865.7</td>
<td>894.8</td>
<td>853.2</td>
<td>Active</td>
</tr>
<tr>
<td>SlidingMode04</td>
<td>858.8</td>
<td>856.8</td>
<td>852.3</td>
<td>865.3</td>
<td>Active</td>
</tr>
<tr>
<td>MinJerkTargetDeltaActive01Gene</td>
<td>834.9</td>
<td>829.8</td>
<td>921.5</td>
<td>748.2</td>
<td>Active</td>
</tr>
<tr>
<td>xLandingEdgeCenterActive</td>
<td>826.6</td>
<td>821.2</td>
<td>775.7</td>
<td>877.5</td>
<td>Active</td>
</tr>
<tr>
<td>MinJerkTargetDeltaActiveRootTGene</td>
<td>825.4</td>
<td>824.3</td>
<td>891.1</td>
<td>759.8</td>
<td>Active</td>
</tr>
<tr>
<td>FlatLinearJerk01SkewSemiCross</td>
<td>823.9</td>
<td>822.8</td>
<td>776.6</td>
<td>871.2</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearJerkSimpleSemi02</td>
<td>820.8</td>
<td>815.3</td>
<td>722.6</td>
<td>919.0</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearJerk01SemiCross</td>
<td>819.5</td>
<td>815.2</td>
<td>686.3</td>
<td>952.8</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearJerkSpikeRemoval02</td>
<td>817.8</td>
<td>815.6</td>
<td>710.5</td>
<td>925.1</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>RCollisionAvoidSemi03</td>
<td>814.8</td>
<td>810.4</td>
<td>728.7</td>
<td>900.9</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearAcceleration01SkewSemiCross</td>
<td>813.8</td>
<td>810.2</td>
<td>727.3</td>
<td>900.3</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearJerkSimpleSemi05</td>
<td>812.2</td>
<td>807.3</td>
<td>713.1</td>
<td>911.2</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>PureSkyhookSemiCross</td>
<td>806.9</td>
<td>803.3</td>
<td>735.1</td>
<td>878.7</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>xMinTimeHardenSemiCross</td>
<td>806.7</td>
<td>804.2</td>
<td>737.5</td>
<td>876.0</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>xLandingEdgeCenterSemiCross</td>
<td>806.5</td>
<td>801.8</td>
<td>784.7</td>
<td>828.3</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearSemiJerk01</td>
<td>804.9</td>
<td>801.8</td>
<td>768.8</td>
<td>841.0</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearJerkSimpleSemi04p</td>
<td>803.7</td>
<td>797.3</td>
<td>725.5</td>
<td>882.0</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearJerkSimpleSemi01</td>
<td>802.4</td>
<td>797.3</td>
<td>791.0</td>
<td>813.9</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearJerkSpikeRemoval04</td>
<td>802.4</td>
<td>798.0</td>
<td>720.4</td>
<td>884.4</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>ActiveAdaptiveJerk01Gene</td>
<td>800.3</td>
<td>793.4</td>
<td>760.2</td>
<td>840.4</td>
<td>Active</td>
</tr>
<tr>
<td>FlatLinearAcceleration01SemiCross</td>
<td>797.8</td>
<td>794.9</td>
<td>699.5</td>
<td>896.1</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SlidingModeSemi01</td>
<td>797.4</td>
<td>793.8</td>
<td>786.5</td>
<td>808.3</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SlidingModeSemi02</td>
<td>788.8</td>
<td>782.1</td>
<td>670.4</td>
<td>907.1</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SlidingMode01</td>
<td>785.5</td>
<td>776.0</td>
<td>797.5</td>
<td>773.6</td>
<td>Active</td>
</tr>
<tr>
<td>SlidingMode05</td>
<td>775.4</td>
<td>769.7</td>
<td>754.6</td>
<td>796.2</td>
<td>Active</td>
</tr>
<tr>
<td>LoJerkConstantKAdaptiveJerkFilterGene</td>
<td>766.0</td>
<td>759.1</td>
<td>655.6</td>
<td>876.4</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SlidingMode03A</td>
<td>757.5</td>
<td>750.6</td>
<td>812.9</td>
<td>702.1</td>
<td>Active</td>
</tr>
<tr>
<td>RattlespaceCollisionActive04Gene</td>
<td>750.9</td>
<td>748.5</td>
<td>860.1</td>
<td>641.8</td>
<td>Active</td>
</tr>
<tr>
<td>EulerLagrangeSemi01</td>
<td>745.8</td>
<td>742.7</td>
<td>598.7</td>
<td>892.8</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SlidingMode03</td>
<td>744.8</td>
<td>733.4</td>
<td>881.1</td>
<td>608.5</td>
<td>Active</td>
</tr>
<tr>
<td>xMinTimeHardenSemi01</td>
<td>735.2</td>
<td>725.0</td>
<td>786.5</td>
<td>684.0</td>
<td>Active</td>
</tr>
<tr>
<td>FlatLinearJerkSimpleSemi04</td>
<td>734.2</td>
<td>712.8</td>
<td>543.8</td>
<td>924.5</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkJRangeGene</td>
<td>728.1</td>
<td>721.5</td>
<td>746.6</td>
<td>709.6</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>RCollisionAvoidSemi05</td>
<td>727.9</td>
<td>-2E22</td>
<td>529.5</td>
<td>926.3</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SlidingMode02</td>
<td>727.4</td>
<td>715.3</td>
<td>765.6</td>
<td>689.3</td>
<td>Active</td>
</tr>
<tr>
<td>RCollisionAvoidSemi05p</td>
<td>722.2</td>
<td>421.8</td>
<td>605.2</td>
<td>839.2</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearAccSimpleSemi04</td>
<td>719.5</td>
<td>715.1</td>
<td>625.3</td>
<td>813.7</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>MinJerkTargetDeltaFuzzySplineGene</td>
<td>716.1</td>
<td>711.9</td>
<td>829.8</td>
<td>602.4</td>
<td>Active</td>
</tr>
<tr>
<td>RCollisionAvoidSemi04</td>
<td>712.0</td>
<td>-1E95</td>
<td>549.2</td>
<td>874.7</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearJerkSimpleSemi03</td>
<td>550.5</td>
<td>543.0</td>
<td>337.3</td>
<td>763.6</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>FlatLinearJerk01SkewSemi</td>
<td>533.7</td>
<td>-4E115</td>
<td>397.7</td>
<td>669.8</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearAccSimpleSemi03</td>
<td>522.7</td>
<td>518.2</td>
<td>274.8</td>
<td>770.5</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SigmoidLinearSemiAcc01</td>
<td>442.5</td>
<td>430.3</td>
<td>244.4</td>
<td>640.6</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkSkyhookPassiveFilteredRoadGene</td>
<td>410.3</td>
<td>395.5</td>
<td>291.5</td>
<td>529.1</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SkyhookEmulationGene</td>
<td>386.2</td>
<td>377.6</td>
<td>255.7</td>
<td>516.8</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkFuzzySplineGene</td>
<td>383.7</td>
<td>360.4</td>
<td>291.2</td>
<td>476.2</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearAccSimpleSemi04p</td>
<td>375.1</td>
<td>360.4</td>
<td>198.9</td>
<td>551.3</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkConstantKActivePassiveGene</td>
<td>374.6</td>
<td>367.0</td>
<td>271.5</td>
<td>477.6</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkConstantKGene</td>
<td>368.3</td>
<td>355.5</td>
<td>278.7</td>
<td>457.8</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SkyhookGene</td>
<td>352.6</td>
<td>345.6</td>
<td>227.6</td>
<td>477.6</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkForceSkyhookGene</td>
<td>342.5</td>
<td>324.3</td>
<td>237.2</td>
<td>447.9</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkNonlinearFuzzyGene</td>
<td>338.9</td>
<td>-6.1E6</td>
<td>366.7</td>
<td>311.0</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>NoJerkForceSkyhookGene</td>
<td>326.8</td>
<td>302.9</td>
<td>77.0</td>
<td>576.5</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearAccSimpleSemi02</td>
<td>324.7</td>
<td>321.1</td>
<td>236.4</td>
<td>413.0</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearAccSimpleSemi05</td>
<td>322.3</td>
<td>319.9</td>
<td>224.2</td>
<td>420.5</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkNonlinearGene</td>
<td>321.7</td>
<td>313.8</td>
<td>223.3</td>
<td>420.1</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>PassiveGene</td>
<td>321.0</td>
<td>306.4</td>
<td>248.7</td>
<td>393.4</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SkyhookEmulationFilteredRoadLimitedSlope</td>
<td>318.7</td>
<td>-2E95</td>
<td>98.6</td>
<td>538.9</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FilterWithStiffeningSemi01</td>
<td>317.3</td>
<td>316.8</td>
<td>167.2</td>
<td>467.4</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkSkyhookGene</td>
<td>311.5</td>
<td>306.7</td>
<td>273.7</td>
<td>349.4</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>PassiveRolloff</td>
<td>305.5</td>
<td>297.1</td>
<td>240.7</td>
<td>370.3</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>EulerLagrangeSemi02</td>
<td>303.3</td>
<td>288.0</td>
<td>190.9</td>
<td>415.7</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>LoJerkNonlinearActivePassiveGene</td>
<td>299.9</td>
<td>275.0</td>
<td>398.4</td>
<td>201.3</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>StiffnessNearness</td>
<td>294.2</td>
<td>276.6</td>
<td>365.7</td>
<td>222.7</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>CSlopeWithK01</td>
<td>293.0</td>
<td>283.6</td>
<td>270.9</td>
<td>315.2</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SkyhookPassiveFilteredRoadGene</td>
<td>290.6</td>
<td>276.5</td>
<td>319.0</td>
<td>262.3</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SkyhookOnOffFilteredRoadGene</td>
<td>286.7</td>
<td>284.9</td>
<td>199.4</td>
<td>374.0</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SkyhookPassiveFilteredRoadLimitedSlopeGene</td>
<td>280.6</td>
<td>260.5</td>
<td>281.6</td>
<td>279.6</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearAccSimpleSemi01</td>
<td>255.7</td>
<td>256.7</td>
<td>311.5</td>
<td>199.9</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>TwoLinearFuzzyGene</td>
<td>244.4</td>
<td>231.8</td>
<td>191.0</td>
<td>297.8</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>RCollisionAvoidSemi01</td>
<td>234.3</td>
<td>204.1</td>
<td>99.9</td>
<td>368.7</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>NoJerkSkyhookGene</td>
<td>220.8</td>
<td>210.1</td>
<td>-64.1</td>
<td>505.6</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>AdaptiveFuzzyDamperWithDecayGene</td>
<td>180.4</td>
<td>-640</td>
<td>-177.4</td>
<td>538.1</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>RCollisionAvoidSemi03AC</td>
<td>-414.1</td>
<td>-431</td>
<td>-438.5</td>
<td>-389.7</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>RCollisionAvoid03</td>
<td>-1539.0</td>
<td>-1527</td>
<td>769.6</td>
<td>-3847.5</td>
<td>Active</td>
</tr>
<tr>
<td>FlatLinearJerk01Semi</td>
<td>-3312.8</td>
<td>-2E12</td>
<td>-4634.1</td>
<td>-1991.6</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SkyhookPassiveSpikeRemovalGene</td>
<td>-3899.4</td>
<td>-2E26</td>
<td>-3780.5</td>
<td>-4018.3</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>FlatLinearAcceleration01Semi</td>
<td>-4095.0</td>
<td>-4.8E5</td>
<td>-6645.0</td>
<td>-1545.0</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>RCollisionAvoidJerk01</td>
<td>-4956.6</td>
<td>-6105</td>
<td>-921.1</td>
<td>-8992.1</td>
<td>Active</td>
</tr>
<tr>
<td>RCollisionAvoid02</td>
<td>-9613.1</td>
<td>-9650</td>
<td>200.7</td>
<td>-19426.8</td>
<td>Active</td>
</tr>
<tr>
<td>FlatLinearAccSimpleSemi00</td>
<td>-11326</td>
<td>-3E28</td>
<td>-11365.1</td>
<td>-11287.6</td>
<td>Semi-Active</td>
</tr>
<tr>
<td>SkyhookPassiveGene</td>
<td>-12906</td>
<td>-3E27</td>
<td>-17679.8</td>
<td>-8131.9</td>
<td>Semi-Active</td>
</tr>
</tbody>
</table>
The final scores were taken from the median results for the final generation. The median scoring gene was used rather than the mean to avoid skewing of results due to the fact that some generations may have a small proportion of genes that encounter very rough roads and obtain extremely low scores. The means are shown above, and the effect of skewing is clear in some cases where there is a large difference between the mean and the median. The overall score then is an average of the smoothness and tracking scores of the median element, and these scores are also shown above.

The data has been collected from the final generation of a long process of evolution. The final generation has been run with very “cool” parameters, which is to say that there is very little mutation in the final stages. Furthermore, the selection process in these final stages employs highly elitist selection and lower genetic variability, so the final measurement is most likely to apply to the best performing genes.

Even so, there is some variability in the results due mainly to variations in roads encountered by the controls. This variability can be approximated by taking the standard deviation of a sample of the final ten generations of the genetic evolution. Nine generations were chosen.

The median scores of the final 9 generations for the highest scoring control, FlatLinearJerk01Skew, are:

982.13, 982.78, 980.49, 982.06, 980.8, 980.92, 981.25, 982.36, 982.06.
The standard deviation for these results is 0.75. So the results are very tightly clustered. The standard deviation of the last 9 median scores for the benchmark ActivePureSkyhookGene control is 2.9. The standard deviation for the other benchmark, PassiveGene, is 11.1. Thus as the controls perform better, there is less variation in the results, as would be expected. The ordering of the performance scores must be judged with this slight variability in mind.

The evolved control parameters for the various genes can be viewed by running the EA test bed program, SuspensionTest, clicking on Tools|Parameters, selecting the desired algorithm, and then viewing the parameters in View|Current Generation. Some kind of graphical representation may accompany the data.

Firstly, the results for linear controls and “modified” linear controls are described. The main modifications take the form of the inclusion of virtual bump stops, and crossover removal in the case of semi-active controls.

There are three controls that are instances of purely linear control over acceleration, and this includes the two benchmarks. Table 6 summarises the control parameters found by the evolutionary algorithm for the highest performing gene instance in the final generation of the passive (PassiveGene), skyhook (ActivePureSkyhookGene) and the general linear control (FlatLinearAcceleration01). In a sense the skyhook has a damping coefficient for a damper attached between the chassis and “sky”, instead of between the wheel and chassis (stroke). The most general linear control over acceleration could be thought of as having two springs and two dampers, one each attached to the sky and the wheel. (It should perhaps be repeated that springs or dampers cannot actually be attached to the sky, but control forces can be implemented by electronic systems which produce the same forces as if there was a spring or damper attached to the sky.)
Control | Spring Rate (Sky) $k$ Nm$^{-1}$ | Damper Rate (Sky) $c$ Nsm$^{-1}$ | Spring Rate (Stroke) $k$ Nm$^{-1}$ | Damper Rate (Stroke) $c$ Nsm$^{-1}$ | EA Score
---|---|---|---|---|---
PassiveGene | - | - | 917.7 | 70.47 | 321.0
ActivePure-SkyhookGene | - | 373.3 | 1,005.4 | - | 958.3
FlatLinear-Acceleration01 | 81.0 | 332.0 | 980.5 | 4.329 | 976.4

Table 6 Comparison of Linear Controls over Acceleration

It is clear from this data that the final control in the table has coefficients almost exactly like the skyhook; the stroke spring rate and the damping rate of the damper attached to the sky are almost the same as the skyhook, while the other two components are relatively much smaller. In fact the damping rate for the damper “attached to the chassis”, as in a passive suspension, is extremely low, 4.329, indicating that it is the damper that is the most challenging component in a passive and semi-active system (refer to the discussion in section 4.3).

It is interesting to compare the purely linear control over acceleration with the purely linear control over jerk. Table 7 below shows the evolved coefficients for the control over jerk. These are the coefficients as represented in equation 4.7. (In the following table, to be consistent with equation 4.7, the control $u$ is taken as the control directly over jerk, which is why Newtons do not appear in the units.)

| Coefficient | $\beta_1$ | $\beta_2$ | $\beta_3$ | $\beta_4$ | $\beta_5$ | $\beta_6$
|---|---|---|---|---|---|---
| Term | $y$ (Sky) | $\dot{y}$ (Sky) | $\ddot{y}$ (Sky) | $s$ (Stroke) | $\dot{s}$ (Stroke) | $\ddot{s}$ (Stroke)
| Units | s$^{-3}$ | s$^{-2}$ | s$^{-1}$ | s$^{-3}$ | s$^{-2}$ | s$^{-1}$
| Value | 0.00901 | 0.304 | 1.885 | 1.303 | 3.502 | 0.00475

Table 7 Coefficients for Linear Control over Jerk

As with the skyhook, coefficients for the stroke tend to favour the lower order derivatives: $s$ and $\dot{s}$. In fact the coefficient of stroke acceleration, $\ddot{s}$, is virtually zero. On the other hand the coefficients for absolute chassis movement are larger with the higher derivatives: $\dot{y}$ and $\ddot{y}$. The coefficient of absolute chassis movement, $y$, is almost zero. (This bodes well for control
implementation since the acceleration of the chassis is directly measured with an accelerometer, while the other two variables, \( y \) and \( \dot{y} \), depend on numerical integration. Indeed, the highly error-prone double integration can be dropped completely.)

The effect of including virtual bump stops is examined next, specifically the virtual bump stops using the skew function represented by figure 4.7 and explained in section 4.5.2.2. Recall that the function will apply the increase only if the physical parameter has the same sign as the stroke. (So if the chassis is close to a rattlespace limit but moving away from it, the increase in control strength is not applied.)

The control coefficients for the FlatLinearAcceleration01 and the FlatLinearAcceleration01Skew controls are represented schematically below in figure 5.6. Recall that the coefficients in the skewed example change with stroke, \( s \). (It is assumed below that the parameters, \( y \), \( \dot{y} \), \( s \), and \( \dot{s} \), are positive, otherwise the skewing is reversed.) It is clear that the linear control becomes roughly a stiffer version of a skyhook control as the suspension approaches the rattlespace limit. The virtual bump stop created a modest improvement from 976 to 978, and produced the second highest scoring control.

Virtual bump stops for jerk control also improved performance, from 965 to 982. Again there was a stiffening of jerk control near the rattlespace limit. In the end, the linear control over jerk with virtual bump stops was the highest scoring control of all.

![Figure 5.6 Coefficients for Linear Control over Acceleration](image-url)
As expected, when simply clipped for a semi-active system, the linear controls performed very badly. The FlatLinearAcceleration01Semi control obtained a score of -4,095.0, and the FlatLinearAcceleration01SkewSemi obtained a score of -21,216.

When crossover jerk removal was included, the scores improved dramatically. The FlatLinearAcceleration01SemiCross and FlatLinearAcceleration01SkewSemiCross both applied the crossover removal algorithm described in section 4.8.3. The parameters for crossover removal were also determined by the EA. The FlatLinearAcceleration01SemiCross obtained a score of 797.8, and the score for the FlatLinearAcceleration01SkewSemiCross control was 813.8. These performed vastly better than the benchmark semi-active systems, the Passive, with a score of 321.

These were among the highest performing semi-active systems. They were eclipsed by the jerk analogues of the controls mentioned in the previous paragraph:

The FlatLinearJerk01SemiCross with score 819.5 and,
FlatLinearJerk01SkewSemiCross with score 823.9.

In summary the following results are clear for linear systems or “modified” linear systems:

- Linear systems perform well overall,
- Controls over jerk outperformed controls over acceleration,
- Bump stops or virtual bump stops improved performance,
- Some form of crossover jerk removal is necessary for semi-active systems.

In fact these results were consistent in evolutionary algorithms performed for papers published during the research.

The improvement created by using crossover jerk removal is obvious. A number of crossover jerk removal methods have been attempted. Some of these have been discussed in the papers published by the author during the PhD research. At one end is the simple global control, the no-jerk skyhook, and at the other is the method described in section 4.8.3.
The no-jerk skyhook (NoJerkSkyhookGene) in fact did not perform as well as the passive. The NoJerkSkyhookGene scored 221 while the Passive scored 321. This indicates that although it removes crossover jerk, it is not a high-performance method globally. When the method was used with maximum and minimum damping rate limits, the NoJerkForceSkyhookGene, the method performed only slightly better than the passive. (Note that when these limits are close this method is virtually indistinguishable from the passive, so this can allow the EA to find a compromise between the two.)

Variations of the author’s lo-jerk skyhook (discussed in section 4.8.3) performed somewhat better. The “linear lo-jerk skyhook” is a novel global control developed by the author using the algorithm,

\[
I = \begin{cases} 
K\ddot{y} & \text{where } \dot{y} > 0, \\
0 & \text{otherwise.}
\end{cases}
\]

(Storey et al., 2006). This produced some improvement over the no-jerk skyhook. The LoJerkConstantKGene, which uses this algorithm with maximum and minimum damping rates, scored 368, slightly better than the passive. The highest scoring lo-jerk variant, the LoJerkConstantKAdaptiveJerkFilterGene obtained a respectable score of 766.

When a crossover jerk removal algorithm was applied to modify a high-performance “target” control however, there was much greater success. Such methods use a local crossover method in conjunction with a high-performance “target” method. For this reason the crossover algorithm is here referred to as a “local” method. They are “activated” when crossover is imminent, especially when moving out of the operational range of the target method, as described in section 4.8.

During the course of the research a number of crossover removal methods have been invented by the author and have been used in the numerical experiments. For example, the FlatLinearJerkSpikeRemoval04 is a linear control over jerk that used an old method to anticipate crossover, and it obtained a respectable score for a semi-active control: 802.

Initial attempts at crossover removal used heuristic approaches. The theory of section 4.8.3 developed after looking more carefully at the physics of the problem. The use of a more complete physical model was rewarded with the most successful crossover reduction methods.
of the numerical experiments. In fact the highest performing semi-active suspension, the FlatLinearJerk01SkewSemiCross, uses this crossover algorithm with a modified linear target control (linear control over jerk with a virtual bump stop) achieving a score of 824.

All else being equal it should be expected that the semi-active method would perform worse than the target method, at least if the target is a high-performance control (in some cases where the target is a very poor performer, it may happen that the crossover removal acts to soften and improve the control, but there is little point in examining such cases). Furthermore, all else being equal, a semi-active control with crossover removal should perform better than the target control that is merely clipped for a semi-active suspension.

This expectation was supported by experiment. This can be seen in the relative performances in table 8 below.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Target Control</th>
<th>Type</th>
<th>Crossover</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlatLinearAcceleration01</td>
<td>Linear over Acceleration</td>
<td>Active</td>
<td>-</td>
<td>976.4</td>
</tr>
<tr>
<td>FlatLinearAcceleration01SemiCross</td>
<td>Linear over Acceleration</td>
<td>Semi-Active</td>
<td>Yes</td>
<td>797.8</td>
</tr>
<tr>
<td>FlatLinearAcceleration01Semi</td>
<td>Linear over Acceleration</td>
<td>Semi-Active</td>
<td>No</td>
<td>-4095.0</td>
</tr>
<tr>
<td>FlatLinearJerk01</td>
<td>Linear over Jerk</td>
<td>Active</td>
<td>-</td>
<td>965.0</td>
</tr>
<tr>
<td>FlatLinearJerk01SemiCross</td>
<td>Linear over Jerk</td>
<td>Semi-Active</td>
<td>Yes</td>
<td>819.5</td>
</tr>
<tr>
<td>FlatLinearJerk01Semi</td>
<td>Linear over Jerk</td>
<td>Semi-Active</td>
<td>No</td>
<td>-3312.8</td>
</tr>
</tbody>
</table>

Table 8 Comparison of Linear Controls

There are two highly performing active controls which were based on theoretical notions that are not explicitly linear but which actually are equivalent to special cases of linear controls with virtual bump stops. As could be expected, these did not perform as well as the general linear controls, but they performed quite well and they may indicate significant underlying theoretical factors.

This is especially the case for the ActiveAdaptiveJerkFilterGene control with a score of 974.4. This kind of control was described in the paper published during the research in
2006 (Storey et al., 2006). It was designed as an initial form of rattlespace constraint, but it uses decay rates that are based loosely on what could be described as varying exponential decay rates (for either acceleration or jerk). In the end, the control comes down to a less flexible variant of a linear control with a virtual damper. The relatively high performance of this somewhat simpler control however indicates that variable rates of exponential decay could be used as a simple method of suspension control. This notion was not carried further however when it was found that the method was a special case of the more general linear method with variable stiffness.

BackMomentumActive01Gene performs only slightly better than the benchmark skyhook but is much more complex. The control increases the return jerk based on a complex combination of stroke, absolute chassis height and chassis acceleration. This uses third-order functions depending on stroke and is again a special case of a linear control with virtual bump stop.

Now the nonlinear, active controls are examined. These are controls that are nonlinear and are not “modifications of the linear control”.

The highest performing nonlinear method is RCollisionAvoidJerk03 with a score of 977.5. This is a variable hardness form of rattlespace constraint control described in section 4.7.5. This particular control uses constant jerk to avoid the rattlespace limit. It has two modes, a hard mode, when collision with the rattlespace is imminent, and a soft mode when there is little danger of collision with the rattlespace. Given that this method is the third highest scoring control, this is a control worthy of further investigation.

The control EdgeJerk01 performs only slightly less well than the benchmark skyhook; it scores 948 compared to the skyhook’s 958. The EdgeJerk01 control uses complex edge constraint methods as discussed in section 4.7.4. Recall that this method attempts to calculate the control “strength” (jerk in this case) to maintain the trajectory within the anticipated rattlespace. This control uses rattlespace edge targeting discussed in section 4.7.4 with edge filtering (as illustrated in figure 4.44 and performed in the Java class RattlespaceTarget01).
Next the semi-active, nonlinear controls are examined. Most of these are simply semi-active adaptations of active target controls, with crossover removal added in. The
RCollisionAvoidSemi03 control for instance, which scores 815, is similar to the
RCollisionAvoidJerk03 control with a form of crossover removal.

There are two controls that deserve mention which are high performing semi-active variants of rattlespace constraint controls: the xMinTimeHardenSemiCross which scored 806.74, and the xLandingEdgeCenterSemiCross which scored 806.51. The
xMinTimeHardenSemiCross uses a variant of edge constraint similar to the EdgeJerk01 control, except that the jerk strength is calculated based on the maximum travel of the minimum-time, bang-bang jerk control (described in section 4.6). It is difficult to calculate the jerk needed to reach a given distance, so an iterative approach is used with the jerk increasing in each step until the suspension is stiff enough to avoid colliding with the rattlespace limit. Rattlespace edge filtering is also used (as described in section 4.7.4).

The xLandingEdgeCenterSemiCross control was perhaps the last control developed, and it is in many ways incomplete. This control was based on the more mature theory developed in section 4.7. That this first attempt could perform so well is somewhat encouraging, although admittedly it scored lower than the simpler linear controls. This is an edge constraint method which uses the landing-surface jerk to determine the jerk required to avoid an edge collision. This has the advantage, as discussed in section 4.7.3.3, that rebound is minimized and hence stability problems potentially avoided.

The semi active variants just mentioned performed relatively better than their active variants, xMinTimeHarden and xLandingEdgeCenter, when compared to the linear controls. This may be because the passivity constraint is less likely to affect controls when they stiffen on approach to the rattlespace limits, as discussed in section 4.8.4.
6. Physical Experiment

The physical experimental rig was designed for testing various control algorithms applied to a semi-active suspension. A reprogrammable MCU (microcontroller unit) provides electronic control. Ironically, as this thesis was being researched a new range of MCUs from Atmel, the XMEGA range, was developed.

The use of an MCU, acting independently of a PC, demonstrates that the algorithms employed here are not too demanding to be employed on board a dedicated MCU. It should perhaps again be emphasised that processor-hungry evolutionary algorithms have been used for optimization of control algorithms, they have not themselves been considered for on-board control.

Preliminary designs for the physical rig were discussed with Peter Tkatchyk (see figure 8.42), a lab technician at RMIT Bundoora campus. The rig needs to be a fair test of the algorithms in as realistic a setting as possible given constraints of time and a very limited budget. The first designs involved shakers that could shake a rig containing a small controllable damper and a spring. Commercially available controllable dampers, however, are not small enough to make this kind of rig feasible.

It was decided to try for a rig that could be mounted onto a small trailer, using the trailer as the unsprung mass analogue and a suspended mass as the chassis analogue. The suspended mass was comprised of gym weights in the final design.

A number of rig designs were considered. One possibility was to simply place a spring and damper under a weight and mount them on a base. This arrangement however would need precise mechanical fitting if it was not to shake, jam, or fall over. This was considered too unwieldy.

Another possibility is simply to turn the suspension upside down and literally suspend the load weight. In fact, the term “suspension” derives from the fact that carriages were originally
suspended below their supports (Bastow et al., 2004, p3). This orientation would be extremely simple to engineer with no effect on the viability of experimentation. The only difference is the spring equilibrium force. Even so, it was considered that this might not be as convincing a test of suspension control as one that operated in the conventional orientation.

The final design settled on was a pivoted arm suspended above a base. Gym weights could be added to the arm to simulate the chassis weight, as explained in the following sections. The weight at the top of the experimental rig is a chassis analogue. For brevity, in this context this weight will sometimes simply be referred to as the “chassis”. Similarly the distance between the top and base may be referred to as the “stroke”. This design has the advantage that effective spring and damper rate can be altered by moving them relative to the pivot.

While it proved quite easy to perform experiments to determine the damper parameters and to control the force of the suspension system, the ability to test the rig over terrain proved quite difficult. Specifically it proved difficult in the constraints of the experiment to get absolutely uniform tests over rough terrain. It was decided therefor to use long runs for this kind of experiment and summarise the results. The controls that bring a suspended system smoothly to rest, on the other hand, could be repeated reliably in the lab and are reliable results for a control algorithm that may have substantial commercial applications, and has great relevance for suspension systems. The experiments over rough terrain however do suffice as a proof that the algorithms can be applied reliably using even very cheap accelerometers and microprocessors. Sophisticated real-time optimal control is viable with modern equipment: the numerical experiments show that even relatively simple modifications to linear controls are worthy of further investigation.

6.1. Experimental Rig

The physical frame was built at RMIT Bundoora campus by Peter Tkatchyk (see figure 8.42) according to this author’s original design, shown below in figure 6.1. The completed rig (unpainted) is shown in figure 6.2. Load weights are placed on an arm attached to a pivot mounted on a frame.
Figure 6.1 Original Design for the Experimental Rig

Figure 6.2 Physical Rig
6.1.1. Controllable Damper

Perhaps the most crucial component of the experimental rig is the controllable damper. The damper has been supplied by Lord Corporation (Lord, 2009) and is an RD-1005-3 damper. Note that the datasheet for the damper supplied by Lord Corporation was available from the website at the time of purchase. (It can be found in the supplied material: PhD\Experiment\Lord RD 1005 3 Damper.pdf.) The damper is depicted in figure 8.43 and can be seen attached to the rig in figure 8.42 and figure 6.2 above. The damper’s technical datasheet states that, “Continuously variable damping is controlled by the increase in yield strength of the MR fluid in response to magnetic field strength.” The strength of the magnetic field and the subsequent force applied by the damper are determined by the current supplied by the damper.

In order to simplify the electronic control of the damper, Lord supplied a component which takes a 5 volt input signal and produces a 12 volt current output appropriate for the control of the damper, the LORD Wonder Box™ Device Controller Kit, RD-3002-03 (Lord, 2008) shown in Figure 8.44. (The datasheet for the controller can be found in PhD\Experiment\LORD Wonder Box Device Controller Kit.pdf.) The BNC connector, at the right in figure 8.44, takes a 5 volt signal (as supplied by the microcontroller in the final experiments) and outputs the 12 volt control via the banana plug connectors at the top. The unit comes supplied with a 12 volt power plug pack (with a US mains plug). This was swapped for a current-limited power supply.

Given the number of input and output transducers, a simple “jiffy box” was used to house the controller (refer to figure 6.3). This unit was designed and built by this author. This box also housed a 5 volt regulator, and took power from a 12 volt supply. Apart from the 5 volt regulator, the main purpose of the box is simply to tidy up some of the connections to the microcontroller and the Lord controller. The signals to and from the microcontroller were placed inside a shielded cable. As discussed below, the original shielding for the potentiometer output was insufficient and more careful shielding was needed. Note that the microcontroller has been housed separately in order to expedite reprogramming in
experiments, but the MCU could easily be run independently of the PC once a suspension control algorithm has been programmed.

Figure 6.3  Control Box for Rig

The controller box from Lord Corporation was designed to handle a 5 volt signal input, and a 5 volt PWM signal from the MCU was used for control. A faint but distinct high-pitched buzzing sound emanated from the damper under load when the raw PWM signal was used, no doubt with a harmonic frequency of the duty-cycle rectangular wave. An attempt was made to use active components for the PWM filter to remove the sound and supply the power needed, but eventually a simple passive low-pass filter was found to be sufficient (\( R = 330\Omega, C = 1.5\mu F \)). The sound disappeared.

According to the Lord website (Lord, 2009), “If [the damper coils] are left on too long (perhaps 15 minutes) at 2 amps, they will get very hot. If one limits the current to say, 1 amp, this would be better.” A current-limited power supply was purchased for the damper supply. This was used to supply the 12 volt supply, but the current output was limited at slightly less than 0.5 amps during experimentation. More than once during experimentation this possibly saved the damper coils from overheating, because in the course of complex experiments it is possible to unknowingly leave the coil on for long periods of time (the power supply’s ammeter was never observed to read above 0.5 amps).
6.2. *Electronics for Control*

A number of options were considered for the electronics to read movement data and for outputting to the controllable damper. A high priority for this experiment was a low time gap between MCU input and output. As explained in section 2.10.1, the damper has a latency of at most 10 ms under the most unfavourable conditions (claimed by Lord Corporation). The goal was to keep MCU latency lower than this if possible. As a rule of thumb, a vehicle travelling at 100 km/hr will traverse approximately 28mm (approximately one inch) in each millisecond (1 ms). Latency therefore limits the size of the bump that can be responded to by an electronic system. A latency of 10 ms corresponds to being unable to respond to a 10 inch bump, at that speed.

The simplest options for controlling the experimental rig all seemed to involve high latencies. A “Parallel Port Interface”, sold by Dick Smith (K2805) (Dick Smith Electronics Limited, 2006, p239), has 10 analog inputs and two analog outputs and allows direct control via a computer, but the serial port is too slow for our purposes and, in any case, negates the benefits discussed above of using an MCU.

Distance measuring systems suffer similar problems. For example, laser mice mechanisms were considered for providing distance measurements. However, mice also use serial communication with high latency. In the end a simple potentiometer was used for measuring the distance movement between the end top arm and the base.

There are a number of possible microcontrollers (MCUs) that can be used to provide a proof of concept for the on-board control of physical systems: the 8051 family, the Microchip PIC range, Atmel AVRs, the Texas Instruments MSP430 family, the Rabbit Semiconductor range of MCUs (Edwards, 2005, p16), and others. The chip used here is the ATmega644 from the Atmel AVR range. This chip has multiple analog inputs and outputs, and it allows ISP (In-System Programming), which is to say that it can be programmed and reprogrammed while in the target circuit. The ATmega644 contains 64K of flash memory, which is reprogrammable effectively indefinitely (Barnett et al., 2006). In the experiments of this thesis an STK500 development board from Atmel, shown in figure 6.4, was used as the development platform.
for the electronic control. The board also contains a number of input and output connectors typical of MCU development boards.

Figure 6.4 STK 500.

Any one from a large range of ATmega processors can be accommodated by this board. Figure 6.4 shows an ATmega644 as used in the experiments. The ATmega644 contains 64K of memory and this was sufficient to run a menu system that allowed various features to be turned on and off without significant reprogramming, although the process of transferring the machine code to the chip via RS232 was somewhat slow.

Programming using ISP is usually performed from a computer via an RS232 or USB serial connection. Here, an RS232 connection was used (highlighted in figure 6.4). With ISP, the same rig can be run with many different control algorithms; C programs are compiled on a computer, and MCU machine code is transferred to the microcontroller via the RS232 connection dedicated to ISP (marked RS232 CTRL on the development board).

Using this technique, any processing on the PC is clearly independent of the microprocessor control of the suspension. The suspension control is thus in an environment as close to the target as possible, providing verification that the algorithms used are simple enough and fast enough to apply using a microprocessor.
The STK has two RS232 ports, one for ISP during reprogramming and one for USART communication as programs are running, which communicates via a standard “terminal” program (a freeware program called “Term” was used). The USART communication was invaluable for experimental purposes, although it has no role in the suspension control. It is becoming more-and-more difficult to obtain a computer with two RS232 ports (also called “com” ports). During experimentation a new computer was purchased that had no com ports, so a card containing two com ports had to be installed.

The program on the PC for compiling C code and for transferring machine code to the MCU is AVR Studio, which was downloaded from the Atmel website at,

http://www.atmel.com/dyn/Products/tools_card.asp?tool_id=2725

AVR Studio is an integrated development environment for developing source code in C and transferring the compiled machine code to the AVR microprocessor. The Atmel website is also the source of the datasheets for the ATmega range of chips, such as the ATmega644 used in the experiments.

It was necessary to become familiar with the interrupts, and the analog-to-digital conversions of the AVR chips. A couple of texts were useful, particularly Barnett et al. (2006) and Pardue (2005). Atmel’s website (www.atmel.com) also had useful examples, but a number of other sites were also helpful. The following site contains useful code:

http://winavr.scienceprog.com/

Another good source is,

http://main.linuxfocus.org/common/src/article231/haraleit.pdf

The following site from Atmel has many links to useful data.


