Modeling and Simulation of Missile Launcher System Wear During Captive Carry

by

William Douglas McGinnis

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 2, 2014

Keywords: Hellfire, Missile, Dynamics, Wear, Modeling

Approved by

David Beale, Chair, Professor of Mechanical Engineering
Robert Jackson, Professor of Mechanical Engineering
George Flowers, Professor of Mechanical Engineering

Abstract

Across the branches of the United States and foreign militaries, missiles are used for a variety of missions. Many of these missiles mount to their respective vehicles by means of a rail system. While the service life of an as-designed rail system in a laboratory or other well-defined environment is generally known, the wear rates of manufactured systems at the upper and lower limits of specified clearance, and under varied vibrational loads in arbitrary environments has not been well established.

By leveraging multibody vibrational simulation and mathematical wear models, a given rail system has been assessed for its robustness to a variety of parameters. This was done by developing a modification of the commonly used Archard wear model, and by developing a wear prediction plugin for the multibody simulation software package, MSC.ADAMS.

The parameters for the wear model were first generated using a simple laboratory test on a Friction and Wear tester, where 3 aluminum samples were fretted against a square coupon to generate experimental wear mass data points. Using these data points, a wear model was developed, and then validated for the simulation of wear in rattling contacts with a simple shaker table fixture of a square tube with a free-floating cube inside it. This fixture was vibrated for a period of time and the experimental wear mass of the cube was compared to the predicted wear mass from ADAMS.

Finally, a multi-body simulation of a Hellfire missile launcher system was developed in order that generalized relationships could between system configurations and total service life could be found. This simulation model was validated with another shaker table experiment in which an instrumented missile system was vibrated in each axis at key frequencies found in the MIL-810G standard for accelerated vibration testing of military hardware. The

experiment consisted of 4 sine tones in each of the three axes, and through an iterative tuning process, the ADAMS model's prediction of the frequency spectrum of the bulk missile motion was matched closely to the experimental spectra.

After the model's dynamics were validated, a third and final validation experiment was conducted to validate the wear predictions for 2 critical components of the system. In this experiment a full missile system was mounted to a shaker table and vibrated to create wear on the mid and rear shoes. This experimental wear was compared to predictions from ADAMS to finally validate the model. The errors of the two wear mass predictions were 6.50% and 3.35% respectively.

Once the multibody simulation wear prediction model was validated, a missile launcher systems of various dimensions and under varied vibration loads were simulated, and their wear rates were compared to the reference configuration upon which the stated service life and maintenance schedules are based. With this information a service life penalization methodology was proposed to more effectively schedule replacement of missile launcher rails in the field.

Acknowledgments

I would like to thank and acknowledge his advisor, Dr. Beale, for countless advising meetings. Dr.'s Flowers and Jackson for notes and guidance, and Brittany Griffin for providing scope and direction. This body of work was made possible by the hard work and help of Sara Pope, Justin Harrison and Haoyue Yang, who all assisted in the various experiments conducted.

Finally, I thank Doug and Teri McGinnis for enabling and encouraging all the many years of my education, and Claire Cooper for supporting me through the end.

Table of Contents

List of Figures

List of Tables

List of Terms

Symbol	Description
MBS	Multibody Simulation
FEA	Finite Element Analysis
DOF	Degree of Freedom
F	Force
ω	Penetration Depth
c_{max}	Maximum Damping
d	Penetration Depth
k	Contact Stiffness
e	Force Exponent
q_0	Trigger Distance
R	Effective Radius
E	Young's Modulus
E^*	Effective Young's Modulus
ν	Poisson's Ratio
A	Contact Area
P	Contact Pressure
H	Hardness
S_y	Yield Strength
C_R	Coefficient of Restitution
$\dot{\delta}$	Penetration Velocity
F_f	Frictional Force
μ	Friction Coefficient

N	Contact Normal Force
v_{slip}	Contact Slip Velocity
VDV	Vibration Dose Value
W	Wear Volume
s	Sliding Distance
k_{abr}	Abrasive Wear Coefficient
k_{adh}	Adhesive Wear Coefficient
k_{imp}	Impact Wear Coefficient
I	Total Impact Energy
α	Proportion of Impact Energy Dissipated by Slip
σ	Average Normal Contact Surface Stress
K	Dimensional Lumped Wear Coefficient
C_1	1st Parameter of Modified Archard Model
C_2	2nd Parameter of Modified Archard Model
C_3	3rd Parameter of Modified Archard Model
J	Moment of Inertia
T	Torque
$\ddot{\theta}$	Angular Acceleration
r_{avg}	Average Radius
M_{wear}	Wear Mass
ρ	Density
k_{rs}	Rattle Space Coefficient
A_{ti}	Cross Sectional Area of Test Item
A_{encl}	Cross Sectional Area of Enclosure
T_{SL}	Designed Service Life
$T_{SL,real}$	Adjusted Service Life
K_{SL}	Service Life Penalization Factor

Chapter 1

Introduction

1.1 Introduction

The AGM-114 Hellfire air-to-surface missile is one of the most prevalent and comprehensive weapon systems in the US Army's arsenal. It's versatility and efficacy has propelled is evolution from its original design intent of a helicopter launched tank-buster ("HELicopter FIRE and forget"), to being utilized on not only rotary and fixed wing aircraft, but also sea vessels and some land based vehicles. These missiles have found use on Apache, Super Cobra and Kiowa helicopters, as well as Predator unmanned combat air vehicles. Hellfire missiles have also been successfully fired from Humvees, C-130 Hercules and coastal assault boats[28].

Figure 1.1: A predator unmanned combat air vehicle launching a Hellfire missile

The roughly 100 pound missiles measure approximately 64 inches in length and 7 inches in diameter, and contain as much as 20 lbs of high explosive. A semi-active laser homing radar seeker controls the missile as it approaches its target, as far as 5 miles away, at 950

mph. Hellfire missiles in their various iterations have flown in action in Operation Just Cause, Operation Desert Storm, Operation Allied Force, Operation Enduring Freedom and Operation Iraqi Freedom[28].

Hellfire missiles mount to their vehicle, whatever it may be, by means of a rail system. Within a specified design life, it is expected that the rail will properly restrain the missile in flight and guide it upon launch. In the varied environments of modern military engagement, however, the rails have been occasionally observed to wear much more quickly than expected. This accelerated rail wear poses and immediate threat to the warfighter, as a jammed missile can put those in the helicopter in grave danger.

Figure 1.2: 2 Hellfire missiles shown loaded into their mounting rails

In light of this risk, it is clear that a means of estimating the usable lifetime of the rail system in any environment or configuration is critical to the safety and lethality of the warfighter.

Despite being commonly overlooked, wear and tribology in general are absolutely critical to nearly every kind of mechanical system. Wear can massively impact the strength, performance, and safety of a system, and without a good understanding of the principles which govern it, engineers are incapable of accounting for the potentially disastrous consequences that can result. The danger of wear in design and maintenance stems from the difficulty of detecting, predicting or simulating it.

In order to facilitate the analysis in wear among practitioners, a simplification of the simulation of wear must be developed. Similar to the introduction of finite element analysis for failure modeling, a rapid and simple process and methodology for predicting wear in multi-body systems using already existing and accepted multi-body codes will introduce this critical analysis into fields that will benefit from it, but otherwise may not have the expertise or motivation to pursue it.

This thesis develops and validates a methodology for predicting wear on arbitrary missile launcher systems, and proposes an application of the results of these predictions for the existing fleet. The same methodology can be directly extended to other systems of interest.

The document is divided into 7 chapters, the first of which is this introduction.

Chapter 2 contains background information on multibody simulation, which problems are well suited to it, how to model contact between bodies using it, and how to select the best integrator and formulator for a problem.

Chapter 3 is background information on wear modeling and simulation literature, and includes a proposed modification of the commonly used Archard wear equations.

In Chapters 4 through 6, 3 case studies with experiments are presented which show the development of different multibody simulation models, and the validation of the wear models which can be applied to them.

Case Study 1: Persistent Contact The first case is the simplest, the case of a persistent contact. In this constant load, oscillatory motion wear case, laboratory experiments are used to validate the use of multi-body code in to accelerate the simulation of various

wear cases. Because tight control over contact loads and sliding distance are possible with this setup, it can be used to develop a model for the dimensional lumped wear coefficient in terms of normal load. Further, the quality of control and measurement of loads and speeds in this experiment means that a model for the friction of the contact can be developed.

Case Study 2: Intermittent Contact The second case is an intermittent contact. In this case, a 1 DOF input incites a 6 DOF response in a rattling contact. With the exact mode of wear ambiguous, this experiment is intended to validate the use of multibody code and the wear model developed in case study 1 to predict wear in systems with complex rattling motion.

Case Study 3: Hellfire Missile Launcher System The final case study is the Hellfire missile launcher system. In this system, a mix of persistent and intermittent contacts encounter abrasion, adhesion and possibly impact. The modeling techniques developed in the previous case studies are applied to this real-world system to develop a service life penalization methodology which proposes an amendment to the designed service lives of parts depending on their actual manufactured dimensions.

It is important to note that these 3 case studies are ordered in terms of complexity, not chronologically. In actuality, they were all completed simultaneously, and information from each case study informed the others. For example, the friction and wear models developed in case study 1 were used in case studies 2 and 3, and the contact models developed in case study 3 were used in case studies 1 and 2.

Chapter 2

Multibody Simulation

2.1 Introduction

Multibody simulation (MBS) is a numerical method in which a system is broken down into constituent rigid bodies. These bodies interact with each other via kinematic constraints and force elements. Kinematic constraints can include joints such as cylindrical joints or planar connections, while force elements can include contact models, unilateral friction models, and springs. With the system modeled, a solver is used to formulate and simulate the resultant equations of motion numerically. The rapid development time line of multibody simulation has led to its prevalence in the automotive and prosthetic industries, as well as in academic research [12][36][41].

Many numerical packages exist for the integrated modeling and simulation of multibody systems, including ADAMS[3], LMS DADS[5], Working Model[8], and Chrono:Engine[7], among others. Further, a number of specialized software packages exist which are used to model specific types of problems, such as BikeSim, CarSim, and TruckSim [6].

The primary difference between the various multibody simulation softwares, mathematically, is how they handle contact mechanics. Some use penalty methods, attempt adapt Hertzian contact mechanics to arbitrary shapes, or simply use linear springs (bed-of-springs).

The second major differentiator between multi-body simulation packages is the solver. A solver is a piece of mathematical software which 'solves' a problem. In the context of multibody simulation, the problem to be solved is the systems equations of motion, so the difference between packages is often the integrator used to propagate the dynamics, and the method used to formulate the equations themselves.

In this thesis, ADAMS was used exclusively.

ADAMS is short for "Automated Dynamic Analysis of Mechanical Systems". It is a multibody dynamics and simulation software developed by Mechanical Dynamics Incorporation and now owned by MSC Software Corporation.

2.2 Multibody Simulation vs. Finite Element Analysis for Wear Simulation

Before analyzing how exactly multibody simulation can be used to simulate wear in complex systems, it is first necessary to determine whether multibody simulation is the correct tool for this task. In practice, the primary difference between tasks for Finite Element Analysis and multibody simulation is the scale of the system to be analyzed. As shown in Figure 2.1, there are 3 scales: system, subsystem and components. The Hellfire missile launcher problem is a subsystem scale problem, where the helicopter is the system, and the rail alone is a component. For any problem other than component scale, the more precise FEA tends to be prohibitively slow, so multibody simulation is generally used instead.

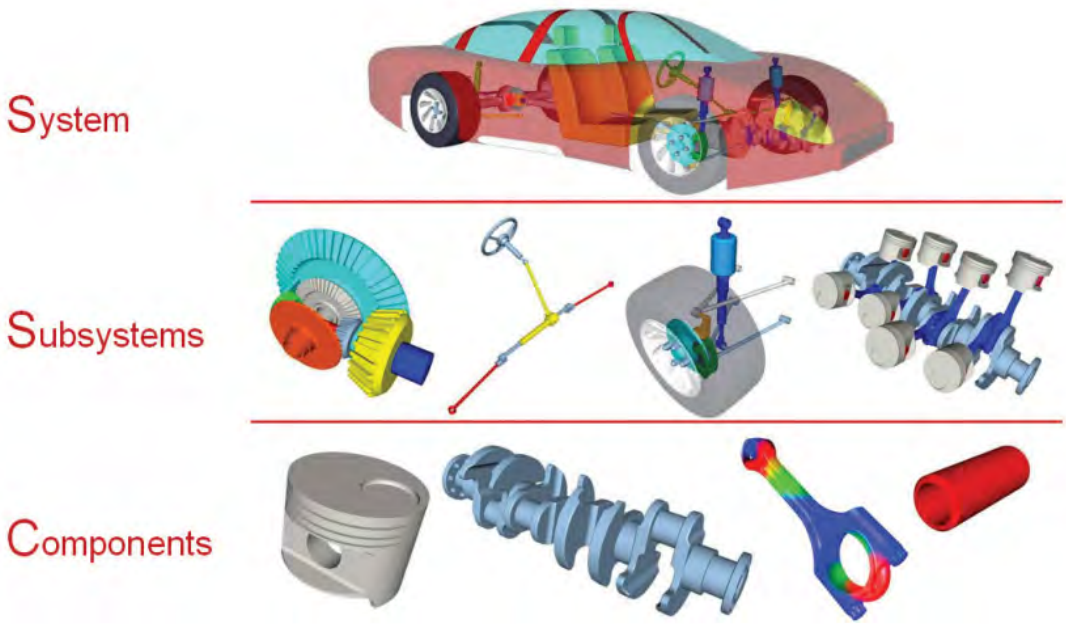

Figure 2.1: The three scales of system design[2]

The problem of simulating wear in multibody systems has been addressed in literature. Work done in this area is often either pertains to the design and manufacture of knee replacements [49][36] or to the modeling of revolute joints with clearance [17][35]. Almost all approaches previously taken combine Archard's wear models [11] and a either a Finite Element (FE) or a multibody approach to simulate the wear.

To the user, the primary differences between an FE and a multibody based approach are development time and run time. FE models are generally much slower to create and much slower to run than multibody simulation models. This time benefit for MBS comes at the cost of accuracy and extensibility; where multi-physics effects and localized geometry changes can be accounted for relatively simply in FE models, these same effects are incredibly difficult to account for in MBS. The question therefore becomes whether the simplified models which can be quickly and easily created in multibody software can be used to simulate wear to a sufficient accuracy, or whether the time investment of an FE model is required to achieve sufficient accuracy.

Zhao and Fregly[49] used a multibody code to simulate wear in an artificial knee using a variety of techniques. They found that surface evolution (the changing of model geometry over the course of the simulation due to wear) had large impact on the predicted wear profile and wear depth predicted, but that it had nearly no impact on the total predicted wear volume. This implies that if the quantity of concern is the total wear volume or mass, that a multibody code without surface evolution is a suitable approach.

In cases where experimental data was available, the Archard model approach was found to be accurate for the systems analyzed [49][35].

In light of these cases, the approach taken was a multibody based simulation model which uses no surface evolution and an Archard wear model was deemed suitable, as will be seen, however, the Archard models for wear had to be modified slightly to better match experimental data gathered in each case study.

A more in-depth discussion of wear modeling and Archard's wear equations will be presented in the next chapter.

2.3 Contact Mechanics

Being one of the primary differentiators between MBS packages, the field of contact mechanics is critical to the understanding of modeling multibody systems. Generally, contact mechanics can be separated in to two primary types of contact, persistent and intermittent.

Intermittent contact is an impulsive contact that happens over a short period of time. An example of this is a ball bouncing off a wall. The two bodies are separate for a period of time, then a short impulse of contact occurs when the ball hits the wall. Intermittent contact itself is separated into two regimes: compression and restitution. In the compression regime, the bodies continue to move towards each other even after contact, compressing the material and converting kinetic energy into stored potential energy. In the restitution regime, the potential energy is both dissipated by internal damping and converted back into kinetic energy, as the bodies separate[33].

Persistent contact is when two bodies are in contact for a longer period of time. An example of persistent contact is a ball resting on a table. The separation velocity of the ball is zero, even after contact, therefore it is a persistent contact [33].

The contact mechanics problem which must be solved for the multibody dynamic simulation is the calculation of normal and tangential reaction forces given the dynamics and interference of each body. For both persistent and intermittent contact, there are very few cases in contact mechanics with analytic solutions. Occasionally, thought, simplifications of geometry or neglecting certain effects can make the problems solvable.

The most famous solution which does exist is the Hertz solution for elastic spherical contact. Between 1886 and 1889 Heinrich Hertz published two papers on what would become the field of contact mechanics. In these papers, he solved the problem of how two axisymettric objects behave when in contact with each other. The solution to this problem relied on the

assumptions that the strains of the objects are low, that each body's characteristic radius is much larger than the contact radius, that the surfaces themselves are non-conforming and continuous, and that there is no friction between the objects. If all of these assumptions can be made, the axisymettric contact is said to be 'Hertzian' [26][22][21].

For non-axisymettric solids, with contact friction, changing geometry (due to rotation), possibly large contact areas, and possible plastic deformation, however, no closed form solution exists. Because of this, fitted models are used, which must be parameterized using experimental data.

MSC.ADAMS utilizes such a model, a non-linear spring dashpot: the IMPACT model.

2.3.1 IMPACT Model

The default contact model built into ADAMS is the IMPACT model. This model calculates the penetration depth in a 2 body contact, then applies a spring damper-like force based on the interference (ω), at the center of the contact patch. The equation of the IMPACT model is[13]:

$$F = 0 \; if \; q > q_0 \tag{2.1}$$

$$F = k(\omega)^e - c_{max}\dot{\omega}STEP(q, q_0 - d, 1, q_0, 0) \; if \; q < q_0 \tag{2.2}$$

$$\omega = q_0 - q \tag{2.3}$$

Where q_0 is the location of the target body's surface, and q is the location of the point on the interfering surface which is deepest into the target surface. Where the step function is the ADAMS smooth step function:

$$step(x, x_1, h_1, x_2, h_2) = h_1 + \left(\frac{h_2 - h_1}{x_2 - x_1}\right)(x - x_1) \tag{2.4}$$

$$-(\frac{h_2-h_1}{2\pi})sin\left(\frac{2\pi}{x_2-x_1}(x-x_1)\right)$$

The IMPACT function takes in 5 arguments based on the material properties: Stiffness (k), Force Exponent (e), Maximum Damping (c_{max}), and Penetration Depth (ω). The Stiffness value is a real variable that defines the stiffness of the boundary surface interaction. The Force Exponent defines the deformation characteristic of the interface. Maximum damping is a non negative number that bounds the damping coefficient of the interaction, and Penetration Depth defines the depth at which MSC.ADAMS applies full damping (not to be confused with interference). This smooth step function forces the IMPACT function as a whole to be continuous, which makes it easier to integrate by the solver[19].

In order to establish the behavior of the IMPACT function, a number of analytic solutions were examined and adapted to its form. One of the simplest geometries in the field of contact mechanics to solve is the axisymetric problem of a spherical indenter on an elastic halfspace. A diagram of this problem is shown in Figure 2.2. A number of analytic approaches can be used to model this interaction.

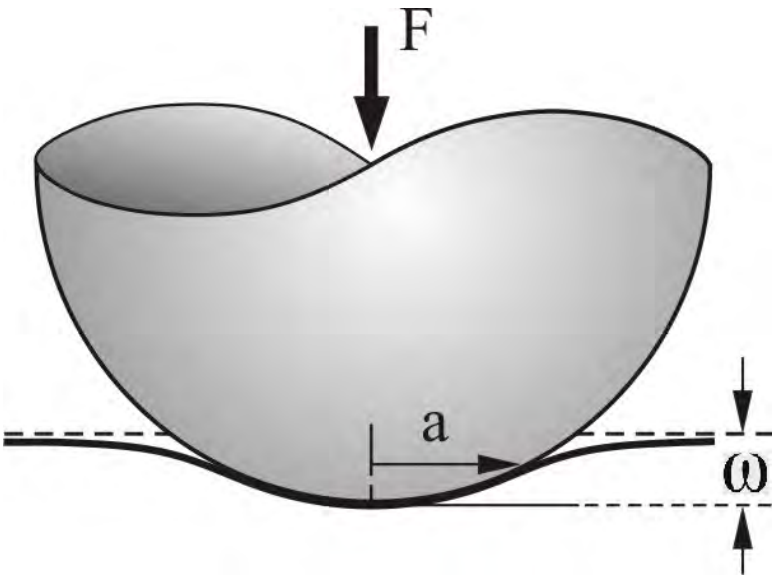

Figure 2.2: Sphere contacting an elastic half-space

As stated earlier, two classes of solution exist for this case, quasistatic (persistent) and dynamic (intermittent) loading. Quasistatic loading describes the case where inertial effects of the bodies are neglected, which infers that time dependent effects do not exist. A quasistatic solution is a series of static equilibrium solutions. Conversely, dynamic loading cases are those for which inertial effects do account for a significant portion of the dynamics, as is the case in bouncing or rattling. Being inherently simpler, most solutions which exist apply to quasistatic loading.

Quasistatic (Persistent) Loading Models

Quasistatic loading, or persistent contact is modeled primarily in 3 ways: elastic, plastic, and elastic-plastic. Of course, MSC.ADAMS assumes the constituent bodies are rigid, so attempting to model fully plastic deformation with a rigid body dynamics package makes little sense. Nevertheless, one modeling method from each type was examined and applied to an ADAMS model of a sphere and half space.

For all quasistatic cases, the time varying terms are assumed to be zero. No damping or penetration depth values are necessary, because in the IMPACT function, those two material properties are multiplied by the time derivative of interference.

Hertz Contact The elastic model examined was the most famous solution in contact mechanics, called the Hertz solution. The Hertz solution to static spherical contact on an elastic halfspace (shown in Figure 2.2) can be fit exactly to the IMPACT model. Hertz states that:

$$F_H = \frac{4}{3} E^* R^{1/2} \omega^{3/2} \tag{2.5}$$

Where F is the applied (and reaction) force for the sphere, R is the radius of the sphere, δ is the interference, and E* is given by:

$$\frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2} \tag{2.6}$$

Where E_1 and ν_1 are the Young's Modulus and Poisson's Ratio of the sphere and E_2 and ν_2 are the Young's Modulus and Poisson's Ratio of the half-space [37][26][22][21]. Immediately it can be seen that the Hertz equation is of a similar form to the IMPACT model. The general IMPACT model with the time dependent terms removed is:

$$F = k(\omega)^e \tag{2.7}$$

Setting this equal to the Hertz solution:

$$F = k(\omega)^e = \frac{4}{3}E^* R^{1/2}\omega^{3/2} \tag{2.8}$$

The force exponent becomes:

$$e = 3/2 \tag{2.9}$$

and the stiffness is given by:

$$k = \frac{4}{3}E^* R^{1/2} \tag{2.10}$$

Using these values in ADAMS (and 0 for damping and penetration depth), the Hertz solution can be simulated exactly and has been verified [19].

Fully Plastic Model While the Hertz solution is fully elastic, another model, based on the concepts of hardness and a geometric truncation model for contact area, is fully plastic. The contact area for a sphere based on geometric truncation is [9]:

$$A = 2\pi R\omega \tag{2.11}$$

Similarly the pressure for fully plastic deformation can be found by recalling that the contact pressure, P, when fully plastic is simply the penetration hardness of the material. Commonly, this is approximated by multiplying the material's yeild strength, S_y, by 2.8.

$$P = H \approx 2.8S_y \tag{2.12}$$

For MSC.ADAMS, the quantity of concern is simply the force as a function of the interference, so the relation is simplified to:

$$F = 2\pi R\omega H \approx 5.6\pi R\omega S_y \tag{2.13}$$

Recall the IMPACT function without dynamic terms:

$$F = k(\omega)^e \tag{2.14}$$

For the fully plastic model the stiffness is therefore:

$$k = 5.6\pi RS_y \tag{2.15}$$

And the force exponent:

$$e = 1 \tag{2.16}$$

And all other terms (damping and penetration depth) are 0. Many criticize this model for producing unrealistically low contact areas and therefore unrealistically high contact

pressures. Nevertheless, this model is very prevalent in the modeling of fully plastic contact mechanics [23][24].

Jackson-Green Model Jackson and Green proposed a elastic-plastic model for spherical contact, for which the point of initial yieding is determined by calculating the critical values of interference (ω_c), area (A_c), or pressure (P_c)[25].

$$\omega_c = \left(\frac{\pi C S_y}{2E'}\right)^2 R \tag{2.17}$$

$$\tag{2.18}$$

where:

$$C = 1.295exp(0.736\nu) \tag{2.19}$$

$$A_c = \pi^3 \left(\frac{C S_y R}{2E'}\right)^2 \tag{2.20}$$

$$P_c = \frac{4}{3}\left(\frac{R}{E'}\right)^2 \left(\frac{C}{2}\pi S_y\right)^3 \tag{2.21}$$

The instantaneous contact force (F), contact area (A), and interference (ω) can then be normalized by their critical values:

$$A^* = A/A_c \tag{2.22}$$

$$F^* = F/F_c \tag{2.23}$$

$$P^* = P/P_c \tag{2.24}$$

$$\omega^* = \omega/\omega_c \tag{2.25}$$

For the approximately elastic regime, the Jackson-Green model matches the Hertz solution[25]:

$$P_e^* = (\omega^*)^{3/2} \tag{2.26}$$

Which holds for $0 \leq \omega^* \leq \omega_t^*$, where $\omega_t^* = 1.9$. For these remaining interferences, $\omega_t^* \leq \omega^*$, the Jackson-Green model states[25]:

$$P_f^* = \left[\exp\left(-\frac{1}{4}(\omega^*)^{5/12}\right)\right](\omega^*)^{3/2} + \frac{4H_g}{CS_y}\left[1 - \exp\left(-\frac{1}{25}(\omega^*)^{5/9}\right)\right]\omega^* \tag{2.27}$$

Substituting in for the normalized interference:

$$P_f^* = \left[\exp\left(-\frac{1}{4}\left(\frac{\omega}{\omega_c}\right)^{5/12}\right)\right]\left(\frac{\omega}{\omega_c}\right)^{3/2} + \frac{4H_g}{CS_y}\left[1 - \exp\left(-\frac{1}{25}\left(\frac{\omega}{\omega_c}\right)^{5/9}\right)\right]\left(\frac{\omega}{\omega_c}\right) \tag{2.28}$$

$$\begin{aligned} P_f^* = &\left[\exp\left(-\frac{1}{4}\left(\frac{\omega}{\left(\frac{\pi CS_y}{2E'}\right)^2 R}\right)^{5/12}\right)\right]\left(\frac{\omega}{\left(\frac{\pi CS_y}{2E'}\right)^2 R}\right)^{3/2} \\ &+ \frac{4H_g}{CS_y}\left[1 - \exp\left(-\frac{1}{25}\left(\frac{\omega}{\left(\frac{\pi CS_y}{2E'}\right)^2 R}\right)^{5/9}\right)\right]\left(\frac{\omega}{\left(\frac{\pi CS_y}{2E'}\right)^2 R}\right) \end{aligned} \tag{2.29}$$

This gives normalized contact pressure as a function of only interference and the material properties for elastic-plastic contact. For the IMPACT model, force is desired, so the relation is multiplied by the contact area. The Jackson-Green model states that for the elastic plastic regime, the contact area is given by[25]:

$$A_F^* = \omega^* \left(\frac{\omega^*}{\omega_t^*} \right)^{0.14 \exp\left(23 S_y / E'\right)} \tag{2.30}$$

$$A_F^* = \left(\frac{\omega}{\omega_c} \right) \left(\frac{\left(\frac{\omega}{\omega_c} \right)}{1.9} \right)^{0.14 \exp\left(23 S_y / E'\right)} \tag{2.31}$$

$$A_F^* = \left(\frac{\omega}{\left(\frac{\pi C S_y}{2E'} \right)^2 R} \right) \left(\frac{\left(\frac{\omega}{\left(\frac{\pi C S_y}{2E'} \right)^2 R} \right)}{1.9} \right)^{0.14 \exp\left(23 S_y / E'\right)} \tag{2.32}$$

Multiplying to get the normalized force as a function of only interference and material properties for the elastic-plastic regime:

$$\begin{aligned} F^* = & \left(\left[\exp\left(-\frac{1}{4} \left(\frac{\omega}{\left(\frac{\pi C S_y}{2E'} \right)^2 R} \right)^{5/12} \right) \right] \left(\frac{\omega}{\left(\frac{\pi C S_y}{2E'} \right)^2 R} \right)^{3/2} \right. \\ & \left. + \frac{4 H_g}{C S_y} \left[1 - \exp\left(-\frac{1}{25} \left(\frac{\omega}{\left(\frac{\pi C S_y}{2E'} \right)^2 R} \right)^{5/9} \right) \right] \left(\frac{\omega}{\left(\frac{\pi C S_y}{2E'} \right)^2 R} \right) \right) \\ & \left(\frac{\omega}{\left(\frac{\pi C S_y}{2E'} \right)^2 R} \right) \left(\frac{\left(\frac{\omega}{\left(\frac{\pi C S_y}{2E'} \right)^2 R} \right)}{1.9} \right)^{0.14 \exp\left(23 S_y / E'\right)} \end{aligned} \tag{2.33}$$

As is immediately apparent, this form is far more complicated than the previously examined models which were easily manipulated into the form of the IMPACT function. As is the case with many contact mechanics models, this manipulation cannot be done for the J-G model. A simple alternative is to simply use a curve fitting tool to approximate the elastic-plastic Jackson-Green model with the IMPACT function. This was done with

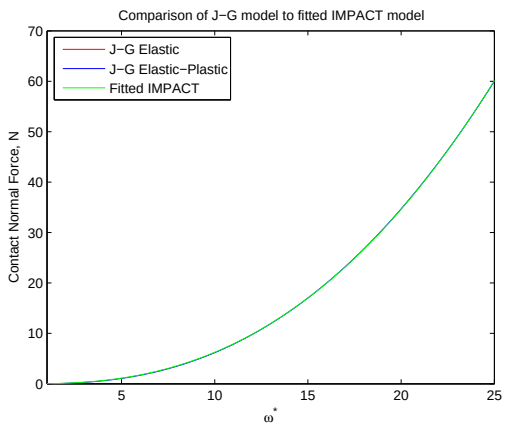

Figure 2.3: Comparison of fitted model to J-G model

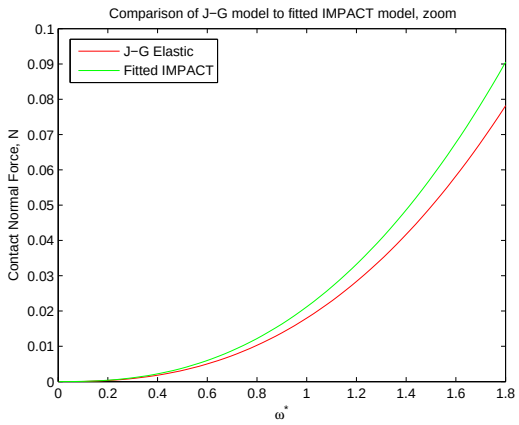

Figure 2.4: Comparison of fitted model to elastic region of J-G model

MATLAB's NLINFIT tool, with a certain material pair yielding the values shown in Table 2.1.

The fit is shown in Figures 2.3 and 2.4, where it can be seen that while it slightly overestimates forces in the relatively small elastic region, it closely matches the model over the larger range.

Kogut-Etsion Model Kogut and Etsion [27], proposed a model for an elastic-perfectly plastic sphere in contact with a rigid halfspace in their 2002 article. Their models were based on FEM data, and separated the problem into 4 piecewise regimes.

Kogut and Etsion define these regimes in terms of the same normalized parameters used in the J-G model:

$$A^* = A/A_c \tag{2.34}$$

$$F^* = F/F_c \tag{2.35}$$

$$P^* = P/P_c \tag{2.36}$$

$$\omega^* = \omega/\omega_c \tag{2.37}$$

The normalized pressure and area are then given in piecewise form:

$$P^* = \begin{cases} \omega^* \leq 1 & (\omega^*)^{3/2} \\ 1 \leq \omega^* \leq 6 & 1.03(\omega^*)^{1.425} \\ 6 \leq \omega^* \leq 110 & 1.40(\omega^*)^{1.263} \\ 110 \leq \omega^* & (2.8S_y)/(P_c) \end{cases} \tag{2.38}$$

$$A^* = \begin{cases} \omega^* \leq 1 & (\pi R\omega)/A_c \\ 1 \leq \omega^* \leq 6 & 0.93(\omega^*)^{1.136} \\ 6 \leq \omega^* \leq 110 & 0.94(\omega^*)^{1.146} \\ 110 \leq \omega^* & (2\pi R\omega)/A_c \end{cases} \tag{2.39}$$

Using the same process as with the J-G model, the equations for contact normal force as a function of only material properties and interference are easily found by substituting in the equations for the critical values and recognizing that:

$$P = P^* P_c \tag{2.40}$$

$$A = A^* A_c \tag{2.41}$$

$$F = PA \tag{2.42}$$

Again, the equation is far too complicated to manipulate into the form of the IMPACT function, so a curve is fit to it to find approximate IMPACT parameters. The fit is shown in Figures 2.5 and 2.6, where it can be seen that just like the fit to the J-G model, while it slightly over-estimates forces in the elastic region, it closely matches the model over the larger range.

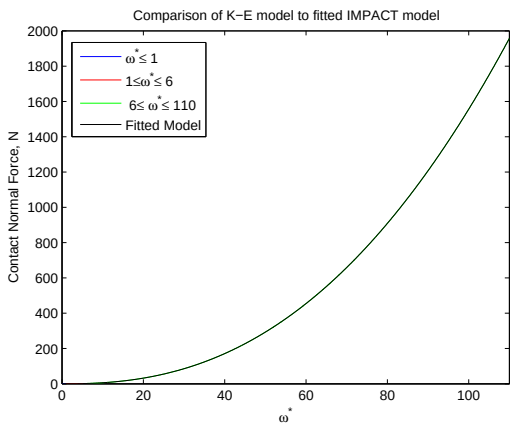

Figure 2.5: Comparison of fitted model to K-E model

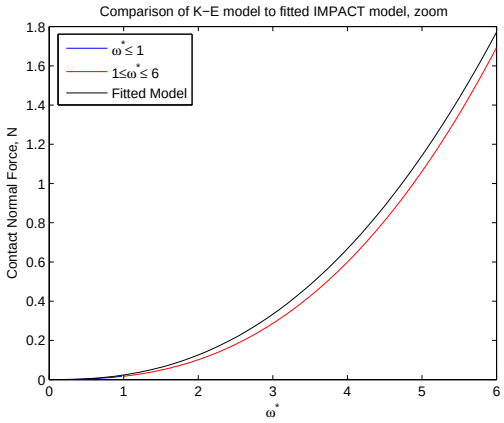

Figure 2.6: Comparison of fitted model to elastic region of K-E model

Summary of Quasistatic Loading Models Take for example the case of 7075-T7 aluminum used for both the half space and the sphere, with the sphere being 5cm in radius. The material properties are approximately [4]:

$$E = 72\ Gpa \tag{2.43}$$

$$\nu = 0.33 \tag{2.44}$$

$$S_y = 435\ MPa \tag{2.45}$$

The effective Young's modulus is given by:

$$\frac{1}{E^*} = \frac{1 - 0.33^2}{72 \times 10^9 Pa} + \frac{1 - 0.33^2}{72 \times 10^9 Pa} \tag{2.46}$$

$$E^* = 40.39\ GPa \tag{2.47}$$

Using these material properties, the ADAMS equivalent contact parameters for each quasistatic loading model were calculated. These values are shown in Table 2.1 (assuming base ADAMS units of Kg-mm-s).

Table 2.1: Quasistatic Loading Contact Parameters

Parameter	Hertz	Plastic	J-G	K-E
Stiffness	12,042,000	382,646	1,643,500	988,250
Force Exponent	1.5	1	2.47	2.4090
Damping	0	0	0	0
Pen. Depth	0	0	0	0

As can be seen from the table, the stiffness values from the fully elastic Hertz model is much higher than the other 3 models which account for plastic deformation. To discern the actual effect of these models on an ADAMS model, the sphere on half-space was modeled. This model is shown in Figure 2.7. Included in the model were both a gravitational force on the sphere, and an additional 1000N force downward.

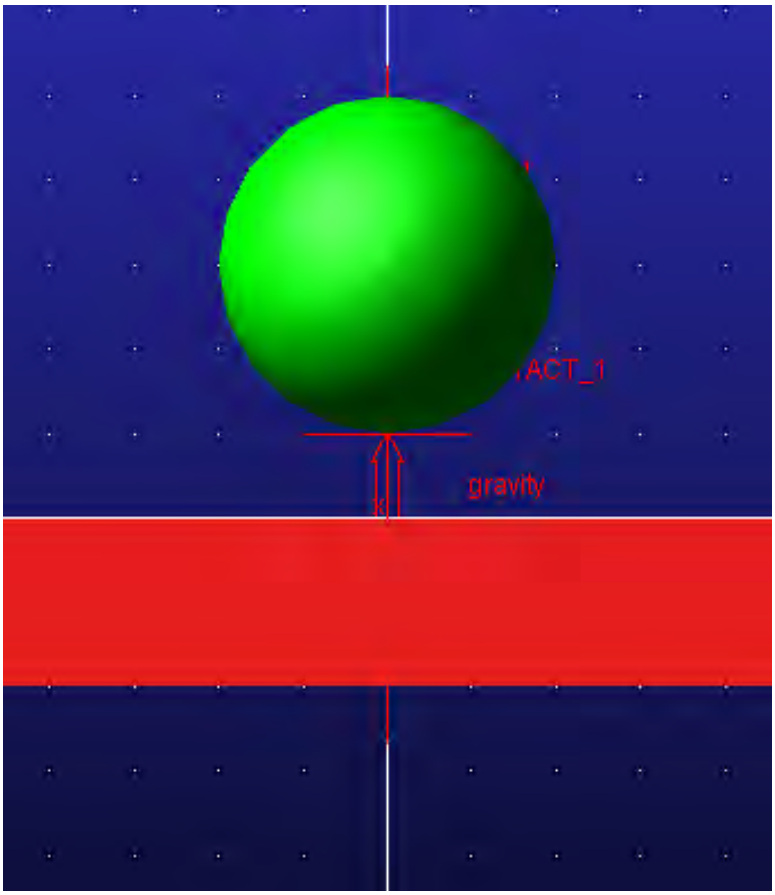

Figure 2.7: ADAMS model of a sphere contacting a half-space

Because the models are all quasistatic, the transient portion of the simulation and only the steady state penetration depth (interference) was examined. The results of the simulations are shown in Table 2.2. In order to simulate the quasistatic solutions, the sphere was set to be just in contact ($\omega = 0$), and a small damping term was added ($c_{max} = 10$), such that the sphere would settle at equilibrium.

Table 2.2: Quasistatic Loading Predicted Penetration Depth

Model	Predicted Pen. Depth
Hertz	0.0020mm
Abb.-Fire.	0.0027mm
J-G	0.0507mm
K-E	0.0581mm

The Hertz and Plastic models, surprisingly, produced similar static penetration depth predictions despite being based on totally different assumptions (fully elastic vs. fully plastic). This is possibly due to their relatively similar force exponents, which overcomes the massive differences in their contact stiffnesses. The two elastic-plastic models produced similar penetration depths, both of which were much larger than the perfectly elastic Hertz model's prediction. Considering the incredibly low critical interferences of spheres, these elastic-plastic models are likely a better indication of reality than the Hertz model in this case.

As a final comparison of quasistatic contact models, small values for max damping and penetration depth were added to each model (10 and 0.01 respectively). With the same MSC.ADAMS model as before, a bouncing sphere was simulated, and the dynamic motion predicted by each model was compared. The comparison of the 4 models examined is shown in Figure 2.8. Before contact occurs, the 4 simulations are all identical (because the ball is in freefall), so the K-E model's plot perfectly obscures the other 3.

As can be seen in the Figure, all of the models produce similar results with the exception of the Hertz model, which predicted a much higher bounce than the others.

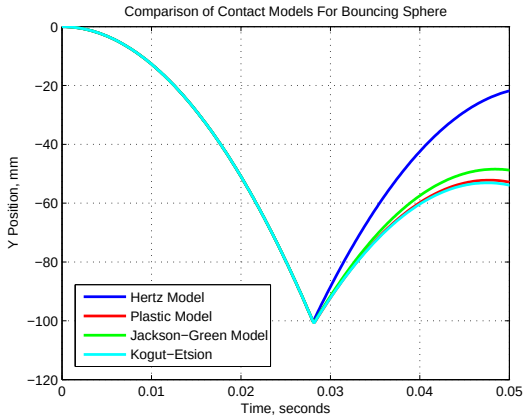

Figure 2.8: Comparison of contact models' impact on dynamics

Dynamic Loading Models

In dynamic loading, or impact, the contact problem is complicated by the inelastic loading of the materials. Permanent deformation removes energy from the collision, effecting the resultant forces and the contact area. This transfer of energy is commonly quantified by the coefficient of restitution:

$$C_R = \frac{relative\ speed\ after\ collision}{relative\ speed\ before\ collision} \tag{2.48}$$

The coefficient of restitution can range between 0.0 and 1.0, where 1.0 is a perfectly elastic collision, and 0.0 is perfectly plastic (no bounce). To understand the significance of this effect on the dynamics of a system, consider again the generic impact of a sphere on a half space. The collision can be separated into two phases, the compression phase, where the penetration velocity of the sphere it positive (going into the half space), and the restitution phase, where the penetration velocity is negative (rebounding off of the half space).

In a perfectly elastic collision, these two phases would be identical other than the sign. Figure 2.9 shows this perfectly elastic case (O-A-C), as well as a real world case where permanent deformation occurs (O-A-B), and a partially elastic case where no permanent deformation occurs, but energy is lost (O-A-D)[20].

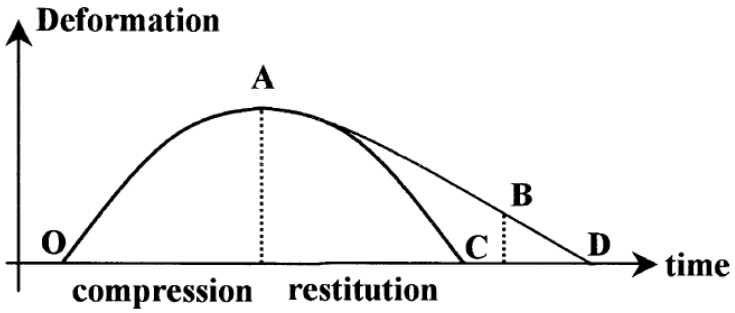

Figure 2.9: Deformation during an elastic and real-world collision[20]

In order to model this phenomena, contact models are developed which are a function of not only penetration depth, ω, but also its time derivative, $\dot{\omega}$. The most common model is known as the spring-dashpot model.

Spring-Dashpot Model The spring dashpot model considers the impact to act as a non-linear spring-damper, the coefficients of which are defined by geometry and material properties. The IMPACT model in ADAMS is, in fact, a slightly modified non-linear spring-damper model. The damping rate however, is multiplied by the ADAMS smooth step function so that it's coefficient increases gradually from 0 to the chosen damping value between the first instance of contact and the chosen penetration depth. This continuity is important for the solver and formulator to function correctly[15].

In practice, these terms are found experimentally because of the strong dependence on both material properties and geometry [30][29].

Practical Use of the IMPACT Function

While many contact models exist for idealized cases with constant and simple geometry, and certain conditions, none exists for arbitrary (and changing) geometry. Because of this, the only way to find the true ADAMS IMPACT function parameters in Equations 2.1 through 2.3 for an arbitrary system is by using rules of thumb and experimental data. Figures 2.10

and 2.11 show the effect of changing the stiffness and damping values respectively in the IMPACT function. The simulation shown is of a ball bouncing on a half space.

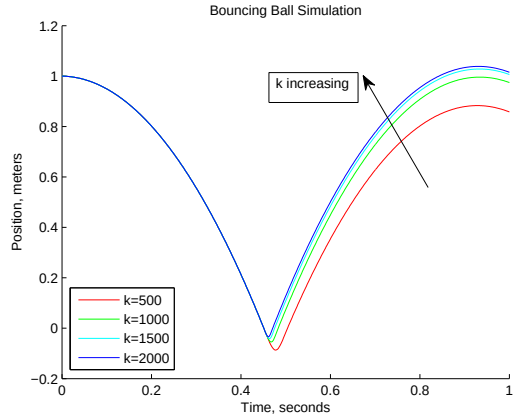

Figure 2.10: Effect of changing stiffness

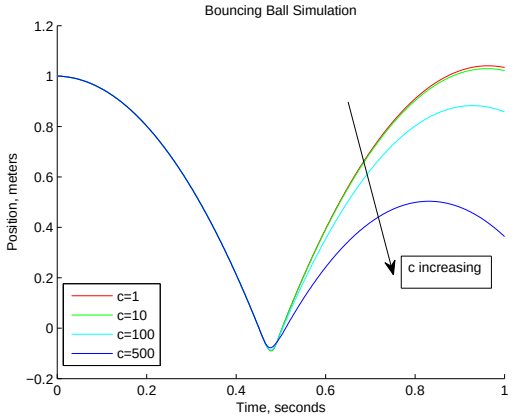

Figure 2.11: Effect of changing damping

By comparing experimental results to a simulation, the parameters can be tuned, as will be demonstrated in the 3rd case study, presented in Chapter 6.

2.3.2 Friction

The force which resists the relative motion of solid surfaces is known as friction. In the context of contact mechanics, the definition can be narrowed to that of dry friction, which resists lateral motion between two solids. Frictional forces are broken down into two categories, kinetic friction, which exists between two surfaces which are moving, and static friction, which exists between stationary surfaces.

The kinetic and static friction regimes are commonly combined into one equation for the magnitude of the friction force:

$$F_f = \mu N \tag{2.49}$$

Where the friction coefficient, μ, is a function of the slip velocity at the contact:

$$\mu = f(v_{slip}) \tag{2.50}$$

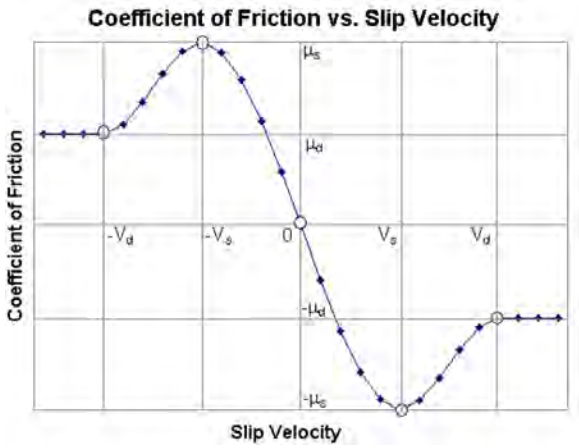

Figure 2.12: ADAMS coulomb friction model

In ADAMS, a model referred to as the Coulomb friction model is used for μ. The Coulomb friction model is created such that there are no discontinuities in the function for μ. An example of this function is shown plotted in Figure 2.12. There are 4 parameters which define the model: μ_s, μ_d, V_s, and V_d. These 4 parameters define the 4 inflection points which the ADAMS model connects with smoothed step functions [19].

Including those from previous sections, there are therefore 8 different parameters needed to characterize the contact mechanics for a pair of bodies interacting. From the IMPACT function: stiffness, force exponent, max damping, and penetration depth are required. From the Coulomb friction model, static friction coefficient, kinetic friction coefficient, static transition velocity, and kinetic transition velocity are required.

In order to determine these parameters for a material pair, experimental data is required. With this data, the parameters can be iteratively found until the simulation replicates the experiment.

2.4 Solver

Multibody simulation software packages formulate and solve a problem referred to as a differential algebraic equation. This equation represents multiple differential equations which fully define the system of interest. First the user builds the system in ADAMS/View

using the user interface, then a formulation tool within the ADAMS/Solver formulates the representative differential algebraic equation, which is then solved with an integrator. There are 3 different formulators build into ADAMS with various pros and cons to each:

I3 This is the default formulation method for ADAMS. Commonly referred to as Index 3 formulation, this formulation is less robust than the other two methods, due to conditional accuracy in velocity and acceleration, but it runs much faster than the other two. It can also be limited in accuracy at high frequencies [43].

SI1 SI1 differs from Index-3 formulation in nearly every metric. It has high accuracy on all terms (position, velocity and acceleration), is more robust than Index-3, and can handle the highest frequencies of all 3 formulation methods. It is, though, much slower than Index-3, and has lower tolerances[43].

SI2 SI2 has better tolerances than SI1, and similar qualities in accuracy, robustness, and speed, but doesn't handle as high frequencies as SI1 does. SI2 does still solve higher frequencies than Index-3 formulation, however [43].

Likewise, ADAMS has a number of possible integrators for the user to choose from. All of them take the differential algebraic equations and initial conditions and integrate the state forward through time. The full list of possible integrators for ADAMS is:

GSTIFF This is the default integrator used in ADAMS. GSTIFF is a variable order, variable step size integrator. It is commonly used in multibody simulation and dynamics applications for it's robustness, speed, and accuracy. It was designed specifically for the integration of numerically stiff systems of differential equations. A system of differential equations is considered stiff when the eigenvalues of the system are very far from each other, and the higher frequency eigenvalues are over-damped. Practically, the consequence of stiff differential equations is very slow and inaccurate solutions from non-stiff integrators, which appears as large spikes in predicted acceleration[44].

WSTIFF WSTIFF differs only very slightly from GSTIFF. Both are variable order, variable step size backward difference formulations. The difference is that the coefficients which govern the integration are step size invariant in GSTIFF, while in WSTIFF they are a function of step size. This allows WSTIFF to handle sudden changes in step size (due to events such as contact) with smaller errors than GSTIFF, at the cost of computation speed.

CONSTANT BDF CONSTANT BDF, like GSTIFF, is a stiff integrator. Unlike GSTIFF, however, it is fixed step, meaning the user must specify the time step before running it. It does have a corrector built in to ensure that the equations of motion are satisfied[43].

ABAM The Adams-Bashforth and Adams-Moulton methods are explicit and implicit respectively. They are combined into a predictor-corrector pair which can be used to solve ordinary differential equations[32].

RKF45 The Runge-Kutta-Fehlberg method is very commonly used for the solving of ordinary differential equations. It is an extension of the seminal Runge-Kutta family of integrators, which uses an embedded high-order method to estimate and control error by adapting the step-size automatically[16].

HHT The Hilber-Hughes-Taylor method is used commonly in structural dynamics for solving linear ordinary differential equations. It is considerably faster than the default GSTIFF integrator in many cases, with very similar results in position and velocity. Large spikes can occur in the acceleration results, however [38].

Newmark The Newmark-beta method was developed in 1959 by University of Illinois professor, Nathan M. Newmark. It was developed originally for use in the area of structural dynamics[39].

HASTIFF The Hiller-Anantharaman Stiff integrator is similar to GSTIFF and WSTIFF, but is intended to have improved stability at smaller time steps. It can only be used with SI1 and SI2 formulations[34].

The models developed in this thesis are all stiff, so the decision needed to be made was between GSTIFF, CONSTANT BDF, WSTIFF and HASTIFF for integrators, and between the various formulation methods. In order to do this, a model (the missile launcher model developed in Chapter 6, with the profile shown in Figure 6.35), was simulated with all 4 to compare the results. Without a known 'true acceleration', it is difficult to determine which of the slightly different methods is most accurate. It is known that ill-suited integrator-formulators tend to generate artificial spikes in acceleration. Because of this, and because the outputs of all integrators tested are nearly identical other than the size of these spikes, VDV is used to evaluate them.

Vibration Dose Value (VDV) is a quantity used in whole body vibration analysis to asses the impact of a vibration profile on human beings. It is calculated by:

$$VDV = \left(\frac{1}{T} \int_0^T a^4 dt \right)^{1/4} \tag{2.51}$$

It can be seen that this is nearly identical to a root-mean-square operation, but uses 4th powers rather than 2nd. This change increases the influence of large spikes on the final value, which is why it was chosen for evaluating the integrators[47]. The raw time-series data for the various integrator-formulators is shown in Figure 2.13, and a zoomed view of the same data is shown in Figure 2.14. In these figures, it can be clearly seen that all integrator-formulators tested capture the same phenomena, and the only significant differences are in the height of the sharp peaks.

Table 2.3 shows the calculated VDV for each integrator-formulator which would successfully run for the model referenced above, normalized by the VDV of GSTIFF-I3 (the default

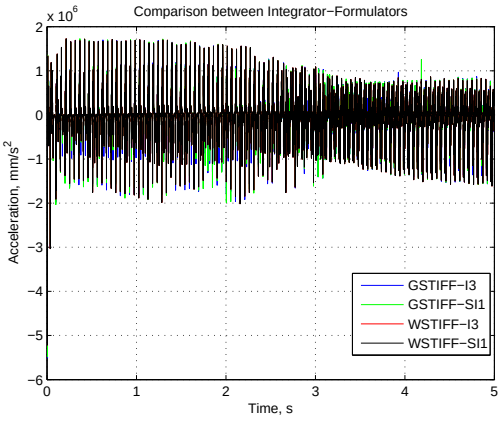

Figure 2.13: Comparison of various integrator-formulators

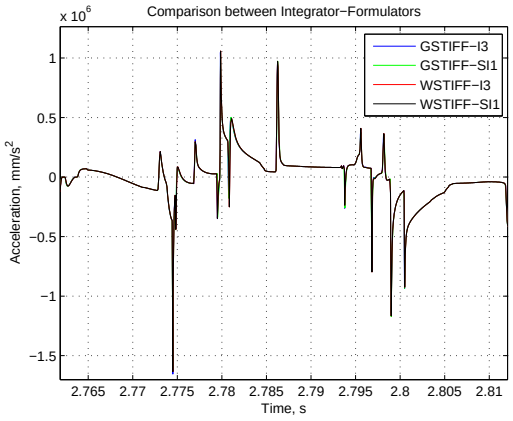

Figure 2.14: Zoomed comparison of integrator-formulators

integrator-formulator). In this case the lower relative VDV implies a better integrator-formulator combination. Because WSTIFF-I3 and WSTIFF-SI1 have identical relative VDV values, the faster running of the two (WSTIFF-I3) was selected as the integrator-formulator to use in this thesis.