The following has some very useful hints and shows a deep experience with AVR programming.

http://www.nongnu.org/avr-libc/user-manual/FAQ.html

(Web sources were available at time of writing.) Written notes were compiled during the process of becoming familiar with some of the range of the ATmega chip family can be found in the appended document “AVR ATmega on the STK500”, in the word file “AVR Notes.doc” (in the directory PhD\Experiment\Electronics\Atmel AVR\My AVR Notes).
ADC (Analog to Digital Conversion) allows analog voltage values to be entered for processing in programs on board the MCU. ADC employs a process of counting steps as a fraction of a reference voltage. For example, if a 10-bit ADC produces the value 960, and the reference voltage is 5 V, then the input voltage is calculated using the following equation,

$$\frac{960}{2^{10} - 1} \times 5.0 = \frac{960}{1023} \times 5.0 = 4.692\text{V}.$$  

The highest possible accuracy is half the voltage of a single step size. In the example above this is approximately 2.5 mV. In practice, inaccuracies are greater due to inaccuracies in the ADC process, but they can also depend on the cleaness of the reference voltage supply used by the ADC. Oversampling can greatly reduce error (Pardue, 2005, p217). Refer to this author’s notes on oversampling in “AVR ATmega on the STK500”.

Analog output is achieved by using PWM (Pulse Width Modulation). A repeating rectangular pulse is produced that has a duty-cycle fraction giving an average voltage with the desired value. So if the required output voltage is low, the fraction of the time that the square wave has a high voltage is proportionally low. Both PWM and ADC on the ATmega644 can have 8-, 9- or 10-bit resolution, although only one timer can cope with greater than 8-bit counting and the number of simultaneous peripherals with 9- and 10-bit resolution is limited. In the experiments performed here the highest output frequency required is in the low audio range, 1 kHz at the very most, and the square-wave output of the PWM is well within the specifications required to produce a smooth output voltage at this frequency.

According to the datasheet, the on-board timers of the ATmega MCUs cannot be relied on for accuracy, although they are very regular (with constant voltage on the VCC pin and constant temperature). To calibrate the on-board timer a unit was built around a timer circuit from a cheap digital clock with an analog readout. The power circuit supplying the 1.5 V to the original clock circuit is simply a voltage divider with a capacitor across the supply. The original clock uses such a small amount of power that no further regulation was needed. The unit has a 1.5 V spike that crosses 2.5 V and so switches between on and off when read by the MCU. This unit supplies a regular spike every two seconds. An oscilloscope was used to check the timing in the finished unit.

The timer can be used to count events in the MCU over a two-second period. The code used for this can be found in the appendix, section 8.17. This can be used then to calibrate the
frequency of timers in the MCU. For example, if the count is regular and 200 events are counted in a two second period then the frequency of the timer is 100 Hz. As it happened, experiments with the timer showed that the on-board timer was accurate enough for the purposes of this thesis and did not require adjustment. (The program used for determining this is contained in the directory,

```
PhD/Experiment/Electronics/Atmel AVR/
```
in the file,

```
Test26 Timer01 - Calibrate Timer/CalibrateTimer0.c
```

### 6.3. Measurement Calibration

A potentiometer was employed to read the “stroke”. The “stroke” here is the displacement between the top arm (representing the chassis) and the base (representing the wheel) measured at the end of the arm. With small stroke and small angular arm movement, the damper and spring extensions are effectively linearly related to the arm movement.

Fishing line was attached to the moving arm of the rig and wound round a pulley attached to potentiometer attached to the base of the rig, as shown in figure 6.5. A second pulley is attached to a spring which maintains tension in the wire. The potentiometer was connected as a voltage divider with the voltage output proportional to the arm movement.
A “digital oscilloscope” has been used here for diagnostic purposes; it has no role in control. The device used was a PoScope Basic (www.poscope.com) which is very affordable. The PoScope has enough accuracy for the frequencies covered by the experiment, which are well within the audio range (say less than 1 KHz). (Figure 8.45 shows the PoScope unit: two input test leads feed into the BNC connectors on the front, and the USB connector on the right provides a channel for control by the PC, and for data acquisition.)

Figure 6.6 shows an example screenshot of the PoScope software’s display on the computer after capturing stroke movement data. This software can be used to transfer data into a text (ASCII) file for processing by other computer programs. A portion of such a file is shown in figure 6.7.
Figure 6.6 PoScope Digital Oscilloscope Screenshot

<table>
<thead>
<tr>
<th>Time, ms</th>
<th>Channel 1</th>
<th>Channel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>-98</td>
<td>1.42</td>
<td></td>
</tr>
<tr>
<td>-96</td>
<td>1.42</td>
<td>0.691</td>
</tr>
<tr>
<td>-94</td>
<td>1.42</td>
<td>0.691</td>
</tr>
<tr>
<td>-92</td>
<td>1.463</td>
<td>0.777</td>
</tr>
<tr>
<td>-90</td>
<td>1.42</td>
<td>0.691</td>
</tr>
<tr>
<td>-88</td>
<td>1.463</td>
<td>0.605</td>
</tr>
<tr>
<td>-86</td>
<td>1.42</td>
<td>0.691</td>
</tr>
<tr>
<td>-84</td>
<td>1.42</td>
<td>0.691</td>
</tr>
<tr>
<td>-82</td>
<td>1.463</td>
<td>0.734</td>
</tr>
<tr>
<td>-80</td>
<td>1.377</td>
<td>0.691</td>
</tr>
<tr>
<td>-78</td>
<td>1.463</td>
<td>0.691</td>
</tr>
<tr>
<td>-76</td>
<td>1.463</td>
<td>0.734</td>
</tr>
<tr>
<td>-74</td>
<td>1.42</td>
<td>0.734</td>
</tr>
<tr>
<td>-72</td>
<td>1.42</td>
<td>0.691</td>
</tr>
<tr>
<td>-70</td>
<td>1.463</td>
<td>0.734</td>
</tr>
<tr>
<td>-68</td>
<td>1.463</td>
<td>0.691</td>
</tr>
<tr>
<td>-66</td>
<td>1.42</td>
<td>0.648</td>
</tr>
<tr>
<td>-64</td>
<td>1.42</td>
<td>0.691</td>
</tr>
<tr>
<td>-62</td>
<td>1.42</td>
<td>0.734</td>
</tr>
<tr>
<td>-60</td>
<td>1.42</td>
<td>0.734</td>
</tr>
<tr>
<td>-58</td>
<td>1.42</td>
<td>0.691</td>
</tr>
<tr>
<td>-56</td>
<td>1.463</td>
<td>0.691</td>
</tr>
</tbody>
</table>

Figure 6.7 Portion of a Text File of Oscilloscope Data Points
It is important to distinguish between voltage data being used in two places. Firstly, voltage data is collected off-board, on the computer, via the oscilloscope; this data is used for calibration, verification and for determining effective spring rate as well as damper characteristics. A Java program, ProcessScope, (written by this author) was used for processing data off-board. Secondly, voltage data is collected on-board the MCU, via the ADC; this data is destined to be used in the on-board control algorithms. The MCU requires velocity and acceleration measures as well as distance. C programs compiled for the ATmega chip, are transferred to the MCU via ISP, as described earlier. These programs are used to control the damper in the rig.

Initially, the potentiometer responsible for measuring distance was unshielded, as shown above in figure 6.5. The potentiometer output contained a well-defined “50-cycle hum” shown in figure 6.8. This is indicative of poor grounding (Australian mains power supply is 50 Hz). To attempt to reduce this hum, shielded boxes were employed and analog data was transferred using balanced shielded XLR connectors as often employed for small signal audio wiring. The shielded potentiometer setup is shown in Figure 8.46. An excellent note on the wiring of XLR cables can be found at the following website,

http://www.rane.com/notell0.html

This method very effectively eliminated interference.

Figure 6.8 50 Hz Interference on Original Potentiometer Input
6.3.1. Distance Calibration

A linear potentiometer was used for measuring displacement. Assuming linearity, the relationship between stroke and voltage can be expressed as,

\[ s = \alpha v. \]

**Equation 6.1**

Here, \( \alpha \) is a constant linear coefficient for converting from voltage, \( v \), to stroke, \( s \). As is usual practice for suspension theory, the “zero” stroke displacement is the equilibrium position under load. In the experiments performed later a “zero” voltage was determined simply by reading the potentiometer output with the rig sitting still at equilibrium.

The value of the conversion parameter, \( \alpha \), was determined by experiments that measured voltage and displacement as the rig was loaded with different masses. The plot of actual voltage vs. displacement is shown in figure 6.9 (the data has been collated in the Excel spreadsheet “Volts Distance Mass” in the directory PhD\Experiment\Rig).

![Figure 6.9 Voltage vs. Displacement (cm)](image)

When regression is performed on displacement versus voltage a slope of -2.66785 is found (with a correlation coefficient of about -0.998). This needs to be divided by 100 to convert centimetres to metres, and multiplied by the ratio of the distance from the pivot to the mass centre and the measuring point respectively (measuring at the mass centre is inconvenient and more error prone). The approximate value for \( \alpha \) therefore is,
\[
\alpha \approx -0.01944 \text{ m/V.}
\]

A one volt difference corresponds to roughly a 2 cm movement.

It is convenient below to work in terms of voltage and to later convert back to physical values using equation 6.1 if needed.

### 6.3.2. MCU Voltage Calibration

The analog voltage read by the MCU is converted to a digital value using ADC as described in section 6.2. This is a straight-forward linear conversion:

\[
v_A = \rho_A x_A.
\]

**Equation 6.2**

Here \(v_A\) is the actual input voltage at the MCU terminals, \(x_A\) is the numerical reading (integer) in the MCU, and \(\rho_A\) is the conversion factor. The subscript “A” stands for ADC.

Note that \(\rho_A\) is the voltage corresponding to a step difference of 1 in the ADC. The reference voltage and the resolution of the ADC affect the conversion. Given that the ADC used a 10-bit reading, and that the reference voltage is 5 volts, the theoretical resolution is,

\[
\rho_A^* = \frac{V_{ref}}{2^{10} - 1} = \frac{5}{1023} = 4.89 \text{ mV.}
\]

This can be used where approximate readings are required. However, it is simple enough to determine the conversion factor experimentally. Different constant voltages can be applied by using a potentiometer as a voltage divider. A small program for the microcontroller reported the ADC measures via RS232 communicating to a “terminal” program on the computer. With nine separate readings and using regression the experimentally determined resolution is,

\[
\rho_A = 4.775 \text{ mV.}
\]

**Equation 6.3**

(Refer to the Excel file “PhD\Experiment\Rig\Calibration Voltage and MCU 01”, sheet 2. Note the linear graph of results, and the very high correlation coefficient.)
regression showed extremely good agreement with equation 6.2, with a correlation coefficient of 0.999997.

“Oversampling” produces higher resolution and more accurate readings at the cost of processor time. If the voltage is oversampled $n$ times and summed to give $X_A$, then the average $x$ value is effectively $X_A$ divided by $n$, and equation 6.2 produces,

$$v_A = \rho_A x = \frac{X_A}{n}.$$  

Equation 6.4

Output using PWM (described above in section 6.2) can also be calibrated. The PWM rectangular wave is filtered by a simple low-pass RC filter as shown in figure 6.10. The resistor and capacitor values used for the filter were $R = 325 \ \Omega$ and $C = 1.55 \ \mu F$ (determined by a multimeter). Some care was used in choosing these values, and in deciding the PWM frequency in order to ensure that the pulse frequency was filtered without affecting the intended output. Since the intended output frequencies are quite low, in the low- to mid-audio range, this did not present a great challenge.

![Figure 6.10 RC Low-pass Filter](image)

The PWM depends linearly on the MCU output number,

$$v_p = \rho_p x_p.$$  

Equation 6.5

The subscript “$P$” stands for PWM. In this case, $x_p$ is the value used by the PWM in the MCU to produce the output voltage $v_p$. Given a 9-bit output with a 5 volt rectangular wave the theoretical conversion factor, $\rho_p$, should be on the order of,
\[ \rho_p^* = \frac{V_{\text{ref}}}{2^9 - 1} = \frac{5}{511} = 9.785 \text{ mV}. \]

Again, this factor can be determined experimentally. This author wrote a small program for the microcontroller to output a particular PWM value. The experimentally determined value of the resolution is,

\[ \rho_p = 9.6 \text{ mV}. \]

**Equation 6.6**

(Refer to the Excel file “PhD\Experiment\Rig\Calibration MCU to Voltage 01”, sheet 2.) The experiment showed very high linearity, with a correlation coefficient of 0.9999993. Solving for \( x_p \) in equation 6.5, gives,

\[ x_p = \frac{v_p}{\rho_p}. \]

This then is the numerical value required by the MCU.

For verification, a MCU program was written by the author which simply accepted voltage input via ADC and echoed exactly the same voltage for output via PWM. Let \( X_A \) represent the numerical value input from the ADC (with oversampling) and \( x_p \) the numerical value output to the PWM. Equation 6.4 and equation 6.5 can be combined to give,

\[ x_p = \frac{v_p}{\rho_p} = \frac{v_A}{\rho_p} = \frac{\rho_A}{\rho_p} X_A. \]

Figure 6.11 shows an experiment with output voltage, generated by the PWM, copying the input voltage, read by the ADC, verifying the conversions. The input was varied using a potentiometer. The graphs have been shifted vertically by 0.2 volts to make the separate graphs clear. The lower graph is the voltage from the rig and the upper graph is the echoed voltage from the MCU. (MCU code for this is available in the file, Experiment\Electronics\Atmel AVR\Test104 Match OutputToInput\MatchOutputToInput.c, or by pressing “L” in the program, Experiment\Electronics\Atmel AVR\Test111 Test Landing 04\TestLand04.c)
Figure 6.11 PWM Voltage Copies ADC

The time lag here is the step time of the MCU, and is quite small, but observable.

6.3.3. Numerical Calculation of Derivatives

Let voltage (corresponding to displacement) be represented as $v$. The successive derivatives, $\dot{v}$ and $\ddot{v}$, can be determined numerically from the measured values of $v$. It is important to stress that there are two places in which numerical methods are used: they can be employed both on-board, inside the MCU, and off-board, using computer manipulation of data collected from the digital oscilloscope. The on-board calculation is destined to be used in the MCU for control purposes, while the computer calculations are used for calibration, verification, and for determining the effective spring rate and damper characteristics.

The on-board numerical methods for determining derivatives are quite different from those used off-board. The off-board numerical methods employ a “sliding window” average. (The Java class Smoother in the ProcessScope package performs this function.) This method
cannot be used effectively for on-board calculations because of the latency it induces into the control process.

On-board signal processing in the MCU employs a moving average (as described in the appendix, in section 8.12). All code for all methods, both on-board and off-board, was developed and tested by the author.

For the sliding-window method in the off-board calculations, an average of a successive number of points is used rather than individual values. An example of the effect of numerical smoothing is shown in figure 6.12, where 13 data points are used in the sliding window. The graph shows the smoothed line running through the actual data points. The smoothed line can be used for the calculation of derivatives, and the derivative can be further smoothed and the second derivative calculated, as shown in figure 6.13 (the derivatives have been scaled to be clearly visible on the same graph).

![Figure 6.12 Sliding Window Smoothing](image)
Figure 6.13 Sliding Window used to Calculate Derivatives

Note that the accelerometer measures the acceleration of the top bar, not the acceleration of the relative movement. The on-board subroutines to perform the smoothing are shown immediately below. As discussed in section 2.3.1.2, Kalman filtering can be achieved by three simple equations given the simple dynamics of the rig itself. Velocity and acceleration are simple time derivatives for distance and velocity respectively. These can be used as the time-update step, applying equation 2.6. Similarly the system dynamics is applied to provide the time-update step for acceleration.

The state estimation steps as calculated on-board are shown in figure 8.47. The acceleration is processed in a few more steps in order to provide control over either acceleration or jerk, as discussed in later sections.

The estimations of acceleration of the stroke require two numerical differentiations (with smoothing as explained in detail above) of the displacement. A further differentiation would give the jerk of the stroke (the displacement between top and bottom arms). This is a superfluous measure, however, as it is the reduction of chassis (top arm) jerk that is required for the estimation of the smoothing performance. This will be estimated from accelerometer measures, which has the advantage that it is not delayed by the latency of the numerical smoothing, used to maintain good estimates of differentiation.
The various Kalman gain factors were determined by experimenting until an acceptable compromise was found between good smoothing (low $K$ values) and a low time lag (high $K$ values). This could be judged by eye from the oscilloscope output.

The estimates of smoothness are for experimental use only, and are not part of the suspension control algorithm; the control algorithms themselves use estimates only of distance, velocity and acceleration, and not jerk. Furthermore, it is important to emphasise that the jerk obtained from chassis movement is responsible for ride smoothness, while stroke jerk measures are irrelevant. Indeed, the wheel’s role is to absorb shocks and to move rapidly with the road while the chassis remains relatively calm and smooth. As a result the stroke jerk will be high, and probably would be corresponding difficult to estimate in any case, while the chassis jerk only requires one numerical differentiation from a direct measure, and is not used for any control purpose in any case, rather it is used only for experimental estimates of performance.

### 6.3.4. Velocity Calibration

This section describes how the velocity calculation on the MCU is calibrated against the independent calculation of velocity from off-board oscilloscope data. The calculation of raw velocity values on the MCU uses simple subtraction of successive distance measures, after smoothing. The frequency of the distance measures is kept constant. Let $f$ be the frequency of measurement, let $X_A$ be the distance measure (oversampled $n$ times), and let $\Delta X_A$ be the difference in these measures from one step to the next. Thus the rate of change in $X_A$ is approximated as,

$$\dot{X}_A \approx f\Delta X_A.$$  

Differentiating equation 6.4 and using the above equation gives,

$$\dot{v}_A = \frac{d}{dt} \rho_A \frac{X_A}{n} = \rho_A \frac{\dot{X}_A}{n} \approx \rho_A \frac{f\Delta X_A}{n}.$$  

In the experiments described immediately below, $f = 100$ Hz. For the purpose of verification the velocity can be echoed to the PC via RS232. For further verification, the value can be echoed to a PWM voltage output, using equation 6.5,
\[ x_p = \frac{v_p}{\rho_p} = \frac{\dot{v}_p}{\rho_p} = \frac{\rho_p \dot{X}_A}{\rho_p n} \approx \rho_p fAX_A. \]

In experiment, this value is centred in the output voltage range by adding 256 to the PWM value.

For the purpose of comparing on-board and off-board velocity measures, a constant velocity was simulated on the MCU using a simple program to produce a triangle wave with constant slope on the up and down portions of the wave, as seen in figure 6.15 (a). The code to set the desired output voltage is quite simple and is shown in figure 8.48. (The following lines of code in the C program, DamperControl01.h, produce the triangle wave output when the lines are uncommented. These lines need to be commented out again to restore the normal operation of this program.)

This code is placed just after the actual distance measures are produced and smoothed; the parameter “Vav” represents the smoothed distance measure. In figure 6.15 (a), this is shown echoed to the output. The triangle wave slope can be varied programmatically. (Refer to AVR Studio project in, PhD\Experiment\Electronics\Atmel AVR\Test105 Test Velocity\TestVelocity01.c.) The screenshot in figure 6.14 shows the velocity step size being changed from the computer via RS232 communications using a “terminal” program. The MCU also periodically reports the results of its calculations. In the example, the calculated velocity (during a downward slope) has the same magnitude as the input value, 60, verifying the calculation.
Figure 6.14 Terminal Communication

The graphs in figure 6.15 part (a) and (b) show the distance (voltage) and velocity outputs. Graph (a) shows the output of the triangle wave, and graph (b) shows the output of the slope calculation converted to a PWM output from the MCU. In these graphs the difference in the velocity $\Delta X_A$ has been set to 15. Applying the conversion factor of equation 6.3, and with a frequency of 100 Hz, the rate of change of voltage should be,

$$\dot{V}_A = \frac{\rho_A}{n} \dot{X}_A \approx \frac{\rho_A / \Delta X_A}{n} = \frac{4.775 \times 10^{-3} \times 100 \times 15}{4} = 1.791 \text{ V.}$$

As can be seen in figure (b), the MCU calculation is close to this value.
Figure 6.15  Constant Rate-of-Change of Voltage
Figure (a) shows the triangle wave output from the MCU and the off-board velocity value calculated in the Java program. Also, two points are taken on the rising line in figure (a): (0.969, 0.505) and (3.187, 4.433). The slope between these points is 1.771 V/s. The heights of the rate-of-change of voltage calculated off-board in figure (a) are 1.789 V/s and -1.789 V/s. The values calculated and output by the MCU in figure (b) are 1.778 V/s and -1.774 V/s. These values all agree to two decimal places.

Using actual input data instead of simulated triangle waves, the on-board MCU velocity output can be directly compared with velocity calculated off-board from oscilloscope data. An example of such a comparison is shown in figure 6.16 (the raw data can be found in the file testVel_Sine01.txt in the folder PhD\Eclipse RSpace Constraint\ScopeDataFiles02). The MCU velocity was scaled down by a factor of 5 to fit within 5 V, and then scaled up by a factor of 5 in the graphing software. The two methods substantially agree. Note the time lag in the on-board calculation because at this point the time update of the Kalman step did not use acceleration for the time update step, since this is yet to be calibrated.

![Figure 6.16 Comparison of Sliding Window with Moving Average](image)

Figure 6.16 Comparison of Sliding Window with Moving Average
6.3.5. Calibration of the Accelerometer

Acceleration is measured independently by a MEMS accelerometer (see section 2.10.2) attached to the experimental rig as shown in figure 6.17. The accelerometer used was an ADXL210 contained on an evaluation board purchased online from Dimension Engineering (Robotshop, 2007). Such is the pace of development with MEMS accelerometers that this particular component became discontinued while writing, but similar and better components are continually being developed. (The datasheet for the component can be found in the folder PhD\Experiment\Accelerometer.) Today accelerometers generally supply digital output using the SPI or I2C communication protocols, which would have been preferable. Unfortunately at the time the rig was built the purchased evaluation board with analog output was widely used.
The calibration of the accelerometer readings was performed by comparison with the distance measures taken in volts. This results in an “acceleration” in units of Vs^-2. The following graph, figure 6.18, shows both the distance and acceleration readings taken from the rig while the top of the rig was being moved up and down by hand. The bottom of the rig is stationary. As expected with a near sine wave movement, acceleration is phase shifted by π radians.

Figure 6.18 Distance and Acceleration, both from Rig

The same Java program used above to compute velocity can be used to produce a further derivative and so provide an estimate of acceleration which can be compared to data supplied by the accelerometer. The graph in figure 6.19 is an example of acceleration calculated in the Java program (based on distance data input to the scope) compared with scope input from the accelerometer. (The data file is testAcc_Wave_04.txt. All such data files are stored in the folder PhD\Eclipse RSpace Constraint\ScopeDataFiles02.) Both the computed acceleration and the accelerometer data have been scaled in order to be clearly visible in the same graph just as the original distance voltage measures were, as well as to make a comparison to verify the accelerometer scaling factor. The calculated acceleration measure has been multiplied by 0.02.
Figure 6.19 Accelerometer compared with Calculated Acceleration

Let $\beta$ be the parameter for conversion between the accelerometer voltage output, $v_{\text{acc}}$, and the “acceleration” (second-order derivative of displacement voltage), $\ddot{v}_A$, so that,

$$\ddot{v}_A = \beta(v_{\text{acc}} - v_0).$$

Equation 6.7

Here $v_0$ is the accelerometer voltage corresponding to zero acceleration. The value of $\beta$ can be estimated by taking the average of $\ddot{v}_A / (v_{\text{acc}} - v_0)$ for a large number of points. With the data as shown in figure 6.19 the average value of calculated acceleration divided by measured acceleration was computed. The value used here is,

$$\beta = 530.$$  

Equation 6.8

This is then the estimate of the acceleration multiplication factor needed to make the accelerometer acceleration match the second-order rate-of-change of voltage. (This was calculated in the Java program, ProcessScopeDemo.)
Given that the calculated acceleration was multiplied by 0.02 (set using AccMultiplier in the Graph Parameters group in ProcessScopeDemo), for the purposes of graphical display the accelerometer value is multiplied by,

\[ 530 \times 0.02 = 10.6. \]

(This is set as accelerometerMult in the Experimental Parameters group.) This multiplier value was used in the graph in figure 6.19, and verifies that the conversion factor is correct.

To verify the on-board calculation of acceleration, the internally generated value is compared with off-board calculations of acceleration. The accelerometer value is oversampled \( n_{\text{acc}} \) times (here \( n_{\text{acc}} = 2 \)) and the summed ADC reading is represented below as \( X_{\text{acc}} \). The value of \( v_{\text{acc}} \) is then calculated in a manner similar to equation 6.4,

\[ v_{\text{acc}} = \frac{\rho_A}{n_{\text{acc}}} X_{\text{acc}}. \]

The same proportionality constant, \( \rho_A \), on the voltage input as in the previous section has been used. In the MCU the measured accelerometer voltage is multiplied by the conversion factor, \( \beta \), to provide an estimate of true acceleration. Combining the previous equation with equation 6.7 and supposing that the zero acceleration voltage corresponds to an ADC value of \( X_0 \) divided by \( n_{\text{acc}} \), produces,

\[ \ddot{v} = \beta \left( \frac{\rho_A}{n_{\text{acc}}} X_{\text{acc}} - v_0 \right) = \beta \frac{\rho_A}{n_{\text{acc}}} (X_{\text{acc}} - X_0). \]

To maintain the output within the voltage output limits of the MCU, the acceleration needs to be scaled down and centred. A factor of 0.02 was used for scaling. For the PWM then, the output value is,

\[ x_p = \frac{0.02}{\rho_p} \ddot{v} + 256 = \frac{0.02}{\rho_p} \beta \frac{\rho_A}{n_{\text{acc}}} (X_{\text{acc}} - X_0) + 256. \]

Note that the zero value of accelerometer output needs to be estimated. This may drift with changing temperatures and other conditions. If the zero value is itself estimated using a moving average with a very long time constant, then the acceleration measure becomes self-zeroing. The voltage then becomes self-zeroing. The first measured voltage is used as the initial zero estimate. Thus the rig needs to be stationary at startup, and the measurement
circuits need to be fully operational by the time the microprocessor starts measuring. The equilibrium (zero position) is slowly adjusted using the moving average with an extremely long time constant (corresponding to a low value for $\alpha$ in the moving average calculation given in section 8.12).

As a simple demonstration of the exponential decay to perform the self-zeroing, an inaccurate initial estimate, $X_0$, can be coded. The exponentially-weighted moving average uses a long time constant. In figure 6.20 the time constant is about 0.5 sec ($\alpha = 0.02$), and the exponential decay down to the true zero value is obvious (for the test, the rig remains at rest throughout). In the actual code used in experiments, the initial value is determined from the initial reading and, as for the displacement readings, the rig must be initially stationary with the accelerometer circuits fully operational by the time the first microprocessor reading is taken. The time constant for self-zeroing is quite long, of the order of ten seconds ($\alpha = 0.0005$). With these values there are no noticeable artefacts on the control produced by the self-zeroing.

![Figure 6.20 Self Zeroing Demonstration](image)

In the following graph, the voltage acceleration has been computed by the MCU (and scaled by the same factor as the calculated acceleration, 0.02). The two estimates are compared on the one graph at the same scale. The two graphs substantially agree, and this further verifies the method used above.
The smoothness of the jerk estimation is affected by high frequency noise. This can be reduced by two processes, first by the low-pass filter used to limit noise on the input. The experimental filter, as discussed in section 6.3.2 has a low-pass filter where \( R = 325 \, \Omega \) and \( C = 1.55 \, \mu F \) which gives a cut-off frequency of 320 Hz, which is high enough to allow useful data through, but which will smooth out noise at higher frequencies. Numerical smoothing on the acceleration data using Kalman filtering has a further similar effect as an analogue filter except that the filtering is performed at discreet points. This suggests a separate filter could perhaps be used before accelerometer inputs used to estimate jerk. It should be noted too that the accelerometer itself has a low-pass filter on its output to perform integration on its digital-to-analogue converter. In order to estimate the jerk the acceleration was smoothed numerically up to the point that the acceleration did not have undue latency but there was little noticeable noise on the acceleration graph. This was done by running the rig with fast top-bar oscillations and judging the best form of filtering. Jerk estimates were then simply taken by subtracting successive top-bar acceleration measures. It should be remembered too that jerk is not estimated for on-board calculations, it is used for experimental measures of performance.

Figure 6.21 MCU Output compared to Calculated Acceleration
6.3.6. Calibration of Damper and Spring Forces

The datasheet for the damper supplied by Lord Corporation shows a graph of “Typical Force vs. Velocity” (see figure 6.22). (The graph is taken from the datasheet found in the supplied material: PhD\Experiment\Lord RD 1005 3 Damper.pdf.) The offset force at zero stroke velocity “is due to gas precharge required for temperature compensation and to prevent cavitation”. The spring rate of the gas precharge for this experiment is small relative to the spring attached to the rig. Furthermore it is assumed that the precharge adds linearly to the springing of the rig, and the experiments below will determine the combined effective spring rate. In the graph shown in see figure 6.22 hysteresis effects are assumed negligible, and this will also be assumed here. The voltages are missing from this graph so the actual characteristics must be determined experimentally.

![Figure 6.22 Damper Characteristics (from Supplier’s Datasheet)](image)

At this point the damper force is estimated using the data output from the rig. It has been assumed that the voltage measure versus displacement is linear, and so the second rate of change of the voltage is proportional to force. It is also assumed that the damper’s effect on
the second rate-of-change of voltage is approximately that given by the suppliers, as depicted in figure 6.22. Further assumptions are:

- mass is constant (so force is proportional to acceleration),
- top arm displacement is small (so rotational movement can be ignored),
- the spring is linear (including the springing of the damper’s “gas precharge”), and
- damper force depends on stroke velocity and applied voltage.

Under these assumptions, the following model can be used,

$$\ddot{v} + A_D(\dot{v}, V) + k_e v = 0.$$  

**Equation 6.9**

Here \( v \) represents the voltage measure corresponding to the displacement as above, \( k_e \) represents the equivalent “spring rate”, and \( A_D \) is the damper’s characteristic function, dependent on velocity and voltage applied to the “Wonder Box”, \( V \). It is assumed that the function \( A_D \) has approximately the same shape as the force profile shown in figure 6.22, but this will be examined experimentally below.

To determine the damper’s characteristics the effective spring rate, \( k_e \), must first be estimated (for brevity this will here be referred to as just the “spring rate”). This is determined from the relationship between distance and acceleration with the rig under oscillation with no damping input (and with very small damping force). Unfortunately, the rig cannot be kept oscillating by hand since any external force will make equation 6.9 invalid, and corrupt the data.

The simplest method is to displace the rig away from equilibrium, let it go, and record the motion of the rig with no external force, recording the transient response of the rig. These will be referred to as “transient” or “drop” experiments. A surprising amount of information can be gleaned from such experiments. As expected the responses vary greatly with different voltages applied to the damper.

The transient experiments needed to be performed carefully. The release must be quick and clean; it is only when the rig has been released that data can be used. A belt tied around the
central shaft through the gym weights allowed a smoother, quicker release than simply holding the weights by hand on either side.

![Figure 6.23 Oscilloscope Data for Transient](image)

Figure 6.23 shows a graph of data collected for a transient. (The data for this example can be found in the file DistAcc_00_00.txt.) While the drop occurs slightly before the zero time mark on the graph, it takes a small amount of time to fully release the rig. Therefore the first part of the data needs to be removed (and some header lines inserted for the code used here). The data can be zeroed by checking that the zero values match the equilibrium values, as shown in figure 6.24. Once equilibrium has been reached, after about one second, the data adds no useful information for regression analysis and these points can also be deleted.
To attempt to determine the effective spring rate, linear regression was first used to fit data points to the following expression,
\[ \ddot{v} = -k_e v. \]

Equation 6.10

This is used only in cases where no voltage is applied to the damper. Even though the damper term has been removed, the damper remains attached to the rig. As discussed above, there is some small amount of springing as a result of the pressurized gas in the damper, and keeping the damper attached is a convenient way to remove this artefact of the damper from the calculation; it simply includes any possible damper springing in the spring rate (this springing is also assumed to be linear, and is in any case much lower than the rate of the coiled spring). The zero distance is measured as the distance at equilibrium, so the small damper force offset at zero, visible in the datasheet, is also factored into the spring equilibrium position.

Acceleration data can be collected from three sources,
- directly from the accelerometer,
- from on-board calculations of derivatives in the MCU, and
- from numerical processing off-board of oscilloscope data.

Note that the accelerometer data measures the top beams acceleration, while the other two measure the acceleration of the top beam relative to the base. In the calibration examples, the base is at rest and all values measure the same acceleration.

The accelerometer voltage is proportional to acceleration, but it is taken from a new device, the accelerometer, and it therefore has a different scale from the acceleration calculated from the potentiometer. Suppose in each of the three measurement sources, the acceleration measure is represented as \( a \), and the coefficient of conversion into potentiometer voltage acceleration is represented as \( \gamma \). Therefore,
\[ \ddot{v} = \gamma a. \]

Equation 6.11
In the case where $a$ represents the direct accelerometer voltage, the scaling factor is $\beta$, as in equation 6.7 above. In the case of the on-board acceleration calculation, the corresponding MCU output has been scaled down by a factor of 0.02 in order to fit the 5 volt output range of the MCU. Thus it needs to be multiplied by 50 to give a true reading of acceleration. Finally, the off-board calculated of acceleration is unscaled (when used in the regression calculation, although it has been scaled down by 0.02 when shown graphically, for visual purposes). This produces the following scaling factors for the three sources,

- direct from the accelerometer: $\gamma = \beta = 530$,
- from the MCU: $\gamma = \frac{1}{0.02} = 50$ and,
- from numerical processing unscaled, i.e., $\gamma = 1$.

The method of least squares can be applied to the data from a transient experiment to determine a regression coefficient for the factor acceleration against distance. If $a_i$ is a set of acceleration measures gathered from experiment and $v_i$ is a sequence of voltage measures representing distance, then the regression factor, $K$, can be experimentally determined for the relationship,

$$a_i \approx -Kv_i.$$  

**Equation 6.12**

Linear regression coefficients are determined using the least squares method (Strang, 1980, p138). In the simple case of the linear equation 6.12, regression produces,

$$K = -\frac{\sum a_i v_i}{\sum v_i^2}.$$  

(Note that there is no constant in equation 6.12.) The data shown in table 9 applies regression to acceleration data collected from a number of sources. The names of the source files for the data are shown, and the regression values were calculated in this author’s Java program (ProcessScopeDemo).

Combining equation 6.10 and equation 6.11 gives the following equation for the effective spring rate:
\[ k_e = \gamma K. \]

**Equation 6.13**

In table 9 the effective spring rate measures are in agreement, albeit with a slightly higher value from the off-board numerical processing.

<table>
<thead>
<tr>
<th>Data Source File</th>
<th>( K, k_e ) estimate ((k_e = \gamma K))</th>
<th>( \gamma = \beta \approx 530 )</th>
<th>( \gamma = 50 )</th>
<th>( \gamma = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( K )</td>
<td>( k_e )</td>
<td>( K )</td>
<td>( k_e )</td>
</tr>
<tr>
<td>DistAcc_00_01.txt</td>
<td>0.32949</td>
<td>174.630</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DistAcc_00_02.txt</td>
<td>0.33227</td>
<td>176.103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DistAcc_00_03.txt</td>
<td>0.31783</td>
<td>168.450</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DistAcc_00_04.txt</td>
<td>0.31172</td>
<td>165.212</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DistMCU_00_01.txt</td>
<td></td>
<td></td>
<td>3.5717</td>
<td>178.585</td>
</tr>
<tr>
<td>DistMCU_00_01.txt</td>
<td></td>
<td></td>
<td>3.5245</td>
<td>176.225</td>
</tr>
<tr>
<td>DistMCU_00_01.txt</td>
<td></td>
<td></td>
<td>3.4175</td>
<td>170.875</td>
</tr>
<tr>
<td>DistMCU_00_01.txt</td>
<td></td>
<td></td>
<td>3.4353</td>
<td>171.765</td>
</tr>
</tbody>
</table>

**Table 9 Regression Estimates of Spring Rate**

If acceleration, \( a \), is graphed against velocity, as shown in figure 6.25 (based on data in the file DistAcc_00_01.txt) a decreasing spiral is found. This results from the fact that spring potential energy is converted into kinetic energy and back again, and is eventually dissipated because of damping. If the calculated acceleration due to the spring is removed from this plot, calculated as \( k_e v \), then a plot of the damper acceleration remains, which is proportional to damper force (to be accurate, this is the damping force minus the spring force due to the precharge that is included in the spring rate determined earlier). Refer to figure 6.26.
Figure 6.25 Spiral as Energy Decreases

Solving equation 6.9 for the damper function produces,

\[ A_D(\dot{v}, V) = - (\dot{v} + k_v) = -\gamma (a + K_v) \]

A plot of \(- (a + K_v)\) against velocity \(\dot{v}\) shows a graph proportional to the damping function, \(A_D\), for a given input voltage, \(V\). In the case where no voltage is applied to the damper this produces a curve for \(A_D(\dot{v}, 0)/\gamma\). Using estimates of the spring rate, \(K\), the result is a curve that closely corresponds to the graph of zero damper current in the datasheet in figure 6.22 (refer to figure 6.26).

Figure 6.26 Approximate Damper Force

To estimate damper characteristics, a function needs to found which approximates the shape of the damper’s characteristics as given by the supplier (shown in figure 6.22). Such a function is the following,
\[ f(x) = \begin{cases} 
  s_1 x + s_2 x & \text{if } -1 \leq s_1 x \leq 1, \\
  1 + s_2 x & \text{if } 1 \leq s_1 x \\
  -1 + s_2 x & \text{if } s_1 x \leq -1.
\end{cases} \]

**Equation 6.14**

This has the shape of the dotted line in the graph in figure 6.27. The values of \( s_1 \) and \( s_2 \) are parameters of this function.