Table 2.3: VDV/VDV-Benchmark

Integrator-Formulator	VDV/VDV-Benchmark
GSTIFF-I3	1.0000
GSTIFF-SI1	1.0161
WSTIFF-I3	0.9891
WSTIFF-SI1	0.9891

2.5 Subroutines in ADAMS: Contact Area

While most common functions and features are built into ADAMS already, occasionally a user-written subroutine is required to perform a specific task. One such example of this is for storing the contact area. By default, the contact area is not stored in the information about the contact during simulation (e.g. normal force, location, slip velocity, ect.). Of course, in contact mechanics and wear simulation, the contact area is a critical value for the calculation of contact pressure and other quantities.

This outline will detail the process for compiling and using a subroutine to calculate and store contact area in ADAMS, specifically in FORTRAN. This is also possible in C, which will not be discussed here. There are a few software requirements accomplish this task:

- Intel FORTRAN compiler
- ADAMS/Car

To actually compile the subroutine, the Intel FORTRAN compiler is required, Gfortran is not supported. ADAMS/Car is also required for certain packages related to linking the subroutine to the ADAMS solver.

The subroutine itself is written in FORTRAN, and is meant to replace the functions CNFSUB and CFFSUB, which calculate the normal force and frictional force of the contact respectively for the solver. The difference between the user-written version and the default version is the addition of a common block in CFFSUB which holds the contact area value. This variable is passed to the ADAMS solver via VARSUB, which can be stored in a state variable.

The contact area is calculated by assuming the penetration depth is very small, which means the area can be approximated by dividing the intersecting volume by the penetration depth.

Using the custom FORTRAN code, the next step is to compile and link it to the solver library. The desired outcome is a Windows DLL, which the ADAMS solver can access and use in a model. To do this, first run the Intel FORTRAN compiler, and change the directory to wherever the FORTRAN code is stored. The pathnames will change depending on the specific machine and version of ADAMS being used, but the command shown in Figure 2.15 will compile and link the script.

If successful, the compilation will yield a contactsub.dll file. With this created, the next step is to actually integrate this DLL into an ADAMS model. For the sake of example, take the model shown in Figure 2.16, which simulates a sphere falling onto a block. The goal of this

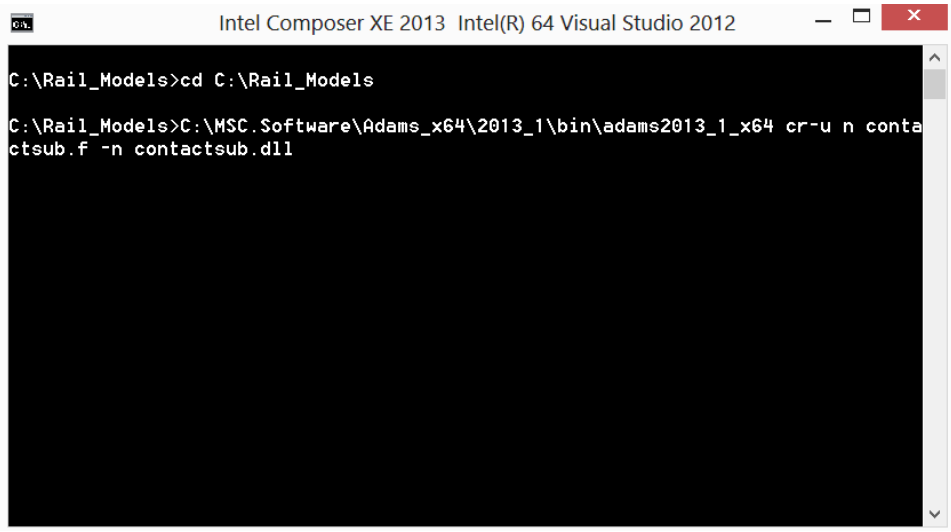

Figure 2.15: Compiling and linking a subroutine in FORTRAN for ADAMS

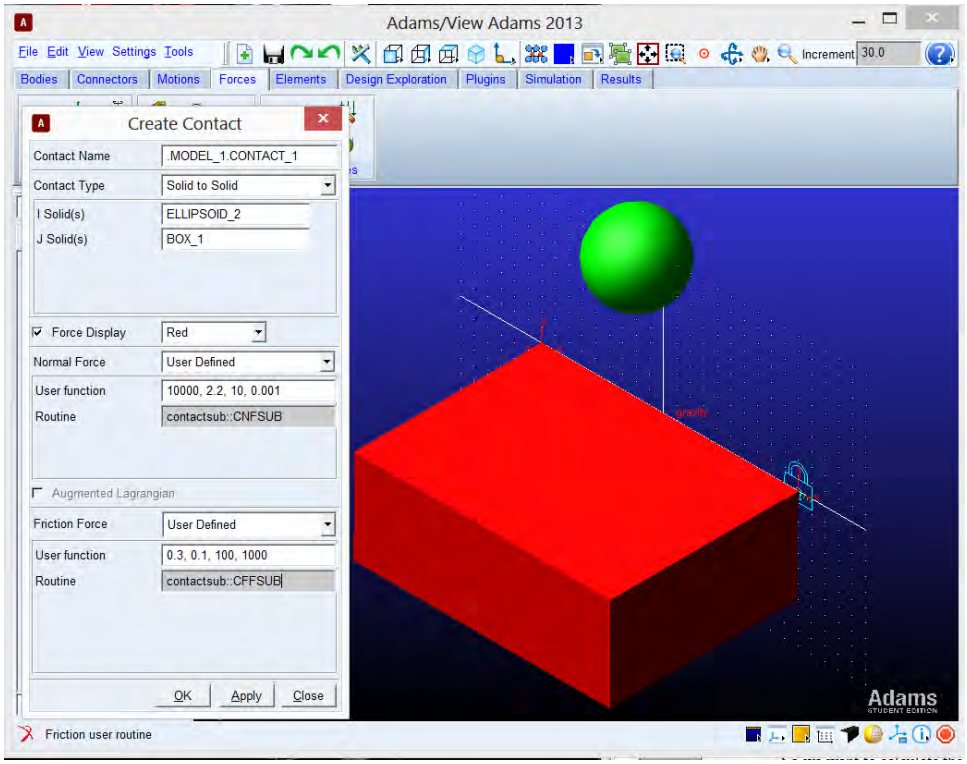

Figure 2.16: Creating contacts in ADAMS using compiled subroutine

model is to simulate the area of contact of this ball dropping onto the box, which is grounded. First create the contact, using the same prompt as for a normal contact. Instead of selecting IMPACT, COULOMB or POISSON, select "user defined" for both the contact and friction

models. In the 'User Function' box, enter the 4 parameters for IMPACT and COULOMB respectively, with each number separated by columns. For the 'Routine' box under normal force, enter: "contactsub::CNFSUB". For the 'Routine' box under friction force, enter: "contactsub::CFFSUB". This tells the ADAMS solver to use the modified CFFSUB and CNFSUB instead of the default versions. An example of this process is shown2.16.

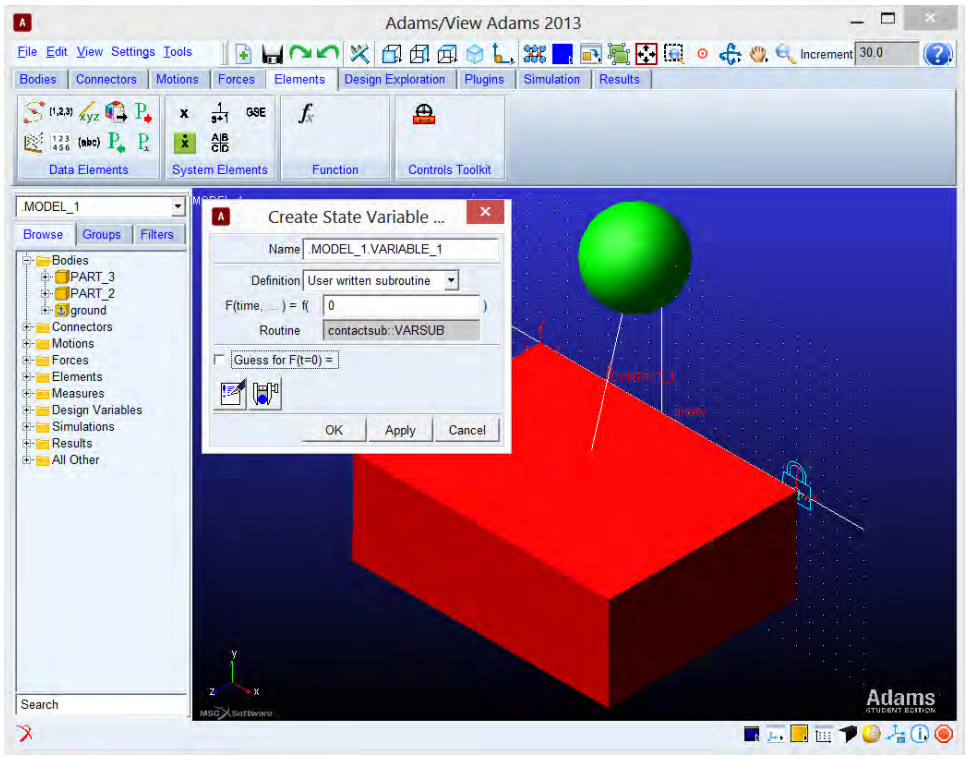

Figure 2.17: Creating a state variable to track the contact area

The next step is to create a state variable to track the contact area. To do this, select state variable out of the elements tab, and select user written subroutine. The routine to be tracked by the subroutine is "contactsub::VARSUB", which as can be seen in the FORTRAN code, simply stores the contact area in a common block. An example of this dialog is shown in Figure 2.17.

Next a REQUEST must be created to actually save these values for post-processing. Select the option to define the request using function expressions. Track the variable by entering the following into the F1 dialog box:

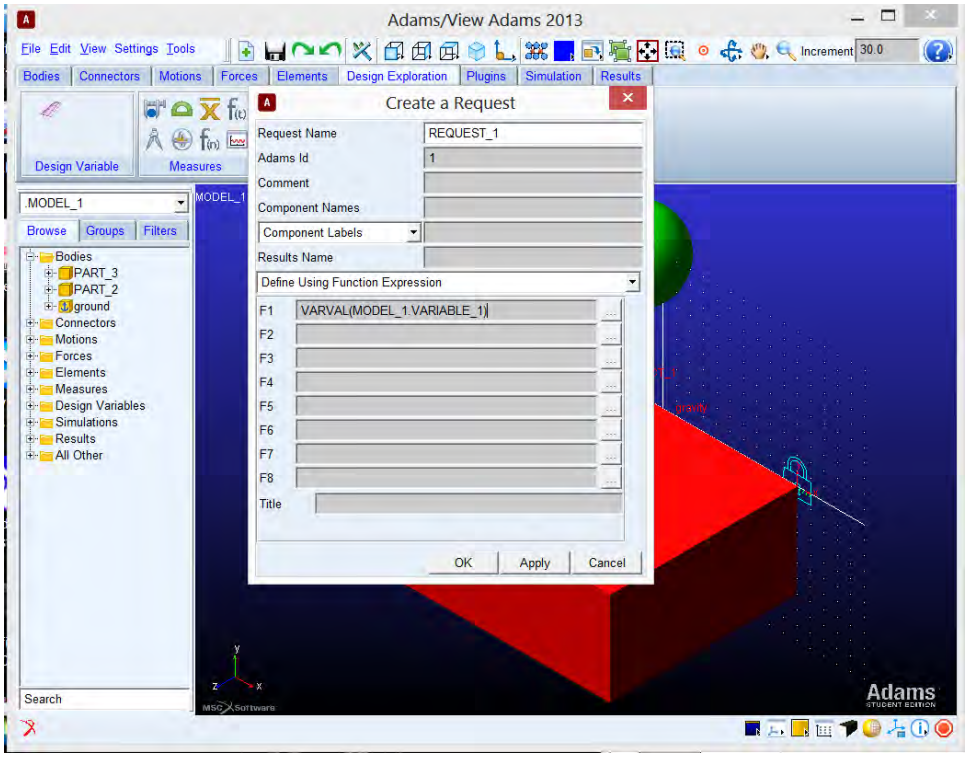

Figure 2.18: Creating a request to store the state variable

```
"VARVAL(MODEL_1.VARIABLE_1)"
```

An example of this dialog is shown in Figure 2.18.

Finally, the DLL itself must be associated with the solver. To do that, goto Settings on the top menu bar, the solver submenu, and select executable. Here the solver library must be set to the location of the DLL, as shown in Figure 2.19.

With this done, the ADAMS model can be run as normal, and in the Post-Processor, the contact area will be stored under:

```
REQUEST_1\U1
```

It can be plotted as with any other quantity in the post-processor, but will by default have no units associated with it. The units will be whatever base length unit has been selected for the model, squared. The plotted contact area for the falling ball model is shown in Figure 2.20.

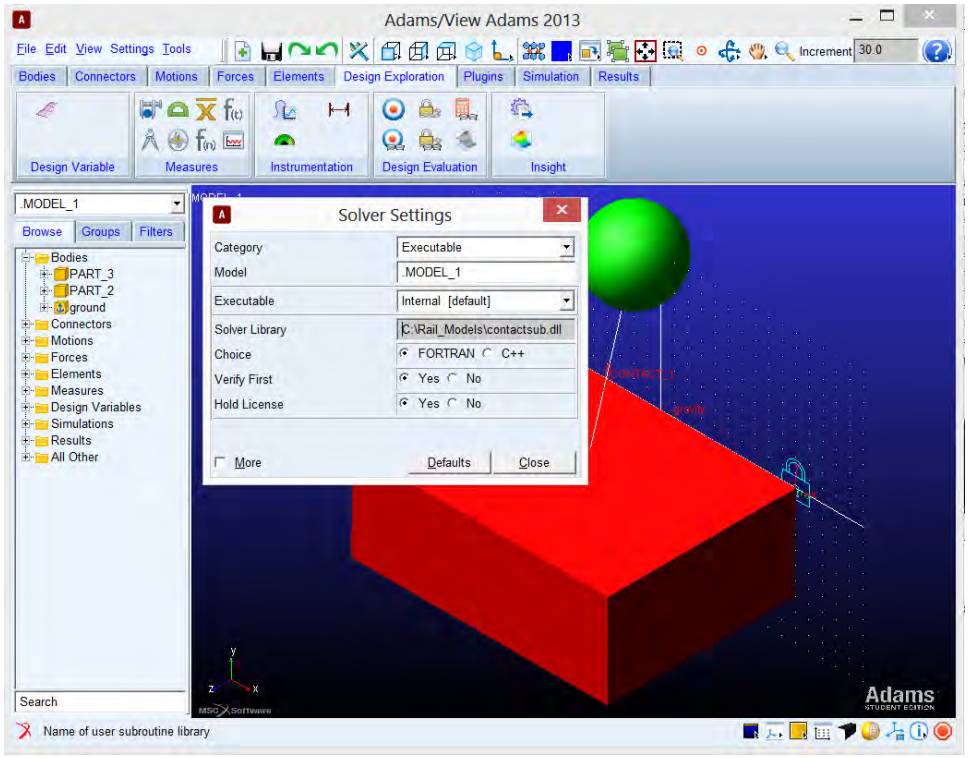

Figure 2.19: Linking the modified DLL to the ADAMS solver

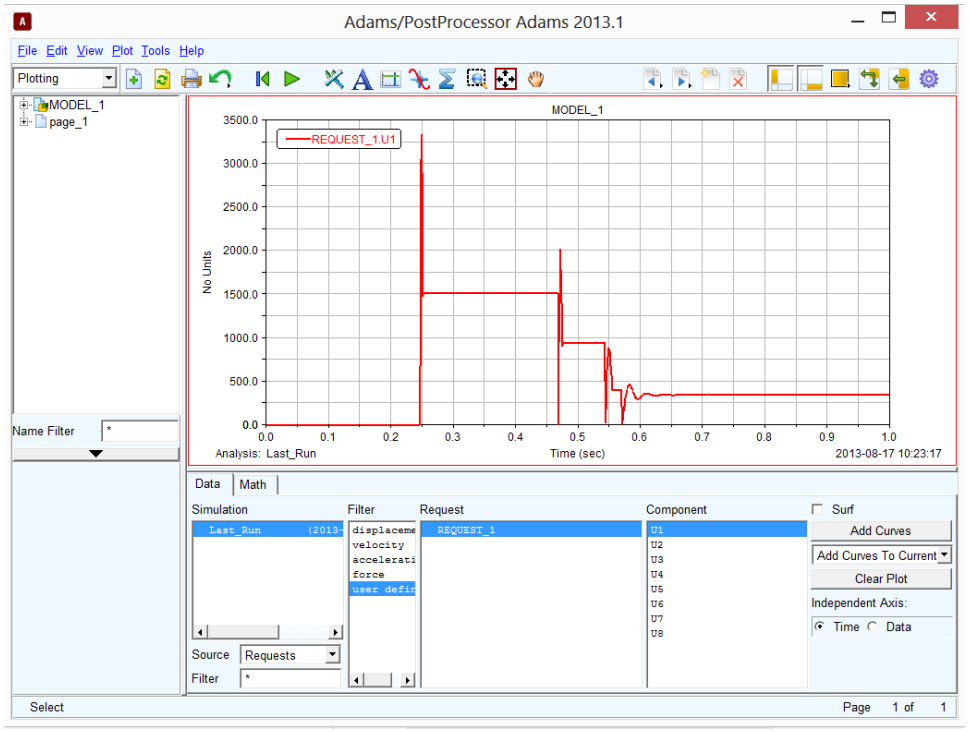

Figure 2.20: Plotting the contact area in the ADAMS Post-Processor

Chapter 3

Wear Modeling

Wear is the removal of material from a surface, caused by interaction with another solid. It is characterized by the physical interaction between asperities, or surface roughness. Wear can be separated into a number of sub-categories, separated by the means in which the asperities interact.

3.1 Types of Wear

There are 5 primary modes of wear in mechanical systems: abrasion, adhesion, impact, fretting and erosion. They differ from each other not only by the processes which governs them, but also by the changes to the system which can mitigate them. This is important because, for example, while hardening the materials may lessen the impact of one mode of wear, it can actually increase that of another.

Some forms of wear are more common in general applications than others, while other types are less common but very destructive. While often debated, it is generally considered that the most important types of wear are adhesion and abrasion. Adhesion tends to be the most common form of wear, but also has relatively low wear rates. The somewhat less common abrasive wear, however, has high wear rates, and can be very destructive [42].

3.1.1 Abrasive Wear

Abrasive wear is the removal of material from a relatively soft surface by another, harder, surface. A number of mechanisms can cause abrasion, including ploughing and cutting. They all differ slightly, but can be identified by inspection. Ploughing is when an asperity sliding along a surface creates a deep groove, displacing material to the sides, while

producing relatively little wear debris. Ploughing is shown in Figure 3.1 and as 'Plastic Flow' in Figure 3.2. Cutting resembles traditional machining, albeit on a smaller scale. Small chips or fragments are created by cutting, creating a large amount of wear debris. This process is also shown in Figure 3.2[46].

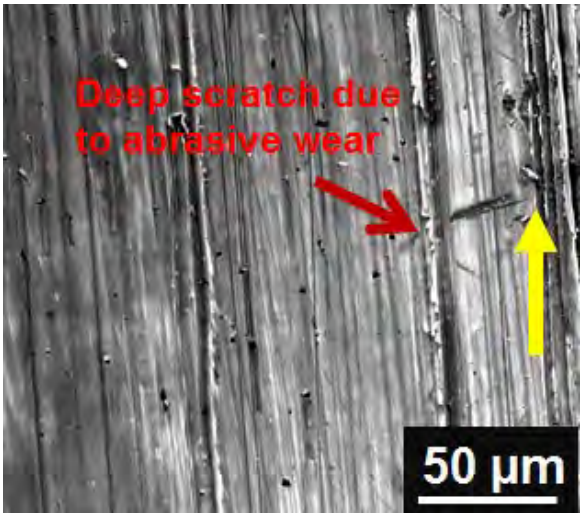

Figure 3.1: Abrasive Wear: Ploughing shown through scanning electron microscope

The primary quantity of interest in this thesis is the wear volume and mass at each contact interface. There are a number of models for wear, which vary depending on material, conditions and other parameters. One of the most common models is Archard's, which states:

$$W = \frac{k_{abr} N s}{H} \tag{3.1}$$

Where W is the total worn volume, s is the sliding distance, N is the applied load, and H is the penetration hardness. The non-dimensional scaling factor, k_{abr}, is related to the likelihood of two asperities creating a wear particle in their collision. This value is found experimentally, and is colloquially known as the non-dimensional abrasive wear coefficient [31][11][42][10].

Assuming known and constant material properties (hardness and wear coefficient), the wear volume is seen to be a function of only the normal load and sliding distance at the

contact interface. These values must be derived from a dynamic simulation. Most multi-body dynamics codes will report the contact's normal force as well as the relative velocity between the two bodies, from which sliding distance can be found.

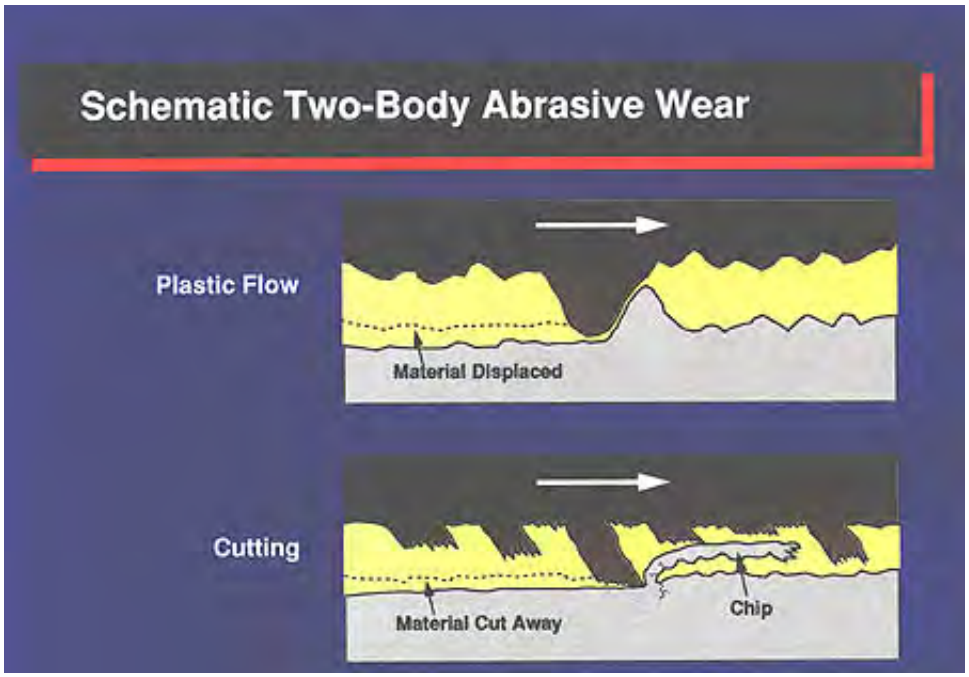

Figure 3.2: Schematic of abrasive wear[10]

3.1.2 Adhesive Wear

Adhesive wear exists in contacts between chemically compatible materials, where the contact of asperities causes transfer of material from one surface to another. This transfer is generally due to the direct contact between asperities during relative motion. The impact of these surface features causes plastic deformation, which creates wear particles. If the two surfaces and the wear particles themselves are chemically attractive, the wear particles will bond to surfaces again, creating material transfer. An example of this material transfer is shown in Figure 3.3, and a schematic of this process taking place is shown in Figure 3.4.

The accepted model for adhesive wear is very similar in form to that of abrasive wear. Holm and Archard proposed that adhesive wear can be modeled by the equation[42]:

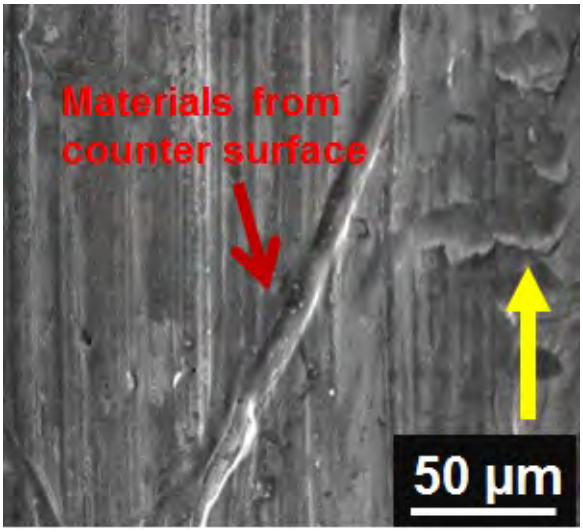

Figure 3.3: Adhesive Wear: Material transfer shown through scanning electron microscope

$$W = \frac{k_{adh} N s}{H} \tag{3.2}$$

Note that the equation is of identical form to that of abrasive wear, all that differs is the value of the wear coefficient, which will differ between adhesion and abrasion models. Many argue that this is a simple coincidence, while others contend that it is a consequence of a fundamental relationship between the two modes[45][46][10].

3.1.3 Impact

Impact is characterized by high boring velocities in a contact. This is the component of the relative velocity between two bodies which is normal to the surfaces themselves. Impact can cause pitting or other surface defects. A diagram of impact wear is shown in Figure 3.5.

Impact was first postulated by DeGee to be a form of adhesive wear [18]. In line with that, a model similar to Archard's adhesive wear equation can be used to model it. Rabinowicz proposed[42]:

$$W = \frac{k_{imp} N s}{H} \tag{3.3}$$

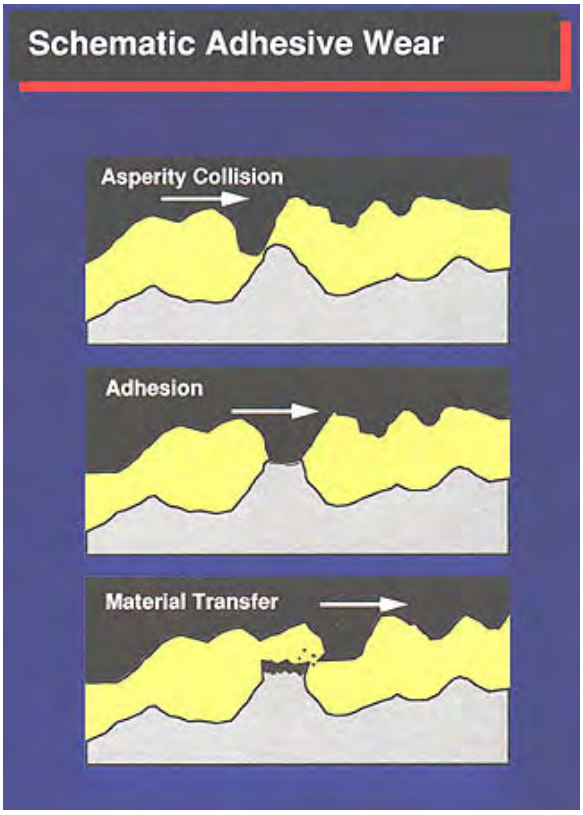

Figure 3.4: Schematic of adhesive wear[10]

Where k_{imp} is the non-dimensional wear coefficient of impact, N is the normal load, s is the sliding distance, and H is the penetration hardness. He expands that the normal load can be re-written as:

$$N = \frac{F_{fric}}{\mu} \tag{3.4}$$

Where F_{fric} is the frictional force, and μ is the coefficient of friction. Substituted in to the wear equation:

$$W = \frac{k_{imp} F_{fric} s}{\mu H} \tag{3.5}$$

The quantity $F_{fric}s$ is equivalent to the energy dissipated by sliding at the contact itself, so a value, α is defined as the proportion of total impact energy which is dissipated by slip. Putting all of this together, the wear can be re-defined as:

$$W = \frac{KIn}{H} \tag{3.6}$$

Where K is:

$$K = \frac{\alpha k_{imp}}{\mu} \tag{3.7}$$

I is the average total energy expended in an individual impact, and n is the number of impacts. Rabincowicz designed an experiment based on this equation in 1952, where he concluded that the model is not only effective, but also confirms DeGee's conclusion that impact wear is related to adhesion [42][18][10].

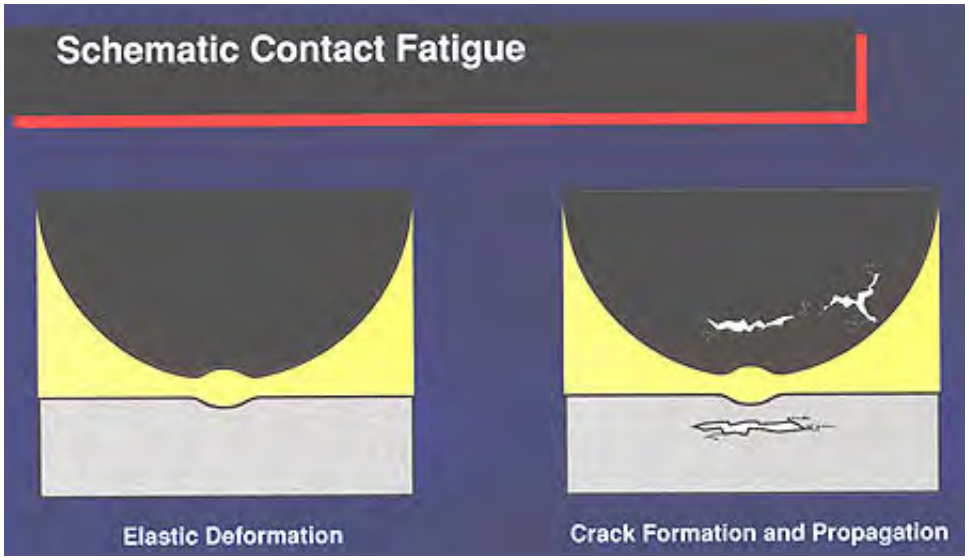

Figure 3.5: Schematic of impact wear[10]

3.1.4 Fretting Wear

Fretting wear is characterized by repetitive periodic relative motion of low amplitude. Because the relative motion is both short and repeated, any wear particles produced by the motion become entrapped between the surfaces, as opposed to abrasion, where the wear particles are often left behind, lost, and therefore removed from the wear process. This

accumulation of wear particles (which are often very hard due to work hardening) accelerates wear[46].

As a rule of thumb, abrasion becomes fretting when the amplitude of the relative motion is less than the characteristic diameter of the contact area.

3.1.5 Erosive Wear

Erosive wear is defined by the wear caused by impacting particles of solid or liquid on a surface. A simple example of erosion is sand-blasting. In these cases the sliding distance and the time in contact are both very small, but over time and with enough particles, significant wear can occur[46].

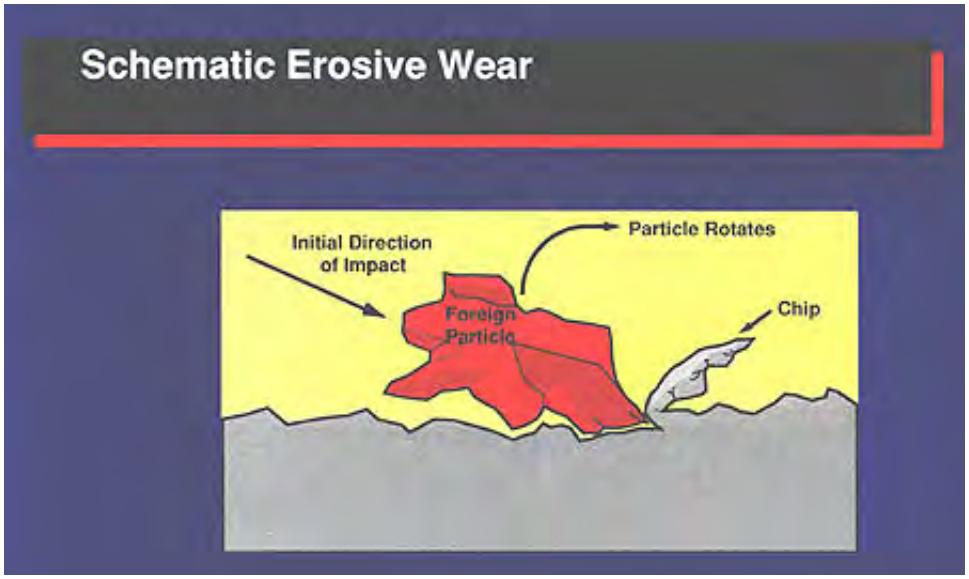

Figure 3.6: Schematic of erosive wear[10]

3.1.6 Engineering Wear Model

It can be seen that the equations which model most of the modes of wear are of identical forms, and differ only by their respective coefficients (k_{abr}, k_{adh}, and k_{imp}). This means that the models themselves depend very strongly on the experiments from which the specific

wear coefficients are derived. Experiments which closely resemble the true environment of the contact being modeled must be used to build accurate wear models.

While the wear coefficients all vary differently with material properties, environmental properties and other variables, they can be lumped together if the materials and environment used in the experiment closely match those of the application. In this case the lumped wear model is given by:

$$W = KNs \tag{3.8}$$

Where K is the dimensional lumped wear coefficient. While the Archard equations imply that the wear coefficients depend only on material properties, in real engineered systems this is rarely the case. The dimensional lumped wear coefficient is a function of at least the contact pressure at that instant, the materials penetration hardness and possibly other environmental terms. An experiment was required to properly quantify this function in terms of the contact pressure, σ.

$$K = f(\sigma, H, ...) \tag{3.9}$$

The exact model for K was unknown, but certain characteristics were assumed. In the context of multibody simulation, the wear is approximately 0 when the contact pressure is 0, because all of the bodies are modeled as smooth, and the contact force is a function of only the interference between the two bodies. While the two bodies can in some cases be in contact with no contact pressure, but wearing due to adhesion, this mode is neglected. Nevertheless, the value of the wear coefficient at 0 is unknown, because any real number multiplied by the 0 of contact pressure fulfills the boundary condition. Similarly, if there is an infinite contact pressure and a non-zero sliding distance, the wear is expected to be non-zero and positive. Because of these conditions, an exponential model was chosen.

$$K(\sigma) = C_1\sigma^{(1/C_2)} + C_3 \tag{3.10}$$

The model is further modified by recognizing that when the normal force is infinite, the wear is not also infinite. This is because the loading body will simply punch a hole in the surface, which will result in a finite mass loss from the surface. Because of this, any datapoints for which the surface pressure is above twice the yield strength (implying bulk plastic deformation) are assumed to have zero wear. The wear in this case is, of course, not zero, but with no good model of the wear in these cases for arbitrary geometries, some assumption must be made. If a simulation produces a significant number of these high pressure datapoints, further investigation will be required to appropriately model the wear. The full wear model is therefore:

$$K(\sigma) = C_1\sigma^{(1/C_2)} + C_3 \quad for: \ \sigma <= 2S_y \tag{3.11}$$

In order to predict wear volume over time, it is convenient to work with the time derivative of wear volume. First take a differentially small sliding distance and the wear volume associated with it:

$$dW = KNds \tag{3.12}$$

Assuming that K varies with σ according to Equation 3.11, and that there are no data points above $2.8S_y$:

$$\frac{dW}{dt} = K(\sigma)Nv_{slip} \tag{3.13}$$

$$\frac{dW}{dt} = C_1v_{slip}N\sigma^{(1/C_2)} + C_3Nv_{slip} \tag{3.14}$$

Where v_{slip} is the time derivative of sliding distance. With this model, the predicted wear for a period of time t_{end} long is given by:

$$W(t_{end}) = \int_0^{t_{end}} \left(C_1 v_{slip} N \sigma^{(1/C_2)} + C_3 N v_{slip} \right) dt \tag{3.15}$$

With this model the wear volume for any time period can be estimated.

Chapter 4

Case Study: Persistent Contact

4.1 Introduction

Wear coefficients are commonly found in practice by using machines such as the one shown in Figure 4.1, a CETR/Bruker UMT-3 friction and wear tester, which applies certain loads and relative motions to a material to find experimental wear values. With this measured wear, load and relative motion data, experimental wear coefficients for various loads and speeds can be calculated for a material pair. In order to find the wear coefficient for the Hellfire missile launcher system, a setup such as this was used. Likewise, this data was used to find friction model parameters for the material pair of interest.

4.2 Experimental Fixture

A test fixture was developed to simulate a persistent wearing contact, in order to validate the use of multi-body simulation to replicate similar phenomena. The goal was to create wear in a very controlled environment, and one in which the materials could be easily changed. The fixture was also designed such that the lower portion of the fixture could be filled with sand, to investigate how the wear and friction models of a material change with the presence of the foreign particulates.

Shown in Figure 4.3, the test fixture consists of two bodies, the upper fixture (shown in silver), and the lower cup (shown in blue). The upper fixture holds three sample pieces, which are shown in yellow in Figure 4.4.

The red square shown in Figure 4.4 is the other test sample, and is fastened to the lower cup. The upper fixture is loaded with a test load to create a contact pressure between the

Figure 4.1: Abrasive wear testing machine

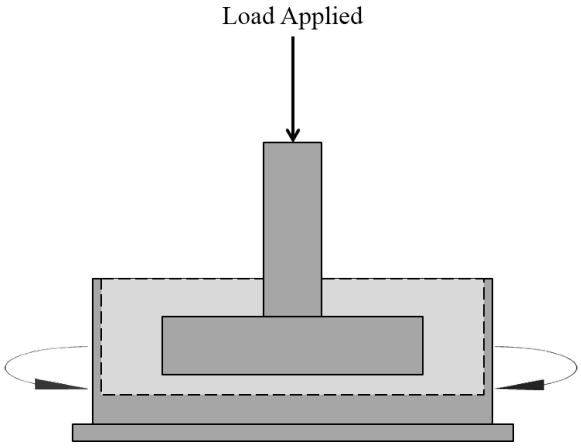

Figure 4.2: Diagram of test fixture, lower section rotates, upper does not

Figure 4.3: Entire persistent contact test fixture

yellow upper samples and the red lower sample. The lower cup is then rotated about the concentric axis, to create a relative sliding motion between the two fixtures.

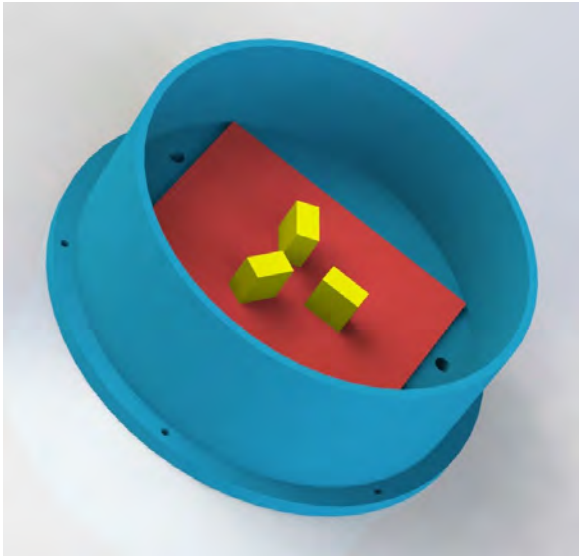

Figure 4.4: Persistent contact test fixture sample pieces

This test fixture will produce a constant vertical load and oscillatory, persistent contact, from which lumped wear coefficients can be calculated.

4.2.1 Materials

The material used for both the upper and lower samples was 7075-T7 Aluminum Alloy. The samples were both anodized according to MIL-A-8625 Type II Class 1. The materials and anodization of all parts used in this thesis were all selected to match each other as closely as possible.

4.3 Model Development

To create a multi-body simulation of the test fixture, the fixture was first modeled in SolidWorks. Shown in Figures 4.3 and 4.4, the SolidWorks model consisted of 6 parts, combined into a single assembly which was saved in a Parasolid format.

The Parasolid file of the assembly was then imported directly into ADAMS, where each piece was given a density so that their masses and inertial tensors could be calculated. Parasolid was used as the imported file format, because internally, ADAMS uses Parasolid as the

format for solids. Next the model had to be constrained. The upper samples were fastened to the upper fixture with lock joints, as was the lower sample to the lower fixture. The lower fixture was also fastened to ground rigidly. The upper fixture was given a cylindrical joint to ground to allow it to both rotate (to create the relative motion between the fixtures), and move vertically.

A number of forces were also included in the model. Each of the 3 upper samples were given solid to solid contact with the lower sample, with the contact parameters listed in Table 4.1.

Table 4.1: Case Study 1: Contact Parameters

Parameter	Value
Stiffness	1000
Force Exponent	2.0
Damping	500
Penetration Depth	0.01

The upper fixture was also given a downward force to simulate the load provided by the test fixture in the experiment. This downward force is either a constant value or changed over time according to a spline created from experimental data.

Finally, an angular motion was imposed on the cylindrical joint between the upper fixture and ground. This motion can be governed by either an equation or a spline created from experimental data[14].

The configuration of the model is shown in Figure 4.5.

With this model, any load and input motion can be simulated, and the average wear rate can be found. A screen capture of an example simulation is shown in Figure 4.6. In that Figure, the red arrows denote forces in the model, the 3 contact forces are equal, and their sum is the downward arrow denoting the load, as expected.

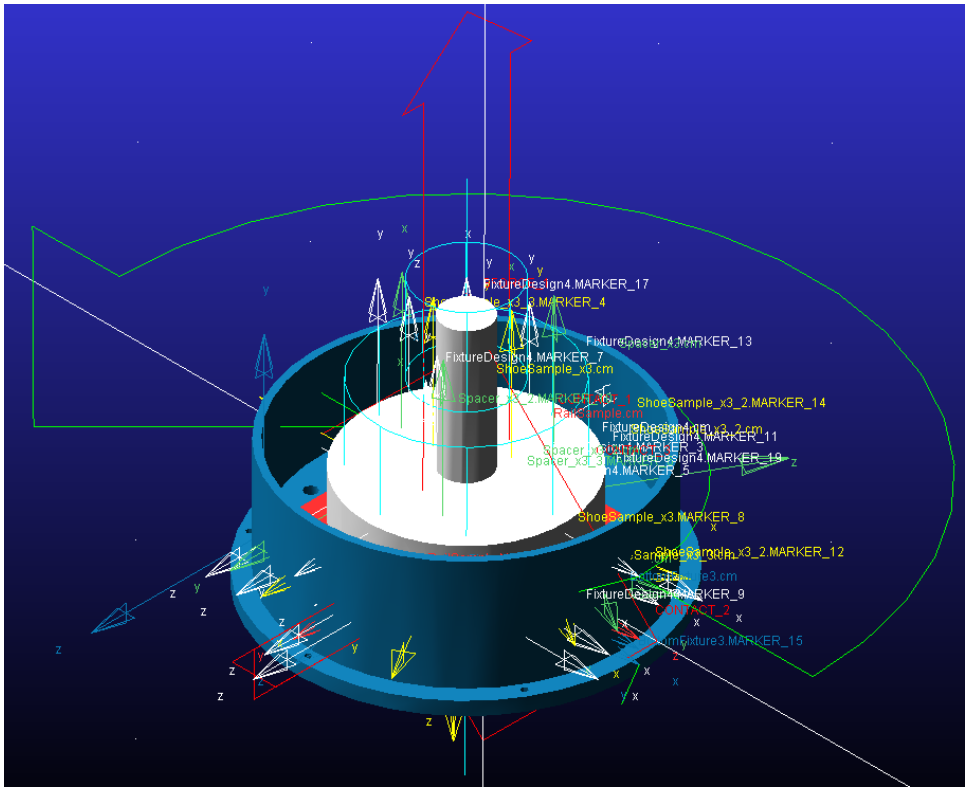

Figure 4.5: ADAMS model of persistent contact test fixture

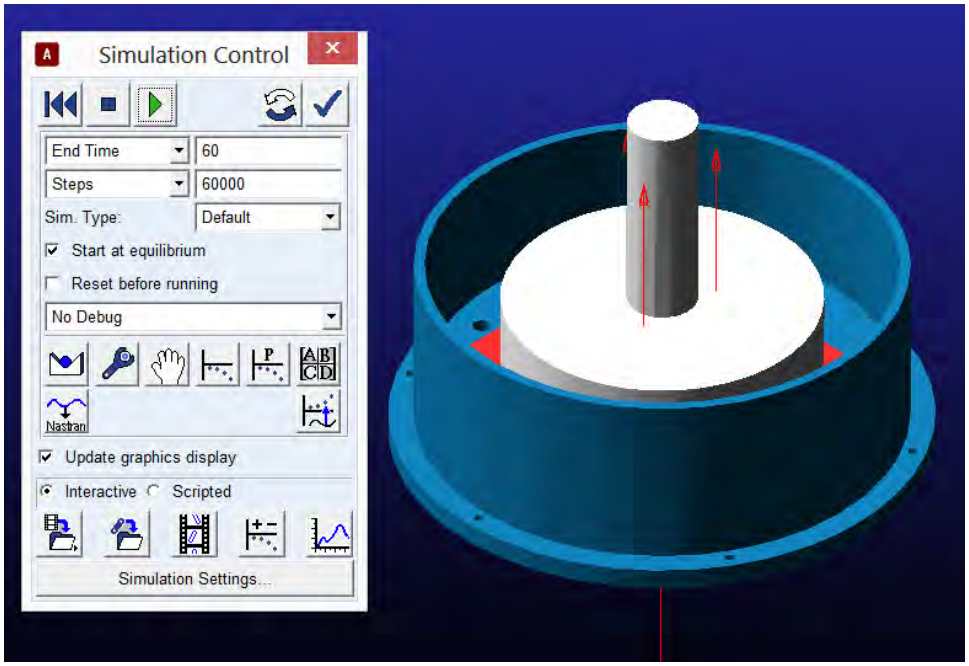

Figure 4.6: ADAMS simulation of persistent contact test fixture

4.3.1 Friction Parameter Estimation

In order to determine the friction parameters for a set of materials, one set of the experimental data was used. In this dataset, there were angular position, angular velocity, vertical force, torque and time data. The equation of motion for the system is:

$$J\ddot{\theta} = T_{in} - 3(F_{fric} r_{avg}) \tag{4.1}$$

Where J is the polar moment of inertia of the fixture, $\ddot{\theta}$ is the angular acceleration of the fixture, T_{in} is the input torque, r_{avg} is the radial distance to the center of the test items, and F_{fric} is defined by:

$$F_{fric} = N\ \mu(v_{slip}) \tag{4.2}$$

Where v_{slip} is the function of slip velocity which returns the coefficient of friction. The experimental data can therefore be used to calculate instantaneous $\mu(v_{slip})$ by rearranging the equation of motion as:

$$\mu(v_{slip}) = \left(\frac{T_{in} - J\ddot{\theta}}{3 r_{avg} N} \right) \tag{4.3}$$

And plotted relative to slip velocity, where the average bulk slip velocity is approximately:

$$v_{slip} \approx \dot{\theta} r_{avg} \tag{4.4}$$

The fitted model is shown plotted in Figure 4.7. The models were fit using MATLAB's NLINFIT function, and the equation form of the ADAMS friction model:

$$\mu(V) = \begin{cases} V = 0 & 0 \\ |V| = V_s & -sign(V)\mu_s \\ |V| >= V_d & -sign(V)\mu_d \\ V_s < |V| < V_d & -step(|V|, V_d, \mu_d, V_s, \mu_s)sign(V) \\ -V_s < V < V_s & step(V, -V_s, \mu_s, V_s, -\mu_s) \end{cases} \tag{4.5}$$

Where the step function is the ADAMS smooth step function:

$$step(x, x_1, h_1, x_2, h_2) = h_1 + \left(\frac{h_2 - h_1}{x_2 - x_1}\right)(x - x_1) - (\frac{h_2 - h_1}{2\pi})sin\left(\frac{2\pi}{x_2 - x_1}(x - x_1)\right) \tag{4.6}$$

The parameters which define the model are shown in Table 6.1.

Table 4.2: Estimated Friction Parameters

Parameter	Value (Dry)
Static Coefficient	0.1799
Dynamic Coefficient	0.166
Stiction Transition Velocity	6.46 mm/s
Friction Transition Velocity	12.0 mm/s

With the estimated friction values, the ADAMS model was complete, and ready for dynamic validation. In order to do this, the experimental position and vertical force data was splined and input into the model, which was simulated for 10 seconds. The data used for the dynamic validation was from a separate experimental run than was used to develop the friction model. The torque required to create this motion in ADAMS was then exported and compared to the experimental torque data. The comparison between the two is shown in Figure 4.8.

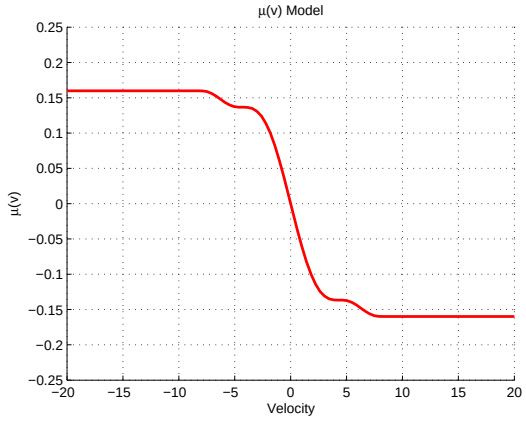

Figure 4.7: Fitted friction model

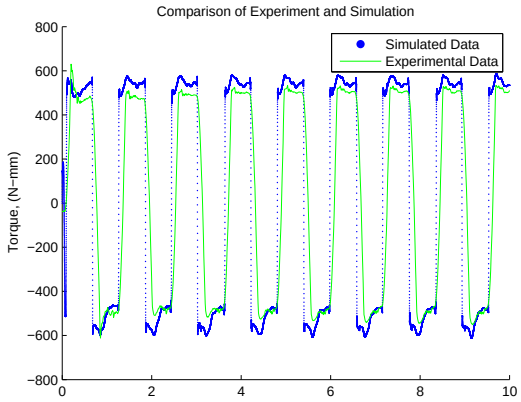

Figure 4.8: Comparison of experimental and simulated torques

As can be seen the experimental data closely matches that of the simulation, demonstrating accurate estimation of the friction model parameters.

4.4 Dimensional Lumped Wear Coefficient Modeling

With the ADAMS model completed, the experiment was run with a set of loads to find an experimental lumped wear coefficient model. Recall from the previous chapter:

$$K(\sigma) = C_1\sigma^{(1/C_2)} + C_3 \tag{4.7}$$

And therefore:

$$\frac{dW}{dt} = KNv_{slip} \tag{4.8}$$

$$\frac{dW}{dt} = C_1 v_{slip} N\sigma^{(1/C_2)} + C_3 N v_{slip} \tag{4.9}$$

Where v_{slip} is the time derivative of sliding distance. For the constant load cases examined in this chapter, the normal force is assumed to be time invariant. With this model, the predicted wear for a period of time t_{end} long is given by:

$$W(t_{end}) = \int_0^{t_{end}} \left(C_1 v_{slip} N \sigma^{(1/C_2)} + C_3 N v_{slip}\right) dt \tag{4.10}$$

The dimensional wear coefficient for a given set of experimental data can be found easily by parsing the load, torque, position, angular velocity and wear mass data. The normal force is given directly by the friction and wear tester. The average slip velocity is given by:

$$v_{slip} = \dot{\theta} r_{avg} \tag{4.11}$$

Where $\dot{\theta}$ is the angular velocity and r_{avg} is the radial distance from the center of the fixture to the centroid of the upper sample. Because of the assumption that K is a function of only contact pressure, which is constant throughout the test:

$$\frac{dW}{dt} = K(\sigma) N v_{slip} \tag{4.12}$$

$$W = \int_0^{t_{end}} \frac{dW}{dt} dt \tag{4.13}$$

$$W = \int_0^{t_{end}} K(\sigma) N v_{slip} dt \tag{4.14}$$

$$W = K(\sigma) N \int_0^{t_{end}} v_{slip} dt \tag{4.15}$$

$$K(\sigma) = \frac{W}{N \int_0^{t_{end}} v_{slip} dt} \tag{4.16}$$

Because wear mass, not volume, is measured in the experiment, recall that the wear mass is simply the product of the wear volume (W) and the bulk material density (ρ):

$$M_{wear} = W \rho \tag{4.17}$$

Giving the final relationship:

$$K(\sigma) = \frac{(M_{wear}/\rho)}{N r_{avg} \int_0^{t_{end}} \dot{\theta} dt} \tag{4.18}$$

Using this equation, the dimensional lumped wear coefficients for a number of loads were calculated, to allow a $K(\sigma)$ model to be fitted to experimental data. These data points and the fitted model are shown in Figure 4.9, and the coefficients which define the $K(\sigma)$ model are listed in Table 4.3. The Figure shows the relatively large scatter in the experimental data. Because of the uncertainty inherent in this type of experiment, a secondary validation experiment is required to ensure that the fitted wear model accurately represents the behavior of the materials of interest. The following chapter will investigate this further, validating the wear model.

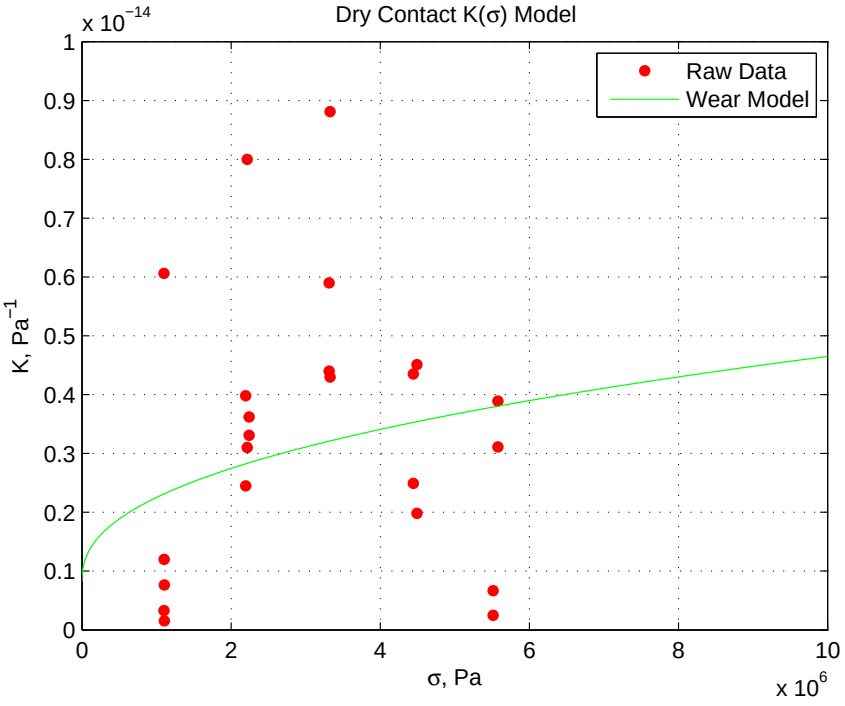

Figure 4.9: Experimental model for $K(\sigma)$

Table 4.3: Wear Model Parameters

Parameter	Value
C_1	2.3685×10^{-15}
C_2	6.3197
C_3	-9.5089×10^{-16}

4.4.1 Wear Prediction and Post-Processing

With the newly formed wear model, the wear resulting from arbitrary loadings and motion profiles can be predicted. After the simulation is completed in ADAMS, the ADAMS Post-Processor is used to export the data. Each contact has all of it's data stored in data structures referred to as 'Tracks'. Each track is a contact instance, such that when there are two areas of contact between the two solids at the same time, a new track is created to store the data from the 2nd contact area.