The coefficients for the following linear function have been found using least squares (Strang, 1980, p138) to fit data to a particular function, \( f \). (The method of least squares is here being used to find linear coefficients, one of which is multiplied by a linear function, as explained in Strang.) Next the least squares fit to the following function is determined,

\[ \alpha_i \approx -C f(\dot{v}_i) - K v_i, \]

**Equation 6.15**

where \( C \) and \( K \) are linear coefficients, \( \alpha \) is the measured accelerometer data, and \( f \) is the function of equation 6.14 (with given parameters, \( s_1 \) and \( s_2 \)). In matrix form this can be represented as,

\[
Ax = \begin{bmatrix} f(\dot{v}_1) & v_1 \\ \vdots & \vdots \\ f(\dot{v}_n) & v_n \end{bmatrix} \begin{bmatrix} -C \\ -K \end{bmatrix} \approx \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} = b.
\]

To simplify expressions \( \dot{f} \) is used in place of \( f(\dot{v}_i) \). Applying matrix regression techniques produces (Strang, 1980),

\[
\begin{bmatrix} -C \\ -K \end{bmatrix} = x = (A^T A)^i A^T b = \frac{1}{\sum f_i^2 \sum v_i^2 - (\sum v_i f_i)^2} \begin{bmatrix} \sum v_i^2 & -\sum v_i f_i & \sum f_i a_i \\ -\sum v_i f_i & \sum v_i^2 & \sum v_i a_i \\ \sum f_i a_i & \sum v_i a_i & \sum f_i^2 \sum v_i^2 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}.
\]
Figure 6.27 shows an instance (taking data from the file DistAcc_00_03.txt) using a particular curve, \( f \), shown as a dotted line, to approximate the damper function. The parameters of the \( f \) function are,

\[
s_1 = 1.0264 \quad \text{and} \quad s_2 = 0.0526.
\]

Using these values, the regression coefficients are,

\[
C = 0.03606 \quad \text{and} \quad K = 0.3293.
\]

Note that the spring rate has been used to remove the spring component from the graph of acceleration data leaving the damper component. This has been done because rearranging equation 6.15 produces,

\[
Cf(\dot{v}_i) \approx -a_i - Kv_i.
\]

(The graph below can be obtained by running ProcessScope with the file data for file DistAcc_00_03.txt and pressing the button Damper Force. The data file is contained in the folder, PhD\Eclipse RSpace Constraint\ScopeDataFiles02. The values of \( s_1 \) and \( s_2 \) are represented in the Java program as slope1 and slope2 respectively. Running this code produces the regression coefficients of \( C \) and \( K \) above in the “console window” in Eclipse, or from the “DOS” window if run outside the Eclipse IDE.)

![Graph](image)

**Figure 6.27  Least Squares Fit**

The same process has been repeated and the data has been collected in a spreadsheet (the Excel spreadsheet found in the file SpringWithDamperLeastSquares01 in the folder...
PhD\Experiment\Rig\ in the first sheet, named “SpringWithDamperLeastSquares01”). The first four rows of results in this table produce data in exactly the same way, with zero control voltage applied to the damper.

The same process can be applied but with a constant voltage applied to the damper control. Thus a set of curves can describe the damper response at various values of control voltage. From the supplier’s datasheet it should be expected that roughly the same shape of curve will result, but with larger acceleration values corresponding to larger values of applied voltage.

Thus the same experiments, producing transient responses, are performed except that a different constant voltage is applied to the damper during the experiment (actually applied to the control device supplied by Lord, which converts the signal level control voltage to current, which is output to the damper’s magnetic coils, as explained above). Again, off-board calculations determine distance and velocity, and acceleration is taken from the accelerometer.

Because the extraction of data from transients is important for determining the damper’s response, it is briefly described here in slightly more detail using a particular example. Suppose that the MCU outputs voltage corresponding to a PWM value of 100 (giving 0.815 volts at the low-pass filter, under the load of the damper). The weights on the rig are then physically lifted and dropped, giving the oscilloscope data as shown below in figure 6.28. (The red line represents displacement taken from the potentiometer, and the blue is accelerometer data taken from the accelerometer.)
The scope data is then saved to an ASCII “text” file, in this case to the file DistRawAcc_100_01.txt. The first two lines of the file are modified using an editor: the first line contains comments pertinent to the experiment and the second line contains information for zeroing the distance and acceleration input. This file can then be processed by the program ProcessScopeDemo.java, in the ProcessScope package. The input can be smoothed and graphed, as below in figure 6.29.
Figure 6.29 Graph of Raw Data

The zero values, the dashed lines in the figure above, are set so that they correspond with the equilibrium location in the graph (which has been done in the file DistRawAcc_100_01.txt). An example of the graph accelerometer data against velocity, minus the spring acceleration, is shown in figure 6.30. Removing an estimate of the acceleration due to the spring makes the graph clearer.
This graph has the expected shape for damper acceleration except for the tail at the top. It takes a small amount of time to drop the weight and the data before the zero time is corrupted by the fact that some external force is still being applied. (From the graph tail it can be seen that this is a larger force at lower velocity, as would be expected from such an external force.) Therefore these points are removed when calculating the least squares values.

Also, once equilibrium is reached, noise in the data collection is greater than the zero data and this affects the least squares approximation, even though it is clear that the system has no more useful data to contribute because it has reached equilibrium. Thus data is also removed from the end of the input, after the system has reached equilibrium. In the case above, data is dropped after 0.8 seconds (the system reaches equilibrium faster when greater voltage is applied to the damper control). The resulting damper acceleration graph minus these data points is shown below in figure 6.31. The $f$ function parameters, parameters, $s_1$ and $s_2$ can be adjusted in the off-board Java program, ProcessScopeDemo, until a good fit is observed, and the regression parameters, C and K, are found by regression as described earlier.
Graphs collected in the same manner for a number of different values of PWM output are shown in figure 6.32. The curves are all on the same scale and show clearly that the force applied by the damper increases with larger applied voltage. Experiments with the higher values of control voltage were more difficult because they reached equilibrium quite quickly and had less data points. The lines on all these graphs were fitted by eye by the author.
Figure 6.32 Acceleration vs. Velocity for Different Damper Control Voltages

Some 40 such transients have been recorded and processed in this way (the files can be found in the directory PhD\Eclipse RSpace Constraint\ScopeDataFiles02, in the files named DistAcc_X_X.txt). The various damper curves have been approximated by a function, $f$, described by equation 6.14. The parameters were simply varied by hand using sliders, varying the values of $s_1$ and $s_2$ to produce a good fit with the data. The values of $C$ and $K$, as in equation 6.15, were then determined by least squares. (Data is stored in the Excel spreadsheet, SpringWithDamperLeastSquares01, in the folder PhD\Experiment\Rig.) The raw data has been collected together below in table 10.
### Table 10 Raw Data for Damper Acceleration Calculations

<table>
<thead>
<tr>
<th>Filename</th>
<th>PWM number</th>
<th>PWM Voltage volts</th>
<th>s1</th>
<th>s2</th>
<th>C</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>DistAcc_00_01</td>
<td>0</td>
<td>0</td>
<td>1.0264</td>
<td>0.0526</td>
<td>0.03375</td>
<td>0.3394</td>
</tr>
<tr>
<td>DistAcc_00_02</td>
<td>0</td>
<td>0</td>
<td>1.0264</td>
<td>0.0526</td>
<td>0.03939</td>
<td>0.3334</td>
</tr>
<tr>
<td>DistAcc_00_03</td>
<td>0</td>
<td>0</td>
<td>1.0264</td>
<td>0.0526</td>
<td>0.03606</td>
<td>0.3293</td>
</tr>
<tr>
<td>DistAcc_00_04</td>
<td>0</td>
<td>0</td>
<td>1.0264</td>
<td>0.0526</td>
<td>0.03467</td>
<td>0.3237</td>
</tr>
<tr>
<td>DistAcc_010_01</td>
<td>10</td>
<td>0.0551</td>
<td>0.7106</td>
<td>0.0474</td>
<td>0.03604</td>
<td>0.3234</td>
</tr>
<tr>
<td>DistAcc_020_01</td>
<td>20</td>
<td>0.1491</td>
<td>0.6578</td>
<td>0.0578</td>
<td>0.03536</td>
<td>0.3415</td>
</tr>
<tr>
<td>DistAcc_025_01</td>
<td>25</td>
<td>0.1972</td>
<td>0.6578</td>
<td>0.0578</td>
<td>0.03349</td>
<td>0.3336</td>
</tr>
<tr>
<td>DistAcc_030_01</td>
<td>30</td>
<td>0.2434</td>
<td>0.6578</td>
<td>0.0736</td>
<td>0.03221</td>
<td>0.3335</td>
</tr>
<tr>
<td>DistAcc_040_01</td>
<td>40</td>
<td>0.3378</td>
<td>0.6578</td>
<td>0.0736</td>
<td>0.03331</td>
<td>0.3345</td>
</tr>
<tr>
<td>DistAcc_050_01</td>
<td>50</td>
<td>0.43</td>
<td>0.7368</td>
<td>0.1422</td>
<td>0.02472</td>
<td>0.3393</td>
</tr>
<tr>
<td>DistAcc_060_01</td>
<td>60</td>
<td>0.5117</td>
<td>0.7368</td>
<td>0.1422</td>
<td>0.02697</td>
<td>0.3283</td>
</tr>
<tr>
<td>DistAcc_070_01</td>
<td>70</td>
<td>0.58</td>
<td>0.7368</td>
<td>0.1422</td>
<td>0.02618</td>
<td>0.3251</td>
</tr>
<tr>
<td>DistAcc_075_01</td>
<td>75</td>
<td>0.6317</td>
<td>0.7368</td>
<td>0.1422</td>
<td>0.026</td>
<td>0.3282</td>
</tr>
<tr>
<td>DistAcc_080_01</td>
<td>80</td>
<td>0.6661</td>
<td>0.7368</td>
<td>0.1422</td>
<td>0.02837</td>
<td>0.3234</td>
</tr>
<tr>
<td>DistAcc_090_01</td>
<td>90</td>
<td>0.7412</td>
<td>0.7368</td>
<td>0.1422</td>
<td>0.03118</td>
<td>0.329</td>
</tr>
<tr>
<td>DistAcc_100_01</td>
<td>100</td>
<td>0.8174</td>
<td>0.7368</td>
<td>0.1422</td>
<td>0.03075</td>
<td>0.3237</td>
</tr>
<tr>
<td>DistAcc_100_02</td>
<td>100</td>
<td>0.8154</td>
<td>0.7368</td>
<td>0.1422</td>
<td>0.03401</td>
<td>0.2799</td>
</tr>
<tr>
<td>DistAcc_110_01</td>
<td>110</td>
<td>0.8882</td>
<td>0.7106</td>
<td>0.1264</td>
<td>0.03936</td>
<td>0.3261</td>
</tr>
<tr>
<td>DistAcc_120_01</td>
<td>120</td>
<td>0.9613</td>
<td>0.7106</td>
<td>0.1264</td>
<td>0.04565</td>
<td>0.325</td>
</tr>
<tr>
<td>DistAcc_125_01</td>
<td>125</td>
<td>1.0077</td>
<td>0.7106</td>
<td>0.1264</td>
<td>0.02969</td>
<td>0.3219</td>
</tr>
<tr>
<td>DistAcc_125_02</td>
<td>125</td>
<td>0.9973</td>
<td>0.7106</td>
<td>0.1264</td>
<td>0.0422</td>
<td>0.3094</td>
</tr>
<tr>
<td>DistAcc_130_01</td>
<td>130</td>
<td>1.0331</td>
<td>0.7106</td>
<td>0.1264</td>
<td>0.04377</td>
<td>0.3324</td>
</tr>
<tr>
<td>DistAcc_140_01</td>
<td>140</td>
<td>1.1048</td>
<td>0.7106</td>
<td>0.1264</td>
<td>0.05521</td>
<td>0.3279</td>
</tr>
<tr>
<td>DistAcc_150_01</td>
<td>150</td>
<td>1.1789</td>
<td>0.7106</td>
<td>0.1264</td>
<td>0.05601</td>
<td>0.3327</td>
</tr>
<tr>
<td>DistAcc_150_02</td>
<td>150</td>
<td>1.1771</td>
<td>0.7106</td>
<td>0.1264</td>
<td>0.06585</td>
<td>0.3457</td>
</tr>
<tr>
<td>DistAcc_160_01</td>
<td>160</td>
<td>1.2512</td>
<td>0.6052</td>
<td>0.1264</td>
<td>0.07302</td>
<td>0.3171</td>
</tr>
<tr>
<td>DistAcc_170_01</td>
<td>170</td>
<td>1.3308</td>
<td>0.6052</td>
<td>0.1264</td>
<td>0.07513</td>
<td>0.2397</td>
</tr>
<tr>
<td>DistAcc_175_01</td>
<td>175</td>
<td>1.3654</td>
<td>0.6052</td>
<td>0.1264</td>
<td>0.08486</td>
<td>0.3809</td>
</tr>
<tr>
<td>DistAcc_180_01</td>
<td>180</td>
<td>1.4002</td>
<td>0.6052</td>
<td>0.1264</td>
<td>0.08708</td>
<td>0.3791</td>
</tr>
<tr>
<td>DistAcc_190_01</td>
<td>190</td>
<td>1.4723</td>
<td>0.6316</td>
<td>0.0948</td>
<td>0.10386</td>
<td>0.3826</td>
</tr>
<tr>
<td>DistAcc_200_01</td>
<td>200</td>
<td>1.5393</td>
<td>0.6316</td>
<td>0.1526</td>
<td>0.08804</td>
<td>0.2933</td>
</tr>
<tr>
<td>DistAcc_210_01</td>
<td>210</td>
<td>1.6083</td>
<td>0.6316</td>
<td>0.1526</td>
<td>0.09828</td>
<td>0.3455</td>
</tr>
<tr>
<td>DistAcc_220_01</td>
<td>220</td>
<td>1.6806</td>
<td>0.6316</td>
<td>0.1158</td>
<td>0.11733</td>
<td>0.3704</td>
</tr>
<tr>
<td>DistAcc_230_01</td>
<td>230</td>
<td>1.7520</td>
<td>0.6316</td>
<td>0.1158</td>
<td>0.11943</td>
<td>0.3293</td>
</tr>
<tr>
<td>DistAcc_240_01</td>
<td>240</td>
<td>1.8240</td>
<td>0.5264</td>
<td>0.2368</td>
<td>0.06814</td>
<td>0.2683</td>
</tr>
<tr>
<td>DistAcc_240_02</td>
<td>240</td>
<td>1.822</td>
<td>0.7368</td>
<td>0.1948</td>
<td>0.08613</td>
<td>0.3184</td>
</tr>
<tr>
<td>DistAcc_250_01</td>
<td>250</td>
<td>1.888</td>
<td>0.7106</td>
<td>0.2158</td>
<td>0.07771</td>
<td>0.3183</td>
</tr>
<tr>
<td>DistAcc_260_01</td>
<td>260</td>
<td>1.97</td>
<td>0.8422</td>
<td>0.2474</td>
<td>0.07245</td>
<td>0.2878</td>
</tr>
<tr>
<td>DistAcc_270_01</td>
<td>270</td>
<td>2.041</td>
<td>0.8422</td>
<td>0.2474</td>
<td>0.07553</td>
<td>0.3045</td>
</tr>
<tr>
<td>DistAcc_275_01</td>
<td>275</td>
<td>2.077</td>
<td>0.8422</td>
<td>0.221</td>
<td>0.08137</td>
<td>0.3502</td>
</tr>
</tbody>
</table>

The “spring rate” in these experiments is consistent with a value of $K = 0.32723$. Also, the relationship between voltage applied to the damper and the PWM output is linear, as shown in the graph in figure 6.33. Thus the following formula is derived,

$$V \approx 0.00754n,$$

where $n$ is the PWM value and $V$ is the damper voltage. (These values have been calculated in the Excel spreadsheet.)
6.3.7. Estimate of Damper Inverse Function

Using the data gleaned above, the damper acceleration profile, $A_d(\dot{v},V)$, can be determined experimentally. For the purpose of control, it is desirable to invert this function and estimate the voltage that needs to be applied for a given acceleration when the stroke velocity is,

$$V = D_j(a, \dot{v}).$$

A few ideas were given some preliminary investigation, but it was decided to eventually use a purely linear approximation. This section shows some of the details of how the inverse function was derived. The most important part of this section, in the broader context, is the inverse function as shown in figure 8.49.

In the estimate given in table 10 the slope of the initial portion of the curve is $C s_1$, where $C$ is determined by least squares. Suppose a constant slope is assumed for the initial part of the curve near zero that has a reasonable fit to all graphs, using $c_x$ as in the function $f_z$ in equation 6.16.

$$f_z(\dot{v}) = \begin{cases} \min \left[ c_x \dot{v}, c_f + b_f \dot{v} \right] & \text{if } x > 0, \\ \max \left[ c_x \dot{v}, -c_f + b_f \dot{v} \right] & \text{if } x \leq 0. \end{cases}$$

Equation 6.16
The average value of $C_{s_1}$ is approximately 0.039, but it is larger for the larger values where it is a more significant part of the curve. After some experimentation it was thought that 0.05 is a better fit to the data.

In equation 6.16 the value of $c_e$ is now constant but the values of $c_f$ and $b_f$ vary for different values of PWM voltage. Values consistent with the data in table 10 are sought where approximately $c_f \approx C$ and, $b_f \approx C_{s_2}$. From the data in table 10 acceleration values given by,

$$C + C_{s_2} \dot{v},$$

This can be plotted against PWM values, $n$, holding $\dot{v}$ constant, which allows the possibility of finding a simple linear regression line for each of these plots. An example of such a plot is given below in figure 6.34 for $\dot{v} = 5$ volts/s. The regression line is included in this graph.

![Figure 6.34 Example of Acceleration vs. PWM](image)

The slopes and intersections of the regression lines are found to depend linearly on the velocity giving,

$$a = c_R + b_R n$$

where,

$$c_R = 0.022397 - 0.000169615 \dot{v}$$

and,

$$b_R = 0.00026872 + 6.22226 \times 10^{-5} \dot{v}.$$
acceleration is represented as $a$. These coefficients are easily calculated for a given value of velocity. More importantly, the function is easily inverted, giving the PWM output required for a desired acceleration at a given velocity,

\[ n = \frac{(a - c_R)}{b_R}. \]

Of course, there are other conditions in which the damper could be operating. Figure 8.49 shows the entire Java function used to represent the approximation to the damper’s inverse function.

There are a number of modes in which the damper can be operating; these modes are given a value in the parameter, `DamperMode`. This parameter can be useful for indicating the mode of operation. Refer to the example in figure 6.35 below.

- In mode zero the damper is operating in the desired linear range.
- Mode -1 is when the damper cannot supply the desired control force because of the passivity constraint.
- Mode 1 is when the stroke velocity is too low to supply the required force, no matter how large a voltage is applied. In this case the damper voltage is limited to the minimum needed for the maximum attainable force at that stroke velocity.
- Mode 2 is when the desired force is so small that even if no voltage is applied the damper still supplies a larger force. In this case the PWM output is 0.

The mode is only important to experimenters; it has no effect on damper function.
A second inverse function was developed that attempted to match the observed data more closely. Firstly, a function for acceleration in terms of PWM output and velocity was produced. This has the form shown in figure 6.36 where an example of the function is mapped on top of data from a recorded transient (refer to section 6.3.6). (This graph can be produced from the ProcessScopeDemo program by pressing the button labelled “Damper Force 2”.)

Figure 6.35 Damper Inverse Approximation and Modes

Figure 6.36 Damper Curve Fit
The graph has three linear sections, as in the graph supplied by LORD (see figure 6.22). The linear section through zero is independent of supplied voltage, while the two lines parallel to each other are linear in damper supplied voltage (PWM value) and in velocity. The curve itself is calculated using the Java code below. There are five parameters: $d, D, \alpha_I, \beta_I$ and floor. The parameters were set using sliders, matching the curves to data by eye.

```java
private double accelerationFunction(double n, double velocity)
{
    double v=velocity;
    if (v<0) v=-v;
    n=n-floor;
    if (n<0) n=0;
    double acc=d+n*alphaI*(D+v)*1E-4, t1=betaI*v;
    if (t1<acc) acc=t1;
    if (velocity>0) acc=-acc;
    return acc;
}
```

The section that is linear in both PWM value and velocity is most crucial to the inverse function. This is easily inverted algebraically. The problem becomes what to do with accelerations that are outside the line through zero: $\text{acc} > \beta_I v$. Here the closest achievable value of acceleration is used (given by the variable t1 in the code above).
The second inverse function uses the MCU code immediately below.

```c
double decay=5, prevN=0;
double damperInverse02(double acc, double velocity){
    double n=0;
    DamperMode=-1;
    double velLim=0;
    if (prevN<floor*0.125) velLim=0.5;
    if ((acc>0 && velocity<velLim)||(acc<0 && velocity>velLim)){
        double v=velocity;
        if (v<0) v=-v;
        if (acc<0) acc=-acc;
        n=((acc-d)/(alphaI*(D+v)))*1E4;
        n=n+floor;
        if (acc>betaI*v) n=betaI*v*n/acc;
        DamperMode=0;
    }
    if (n<floor){
        if (prevN>floor){
            n=floor;
        }else{
            prevN=prevN-decay;
            if (n<prevN)n=prevN;
            if (n<0) n=0;
        }
        if (DamperMode==0) DamperMode=2;
    }prevN=n;
    // if (DamperMode==-1) n=0; // See passivity constraint as zero in scope
    return n;
}
```

6.4. **Overshoot Experiments**

This section explains the overshoot experiments. These experiments demonstrate the bang-off-bang algorithm that avoids overshoot described in section 4.7.3.7. These experiments also provide validation of the rig input and output algorithms, and allow testing of the capacity of the rig to adapt to given conditions given the accuracy limits of the various components, input transducers, the damper control, the MCU ADC and PWM conversions, as well as the computational limits of the MCU. These experiments however are of independent interest into the viability of the controls developed in this thesis for handling overshoot.

The MCU can communicate via RS232 to a terminal program on a computer, in which parameters can be examined and values set. A screenshot of the terminal program
communicating with the MCU is shown below in figure 6.37. It is important to note that the control algorithm runs independently of the computer, and once the parameters are set, the control will run with the computer disconnected.

![Terminal Program (Screenshot)](image)

Figure 6.37 Terminal Program (Screenshot)

The rig was initially tested using numerical methods without using “live” data. This served two purposes. Firstly, it verifies that the Java code used on the computer also works in the C programming language on the MCU. (The code generally requires minor changes, but it needs checking. Furthermore, the code in the MCU has a lower numerical accuracy.) Since the MCU produced almost exactly the same result as found in the computer, the numerical methods in the MCU are confirmed.

Secondly, during these initial tests, distance, velocity and acceleration data were collected from the ADC input, even though it was not used for the numerical method. The reading and outputting of data take time and so overall timing on the MCU itself could be tested, even though the input data was not used.
In figure 6.38 below, the Newton Raphson method of section 4.7.3.6 has been used. (The initial values are \( d = 0.6, \ v = 0.063 \) and \( a = -1.074 \). The jerk and acceleration limits are 1 and 2 respectively.) The slowest response here is about 5 ms per step. The step size of the example shown in figure 6.38 is 0.1 s. The oscilloscope output shows clear steps of a larger size, indicating the speed limit of the algorithm. (The code to produce these results is to be found in the program, TestLand02.c, in the folder,

\[ \text{PhD\Experiment\Electronics\Atmel AVR\Test109 Test Landing 02.} \]

![Figure 6.38 Landing-Surface Acceleration, using Newton Raphson](image)

Conservatively the step size can be set at 7.5 ms (setting the 8-bit counter, TCNT0, to 197 at the start of the interrupt routine).

(The MCU code for the following experiments is contained in the program, TestLand04.c, in the folder,

\[ \text{PhD\Experiment\Electronics\Atmel AVR\Test111 Test Landing 04.} \]

During each 7.5 ms interrupt, the current distance, velocity and acceleration are estimated. Next the bang-off-bang (BOB) variant of the landing-surface method is called:

```
jerk=processContinuousBOB(voltage, velocity, acceleration);
The goal acceleration is calculated using the jerk.
accelerationEstimate=accelerationEstimate+step*jerk;
```
The inverse function is then called (in the routine damperExperiment) to determine a control voltage for the nonlinear damper control.

```plaintext
outVar = damperInverse01((accelerationEstimate - springAcc)/BETA, velocity);
```

This removes the acceleration that is already supplied by the spring, before calculating the required output value for the PWM (outVar). Figure 6.39 shows a digital oscilloscope recording of a drop experiment using this method.

![Digital Oscilloscope Recording](image)

**Figure 6.39** Data from Drop Experiment using Landing Surface

Experiments have been performed with a number of jerk control values, ranging over five orders of magnitude: 1 to 100,000 Vs\(^{-3}\). (The raw collected data can be found in the files jLand_xxxxxx.txt in the folder PhD\Eclipse RSpace Constraint\ScopeDataFiles02. All such data files in this section are contained in this folder.) Graphs of the various results are given below in figure 6.40. Note that time to reach zero decreases with increasing jerk control limit, as expected. With jerk values of 1 up to about 10,000 (four orders of magnitude) the method does well in reaching zero without overshoot. Beyond this the full control strength is needed. But even at these values the overshoot is constrained, even up to control limits of 100,000, and these would be acceptable for some applications, such as landing gear responses.
In some of the responses, a sawtooth control patterns is noticeable, for example where the jerk limit is 100. This is slightly different from the pure response that might be expected. However, with only roughly estimated state values (acceleration is particular), the system is adjusting to divergence from expected response by slowly adjusting control. Since the method responds to current states it adjusts for errors by slowly changing the control, rather than by making large changes.

In the cases where the control is about 500, a large control voltage is noticed as the system approaches rest. It should be remembered that the control voltage is not the same as the control force. The control force is affected by velocity and also by the spring force.
The following demonstrates a number of cases where actual acceleration is calculated off-board (by the software developed for the calculations in section 6.3.5). Graphs are shown in figure 6.41 (note the larger Voltage axis scaling in the last graph). The control values are 10, 100, 1000 and 10000 respectively. Acceleration \( (V_s^2) \) is scaled by 0.1 to fit clearly on the same graph as displacement and control voltage. (The raw data is contained in the files bLand_xx.txt.)

![Graphs showing displacement, voltage, and acceleration for different control values.](image)

**Figure 6.41 Landing Surface with Acceleration**

The acceleration is consistently small for controls with slow decay, but it increases slightly near the end for control values around 1,000. Such an increase in acceleration is part of the method for controls with a fast return to zero, and the effect is more pronounced with larger jerk limits. The method seems quite robust in handling the crucial final approach, even though this is the part of the control that is most subject to the damper nonlinearities.

The peak acceleration in the case when the control jerk limit set to 1,000 is 31 \( V_s^2 \) (recall that acceleration is scaled by 0.1) which is approximately 0.6 \( m/s^2 \) (using equation 6.1) or 0.06 g. The peak acceleration in the case in which the jerk limit is 10,000 is 180 \( V_s^2 \) which is approximately 3.5 \( m/s^2 \) or 0.36 g.
The controls examined above are implemented as bang-bang controls. The cases below all have a quite large jerk limit, of 8000 Vs\(^{-3}\), but with a range of acceleration limits. The graphs below include acceleration calculated on-board. Refer to figure 6.42 (the smoothing here is needed to calculate reliable acceleration values). The acceleration values used were 15, 50, 100 and unrestricted respectively. (The raw oscilloscope data is contained in the files bLand_008000_xx.txt.) The peak accelerations correspond roughly with these values. Thus the bang-off-bang control appears to be quite robust when used as a discrete method.

**Figure 6.42 Bang-Off-Bang Experiments**

Data collected using the second damper inverse function, `damperInverse02()`, is shown below in figure 6.43. The data is very similar to that collected for the purely linear inverse function, `damperInverse01()`. The graphs here show the acceleration calculated off-board as well. (The raw oscilloscope data is contained in the files bLand2_008000_xx.txt.)
Figure 6.43 Using the Second Damper Inverse Function

All of the experiments above begin from rest. One way to test the adaptability of the algorithm to various initial conditions is to only turn the damper algorithm on when a certain acceleration has been reached (a given acceleration value is more consistent than using velocity or distance as acceleration is the most critical parameter). Thus the next few experiments allow the acceleration to reach a certain value before the damper algorithm “kicks in”. In figure 6.44 the acceleration values have been set to 0 (unrestricted), 50, 100, 150, 175 and 200 respectively. (The raw oscilloscope data is saved in the files Kick_XXX.txt.)
The control limits on jerk and acceleration are quite low: 900 and 50 respectively. It is clear that without “kick in” the response is quite slow. As the acceleration “kick in” value increases however, the control responds more quickly, up until the kick in reaches 200 producing overshoot (although the overshoot is small). Thus the one algorithm has been able to adapt its response to the initial conditions.

The onset of overshoot at the limits of the rig’s functionality could be due to a number of factors: transducer tolerances, the problems of implementing a non-standard jerk control, the problems caused by subtracting spring acceleration, nonlinearities in the damper response, hysteresis in the damper response, electrical interference, latency caused by the MCU step size, as well as measurement latencies (and hysteresis in the accelerometer). Furthermore, when the method was tested with values read from the rig and transferred to computer, the numerical methods seemed to have the same similar limits as the rig. That is, even though the rig’s response has more adjustments caused by measurement errors, it seems to reach overshoot problems only a little earlier than the pure numerical methods. In all, given the modestly priced components used in the rig, it performed quite well in the above tests.
After these experiments it was decided to experiment with a digital accelerometer. The accelerometer was made to work but for various reasons, outlined in appendix 8.22, it was decided to run without this device.

6.5. Suspension Algorithm Tests

The physical experiments discussed in this section verify the practicality of the controls discussed in the numerical experiments. The controls can be implemented with even relatively modest equipment. The parameters used have been chosen by subjective judgement, thus the numerical performance results are suggestive rather than definitive. Nonetheless, as a proof of concept, the experiments show that the control algorithms are relatively easily implemented.

6.5.1. Damper Function and Inverse

The damper function and its inverse derived in section 6.3.7 do not fit the damper response curves as well as the function developed in this section. This section uses a piecewise-linear approximation which is still relatively easy to invert, but with scaling centred on a point, \((D,d)\), rather than zero. Figure 6.45 shows a portion of the function for a constant value of PWM output. The formula for the uppermost linear portion in figure 6.45 is

\[ a = d + (n - n_{\text{floor}})\alpha_i (D + v), \]

where \(a\) represents acceleration, \(n\) is the PWM output, \(v\) is the velocity, and \(\alpha_i, d, D,\) and \(n_{\text{floor}}\) are parameters. If \((n - n_{\text{floor}})\) is negative, then \(a\) becomes \(d\). The middle linear portion has the formula, \(a = \beta_j v\), and is independent of input, \(n\).
The parameters are adjusted by hand against graphs of acceleration versus velocity obtained from the digital oscilloscope (an example is shown in figure 6.46 below) until a good match is obtained across the range of PWM values. (This graph can be viewed by clicking the button “Damper Force2” in the “ProcessScopeDemo” Java program.) Values obtained with the new potentiometer are:

$$\alpha_i = 0.5526 \times 10^{-4}, \beta_i = 0.1358, d = 0.025, D = 10.0, \text{ and } n_{\text{floor}} = 75.$$

Inverting the outermost linear portions of the force function presents no problems. However, the PWM value cannot produce acceleration magnitudes above \( \beta_i v \), and there is little control available for forces below \( d \). The PWM control under these conditions has been chosen to approximate the controllable portion of the damper force curve. (Refer to the function damperInverse02 in the program ProcessScopeDemo.java.)
6.5.2. Experimental Algorithms

This section explains the control algorithms used in the physical experiment with the rig subject to stochastic road input. The algorithms chosen are the semi-active linear algorithms: passive, skyhook, linear acceleration and linear control over jerk also including modifications for virtual bump stops. Note that the rig is a semi-active rig and so active controls are impossible. The target algorithms are either clipped, or are modified with crossover removal (as described in section 4.8.3). The controls chosen here are the relatively simple ones that have been shown in the numerical experiments of section 5.7 to perform quite well.

In the previous “drop” experiments the base of the rig has remained stationary. However, for the experiments where the entire rig is subject to vibration, there are potentially six measures that are needed: \( s, \dot{s}, \ddot{s}, \) and \( y, \dot{y}, \ddot{y}. \) In fact, as shown in section 5.7, the two most crucial measurements for jerk control, \( s \) and \( \dot{y}, \) are read directly. The next two most important are the velocities, \( \dot{s} \) (for the passive and for crossover removal) and \( \ddot{y} \) (needed for the skyhook and other controls).

The value of \( \dot{s} \) is estimated by smoothing the distance measure difference from one time step to the next:

\[
\dot{s}_{t+1} = \alpha (s_{r+1} - s_r) + (1 - \alpha) \dot{s}_r,
\]

where \( \alpha \) is the smoothing factor (see section 8.12). In fact, the estimation of acceleration improves the linear approximation (as in the scalar Kalman filter of equation 2.6). Thus the following modified formula can be used,

\[
\dot{s}_{t+1} = \alpha (s_{r+1} - s_r) + (1 - \alpha) (\dot{s}_r + \ddot{s}_r \Delta t),
\]

where \( \Delta t \) is the step size.

The value of \( \ddot{y} \), however, requires integration of \( \ddot{y} \). Integration introduces the problem of integration drift due to the possible zero errors in \( \ddot{y}. \) There is then a compromise between latency, reducing measurement errors as well as integration drift. For the on-board integration calculation, simple addition might seem sufficient giving,
\[ \dot{y}_{t+1} = \dot{y}_t + \ddot{y}_{t+1} \Delta t, \]

where \( \dot{y} \) is the estimated velocity, \( \ddot{y} \) is the measured acceleration. Suppose there is a small non-zero offset of \( \varepsilon \) in the acceleration measure. Exponentially weighted moving averages (see section 8.12) could be used as in the self-zeroing algorithm of section 6.3.5, to reduce the effects of integration drift. Small factors, \( \alpha \) and \( \beta \), might be applied as in the following,

\[ \dot{y}_{t+1} = (1 - \alpha) \dot{y}_t + (1 - \beta) \ddot{y}_{t+1} \Delta t. \]

Overall drift can be estimated by solving,

\[ \dot{y}_\infty = (1 - \alpha) \dot{y}_\infty + (1 - \beta) \varepsilon \Delta t, \]

giving,

\[ \dot{y}_\infty = \frac{(1 - \beta) \varepsilon \Delta t}{\alpha}. \]

Clearly, when the value of \( \alpha \) is very small, the estimate of \( \dot{y} \) “blows up”. The value of \( \alpha \) therefore requires a compromise between a small value for accurate integration, and a relatively large value for keeping the integration from “blowing up” (changes in \( \beta \) have little effect and \( \beta = 0 \) can be used).

Drift from non-zero acceleration offsets is potentially aggravated by tilt. The acceleration of gravity is quite large compared to chassis vertical accelerations, so even a small tilt can perturb acceleration measures. Methods to help overcome this by actually measuring tilt and calculating its effect are possible, but this level of complication was not felt warranted in the following experiments. Efforts were taken to minimize tilt anomalies simply by minimizing rig tilt as much as possible, although it cannot be reduced to zero altogether.

The method used for adjusting Kalman multipliers and other moving-average coefficients was the same as that used in the physical experiments of section 6.4. The author experimented with different values until a compromise was reached that seemed to give as smooth a result as possible, with relatively accurate estimation and with as low a time lag as possible. With the bottom of the rig stationary the measures for \( s \) and \( y \) should be the same. Thus the corresponding time derivatives, \( \dot{s} \) and \( \dot{y} \), can be compared against each other, by simply bouncing the top of the rig.
It was found that the first integral of the accelerometer reading gave a surprisingly good self-zeroing integration. The drift was greater on the second integral, to determine $y$, as could be expected since the first drift adds somewhat to the second. Luckily, absolute height, $y$, was not crucial for any of the controls for the experiment, and indeed could be eliminated from them entirely. In the end, decay values were found that retained good agreement with the stroke values (with the base of the rig stationary), and which did not suffer appreciably from drift. The values used were $\alpha = 0.025$ for the first integral and $\alpha = 0.035$ for the second.

The control algorithms used in the physical experiment are all based on four linear controls: passive, skyhook, linear acceleration and linear jerk. The “virtual bump stop” modifications to these controls (which result in strictly nonlinear controls) have been discussed above in section 4.5.2, and fits the profile given in figure 4.7, with stiffening only on approach to the rattlespace limit. This was shown to improve on strictly linear controls in the numerical experiments and is a very simple to program in the microprocessor and runs very quickly. This modification applies to all the linear controls, doubling the number of possible controls to eight. A further modification is to apply the crossover removal algorithm discussed in section 4.8.3 given by equation 4.29 to equation 4.34. This is applied when the stroke rate is close to changing direction and is further explained below. This again doubles the number of possible algorithms to sixteen. This control also results in vast improvements in the numerical experiments. Out of these possibilities twelve were selected for the experiment, as shown in table 11 (the controls included are identified by their menu character in the AVR program).