The contact area, stored in a variable request, must also be exported from the ADAMS Post-Processor.

In order to streamline this laborious process, a plugin was developed for ADAMS to perform this operation automatically.

The plugin is largely written in C, as a Windows Executable. ADAMS Command Language is used to automatically export all of the data, call the C script, and then import the results. All code is attached in the Appendix.

Because the wear prediction operation is done as a post-processing application, the first step is to create and run the model as normal. After the model has finished running, the plugin must be imported to the model (Figure 4.10). This operation should only need to be done once per model.

With the plugin imported, it can be located in the list of available dialog boxes to display as shown in Figure 4.11.

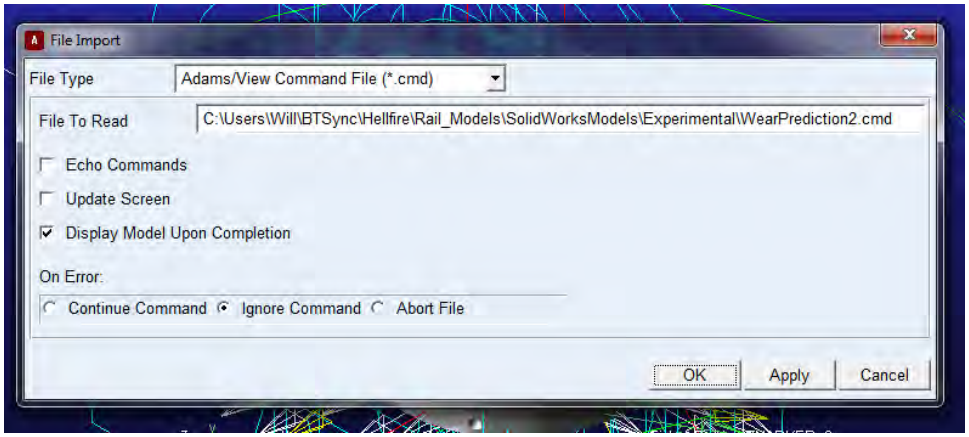

Figure 4.10: Import the command file containing the plugin

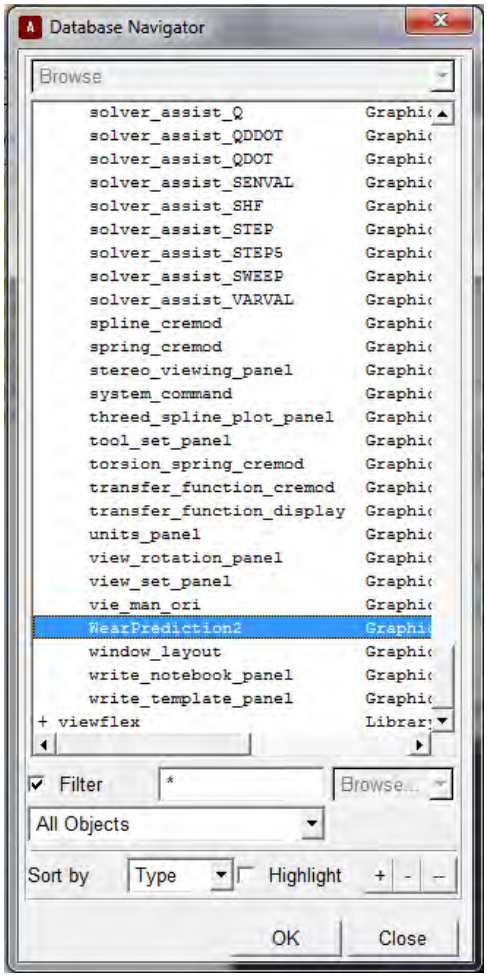

Figure 4.11: Display the custom dialog box

Once opened, the dialog box should be filled out with the name of the contact to calculate wear from, the name of the variable which tracks contact area for that contact,

and the wear model coefficients to be used. An example is shown in Figure 4.12. Once the script has finished running (it may take a few minutes), the wear data will appear in the data navigator as shown in the left of Figure 4.12.

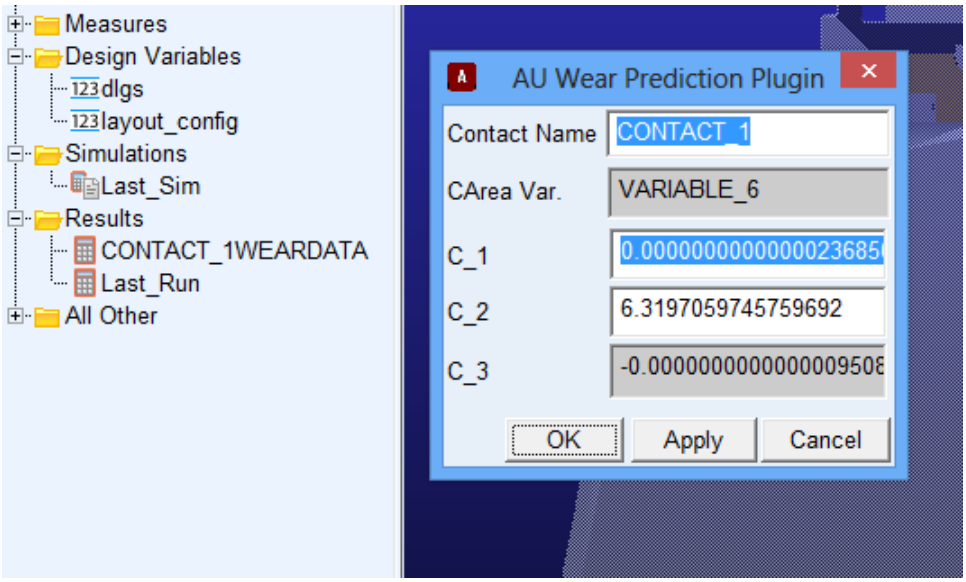

Figure 4.12: Example inputs; output in the data navigator

Now the wear volume over time (in cubic microns) can be plotted in the ADAMS post-processor as shown in Figure 4.13.

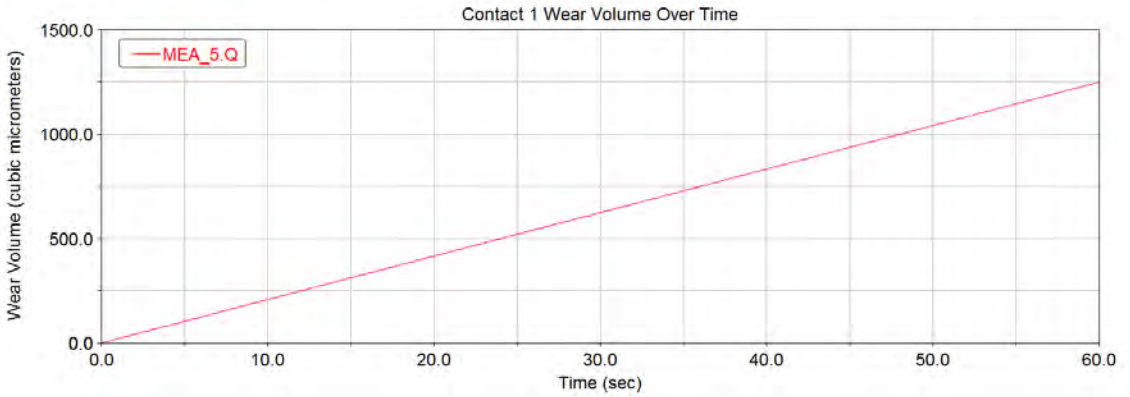

Figure 4.13: Plot newly generated wear data

4.5 Conclusion

In conclusion, a modified Archard wear model and a friction model can be developed for a material pair with a simple laboratory experiment. The friction model was easily validated using experimental dynamic data and ADAMS. The wear model will be validated in further experiments presented in Chapters 5 and 6.

With the friction and wear models developed, they were integrated into ADAMS by building a plugin which automatically exports the relevant data for a given contact, calculates wear in an external C application, and re-imports the calculated wear volume into the ADAMS Postprocessor. The code for this plugin is attached in the appendix.

Chapter 5

Case Study: Intermittent Contact

5.1 Introduction

Wear is often modeled in idealized cases which rely on persistent contact. In real engineering systems, however, the most wear prone areas are often those which come in and out of contact with each other. These 'rattling' cases are much more complex to model due to both the uncertainty of the dynamics and the ambiguity of the wear mechanism.

It is unclear whether rattling cases behave as repeated persistent contact events, and are therefore model-able by the modified Archard wear model used for the persistent contact case study, or whether an entirely new model entirely is required to properly predict wear in a rattling system.

For persistent contact, the wear coefficients are calculable with a simple experiment, so it would be ideal if those coefficients were also applicable to intermittent contact cases. This chapter presents a case study to validate the wear model developed in the previous chapter for use in a rattling system.

5.2 Rattling Wear

Take for example the case of a cube (test item) inside a very long rectangular tube (enclosure). Figure 5.3 shows an example cross section of this where the test item is marked with "A" and the enclosure is marked with "B". If the enclosure is vibrated, the test item will rattle inside it, impacting the various sides and if possible, tumbling.

The mode of wear in this case is very ambiguous. The relative penetration velocity of the test item and the enclosure at the time of impact is generally low, due to relatively short

periods of acceleration in a given direction, so the wear is not expected to clearly behave as impact. Further, the contact is not persistent, but rather comes in short periods of contact separated by full separation, so fretting may not fully govern the wear regime, despite short sliding distances.

5.3 Experimental Procedure

The hypothesis presented is that given an enclosure and test item, rattling contacts will behave like short, repeated persistent wear instances, and that if the dynamics are fully known, the contact wear can be modeled as a sum of repeated persistent contact wear events, such as those modeled in the previous chapter. This is to say, that the inclusion of relative velocities normal to the surfaces does not significantly impact the mechanisms of wear, or the lumped wear coefficients which model them. An experiment was developed to test this hypothesis.

5.3.1 Single Case

In order to determine if rattling contacts behave like short repetitive wear events, an experiment was designed to simulate this kind of wear. The procedure was to first calculate an lumped wear coefficient model as a function of pressure using the methodology from the previous chapter.

Second, a physical fixture of the rattling case, shown in Figures 5.1 and 5.2 was built and installed onto a shaker table. This fixture is a rectangular tube enclosure, with the ends capped off. Inside the enclosure sits a small cube, which is free to rattle within the enclosure. Both the enclosure and the cube inside it were made of anodized aluminum, such that the contact and friction models from the previous chapter could be used.

Finally, a multibody simulation model of the rattling fixture was built in ADAMS.

Using the dynamic values output from the ADAMS model and the lumped wear coefficient model from the wear test, a predicted wear rate was generated for the vibration

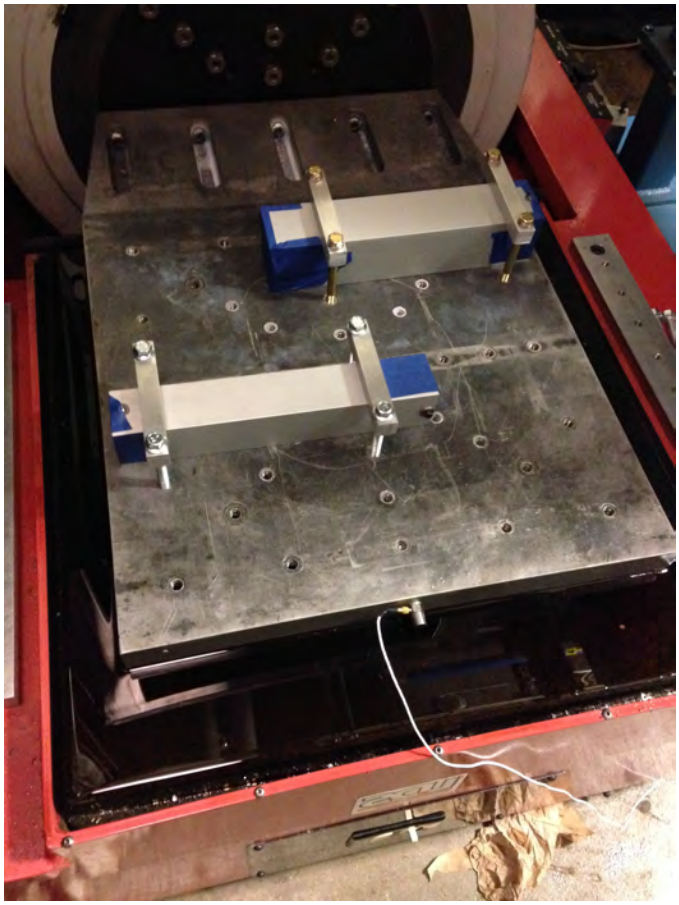

Figure 5.1: Experimental fixture on shaker table for transverse shaking (Cases A and B)

profile performed on the shaker table. By weighing the test item before and after the test, experimental values for wear rate were calculated. Finally, these values were compared to each other to either validate or invalidate the hypothesis for each case.

5.3.2 Multiple Cases

Rattling contacts can be described by the quantity known as rattle space, k_{rs}, which is the non-dimensional quantity:

$$k_{rs} = \frac{A_{ti}}{A_{encl}} \tag{5.1}$$

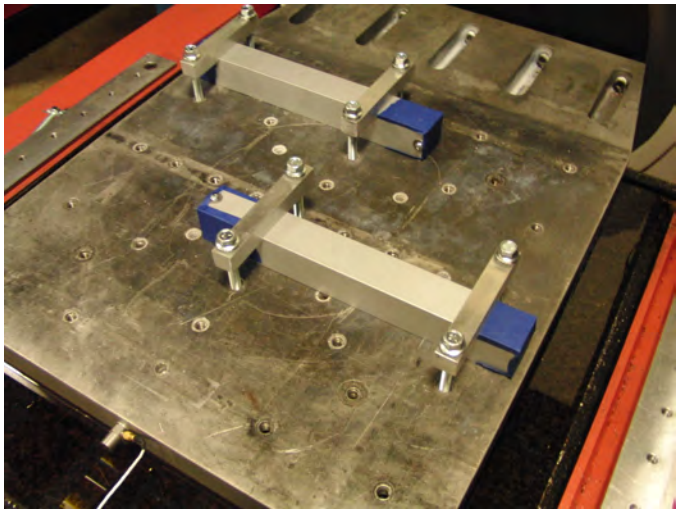

Figure 5.2: Experimental fixture on shaker table for transverse shaking (Cases C and D)

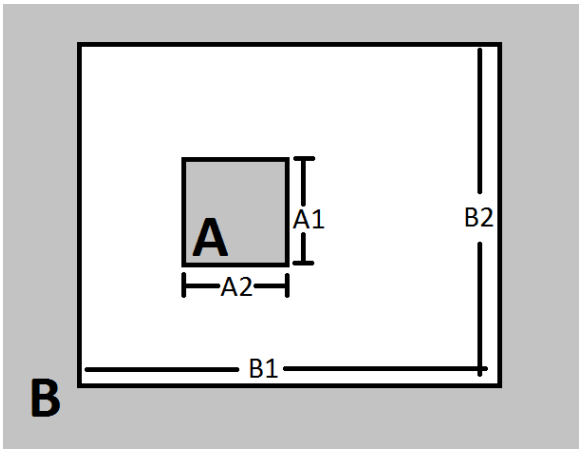

Figure 5.3: Generalized rattle space diagram

Or the ratio of test item area to enclosure area. Area is considered rather than volume due the fact that the enclosures will only be excited in one axis, and are very long in another axis, meaning the motion is approximately planar. In the case described in Figure 5.3:

$$k_{rs} = \frac{A1 \times A2}{B1 \times B2} \tag{5.2}$$

Four cases were selected to characterize the whole spectrum of rattle spaces. Using a 1 inch cube for a test item, the size of the enclosures and their corresponding rattle spaces

Table 5.1: Enclosure Sizing

Case	Rattle Space	Enclosure Area	Enclosure Side L
A	0.25	4.000 in^2	2.0 in
B	0.44	2.250 in^2	1.5 in
C	0.64	1.562 in^2	1.25 in
D	0.79	1.265 in^2	1.125 in

are shown in Table 5.1. The experimental procedure outlined above was repeated for all 4 cases, to generate experimental wear values for each rattle space.

A further 2 cases have closed form solutions. When the rattle space is 1, there is no space for relative motion between the enclosure and test item, meaning the wear mass will be 0 and the contact forces will be easily calculable using Newton's second law. When the rattle space is 0, the test item has an area of 0, which means that wear, normal force and relative velocity will all be 0. Alternatively, if the rattle space is 0, the enclosure may be infinitely large and the test item will never touch it's sides, resulting in no wear.

5.4 Model Development

A multibody simulation model was developed in ADAMS to estimate the wear power for each rattle space given an input vibration profile. The ADAMS model used a simple square tube enclosure and a cube for the test item. A very small roundover (0.02" radius) was added to all edges to better model the real fixture. If this step is not done, the dynamics and forces predicted by the ADAMS solver will not change significantly, but the contact area will. This produces artificially high contact pressure predictions. The mass properties of each were selected to match those of the real fixture. The contact parameters used are shown in Table 5.2.

Contact models with these parameters were created between the test item and the enclosure. The only other force present was gravity. Finally to complete the model, a motion

Table 5.2: Contact Parameters

Parameter	Value
Static Coefficient	0.1799
Dynamic Coefficient	0.1660
Stiction Transition Velocity	6.4634 mm/s
Friction Transition Velocity	12.00 mm/s
Stiffness	1000
Force Exponent	2.0
Damping	500
Penetration Depth	0.01

generator between the ground and the enclosure was created to simulate the vibration input of the shaker table.

The completed ADAMS model is shown in Figure 5.4, in the midst of a simulation.

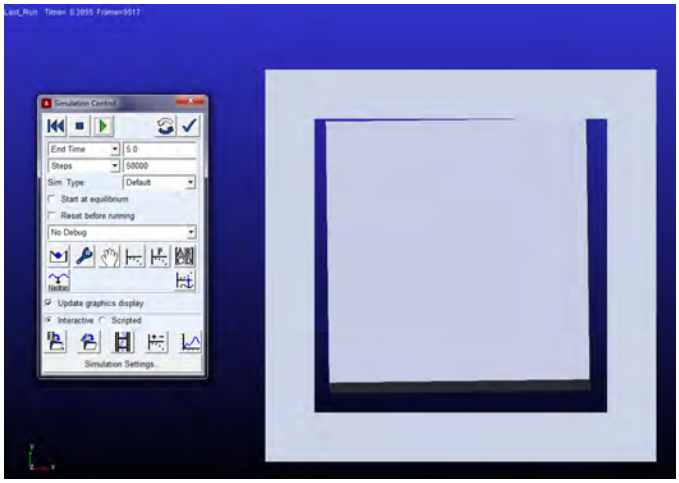

Figure 5.4: Simulation of rattling cube in ADAMS

Using the process outlined in Chapter 3, a variable was created to track the contact area for each solid-solid contact.

5.5 Results

Before even making any wear predictions, the ADAMS models showed interesting results. Figure 5.5 shows the average predicted normal force and slip velocity for each case

simulated. The vibration used for each case was 15G amplitude at 20Hz. As one would expect, the tightest case (Case D) has the lowest normal forces and slip velocities, but the loosest case (Case A) actually is predicted to have lower slip velocity than the 2nd loosest (Case B).

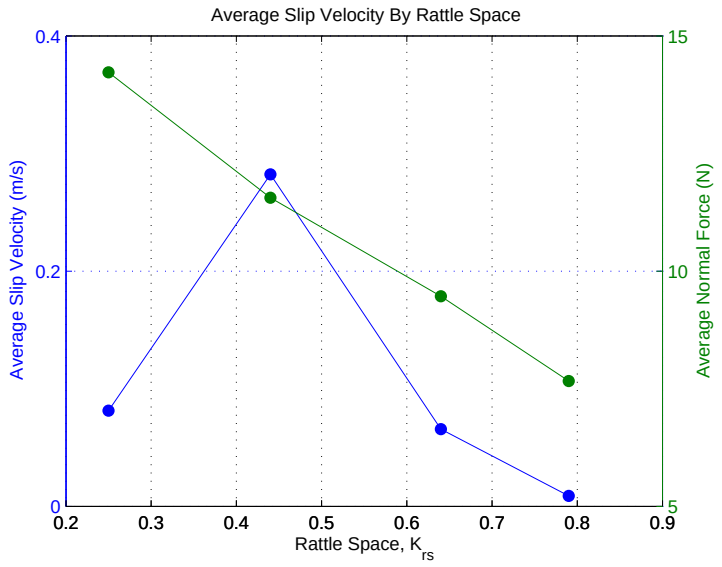

Figure 5.5: Predicted normal forces and slip velocities from ADAMS model

By tracking the predicted contact area, the average contact surface pressure can be found, as is shown in Figure 5.6. Again, case A seems to be different than the other cases, with drastically higher predicted stresses. In fact, many of the predicted stresses are well above the yield strength of the grade of aluminum used in the experimental fixture.

In light of these extreme predictions, the two looser cases (A and B) are not expected to be well modeled by the modified Archard equation or the multibody simulation code. Despite this fact, all cases were still run in both simulation and experiment. Figure 5.11 shows the predicted wear rates (in mg per hour) of each case, with the experimental values measured from 90 minutes of experimental time on the shaker table. The test was done in 90 minute increments due to facility use restrictions.

After just one increment, as can be seen in the figure, the presumption that the model would not hold for the looser cases was confirmed in a resounding fashion. As will be shown

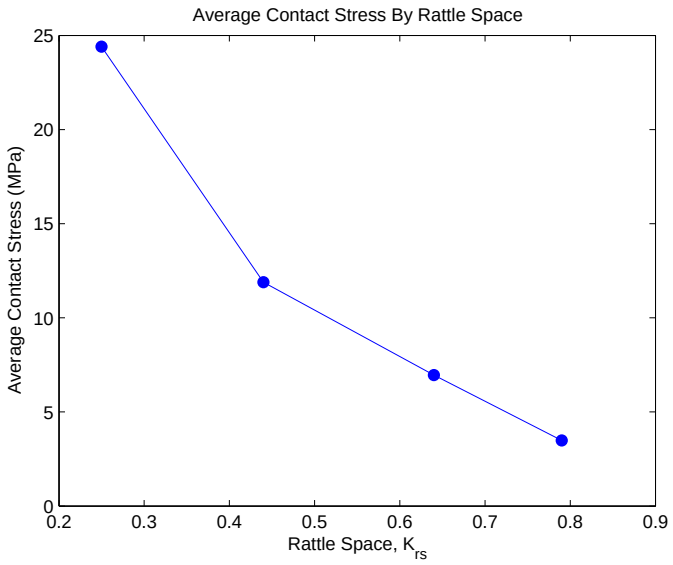

Figure 5.6: Predicted contact stresses from ADAMS model

in the next chapter, the Hellfire Missile Launcher system typically has rattle spaces near 0.8, so successful wear predictions in similarly sized rattle boxes were taken as validation of the wear prediction process for this system.

After this 90 minutes, each experimental test item was also photographed. Figures 5.7, 5.8, 5.9, and 5.10 show the sample of case A, B, C, and D respectively. While the plot in Figure 5.11 seemed to indicate that case A was better modeled than case B, the pictures show that case A has in fact undergone a very different wear and deformation regime than the other samples, exhibiting plastic flow of material from the corners, indicating that the impact stresses at and near the corners exceeded the material's yield strength.

With cases A and B confirmed to be badly modeled by this process, cases C and D were more closely examined. 2 more 90 minute experiments were run on each sample for a total of 4.5 hours of testing on each. Figure 5.12 shows the predictions for just these two cases along with the experimental results. These values match up much more closely. The experimental values and error values are shown in Tables 5.3 and 5.4.