<table>
<thead>
<tr>
<th>Control Type</th>
<th>Clipped</th>
<th>Crossover Removal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linear</td>
<td>Bump Stop</td>
</tr>
<tr>
<td>Passive</td>
<td>‘p’</td>
<td>‘P’</td>
</tr>
<tr>
<td>Skyhook</td>
<td>‘u’</td>
<td>‘U’</td>
</tr>
<tr>
<td>Linear Acceleration</td>
<td>‘G’</td>
<td></td>
</tr>
<tr>
<td>Linear Jerk</td>
<td>‘o’</td>
<td>‘O’</td>
</tr>
</tbody>
</table>

Table 11 Control Combinations for Experiment

One point that needs to be noted is that in the numerical experiments, semi-active systems are assumed to use a purely linear damper control in which the damping rate, $c$, could be varied. Although the supplied physical damper is usefully “linear”, it is actually piecewise linear. As discussed in section 6.5.1, the required damper force needs to be inverted. Nonetheless, the
damping rates for the numerical experiments were calculated on the basis of force (equivalently acceleration) for the controls of interest in this section.

Furthermore, the damping force function near crossover (which is where the crossover control is supposed to apply) has been shown to be very close to that of an ideal damper, and the potential damping rate is quite large. Thus the mechanisms that deal with crossover should require forces that lie below the absolute controllable limits of the MR damper. There is a subtle loss of controllability for very low forces from the damper, but these should not adversely affect crossover. The major problem in adapting the numerical control is the inaccuracies in the measure of stroke acceleration. This might have been helped by the use of an extra accelerometer on the base of the rig. The double differentiation process, however, has relatively large inaccuracies.

It is important to distinguish between the actual spring and the linear coefficient of relative displacement, $s$, which can be thought of as the “virtual spring”. The skyhook damper force can be seen to depend on a “virtual spring”, as discussed in section 2.7. In the experiments below, the linear passive and the skyhook use the actual spring. The skyhook however can still have crossover jerk because of clipping. Note too that the passive suspension of the rig uses a “virtual passive damper” with force proportional to stroke rate, $\dot{s}$, which ideally should not suffer crossover jerk.

At first it was thought that the experiment would use a measure of tracking that penalized collision with the rig’s rattlespace, as would seem natural. However, for the purpose of the experiment this is not necessary. The “rattlespace” of the experiment can lie within the physical rattlespace of the rig. In a sense, the tracking performance of the rig is measured by the tracking performance within a “virtual rattlespace”, smaller than the rattlespace of the rig.

This has a number of advantages, the foremost being that experiments can test the capacity of controls to resist travel beyong rattlespace limits without risking damage to the test equipment as it suffered violent collisions with its actual travel limits. The rig experiments then can hit the “virtual rattlespace” limits, fully testing the algorithms without risking damage to the rig. The measure of tracking performance can then use exactly the same tracking performance measure as the numerical experiments.
Furthermore, the rig’s actual travel is quite large, and experiments that tested its actual limits would require travel over large bumps risking the problem of tilt (risking the anomalies of tilt, discussed above in this section). The rig’s travel is large enough that even significant bumps can be constrained without hitting the actual rattlespace limits, while providing a realistic and substantial test of tracking capacity within the “virtual rattlespace”.

The linear coefficients chosen were based on drop and hand bounce experiments. The method of judging a coefficient depends on the parameter and the control. To judge the effect of the coefficient of stroke displacement, \( s \), on jerk control for example, it is necessary to hold the “chassis” away from zero for some time, to allow time for the integration to have an effect. To judge the effect of the coefficient of acceleration, it is necessary to produce relatively high accelerations. The best way to do this seemed to be to shake the “chassis” vigorously with relatively small but high oscillations.

A number of feedback controls have been implemented for the numerical experiments. The passive and skyhook controls each use a coefficient of 16 applied to the stroke and chassis velocities respectively. (The units for the velocity coefficients as used in the C program are \( V^{-1}s \). The units for coefficients of voltage are \( V^{-1} \), and the coefficients for the double rate of change of stroke and chassis voltage are \( V^{-1}s^2 \).)

The “virtual bump stop” controls all use the piecewise linear parameter profile as shown in figure 4.7 and, as discussed in this section is asymmetric. The parameter values used in the physical experiment are shown in table 12. Note that the values \( c_1 \) and \( c_2 \) represent fractions of the distance from equilibrium to the “virtual rattlespace” limit. In this table, \( s \) represents stroke and \( y \) represents chassis height (units V).
<table>
<thead>
<tr>
<th>Control Variable</th>
<th>Parameter Coefficient</th>
<th>$s_1$ (Low)</th>
<th>$c_1$</th>
<th>$s_2$ (High)</th>
<th>$c_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td>$s$</td>
<td>0</td>
<td>0.3</td>
<td>150</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>$\dot{s}$</td>
<td>7</td>
<td>0.1</td>
<td>30</td>
<td>0.7</td>
</tr>
<tr>
<td>Skyhook</td>
<td>$s$</td>
<td>0</td>
<td>0.3</td>
<td>150</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>$\dot{y}$</td>
<td>7</td>
<td>0.1</td>
<td>30</td>
<td>0.7</td>
</tr>
<tr>
<td>Acc.</td>
<td>$s$</td>
<td>100</td>
<td>0.1</td>
<td>600</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>$\dot{s}$</td>
<td>3</td>
<td>0.3</td>
<td>17</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>$y$</td>
<td>50</td>
<td>0.2</td>
<td>100</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>$\dot{y}$</td>
<td>10</td>
<td>0.2</td>
<td>25</td>
<td>0.7</td>
</tr>
<tr>
<td>Jerk</td>
<td>$s$</td>
<td>500</td>
<td>0.5</td>
<td>1000</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>$\dot{s}$</td>
<td>200</td>
<td>0.4</td>
<td>2000</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>$\dot{y}$</td>
<td>1000</td>
<td>0.1</td>
<td>1500</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>$\dot{\dot{y}}$</td>
<td>150</td>
<td>0.5</td>
<td>200</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 12 Skew Parameters for “Virtual Bump Stops”

Given that the road response characteristics were quite variable the control parameters were determined by experimenting with response to hand oscillations. This was not ideal but is within the proof of concept of the experiments performed over terrain. What would be ideal would be to determine the parameters using evolutionary algorithms to determine the parameters.

Crossover removal was tested with a simple “transient” experiment. The “target” damper acceleration was,

$$a_d = \begin{cases} -A & \text{if } s > 0, \\ 0 & \text{otherwise}. \end{cases}$$

Equation 6.17

Here, $A$, is a constant. The value $A = 400$ was used in the experiments below. This is purely a test target control; it allows the testing of crossover removal in exactly the same way as a similar target rule was used in the numerical demonstration at the end of section 4.8.3, and illustrated in figure 4.51. The simplicity of the target makes the effect of crossover removal more obvious (and the test is relatively easy to perform). The method of section 4.8.3 was used. (The C-code for the microprocessor is contained in the header file `semiactivecrossover01.h` in the folder, 

321
Figure 6.47 shows the effect of the target control as the system approaches crossover. The chassis displacement is shown in the upper red graph. The blue line shows the damper control build-up on approach to crossover, as the damper stiffens up in an attempt to maintain the acceleration. There was also an audible jolt with this “control”.

![Diagram](image)

**Figure 6.47 Test Target with No Crossover Removal**

The parameter ALPHA represents a correction for stochastic road movement (as explained in section 4.8.3). This value was set to 2 as it always seemed to become this value in the numerical evolutionary algorithms. The main parameter affecting crossover removal is JERK_CROSS, which is the jerk rate at which acceleration is dropped on approach to crossover. When this limit is high (about 30,000 or above in the numerical experiments) there was little effect. However, as this value is made lower (15,000) crossover removal can be observed, as shown below in figure 6.48 (a). Figure 6.48 (b) shows the effect when the crossover jerk limit is 6,000. (The value 6,000 was used in the experiments.)

![Diagram](image)

![Diagram](image)

(a) CROSS_JERK=15,000

(b) JERK_CROSS=6,000
Another parameter that can be used for crossover removal is to simply apply a maximal value for the damping rate. That is to say, limit the force that can be applied by the damper to,

\[ |a_d| \leq c_{\text{max}} s \]

Equation 6.18

for some value of \( c_{\text{max}} \). An example using damping rate limit only is shown below in figure 6.49. This seems to have a smoother profile, possibly due to the fact that stroke rate is somewhat better estimated than stroke acceleration. For the physical experiment the value \( c_{\text{max}} = 50 \) was used and the target damper force magnitude was kept below the minimum of both the values as given in equation 6.17 and equation 6.18.

![Crossover Removal using Damping Rate Limit](image)

In the end, the entire code was contained in a very small C header file. (Contained in SemiActiveCrossover01.h in the folder PhD\Experiment\Electronics\Atmel AVR\HeaderFiles.) The code used for the crossover itself is extremely simple. The code is shown in appendix 8.24.

**6.5.3. Implementation & Results**

For the purpose of testing, a program was written by the author for the AVR microcontroller which could change controls types and run each control for a set period of time calculating
comfort and tracking scores, as well as recording maximum jerk magnitude and maximum stroke magnitude. The program could be run without anyone attending to the rig. Once complete, data stored in arrays would be sent to the laptop via an RS232 connection. The controls run on the AVR microcontroller independently of the laptop, which is simply used to collect data and send instructions to initiate the data collection.

Given the small step size, 0.75 ms, and the fact that the step sizes were equal, the integrals of the penalties for comfort and tracking were calculated using a simple sum. Thus the comfort measure was calculated using,

\[ \sum (\dot{y}_{i+1} - \dot{y}_i)^2, \]

which is the numerical equivalent of equation 3.3. And the tracking score was calculated using,

\[ \sum \phi(\xi_i), \]

which is the numerical equivalent of equation 3.6. The penalty function, \( \phi \), is exactly the same as used in the numerical experiment, given by equation 3.7. The limits of the “virtual rattlespace” were set at 0.35 V above and below equilibrium. As described earlier, the measures from the potentiometer and the accelerometer are self-zeroing. (The routines to calculate the performance scores and to save the results for display are contained in the C header file, PerformanceCalculation.h, in the folder, PhD\Experiment\Electronics\Atmel AVR\HeaderFiles).

In the end, the code used about 60% of the code space and 80% of the data space. The AVR controller is a 64 K byte device, but this includes code for RS-232 communication, data collection, etc. The amount of memory needed for the actual control logic for the damper controls is extremely small.

Previously the XLR connector, power in, rig connectors, scope connectors, and the STK500 development board (containing the AVR MCU) had been connected using a couple of breadboards on a bench separate from the development board. A box was built mainly to house the STK500 development board and the associated wiring. The box is shown below in figure 6.50. Once connected, slightly different values were needed for the damper function (refer to section 6.5.1) and it was recalibrated. The new values are,
\( \alpha_j = 1.7368 \times 10^{-4}, \beta_j = 0.0726, d = 0.03, D = 13.29, \) and \( n_{\text{floor}} = 35. \)

The rig was then placed into a truck with a separate compartment for the rig and electronics, as shown in figure 6.51. This rig was then taken on a rough dirt road and data was collected while running over the road at approximately 35 kph.
The various algorithms were run for 20 seconds each, with results recorded for each algorithm. There were some algorithms that were run while the vehicle was turning, and seemed to have less rough conditions. The results are summarised below in table 13. The first column shows the menu character that is used to select the control from the RS 232 connection. The second shows the basic type of control: passive, skyhook, control over acceleration, and control over jerk (refer to section 4.5.1). The next column indicates whether or not “virtual bump stops” were used (refer to section 4.5.2.2). The next column refers to whether or not virtual crossover was employed. Note that the performance measures are sums, so lower scores indicate better performance.
<table>
<thead>
<tr>
<th>Menu Character</th>
<th>Basic Control Type</th>
<th>Virtual Bump Stop</th>
<th>Crossover Removal</th>
<th>Comfort Performance ×10⁴</th>
<th>Tracking Performance</th>
<th>Max Jerk</th>
<th>Max Stroke s</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘p’</td>
<td>Passive</td>
<td>No</td>
<td>No</td>
<td>37.8</td>
<td>2.072</td>
<td>162</td>
<td>1.15</td>
</tr>
<tr>
<td>‘P’</td>
<td>Passive</td>
<td>Yes</td>
<td>No</td>
<td>26.9</td>
<td>1.606</td>
<td>137</td>
<td>0.81</td>
</tr>
<tr>
<td>‘u’</td>
<td>Skyhook</td>
<td>No</td>
<td>No</td>
<td>114.1</td>
<td>1.198</td>
<td>196</td>
<td>0.88</td>
</tr>
<tr>
<td>‘U’</td>
<td>Skyhook</td>
<td>Yes</td>
<td>No</td>
<td>215.9</td>
<td>828</td>
<td>205</td>
<td>1.307</td>
</tr>
<tr>
<td>‘e’</td>
<td>Skyhook</td>
<td>No</td>
<td>Yes</td>
<td>9,203</td>
<td>26.3</td>
<td>624</td>
<td>0.455</td>
</tr>
<tr>
<td>‘E’</td>
<td>Skyhook</td>
<td>Yes</td>
<td>Yes</td>
<td>1,812</td>
<td>219</td>
<td>471</td>
<td>0.826</td>
</tr>
<tr>
<td>‘G’</td>
<td>Acc.</td>
<td>Yes</td>
<td>No</td>
<td>2,252</td>
<td>101</td>
<td>448</td>
<td>1.015</td>
</tr>
<tr>
<td>‘Q’</td>
<td>Acc.</td>
<td>Yes</td>
<td>Yes</td>
<td>17,511</td>
<td>0.0423</td>
<td>561</td>
<td>0.327</td>
</tr>
<tr>
<td>‘o’</td>
<td>Jerk</td>
<td>No</td>
<td>No</td>
<td>16,251</td>
<td>67.0</td>
<td>619</td>
<td>1.649</td>
</tr>
<tr>
<td>‘O’</td>
<td>Jerk</td>
<td>Yes</td>
<td>No</td>
<td>3,595</td>
<td>93.7</td>
<td>534</td>
<td>1.436</td>
</tr>
<tr>
<td>‘m’</td>
<td>Jerk</td>
<td>No</td>
<td>Yes</td>
<td>483</td>
<td>75×10⁴</td>
<td>229</td>
<td>0.287</td>
</tr>
<tr>
<td>‘M’</td>
<td>Jerk</td>
<td>Yes</td>
<td>Yes</td>
<td>2,627</td>
<td>1.692</td>
<td>667</td>
<td>0.400</td>
</tr>
</tbody>
</table>

Table 13 Results of Run on Road

There are suggestive patterns that emerge from the data. Firstly it can be seen that the passive control is quite soft, resulting in low comfort scores (low values are desirable here) but relatively high tracking scores. The skyhook is not as soft, and improves greatly on the tracking for the version with the virtual bump stop. The use of crossover removal with the skyhook does not produce a lower comfort measure but produces a better tracking performance, perhaps indicating that the system has a better mix of strategies overall. Similarly, crossover produces better tracking with the acceleration control.

The results however should not be viewed as conclusive without further investigation. There are two main reasons for this. Firstly, the control parameters were determined subjectively, and the final control parameters were not tuned to the roughness of the conditions encountered. Secondly, it was impossible under the test conditions to run the controls over the exact same bumps for each control type. Nonetheless, as proof of concept they show that the controls suggested by the numerical studies bear up to first-order physical experimentation.

In the case of jerk control the inclusion of the virtual bump stop produced improvement in tracking as expected. Indeed the control over jerk with bump stop and crossover removal had the best tracking measure of all, 1.692, at the cost of a little comfort due to the bump stop. These experiments taken alone indicate that control over jerk is effective as a control, and that bump stops and crossover removal may be effective.
A record of the stroke measures and the chassis height measures of a short section of road (about 5 seconds) are shown below in figure 6.52. This clearly shows that the chassis height and stroke are different. Thus, for example, the skyhook, which responds to chassis velocity, will respond differently to the passive, which responds to stroke velocity.

Figure 6.52 Chassis and Stroke
7. Summary and Conclusions

This chapter summarizes the theory and results of the thesis. Also, conclusions arising from the research are presented.

7.1. Overview

Passive suspension control has in the past rested mainly on linear theory. With the virtually unlimited flexibility of electronic systems, a much greater range of control laws becomes viable. As noted by Ross, “a critical part of designing a practical control system is … getting the problem formation right in the first place!” (2009, p51) In particular, it has been shown that the most often used performance measures for comfort, least squares acceleration, is fundamentally flawed. In fact, it is necessary when dealing with controlled suspensions to reassess performance measures in general. This thesis has attempted a very general first-order overview of suspension control laws and their performance goals, using analytical and numerical techniques, and with a number of novel approaches.

The main focus of the thesis is on limited-stroke suspensions, with constrained suspension travel limits, concentrating particularly on the goals of comfort and tracking. The central example of such a suspension is a road-going passenger vehicle, but all suspensions have limited stroke, and some elements of the theory could be applied even more generally.

The control dealing with the problem of coming to rest with constrained jerk and no overshoot may particularly have application in other areas. This control has the potential to reduce settling time and it could also be applied to control systems with similar performance goals, such as robotic arms movement, the parking of hard drive heads, heavy door closing, an elevator’s approach to a floor, satellite rotation or possibly even to the control of airplane landing gear (this control is discussed in section 4.7.3 and has been investigated physically in section 6.4).
A large number of electronic suspension control algorithms, including many new heuristic controls developed by this author, were investigated numerically in this thesis. Systems were compared by modelling different controls in a consistent environment using evolutionary algorithms. Highly scoring controls were chosen for implementation in a simple physical experimental rig.

### 7.2. Numerical Modelling Investigation

As described in chapter 5, the control parameters for various controls were coded as genes and were optimized using evolutionary algorithms. Thus the algorithm can be conveniently identified by the name of the gene used in the EAs.

The effect of evolutionary temperature (see section 2.12.2) on numerical evolution was apparent. In preliminary tests a “hot and fast” evolutionary schedule, HotQuickGenDB, could be used. This confirms the viability of the control, but it is too “hot” (refer to section 2.12.2) to produce reliable optimality. For the final tests, a schedule which began very “hot” and which cooled very slowly, LongGenDB, produced much better results at the expense of time. The final run of the evolutionary algorithms took almost 10 days to complete, with an average of 2 hours for each control. This was judged to be about the minimum time needed to produce reliable results.

What is immediately clear is that the skyhook benchmark control, ActivePureSkyhookGene, performs quite well despite the fact that it is one of the simplest algorithms tested. The linear skyhook was implemented very early in the research and has been included in a number of papers published by the author during the research. It was consistently found to perform well. It certainly outperforms the linear passive control. There were only 12 controls that outperformed the skyhook, some being semi-active controls, and some performing very much better. In contrast, there were 81 controls that outperformed the benchmark linear passive control.
During the course of the research, evolutionary algorithms were implemented and presented for conference and journal papers. The various results showed consistency where there was overlap. The ordering of the final results was reasonably representative of the relative performance of the different control algorithms. Even so, in interpreting numerical results there are qualifying factors to consider.

Firstly, the evolutionary process does not find optimal results, rather it produces suboptimal results (refer to section 2.12.1). There always exists the possibility of higher performing results. This should to be balanced against the fact that controls produced by evolutionary algorithms have a degree of inherent “robustness”, in the sense that the resulting systems generally have a reasonable degree of tolerance to variations in control parameters, and they can be resilient to changes in external conditions (refer to section 2.3.2).

Optimization using Pareto fronts has been implemented in the experiments as a means of increasing genetic variation. With multi-objective performance measures there may be a different ordering when the separate performance elements are weighted differently, corresponding to intersecting Pareto fronts. Raw scores for the separate factors have been included in the results. Note however, that this is with particular weights of the separate factors; final Pareto fronts have not been found.

It is important to distinguish between active and semi-active controls, since active controls are free from the passivity constraint. However, comparing both active and semi-active controls in the same modelled environment produces a reasonable indication of the relative strengths of the controls. One point is clear: an active control that is outperformed by a number of semi-active controls has little chance of being a viable suspension control.

Taking into account all the qualifications discussed above, the controls that perform well in the numerical experiments are good candidates for practical electronic controls and are worthy of further investigation. The results provide good comparative data on candidate suspension control systems.

As noted in section 5.7 the linear controls perform quite well, including the linear skyhook. Small modifications improved the linear controls. “Virtual bump stops”, for instance, improved suspension performance in numerical experiments (refer to section 5.7). The term
is used here to generally describe control modifications that increase control strength with approach to the rattlespace limit. It should not be inferred that “Virtual bump stops” are just controls that model actual bump stops.

The numerical results show that controls over jerk generally outperformed controls over acceleration. It was difficult at the outset to anticipate this result because jerk control is less direct and “slower” than control over acceleration. It is a strong recommendation of this thesis that controls over jerk be considered for future investigation.

There were two input transducers on the physical rig. An accelerometer was placed at the end of the top arm (“chassis”), and a potentiometer registered the relative movement between the top arm and base (between the “chassis” and “wheel”). If the accelerometer were to be placed on the “wheel” rather than the “chassis”, there would have been a higher measurement error because of the greater vibration of the “wheel”.

The numerical experiments with linear controls confirmed that this was a good arrangement for the collection of data, in the sense that it was higher derivatives of the chassis movement and lower derivatives of relative stroke movement that were given higher linear coefficients. In the linear control over jerk, the numerical experiments indicated that the chassis acceleration coefficient, \( \ddot{y} \), should be quite high. This measure can be estimated directly from an accelerometer on the chassis.

Semi-active suspensions suffer from a problem that active systems do not: the damper can only supply force in one direction. For example, if the damper is compressing, then the damper can only supply an upward force on the chassis. This is known generally as the “passivity constraint” (explained in section 2.6). It becomes a problem for controllable dampers when designers try to adapt a semi-active control to follow a desirable control law. The obvious thing to do is to clip the control within the passivity constraint, but this produces enormous jerk when the stroke changes direction (discussed in section 4.8.1). For the sake of discussion, this problem is here termed “crossover jerk”.

In regard to semi-active suspensions one result of this thesis is clear: some form of crossover jerk removal is necessary. An early method to address the problem, called the “no-jerk
skyhook”, has been known since 1997 (Reichert, 1997; Ahmadian et al., 2004). In fact, the no-jerk skyhook performed quite badly in the numerical experiment, obtaining a score of 221, even lower than the passive with 321.

This author’s “lo-jerk skyhook” removes the jerk that can accompany a change in the direction of stroke velocity (refer to section 4.8.1). This idea was made explicit in the paper of 2006 and 2008 (Storey et al., 2006; Storey et al., 2008). The notion that a high performing control could be used outside the period when stroke rate changes was discussed in the 2006 paper, and made explicit in the paper of 2008. Thus a crossover removal method can work in tandem with a high-performing “target” control, with the crossover removal method cutting in when crossover is imminent.

After looking carefully at the physics of the problem of “crossover removal”, a method was developed that performed very well in numerical experiments (see section 4.8.3). In fact, it outperformed other previous methods of addressing the problem developed by the author. The highest performing semi-active control used this crossover removal method combined with a distinct high performance “target” control, namely the linear control over jerk with virtual bump stop (also confirming the high performance of the target control: the linear control over jerk with virtual bump stops).

Over time, subtly different forms of crossover removal were developed. One of the problems is that different forms of crossover removal make it difficult to judge the relative impact of the target control. However the same crossover removal method was used for the following (including the highest performing semi-active control):

```
FlatLinearJerk01SkewSemiCross, FlatLinearJerk01SemiCross,
FlatLinearAcceleration01SkewSemiCross, PureSkyhookSemiCross,
xMinTimeHardenSemiCross, xLandingEdgeCenterSemiCross, and
FlatLinearAcceleration01SemiCross.
```

These show roughly the same relative order of success as the active controls on which the target controls are based.

“Rattlespace constraint” controls performed very highly given their complexity and given the fact that their development for suspension control occurred relatively late in the research.
RCollisionAvoidSemi03 achieved a score of 815 while the xMinTimeHardenSemiCross and xLandingEdgeCenterSemiCross both achieved a score of roughly 807 (compared to the highest performing semi-active score of 824). In this author’s estimation, this approach will probably eventually derive the highest performing suspension controls.

7.3. Physical Experiments

There were two main types of experiments performed for this thesis. The first are the “drop” or “transient” experiments performed without a moving base. The second are the experiments with the rig placed on a moving platform.

The initial transient experiments permitted the collection of data on the rig’s parameters, most importantly on the damper control force vs. voltage control function. The damper’s control was verified by applying the control technique of section 4.7.3.7 for bringing the top arm smoothly to rest without overshoot. The landing-surface method proved effective over a wide range of parameter values in physical experiments.

The experiments on the truck platform, explained in section 6.5, verify the viability of control over jerk. The experiments demonstrate that the controls used in the numerical experiments are practical. These controls have been implemented in a cheap, modest system showing that they are more than simple enough to be used with modern microprocessor control. The experiments also show that virtual bump stops can be effective in improving the compromise between comfort and tracking, and that crossover removal improves comfort.

7.4. Original Contribution

Linear controls have proven very successful in the numerical experiments. Linear systems include the traditional linear passive suspension and the “linear skyhook control” (refer to section 2.4). Linear controls can depend more generally on any linear combination of stroke
displacement and absolute chassis displacement, as well as the first and second time derivatives of these (see equation 4.6 and equation 4.7).

Evolutionary algorithms have been used to determine linear coefficients for a general linear control over acceleration. The results show that the optimization process favoured the skyhook in the sense that the linear coefficients chassis velocity and stroke displacement were much higher than the coefficients of chassis displacement and stroke velocity.

Controls over jerk generally outperformed controls over acceleration (refer to section 5.7). This is strong evidence for the practicality of control over jerk. Indeed, such controls can be conveniently implemented because they employ direct measures of chassis acceleration, which are readily collected using accelerometers (as discussed in section 7.2).

Nonlinear adjustments in the form of “virtual bump stops” (refer to section 4.5.2.2) led to improvements on purely linear controls. The term “virtual bump stop” here refers to controls that employ a deliberate stiffening of the suspension as it approaches the rattlespace limit. The analogy with actual bump stops should be obvious except that some virtual bump stop force profiles cannot be achieved with neoprene or rubber bump stops. These have been investigated in the case of controls using jerk, as well as controls over acceleration. In both cases virtual bump stops brought performance improvement.

Throughout the research linear controls consistently performed well. Some of the modified linear systems are quite complex: the FlatLinearJerk01SkewSemiCross gene for example (the highest performing semi-active control based on a linear method) had 30 real-number components in the genome (compared with just 2 genes each for the two benchmark controls).

The purely linear skyhook described in section 2.4 is not optimal in any mathematical sense, although the damper in the nonlinear and very uncomfortable on-off skyhook is optimal for energy absorption, as shown in section 4.2. There were a number of controls that performed much better than the linear skyhook in the numerical experiments. Nonetheless, it performed very well and was consistently at the high end of the performance range.
The skyhook’s success can be explained to a great extent by comparing its transmissibility with that of the passive suspension (see section 2.4), and by the fact that it has good energy absorbing characteristics, as described in section 4.2. Furthermore the pure skyhook does not produce sharp jerks over road slope discontinuities, as discussed in section 4.3.

Section 4.3 explores the general issue of the susceptibility of a suspension control to produce high jerk over road discontinuities. This matter is worthy of a short section in the thesis because it not only points out an easily characterized theoretical superiority of such controls, but it also highlights some practical problems with them. For instance, such controls require extremely fast control responses, because they have to react to instantaneous changes in road slope or height. In the absence of very fast control response, a semi-active control will perform best over road slope discontinuities if the damper is generally soft when such discontinuities are encountered.

Section 4.2 critiques the claim that “since the semiactive damper does not add any energy into the system, the system is stable” (Song et al., 2003, p227). In fact a very uncomfortable semi-active system could be designed with the damper deliberately employed to increase the kinetic energy of the chassis where possible. The energy could effectively “ratchet upwards” because of the damper. It is not strictly correct to say that such a damper is dissipative, and it is not strictly the case that such a damper “cannot input energy into the controlled system” (Dyke et al., 1996, p565).

Chapter 3 contains a theoretical investigation of comfort and tracking measures and provides justification for the performance measures used in the experiments. Ironically, the virtually limitless flexibility of electronic control and of numerical optimization creates its own set of problems. Perhaps the most important example of this is the suspension control defect of “acceleration discontinuities” (sudden changes in acceleration, which are spikes in jerk). These deleterious artefacts are masked by least squares measures of acceleration.

There are a number of ways in which acceleration discontinuities can be produced. The on-off skyhook has the property of absorbing chassis kinetic energy at the maximum rate for a semi-active suspension (see section 4.2), but it has been shown to produce acceleration discontinuities (McLellan, 1998, p55; Guglielmino et al., 2008, p71). Controls that are switched between full on and full off are generally more convenient to engineer than controls
with a continuous range of output. This control defect could also arise from the use of minimum-time bang-bang control over acceleration (refer to section 2.3.2.3). Sliding-mode control is known to produce chattering if the control is not modified near equilibrium (discussed in section 2.3.2.4).

Force discontinuities are also produced by clipped semi-active controls, as discussed in section 4.8, and the discomfort caused by this has been shown experimentally (Ivers and Miller, 1991, p337). The necessity of crossover removal, so obvious in physical experiments, might be masked altogether if least squares acceleration is used as the performance measure of smoothness.

One of the major theoretical conclusions of this thesis is that jerk is a more appropriate measure of comfort than acceleration. This notion has been defended at length in chapter 3. The matter is complex and a number of factors need to be taken into consideration (as discussed in section 3.1). Perhaps the most conclusive evidence for the superiority of jerk over acceleration as a measure of comfort is to be found in experiments into human movement. These indicate that jerk and perhaps even higher-order derivatives of motion are used by human biological systems for the control of motion (Flash and Hogan, 1985; Harris, 2004).

Just as comfort performance measures should reflect true suspension goals so should tracking performance measures. What seemed at first to be a relatively simple matter emerged as a complex and intriguing theoretical problem. This matter has been discussed in detail in section 3.2.

Hitting very hard against the limits of the suspension travel is damaging to a vehicle and is extremely uncomfortable, and travel beyond the suspension travel limits is clearly infeasible. A real suspension has failed catastrophically once the chassis has smashed through the rattlespace limit. So how can travel outside the rattlespace be allowed in the model? Surely, optimization techniques that allow travel outside the rattlespace limits are unrealistic? Tracking performance measures that involve penalties for rattlespace travel limits have not been proposed previously, to the best of this author’s knowledge.
The matter is further complicated by an inconvenient fact: in the real world, drivers slow down when they approach large and potentially uncomfortable bumps. Indeed this phenomenon is purposefully exploited by speed bumps.

It seems at first possible to include the discomfort of hitting hard against rattlespace limits as a component of the performance measure, unifying, as it were, comfort and tracking into one performance measure. This would seem at first to be an elegant method that also solves the problem of travel outside the rattlespace. The problem is that drivers respond to large jolts, especially the threat of rattlespace collisions, by slowing down, altering the road vertical height profile as a function of time. Without driver slowdown these jerks have effects on optimization that are irrelevant to the suspension problem because they are not experienced in real driving.

An important aspect of this problem then is the fact that vehicle suspension models are unrealistic if they do not include driver slowdown. Even the worst suspensions (within reason) can be made smooth and comfortable simply by going slowly enough. On the other hand, even the best suspensions can be made uncomfortable over soft bumps, simply by going fast enough. Changing forward speed changes the bump profile as a function of time. One consequence of this is that real drivers avoid excessively rough conditions by slowing down. They do this to avoid both discomfort and vehicle damage. This effect can be readily observed by watching cars as they pass over speed bumps, which are of course specifically designed for this purpose.

Including rattlespace limits in the model and using a “unified” measure of both comfort and tracking may be an elegant solution, but it is ultimately also unrealistic. And it is likely to produce anomalous results in evolutionary algorithms (refer to section 3.2).

It would seem then that the most “accurate” approach would be to realistically model driver slowdown in the evolutionary algorithms. Frustration at having to slow down would then be at least a factor in a measure of tracking. However, the method is untried. The method itself needs research.

On balance it was felt best to proceed with another method that did not diverge too greatly from standard methods. This measure penalizes travel near and beyond the rattlespace travel
limits, as explained in section 3.2. This method has the advantage that it deviates only slightly from traditional techniques.

Nonetheless, an outline for a method for modelling driver slowdown was proposed. Rather than model a host of complex psychological factors, driver slowdown could be approximated by making simplifying assumptions about driver behaviour. In particular, it can be assumed that only under the roughest conditions would an experienced driver allow the suspension to hit violently against the rattlespace limits. An iterative method could be used to approximate this behaviour.

Although driver slowdown was not modelled in the numerical experiments, it is an important factor in the theoretical issue of tracking measures, and the entire discussion of section 3.2 is needed in order to fully explain the reasoning behind the compromise tracking performance measure used in this thesis.

It is clear that electronic suspension controls cannot be properly tested in evolutionary algorithms unless there are road surfaces that threaten to create collisions with the rattlespace limits; at least a small proportion of bumps should be at least one half the rattlespace in height. If such bumps do not occur with reasonable frequency, then the optimization process can produce softer and softer suspensions, at the extreme producing the softest possible suspension: the “flat” control, which simply keeps the chassis flat (refer to section 3.2). The random roads used in the numerical experiments erred perhaps on the side of being rougher than necessary.

It is certainly true that “the Linear Quadratic Regulator (LQR) has been used as one of the main control techniques for dealing with active suspension design” (Camino et al., 1999, p3168). The LQR control has been the basis of controls developed by a number of authors (Yedavalli and Liu, 1994; Tseng and Hendrick, 1994; Giorgetti et al., 2006; Takahashi et al., 2000; Johnson and Erkus, 2002). For the sake of discussion it was even dubbed by Wagner and Liu, the “optimal control” (2000, p568), although of course it is optimal only for the particular case of the LQ problem (MacCluer, 2005, p151). LQR control has been extended above to control over jerk (refer to section 4.5.1).
The minimum-time control of section 2.3.2.3 can also be extended to controls over jerk (refer to section 4.6). The minimum-time control over acceleration was used by Pontryagin as a first example of optimal control with control constraints (Pontryagin et al., 1986), and the control has a long history. This author found the minimum-time, bang-bang control to return a system to rest (including zero acceleration) using jerk and proved this method using Pontryagin’s Principle (in section 4.6).

A simple, closed-loop, discretised version of the control (i.e. a real-time optimal control, RTOC) for implementing this bang-bang control was found by the author (see section 4.6.1), although, as described in section 2.3.2.3, a literature search revealed that this method seems to have been applied by Koh et al. in the area of mechatronics (1999) although it is not analysed.

The theory for a separate, novel category of suspension controls has been developed for this thesis in section 4.7. These are termed “rattlespace constraint” controls, in which the trajectory of the chassis is controlled to remain within the rattlespace limits, at the same time applying smooth movements to do so. This category of controls is distinguished by the fact that they are based on models of chassis trajectories as well as some kind of “prediction” of the future rattlespace trajectory, as described in section 4.7.

Two broad variants of this kind of control are identified: firstly, controls that stiffen when they determine that the maximum travel is likely to hit or come close to a rattlespace limit, termed here “variable hardness” controls, and secondly, controls that plot a trajectory as they approach a given limit, called “edge constraint” controls.

The second subcategory is more complex. These controls require a projection of the likely future movement of the suspension rattlespace. This is a complex stochastic system with constraints on two sides. It can be difficult to distinguish whether a momentary change in road height signals a large bump or is merely part of a small corrugation. Ultimately, this is a matter of road height statistics, but simple heuristic methods have been developed above for anticipating future road movement.
The problem of rattlespace constraint is naturally associated with the problem of approaching a limit curve without overshoot. This “displacement constraint” problem (refer to section 4.7.3) is of some interest in its own right.

It soon became clear that in this problem there is a trade-off between smoothness and “rebound”. Suppose for the sake of argument that the chassis approaches the bottom rattlespace limit and manages just to skim the limit with zero relative velocity, but there is some residual acceleration. How is this acceleration to be handled? The acceleration cannot be suddenly dropped to zero without causing a spike in jerk. A smooth jerk will need to be applied to reduce this residual acceleration. But if the residual acceleration is too large, a smooth decrease in acceleration may not be sufficient, because the chassis will rebound back towards the opposite rattlespace limit. The residual acceleration “pushes” the chassis back towards the other rattlespace limit.

In section 4.7.3.1, if the initial instant is ignored, it has been shown that rebound acceleration can be reduced at the price of larger control force at the start. In this preliminary analysis it is shown that constant energy absorption can produce a considerable rebound force, contributing potentially to instability. About one-and-one-half to two times this initial force (depending on the method of force decay) is required to produce a smooth, controlled response later. This seems to suggest that larger control forces than are necessary for energy absorption alone are needed to produce stability. Finding a more definitive solution to this problem that did not ignore the acceleration of the initial instant became an important focus of the research.

Over many months of experimentation, a number of controls were developed for reaching rest (zero distance and velocity, as well as zero acceleration) from arbitrary initial conditions. The initial controls worked as open-loop, planned controls (with pre-planned movement), but they failed when adapted as feedback controls. These preliminary investigations are briefly explained in section 4.7.3.2. A control was initially found that seemed to work in all initial conditions, but it was very complex and highly inelegant (as is readily evident in the example at the end of section 4.7.3.2).

Minimum-time controls over jerk were also being investigated at this time. These controls do not suffer from acceleration discontinuities, either at the initial time or at the time they reach
zero. The problems of bringing a system to zero in minimum time using constrained acceleration or jerk are solved using the bang-bang minimum-time controls of sections 2.3.2.3 and 4.6. However, the added constraint that the trajectory should not overshoot complicates the optimization problem.

What seemed at the time to be a quite startling and curious phenomenon was observed while experimenting with a graphical computer program (written by this author during the course of the PhD research) that implemented the bang-bang control over jerk with arbitrary initial conditions (arbitrary initial distance, velocity and acceleration at time zero). No matter what initial conditions were used, the minimum-time bang-bang control that returned to zero using least jerk without overshoot was always the control that has only one intermediary control switch (instead of two in the general case). This was observed again and again, with a wide range of initial conditions. An illustrative example can be seen above in figure 4.29 (b) in section 4.7.3.3. This method also provides a simple and elegant feedback control as discussed below, neatly solving a problem that consumed many hours of investigation.