Figure 5.7: Test item from case A after 90 minutes

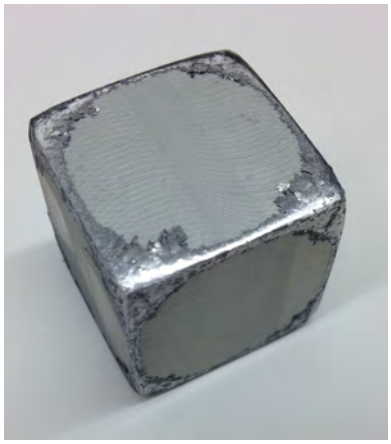

Figure 5.8: Test item from case B after 90 minutes

Figure 5.9: Test item from case C after 90 minutes

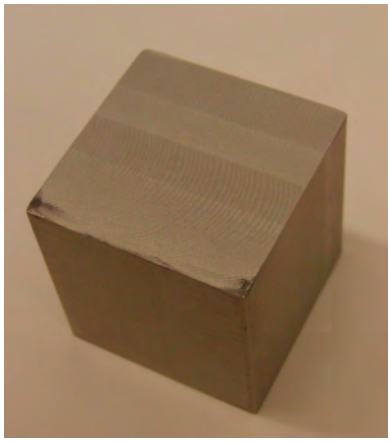

Figure 5.10: Test item from case D after 90 minutes

Table 5.3: Wear Rate, Experimental Results

Case	Rattle Space	Wear Rate 1	Wear Rate 2	Wear Rate 3	Average Wear Rate
Min	0.00	0.00	0.00	0.00	0.00
A	0.25	21.1333 mg/hr	n/a	n/a	21.1333 mg/hr
B	0.44	97.2000 mg/hr	n/a	n/a	97.2000 mg/hr
C	0.64	1.5333 mg/hr	2.2333 mg/hr	2.3333 mg/hr	2.0333 mg/hr
D	0.79	0.2666 mg/hr	0.1666 mg/hr	0.8666 mg/hr	0.4333 mg/hr
Max	1.00	0.00	0.00	0.00	0.00

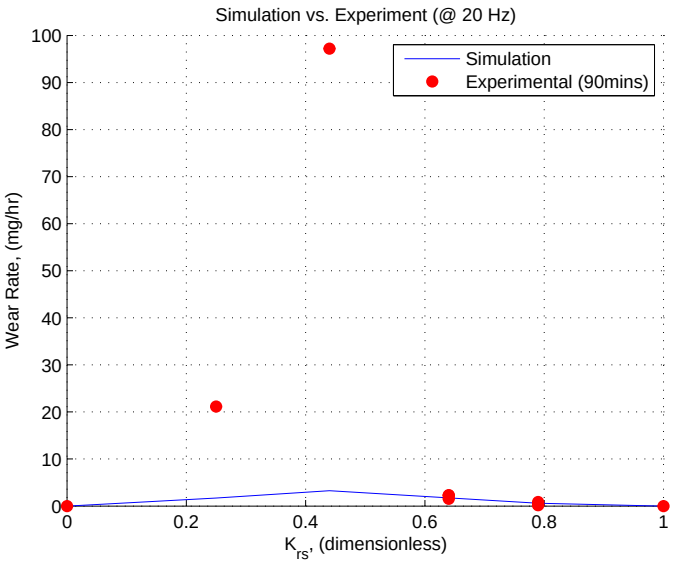

Figure 5.11: Predicted wear rates from ADAMS model (All cases), plot of data from Table 5.3

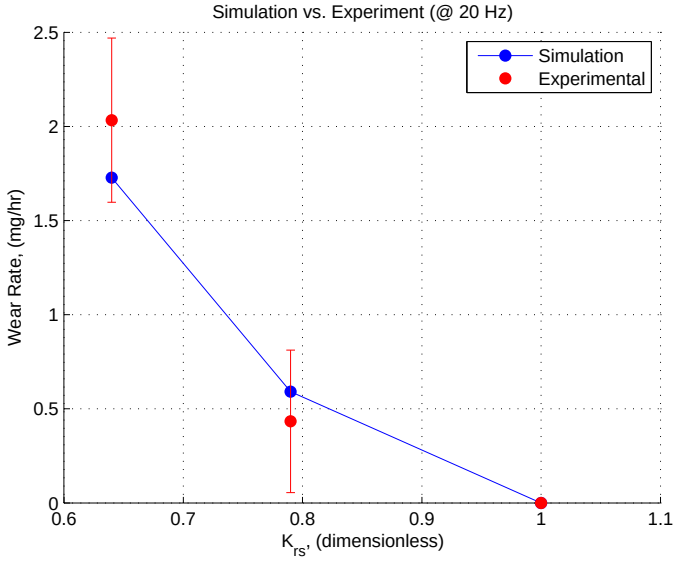

Figure 5.12: Predicted wear rates from ADAMS model (Cases C and D only), plot of data from Table 5.4

5.6 Conclusion

In this chapter, a multibody simulation model was developed to predict wear for rattling systems. It was demonstrated that this methodology can be used for lower contact stresses, but that as the contact stress approaches and surpasses the yield strength, the validity of the

Table 5.4: Wear Rate, Simulated Results and Error Based on Average Experimental Wear Rate

Case	Rattle Space	Simulated Wear Rate	Abs. Error	Percent Error
Min	0.00	0.00 mg/hr	0.00	0.00%
A	0.25	1.7124 mg/hr	19.4209 mg/hr	91.03%
B	0.44	3.2622 mg/hr	93.9378 mg/hr	96.6%
C	0.64	1.7278 mg/hr	0.3055 mg/hr	15.02%
D	0.79	0.5910 mg/hr	0.1577 mg/hr	36.39%
Max	1.00	0.00 mg/hr	0.00	0.00%

dynamic simulation and the validity of the experimentally derived wear model both break down. Because of this, wear predictions for looser configurations are much less accurate than those of tighter configurations. The more accurate predictions for this experiment did occour in the rattle space ranges expected for the Hellfire Missile Launcher System, of 0.8.

Also presented were a modified Archard wear equation in which the dimensional wear coefficient is a function of contact stress, and the concept of 'rattle space' to parameterize a rattling system.

Chapter 6

Case Study: Hellfire Missile Launcher System

6.1 Introduction

The final case study is the Hellfire missile launcher system discussed in the first chapter. In this case, extreme wear on the components of the missile launcher occurs during captive carry flight. Examples of this severe wear are seen in Figures 6.1 and 6.2. In these photographs the drastic wear is clearly evident.

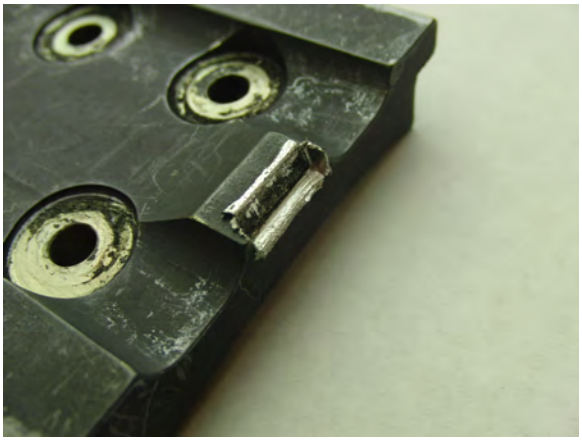

Figure 6.1: Example of wear on a mid shoe

Figure 6.2: Example of wear on rail extrusion

With wear on the shoe (Figure 6.1) and the rail extrusion (Figure 6.2) possible, it is important to note that the wear on the shoes is preferable. Due to the nature of the system (shoes mounted to missiles and rail mounted to vehicle), the shoes are relatively disposable. With this in mind, the goal of the designer is to both reduce aggregate system wear and to minimize the portion of that wear which occurs on the rail extrusion.

In light of this risk, it is clear that a means of estimating the usable lifetime of the rail system in the presence of contaminants such as sand is critical to the safety and lethality of

the warfighter. Using the methodologies presented in previous chapters and case studies, a model is to be developed to predict system wear.

6.2 Hellfire Missile Launcher System Overview

In a system as complex as a Hellfire missile launcher, a number of nomenclature points first had to be established to ensure consistency and clarity. Wherever possible, the terminology was picked to match with that used originally by the designers of the Hellfire missile system itself. The first distinction to be made was between the rail system and the missile system.

The rail system is comprised of a rail extrusion and 9 subsystems which are physically attached to it. The rail serves as a temporary mount for the Hellfire missile before firing. It physically and electronically connects the Hellfire missile to the delivery vehicle. The missile system is comprised of the two shoes which fasten to the missile and allow it to ride on the rails of the rail system, as well as the missile itself.

6.2.1 Faces

The system has 6 primary faces. They are denoted as follows:

Top The top face is the face which faces the vehicle. It is opposite of the rails, and contains features such as the grounding wire, latch cover and many of the mounting bolts. The top face of the shoes are the ones which face the rails of the rail system.

Bottom The bottom or rail face is the most prominent face of the rail system, and contains the rails themselves. On the shoe system, the bottom face corresponds to the one which is against the missile itself.

Front The front face is the face which faces towards the direction of firing. It contains the front sensor apparatus.

Back The back face is the face opposite the front face, and contains the back face cap.

Left The left face is the face on the left side if the viewer were standing at the rear of the rail and facing towards the front face.

Right The right face is the face on the right side if the viewer were standing at the rear of the rail and facing towards the front face.

6.2.2 Coordinate Systems

The general coordinate system will be centered at the center of mass of the rail, and affixed to it's body. The z-direction (Longitudinal) will be from the center of mass along the long axis of the rail in the direction of firing. The y-axis (Vertical) will extend directly vertical, towards the rail and the mounting vehicle. The x-axis (Transverse) will complete the frame according to the right hand rule.

6.2.3 Sub-Assemblies

There are 9 primary sub-assemblies on a Hellfire missile rail.

The front sensor apparatus, shown in Figure 6.3 is the subassembly on the front end of the rail system which has a retracting cover and an electrical connector mounted on it. It is mounted with two mounting bolts (left and right), which fasten to the rail extrusion itself. The main body remains fixed, while the cover rotates on an axle, loaded by a spring. The electrical connector is fixed to the main body, and covered by the cover. A wire extends from the connector to the sensor probes through the rail extrusion, to which the wire is fastened with 2 tie downs (front and rear).

The rail extrusion subsystem has only one component, the rail itself, shown in Figure 6.4. It is a single extruded piece of aluminum which is then machined and coated to form the chassis for the entire rail system.

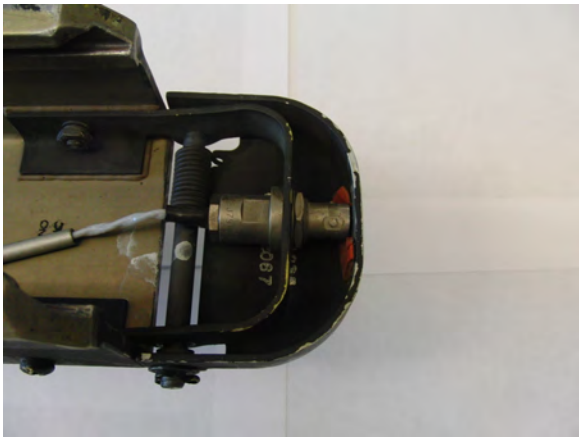

Figure 6.3: Front sensor apparatus with cover closed

Figure 6.4: Rear end of the rail extrusion

The mickey mouse ears apparatus, shown in Figure 6.5, serves as a dust and contaminant shield for the sensor probes for when there is no missile in the rail. It is a simple linkage system with a main body, finger (which is deployed by the missile to stow the ears), 2 links and a push-rod, and the 'ears' themselves, which cover the sensor probes. The apparatus is mounted to the rail extrusion with two mounting bolts (front and rear).

The sensor probes, shown in Figure 6.6 are two tubular electrical connection probes which are attached to the rail. When the missile is loaded into the rails, the probes fasten to it, connecting the missile to the rail electronically. At the rear of the probes, a number of cables connect to the various sensors and other devices along the rail. The springs shown in the figure are referred to as double barrel springs, and act on the mid shoe in the opposite direction as the latch.

The latch cover, shown in Figure 6.7, is a tub which houses the exposed portions of the latch. It is fastened to the rail extrusion with 6 bolts.

On some of the rails, a grounding apparatus can be found on the rear end of the top face, as shown in Figure 6.8. In more recently manufactured rails, this grounding wire is either removed or not present at all.

The latch itself, shown in Figures 6.9 and 6.10 from the rail and top faces respectively, retains the missile in the rail on the longitudinal axis. It does this by pressing against the

Figure 6.5: Mickey mouse ears apparatus from the rail face

Figure 6.6: Example of sensor probes, with double barrel springs

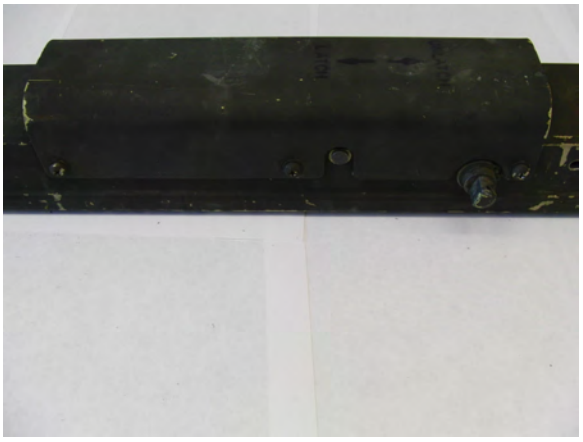

Figure 6.7: Example of a latch cover from left face

Figure 6.8: Example of an old style hard wire grounding apparatus

mid shoe, loaded by two concentric coil springs. The latch face is on a large rocker arm, which can be engaged or disengaged with an engagement rod. An electrical contact switch is engaged to determine whether or not the missile is loaded.

On the rear end of the rail face, at the location of the rear shoe - rail interface, there is a leaf spring. This spring is both adhered and bolted to the rail extrusion. It presses downward on the rear shoe to hold it against the rail firmly. An example of a rear spring is shown in Figure 6.11.

The exposed back face of the rail extrusion is protected by a back face cap as shown in Figure 6.12. It is fastened to the rail with two mounting bolts.

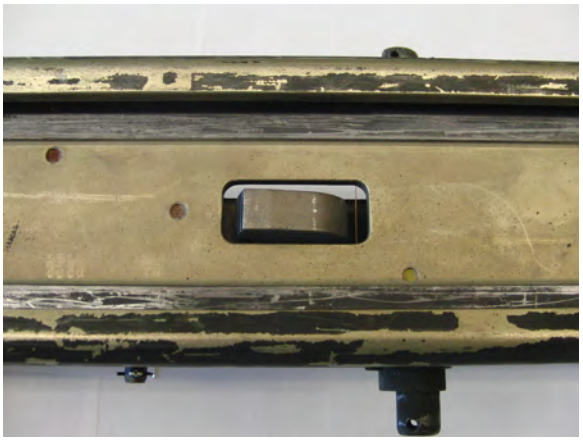

Figure 6.9: Latch from the rail face

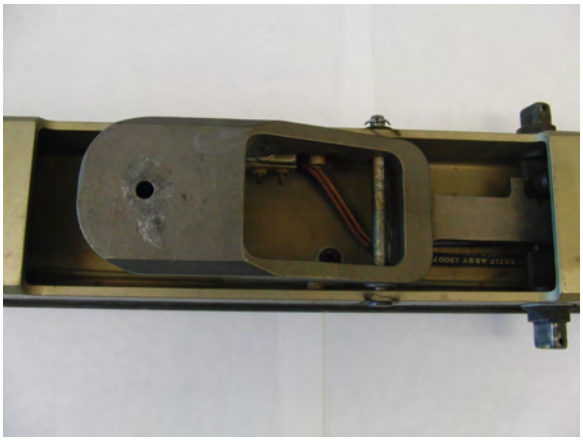

Figure 6.10: Latch from top face

Figure 6.11: Example of a rear spring

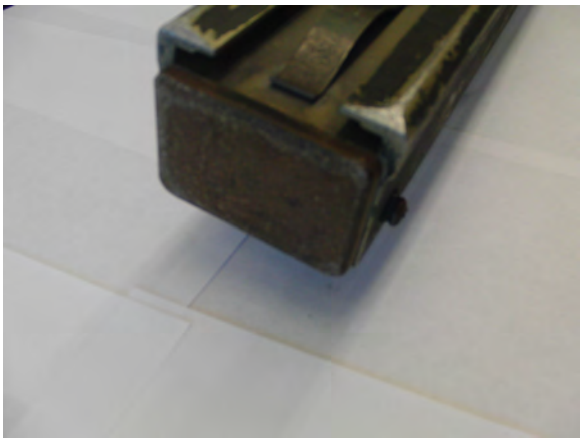

Figure 6.12: Example of a back face cap

The rear shoe, shown in Figure 6.13, is a single piece of extruded and machined aluminum. It mounts to the missile with 4 bolts.

The mid shoe, shown in Figure 6.14, is also a single piece of extruded and machined aluminum, while it is very similar to the rear shoe, it is a different size and also has a protrusion on its front side to enlarge the interface with the latch. It mounts to the missile with 4 bolts.

6.3 Preliminary Wear Analysis

To begin the analysis of the Hellfire Missile Launcher System, a set of brand new shoes and rail were shaken to create wear typical of the captive carry environment. This was used

Figure 6.13: Example of a rear shoe

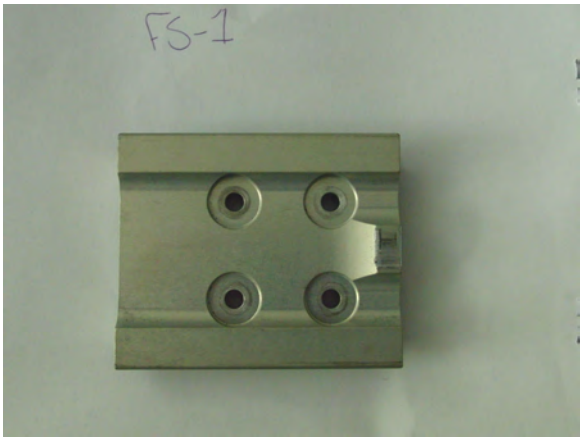

Figure 6.14: Example of a mid shoe

to identify problem areas and to establish the starting condition of the system in terms of wear resistance.

6.3.1 New Experimental Rail and Shoes

Brand new shoes and a brand new rail were used in the experiment. The only out-of-the-ordinary aspect of the setup was that the rail-rear shoe interface was out-of-spec loose. The rear leaf spring was not engaged when the system was static.

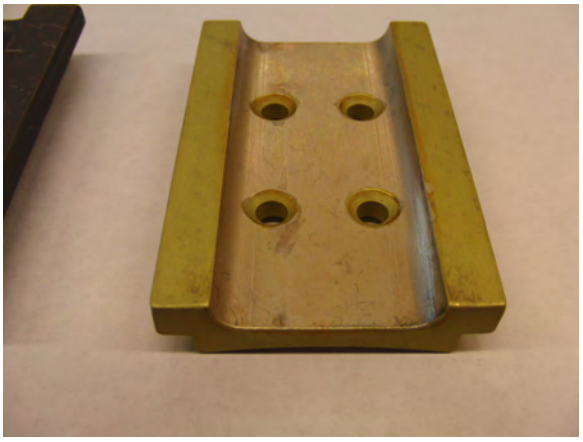

Figure 6.15: Before: rear shoe

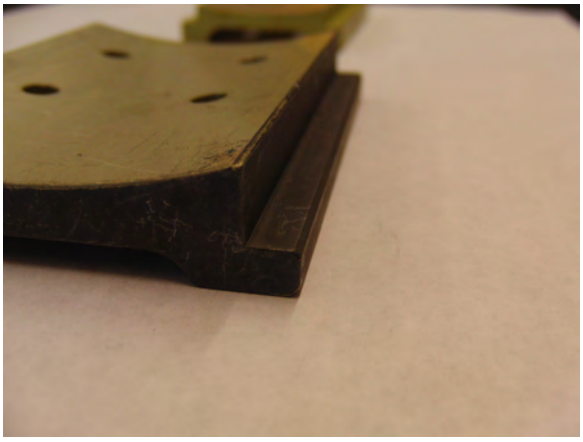

Figure 6.16: Before: mid shoe faces

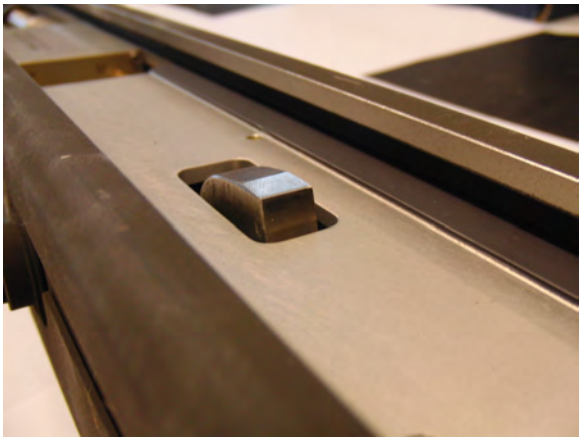

Figure 6.17: Before: latch face and rail profile

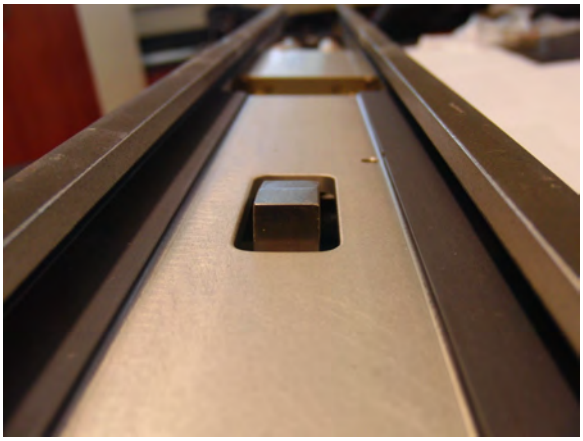

Figure 6.18: Before: latch face and rail profile

Figure 6.19: Before: rear leaf spring and rail edge

Figure 6.20: Before: mickey mouse ears finger

6.3.2 Worn Experimental Rail and Shoes

The most significant wear ended up being at the latch face-mid shoe interface, shown in Figure 6.64. A massive amount of material was removed here. Another interesting phenomena was the 'bumpy' surfaces of both the mid and rear shoes, shown in Figures 6.21 and 6.24. This seemed to indicate a more complex motion than just sliding and perhaps adhesive wear. Slight wear, shown in Figure 6.23, occurred on the top of the rear shoe, where it was in contact with the rear leaf spring.

On the rail, there was only slight wear from the rail's interaction with the rear shoe (Figure 6.25), the mid shoe (Figure 6.26), the missile itself (Figure 6.27), and on the mickey mouse ears finger (Figure 6.28).

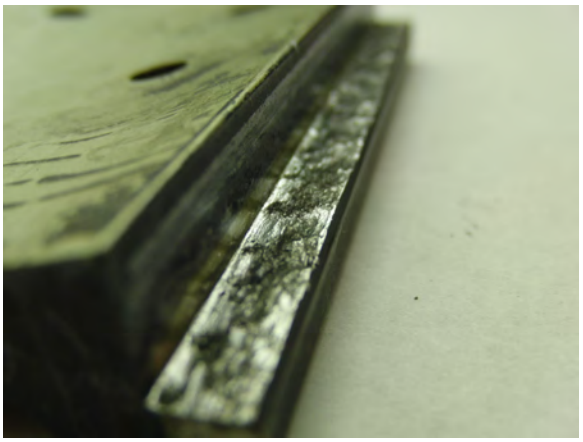

Figure 6.21: After: mid shoe worn surface

Figure 6.22: After: mid shoe heavy wear from latch face

Figure 6.23: After: rear shoe wear from leaf spring

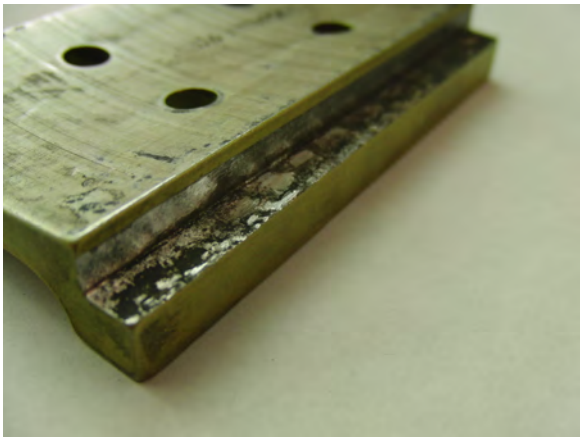

Figure 6.24: After: rear shoe worn surface

6.4 System Configurations

There are a number of parameters which affect the wear rates and associated service life of a missile launcher system. They include the environment in which the system is operating, the tolerances of the manufacturing process, and the vehicle to which the system is mounted.

Figure 6.25: After: rail edge wear from rear shoe

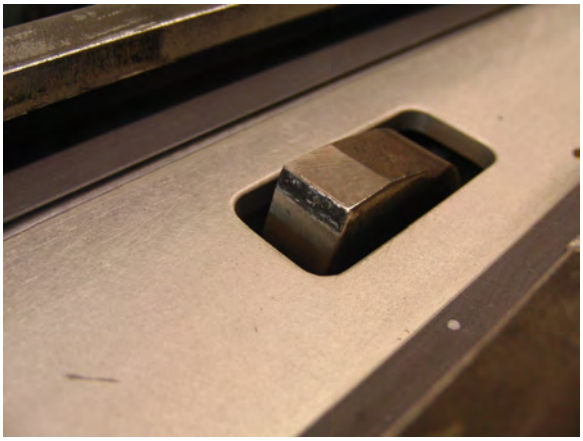

Figure 6.26: After: latch face wear from mid shoe

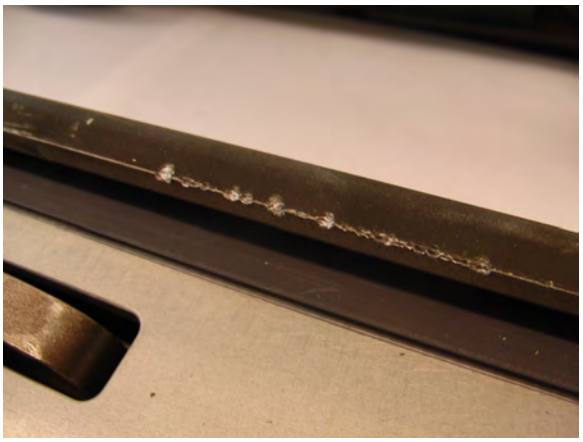

Figure 6.27: After: rail damage from missile

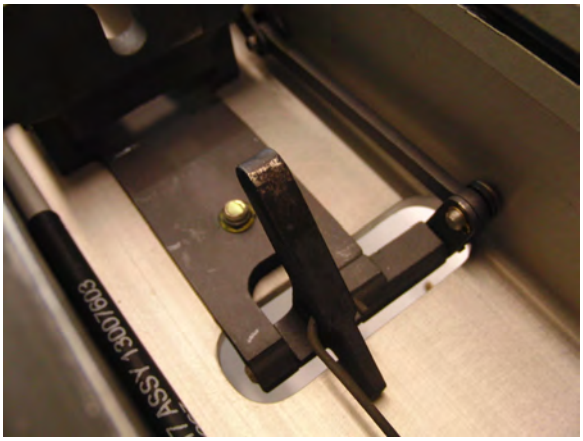

Figure 6.28: After: mickey mouse ears finger wear

There are a number of parameters which affect the wear rates and associated service life of a missile launcher system. They include the environment in which the system is operating, the tolerances of the manufacturing process, and the vehicle to which the system is mounted.

While the Hellfire missile launcher system was likely designed with a certain configuration in mind, the long life of the program and the systems increasing prevalence have led to a number of variables which effect the wear of the missile launcher, but have not been well characterized or accounted for yet. Qualitatively, the much higher wear at the front and rear shoe interfaces relative to the mid shoe seemed to indicate that the missile's relative motion

with the rail was generally oscillating about the mid shoe, leading to higher slip velocities at the ends of the rail than at the middle.

6.4.1 Environments

As military engagement in the Middle East and North Africa continues, sandy environments continue to accelerate wear on critical hardware. The extremely hard particulates and very fine grain size have been shown to accelerate wear considerably. In order to properly quantify this, a methodology for finding friction and wear models for arbitrary material pairs and contaminants or lubricants is required. In this report, the simplest case of no contaminant will be examined to demonstrate the methodology of analyzing a configuration.

As military engagement in the Middle East and North Africa continues, sandy environments continue to accelerate wear on critical hardware. The extremely hard particulates and very fine grain size have been shown to accelerate wear considerably. In order to properly quantify this, a methodology for categorizing the sand present in an area and then quantifying it's impact on wear rates is required.

As military engagement in the Middle East and North Africa continues, sandy environments continue to accelerate wear on critical hardware. The extremely hard particulates and very fine grain size have been shown to accelerate wear considerably. In order to properly quantify this, a methodology for finding friction and wear models for arbitrary material pairs and contaminants or lubricants is required. In this report, the simplest case of no contaminant will be examined to demonstrate the methodology of analyzing a configuration.

6.4.2 Tolerances

As with any manufactured product, the dimensions of all rails are not identical. Quality control measures bound the dimensions to within some range, but there is still variance within that range. In order to characterize the change in service life based on the geometry of the rail, the boundary configurations were examined: tight and loose. As the names imply,

these configurations correspond to the tightest and loosest rail shapes which fall within the specified tolerances.

As with any manufactured product, the dimensions of all rails are not identical. Quality control measures bound the dimensions to within some range, but within that range there are a number of configurations. In order to characterize the change in service life based on the geometry of the rail, 3 configurations are to be examined: tight, design and loose. As the names imply, these configurations correspond to the tightest and loosest rail shapes which fall within the specified tolerances, as well as the exact specified design dimensions.