A mathematical proof was found by this author that this is indeed the minimum-time control using least jerk to reach zero without overshoot (refer to section 4.7.3.4). For the purposes of discussion this control was termed the “landing surface control” (because the initial point is on the landing surface). The proof involved equations that later were employed in implementing a feedback version of the control.

Other controls were also found that could reach zero without overshoot and with even lower jerk values, but all these involved some amount of rebound. These have generally been termed “skim” controls for the sake of this discussion, because they “skim” the time axis as explained in section 4.7.3.5. The absolute minimum jerk is achieved by the “minimum-jerk skim” control, as proven in section 4.7.3.5.

An example using a jerk limit intermediate between the landing-surface and the minimum jerk skim is illustrated in figure 4.37. These also have intermediate rebound between the zero rebound of the landing-surface method and the maximum rebound of the minimum-jerk skim control. Thus there is a trade-off between smoothness and rebound.
Showing that the general skim controls are minimum-time, or at least are local minima, requires showing that small perturbations (as employed in the proof of Pontryagin’s theorem) produce only increases in overall control time. This is explained in more detail in the appended section 8.16 and is outlined in section 4.7.3.5. Such increases have been shown in numerical testing.

To complete the range of minimum-time controls for jerk constraint limits that are larger than the landing-surface control requires just the standard minimum-time controls that are unconstrained by displacement, since these do not overshoot in any case.

This type of control and bang-off-bang variants developed above (refer to section 4.7.3.7) may be of independent interest. Since acceleration is smoothly released and brought to zero, there is good reason for thinking that movement at the completion of the control is not subject to lingering vibrations, and settling time should be reduced. As remarked by Koh et al., “the anti-vibration is the key factor for determining the life cycle of the mechanism. In robotic systems, the jerk constrained motion guarantees a smooth and stable motion” (1999, p273). Thus a smoother settling control can also reduce wear and tear on the mechanism.

The control of satellite rotation investigated by Zadeh (2004) was developed and tested numerically using the evolution of fuzzy logic controls, and it was found to decrease costly settling time (Kirk, 1970). The control force tails off at the end in a way that has some similarities with the landing-surface control. By decreasing or removing vibrations at the end of the control movement, the satellite is ready for use in shorter time; the system becomes operational almost immediately after the rotational movement is complete rather than having to wait for extraneous vibrations to die out.

The landing-surface control has a wide range of applications where a fast movement is followed by a precise target positioning without residual vibrations. Such applications include robotic arm approach to another machine (Constantinescu and Croft, 2000), the closing of a heavy door, lift control (Peters, 1995), parking a hard-drive head (Chang and Hori, 2006), rotation of satellites (Zadeh, 2004), or even for aircraft landing gear (Krüger, 2002).

The landing-surface control is simple, elegant and stable when used as a closed-loop control (refer to section 4.7.3.3), and it seems to resolve all the problems of the heuristic controls
developed in section 4.7.3.2. The main reason for this is Bellman’s principle of optimality (Kirk, 1970, p54) (refer also to the discussion in section 2.3.2.3); “If [control] $u$ is optimal on $[t_1, t_2]$, it is optimal on every subinterval” (MacCluer, 2005, p121). The control then is a perfect candidate for a real-time closed-loop control, using only state estimations as in conventional feed-back systems controlled by a microcontroller.

The algebra for the control algorithm requires the solution of a fourth-order polynomial, and while such equations do have a closed-form solution (Dixon, 2008, pp385-91), an iterative method using Newton Raphson was employed here (as discussed in section 4.7.3.6). The discretised control was tested in a relatively slow microprocessor (18 MHz) and with cheap input transducers in the physical test rig and was found to operate comfortably at around 7.5 ms intervals (refer to section 6.4).

The “landing surface” control can be used as the final stage of a closed-loop control that includes other constraints. The extremes of the control are easily calculated in real time; the maximum distance, velocity and acceleration are found by very simple algebra. By calculating these extremes and only switching to this control when needed, it can be adapted to deal with what is generally the most difficult part of such a movement: the final “docking” movement, as demonstrated in section 4.7.3.7.

This results in “bang-off-bang” controls. A “bang-off-bang” control was also developed that applied acceleration limits, although this requires further research as discussed in section 4.7.3.7. A bang-off-bang control was implemented in the physical rig, as shown in section 6.4.

Assuming the conjectured optimality of the general “skim” controls developed in section 4.7.3.5, these provide a theory for using constrained jerk for achieving equilibrium in a rattlespace constraint, shown at the end of that section. The details of the control (especially the rate of control decay when moving from one rattlespace edge to the other) depend on the relative importance of smoothness compared to settling time.

Because of the passivity constraint (see section 2.6) there is a propensity for a semi-active suspension to produce sudden changes of acceleration when the stroke rate changes direction.
This problem has been known since at least 2004 when Ahmadian et al. proposed a global control law that alleviated the problem (2004). In 2006 the author published a paper which explained that the problem was manifested specifically when the stroke rate approaches zero (Storey et al., 2006). For the sake of the discussion, the point at which the stroke rate reaches zero is here termed “crossover”: the stroke rate “crosses” between negative and positive, or vice versa.

A number of methods for anticipating and removing “crossover jerk” were developed over time. Eventually a method was produced which assumes that a controlled near-constant jerk can be used to bring the damping force to zero smoothly (Storey et al., 2008). The physical analysis of this method has been explained in section 4.8.3. The method produced vast improvements for semi-active controls in the numerical experiments, and was clearly the best method found for removing “crossover jerk”.

Evolutionary algorithms have been used in this thesis for the optimization of candidate suspension control algorithms, and also to test and compare the controls in a consistent environment. In this way evolutionary algorithms are being used as a first-order general design tool, comparing a range of radically different controls.

The use of evolutionary algorithms as a first-order design tool has strengths and weaknesses. A positive is the fact that evolutionary algorithms are robust, in the sense that the controls that survive evolution tend to cope well with minor environmental variations and deviations in control parameters, as described in section 2.3.2. Most importantly, evolutionary algorithms are quite capable of deriving useful results for problems that are intractable with analytical approaches. Balanced against this is the fact that evolutionary algorithms derive suboptimal results, in the sense defined in section 2.12.1.

In the numerical experiments performed here, a wide variety of controls, of varying degrees of sophistication, could be compared against the two benchmarks: the passive and the linear skyhook. These were tested over random road surfaces, containing a high proportion of very rough roads with bumps larger than half the rattlespace width. The relative performances of the various controls in the numerical experiments show some clear and compelling results, discussed above.
As for the evolutionary algorithms themselves, it was observed that if the evolution was too “hot” (with large numbers of large mutations) genes tend to exploit “easy advantages”. For example, controls that had jerk or acceleration limiting parameters tended to set those parameters low, easily allowing the control to be soft. This has the effect of masking higher performing parameter combinations. To avoid this, cooling must be done slowly.

If a particular road surface, or a particular set of road surfaces, was used for the modelling, the evolutionary algorithm might be biased towards controls that handle particular features of those surfaces. To avoid any source of bias, the road inputs used for the EAs in this thesis were randomized. However, randomizing road inputs slowed down the numerical evolution. This is because some roads were smoother than others and so in some cases, “worse” genes outperformed “better” ones. To compensate for this required a much longer period of evolution than might be the case with a particular road input set. In this author’s estimation this precaution of producing random input was probably not warranted. It may have sped up evolution considerably if a given large set of roads had been employed right throughout the evolutionary process, with the genes all competing under exactly the same conditions.

Overall, high performing suspension controls in the numerical experiments are deserving of further research. Given the inherent robustness of EAs and the fact that test road conditions in the numerical experiments were very rough, the controls that performed well in those experiments should be highly tolerant of parameter variation.

### 7.5. Further Investigation

This thesis aims at a first-order understanding of the factors involved in developing an electronic control for suspension systems. The thesis itself is intended as a basis for further research.

While the numerical experiments do not constitute formal proofs, they provide a useful guide to the relative superiority of any given control technique. For a control adapted for a given application, further investigations would be needed. This could entail either further computer modelling or physical prototyping.
The rattlespace constraint controls developed for the numerical experiments performed quite well, especially the semi-active variants. The theory for these controls came somewhat late in the research. The theory developed here provides a basic set of methods for targeting distance limits and anticipating possible future rattlespace movement. These controls have immediate application to control problems beyond suspension control, as outlined in section 4.7.3.

The question of how to approach a rattlespace limit that is undergoing stochastic movement has been discussed and simple algorithms have been investigated. Smoothed targets slightly inside the rattlespace edge allowed smoother and improved control in some of the experiments (see section 4.7.4). The problem of distinguishing potential large bumps from small corrugations needs further investigation: over-react to potential bumps and the suspension becomes uncomfortably hard, under-react and the suspension becomes dangerously soft. Simple heuristics applied to these problems produced highly performing controls. There is enormous scope for further research.

The minimum-time “skim” controls of section 4.8 go a long way to providing a toolbox of controls that allow fast bang-bang and bang-off-bang triple-integrator controls to be employed for moving a suspension smoothly between rattlespace limits. They provide a theory for controlling the rebound which contributes to instability. As shown in section 4.7.3.5 a control that produces a trajectory on the boundary of the state space for the distance constraint problem can be broken into sections that “skim” the time axis. Each section can be analysed using Pontryagin’s theorem unconstrained by distance.

As described in section 4.7.3.6 the landing-surface control can be implemented as a real-time optimal control. The control brings a system to rest without overshoot in minimum time and with minimum jerk. Physical experiments should be performed to investigate the effects of such controls on settling time.

Furthermore, the landing-surface control can be used as a part of a variety of possible controls that use it to handle the final, delicate “docking” movement, as discussed in section 4.7.3.7. Such methods can include set jerk limits, acceleration limits and even velocity limits. Further investigation into these possibilities is needed.
One control technique that requires further investigation is how to recognize different road types and respond accordingly. Thus a control could recognize highway conditions, for example, and become smoother, but they could stiffen over patches of rough terrain. The suspension could adjust to various road conditions: freeway, inner city streets, off road etc.

Look-ahead sensors also offer exciting possibilities, especially for the rattlespace constraint controls of section 4.7, since these can potentially target more certain trajectories.

The reasons for considering jerk to be a better measure of comfort than acceleration are examined in full in section 3.1. Empirical evidence to show that jerk is more “natural” rests on a few small experiments. Indeed, these experiments suggest that even higher order derivatives than jerk might be better indicators of comfort, especially for low frequencies, as explained in section 3.1. On the other hand, the empirical evidence for least squares acceleration as an appropriate comfort measure rests on a large experimental sample (refer to section 2.9.1), but it was performed well before the advent of electronically controlled suspension, and it did not even consider jerk as an alternative.

There is room for further investigation into the performance measures. All else being equal, it is clear that jerk provides a much better measure of smoothness and comfort than acceleration, and there are empirical experiments which support this claim. As explained in section 3.1, a jerk measure will also indicate high accelerations, but an acceleration measure can mask extremely rough acceleration “discontinuities”. However, when accelerations are extremely rough, as when discomfort is caused by resonance in body tissue (refer to section 2.9.1) the energy input into the body is extremely uncomfortable. At high frequencies, a mixture of jerk and acceleration may provide a more accurate measure of comfort. Further testing is required, but the arguments of section 3.1 indicate that, all else being equal, jerk is a better indicator of comfort than acceleration.

Tracking performance is a complex matter, and the discussion in section 3.2 raises a number of new issues requiring further investigation. Tracking involves a mixture of road holding, which also requires a smooth chassis movement and an associated smooth road normal force, as well as the capacity for the suspension to minimize damaging collisions with the rattlespace limits. There is a complex interplay between the avoidance of rattlespace edge collision, reducing damage, reducing high force and jerk, and improving comfort. And all
these are related to driver slow down. This thesis has raised a number of important issues for tracking measures, but there have also been practical suggestions for ways of proceeding with further investigations.

A simple, iterative numerical method for modelling driver slowdown has been proposed above in section 3.2, although it has not been used here. This method does not require psychological modelling and it may be somewhat simplistic, but it allows the modelling of a factor that could not be accounted for otherwise. It would be interesting to see if optimization using driver slowdown produces superior results.

Tracking is highly sensitive to the rate of occurrence of very large bumps (at least half the rattlespace height, as explained in section 3.2). Without such bumps in the EAs, optimization will settle on absurdly soft suspensions. The rate at which such bumps occur needs investigation. The models in this thesis have almost certainly erred on the side of having too high a rate of large bumps, to ensure that the suspensions do not produce extremely soft suspension. Even so, this author is of the opinion that road discontinuities and road sloped discontinuities form the major source of driver irritation for most “large bumps” and these bumps are rarely more than half the rattlespace height. Driver slow down is a factor here as well.

Generally speaking, the smoother the chassis trajectory, the smoother the road normal force. Also, with constant forward velocity the impact of road normal force on braking and acceleration is irrelevant. Thus road normal force considerations have been subsumed under the comfort and tracking measures used here. There are many ways however that a suspension control might alter its response to improve normal force smoothness under acceleration, braking and cornering. The effect of tyre resonance on road normal force also needs to be included in such an analysis.

As noted in section 4.1 the use of the center of the rattlespace as the suspension equilibrium position has been mathematically convenient and was felt to be sufficient for the high-level investigation of this thesis. However, heavily loaded vehicles generally will have a lower equilibrium position, and can cause great damage when bottoming. This matter requires further investigation.
Independent suspension has historically produced great benefits to suspension ride so there are ad hoc considerations that might lead us to suppose that independent electronic control will have similar benefits. Certainly, independent smooth jerk control of absolute chassis height at each corner will tend to reduce overall displacement including rotational movements: roll and yaw in cornering, and pitch during braking and acceleration. Nonetheless, investigation employing models with a larger number of degrees of freedom, including rotations, is needed.

One method of improving the responsiveness of a semi-active suspension might be to increase the crossover jerk limit near rattlespace limits, further increasing the subtlety of “virtual bump stop” control. This could be relatively easily carried out with the programs produced for the numerical experiments.

As discussed in section 7.2, results from evolutionary algorithms might have been derived much more quickly if random roads had not been used in each case. It is worth investigating the question of what is the minimal amount of data needed to avoid biasing results, as this could drive an enormous speed up in evolution.

It is hoped that the work of this thesis will contribute to future electronic suspension control and related control problems by providing a basic overview of control techniques. Some of the controls discussed above are simple modifications of basic linear systems that have a legacy in traditional mechanical engineering, while others are developed from theory that has been applied in aerospace engineering. It is anticipated that the more sophisticated “aerospace” techniques will have more and more application as transducers become cheaper and more accurate, and as microprocessors become faster, with more memory.
8. Appendix

The first section of the appendix discusses terms used throughout the thesis. In some cases alternative terms are available. The term “rattlespace” for example is also known as the “working space”. The term “jerk”, used for the rate of change of acceleration, is of central importance to the thesis and is defined in detail in the text and is not discussed here (refer to the discussion in section 3.1).

The remainder of the appendix covers some details of mathematical proofs, program code and physical experiment details that would distract from the explanations contained in the thesis.

8.1. Basic Nomenclature

There is a wide variation in the use of terms for the description of parts of a suspension system. For example, the amount of extension of the main spring and damper is known by a number of terms: stroke, extension, relative displacement, rattlespace, working space, etc. “Stroke” generally refers to displacement about equilibrium, which is equivalent to the term “relative displacement”. The term “rattlespace”, on the other hand, will generally refer to the range of the suspension travel.

“Relative velocity” is a commonly used term for rate-of-change of stroke (Ivers and Miller, 1991, p337; Ahmadian et al., 2004, p580). In this thesis the suspension travel limits will generally be referred to as the “rattlespace limit”, and the term “stroke” will be reserved as a convenient term for suspension displacement around the zero, equilibrium position.

The damper in a vehicle suspension system is commonly known as a “shock absorber”. But “the implication that shocks are absorbed is misleading.” (Dixon, 2008, p2) The term “damper” is used throughout the thesis.
8.1.1. Road and Chassis Height

Figure 8.1 illustrates the typical quarter-car model with two degrees of freedom.

![Figure 8.1 Schematic of the Two DOF Quarter-Car Model]

In this model the suspension and the tyre are both represented as spring and damper systems, but other models can be placed here such as hydraulic actuators in the case of active systems or more accurate tyre models (Lee et al., 2006, p7; Lot and Massaro, 2006; Lehtonen et al., 2006). If the tyre is not included in the model, then the unsprung mass is neglected and the road height, \( r \), becomes the height of the bottom of the suspension.

The unsprung mass is centred at the tyre hub and represents the mass of the wheel as well as other masses that move with the wheel, such as portions of the axle and wheel mounting. It has also been referred to as the “axle mass” (Ivers and Miller, 1991, p337).

In most models in the literature the various height measures, \( r \), \( u \) and \( y \), are centred about the static equilibrium position (Meirovitch, 1985, p97) which simplifies the equations. However, diagrams can often be made more intuitive if the chassis is offset from the road and shown in figure 8.2.
Figure 8.2 Graphs with the chassis above the road are more easily interpreted

Also road height is often modelled as a function of time, whereas road height is generally visualised as a function of distance along the road. Where the forward velocity of the car is constant the graphs are proportional. For the purposes of this thesis, road height is expressed as a function of time: $r(t)$, $u(t)$ and $y(t)$.

The stroke (or suspension extension) is the difference between the chassis height and the road, or the between the chassis and the “unsprung mass”, made up of the tyre and axle if this is included in the model. Thus the stroke is given as, $s(t) = y(t) - r(t)$, (using the variables of figure 2.1a). It is convenient mathematically to zero the stroke and represent vehicle height centred on zero, as in the second diagram below. More precisely, the road height and vehicle height, $x$ and $y$, are translated so that $x = 0$ and $y = r$ when the system is at equilibrium. This leads to diagrams in which the chassis crosses the road, which can be confusing for those who are not used to employing this technique.

Road heights are modelled here as a function of time. Since the vehicle might not be moving forward with a constant speed, the road-height profile as a function of time will be
correspondingly different from the road-height profile as a function of distance. This is also slightly unintuitive and is, again, simply a mathematical convenience. When presenting models for public demonstration it can be much more convincing to model the suspension as travelling at a height above the road and moving at a constant speed along the road.

8.1.2. Stroke, Rattlespace

The term “stroke” here refers to the extension of the sprung portion of the suspension. It is the distance between the chassis and the unsprung portion of the suspension (Redfield and Karnopp, 1988). Stroke will be symbolized by the symbol $s$, and it is equal to the difference between the equilibrium positions of the chassis height and the unsprung portion of the suspension. The distance of the compression and extension of the suspension and is given by the formula,

$$ s(t) = y(t) - u(t). $$

For the single DOF model the road forms the base of the spring and damper portion of the suspension,

$$ s(t) = y(t) - r(t). $$

The stroke velocity is sometimes referred to as the “relative velocity”, the “absolute velocity” being the vertical velocity of the body (Ivers and Miller, 1991, p337; Ahmadian et al., 2004, p580). Stroke velocity is also referred to as “rattle velocity” (Lauwerys et al., 2004, p1482). The term “stroke velocity” is used throughout this thesis. Just as the stroke velocity is sometimes called relative velocity, the stroke is sometimes referred to as the “relative displacement” (Wagner and Liu, 2000, p567).

The suspension travel has set limits and it cannot extend outside a finite range. The term, “rattlespace” (Burton, 1993; Hrovat and Hubbard, 1981; Hyvärinen, 2004; Takahashi et al., 2000), refers to the space within the limits of the vertical movement of the suspension, between the point of suspension compression where the tyre mounting collides with the car chassis, and the point at which it stretches and the tyre mass jolts down violently on the chassis. McLellan defines the rattlespace as “… the space provided between the axle and vehicle body for the suspension; this rattlespace must not be exceeded by compression of the
suspension or else very jarring impacts will occur” (1998, p27). Takahashi et al. refer to the “required rattlespace, i.e., the range of the relative body to axle displacement” (Takahashi et al., 2000). This is the sense of the term “rattlespace” as used in the thesis.

The term “working space” is also used (Sims and Stanway, 2003). Note that the suspension vertical travel limits are the true limits of travel before destruction of the suspension: if there are bump stops the true limits are at the very limit of the bump stop compression.

For basic mathematical models, it is convenient to assume that the zero, resting point of the suspension is half way between the rattlespace limits. Thus, if the resting point of the stroke is zero, the limits of the rattlespace can be conveniently represented algebraically as,

\[ -R \leq s(t) \leq R. \]

Here \( R \) is half the full rattlespace travel distance. This will not be the case in a real vehicle, and in fact the equilibrium position will change with the loading of the vehicle.

One of the objectives of a suspension system is to stay within the rattlespace (within the vertical travel limits). Hitting up hard against the rattlespace limits has been called “topping” and “bottoming” (Lord, 2006). Presumably topping occurs when the unsprung mass hits up hard against the sprung mass, and bottoming occurs when the damper reaches the “top” of its full extension.

The extremes of the rattlespace can also be modelled with a critically damped or overdamped extremely stiff spring and damper. This will be referred to as the “bump stop” (Lee et al., 2006, p8). This was used in some experiments with the stiffness set at a nominally high value.

**8.1.3. “Damping Coefficient” and “Damping Constant”**

If the D.E. of a general second-order system is,

\[ \ddot{y} + 2\xi\omega_n \dot{y} + \omega_n^2 y = \omega_n^2 u, \]

then the term, damping coefficient, is defined as,

\[ \alpha = \xi \omega_n. \]

(Distefano et al., 1997)
For a single degree-of-freedom suspension the equation of motion is,
\[ m\ddot{y} + c\dot{y} + ky = 0, \]
which gives,
\[ \alpha = \frac{c}{2m}. \]
The value \( c \) in this equation is sometimes known as the damping constant. With semi-active systems the damping rate is not in fact constant and so the term “damping rate” will be used throughout this thesis.

A number of other terms covering second-order systems are standard:
- \( \omega_n \) is the undamped natural frequency,
- \( \zeta \) is defined as the damping ratio,
- \( \omega_d = \omega_n \sqrt{1 - \zeta^2} \) is referred to as the damped natural frequency and,
- the time constant, \( \tau \), is the inverse of the damping coefficient.

### 8.1.4. Soft and Stiff Suspensions

The terms “soft”, “stiff” and “hard” in general indicate the level of force applied by a suspension. A stiff or hard suspension tends to apply a greater force than a soft one. In a passive system, a stiffer spring is one that has a larger spring constant, and a stiffer damper has a higher damping coefficient. Where the level of jerk (rate of change of acceleration) that can be applied indicates “stiffness”, of course the higher levels of jerk are characteristic of a stiff suspension.

Perhaps intuition has become attuned to thinking of “soft” suspensions in terms of passive components. A suspension that is soft in terms of acceleration or jerk may in fact need to increase damping force: a semi-active system under breaking may need to stiffen in order to stop the car from dipping over the front tyres. This stiffening reduces car-body acceleration and so is “softer” in terms of acceleration.

Even though there is this ambiguity in usage the terms stiff and soft are often useful as a descriptive aid. Of course the ambiguity must be resolved by context.
8.1.5. Sigmoid Function

A sigmoid function is an s-shaped function (see figure 8.3) that is often used in neural networks as a form of threshold function. It is used instead of a simple step function. It is here used in some EAs as a way to switch between one response and another.

![Sigmoid Function](image)

**Figure 8.3 Sigmoid Function**

The sigmoid function can be implemented in a number of ways. The following formula has been used,

\[
\frac{1}{1 + e^{-x}}.
\]

(Singh, 2001)

The hyperbolic tan function, tanh, is also used (Galkin, 2006). The sin function can also be used,

\[
\sigma(x) = \begin{cases} 
-1 & \text{if } x \leq -\pi / 2 \\
1 & \text{if } x \geq \pi / 2 \\
\sin(x) & \text{otherwise}
\end{cases}
\]

Clearly, each of these functions can be magnified and translated and evolvable parameters can be used to determine the parameters of the sigmoid functions.

8.1.6. Fitness Function

The measure of performance in various optimization problems is given a number of different names depending on context. In the use of evolutionary algorithms the measure of
performance is usually referred to as the objective function or the fitness function. The term objective function is also used mathematically to refer to measures of performance in variational calculus or mathematical programming. The terms performance index or cost function (Song et al., 2003) are also used, although the later term generally occurs in economic applications.

The cost function often takes the form of a functional, so the cost function can sometimes also be referred to as the cost functional (Camino et al., 1999; Takahashi et al., 2000).

8.2. Maple Derivation of LQR Coefficients

The Maple mathematical software package (Maple 7) was used to derive the LQR linear feedback control coefficients for the linear control problem using control over acceleration of section 2.3.1, and the similar problem using control over jerk of section 4.5.1. Both problems can be expressed in the form of the cost functional of equation 2.3 for the linear system described by the matrix equation of equation 2.4.

In the case of the single DOF control over acceleration (also known as the double-integrator) the following matrix values are substituted into equation 2.3 and equation 2.4:

\[
A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, Q = \begin{bmatrix} \alpha & 0 \\ 0 & 0 \end{bmatrix}, \text{ and } R = [1].
\]

The following Maple code will solve the Riccati equation,

```maple
restart;
with(plots):
with(linalg):
with(LinearAlgebra):

Warning, the name changecoords has been redefined
Warning, the protected names norm and trace have been redefined and unprotected
Warning, the assigned name GramSchmidt now has a global binding

> A:=<<0,0>|<1,0>>;
#<<a,d>|<b,e>|<c,f>>
B:=<<0,1>>;
Q:=<<alpha,0>|<0,0>>;
R:=<<1>>;
S:=<<a,e>|<e,b>>;
```
\[ A := \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \]
\[ B := \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]
\[ Q := \begin{bmatrix} \alpha & 0 \\ 0 & 0 \end{bmatrix} \]
\[ R := \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]
\[ S := \begin{bmatrix} a \\ \epsilon b \end{bmatrix} \]

\[ Z1 := \text{MatrixAdd}(\text{Multiply}(\text{Transpose}(A), S), \text{Multiply}(S, A)) \]
\[ Z2 := -\text{Multiply}(\text{Multiply}(\text{Multiply}(S, \text{Multiply}(B, \text{MatrixInverse}(R))), \text{Transpose}(B)), S) \]
\[ Z := \text{MatrixAdd}(\text{MatrixAdd}(Z1, Z2), Q) \]

\[ Z1 := \begin{bmatrix} 0 & a \\ a & 2 \epsilon \end{bmatrix} \]
\[ Z2 := \begin{bmatrix} -\epsilon^2 & -\epsilon \beta \\ -\epsilon \beta & -\beta^2 \end{bmatrix} \]
\[ Z := \begin{bmatrix} -\epsilon^2 + \alpha & a - \epsilon \beta \\ a - \epsilon \beta & 2 \epsilon - \beta^2 \end{bmatrix} \]

\[ #e := \sqrt{\beta}; \]
\[ Z; \]
\[ e := b*b/2; \]
\[ Z; \]
\[ e := 1/2 b^2 \]
\[ Z := \begin{bmatrix} 1/4 b^4 + \alpha & a - 1/2 b^3 \\ a - 1/2 b^3 & 0 \end{bmatrix} \]

\[ b := \sqrt{2} \alpha^{(1/4)} \]

\[ Z := \text{MatrixAdd}(\text{MatrixAdd}(Z1, Z2), Q) \]

This gives the coefficients as shown in equation 2.5. (This can be found in the file “Maple Experiments\LQR\LQR acc.mws”)
In the case of control over jerk (the triple-integrator) the following matrix values are used,

\[
A = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix},
B = \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix},
Q = \begin{bmatrix}
\alpha & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}, \text{ and } R = [1].
\]

The following Maple code (which can be found in the file “Maple Experiments\LQR\LQR jerk.mws”) solves the Riccati equations,

```maple
restart:
with(plots):
with(linalg):
with(LinearAlgebra):

A := <<0, 0, 0>|<1, 0, 0>|<0, 1, 0>>;
#<<a,d>|<b,e>|<c,f>>
B := <<0, 0, 1>>;
Q := <<\alpha, 0, 0>|<0, 0, 0>|<0, 0, 0>>;
R := <<<1>>;
S := <<a, d, e>|<d, b, f>|<e, f, c>>;

Z1 := MatrixAdd(Multiply(Transpose(A), S), Multiply(S, A));
Z2 :=-
Multiply(Multiply(Multiply(S, Multiply(B, MatrixInverse(R))), Transpose(B)), S);
Z := MatrixAdd(MatrixAdd(Z1, Z2), Q);
```

360
\[
Z1 := \begin{bmatrix} 0 & a & d \\ a & 2d & e + b \\ d & e + b & 2f \end{bmatrix}
\]

\[
Z2 := \begin{bmatrix} -e^2 & -ef & -ec \\ -ef & -f^2 & -fc \\ -ec & -fc & -c^2 \end{bmatrix}
\]

\[
Z := \begin{bmatrix} -e^2 + \alpha & a - ef & d - ec \\ a - ef & 2d - f^2 & e + b - fc \\ d - ec & e + b - fc & 2f - c^2 \end{bmatrix}
\]

\[
f := \frac{c^2}{2}
\]

\[
Z := \begin{bmatrix} \frac{e}{2} & \frac{e}{2} c^2 & d - ec \\ \frac{e}{2} c^2 & 2d - \frac{1}{4} c^4 & e + b - \frac{1}{2} c^3 \\ d - ec & e + b - \frac{1}{2} c^3 & 0 \end{bmatrix}
\]

\[
d := \frac{1}{8} c^4
\]

\[
ez := \begin{bmatrix} \frac{1}{8} c^3 & \frac{1}{8} c^4 - ec \\ \frac{1}{8} c^4 - ec & e + b - \frac{1}{2} c^3 \\ \frac{1}{8} c^4 - ec & e + b - \frac{1}{2} c^3 & 0 \end{bmatrix}
\]

\[
e := \frac{1}{8} c^3
\]

\[
F := \text{Multiply}(\text{Transpose}(B), S)
\]

\[
F := \begin{bmatrix} \frac{1}{8} c^3 & -\frac{1}{2} c^2 & -c \end{bmatrix}
\]
This then is the derivation of the coefficients of equation 4.4.

### 8.3. Maple Derivation using Euler-Lagrange

The following code uses the Euler-Lagrange equations to find the coefficients of the linear method that optimizes the quadratic cost function for control over acceleration. The meaning of the code should be clear from explanations in section 2.3.1. (This following code can be found in the file PhD\Maple Experiments\DESolution\Acceleration01.mws).

Find the solution to the Euler-Lagrange equation for the problem where
J = integral (d2 y^2 + y''^2) from zero to infinity

The E-L equation becomes
fy + d2^2 fy''/(dx)^2 = 0

i.e
2 d2 y + 2 y(4)=0

The characteristic equation is
v^4 = d2

Thus we have solutions
y(t)=A exp(-bt)sin bt + B exp(-bt) cos bt

where b is the fourth root of d2 on root 2. In the evolutionary algorithm b can be the evolved parameter.)

We solve:
y(0)=d, y'(0)=v and find y''(0)

> restart;
> with(plots):

Warning, the name changecoords has been redefined
## Define y
\[ y := A \exp(-b t) \sin(b t) + B \exp(-b t) \cos(b t) \]
\[ dy := \frac{d}{dt} \left( A \exp(-b t) \sin(b t) + B \exp(-b t) \cos(b t) \right) \]

## Solve for initial conditions given \( y(0) = d \) and \( y'(0) = v \)
\[
t := 0;
\]
\[ y; \]
\[ dy; \]
\[ \text{solve}\{y = d, dy = v\}, \{A, B\}; \]
\[ \text{unassign}('t'); \]

\[
\begin{align*}
  t & := 0 \\
  B & \\
  A b - B b \\
  \left[ B = d, A = \frac{d b + v}{b} \right]
\end{align*}
\]

## Set these values for A and B and find \( y''(0) \)
\[
A := (d b + v)/b; \quad B := d;
\]
\[ d2y := \frac{d}{dt} (dy); \]
\[
\begin{align*}
  t & := 0 \\
  d & \\
  v & \\
  -2 (d b + v) b
\end{align*}
\]
\[ \text{simplify}\{d2y + 2 b^2 y + 2 b dy\} ;
\]

\[
0
\]
8.4. Transmissibility for Passive and Skyhook Suspensions

The following Maple code derives the transmissibility for the purely linear passive and the skyhook suspensions. (This code can be found in the file PhD\Maple Experiments\Passive Skyhook Transmissibility.mws.)

Transmissibility for passive and skyhook
> ################################# Initialize #################################
restart:
with(plots):

with(DEtools):
Warning, the name changecoords has been redefined

> # Passive single DOF system
P:=2*zeta*(w);
Q1:=1-(w)^2;
TP:=simplify(sqrt((1+P^2)/(Q1^2+P^2)));
>
\[
\begin{align*}
P &:= 2 \zeta w \\
Q1 &:= 1 - w^2 \\
TP &:= \sqrt{\frac{1 + 4 \zeta^2 w^2}{1 - 2 w^2 + w^4 + 4 \zeta^2 w^2}}
\end{align*}
\]

> # Skyhook single DOF system
P:=2*zeta*(w);
Q1:=1-(w)^2;
TA:=simplify(sqrt((1)/(Q1^2+P^2)));
>
\[
\begin{align*}
P &:= 2 \zeta w \\
Q1 &:= 1 - w^2 \\
TA &:= \sqrt{\frac{1}{1 - 2 w^2 + w^4 + 4 \zeta^2 w^2}}
\end{align*}
\]

[10.96978202, -5.992819231 ]

> zeta:=0.2;
Passive01:=plot(TP, w=0..5, color=black):
zeta:=0.6;
Passive02:=plot(TP, w=0..5, color=black):
zeta:=1.0;
Passive03:=plot(TP, w=0..5, color=black):
zeta:=0.4;
Passive04:=plot(TP, w=0..5, color=black):
display(Passive01, Passive02, Passive03, Passive04);
\[
\begin{align*}
\zeta &:= .2 \\
\zeta &:= .6 \\
\zeta &:= 1.0 \\
\zeta &:= .4
\end{align*}
\]
> zeta:=0.2;
Active01:=plot(TA, w=0.01..5, color=black):
zeta:=0.6;
Active02:=plot(TA, w=0..5, color=black):
zeta:=1.0;
Active03:=plot(TA, w=0..5, color=black):
zeta:=0.4;
Active04:=plot(TA, w=0..5, color=black):
#zeta:=0.7071;
#Active05:=plot(TA, w=0..5, color=red):
display(Active01, Active02, Active03, Active04);
unassign('zeta','w');
D2:=simplify(diff(TA,w));
solve(D2=0,w);

# two-state feedback on body acceleration
# (Savaresi et al., 2003, 2266)
TAcc:=simplify((alpha-w^2)/(1-w^2));
alpha:=1;
ActiveAcc01:=plot(TAcc, w=0.01..5, color=black):
display(ActiveAcc01, Active02, Active03, Active04);

> alpha:=1;

\[ D2 := -2 \frac{w (-1 + w^2 + 2 \zeta^2)}{\sqrt{1 - 2 w^2 + w^4 + 4 \zeta^2 w^2 (1 - 2 w^2 + w^4 + 4 \zeta^2 w^2)^2}} \]
\[ 0, \sqrt{1 - 2 \zeta^2}, -\sqrt{1 - 2 \zeta^2} \]

\[ TAcc := -\frac{\alpha - w^2}{-1 + w^2} \]
8.5. RMS Acceleration in “Elevator Example”

Refer to figure 3.2. The following analysis compares two methods, each covering the same distance, \( D \), over the same time period, \( T \). The algebra is greatly simplified by the symmetry of the examples. The bang-bang acceleration control, of figure (a), switches at time \( T/2 \). The bang-bang jerk control switches at \( T/4 \) and \( 3T/4 \). The distance covered by the acceleration control, using an acceleration of \( A \) is,

\[
D = 2 \times \frac{1}{2} A \left( \frac{T}{2} \right)^2 = \frac{AT^2}{4}.
\]

The distance covered by the jerk control, using a jerk of \( J \) is,

\[
D = 2 \left[ \frac{1}{6} J \left( \frac{T}{4} \right)^3 + \frac{1}{2} J \left( \frac{T}{4} \right)^2 \times \frac{T}{4} + \frac{1}{4} J \left( \frac{T}{4} \right)^2 - \frac{1}{6} J \left( \frac{T}{4} \right)^3 \right] = \frac{JT^3}{32}.
\]

Equating these distances and solving produces,
\[JT = 8A.\]

**Equation 8.1**

Now the RMS acceleration from the bang-bang control over acceleration is simply \(A\). The bang-bang control over jerk has an RMS acceleration of,

\[
\sqrt[4]{\frac{4}{T} \int_{0}^{T} (Jt)^2 \, dt} = \frac{JT}{4 \sqrt{3}}.
\]

Using equation 8.1, the RMS acceleration for the jerk method is,

\[
\frac{JT}{4 \sqrt{3}} = \frac{8A}{4 \sqrt{3}} = \frac{2A}{\sqrt{3}} \approx 1.1547A.
\]

Thus there is an RMS acceleration difference of only about 15 %, while the acceleration control has three infinite jerk spikes.

### 8.6. LQ Problem over Jerk with Finite Time

In section 4.5.1 it is claimed that the LQ problem over finite time requires a control that is quadratic with respect to time. The problem is similar to the analogous problem of control over acceleration as presented in Ross (2009, p42). The problem can be stated as:

Minimize the \(L^2\) norm for control using,

\[
J = \int_{0}^{T} \frac{1}{2} u^2(t) \, dt,
\]

where \(u\) represents control over jerk.

The equations of motion are,

\[
\begin{bmatrix}
\dot{x}(t) \\
v(t) \\
a(t) \\
u(t)
\end{bmatrix} = \begin{bmatrix}
v(t) \\
a(t) \\
u(t)
\end{bmatrix} = f(x, u),
\]

where \(x, v\) and \(a\) represent displacement velocity and acceleration respectively. The control Hamiltonian as in Ross (2009, p43) is given as,

\[
H = \frac{1}{2} u^2 + \lambda_x f = \frac{1}{2} u^2 + \lambda_x v + \lambda_a a + \lambda_u u.
\]
Applying Pontryagin’s principle this must be minimized with respect to the control, \( u \), and since the control is constrained, the formula \( \frac{\partial H}{\partial u} = 0 \) can be applied, giving,

\[
u = -\dot{\lambda}_u.\]

**Equation 8.2**

The costate equations,

\[
-\dot{\lambda} = \frac{\partial H}{\partial x},
\]

produce the DEs,

\[
\dot{\lambda}_x = 0, \quad \dot{\lambda}_v = -\lambda_x, \quad \dot{\lambda}_u = -\lambda_v.
\]

Solving these gives,

\[
\lambda_u = \frac{1}{2} c_1 t^2 + c_2 t + c_3,
\]

for constants \( c_1, c_2 \) and \( c_3 \). Together with equation 8.2 this produces the desired result.

3-D Plot of Landing Surface

Matlab was used for the generation of the three-dimensional plot of the landing surface. The code calls a function, `distToSwitchingPlane`, which is essentially the same as the function shown in section 8.8, except with Matlab syntax. The MATLAB code to generate the graph is shown directly below.

```matlab
% Set up the parameters
clear
jerk=2;
accLim=5;
velLim=8;
umPts=60;

VEL=linspace(-velLim,velLim,numPts);
ACC=linspace(-accLim,accLim,numPts);
% Set up the mesh
[v,y]=meshgrid(VEL,ACC);

% calculate the points
z=eye(numPts);
for c=1:numPts
    for j=1:numPts
        vel=VEL(c); acc=ACC(j);
        z(j,c)=-distToSwitchingPlane( 0,vel,acc,jerk );
        if z(j,c)<0
            z(j,c)=z(j,c);
        end
    end
end
```
In order to make the zero level obvious, the positive values have been offset in figure 8.4 below. Note the region between the final switching curve and the plane where distance equals zero.

Figure 8.4  3-D Plot with Offset at Zero
8.7. *Alternative Proof for Minimum-Time Control*

The method for the proof presented here is the same as that used in Hermes and LaSalle section 13 (1969). The method first translates the problem into a simpler version involving an expanding symmetrical, convex and compact subset. Firstly the method is applied to the problem of control using acceleration.

Let the control be $u$ and the distance be $x$. Control over acceleration is expressed as,

$$u = \dot{x},$$

and in matrix notation as,

$$\mathbf{x} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ u \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ u \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \mathbf{A}x + \mathbf{Bu}.$$

The following matrices can be defined,

$$y = \dot{x}, \quad \mathbf{A} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$  

Since $\mathbf{A}^2 = 0$,

$$\mathbf{X} = e^{\mathbf{A}t} = I + \mathbf{At} = \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix},$$

and the inverse is given as,

$$\mathbf{X}^{-1} = e^{-\mathbf{A}t} = I - \mathbf{At} = \begin{pmatrix} 1 & -t \\ 0 & 1 \end{pmatrix}.$$  

This is translated into the simpler problem of finding a control,

$$\dot{y} = Y(t)u(t),$$

using the same $u$ as for the non-translated problem where,

$$\mathbf{Y} = \mathbf{X}^{-1} \mathbf{B} = \begin{pmatrix} 1 & -t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -t \\ 1 \end{pmatrix}.$$  

The time-optimal control is of the form,
\[ u = \text{sgn}(\eta^T Y) = \text{sgn}\left( \begin{pmatrix} n_1 & n_2 \end{pmatrix} \begin{pmatrix} -t \\ 1 \end{pmatrix} \right) = \text{sgn}(n_2 - n_1 t). \]

**Equation 8.3**

where,
\[
\eta = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix},
\]

is constant and the sgn function is such that sgn is positive one when the argument is positive and minus one when the argument is negative. Without loss of generality, the control is constrained to lie between plus or minus one. From equation 8.3 it is readily seen that there is at most one switch of control since the argument, \( n_2 - n_1 t \), can only cross zero once.

The case of control over jerk is then almost exactly the same as the proof for the case of control over acceleration above. The control has the form, \( u = \ddot{x} \),

which in matrix notation is represented as,
\[
\dot{x} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ u \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ u \end{pmatrix} = A \dot{x} + Bu.
\]

This employs the following definitions,
\[
y = \dot{x}, \quad z = \ddot{y}, \quad A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \text{ and } B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.
\]

Because \( A^3 = 0 \), the exponential is simplified giving,
\[
X = e^{At} = I + At + \frac{1}{2} At^2 = \begin{pmatrix} 1 & t & \frac{1}{2} t^2 \\ 0 & 1 & t \\ 0 & 0 & 1 \end{pmatrix},
\]

and the inverse is given as,
\[
X^{-1} = e^{-At} = I - At + \frac{1}{2} At^2 = \begin{pmatrix} 1 & -t & \frac{1}{2} t^2 \\ 0 & 1 & -t \\ 0 & 0 & 1 \end{pmatrix}.
\]

This can be translated into the simpler problem of finding a control,
\[ \dot{y} = Y(t)u(t), \]

using the same \( u \) as for the non-translated problem where,

\[
Y = X^{-1}B = \begin{pmatrix} 1 & -t & \frac{1}{2}t^2 \\ 0 & 1 & -t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{2}t^2 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2}t^2 \\ -t \\ 1 \end{pmatrix}.
\]

The time-optimal control is of the form,

\[
u = \text{sgn}(\eta^T Y) = \text{sgn} \left( \begin{pmatrix} n_1 & n_2 & n_3 \end{pmatrix} \begin{pmatrix} \frac{1}{2}t^2 \\ -t \\ 1 \end{pmatrix} \right) = \text{sgn} \left( n_3 - n_2t + \frac{1}{2}n_1t^2 \right).
\]

Equation 8.4

where,

\[
\eta = \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix}.
\]

Since the argument of the \text{sgn} function in equation 8.4 is a quadratic it can cross zero at most twice and so there are again at most two switches in the control. The rest of the proof is the same.

### 8.8. Java Code for Iterative Minimum-Time Algorithm

A test program was developed that would run the switching algorithm and provide verification of its veracity. The code is embedded in a test program that the author wrote for experimenting with various algorithms altering the initial conditions. A screen shot of the program running is shown below, in figure 8.5. The “chassis” parameters are used to set different initial values of displacement, \( d \), velocity, \( v \), and acceleration, \( a \).
The following code is used to produce the plots for acceleration, velocity and distance.

The Overshoot package in the Eclipse workspace in

PhD\Eclipse RSpace Constraint\Edge Overshoot

contains the class SwitchingPlaneX.java. This class contains many of the methods discussed below. The following is the subroutine that calculates the distance travelled:

```java
public static double distToSwitchingPlane(double d, double v, double a, double jerk)
{
    double dt=d, vt=v, at=a;
    JPlane1=jerk;
    //** is the first jerk positive or negative?
    int phase=0, prevPhase=-1;
    if (a>=0)
    {
        if ((v+a*a/(2.0*jerk))>=0) JPlane1=-JPlane1;
    }
    else{
        if ((v-a*a/(2.0*jerk))>=0) JPlane1=-JPlane1;
    }
    //** velocity at intersection with y''=0
    double vZ=v-a*a/(2.0*JPlane1);
    double tZ=-a/JPlane1;
    //** tH is the time of final switch
```
This program can be demonstrated by running the OvershootDemo.java in the OvershootControl Package. Press buttons “Min-Time Jerk” and “Cts. Min-Time Jerk”. The first runs the method using precalculated parameters. The second runs the method iteratively, as discussed above in section 4.6.1. The code for drawing the graphs are contained in the methods processJerkTest() and processcontinuousJerkTest().

### 8.9. Java Code for Time to Landing Surface

As discussed in section 4.6.3, the iterative algorithm, while simple, does not compute the trajectory of the minimum-time response. In order to find the maximum and minimum distance travelled, it is necessary to first find the time taken to reach the landing surface. The Java code for this algorithm is shown below. This code can be found in the Overshoot package in the Eclipse workspace in PhD\Eclipse RSpace Constraint\Edge Overshoot in the PhD directory. The class SwitchingPlaneX.java contains the code for the method timeToSwitchingPlane() shown immediately below, which in turns calls the method distToSwitchingPlane() given above in section 8.8.

```java
class OvershootControl {
    private static double V_TOLERANCE=1E-6;
    private static double timeToSwitchingPlane(double d, double v, double a, double jerk) {
        // Velocity here is half velocity when y''=0
        vH=vZ/2;
        tH=tZ+Math.sqrt(-vZ/JPlane1);
        aH=JPlane1*(tH-tZ);
        double tSquared = tH*tH;
        dH=d+v*tH+a*tSquared*0.5+JPlane1*tSquared*tH/6;
        tEnd = tH+aH/JPlane1;
        double tHEnd = tEnd-tH;
        double tHEnd2=tHEnd*tHEnd;
        dEnd = dH + vH*tHEnd + aH*tHEnd2*0.5 - JPlane1*tHEnd2*tHEnd/6;
        if (dEnd<0) {
            JOut=jerk;
        } else {
            JOut=-jerk;
        }
        return dEnd;
    }
}
```
throws SuspensionTestException{
    double dEnd=distToSwitchingPlane(d,v,a,jerk);
    double time=0;
    if (Math.abs(dEnd)>DIST_TOLERANCE) {
        double dEndFirst=dEnd;
        double J=JOut;
        // **** Keep doubling until different
        double tLo=0, t=1;
        boolean endLoop=false;
        int count=0;
        do {
            double dt=d+v*t+a*t*t/2.0+J*t*t*t/6.0;
            double vt=v+a*t+J*t*t/2.0;
            double at=a+J*t;
            dEnd=distToSwitchingPlane(dt,vt,at,jerk);
            endLoop=(dEnd>=0 && dEndFirst<0)||(dEnd<=0 &&
            dEndFirst>0);
            if (!endLoop) {
                tLo=t;
                t=2*t;
            }
            count++;
            if (count>MAX_COUNT) throw new
                SuspensionTestException("count in
timeToSwitch()="+count+" MAX_COUNT="+MAX_COUNT+"\ninRattlespaceCollisionXXX.timeToSwitchingPlane()");
            System.out.println("t="+t+" dEnd="+dEnd+"
                dEndFirst="+dEndFirst);
        } while (!endLoop);
        time=t;
        // Now tLo<t and on either side of switching plane
        // **** Keep dissecting till tolerance
        double tHi=t;
        do {
            t=(tLo+tHi)*0.5;
            //System.out.println("t="+t);
            double dt=d+v*t+a*t*t/2.0+J*t*t*t/6.0;
            double vt=v+a*t+J*t*t/2.0;
            double at=a+J*t;
            dEnd=distToSwitchingPlane(dt,vt,at,jerk);
            if ((dEnd>=0 && dEndFirst<0)||(dEnd<=0 &&
            dEndFirst>0)){
                // ** t is too big
                tHi=t;
            }else{
                // ** t is too low
                tLo=t;
            }
            // Maybe loop until these are the same - or
            contradict end conditions
            // because of numerical inaccuracies
            //System.out.println("t="+t+" dEnd="+dEnd+"
        tLo="+tLo+" tHi="+tHi+" tHi-tLo="+(tHi-tLo));
        } while (tHi-tLo>=TIME_TOLERANCE);
        time=tLo;
8.10. Maple Derivation of Jerk for Distance Constraint

The following is the Maple code (contained in submitted files, PhD/Maple Experiments/This Side/ConstantJerkExponentialDecay.mws) used to derive the value for jerk that will result in velocity, distance and acceleration satisfying equation 4.14, which allows the system to decay exponentially to rest with zero distance, velocity and acceleration without overshoot. Maple 7 was the version of the algebraic software used.

The important point to note is that the result is found by solving for jerk, $j$, where distance, velocity and acceleration ($y, y_1$ and $y_2$) satisfy the relation,

$$\frac{y_1^2}{y_2 y} = 1.$$ 

At the end of time $t$, exponential decay can be applied.

```maple
> restart;
with(plots):

Constant jerk is applied for a given time.

> # Main Function
y:=d+v*t+ a*t^2/2+j*t^3/6;
y1:=simplify(diff(y,t)); # velocity
y2:=simplify(diff(y1,t)); # acceleration
y3:=simplify(diff(y2,t)); # jerk
# Test - should give d, v, a, j
t:=0;
y; y1; y2; y3;
unassign('t');
> # Example Plot
d:=4: v:=-3: a:=-1: j:=0:
plotY:=plot(y, t=0..5, color=black):
plotY1:=plot(y1, t=0..5, color=blue):
plotY2:=plot(y2, t=0..5, color=red):
plotSurd:=plot(y1^2-y*y2, t=0..3, color=yellow):
display(plotY,plotY1,plotY2, plotSurd);
```
unassign('d', 'a', 'v', 'j');
y := d + v t + \frac{1}{2} a t^2 + \frac{1}{6} j t^3

y1 := v + a t + \frac{1}{2} j t^2

y2 := a + j t

y3 := j
t := 0
d
v
a
j

> # Find T when v^2/d=a
Eq2 := simplify(y1^2 - y*y2);
Eq3 := solve(Eq2, j);
simplify(Eq3[1]);
simplify(Eq3[2]);

\[
\begin{align*}
\text{Eq2} &:= v^2 + v a t + \frac{1}{2} a^2 t^2 + \frac{1}{3} a^3 j + \frac{1}{12} j^2 t^4 - d a - d j t \\
\text{Eq3} &:= \frac{1}{2} -4 a t^2 + 12 d + 2 \sqrt{-2 a^2 t^4 - 12 a t^2 d + 36 d^2 - 12 t^2 v^2 - 12 t^3 v a}, \\
\text{Eq3} &:= \frac{1}{2} -4 a t^2 + 12 d + 2 \sqrt{-2 a^2 t^4 - 12 a t^2 d + 36 d^2 - 12 t^2 v^2 - 12 t^3 v a} \\
&= \frac{2 a t^2 - 6 d - \sqrt{-2 a^2 t^4 - 12 a t^2 d + 36 d^2 - 12 t^2 v^2 - 12 t^3 v a}}{t^3}
\end{align*}
\]
The Maple code shown below (contained in submitted files, PhD/Maple Experiments/This Side/ConstantJerkConstantJerkDecay.mws) finds the value for initial jerk that will result in velocity, distance and acceleration satisfying equation 4.16. Once this is satisfied, constant jerk can be used to bring the system to rest with zero distance, velocity and acceleration.

Again, the equations are solved for jerk, \( j \), where distance, velocity and acceleration (\( y \), \( \dot{y} \), and \( \ddot{y} \)) satisfy the relation,

\[
\frac{\dot{y}^2}{\ddot{y}y} = \frac{3}{2}.
\]

At the end of time \( t \), constant jerk decay can be applied.

```maple
> restart;
with(plots):
Warning, the name changecoords has been redefined

Constant jerk is applied for a given time.
Warning, the name changecoords has been redefined

> # Main Function
y:=d+v*t+ a*t^2/2+j*t^3/6;
y1:=simplify(diff(y,t)); # velocity
y2:=simplify(diff(y1,t)); # acceleration
y3:=simplify(diff(y2,t)); # jerk
# Test - should give d, v, a, j
t:=0;
y; y1; y2; y3;
unassign('t');
> # Example Plot
d:=4: v:=-3: a:=-1: j:=0:
plotY:=plot(y, t=0..5, color=black):
plotY1:=plot(y1, t=0..5, color=blue):
plotY2:=plot(y2, t=0..5, color=red):
plotSurd:=plot(y1^2-y*y2, t=0..3, color=yellow):
display(plotY,plotY1,plotY2, plotSurd);
unassign('d','a','v', 'j');
y := d + v t + \frac{1}{2} a t^2 + \frac{1}{6} j t^3
y1 := v + a t + \frac{1}{2} j t^2
y2 := a + j t
```
The java code used to find these values is given below. This can be found in the RattlespaceCollisionX.java class definition in the JerkControl package. The main program that runs and calls this code is JerkDemo.java in the same package. (The code for the second method below is also used in the EdgeTargetAncestorX class in the Overshoot package.)
 Outputs the jerk needed to reach target $E=v^2/(ad)$ in time input, $t$. Outputs zero if no jerk needed.

---

These formulas are based on calculations in Maple

//**** Exponential decay
public double findJTillAccOKExp(double d, double v, double a, double t){
    double J=0;
    double t2=t*t, t3=t2*t, t4=t3*t;
    double surd=-2*a*a*t4-12*a*t2*d+36*d*d-12*t2*v*v-12*t3*v*a;
    //System.out.println("surd="+surd);
    double temp=-1;
    if (surd>=0){
        surd=Math.sqrt(surd);
        double j1=(-2*a*t2+6*d+surd)/t3;
        double j2=-(2*a*t2-6*d+surd)/t3;
        //if (j1>j2){ double temp1=j1; j1=j2; j2=temp1;}
        //System.out.println("j1="+j1+"  j2="+j2);
        if (d>0){
            // find least J>0
            if (j1>0) J=j1;
            if ((j2>0) && (j2<J)) J=j2;
        }
    }
    else{
        // find greatest J<0
        if (j1<0) J=j1;
        if ((j2<0) && (j2>J)) J=j2;
    }
    return J;
}

//***** Find the jerk in the first stage for constant jerk decay in second
public double findJerk1ForConstantJerkDecay(double d, double v, double a, double t){
    double t2=t*t, t3=t2*t, t4=t3*t;
    double J=0.5*(4*v*v+2*v*a*t+a*a*t2-6*d*a)/(t*(v*t+3*d));
    //System.out.println("surd="+surd);
    return J;
}

8.11. Java Code for Distance without Overshoot Method

The following is the java code that implements the “landing-plane” method discussed in section 4.7.3. The code finds the time for the first stage that gives an equal jerk in the
opposite direction for the second stage. These methods can be found in the

EdgeTargetAncestorXX class in the Overshoot package.

```java
/********************
"Landing-plane" method, uses 2 equal jerks
This method varies timing to find equal jerk,
rather than varying jerk to put the system on the landing plane
********************/
public double T, jerk1, jerk2, TBack; // output values
public double ReverseJerkFactor=1;
public boolean isEqualJerkTime=false;
private static double TOLERANCE=1E-6, TTop_MAX=100, a_MIN=3E-5;
private static int MAX_NUM_LOOPS = 35;

private boolean isLowTime(double d, double v, double a, double T1){
    //*** Take care of the case that a==0
    // find jerk
    T=T1;
    // Find the jerk in the first stage for constant jerk decay in
    second
    jerk1=findJerk1ForConstantJerkDecay(d,v,a,T);
    // find second stage values
    findSecondStageValues(T, d, v, a, jerk1);
    jerk2=jBack;
    // compare and output
    //System.out.println("T=\"+T+\"  jerk1=\"+jerk1+\"  jerk2=\"+jerk2+
    TBack="+TBack);
    if (d>0){
        return ReverseJerkFactor*jerk1>-jerk2;
    }else{
        return ReverseJerkFactor*jerk1<-jerk2;
    }
}

public void findParamsForTwoEqualJerksJerkCalc(double d, double v, double a) throws SuspensionTestException{
    // The algorithm is stable if n a==0 but not if a is too small
    if (Math.abs(a)<a_MIN) a=0;
    T=0; jerk1=0; jerk2=0; TBack=0;
    //*** Try to find the larger T
    // Find the turning point for jerk1
    double surd=4*v*v-6*a*d;
    isEqualJerkTime = (surd>=0 && d!=0);
    //System.out.println("surd=\""+surd);
    double TTop=0;
    if (isEqualJerkTime){
        //*** Try to find an upper T
        surd=Math.sqrt(surd);
        if (a==0){
            TTop=-1;
            if ((d>0 && v<0)||(d<0 && v>0)){
                TTop=-d/v;
            }
            //System.out.println("a=0 TTop=\"+TTop);
        }else{
            TTop=(-2*v+surd)/a;
            double t2=(-2*v-surd)/a;
            //System.out.println("TTop=\"+TTop+\" t2=\"+t2;
        }
    }
```
if (t2>0) {
    if (TTop<=0) {
        TTop=t2;
    } else {
        if (t2<TTop) {
            TTop=t2;
        }
    }
    //System.out.println("TTop="+TTop);
}

if (TTop<=0) {
    // No value of T to start with found
    T=-1; jerk1=0; TBack=-1; jerk2=0;
} else {
    //**** Halve until a candidate for TBot is found
    // There are stability problems if TTop is too
    large
    // especially if a is very low
    if (TTop>TTop_MAX) TTop=TTop_MAX;
    double TBot=TTop*0.5;
    int count =0;
    while (!isLowTime(d,v,a,TBot)) {
        count++;
        if (count>MAX_NUM_LOOPS) throw new SuspensionTestException("Looping too many times\nTTop="+TTop+" TBot="+TBot+" nd="+d+" v="+v+" a="+a+
"
EdgeTargetAncestorX.findParamsForTwoEqualJerksJerkCalc() - divide
by 2");
        TTop=TBot;
        TBot=TBot*0.5;
        //System.out.println("TTop="+TTop);
    }
    //System.out.println("TTop="+TTop+" TBot="+TBot);
    //**** Keep using averages until tolerance
difference is reached
    count =0;
    while (Math.abs(TTop-TBot)>TOLERANCE) {
        count++;
        if (count>MAX_NUM_LOOPS) throw new SuspensionTestException("Looping too many
times\nEdgeTargetAncestorX.findParamsForTwoEqualJerksJerkCalc()\n"+
"d="+d+" v="+v+" a="+a+"\nTTop="+TTop+" TBot="+TBot+" TTop-TBot="+(TTop-TBot)+" finding average");
        double TMid=(TBot+TTop)*0.5;
        if (isLowTime(d,v,a,TMid)) {
            TBot=TMid;
        } else {
            TTop=TMid;
        }
    }
}
}
This code calls the following two methods. The first method simply finds jerk using equation 4.18.

```java
//***** Find the jerk in the first stage for constant jerk decay in second
public double findJerk1ForConstantJerkDecay(double d, double v, double a, double t){
    double t2=t*t, t3=t2*t, t4=t3*t;
    double J=0.5*(4*v*v+2*v*a*t+a*a*t2-6*d*a)/(t*(v*t+3*d));
    //System.out.println("surd="+surd);
    return J;
}
```

The following method calculates the time and jerk used in the second stage. This is easily calculated since the jerk applied is known.

```java
/*********************/
For the constant jerk, E=3/2 method,
find back jerk and back T, TBack and jBack,
given forward T and initial conditions
*********************/
public double jBack=-1;
public double dTJerk=0, vTJerk=0, aTJerk=0;
public void findSecondStageValues(double T, double d, double v, double a, double j){
    TBack=-1; jBack=0;
    if (T>=0) {
        dTJerk=d+v*T+a*T*T/2+j*T*T*T/6; vTJerk=v+a*T+j*T*T/2;
        aTJerk=a+j*T;
        jBack=aTJerk*aTJerk/(vTJerk*2);
        TBack=-aTJerk/jBack;
    }
    //System.out.println("T="+T+"  j="+j+"  jBack="+jBack+"  TBack="+TBack);
}
```

### 8.12. Exponentially-Weighted Moving Average

The simple numerical technique outlined below is applied at a number of points in this thesis. This uses the following formula in each step of the algorithm,

\[
s_0 = x_0,
\]

\[
s_{n+1} = \alpha s_{n+1} + (1 - \alpha) s_n,
\]

where \( \alpha \) is the smoothing factor and, 0 < \( \alpha \) < 1 (Wikipedia, 2009). This method is also known as moving averages (Weisstein) and it can be derived from the Kalman filter (as discussed in section 2.3.1.2). Here the values of \( x \) are given, and the values of \( s \) are the
moving averages. The method is taken from statistics but it is related to the \( z \)-transform for the low-pass filter (Distefano et al., 1997; Papoulis, 1980).

The most important point for this thesis is the relationship between the smoothing factor, \( \alpha \), and the rate of decay. Suppose the exponential decay rate is \( \beta \). The decay rate is deduced from the case that \( x_0 = 1 \) and \( x_i = 0 \) for all \( i \geq 1 \). Here,

\[
s_i = (1 - \alpha)s_0 = (1 - \alpha) = e^{-\beta h},
\]

where \( h \) is the step size of the method. Thus,

\[
\alpha = 1 - e^{-\beta h}.
\]

**Equation 8.5**

When \( \beta h \) is small this can be approximated by,

\[
\alpha = 1 - e^{-\beta h} = 1 - 1 + \beta h - \frac{1}{2}(\beta h)^2 + ... \approx \beta h.
\]

If this is applied iteratively and the stepsize varies then this approximation to \( \alpha \) can be applied at each step, otherwise the value of alpha can be precalculated from equation 8.5.

The moving average is used for state estimation and has an effect similar to the \textit{a posteriori} step of a Kalman filter (Simon, 2006) without using the system dynamics for the \textit{a priori}, time-update step. Rather than estimate the process and measurement noise, a practical method is to vary the filter gain until an acceptable response is obtained, reducing noise without having too great a latency.

**8.13. Independent Test Programs for Complex Algorithms**

Figure 8.6 shows a screenshot of a test program running in Java. This example is the test program \texttt{AntCollision.java} in the directory \texttt{ThreeJerksMinTime04}, which implements various suspension controls employing minimum-time control over jerk. This program accesses a control algorithm which has code placed in a separate class defined in the file \texttt{RattlespaceCollision05.java}. The test bed allows the checking of various features of the code, such as checking that the maximum and minimum values found are correct, as shown in the screenshot in figure 8.6. This facility seems to have saved a great
deal of time in debugging complex controls. Once debugged, these algorithms could then be employed in the evolutionary algorithms (in the SuspensionTest program).

Figure 8.6 Test bed for a Particular Suspension Control (Screenshot)

Some time was taken to develop code that generated the panels containing sliders, also shown in figure 8.6. In testing code it was sometimes useful to be able to control a slider and observe the effect immediately in the simulation. The panels are driven automatically from a text file, SliderControl.dat, containing the metadata for these panels. Part of the SliderControl.dat file that generated the panels in figure 8.6 is shown below.

```plaintext
// This is the data file for the groups
// The order of the lines determines which elements are in which group
// and what lines they appear on

/******************* CHASSIS & ROAD
Group = Chassis and Road, 900, 7, 408, 361, show, processDraw

Label = chassis, chassis
DoubleSlider = d, 0.884, distance, -3, 3, 1000
DoubleSlider = v, 0.947, velocity, -3, 3, 1000
```
This allowed needed parameters to be quickly added or deleted to the testing software by simply editing the metadata file. The only extra step was the inclusion of the declaration of a matching parameter in the main Java source program. The code that parses this file, produces the sliders and other input controls, and processes the input was developed by the author for this thesis and can be found in the file SliderGroupControl.java. The documentation for the syntax of the metadata file can be found at the top of SliderGroupControl.java.

### 8.14. Landing-Surface Lemma

The following section proves the lemma used in showing that the landing-surface method has the least jerk of all minimum-time methods that do not suffer overshoot (in section 4.7.3.4). Without loss of generality the initial distance, $d$, is assumed positive. Let the initial velocity
and acceleration be represented as $v$ and $a$ respectively. Refer to figure 4.33 above. There is only one intermediate switch in jerk, and jerk is positive and constant, $j$, from time zero to some intermediate time $t_1$, and is negative, $-j$, from $t_1$ to the final time, $t_E$, at which point it reaches rest, simultaneously achieving zero distance, velocity and acceleration. The value, $j$, is a positive parameter of the particular landing surface.

Suppose the initial velocity and acceleration are held constant, and the jerk found is the jerk required by the landing-surface method as a function of initial conditions: $j_L(d, v, a)$. Thus, if jerk $j_L(d, v, a)$ is applied with the initial conditions given by $d$, $v$ and $a$, then the minimum-time response will have only one switch and will begin on the landing surface. This is defined where $d$, $v$ and $a$ are such that there is overshoot if no control (zero jerk) is applied.

In this section it will be shown that $j_L(d, v, a)$ is a continuous, monotonically decreasing function in $d$. Furthermore,

$$\lim_{d \to \infty} j_L = 0.$$ 

The proof breaks down into cases depending on the sign of $a$ and $v$.

It can be shown that,

$$d = \frac{1}{6} j_E^2 t_1^2 - \frac{1}{2} j t_1^3,$$

$$v = -\frac{1}{2} j t_E^2 + j t_1^2,$$

$$a = j t_E - 2 j t_1.$$

**Equation 8.6**

(This is shown in the Maple file PhD\Maple Experiments\Landing Plane\Landing Plane04.mws. Refer also to PhD\Maple Experiments\Landing Plane\Landing Plane07.mws.)

The two time values must be positive: $0 < t_1 < t_E$.

To simplify the algebra, first define,
\[ \beta(j) \overset{\text{def}}{=} 2 - \frac{4v_j}{a^2}. \]

Equation 8.7

The value under the square root, \( \sqrt{\beta} \), must be non-negative. Solving for \( j \) gives,

\[ j = \frac{a^2}{4v} (2 - \beta). \]

Next equation 8.6 is solved for \( v \) and \( a \), and \( t_i \) and \( t_E \) are found in terms of \( \beta(j) \). There are two cases,

Case 1: \( t_i = \frac{a}{2j} \left( -2 + \sqrt{\beta} \right) \) and \( t_E = \frac{a}{j} \left( -1 + \sqrt{\beta} \right) \)

Case 2: \( t_i = \frac{a}{2j} \left( -2 - \sqrt{\beta} \right) \) and \( t_E = \frac{a}{j} \left( -1 - \sqrt{\beta} \right) \)

Equation 8.8

Using these substitutions, \( t_i \) and \( t_E \) can be eliminated from the expressions for \( d \), \( a \), and \( v \), deriving \( d \) as a function of \( a \), \( v \), and \( j \). The following result is found for case 1.

\[ \frac{ad}{v^2} = \delta_1(\sqrt{\beta}). \]

Equation 8.9

where,

\[ \delta_1(x) \overset{\text{def}}{=} \frac{2(4 - 6x^2 + 3x^4)}{3(2 - x^2)^2}. \]

Equation 8.10

In case 2,

\[ \frac{ad}{v^2} = \delta_2(\sqrt{\beta}). \]

Equation 8.11

where,

\[ \delta_2(x) \overset{\text{def}}{=} \frac{2(4 - 6x^2 - 3x^4)}{3(2 - x^2)^2}. \]
Equation 8.12

These equations allow the derivation of the landing surface jerk given the initial conditions, but they exclude the special cases, \( v = 0 \) or \( a = 0 \), which are very easily dealt with independently.

\[
\delta_1(x) = \frac{2(4 - 6x^2 + 3x^3)}{3(2 - x^2)^2}
\]

Figure 8.7 Case 1 Delta Function

The function \( \delta_1 \) is shown above in figure 8.7. Note that this is greater than zero. In case 1, \( a \geq 0 \), otherwise \( \frac{ad}{v^2} < 0 \) under the assumption that \( d > 0 \), contradicting the fact that \( \delta_1 > 0 \) and noting equation 8.10.

Case 1 also implies \( v < 0 \). Since \( a \geq 0 \), there is no collision with zero if the initial velocity is positive and no control is applied: \( j = 0 \). Thus the following results also from equation 8.7,

\[
\frac{d\beta}{dj} = -\frac{4v}{a} > 0.
\]

Equation 8.13

Furthermore \( \beta > 4 \). This follows from the fact that \( a > 0 \) and \( t_i > 0 \) (and \( j \) is assumed positive).
It is clear from figure 8.7 that $\delta_1$ is monotonic decreasing for $x \geq 2$. And so,

$$\frac{d\delta_1(\sqrt{\beta})}{d\beta} < 0.$$  

Using this and equation 8.9 gives the following,

$$\frac{dd}{dj} = \frac{v^2}{a} \frac{d\delta(\sqrt{\beta})}{d\beta} \frac{d\beta}{dj} < 0.$$  

(In the middle expression, the first term is positive, the second term is negative, and the final term is positive.)

Next turn to case 2. The graph of $\delta_2$ is shown in figure 8.8.

$$\delta_2(x) \overset{\text{def}}{=} \frac{2(4 - 6x^2 - 3x^3)}{3(2 - x^2)^2}$$

Figure 8.8 Case 2 Delta Function

Case 2 implies $a < 0$ because $a \geq 0$ and $t_E > 0$ implies $-1 - \sqrt{\beta} \geq 0$, which implies that $-\sqrt{\beta} \geq 1$: a contradiction.

Equation 8.6 implies that,

$$d > 0 \iff \frac{t_1}{t_E} > \frac{1}{\sqrt{2}}$$

and,
\[ v > 0 \iff \frac{t_1}{t_E} > \frac{1}{\sqrt{2}}. \]

In general, in case 2, equation 8.8 implies that,
\[ \frac{t_1}{t_E} < \gamma \iff \sqrt{\beta} > \frac{2 - 2\gamma}{2\gamma - 1}, \]
where \( t_E > 0 \) and \( \gamma > \frac{1}{2} \).

Thus the fact that \( d > 0 \) gives,
\[ \sqrt{\beta} > 0.702414, \]
and it can be shown that at this value, \( \delta_2(\sqrt{\beta}) = 0 \). Thus \( \delta_2 \) is always non-positive under the conditions of the proof.

Case 2 is slightly more complicated than case 1 since \( v \) can be positive or negative. It can be shown that,
\[ v > 0 \iff \frac{t_1}{t_E} > \frac{1}{\sqrt{2}} \iff \sqrt{\beta} < \sqrt{2}. \]

Note that \( \delta_2 \) is asymptotic at \( \sqrt{2} \). Thus there are two subcases of case 2: \( v > 0 \) corresponds to the portion of \( \delta_2 \) where \( \sqrt{\beta} > \sqrt{2} \), and \( v < 0 \) corresponds to the portion,
\[ 0.7021414 < \sqrt{\beta} < \sqrt{2}. \]

Consider the subcase where \( v > 0 \). This occurs when \( \sqrt{\beta} < \sqrt{2} \), and \( \delta_2 \) has a positive slope. Also, recalling that \( a < 0 \),
\[ \frac{d\beta}{dj} = -\frac{4v}{a} > 0, \]
and so from equation 8.11,
\[ \frac{dd}{dj} = \frac{v^2}{a} \frac{d\delta_2(\sqrt{\beta})}{d\beta} \frac{d\beta}{dj} < 0. \]
(The first term is negative, and the last two are positive.)

Next, consider the subcase where \( v < 0 \). This occurs when \( \sqrt{\beta} < \sqrt{2} \), and \( \delta_2 \) has a negative slope. Also,
\[ \frac{d\beta}{dj} = -\frac{4v}{a} < 0. \]
and again, 
\[
\frac{dd}{dj} = \frac{v^2}{a} \frac{d\delta_2(\sqrt{\beta})}{d\beta} \frac{d\beta}{dj} < 0.
\]

(All three terms are negative.)

Thus \( d \) as a function of \( j \) is continuous, monotonic decreasing in all cases and subcases for initial non-zero values of \( a \) and \( v \) where collision would occur with zero control. Thus, the inverse function \( j_L(d, v, a) \) is also continuous and monotonic decreasing in \( d \):
\[
\frac{\partial j_L}{\partial d} < 0.
\]

Furthermore it can be shown that in all cases, 
\[
\lim_{d \to \infty} j_L = 0.
\]

This is shown below in the case \( a < 0 \) (case 2); the other case is similar.
\[
\lim_{j \to 0} d(j) = \lim_{\beta \to 2} a \frac{v^2}{\delta_2(\sqrt{\beta})} = \infty.
\]

Inverting this result, and given that \( j_L \) is continuous, monotonic decreasing, it is clear that the limit of \( j_L \) is zero as displacement approaches infinity.

**8.15. Numerical Methods**

All algorithms used one of two numerical methods, with different algorithms producing different control values being applied in a consistent manner as explained above. The active algorithms produce force values, and the semi-active algorithms produce damper stiffness values. The author wrote a Java program to test the accuracy of numerical methods (refer to figure 8.9). Methods for developing approximations were tested against known solutions. Step-size values could be tested for accuracy.

The predictor-corrector numerical method was tried but was rejected in favour of the methods using a fixed step size since the control algorithms themselves are timed to run with a fixed step size. It was found that the step size for the numerical method with both active and semi-
active techniques was accurate with the step size of 10 ms, which is the step size nominated for a reasonable control of an electronically controlled damper. Furthermore, this stepsize resulted in a reasonably efficient method that did not accumulate numerical errors (a problem that can arise if the step size is too small).

Figure 8.9 Screenshot of Numerical Test Program

In figure 8.9 the Runge-Kutta method (red dashed curve) is being compared with the analytical solution (black solid curve) in the graph on the left. The two solutions overlap each other closely in this diagram. The program being run here is defined in the following file:

```
Java/DEsRoad2Springsx/Main.java.
```

(Here “x” represents the version number.) This test was performed with a passive suspension. The graph on the right can show the numerical error plotted against a wide range of parameters (the screen shot shows the graph of error versus the damping rate, c). The error is calculated by taking the difference between the numerical and analytical solution. Average and maximum errors are also shown.
As discussed above, the main suspension test bed program also opened a form that ran a comparison of the solution with the current step size against a solution with a step size ten times smaller; see figure 8.10. Variations between the two can test the accuracy of the numerical method applied. The screen shot below shows the numerical test form, with the numerical step size of 10 ms.