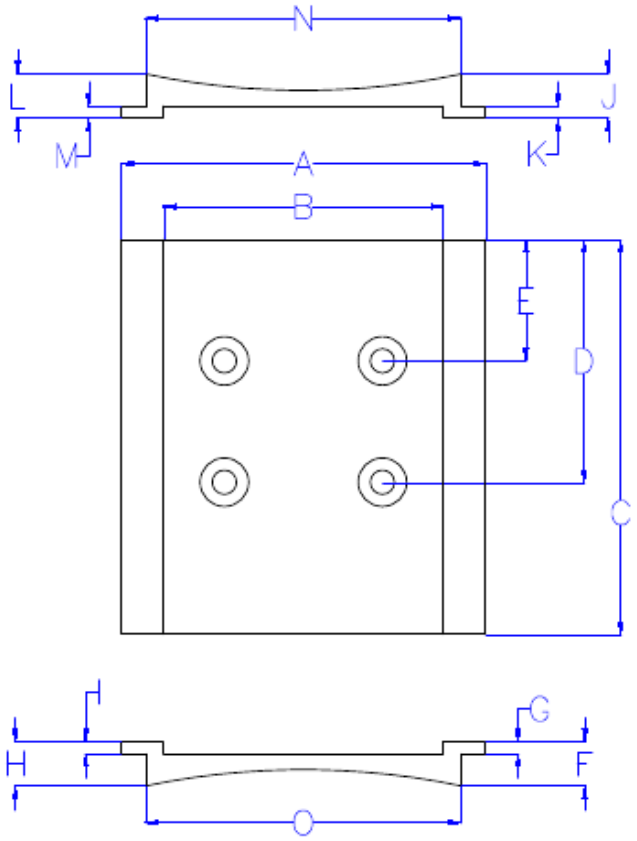

Figure 6.29: Dimensions of a rear shoe

In the general case of a rattling contact, where the two objects have neither constant contact with each other, nor high relative velocities, the system can be characterized by rattle space. In the Hellfire Missile Launcher case, the enclosure is the rail extrusion (Figure

6.30) and the rattling item is the shoe (Figure 6.29). As a simplification, only the rattle space value of the rear shoe/extrusion interface will be considered, because it is easiest to measure in the field.

The area of the enclosure, as drawn in Figure 6.30 is approximated as:

$$\begin{aligned} A_{encl} =& mean\left(L_{shoe} - M_{shoe}, J_{shoe} - K_{shoe}, F_{shoe} - G_{shoe}, H_{shoe} - I_{shoe}\right) B_{rail} \\ &+ \left(2\ mean\left(\frac{A_{shoe} - N_{shoe}}{2}, \frac{A_{shoe} - O_{shoe}}{2}\right) + B_{rail}\right) mean\left(N_{rail}, Q_{rail}\right) \end{aligned} \tag{6.1}$$

And the area of the rattling item, as drawn in Figure 6.29 is approximated as:

$$A_{ti} = N_{rail}N_{shoe} + K_{shoe}(A_{shoe} - N_{shoe}) \tag{6.2}$$

Where the subscript *shoe* denotes a dimension of the shoe, a subscript of *rail* denotes a dimension of the rail, and rattle space (k_{rs}), is found by:

$$k_{rs} = \frac{A_{ti}}{A_{encl}} \tag{6.3}$$

Using these formulas the tightest, designed and loosest configurations correspond to k_{rs} values of 0.8317, 0.7990, and 0.7684 respectively.

6.4.3 Vehicles

As mentioned previously, the Hellfire missile system is used globaly on a number of different vehicles. In this report, the vehicles are modeled using MIL-810G, which contains vibration profiles for many vehicles. Only one vehicles is examined here, but any other vehicle evaluated in MIL-810G can be evaluated in terms of missile launcher wear by using the exact same methodology.

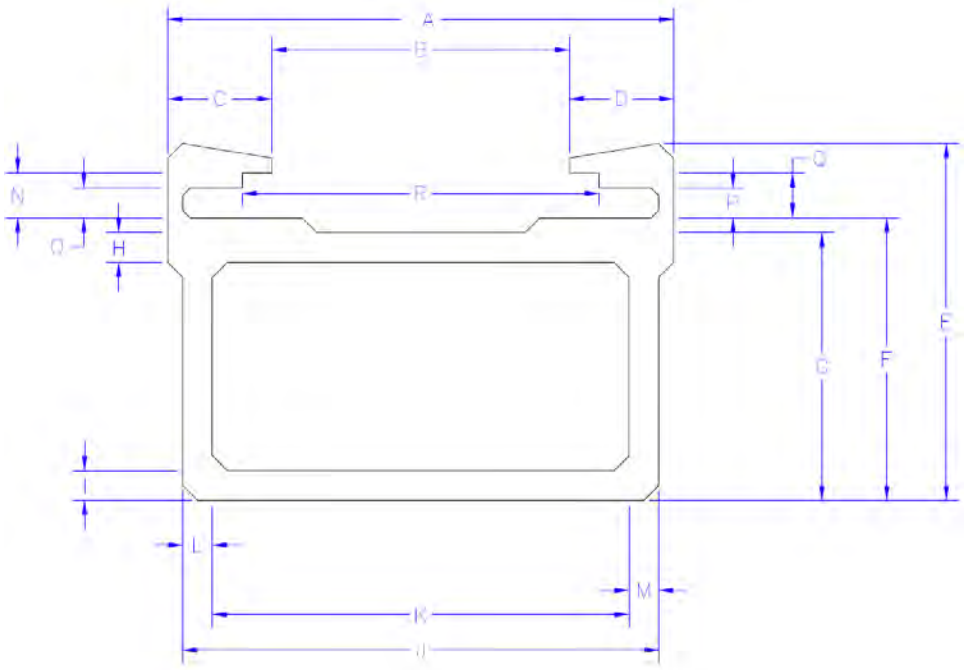

Figure 6.30: Dimensions of rail extrusions

6.5 Dynamic Simulation Model Development

To begin the process of modeling the motion of the missile while loaded into the rail, the components were first modeled in SolidWorks. Using Cornell's SolidLab toolkit, chosen dimensions of each component can be modified with a MATLAB script [48]. Using the dimensions from the tight and loose configurations, 2 CAD models were rapidly generated.

It is critical in the modeling phase case that a small, realistic roundover be applied to all edges of the solid models. Without doing this, the predicted contact areas from the eventual simulation will be impractically low, creating very high contact pressure predictions. In this case, all edges had a small roundover specified in the design, which simplified this process.

The SolidWorks model was created as an assembly of all of the subcomponents (Rail Extrusion, Missile, Mid-Shoe, Aft-Shoe, and Latch). The constraints themselves relating to the assembly (e.g. the shoes mated to the missile) are not carried into ADAMS, so they were only to establish the initial position of the components relative to each other.

Figure 6.31: SolidWorks rendering of rail model

Figure 6.32: SolidWorks rendering of front shoe model

Figure 6.33: SolidWorks rendering of system assembly

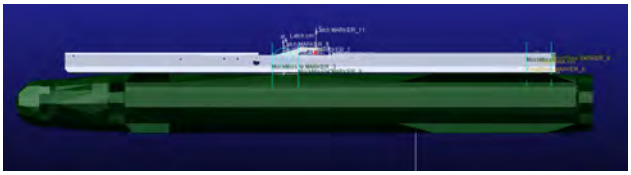

Figure 6.34: ADAMS rendering of system assembly

The assembly from SolidWorks was saved as a parasolid file, and then imported into ADAMS/View as such. Included in the import were the various sub-components, their material properties, mass properties, graphics, and positions. A screen capture of the model can be seen in Figure 6.34.

Because none of the mates from SolidWorks translate into ADAMS, the same relationships had to be re-created. A cylindrical connection was established between the latch and the rail extrusion, and the shoes were each given locked connections with the missile.

In order to simulate arbitrary vibrational inputs, a motion generator function was applied between the center of mass of the rail extrusion and the ground. The vibrational input to the system can then be modeled as a time dependent imposed motion on these connections.

A number of forces also had to be added into the model. The first four were the spring forces from the latch spring, rear leaf spring, and the 2 double barrel springs.

The mounting holes for the latch spring that were created in the SolidWorks model allowed for the simple addition of the spring forces as a 2 point coil spring.

The rear leaf spring was modeled as a point force, using Hooke's law for springs, and set to only act in the vertical direction. Upon examination, the experimental rail did not engage this spring when static, so to properly model it, a step function was used to model the saturation point of the spring. This allows the missile to slide axially without experiencing a spring force, but only the friction between the shoe and the rail, as is the case in the real system.

The double barrel springs were modeled as constant point forces because of the small displacements of those springs.

A number of soild-solid contacts were also defined:

- Mid Shoe-Rail Extrusion
- Mid Shoe-Latch
- Latch-Rail Extrusion
- Rear Shoe-Rail Extrusion
- Missile-Rail Extrusion

Each of these contacts had unique parameters relating to the contact model used in the simulation and to the material properties of the components and coatings themselves[13][19]. The parameters were hand-tuned such that the frequency response of the ADAMS model matched experimental frequency response data from the real system, as will be detailed later in the report[40].

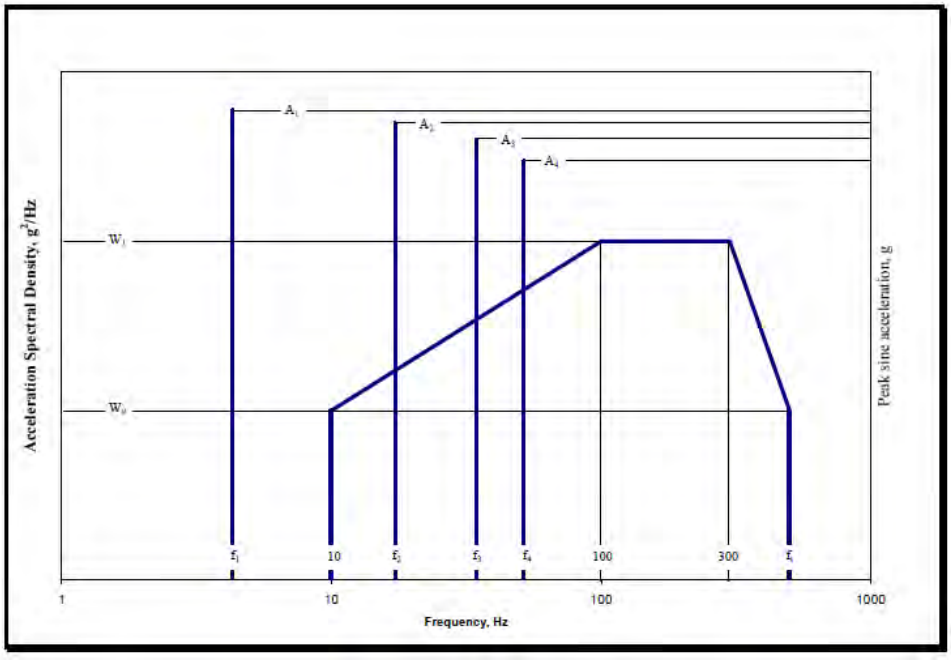

Figure 6.35: MIL-STD-810G Chart for Vibration of Helicopters[1]

Once the model was created, it's purpose was to simulate the conditions the rail system would experience on a certain vehicle. The vibration profile to input is from MIL-STD-810G, shown in Figures 6.35. In this standard, the representative vibration profiles for various vehicles can be found, but only one vehicle is examined in this report. More detail can be found in the standard itself: [1].

6.5.1 Friction Model

The friction model used for all contacts was the one developed in the first case study. The parameters are shown in Table 6.1.

Table 6.1: Estimated Friction Parameters

Parameter	Value
Static Coefficient	0.1799
Dynamic Coefficient	0.1660
Stiction Transition Velocity	6.4634 mm/s
Friction Transition Velocity	12.00 mm/s

6.5.2 Contact Parameters

Found iteratively, the contact parameters were selected to match the experimental data to simulation as closely as possible. To do this, the input data from the experimental rail accelerometers was splined, and imported to a motion controller in the ADAMS model. The bulk motion of the missile from the simulation was then compared to the experiment. The values for the contact are shown in Table 6.2. The default values are the parameters used in the rear and front shoe to rail contacts. The mid shoe's interactions with the latch and the latch's with the rail are special cases that use slightly modified parameters in order to match the longitudinally excited experimental data to the simulation.

The contact parameters used for all contacts in the system are shown in Table 6.2.

Table 6.2: Contact Parameters

Parameter	Value
Stiffness	1,000
Force Exponent	2.0
Damping	500
Penetration Depth	0.01

6.6 Dynamic Model Experimental Validation

In conjunction with AMRDEC, an inert missile and launcher system were instrumented with accelerometers and mounted to a shaker table. Each individual sine tone from the AH-64-Late profile out of MIL-810G was input to the rail (on each axis separately), and the data was gathered. The input data (rail accelerations) was splined in ADAMS and used to force the motion on the system. The output data (missile acceleration) was then exported and compared to the missile acceleration values from the experiment.

Figure 6.36 shows the instumented missile and launcher mounted to the shaker table at AMRDEC in the transverse excitation orientation. The wires to the various accelerometers mounted on the system can be seen in white.

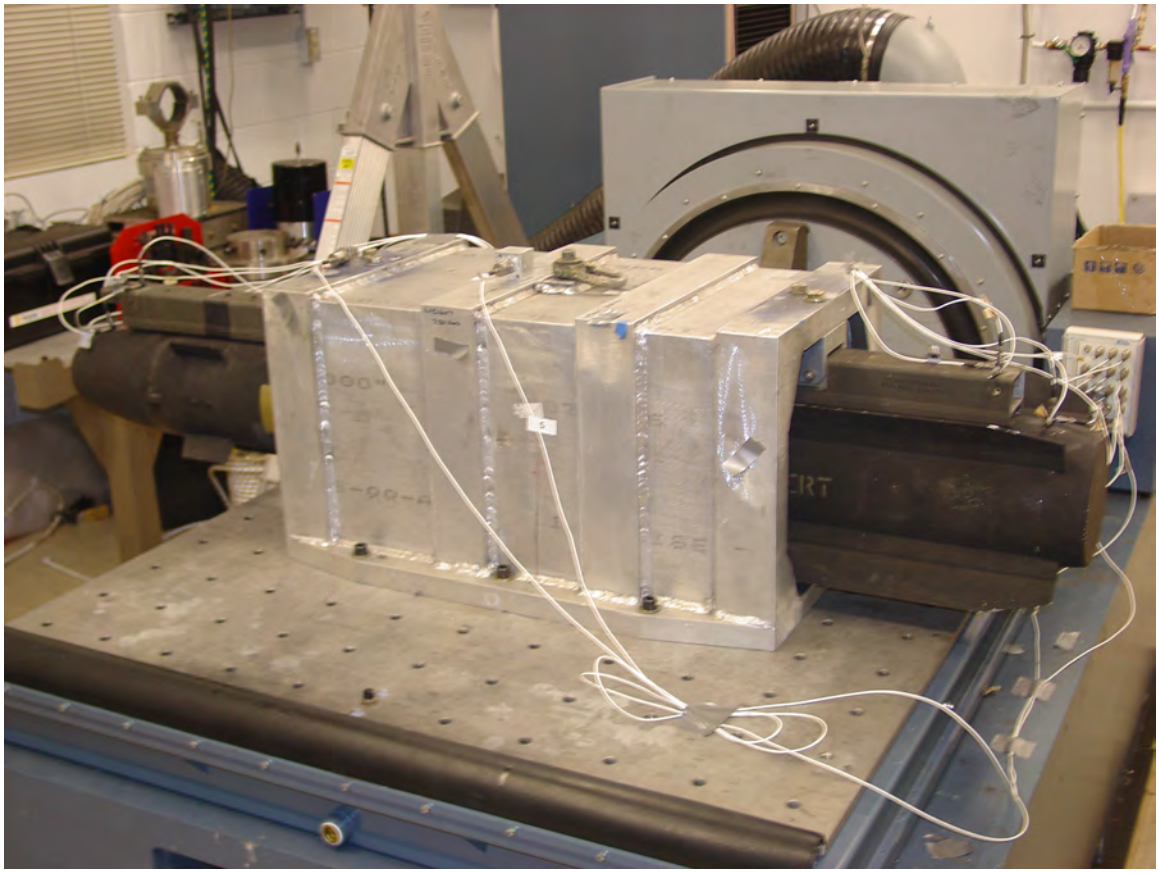

Figure 6.36: Instrumented missile and launcher mounted to shaker table at AMRDEC

Figure 6.37 shows the 2 control accelerometers which are mounted directly to the mounting fixture, and are used by the shaker table control software to control input vibration profiles. They are changed to be in the axis of excitation depending on configuration and should be roughly equivalent to the corresponding axis accelerometers on the rail. Figure 6.38 shows the detail of the accelerometers mounted on the missile nose and the front end of the rail. Note that the acclerometers are not perfectly in axis with each other due to the curvature of the nose.

Figure 6.39 shows detail of the rear end of the missile and of the rail, with accelerometers mounted, and Figure 6.40 shows another view of the front end of the system and the shaker fixture.

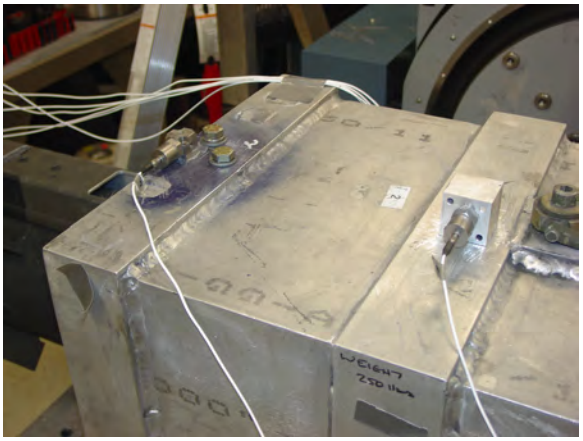

Figure 6.37: Control accelerometers on the shaker table fixture

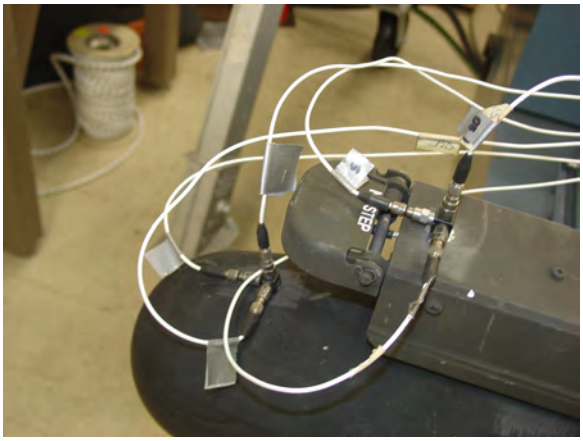

Figure 6.38: Front end of missile with accelerometers mounted

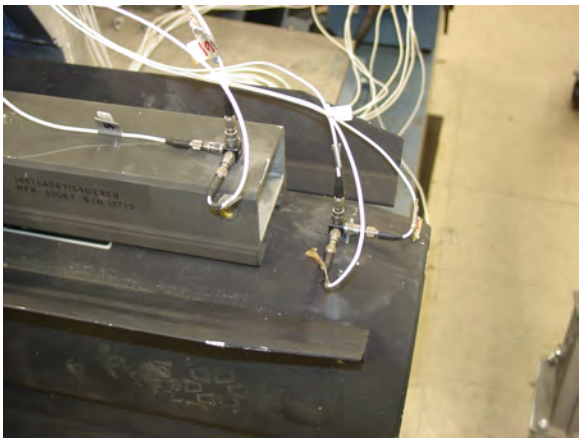

Figure 6.39: Rear end of rail with accelerometers mounted

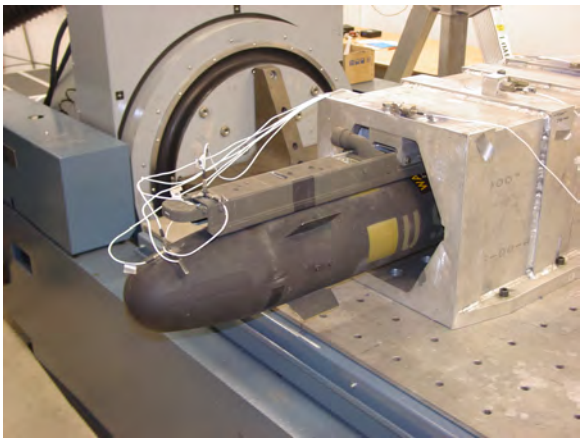

Figure 6.40: Front end of instrumented missile and launcher

Not all 12 test datasets were usable due to bad channels, the shaker not being able to maintain control and run the profile, or other issues. All of those profiles for which good data was gathered are detailed below in both the time and frequency domains.

6.6.1 Vertical Excitation

The profiles examined in this section are from the vertically excited tests. All accelerometer and simulation data is in axis with the excitation. This axis was run first, before the transverse and longitudinal. The vertical case has the most 'rattling' of the 3 axes, as could be clearly heard, seen, and can be found by examining the measured relative motion between

the rail and the missile. This fact was exasperated by the 'looseness' of the experimental rail, for which the rear spring didn't even touch the shoe when static. For many of the configurations, a base random profile was used to help the shaker table maintain control. This can be clearly seen in the frequency domain plots, which show significant content outside the desired sine tone.

There were only usable data from the first two sine tones, 4.88Hz and 19.44Hz.

4.88 Hz Sine Tone

The 4.88 Hz vertical profile matched up extremely well, likely because of little to no relative motion between the missile and rail (no rattling). For this same reason, the 4.88 Hz profiles for all axes matched up very well.

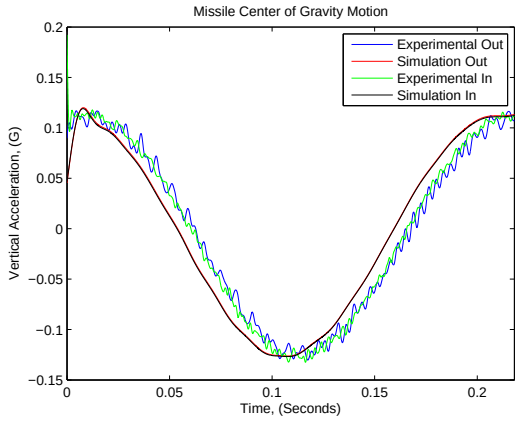

Figure 6.41: Vertical excitation, 4.88Hz, time domain comparison

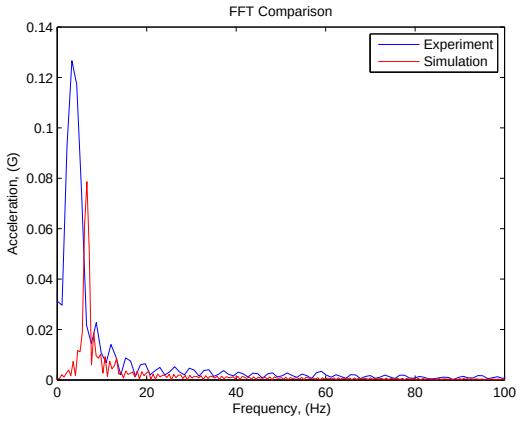

Figure 6.42: Vertical excitation, 4.88Hz, frequency domain comparison

19.44 Hz Sine Tone

The 19.44 Hz vertical sine tone was an intense enough profile to induce rattling. This can be clearly seen by the difference between the experimental input and experimental output in the time domain plot. Despite this the Fourier transforms of the experimental and simulation outputs match up quite well. The time domains do not appear to match up well, but it can be seen that the range of accelerations is quite close to the experimental values, which is

encouraging. This is due to uncertainty in the initial conditions for the simulation. All simulations were run by first simulating 2 seconds of no input, to allow the system to reach a static equilibrium. This is because with no known initial conditions, the best way to remain consistent between simulations was to start at static equilibrium.

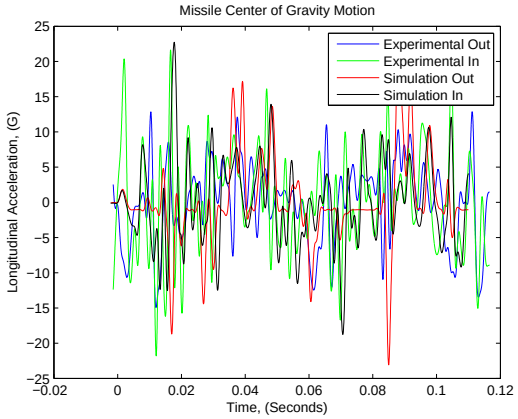

Figure 6.43: Vertical excitation, 19.44Hz, time domain comparison

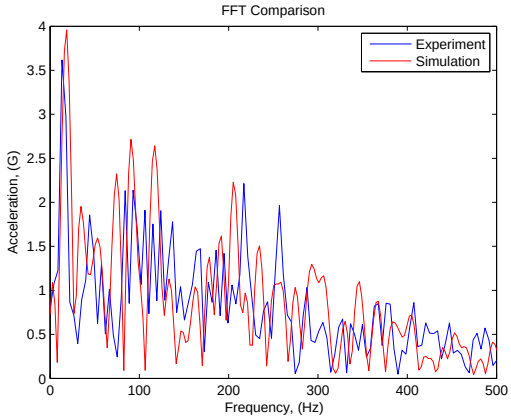

Figure 6.44: Vertical excitation, 19.44Hz, frequency domain comparison

This validation case is representative of the majority of the cases, where the frequency domain is the primary measure of success due to uncertainty in initial conditions.

6.6.2 Transverse Excitation

These profiles are from the transversely (side to side) excited tests. All accelerometer and simulation data is in axis with the excitation. Transverse excitation is dominated by friction as it is a mostly sliding event. Impacts occur as the shoes impact the sides of the rails, with the relative velocities of the impacts dictated largely by the friction of the contact. Usable data was gathered for all 4 sine tones for this axis.

4.88 Hz Sine Tone

The 4.88Hz sine tone for transverse excitation experiment, like the other axes, showed little to no relative motion between the missile and rail. This makes the simulation particularly straight forward, as is reflected in the results, which match the experimental for both

frequency domain and time domain quite well. There is one small spike in the frequency domain of the experiment at around 80Hz which is believed to be an artifact of filtering or data acquisition. This component shows up in the time domain as the high frequency oscillation on the experimental output.