![Numerical Test Frame](image)

**Figure 8.10 Numerical Test Frame**

In the graph on this form the two solutions are indistinguishable. Graphs can be “zoomed” by clicking the mouse on a section of the graph (as part of the FunctionGraph program written by the author). The screen shot in figure 8.11 shows a portion of the graph in figure 8.10 expanded a number of times. The difference between the two approximations after 30 seconds is approximately 0.3 mm. This is accurate enough to provide a fair representation of a suspension’s functionality.
Figure 8.11 Expansion of Two Solutions – The bottom graph is the more accurate

The fourth-order Runge-Kutta method was used where this improved accuracy. The method was implemented using the code shown below in figure 8.12.

```java
/**
   * Runge Kutta
   */

public boolean next(double c){
    this.c = c;
    double k1a=v;
    k1b=acc(t, y, v);
    double k2a=v+h*k1b/2;
    k2b=acc(t+h/2,y+h*k1a/2,v+h*k1b/2);
    double k3a=v+h*k2b/2;
    k3b=acc(t+h/2,y+h*k2a/2,v+h*k2b/2);
    double k4a=v+h*k3b;
    k4b=acc(t+h ,y+h*k3a ,v+h*k3b );
    y=y+h*(k1a+2*k2a+2*k3a+k4a)/6.0;
    v=v+h*(k1b+2*k2b+2*k3b+k4b)/6.0;

    acc=acc(t, y,v);
    t+=h;
    if (t>tEnd) {
        return false;
    }else{
        return true;
    } // end if
}
```

Figure 8.12 Code for the Runge-Kutta Numerical Method
8.16. Conjectured Optimality of Skim Control

As discussed above in section 4.7.3.5 the “no overshoot” problem is constrained in the state-space: \( y(t) \geq 0 \). The invariance of the Hamiltonian (Mesterton-Gibbons, 2009, p180; MacCluer, 2005, p120) applies in the case of control constraints but not for constraints in the state-space. The investigation requires checking the optimality of cases that are on the border of the admissible state-space when rebound occurs, in “a form of integer programming” (Ross, 2009, p64) extended to functionals. (If rebound does not occur the solution is the same as the minimum-time solution unconstrained by distance.) In the case that the minimum-time method overshoots, an admissible, locally optimal control will be on the border of the state space. The following only examines the case that is assumed to be optimal (refer to figure 8.13).

The following is not a proof of the conjecture but indicates what may be required for a proof, as well as showing numerical experiments that have not yet found a counter-example.

The time from \( t = 0 \) up to \( t = t_E \) is the “first stage” of the control. At the end of this stage, the state trajectory “skims” the time axis with both distance and velocity equal to zero,

\[
y(t_E) = \dot{y}(t_E) = 0.
\]

**Equation 8.14**

The acceleration at the point of skim is important in this analysis and is designated as the “rebound acceleration”,

\[
a_R \overset{\text{def}}{=} y(t_E).
\]

**Equation 8.15**

After this point, from time \( t_E \) to \( T \), will be designated the “rebound”. The time taken by the rebound is \( t_R \). Thus the total time \( T \) is given as \( T = t_E + t_R \). The solution being sought is an admissible control to minimize \( T \).
The response during the first stage is unconstrained by distance up to the final point, as is the response during the rebound stage. Bellman’s principle of optimality (Kirk, 1970, p54) stipulates that the two sections of the curve that are not on the boundary, before and after skim, must separately be optimal for their end conditions without state constraints: “If [control] $u$ is optimal on $[t_i,t_f]$, it is optimal on every subinterval” (MacCluer, 2005, p121).

The rebound trajectory, between time $t_E$ and time $T$, is easy to describe since it is a special case of the minimum-time constrained jerk control with zero initial distance and velocity. Following equation 4.21 there is a simple relationship between time taken and residual acceleration,

$$t_R \approx 2.0000247 \frac{a_R}{j} \approx \frac{2a_R}{j}.$$

The first stage, time zero up to $t_E$ is less easily dealt with.

The period between time zero and $t_E$ is also a minimum-time control, but it terminates in the condition given by equation 8.14 and equation 8.15: zero distance and velocity, but a non-zero acceleration, $a_R$. This problem has exactly the same adjoint equations as equation 4.12. Thus the same conclusion can be drawn, that there are at most two intermediate switches of control jerk.

A perturbation of the conjectured one-intermediate-switch control, that results in a two-intermediate switch control will require small changes in switching time. These small changes in switching time, albeit with finite changes in control value, are viewed as
“perturbations” in the generalized proof of Pontryagin’s theorem (Mesterton-Gibbons, 2009, p169; Pontryagin et al., 1986, p87). The only way for a perturbed control to be of the form proposed in the previous paragraph (with two intermediate switches) is if there is a switch of jerk introduced either close to the end or the start of the time period \((0, t_E)\).

Next turn to the case of a control switch a small period of time after the start, \(t = \delta t_0\). Note that \(\delta t_0 < 0\) has no physical meaning. Refer to the graph in figure 8.14. This plot was produced by the author’s software, which calculated the various switching times to 10-digit accuracy.

The introduction of the initial switch produces small changes in the intermediate switching time, which is designated as \(t_1\), and the final time, \(t_E\), giving \(t_1 + \delta t_1\) and \(t_E + \delta t_E\). The values, \(\delta t_1\) and \(\delta t_E\) can be determined from \(\delta t_0\) because skim occurs at \(t_E + \delta t_E\),

\[
y(t_E + \delta t_E) = \dot{y}(t_E + \delta t_E) = 0.
\]

It is important to find the concomitant change in residual acceleration as a result of the small time interval \(\delta t_0\).

![Figure 8.14 Perturbed Control](image)

Since \(T = t_E + t_K\), and using equation 4.21, the variation in \(T\) is given as,
A first-order estimate of this value can be produced by ignoring second-order infinitesimal terms, and using the fact that distance and velocity must be zero at the end. The following has been derived in this manner,

\[
\delta T = \delta \dot{E} + \frac{dt_R}{da_R} \delta a_R = \delta \dot{E} + \frac{2.0000247}{j} \delta a_R.
\]

This value is close in practice.

Using modelled examples, as shown in figure 8.14, a plot of \( T \) against \( \delta \dot{0} \) is derived, at least for particular cases of initial distance, velocity and acceleration, as well as a given value of control jerk, as shown in figure 8.15. Here the accuracy in timing is to 10 decimal places. (Press the button labelled “Skim Test1” in OvershootDemo.java.)

The graph in figure 8.15 is typical for all values input by the author. From the graph \( t_e \) tends to decrease, but smaller values of \( t_e \) produce larger values of residual acceleration, \( a_R \), and the overall effect is that the total time, \( T \), increases with increasing values of \( \delta \dot{b} \). The initial conditions for the results shown in figure 8.15 are,

\[
d = 1.7, \quad v = -4.75, \quad a = 6.75, \quad j = 5.5615.
\]

This result is typical for all initial values input by the author. Of course, in order to prove the conjecture, all possible cases must be shown, and perturbations at the end of the control need to be examined.
8.17. Code for Two-Second Timer Counter

The following code is used in the ATmega644 to count a number of events in a two second period. The output of the timer described in section 6.2 needs to be connected to pin C0. To perform the count, run the method, `twoSecClockStep()`.

```
/******************************************
    Header file for timer calibration
   
Ian Storey
First written: 15 Dec 2009
Last changed:  15 Dec 2009

Tested with the Atmega644 at 8 MHz

The two second timer should be connected to pin C0.
This goes high and low every 2 seconds.

The counter counts in one phase 2 second cycle and
displays the count in the other (otherwise the count is affected)

The cycle is started with a one and ends on the next one.
Experiments show that debouncing is unnecessary.

The phase is set by the variable
calibratePhaseCounter
This has the values:
-1: before count
Counting:
  0: start of count - pin C0 is high
  1: ending of count - pin C0 is low
Displaying:
  2: start of display - pin C0 is high
  3: end of display - pin C0 is low

The count is contained in the variable twoSecondStepCounter
which can just be incremented in code

When there is a change in pin C0 call the function
twoSecClockStep(unsigned short)
The parameter is the value of pin C0.
A change is recognized when
previousPinC0
is not equal to the current value

This assumes an 8MHz clock
******************************************/

unsigned long twoSecondStepCounter=0;
short calibratePhaseCounter=-1;
unsigned short previousPinC0=0xFF;
```
// perform a step
// Note this does NOT count
void twoSecPhaseChange(unsigned short pinC0){
    pinC0 = pinC0 & 1;
    //
    // printf("pinC0=%d calibratePhaseCounter=%d twoSecondStepCounter=%u\n", pinC0,calibratePhaseCounter,twoSecondStepCounter);
    switch (calibratePhaseCounter) {
        case(-1):
            if (pinC0==0){
                calibratePhaseCounter=3; // end of display - ready
            }
            break;
        case(0):
            if (pinC0==0){
                calibratePhaseCounter=1; // ending of count - ready
            }
            break;
        case(1):
            if (pinC0==1){
                calibratePhaseCounter=2; // do display
            }
            // ***** Display the result.
            printf("twoSecondStepCounter=%u\n", twoSecondStepCounter);
            break;
        case(2):
            if (pinC0==0){
                calibratePhaseCounter=3; // end of display - ready
            }
            break;
        case(3):
            if (pinC0==1){
                calibratePhaseCounter=0; // do count
            }
            // ***** initialize count variables
            twoSecondStepCounter=0;
    }
    previousPinC0=pinC0;
    PORTB = ~pinC0;
}

void twoSecClockStep(){
    unsigned short val = PINC;
    twoSecondStepCounter++;
    if (val!=previousPinC0) twoSecPhaseChange(val);
}

8.18. Overview of Test Bed Program Design

The test bed program uses object-oriented design principles. Thus an object of a child class can inherit properties from a parent class; for example, all the genes containing the
suspension control algorithms inherit properties of the gene class, and this allows them to be processed by the routines that run the evolutionary algorithm processes (selection, mutation, and crossover) in a consistent manner.

In object-oriented terminology, functions, subroutines and procedures are known generically as “methods”. A convention often used in object-oriented programming is to name methods or variables by use of the class name followed by a dot, then the method or variable name. For instance the `displayParameters()` method in the `RoadParametersDialog` class could be written as,

```
RoadParametersDialog.displayParameters();
```

In some cases, more than one class has been defined in a single file. For example the `RoadParametersDialog` class and the `PhysicalParameters` class are defined in the file `Parameters.java`, along with a number of other classes related to the setting, saving and reading of parameter values. All genes are defined in the file `Gene.java`.

The class names of the various genes are used throughout to distinguish the various kinds of suspension control algorithms. For example, the term `ActivePureSkyhookGene` has been used throughout to name the purely linear skyhook algorithm applied to an active suspension, and this gene has been defined in the class of the same name.

An HTML file has been written to briefly describe the algorithm applied by most genes developed during the course of the PhD research:

```
SuspensionTestx/Help/genesHTML/Genes.html.
```

This can be accessed from a central page, `SuspensionTestx/Help/Help.html`, which also contains links to gene performance statistics and class documentation for many classes in the main EA program.

From the main window, the “Start” button runs a series of generations using the same mutation control parameters. During the process of development for this thesis, this was a useful testing feature but for the main numerical results quoted below, a system of cooling the evolutionary process was developed.
The “Auto-Evolution” option from the “Tools” drop-down menu, shown in figure 8.16, runs a number of generations of evolution allowing a slow cooling process (see section 2.12.2). The cooling process is controlled by mutation parameters stored in one of a number of databases contained in the subdirectory SuspensionTestx/GenDB. The auto-evolution process also has an option for running evolutionary algorithms on a number of genes, one after the other. With a large number of genes this process can take days. The process is assisted by the fact that the process can be stopped and restarted; stopping in the middle of an auto-evolution did not requiring starting again from the beginning. The various gene types that are run in this way are specified in the file, SuspensionTestx/GenDB/AutoEvolutionList.txt, explained in more detail below. This allowed the test bed to run evolutionary schedules on a large number of different types of genes using the exact same process with exactly the same road types, the results of which could be used for comparing the genes. Sequences of separate tests for a range of genes showed consistent results across different tests with only minor variations in the final ordering of the fitness of the various gene types.

Figure 8.16 “Tools” Drop-Down Menu

The following is a brief outline of the main Java classes used in the project:

- **SuspensionTest** is the main class that sets up the user interface, initializes data and handles user requests.
- **Generation** contains a number of genes of the one type.
- **Gene** is an abstract superclass for all the different types of genes. Different genes will have different controls and different parameters.
- **EAController** is called by SuspensionTest and controls the running of the evolutionary algorithm through a number of generations.
- **GenerationPanelController** has the role of looking after the display during a run of the EA. It contains the main graph of fitness, and a display of a run of the
algorithm. The display of individual suspension runs can be switched on or off from the menu item View, Plot Road versus Genes. This should be turned off when running a long evolutionary algorithm.

- **NumericalControl** controls the numerical methods used. It is initialized with the method, `initialize()`. Each numerical step is performed with the method, `findNextValues()`. This returns a true when the run of the algorithm is complete. For efficiency, this class is instantiated once only and then run for each gene in a generation.

- **Breeder** is the main class for breeding one generation from the previous. The major method of this class is `breed()`, which performs copying, mutation and crossover and returns a new generation object containing the next generation. This class employs a couple of helper classes to perform the copying, mutation and crossover: `Genome` and `GenomeConverter`.

- **Genome** contains the genome information simply as a vector of real numbers (Java `double` data type). The genome class is only needed for mutation and crossover. When mutation or crossover occur, the gene is converted to a genome using `GenomeConverter.geneToGenome()` and converted back using `GenomeConverter.genomeToGene()`.

- **GenomeConverter** contains three main methods: the two converter methods, `geneToGenome()` and `genomeToGene()`, as well as a `copy()` method. These methods are all static i.e., they can be called without an instantiated GenomeConverter object.

- **RoadSet** contains a set (Java `Vector` class) of superposed roads to be used by a gene and controls the creation of roads.

- **SuperposedRoad** is a road formed from the superposition of roads.

- **Road** is the main template (Java `Interface`) for a number of road types that extend it, `SinRoad` (sinusoidal), `SquareBumps`, `RoundBumps` and `Triangular`.

- **AutoEvolution** controls the process of cooling

SuspensionTest calls AutoEvolution which sets up an EACcontroller object when an evolutionary algorithm is initiated. The genes are supplied through a generation object, which contains a set of genes of the one type that is iteratively tested and scored for fitness. The `EACcontroller.runController()` method runs through a number of generations.
Simulations are run using the `runSimulation()` method, and fitness measures of the genes are produced. After this the EAController object calls the breeder object to perform the selection, mutation, and genetic crossover of the evolutionary algorithm.

Routines in the breeder object perform selection, mutation and crossover. Whenever a mutation or crossover event occurs, genes are converted to “genomes”. The genomes are composed of a vector of numbers (actually a Java `Vector` data type). Mutation and crossover are performed directly on the genomes. The `Genome` class contains the methods `mutate()` and `crossover()`. The `GenomeConverter` class performs the conversion between gene and genome and back again, after mutation or crossover.

A number of parameters control the running of the program. Parameters are persistent and are stored in a file `SuspensionTextX/Parameters.dat`. Data capture forms such as the one shown in figure 8.17 can be used to view and set parameters. These are opened from the “Parameters” menu. The various different data-capture forms cover the following groups of parameters: physical, fitness, mutation, road, rattlespace, generations, auto-evolution and graphical parameters. If the file `Parameters.dat` does not exist, a warning is given and default values are loaded.
The last-used generation data for each different kind of gene is contained in a separate data file (streamed object data for each gene type). For example, the generation for the ActivePureSkyhookGene is stored in the file Generations/ActivePureSkyhookGene.dat. When a generation is changed to a new gene type, the saved data is used, if it can be found, otherwise a fresh generation will be formed.

The two main parameters to control the auto-evolution are contained in the AutoEvolution class:

- The boolean variable, useListOfGenes, determines if the currently selected gene is to undergo an evolutionary algorithm or if a list of genes is to be processed.
- The String variable, fileName, is the name of the database file containing the evolution schedule.

There are a number of different Access database files that provide for different types of evolution:
- Very quick evolution for testing, HotQuickGenDB.
- Moderately quick evolution for determining strong algorithms, GenDB.
- Long evolution for careful determination of gene parameters, LongGenDB.

The results from the last type of evolution are used in the numerical experiments.

The list of names of the genes to be used in the auto-evolution process is stored in the flat ASCII file, GeneDB/AutoEvolutionList.txt. A simple example of this file is shown in figure 8.18. Any lines up until the line containing “<start>” are ignored and can be used as comments. Following this in the file is the list of names of genes that are to be tested.

```
This file is used for auto evolution
It contains the list of genes to be tested.

<start>
MinJerkActive01Gene
ActiveAdaptiveJerk01Gene
HardPassiveGene
ActiveAdaptiveJerkFilter01Gene
PassiveGene
LoJerkConstantKAdaptiveJerkFilterGene
```

**Figure 8.18 Simple Example of Contents of ASCII file, GeneDB/AutoEvolutionList.txt**

Statistics for a generation are stored with the generation data, which is itself saved on hard disc. The statistics form, shown in figure 8.19, is produced from data in the generation data files. A graphical overview of the data is also placed in HTML files that can be viewed in a web browser. Also, there is a statistics summary file, which is generated when the user presses the “All Stats Summary” button in the Statistics form, as shown in figure 8.20.
Pressing this button generates a HTML page containing a list of the basic statistics on all tested control algorithms, including the median performance index of the genes in the final generation of evolution. This final list provides an overview of the performance of the various suspension types represented by the corresponding genes.
8.19. Road Surfaces

This section contains details on the road simulations. The details do not affect the thesis as a whole and the main concepts of this section can be gleaned by examining the figures containing graphs of the various types of road surfaces that are combined for test roads in the experiments.

As discussed in section 3.2, in this thesis the vehicle’s forward velocity is assumed constant so road height functions can be represented purely as functions of time.

All simulated road surfaces used in the EAs contained a mixture of corrugations to test comfort as well as larger bumps that test the suspension’s capacity to track the road surface. The various road surfaces that are combined in the final roads can be broken into four groups: sinusoidal corrugations, trapezoidal bumps, square bumps, and corrugated bumps.

Bumps are generated randomly using parameters explained in this section. If only one road surface was used for all the roads and all steps of the evolutionary process, the resulting suspensions might optimize only for some particular property of that road surface, or could exploit some property of the given road. On the other hand, too much variation in the road surfaces greatly slows the evolutionary process, since some genes will achieve higher scores simply because they encounter smoother road conditions. Even with sets of random roads there needs to be some uniformity in the roughness of the road surfaces encountered in the evolutionary process.

Sinusoidal bumps are pure sinusoids or mixtures of pure sinusoids, as depicted in figure 8.21. Figure (a) has a single sinusoidal function and figure (b) is a sum of two sinusoidal functions of different amplitude, frequency and phase shift. The derivative is also calculated and is shown as a dotted line (the derivative will be shown similarly in all examples in this section.).
Figure 8.21 Example Sinusoidal Road Surfaces

Amplitudes, frequencies and phase shifts are based on the following parameters from the RoadParameters class (defined in the file Parameters.java):

- minHeightSinusoidal – minimum amplitude,
- maxHeightSinusoidal – maximum amplitude,
- minHz – minimum frequency (Hz),
- maxHz – maximum frequency (Hz),
- power – amplitude reduction factor (integer).

The number of sinusoidal functions are selected as uniform random variables between minimum and maximum values. The following two Java routines select either floating point (double) values or integer (int) values randomly between minimum and maximum limits.

```java
private double uniformRandom(double min, double max) throws SuspensionTestException {
    if (min > max) throw new SuspensionTestException("min="+min+">max="+max+");
    RoadSet.uniformRandom();
    double temp = min;
    if (min != max) {
```

411
temp=min+Math.random()*(max-min);
} 
return temp;
} 

private int uniformRandomInt(int min, int max)throws SuspensionTestException{
    if (min>max) throw new SuspensionTestException("min="+min+">max="+max+"\n
    RoadSet.uniformRandom()");
    int temp=min;
    if (min!=max){
        temp=min+ (int)(Math.floor(Math.random()*(max-min+1)));
    }
    if (temp>max) temp=max;
    return temp;
} 

The frequency is determined from a logarithmic random variable (by using a uniform random variable over the range of the logs of the maximum and minimum frequency and then taking the exponential). Because the spectral power density rises with frequency, the maximum amplitude that can be chosen decreases with frequency. This is achieved by making the effective maximum amplitude a fraction of the maximum amplitude, minHeightSinusoidal, using the following formula,

\[ h_{\text{max}} = \left( \frac{f_{\text{min}}}{f} \right)^n H_{\text{max}}, \]

where \( f_{\text{min}} \) is the minimum frequency (minHz), \( f \) is the randomly chosen frequency, \( n \) is a power parameter (either 0, 1, 2 or 3, set by the user), and \( H_{\text{max}} \) is the maximum amplitude at the lowest frequency (maxHeightSinusoidal in the RoadParameters class). If \( h_{\text{max}} \) is less than the minimum amplitude set in the parameters, then the height is \( h_{\text{max}} \), otherwise the height is chosen uniformly randomly between minHeightSinusoidal and \( h_{\text{max}} \).

The Java routine for manufacturing a single random sinusoidal road is shown immediately below.

```java
private Road makeSineRoad(double HeightMin, double HeightMax, double freqMin, double lnFreqMin, double lnFreqMax, int power)throws SuspensionTestException{
    double u=uniformRandom(lnFreqMin, lnFreqMax);
    double freq=Math.exp(u);
    double f=freqMin/freq;
    double HeightMaxOmega=HeightMax;
    for (int i=0; i<power; i++){
        HeightMaxOmega*=f;
    }
    return new Road(freqMin, freqMax, HeightMin, HeightMax, HeightMaxOmega, power, freqMin, freqMax, lnFreqMin, lnFreqMax, power);
}
```
The logs of the minimum and maximum frequencies, lnFreqMin and lnFreqMax, are pre-calculated to avoid the time cost of recomputing these values for each road. The SinRoad constructor takes angular frequency in rads/s as a parameter, as well as the phase shift, “delta”, in radians. Hence the frequency is multiplied by $2\pi$.

Trapezoidal bumps are depicted in figure 8.22. Figure (a) shows a single bump and figure (b) shows two bumps summed.

**Figure 8.22 Trapezoidal Bumps**

The length of the bump is taken from the beginning of the first slope to the beginning of the final slope. The bump length is determined by the following two parameters in the RoadParameters object:

- minBumpLength
- maxBumpLength

The bump length is chosen uniformly between these limits.

The other parameters affecting trapezoidal bumps are:

- minHeightTrapezoidal – minimum bump height,
- maxHeightTrapezoidal – maximum bump height,
- minSlopeTrapezoidal – minimum slope of leading or trailing edge,
maxSlopeTrapezoidal – maximum slope of edge.
The height can be negative, but the absolute value of the bump height is taken uniformly between minHeightTrapezoidal and maxHeightTrapezoidal. The slopes of the ramps are taken uniformly from the range minSlopeTrapezoidal to maxSlopeTrapezoidal. The slopes for the ramps at the beginning and the end of the bump are determined independently.

The Java routine for manufacturing a single random trapezoidal bump road is shown immediately below.

```java
private Road makeTrapezoidalBumpRoad(double lengthMin, double lengthMax, double heightMin, double heightMax, double slopeMin, double slopeMax, double totalTime) throws SuspensionTestException{
    double bumpLength = uniformRandom(lengthMin, lengthMax);
    double T = totalTime - bumpLength/2;
    double bumpStart = Math.random() * T;
    double height = uniformRandom(heightMin, heightMax);
    if (Math.random() > 0.5) height = -height;
    double slope = uniformRandom(slopeMin, slopeMax);
    double gapStart = Math.abs(height / slope);
    slope = uniformRandom(slopeMin, slopeMax);
    double gapEnd = Math.abs(height / slope);
    return new TrapezoidalBumpRoad(bumpStart, bumpLength, height, gapStart, gapEnd);
}
```

“Square bumps” actually have rounded ends. Examples of square bumps are shown in figure 8.23. Figure (a) has a single square bump and figure (b) is a sum of two bumps, with different heights, lengths and begin and end slopes.

![Figure 8.23 Square Bumps](image)

(a)

(b)
The bump length for a square bump must be at least long enough that the up and down ramps do not overlap (this is to ensure the slope continuity of the bump). The Java routine for manufacturing a single random square bump is shown immediately below.

```java
private Road makeSquareBumpRoad(double lengthMin, double lengthMax,
                                 double heightMin, double heightMax, double slopeMin, double slopeMax,
                                 double totalTime) throws SuspensionTestException{
    double bumpLength = uniformRandom(lengthMin, lengthMax);
    double T = totalTime - bumpLength / 2;
    double bumpStart = Math.random() * T;
    double height = uniformRandom(heightMin, heightMax);
    if (Math.random() > 0.5) height = -height;
    double slope = uniformRandom(slopeMin, slopeMax);
    double gapStart = Math.PI * Math.abs(height / slope) / 2;
    double gapEnd = Math.PI * Math.abs(height / slope) / 2;
    return new SquareBumpRoad(bumpStart, bumpLength, height,
                               gapStart, gapEnd);
}
```

The ramps at the start and the end of the bumps are segments of a sine wave. The height function for a given road is calculated as in the Java function below. The code for the sinusoidal segment is shown in bold.

```java
public double height(double t){
    double temp = 0;
    if (t > bumpStart && t < bumpStart + gapStart &&
      t < bumpStart + bumpLength) {
        temp += height * (Math.sin(omegaStart * t + deltaStart) + 1) * 0.5;
    } else{
        if (t > bumpStart + bumpLength &&
            t < bumpStart + bumpLength + gapEnd){
            temp += height * (Math.sin(omegaEnd * (t) + deltaEnd) + 1) * 0.5;
        } else{
            if (t > bumpStart && t < bumpStart + bumpLength)
                temp += height;
        }
    }
    return temp;
}
```

The values for omegaStart, deltaStart, omegaEnd and deltaEnd, are pre-calculated during the construction of the SquareBumpRoad object.

The function that calculates the road slope is given by the following code.

```java
public double heightPrime(double t){
    double temp = 0;
    if (t > bumpStart && t < bumpStart + gapStart &&
        t < bumpStart + bumpLength) {
        temp += height * omegaStart * Math.cos(omegaStart * t + deltaStart) * 0.5;
    } else{
        if (t > bumpStart + bumpLength &&
            t < bumpStart + bumpLength + gapEnd){
```
The square bumps do not have slope discontinuities (refer to section 4.3) and so should produce much less discomfort than trapezoidal bumps of comparable shape, especially for suspension algorithms that produce spikes in jerk over slope discontinuities, such as the linear passive suspension.

Another kind of bump called a “corrugated bump” is also used. Pure sinusoids were not used across the entire run. Instead, sinusoids were multiplied by “square bumps”, as defined above, to produce sinusoidal corrugations that exist only for a period of time. Examples of corrugated bumps are shown in figure 8.24. Figure (a) shows a single corrugated bump and figure (b) shows two bumps summed.

![Figure 8.24 Corrugated Bumps](image)

Corrugated bumps are essentially square bumps multiplied by sine bumps. The parameters that determine corrugated bumps, in the RoadParameters class, are the following:

- `minHeightSinBump` – minimum amplitude,
- `maxHeightSinBump` – maximum amplitude,
minSlopeSinBump – minimum slope for ramp (not for the sinusoidal factor)
maxSlopeSinBump – maximum slope for ramp
minHzSinBump – minimum frequency of corrugation
maxHzSinBump – maximum frequency of corrugation
powerSinBump – amplitude reduction factor (integer).
The first four parameters determine the square bump. The last three (and the amplitude)
determine the sine function. Suppose the square function is represented as \( S(t) \), and the sine
function is represented as \( \sin(\omega t + \delta) \). Suppose that \( S(t) \) has the overall amplitude, although it
is irrelevant which is of unit height and which has the overall height. The height of the
corrugated bump can then be represented as,
\[
h(t) = S(t) \sin(\omega t + \delta),
\]
The differential requires the product rule;
\[
\dot{h}(t) = \dot{S}(t) \sin(\omega t + \delta) + \omega S(t) \cos(\omega t + \delta).
\]
The same amplitude reduction scheme as used for sinusoidal bumps, described above, was
used with corrugated bumps.

Java routines were used to calculate both the road height and the height velocity. The
functions for calculating the road height and the derivative of road height are defined in the
functions height() and heightPrime() in the SinBumpRoad class located in the file
Road.java. These functions use values omega, omegaStart, omegaEnd, deltaStart and
deltaEnd, which are calculated during construction to save recalculation.

The various kinds of bumps can be summed, as for example in figure 8.25. A RoadSet
object is used to hold a vector (Java Vector) of Road objects (each instance of one of the
various kinds of bumps). The example of figure 8.25 contains four trapezoidal bumps, five
square bumps and four corrugated bumps. The numbers of the different types of bumps are
determined by the following integer variables from the RoadParameters class:

minNumberSinusoidal, maxNumberSinusoidal,
minNumberSquare, maxNumberSquare,
minNumberTrapezoidal, maxNumberTrapezoidal,
minNumberSinBump, maxNumberSinBump.
The numbers for each type of bump in a particular RoadSet are chosen to be equally likely from within the minimum to maximum values.

![Combined Road](image)

**Figure 8.25 Combined Road**

There is also a boolean parameter in the RoadParameters class, decreaseWhenSuperpose, which activates a routine to decrease bump heights when they overlap. When this boolean is false, overlapping bumps are not changed. Each road class has routines that report the bump height and the start and end times of the bump, `getMaxHeight()`, `getBumpStart()` and `getBumpEnd()`. There is also a routine to reset the height, `setMaxHeight()`. (These are declared in the Road Interface and must be supplied by each Road subclass.) For each pair of Roads in a RoadSet, heights are compared against maximum allowed heights for these road sets. Consider a pair of roads, road A and road B. Suppose also that road A has bump height $h_A$ and a maximum allowed height $H_A$, while road B has bump height $h_B$ and a maximum allowed height $H_B$. A factor is calculated from the ratios of these values,

$$ H = \frac{h_A + h_B}{H_A + H_B} $$

(If one of the roads is a Sinusoidal road, the absolute values of the separate ratios are summed.) If the factor $H$ is greater than one, each of the two road heights is divided by this factor.

The RoadParameters class also contains a boolean parameter, `randomizeEachGene`, which determines if a new set of road surfaces is used for each genome tested. If this is false, the same set of roads will be used for each gene in a generation, but a new set will be chosen for the next generation. In early generations, using the same set of road surfaces can help to speed the evolution by testing the genomes under the same conditions. However, in the later stages, a larger numbers of roads in the road sets produces a better statistical mix of road surfaces and a statistical measure of the overall performance of a generation.
Road Parameters used for generating random roads in the SuspensionTest program are set in the Road Parameters dialog, shown in figure 8.26.

Figure 8.26 Road Parameters Dialog

Samples of roads that this process produces can be viewed by clicking the View menu item and selecting “Sample Random RoadSet...”. This produces a road set and the user can view the various roads by selecting a road from the drop-down list or by using the “Previous” and “Next” buttons.
8.20. **Fitness Measures**

The code for calculating the fitness measures is contained in the class Fitness. The three main methods for setting up the fitness measures are all static methods (static methods can be called without an instantiated object): `initialize()`, `incrementSums()` and `finalizeIntegrals()`. The same fitness routines and measures are used for all numerical controls and genes. The integrals for both the comfort and rattlespace objectives are calculated using Simpson’s method (Kreyszig, 1993, p961).

Once the fitness values are calculated they are used in the instantiation of `FitnessData` objects. The `FitnessData` class is declared in the file, `Fitness.java`. The `FitnessData` instantiation occurs after the jerk and rattlespace integrals are calculated. These are then passed into the `Gene`, using code as in figure 8.27 contained in the `NumericalControl` class. By maintaining both the comfort and rattlespace performance measures, the `FitnessData` object allows both weighted sums and Pareto comparisons to be made.

```java
double JInt = Fitness.getIntegralJerk();
double RattleInt = Fitness.getIntegralRattleSpacePenalty();
FitnessData fitData = new FitnessData(JInt, RattleInt, 0d,
                                        parameters.getFitnessParameters());
gene.setFitnessData(fitData);
```

**Figure 8.27 Calculation of Fitness and Setting of Gene Fitness**

8.21. **Genes and Evolutionary processes**

This section explains the processes by which genes are selected from one generation to the next, and how mutation and crossover are applied to genes. The process used in the test bed program was to have the program rewrite itself. This was done for a number of reasons explained below. This is a complex, non-standard process and warrants explanation in some detail, in order to justify the process and to verify its method.
8.21.1. Genes

“Genes” represent suspension control parameters and these are tested in simulation over a large number of roads. Different gene types are responsible for representing the different suspension control algorithms; each gene class represents a different control algorithm. The gene class thus contains the crucial logic of the control algorithm. Furthermore, the code in the gene class is similar to the control logic in the microprocessor that would control the suspension.

The class hierarchy for the Gene class (defined in the Gene.java file) is depicted in figure 8.28. The Gene class that is the ancestor of all Gene classes (an abstract class) is defined at the beginning of the file Gene.java. The QuarterCarGene class is a subclass of the Gene class and is defined in the file QuarterCarGene.java. All genes that represent active control algorithms are subclasses of QuarterCarGene. The SingleSpringGene is the immediate parent class of all the semi-active controls, and it is itself a subclass of QuarterCarGene.

![Gene Class Structure](image)

Figure 8.28 Gene Class Structure
The active numerical class, `NumericalControlActive`, contains the method `findAccFromGene()`, which returns the acceleration value of every type of active gene (and hence each active suspension control algorithm). This class is defined in the file `NumericalControlActive.java`. In the `findAccFromGene()` procedure, all active genes return the acceleration using a method called `getAcc()`. The `findAccFromGene()` method determines the type of gene and uses the `getAcc` method appropriate for this gene. For example, the `SlidingMode01` gene is called using the statements in the listing below (the procedure call is bolded).

```java
if (gene instanceof SlidingMode01){
    found=true;
    acc=((SlidingMode01)gene).getAcc(t==t0, m, h, t, y, v,
     acc, road,
     isGraphUsed, colorMode);
} // if
```

Each active gene class contains a method `getAcc()` that contains the code for the particular control logic of that gene.

Similarly, the numerical method using semi-active suspensions calls the procedure `findCFromGene()` from the `NumericalControl` class, defined in the file `NumericalControl.java`. Each semi-active gene contains a method `getC()` that is called inside `findCFromGene()`. An example of a call to `getC()` for the semi-active control gene, `SlidingModeSemi01`, is shown below.

```java
if (gene instanceof SlidingModeSemi01){
    found=true;
    //System.out.println("NumericalControl: v=\+v");
    c=((SlidingModeSemi01)gene).getC(t==t0,
     rattlespaceParameters, m, h, t, y, v, numericalData1.getA(), road,
     isGraphUsed, colorMode);
} // if
```

Each semi-active gene class definition contains a call to the method `getC()`, which contains the code for the control logic of that gene.

Once the numerical test bed is written and tested, it is a matter of supplying the control logic for either `getAcc()` or `getC()` for each gene. For example the control logic for the active gene `ActivePureSkyhookGene` is contained in the code listing below.
// ActivePureSkyhookGene
public double getAcc(double m, double h, double t, double y, double v, double prevA, SuperposedRoad road) {
    double s=y-road.getHeight(t);
    double acc=-(kSus*s+cSky*v)/m;
    return acc;
}

The method getAcc() in the semi-active gene NoJerkSkyhookGene is shown below.

/*
NoJerkSkyhookGene
Ahmadian's no-jerk skyhook
*/
public double getC(double m, double h, double t, double y, double v, SuperposedRoad road) {
    double sPrime=v-road.getHeightPrime(t);
    double c=Kj*v;
    if (sPrime<0) {
        c=-c;
    }
    if (c<0) c=0;    // c cannot be negative
    return c;
}

These are quite simple examples. Most genes are more complex, but each gene will call one of these two functions, getAcc() or getC(), and this is where the control logic for that gene is contained. The code for all genes can be found in the file Gene.java, in the class definition for that gene.

8.21.2. Generations

A Generation object (defined in Generation.java) can store a collection (Java Vector) of Gene objects representing the various genes of a generation. The number of genes in a generation is variable, typically in the tens or hundreds. Any one generation holds genes of only one type, each with different parameters. The Generation object is typically saved to hard disk under the name of the gene type. Thus the ActiveAdaptiveJerkFilterGene is saved in the file ActiveAdaptiveJerkFilterGene.dat, the FilterWithStiffening01 gene is saved in the file FilterWithStiffening01.dat, and so on. These are contained in the subdirectory, Java/SuspensionTestX/Generations.
The `Generation` class also calculates and holds various statistics on the generation, such as the mean and median weighted scores and the standard deviation of the weighted scores. It stores this data for previous generations. These results are displayed in the statistics forms discussed in the appendix (section 8.18). The various HTML pages holding the gene statistics are contained in the subdirectory, 

```
Java/SuspensionTestX/stats.
```

### 8.21.3. Genomes and Conversions of Algorithmic Parameters

Genetic mutation and crossover require that suspension parameters change from generation to generation. The suspension control algorithms are most naturally represented as Java variables. In the numerical experiments performed here the data is represented as collections (actually Java `Vectors`) of Java floating-point values (Java `double` variables), although some complex data structures were developed (as seen in the class definitions in the files `FuzzyLinearFunction.java` and `FuzzySplineFunction.java`).

Originally, in the numerical programming of the genomes, genes were represented by arrays of binary values (zero or one), as in Golberg (1989). But when real-valued numbers were used directly as the “genes”, as described in Dumitrescu, Lazzerini et al. (2000) and in section 2.12.2 above, the evolution became much more efficient.

When the parameters are to be altered for mutation or crossover, the gene variables are converted to a collection of floating-point values that represent the “genome” of the gene. The Java variables that represent the suspension parameters are placed in a collection of floating-point values (the Java `Vector` class has been used).

When a gene is involved in mutation or crossover, its parameter variables (the “phenotype” in this analogy) are first converted to a “genome”, and then mutated. After mutation and crossover, the “genome” is converted back to Java variables. This conversion was done for efficiency.
Without the process described below, or something similar, the variable names in the algorithms would be quite unnatural and difficult to decipher, or they would require a complex, error-prone process of hand-coded conversion; for example a variable named “\( k \)” for spring rate, say, must be determined by its position in an array or vector (or collection of some kind). Without the conversion, the crucial algorithmic code would be as completely dissimilar to the code used in the microprocessor (refer to section 6.2).

The process of converting between Java variables and a data structure representing the “genome” was designed so that simple HTML-like code added to the variable declarations facilitated the setting up the genome parameters and their limits, and at the same time uses descriptive variable names that double as standard Java variables. The syntax of this structure is quite simple and is perhaps best explained with an example (refer to figure 8.29).

```xml
<variable> <name>*</variable>
private double ca;
<min>0</min>
<max>2E7</max>
<initial>700E3</initial>
</variable>

<variable> <name>*</variable>
private double cm;
<min>0</min>
<max>2E7</max>
</variable>

<variable> <name>*</variable>
private double cb;
<min>0</min>
<max>2E7</max>
</variable>

<variable> <name>*</variable>
private double k;
<min>0</min>
<max>2E7</max>
</variable>
*/
```
Figure 8.29 Section of Gene.java defining Genome Variables

Figure 8.29 shows a small section of code taken from the Gene.java file that defines a particular gene. The standard java code in this example is shown bolded. All the rest of this code is simply treated as a comment by Java, and is not parsed by the Java compiler. In the example above, four java variables are declared, and the minimum and maximum values for the genomic variables are also declared.

Another piece of code processes this file and parses it for conversion into the genome collection. The HTML-like code is parsed by this separate program and it generates Java code to convert to and from genomes. This processes source files and then changes them, after which the Java source code needs to be recompiled.

The process involves an unusual step, but it has a number of advantages. It creates data structures that are independent of the phenotype for mutation and crossover, and most importantly it leaves standard Java variables intact in the algorithmic code; the code in the numerical experiments is as close to microprocessor code as possible. Because of the separation of suspension algorithmic code from the code for processing the genome, genome selection, mutation and crossover are all carried out independent of the phenotype (the suspension algorithm). Furthermore, once the process is automated and debugged it is error free. If conversion code had to be written by hand for each class, errors would inevitably accumulate. Similarly, non-descriptive variable names generate confusion, waste time, and are responsible for buggy code. Once the process above is automated, the process is relatively quick, convenient and error free.

In the SuspensionTest program, the conversion is activated by clicking on the “Tools” menu and selecting the “Make Automatic Gene Class Definitions” in the main menu (refer to figure 8.30). The GUI control is defined in the file MakeAutoGeneClassDefinitionsFrame.java. The frame controlling the conversion process is opened and the conversion is activated by pressing the button “Start Conversion”. This then runs through the Gene.java file taking all gene definitions and creating the code needed to generate the genome, as well as convert to and from the genome. Some code is then created in the Gene.java file, and there is extra
code created in the GenomeConverter.java file. Once this is finished, the code for the entire program needs to be recompiled.

Figure 8.30 Dialog for Automated Production of Genomic Conversion Code

The routines that produce the “genome” from Java variables and vice versa is contained in GenomeConverter.java. This is used in the evolutionary algorithms for purposes of handling mutation and crossover. This class contains two methods, geneToGenome() and genomeToGene(). The code below is a section in the geneToGenome() method that creates a genome from an instance of the FlatLinearJerkSimpleSemi04 gene. The code instruction gene2.getBetaY0() below, in bold, retrieves the double value of the gene variable betaY0. The function getBetaY0() has been automatically generated by the process outlined above.

```java
if (gene instanceof FlatLinearJerkSimpleSemi04){
    found=true;
    genome.setOriginalGeneType("FlatLinearJerkSimpleSemi04");
    FlatLinearJerkSimpleSemi04 gene2=(FlatLinearJerkSimpleSemi04) gene;
    genome.addDouble(gene2.getBetaY0(), 0.0, 50000.0);
    genome.addDouble(gene2.getBetaY1(), 0.0, 50000.0);
    genome.addDouble(gene2.getBetaY2(), 0.0, 50000.0);
    genome.addDouble(gene2.getBetaS0(), 0.0, 50000.0);
    genome.addDouble(gene2.getBetaS1(), 0.0, 50000.0);
    genome.addDouble(gene2.getBetaS2(), 0.0, 50000.0);
    genome.addDouble(gene2.getK(), 0.0, 5.0E10);
    genome.addDouble(gene2.getKLim(), 0.0, 5.0E10);
    genome.addDouble(gene2.getKPrime(), 0.0, 5.0E10);
}
```
The code section below shows the conversion of a genome back into a gene of the type FlatLinearJerkSimpleSemi04. Note that this passes control back to the gene itself.

```java
if (originalGeneType.equals("FlatLinearJerkSimpleSemi04")){
    found=true;
    gene=new FlatLinearJerkSimpleSemi04(genome);
}
```

This in turn calls a constructor which sets the Java variables from the genome, as below.

```java
// Constructor for Gene from Genome
public FlatLinearJerkSimpleSemi04(Genome genome)
    throws SuspensionTestException{
    genome.formGeneData(this);
    getDataFromGenome(genome);
}
```

The `getDataFromGenome()` method, shown below, contains code for finally converting genome floating-point values into Java variables used by the suspension control algorithm. This code is automatically generated. Note that the variable, `betaY0()` is retrieved using the instruction `doubleElementAt()` in the Genome class, which retrieves a value from a given position on the genome.

```java
//**** Get the gene's data from a Genome
public void getDataFromGenome(Genome genome)
    throws SuspensionTestException{
    int i=0;
    betaY0=genome.doubleElementAt(i++);
    betaY1=genome.doubleElementAt(i++);
    betaY2=genome.doubleElementAt(i++);
    betaS0=genome.doubleElementAt(i++);
    betaS1=genome.doubleElementAt(i++);
    betaS2=genome.doubleElementAt(i++);
    k=genome.doubleElementAt(i++);
    KLim=genome.doubleElementAt(i++);
    KPrime=genome.doubleElementAt(i++);
}
```

The major code for the genome itself is contained in Genome.java. The genome object references the vector containing the genomic data. Each floating-point (double) variable in the genome is contained in a DoubleDataPoint object. This is an inner class of Genome, defined near the bottom of the Genome.java file. This also contains the minimum and maximum allowed values for the floating-point value.
8.21.4. The Mechanics of Mutation and Crossover

In each generation, each gene is given a performance score that is the average of all the performance scores for that one gene run over a number of random road surfaces. Once all the genes have been scored, selection, mutation, and crossover occur. The class responsible for forming the new generation from the old is the Breeder class, defined in the file Breeder.java. This class has a method, breed(), which takes a generation as a parameter and returns a new generation. In the runController() method of the EAController class, for example, the breed() function is called using the following code:

```java
Generation tempGeneration = breeder.breed(currentGeneration);
```

In the method breed() new genes are selected from the old generation and then mutation and crossover occur. The selection process is described in section 8.21.5 below. The method doMutations() controls the mutation process and doCrossovers() controls the crossover process.

The number of mutations is set as a given factor of the number of genes in a generation (possibly greater than one). This factor is determined by the MutationParameters class attribute, mutationRate. The number of mutations for a generation is determined by the code line shown below.

```java
int numMutations = (int)(mutationRate*gen.getNumGenes()*numDoublePerGene);
```

The number of mutations performed is calculated by multiplying the mutation rate by the number of genes and the number of floating-point values per gene. This gives a uniform mutation rate across all the floating-point values in the genome.

Genes are chosen at random for mutation. When chosen, the genome of that gene is formed using the geneToGenome() method. Mutation is performed on the Genome using the method, Genome.mutate(), defined in the file Genome.java. The Genome.mutate() method has a boolean parameter, doDouble. This ensures a certain number of large mutations, even in the cooler parts of the evolution. The mutate() method randomly determines a floating-point value to mutate and then calls the mutate() method in the class DoubleDataPoint to perform the mathematics of the mutation on that floating-point value. The method DoubleDataPoint.mutate() is shown in the code listing in figure 8.31.
public static final double PROP_HIGH=0.4, HIGH_MULT=1.945, LOW_MULT=0.37;

/**
   * Perform a mutation on a single floating-point data element in the gene
   ***/
public void mutate(boolean doDouble)throws SuspensionTestException{
    double x = value;
    if (doDouble){
        //**** Perform a large set mutation
        // This ensures some large mutation even when the evolution is cooling
        //System.out.println("DoubleDataPoint.mutate():
        //mutationParameters.getTemperature()="+mutationParameters.getTemperature()+"
        doDouble="+doDouble);
        if (Math.random()<PROP_HIGH){
            x=HIGH_MULT*x;
            //System.out.println(">");
        }else{
            x=LOW_MULT*x;
            //System.out.println("<");
        }
    }else{
        //*** Perform the "multiplicative lognormal perturbation" mutation
        double temp=mutationParameters.getTemperature();
        x = x*Math.exp(temp*NormRandom.random());
        //System.out.println("-");
    }
    //**** If the range endpoints are of opposite sign, then flip to negative
    // every so often
    // if (lowFactor*highFactor<0 &&
    Math.random()<mutationParameters.getProbFlipToNegative()){  
        System.out.println("Swap to negative");
        if (x<0){
            x=x*(highFactor/lowFactor);
        }else{
            x=x*(lowFactor/highFactor);
        // if
    }// if
    //**** Ensure that the mutations are within setlimits, lowFactor and highFactor
    if (x<lowFactor) {  
        x=lowFactor;
    }  
    if (x>highFactor) x=highFactor;
    setValue(x);
}

Figure 8.31 Code for Mutation using Gaussian Distribution

In order to generate a normal distribution of values, Sun’s Random.nextGaussian() method is used. Sun’s web documentation states that this method “returns the next pseudorandom, Gaussian (‘normally’) distributed double value with mean 0.0 and standard
deviation 1.0 from this random number generator’s sequence” (Sun). This was tested by the author in a program Java/Random/Demo.java. This was not a rigorous mathematical test, but, at first appearance, it produced results that were consistent with the Gaussian distribution claimed. In the line bolded in the listing in figure 8.31 the mutation procedure follows the “multiplicative lognormal perturbation” method of equation 2.12 (Dumitrescu et al., 2000).

In the Breeder class, the method doCrossovers() has similarities to doMutations(). The number of crossovers are determined by multiplying the crossover rate by the number of genes in the generation, as in the code below. Two distinct genes are chosen at random in Breeder.doCrossovers(). The genome for each of these is created and the Genome.crossover() method is called to perform the crossover on the two genomes. The listing of Genome.crossover() is shown in figure 8.32.

```java
/** Performs crossover between random bit-points **/
public void crossover(Genome genome2)
    throws SuspensionTestException{
    int s = numDataPoints();
    for (int i=0; i<s; i++){
        DoubleDataPoint DP1 = getDataPoint(i);
        DoubleDataPoint DP2 = genome2.getDataPoint(i);
        double v1=DP1.getValue(), v2=DP2.getValue();
        if (v1!=v2){
            //System.out.print("DP1="+DP1+" DP2="+DP2);
            double a= Math.random();
            double y1 = a*v1 + (1-a)*v2;
            double y2 = a*v2 + (1-a)*v1;
            DP1.setValue(y1);
            DP2.setValue(y2);
            //System.out.println(" => a="+a+" DP1="+DP1+" DP2="+DP2);
        }
    }
}
```

Figure 8.32 Listing of Genome.crossover()

The main code section that performs the mathematical steps of the crossover algorithm is shown in bold font in figure 8.32. Continuous crossover as discussed in section 2.12.2 was used. For each data point a random fraction is chosen and the corresponding data points on the separate genomes are mixed according to this fraction.
8.21.5. Selection

The first step in the selection process is the sorting of genes. Genes are sorted in the `sortGenes()` method defined in the Generation class. In the case of weighted sum optimization, the genes are simply sorted by fitness measure. In the case of Pareto optimization, the genes are sorted by dominance. The genes that are dominated the least number of times come first in the ordering.

Dominance is determined by Pareto ordering, as described in section 2.12.2, and so one gene dominates another if both its performance measures for “jerk” (comfort measure) and “rattlespace” (measure of capacity to stay within rattlespace limits) are greater than the others. There is an exception to this rule in the program used in this thesis: a gene with a negative value for either rattlespace or jerk is dominated by one that has positive values for rattlespace and jerk. (This reflects the notion that a negative score is “infeasible” and feasible systems are given preference to infeasible ones.) In determining dominance, the function `FitnessData.dominates()` is called, defined in the file Fitness.java. The listing of this routine is shown in figure 8.33.

```java
/**
 * Determines Pareto dominance
 */
public boolean dominates(FitnessData fit){
    boolean b=(fit==null);
    if (!b) {
        double j2=fit.getJerk(), r2=fit.getRattleSpace();
        if ((jerk<=0 || rattle<=0) && j2>0 && r2>0) {
            b=false;
        } else {
            if (((j2<=0 || r2<=0) && jerk>0 && rattle>0) {
                b=true;
            } else {
                b=(
                    (jerk>=j2 && rattle>=r2));
                if (b && jerk==j2 && rattle==r2) b=false;
            }
        } // if
    }
    return b;
}
```

Figure 8.33 Listing of the dominates() Method
The ordering within genes that are dominated the same number of times is weighted sum ordering. Figure 8.34 for example, shows 20 genes in a sample run producing the ordering shown. Here, “j:” precedes the jerk score and “r:” precedes the rattlespace score. The number of times a gene is dominated by other genes is shown as “front:”. (Binary insertion sort was used to perform weighted sum and Pareto sorting.)

<table>
<thead>
<tr>
<th>Ordering</th>
<th>Gene Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>(j:989.6, r:1000.0) front:0</td>
</tr>
<tr>
<td>1:</td>
<td>(j:991.0, r:991.9) front:0</td>
</tr>
<tr>
<td>2:</td>
<td>(j:991.3, r:987.2) front:0</td>
</tr>
<tr>
<td>3:</td>
<td>(j:991.6, r:932.2) front:0</td>
</tr>
<tr>
<td>4:</td>
<td>(j:993.1, r:928.0) front:0</td>
</tr>
<tr>
<td>5:</td>
<td>(j:987.1, r:1000.0) front:1</td>
</tr>
<tr>
<td>6:</td>
<td>(j:992.3, r:926.8) front:1</td>
</tr>
<tr>
<td>7:</td>
<td>(j:992.6, r:200.6) front:1</td>
</tr>
<tr>
<td>8:</td>
<td>(j:986.8, r:1000.0) front:2</td>
</tr>
<tr>
<td>9:</td>
<td>(j:989.7, r:964.6) front:2</td>
</tr>
<tr>
<td>10:</td>
<td>(j:992.2, r:870.2) front:2</td>
</tr>
<tr>
<td>11:</td>
<td>(j:982.4, r:997.7) front:3</td>
</tr>
<tr>
<td>12:</td>
<td>(j:987.7, r:967.6) front:3</td>
</tr>
<tr>
<td>13:</td>
<td>(j:992.1, r:862.2) front:3</td>
</tr>
<tr>
<td>14:</td>
<td>(j:991.3, r:918.2) front:4</td>
</tr>
<tr>
<td>15:</td>
<td>(j:989.9, r:914.7) front:6</td>
</tr>
<tr>
<td>16:</td>
<td>(j:990.4, r:913.0) front:6</td>
</tr>
<tr>
<td>17:</td>
<td>(j:992.3, r:-518.8) front:17</td>
</tr>
<tr>
<td>18:</td>
<td>(j:997.5, r:-18700.8) front:17</td>
</tr>
<tr>
<td>19:</td>
<td>(j:949.6, r:-16514.0) front:18</td>
</tr>
</tbody>
</table>

Figure 8.34 Example of Pareto Sorting

The screenshots in figure 8.35 show examples of Pareto fronts for a passive and a skyhook suspension. Each point on the graph represents the response of a particular suspension. It is clear that the skyhook dominates the passive. For the example, the same set of roads was used with each of the separate genes. When random road sets are used, the Pareto fronts are less distinct.

Figure 8.35 Examples of Pareto Fronts
After sorting, control is passed to the Breeder.breed() method, in the file Breeder.java. This method performs selection before mutation and crossover. The following parameters of the MutationParameters class control the selection process:

- numberPerGeneration, the number of genes in a generation,
- proportionKept, a factor that retains a certain proportion of a generation,
- weightedFreqBottom is used in scaling the frequency weightings for selection (described below).

Of these, the values numberPerGeneration and proportionKept are varied in the AutoEvolution scheme as part of the “cooling” process over a number of generations. Following the elitist approach, as defined in section 2.12.2, a certain proportion of the highest scoring genes from the previous generation are included in the next (this is why sorting is needed, but sorting is also useful when displaying a graphical snapshot of the previous genes’ performance). The parameter proportionKept contains this proportion, and it diminishes as the evolutionary process cools.

Before selection, genes are given a weighting that determines their frequency in the next generation. The probability of selecting the $i^{th}$ gene is given by equation 8.16, which is similar to equation 2.11 except that weightings, $w$, are used instead of pure fitness values.

$$ p_i = \frac{w(x^i)}{\sum_{j=1}^{n} w(x^j)} $$

Equation 8.16

The weightings are calculated from fitness scores but are modified (as explained in the next paragraph) to give a better spread of frequency weighting between the top and bottom scoring genes. The Java method that calculates the probability weightings is Breeder.findFrequencies(). This function calls in turn one of two methods: in the case of Pareto optimization, findFrequenciesUsingPareto() is used, and findFrequenciesForWeightedSum() is called if weighted sums are used for the optimization. The probability weighting, once calculated, is placed in a gene object, using the gene.setFrequency() method.
The code for calculating the probability weightings in the case of weighted sum selection is shown in figure 8.36. First the following factor is calculated,

\[ F_i = \frac{f(x_i)}{1000} - 1. \]

The function \( f \) is the fitness score of gene \( x_i \). This calculation is shown in bold in the listing in figure 8.36. Since the highest fitness score is 1000, the maximum value of \( F \) is 0. This value is then weighted according to the following function:

\[ w_i = e^{F_i} - L. \]

The value of \( L \) is the least value of \( e^{F_i} \) minus a certain small value, \( \text{weightedFreqBottom} \) (in the \( \text{MutationParameters} \) class). If this is negative, then \( L \) is zero. This “normalizes” the weightings so that the lowest weighting is less than \( \text{weightedFreqBottom} \).

```java
private double findFrequenciesForWeightedSum(int numToChooseFrom) throws SuspensioTestException {
    //*** This determines the relative weights of the positive and non-positive genes
    double weightedFreqBottom = parameters.getMutationParameters().getWeightedFreqBottom();
    //**** Perform the transformation of fitness values
    //**** First find exp(-1+fit/1000) and least
    double least = 300, sumFrequencies = 0;
    System.out.println("numToChooseFrom="+numToChooseFrom);
    for (int i=0; i<numToChooseFrom; i++) {
        Gene gene = (Gene) currentGeneration.getGene(i);
        double fit = gene.getFitnessData().weightedFitness();
        double F = Math.exp(-1d+fit/1000d);
        gene.setFrequency(F);
        if (F<least) least = F;
        sumFrequencies+=F;
    }
    //**** Find the bottom, and subtract from weights - if >0
    double bottom = least - weightedFreqBottom;
    System.out.println("least="+MakeNumberString.DoubleToString(least,2)+" bottom="+MakeNumberString.DoubleToString(bottom,2));
    if (bottom>0) {
        sumFrequencies = 0;
        for (int i=0; i<numToChooseFrom; i++) {
            Gene gene = (Gene) currentGeneration.getGene(i);
            double F = gene.getFrequency();
            F = F-bottom;
            sumFrequencies += F;
            gene.setFrequency(F);
        }
    }
    //*** show results
```
In the case of Pareto optimization the calculation of weightings is based on a factor, $F$, which is simply calculated as,

$$F_i = \frac{1}{\text{dom}(x^i) + 1},$$

where $\text{dom}(x^i)$ is the number of times that a gene is dominated by other genes. A value, $L$ is found, which results by subtracting $\text{weightedFreqBottom}$ from the weight of the least scoring gene. If this is negative, then $L$ is zero:

$$L = \max\left(\min_i F_i - B, 0\right)$$

where $B$ represents $\text{weightedFreqBottom}$. The probability weightings are then calculated as,

$$w_i = F_i - L.$$

After this, if the rattlespace or the jerk factor is negative, the weighting is further discounted.

The complete code for the calculations of the Pareto weightings can be found in `Breeder.findFrequenciesUsingPareto()`.

Except for a small number of highly scoring genes, most genes in a generation are chosen. (Note that the number of genes in the next generation may be larger or smaller than the number of genes in the previous.) The line that determines this number of genes chosen using elitism is shown in bold in figure 8.37.
Generation tempGeneration =
    new Generation(currentGeneration, parameters, version);
    // Number in the old generation
    int numInOldGeneration = currentGeneration.getNumGenes();
    double ProportionMustInclude =
        parameters.getMutationParameters().getProportionKept();
    int mustInclude = (int)
        (ProportionMustInclude * numInOldGeneration);
    /**
     * The number in the new generation
     * may be different from the number in the previous.
     * This number is a parameter in MutationParameters
     */
    numGenes =
        parameters.getMutationParameters().getNumberPerGeneration();
    //System.out.println("***** mustInclude="+mustInclude+" numGenes="+numGenes+" numInOldGeneration="+numInOldGeneration);
    /** Form frequency weightings for genes.
     * This calls a method that forms different weightings depending on whether
     * Pareto optimization or weighted sums are used.
     * findFrequencies(numInOldGeneration);
     */
    //** Make a new generation
    /**
     * Perform SELECTION.
     * This calls a routine that does the selection,
     * either Pareto or weighted sum.
     */
    randomSelectionUsingFrequencies(mustInclude, numGenes, tempGeneration);
    //********** MUTATION
    doMutations(tempGeneration);
    //********** CROSSOVER
    doCrossovers(tempGeneration);
    //tempGeneration.printCurrentGeneration();
    return tempGeneration;
}

Figure 8.37 Main Calling Routine for the Breeding Processes: selection, mutation and crossover

The routine that calculates the probability weightings is, findFrequencies(), and the line that calls this method has been highlighted in figure 8.37. This calls the methods to create the frequencies as already explained.

The routine shown in figure 8.37 also calls the procedure to perform selection, randomSelectionUsingFrequencies(). The listing for this routine is shown in figure 8.38. This begins by copying the highest performing genes directly. The number of genes copied is given by the variable, mustInclude. These are simply copied from the highest
scoring genes in the previous generation. Next, the weights are summed, and a cumulative sum is placed in an array. This assists with the selection process. A uniform random variable is generated and is used for selecting an element from the array corresponding to equation 8.16. This code has been tested using test routines to display cumulative sums and selected values to the “DOS window”.
Random Selection using frequencies.

**private void randomSelectionUsingFrequencies(int mustInclude, int numGenes, Generation tempGeneration)**

```java
// System.out.println("Current Generation: ");
currentGeneration.printCurrentGeneration();

/**
   * Copy some of the highest performing genes directly.
   * This copies mustInclude number
   */
int numInGen=currentGeneration.getNumGenes();
System.out.println("numInGen="+numInGen+ "
mustInclude="+mustInclude);
for (int i=0; i<mustInclude; i++){
    Gene gene=currentGeneration.getGene(i);
    Gene gene2=GenomeConverter.copy(gene);
    if (i<mustInclude) tempGeneration.addGene(gene2);
}

//**** Find cumulative sums and total sum.
// The cumulative sum is placed into an array: cumulative.
double[] cumulative=new double[numInGen];
double totalSum=0;
for (int i=0; i<numInGen; i++){
    double freq=currentGeneration.getGene(i).getFrequency();
totalSum+=freq;
cumulative[i]=totalSum;
//System.out.println("i:"+i+" freq="+freq+" totalSum="+totalSum);
}
int size=cumulative.length;
//showCumulative(cumulative);

//**** Select for all
for (int i=mustInclude; i<numGenes; i++){
    //**** Find a random number within 0 to totalSum
double r=totalSum*Math.random();
    //**** Binary search for the relevant gene number
    int geneNum=binarySearch(r, cumulative, size);
    //System.out.println("r="+r+" geneNum="+geneNum);
    //**** Add the gene to tempGeneration
    Gene gene=GenomeConverter.copy(currentGeneration.getGene(geneNum));
   //System.out.println("\ncurrentGeneration.getGene("
+geneNum+ ")+" + currentGeneration.getGene(geneNum).toFullString());
gene.isFitnessDetermined();
tempGeneration.addGene(gene);
    }
//System.out.println("Next Generation: ");
tempGeneration.printCurrentGeneration();
```
8.22. Calibration of Digital Accelerometer

The most difficult factor to measure accurately is “chassis” movement, and this factor is crucial in some control algorithms. The potentiometer only measures the relative movement of the “chassis” and “wheel”. In an attempt to improve the accuracy of the measure of chassis movement it was decided to add a digital accelerometer to the rig.

An ADXL345 was chosen. These are extremely small and a “breakout board” made the soldering and experimentation feasible (SEN-09156 from sparkfun electronics, http://www.sparkfun.com/commerce/product_info.php?products_id=9156). The breakout board is shown in figure 8.39. It is somewhat odd that despite the sophistication of these devices, much of the code found for their application does not perform full error trapping and/or uses timing loops rather than flag reading to determine timing. While code for the Arduino could be found, this did not work on the ATmega644 on the STK500 development board. Code was eventually developed that successfully communicated with the accelerometer. (Refer to the word file “AVR Notes.doc” in the directory PhD\Experiment\Electronics\Atmel AVR\My AVR Notes).

![Accelerometer Breakout Board](image-url)
Digital communication was performed using the I²C protocol. I²C stands for “Inter-Integrated Circuit” which is an industry standard developed by Philips. The AVR version is called the “2-wire Serial Interface” (TWI). I²C is designed for short communication on a single board. Long wires connected to sensors off-board may malfunction due to signal degradation. It was found that the board would not run at 400 Hz with a 1.5 m ribbon used between the MCU and the accelerometer. It was decided therefore to use a high-frequency video cable for the connection and surround the accelerometer with a Faraday cage, as done with the potentiometer above.

The ADXL345 is a 3 V device and therefore presents problems when connecting to a 5 V MCU. Devices that are perfect for this purpose are “bi-directional level shifters”. These devices allow the MCU to run at 5 V.

A small program was developed to test the statistical properties of the ADXL readings. The main factor affecting the reading is the supply voltage to the accelerometer. If this is not clean the reading will not be clean. When an off-board box was used for readings the readings consistently had a high standard deviation, of about 5 (with a sample of size 1,000). When a large capacitor was placed across the 3 V to the board, this reduced to about 1.2. The problem, of course, is that a large capacitor takes a long time to charge up. (A 10 μF capacitor was also placed across the 5 V power pins of the analog accelerometer.) The device ran successfully on the rig with a connecting line of about 5 m at 100 kHz. The device was placed on the rig at the end of the upper arm, as shown in figure 8.40.

![Figure 8.40 ADXL345 Housed on the Rig](image)
For this setup a new, slightly more accurate potentiometer was used and the approximate value for $\alpha$ in equation 6.1 with the new setup is,

$$\alpha_d \approx -0.01690 \text{ m/V}.$$  

(Refer to the data file Volts Distance Mass Digital Accelerometer.xlsx in the folder PhD\Experiment\Rig.)

The input ADC and PWM conversion equations are still given as in equation 6.2 and equation 6.5 with the same conversion factors (see equation 6.3 and equation 6.6).

The two accelerometers need to be calibrated. The same steps as used before for the analog accelerometer can be used again to calibrate both accelerometers. The same accelerometer experiments as performed in section 6.3.5 were carried out using the digital accelerometer output converted to a voltage so that the data could be read using the digital oscilloscope. This produces the following version of the equation for conversion of the accelerometer output,

$$\vec{v}_A = \beta_D (v_{acc} - v_0) = \beta_D (\rho x_D - v_0),$$

where $\beta_D$ is the digital accelerometer conversion factor and $x_D$ is the digital accelerometer output value. By visual alignment of graphs (using data file DigitalAcc01.txt) a conversion factor of 205.7 was found, while the calculated average of the proportions was 198. Thus a conversion factor of $\beta_D = 200$ was used. From visual examination of the graphs the accelerometer latency seems to be very roughly of the order of 10 ms, and is comparable to the latency of the analog accelerometer. This verifies at least the accelerometers against each other.

As a check on this value, the conversion factor was calibrated against the new distance measure (by multiplying by $\alpha / \alpha_d$), giving $\beta = 610$, and the acceleration value from the analog accelerometer was compared with the digital. The two signals were superimposed on the oscilloscope, verifying the calibration of both accelerometers (as shown in figure 8.41, with slight offset to clarify graphs).
As described in section 6.3.6, the effective spring rate can be estimated from a graph of acceleration minus a constant times velocity versus velocity. The rate that allows the graph to most closely resemble a straight line is an estimate of the spring rate (in this case, the spring rate times the factor used to keep the acceleration graph within the 5 V range). The effective spring rate is estimated at 205 s$^{-1}$.

In the end it was determined that the digital accelerometer did not improve on the accuracy enough, and introduced too much extra latency to warrant inclusion in the final test rig. Perhaps if the digital accelerometer was on the same circuit board and hence able to run faster, or if a faster communication protocol is used, the digital accelerometer might have warranted inclusion. No doubt these properties will improve over time and digital accelerometer readings will be superior but, in this experiment, the digital accelerometer was dropped for later test runs.

One benefit, however, of the digital accelerometer is the verification of the analog accelerometer by comparing signals produced. The output voltage to the damper however must be scaled according to the voltage vs distance factor, of equation 6.1, because of the new potentiometer, and the inverse function needs to be reworked.
8.23. Physical Experiment Components and Some Code

Figure 8.42 Peter Tkatchyk with the Physical Rig

Figure 8.43 RD-1005-3 Damper
Figure 8.44  Lord Controller, RD-3002-0 (Lord, 2008)

Figure 8.45  PoScope Input Hardware
float prevVoltage=0;
void KalmanVoltage(double x){
  if (firstTimeV){
    Vav=x; prevVoltage=x;
    Vav0=x; // or could use Vav0=x; assuming equilibrium at start
    checkDistVoltage();
    firstTimeV=0;
  }else{
    Vav=alphaV*x+(1-alphaV)*(Vav+VelVAv);
    Vav0=alphaV0*x+(1-alphaV0)*Vav0;
    prevVoltage=x;
  }
}
void KalmanVel(double x){
  if (firstTimeVel){
    VelVAv=x;
    firstTimeVel=0;
  }else{
    // Combine measure, x, with prediction based on near constant acceleration
    // Acceleration adjustment - as per Kalman filter
    VelVAv=alphaVelV*x+(1-alphaVelV)*(VelVAv+(XAcc-XAcc0)*step*530.0*2.0*step);
  }
}
void KalmanAcc(double x){
  if (firstTimeAcc){
    XAcc=x;
    XAcc0=x; //(4*x+XAcc0)/5; // or could use XAcc0=x;
    assuming equilibrium at start
    checkAccVoltage();
    firstTimeAcc=0;
  }
}
XAcc = alphaAcc * x + (1 - alphaAcc) * XAcc;
XAcc0 = alphaAcc0 * x + (1 - alphaAcc0) * XAcc0;

```java
private static double damperInverseInterceptInt = 0.022397,
    damperInverseInterceptSlope = -0.000169615,
    damperInverseSlopeInt = 0.00026872,
    damperInverseSlopeSlope = 6.22226E-5;
public int DamperMode = 0;
/**
 * Finds the n value for a given velocity and acceleration
 * Using simply a linear approximation
 * Note that the class attribute, DamperMode,
 * returns the condition:
 * 0 in linear portion
 * -1 outside passivity constraint
 * 1 acceleration cannot be reached - too large
 * least value n to supply closest available acceleration is
 * 2 acceleration cannot be reached - too small
 * output is zero - some damping force is supplied anyway
 * @param acc input acceleration
 * @param vel input rate of change of voltage
 * @return PWM value
 */
private double damperInverse01(double acc, double vel) {
    DamperMode = 0;
    if (acc > 0) {
        if (vel < 0) DamperMode = -1;
    } else {
        if (acc < 0) {
            if (vel > 0) DamperMode = -1;
        } else {
```

**Figure 8.47 On-Board State-Estimation Code**

//***** Velocity Test ***************
//***** Comment out in final version ***************
// Note that this overwrites the voltage measure - Vav
// This code must be commented out in the final version
Vav = prevVoltage + VavStep;
if (Vav > 4000) {
    Vav = 4000;
    VavStep = -fabs(VavStep);
}
if (Vav < 100) {
    Vav = 100;
    VavStep = fabs(VavStep);
}
prevVoltage = Vav;
//**************************

**Figure 8.48 Code for Generating Triangle Wave**
acc=-acc; vel=-vel;
}
}
}
double num=0;
if (DamperMode==0){
    double intercept=damperInverseInterceptInt+damperInverseInterceptSlope*vel;
    double slope=damperInverseSlopeInt+damperInverseSlopeSlope*vel;
    double accComp=C_E*vel;
    if (acc>accComp){
        DamperMode=1;
        acc=accComp; // mode 1
    }
    num=(acc-intercept)/slope; // inverse - slope is never 0
    if vel positive
        if (num<0){
            DamperMode=2;
            num=0; // mode 2
        }
    return num;
}

Figure 8.49 Damper Inverse Function

8.24. AVR C Code for Crossover Removal

The following shows the AVR C code for the crossover removal, as discussed in section 6.5.2.

/**
Ian Storey
Handles crossover jerk reduction
copyright 26-Jan-2011
**/
double cMax=50; //spring constant was 35
double JERK_CROSS=4000, ALPHA=2;

/**********************
    Find the jerk required by crossover reduction
    Finds TC, JX
***********************/
double crossoverLimitAcceleration(double targetDamperAcc, double sv, double sa, double prevDamperAcc){
    double outputAcc=targetDamperAcc;
    //***** Calculate TC and TX
double TX=0;
    if (sa!=0){
        TX=-sv/(ALPHA*sa);
if (TX>0){ // There IS a danger of impending crossover
double JX=-prevDamperAcc/TX; // jerk needed for crossover
    //*** Damper acc <0
    double jerk=(targetDamperAcc-prevDamperAcc)*invStep;
    if (JERK CROSS<=JX) {
        if (jerk<JX){
            jerk=JX;
            outputAcc=prevDamperAcc+jerk*step;
            //printf("."); // for testing
        }
    }
}
    //*** Damper acc >0
    if (-JERK CROSS>=JX) {
        if (jerk>JX){
            jerk=JX;
            outputAcc=prevDamperAcc+jerk*step;
            //printf("*"); // for testing
        }
    }
    return outputAcc;
}

double prevSpringAcc;

//**********************
// public jerkForStep
//double prevS=0, prevSV=0;
double accWithCrossoverReduction(double sv, double sa, double prevDamperAcc, double targetDamperAcc) {
    double jerk=0;
    response = 0;
    //**** This force must be opposite in sign to stroke velocity
    // in order to lie inside the passivity constraint
    if ((targetDamperAcc>=0 && sv>=0) || (targetDamperAcc<=0 && sv<=0)) {
        // Target is outside outside passivity constraint
        // just output zero force
        targetDamperAcc=0;
    } else {
        //**** find the crossover jerk
        targetDamperAcc=crossoverLimitAcceleration(targetDamperAcc, sv, sa, prevDamperAcc);
        if ((targetDamperAcc>0 && sv>=0) || (targetDamperAcc<0 && sv<=0)) {
            // outside passivity constraint
            targetDamperAcc=0;
        } else {
            // check inside range of maximum damping rate
            // this is a separate crossover removal that can be dispensed wth
            if (cMax!=0) {
                double maxDampAcc=-cMax*sv;
                // the following works because
                // targetDamperAcc and sv are of opposite
                // therefore targetDamperAcc and maxDampAcc
                // are of the same sign
                if (fabs(targetDamperAcc)>fabs(maxDampAcc)) {
                    //
```java
    targetDamperAcc=maxDamperAcc;
    
    }
  }
  }
  }
  }
  }
  return targetDamperAcc;
```
References


Donahue, M. D. (2001) Implementation of an Active Suspension, Preview Controller for Improved Ride Comfort, *Department of Mechanical Engineering, University of California at Berkeley*.


Hyvärinen, J.-P. (2004) The Improvement of Full Vehicle Semi-Active Suspension Through Kinematical Model *Faculty of Technology, Department of Mechanical Engineering, University of Oulu*, Oulu, Finland, p. 159.


Kasprzak, E. M., Lewis, K. E. and Milliken, D. L. *Tire Asymmetries and Pressure Variations in the Rad
t/Milliken Nondimensional Tire Model*, in SAE 2006 Automotive Dynamics, Stability and Controls Conference and Exhibition. 2006. Novi, MI, USA.


Robson, J. D. and Dodds, C. J. The Response of Vehicle Components to Road Undulations, in Fisita Congress. 1970.


Sun Class Random, [http://download.oracle.com/javase/1.4.2/docs/api/java/util/Random.html](http://download.oracle.com/javase/1.4.2/docs/api/java/util/Random.html), 21 April, 2011.