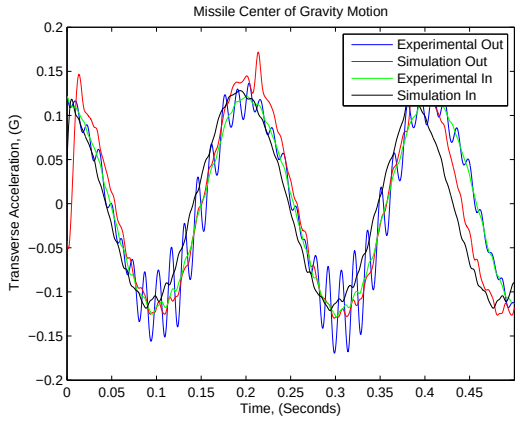

Figure 6.45: Transverse excitation, 4.88Hz, time domain comparison

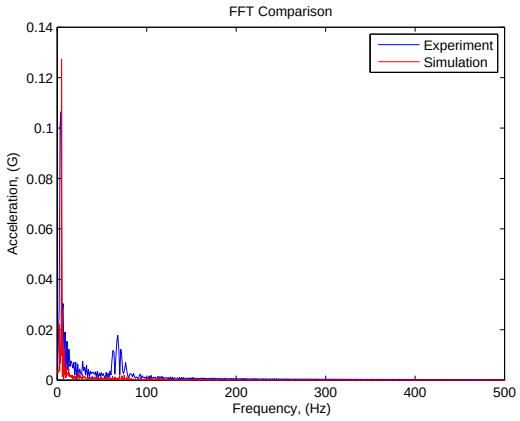

Figure 6.46: Transverse excitation, 4.88Hz, frequency domain comparison

19.44 Hz Sine Tone

At 19.44Hz, the missile began to move relative to the rail for transverse excitation. It is also the frequency at which the integrator began to behave strangely. In the time domain plot, there are very large, short, negative spikes in the simulated output acceleration. This type of artifact is most often attributed to integrator error related to extremely stiff systems of differential equations. It is desirable in this case to either make the set of differential equations being solved somehow less stiff, or to find an integrator which can better handle this hard-to-solve problem. Simply lowering the error bounds on the integrator does nothing to remove the spikes. Nevertheless, the bulk motion predicted appears to match the experiment relatively well, other than at low frequencies on the frequency domain plot.

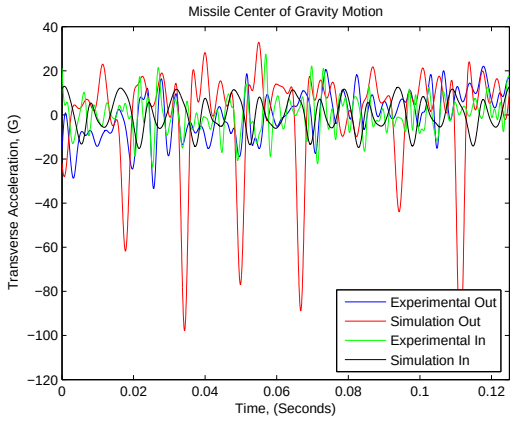

Figure 6.47: Transverse excitation, 19.44Hz, time domain comparison

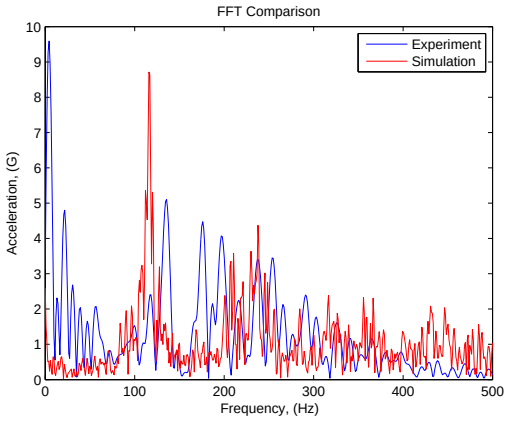

Figure 6.48: Transverse excitation, 19.44Hz, frequency domain comparison

38.88 Hz Sine Tone

Just as with the 19.44Hz case, the stiffness of the problem appears to be effecting the integrator, as can be seen in the spikes of the simulation output in the time domain plot. Otherwise, the time domain actually matches up well, despite starting at static equilibrium. Likewise the frequency domains appear to match each other well.

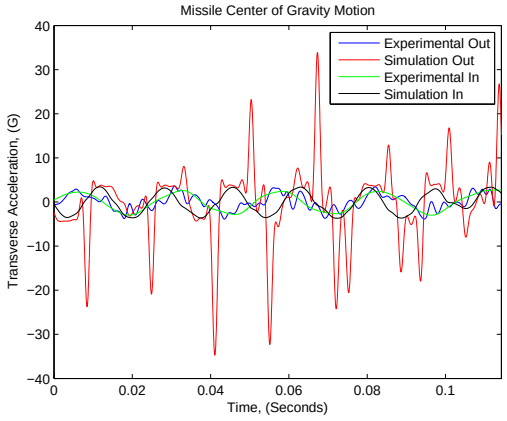

Figure 6.49: Transverse excitation, 38.88Hz, time domain comparison

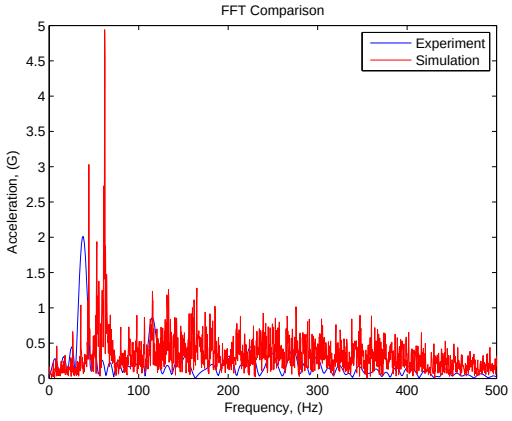

Figure 6.50: Transverse excitation, 38.88Hz, frequency domain comparison

58 Hz Sine Tone

Just as with the 38.88Hz tone and the 19.44Hz tone before it, the stiffness of the problem appears to be effecting the integrator, as can be seen in the spikes of the simulation output in the time domain plot. Otherwise, the time domain actually matches up well, despite starting at static equilibrium. Likewise the frequency domains appear to match each other well.

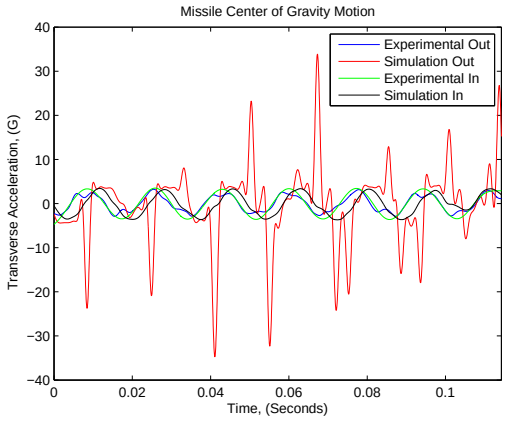

Figure 6.51: Transverse excitation, 58Hz, time domain comparison

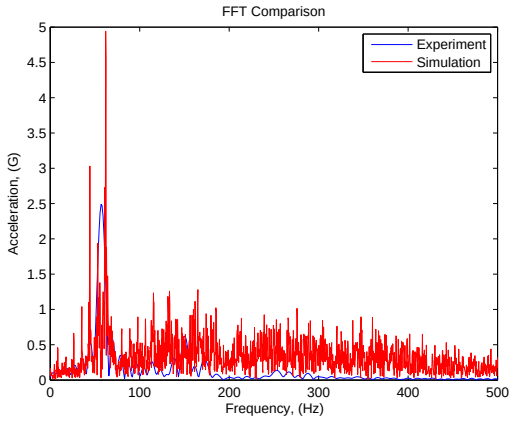

Figure 6.52: Transverse excitation, 58Hz, frequency domain comparison

6.6.3 Longitudinal Excitation

These profiles are from the longitudinally excited tests. All accelerometer and simulation data is in axis with the excitation. Longitudinal excitation is dominated by friction, because it is sliding, but differs from transverse by the inclusion of the compliance of the latch via the latch spring and an opposing force from the double barrel springs. In terms of developing contact parameters, this was the most difficult axis to tune. This was due to the interaction between the mid shoe and the latch, which proved very challenging because of the influence of the various springs, and the very small contact area. This allowed the mid shoe to "integrate through" the latch if the stiffness of the contact was not set high enough, allowing the missile

to fall out of the rail. As mentioned previously, this led to the increase of stiffness at this contact above that of the other contacts in the system.

4.88 Hz Sine Tone

The 4.88Hz sine tone for longitudinal excitation showed no significant sliding in the experimental data, which was easily replicated by the ADAMS model.

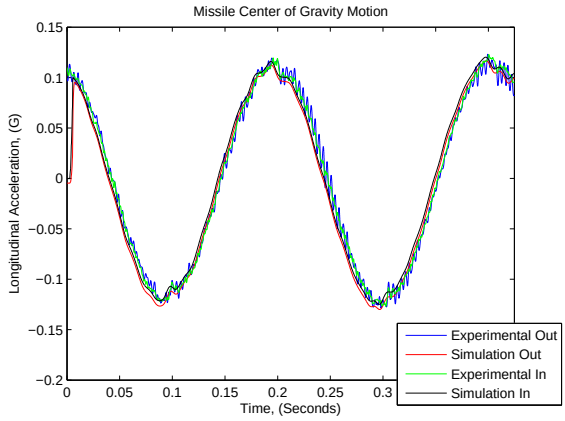

Figure 6.53: Longitudinal excitation, 4.88Hz, time domain comparison

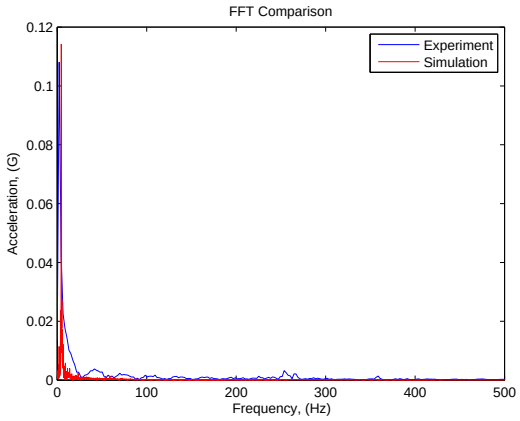

Figure 6.54: Longitudinal excitation, 4.88Hz, frequency domain comparison

19.44 Hz Sine Tone

Again, there are short spikes in the simulated output of the longitudinal 19.44Hz sine tone, which is most likely an integrator error. The frequency domain results match up quite well.

38.88 Hz Sine Tone

The final sine tone tested was the 38.88Hz sine tone in the longitudinal direction. Yet again, short spikes from the integrator skew the time domain data, while the frequency domain data matches up better.

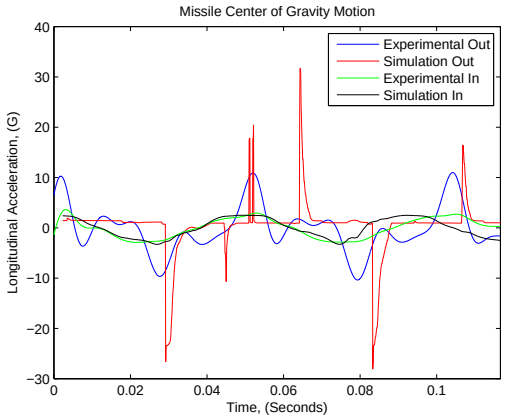

Figure 6.55: Longitudinal excitation, 19.44Hz, time domain comparison

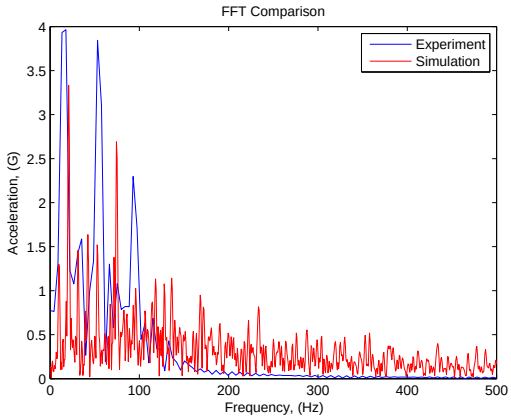

Figure 6.56: Longitudinal excitation, 19.44Hz, frequency domain comparison

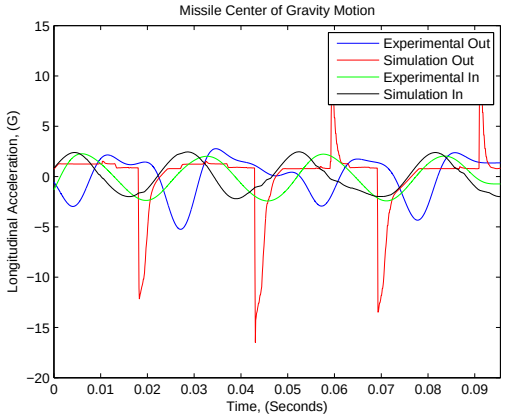

Figure 6.57: Longitudinal excitation, 38.88Hz, time domain comparison

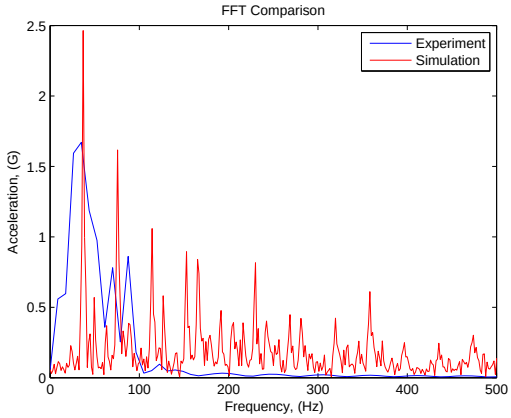

Figure 6.58: Longitudinal excitation, 38.88Hz, frequency domain comparison

6.7 Results

The first step in analysis was visualizing the location of these contacts, and therefore where exactly the wear on the rail is occurring. This is done simply be exporting the contact locations from the ADAMS Post-Processor and plotting these locations on top of the mesh of the rail CAD file (generated using CADExchanger). Figures 6.59, 6.60, 6.61, and 6.62 show the mesh of the rail (zoomed to the relevant section) with the locations of contact for contacts 1, 2, 4, and 5 respectively.

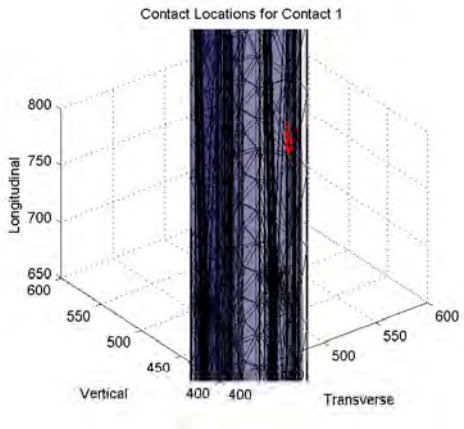

Figure 6.59: Locations on the rail contacted by the front shoe (missile)

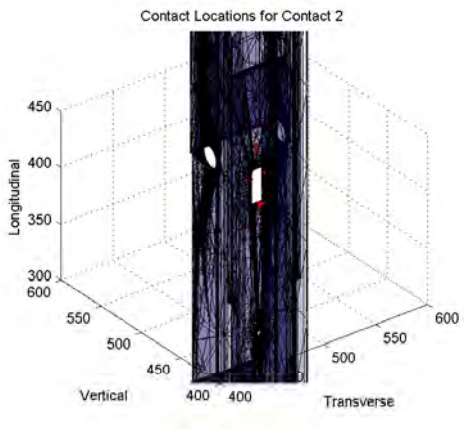

Figure 6.60: Locations on the rail contacted by the latch mechanism

The same simulation was used to generate the contact visualizations as to generate the previous plot showing predicted contact pressures.

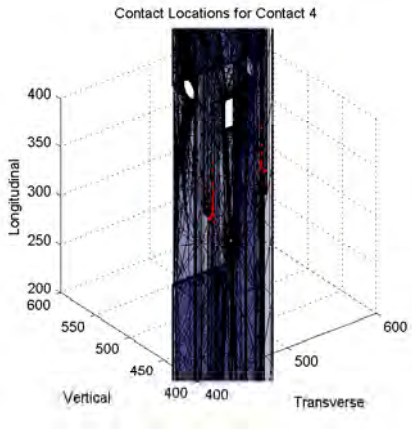

Figure 6.61: Locations on the rail contacted by the mid shoe

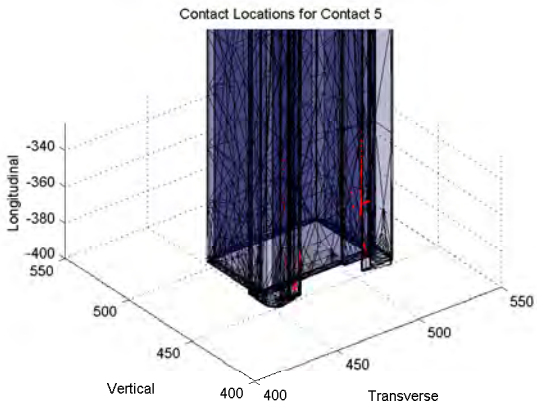

Figure 6.62: Locations on the rail contacted by the rear shoe

With the dynamic model validated, and the wear model developed in previous sections, it is now possible to simulate any input to the missile launcher system and predict the wear. Recall that multibody simulation and the wear model used do not account for any plastic deformation. The first step in the analysis of wear with multibody simulations is to examine the predicted contact surface stress to ensure that there are few cases which may indicate

plastic deformation. Plastic deformation does not generally occur until the surface pressure is above 1-6 times the yield strength.

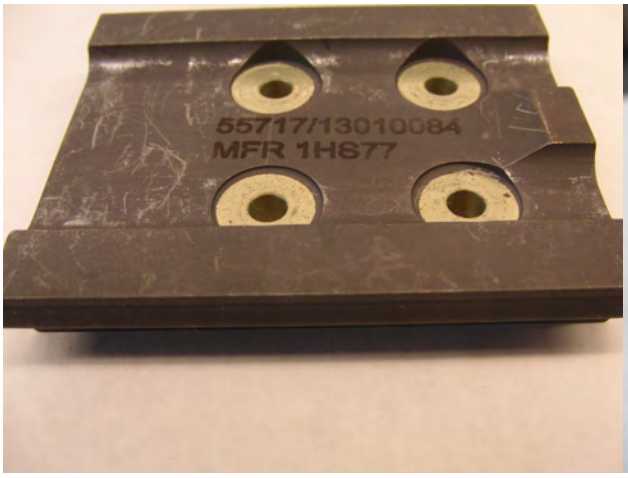

Figure 6.63: Before experiment: mid shoe new surface with no wear

Figure 6.64: After experiment: mid shoe deformation and wear from the latch face

In the experiment conducted to validate the dynamics of the missile launcher model, significant wear and deformation occurred in one particular case (19.44 Hz sine tone, transverse excitation for 12 hours). Figures 6.63 and 6.64 shows this. Figure 6.65 shows the probability density function of contact pressure at each contact in the system. This data was generated using the AH-64 Late profile on a model of the exact rail used in the experimental validation, excited in the transverse direction.

Immediately it can be seen that contact 4 (latch and mid shoe), shows a significant amount of pressures above the yield strength of the material, and significantly more than all other contacts. While few contact instances were above even 2 times the yeild strength of the material (224 Mpa), the significantly higher surface pressure estimates for the latch-shoe interface than the shoe-rail interfaces matches up with the significantly greater wear and deformation seen in the experiment.

With this initial check completed, a validation test for the wear predictions was conducted. This was done by running the system at one sine tone in the transverse direction on the shaker table at AMRDEC for 12 hours, and comparing the experimental mass loss

of the mid and rear shoes to the predicted wear value from the ADAMS simulation. The estimate for the mid and rear shoes was found to have errors of 6.50% and 3.35% respectively, validating the integrated wear model.

Again with the AH-64 Late profile in the transverse direction, the two boundary cases were simulated. Figure 6.66 shows the results of this simulation, with the mass per time wear rate for both configurations, as well as the midpoint between the two for contacts 1, 3 and 5. These 3 contacts correspond to the front, mid and rear shoes respectively with the rail. As expected, the looser configuration corresponds to greater wear on the rail.

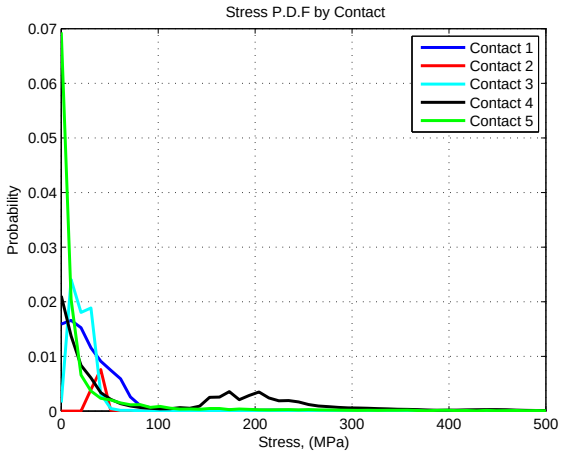

Figure 6.65: Probability density functions of contact pressure at each contact

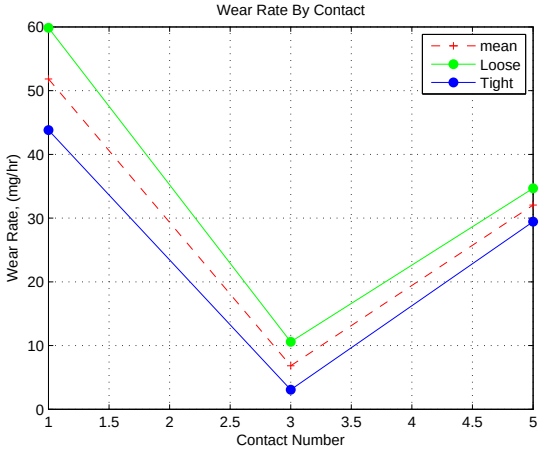

Figure 6.66: Wear rate of each rail contact (front, mid and rear shoe respectively)

Figure 6.67 shows the sum of the 3 rail contacts (1, 3 and 5) for each configuration (loose and tight), plotted relative to rattlespace. The looser configuration has the lower rattle space coefficient. Again, it can be seen that the looser configuration results in more rail wear than the designed intention (midpoint), and the tighter configuration wears less.

Using this data, a service life penalization methodology can be developed. Consider the intended service life, T_{SL}, which is presumed to be based off of the design dimensions (midpoint). By measuring the rattle space of a manufactured (i.e. imperfect) rail, the service life can be modified to better reflect the specific rail in question. Recall that the 'looser' configuration has the lower rattle space value (0.77, and the tighter has the higher value

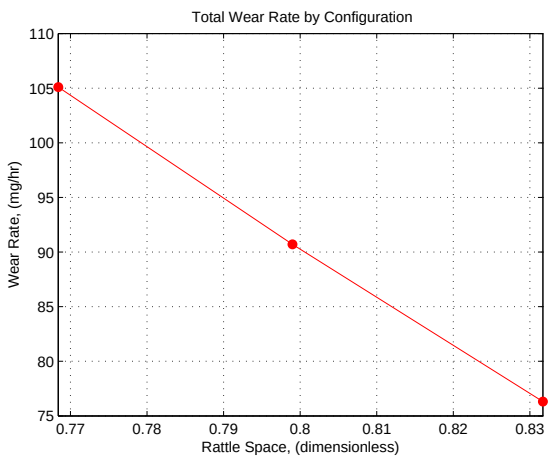

Figure 6.67: Total wear rate for rail at loosest, tightest and midpoint configurations

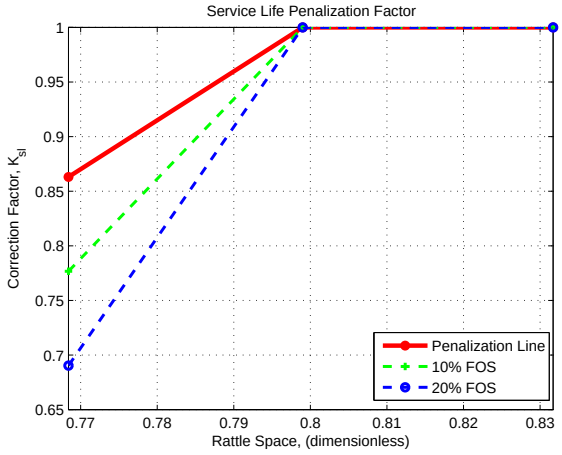

Figure 6.68: Service life penalization for dimensionality effects due to manufacturing tolerances

(0.83). Because the looser configuration wears more quickly, it's service life should likewise be shortened. The equation to account for this is:

$$T_{SL,real} = K_{SL} T_{SL,design} \tag{6.4}$$

Where K_{SL} is the service life penalization factor, calculated from the predicted wear rates. A conservative penalization factor should never increase the predicted service life, but only shorten it. Figure 6.68 shows the service life penalization factor plot for the data in Figure 6.67. As can be seen, the factor is never greater than 1, and as the system becomes looser than designed, the factor decreases, decreasing the real service life ($T_{SL,real}$).

Also plotted in Figure 6.68 are two lines showing values 10% and 20% less than the calculated penalization line respectively. These can be used to generate more conservative $T_{SL,real}$ estimates when the model is uncertain.

6.8 Conclusion

In conclusion, by leveraging multi-body simulation, wear mass for missile launcher systems can be estimated for any arbitrary vibration profile and rail dimension set. Further,

with simple laboratory experimentation, the wear coefficients for the rail and shoe materials can be found. With these capabilities, amendments can be made to the service life and maintenance routines of the systems according to their manufactured dimensions, target vehicle, and deployed environment.

In conclusion, by leveraging multi-body simulation, wear power for missile launcher systems can be estimated for any arbitrary vibration profile and rail dimension set. Further, with simple laboratory experimentation, the wear coefficients for the rail and shoe materials in the presence of any sand can be found. With these capabilities, amendments can be made to the service life and maintained routines of the systems according to their manufactured dimensions, target vehicle, and deployed environment. With more quantitative service life and maintenance timeline estimates, the safety of the warfighters, the efficacy of the weapons and the value to the taxpayers can all be maximized.

Chapter 7

Conclusion

In this thesis, the strengths, benefits and differentiating factors of multibody simulation software (MBS) has been examined. The strengths and weaknesses of the various integrator-formulators available were presented, the manner in which the software handles contact forces was outlined, and the methodology and code for creating a user written subroutine to track contact area were given.

With the process of modeling a system in MBS software understood, the literature on simulating wear was investigated. Having selected multibody simulation as the platform to simulate wear in the given problems, a wear model needed to be developed. Chapter 3 detailed the prominent existing models for wear and proposed a modification to them, which was used throughout the thesis.

Chapters 4, 5, and 6 detailed 3 respective case studies which developed and validated a methodology for wear prognostics in multibody rattling systems. Chapter 4 pertained to the laboratory experiments which were used to develop the friction and wear models used in the remainder of the thesis.

Chapter 5 presented a simple experimental fixture which validated both the MBS model and the wear model. This simple experiment is easily extendible to other materials and contaminants, and therefore can be easily used to develop and validate other wear models in the future.

Chapter 6 used the validated wear model and a MBS model to simulate the wear of a Hellfire missile launcher system while in captive carry. With the ability to quickly simulate wear for arbitrary systems under arbitrary conditions, the differences in wear rate between different launchers was able to be assessed. With this information a methodology known

as "service life penalization" was presented. Using this methodology, existing imperfect rails can be easily assessed based on geometry and target vehicle. Being different than the perfectly manufactured and applied rail, these real world rails have different service lives. The "service life penalization" method uses results from MBS to amend the stated service life of the imperfect rails, reducing the risk of failure due to faster than expected wear.

In total, this thesis developed a methodology for taking an existing product, evaluating the wear rates of critical components under ideal conditions, and then accounting for expected deviations from those ideal conditions. The flowchart shown in Figure 7.1 summarizes this methodology.

The process of wear prognostics is separated into two primary phases, model development and model application. In this thesis, the model was developed using primarily the first two case studies. The final ADAMS model was created in Chapter 6, after which the model application phase was detailed.

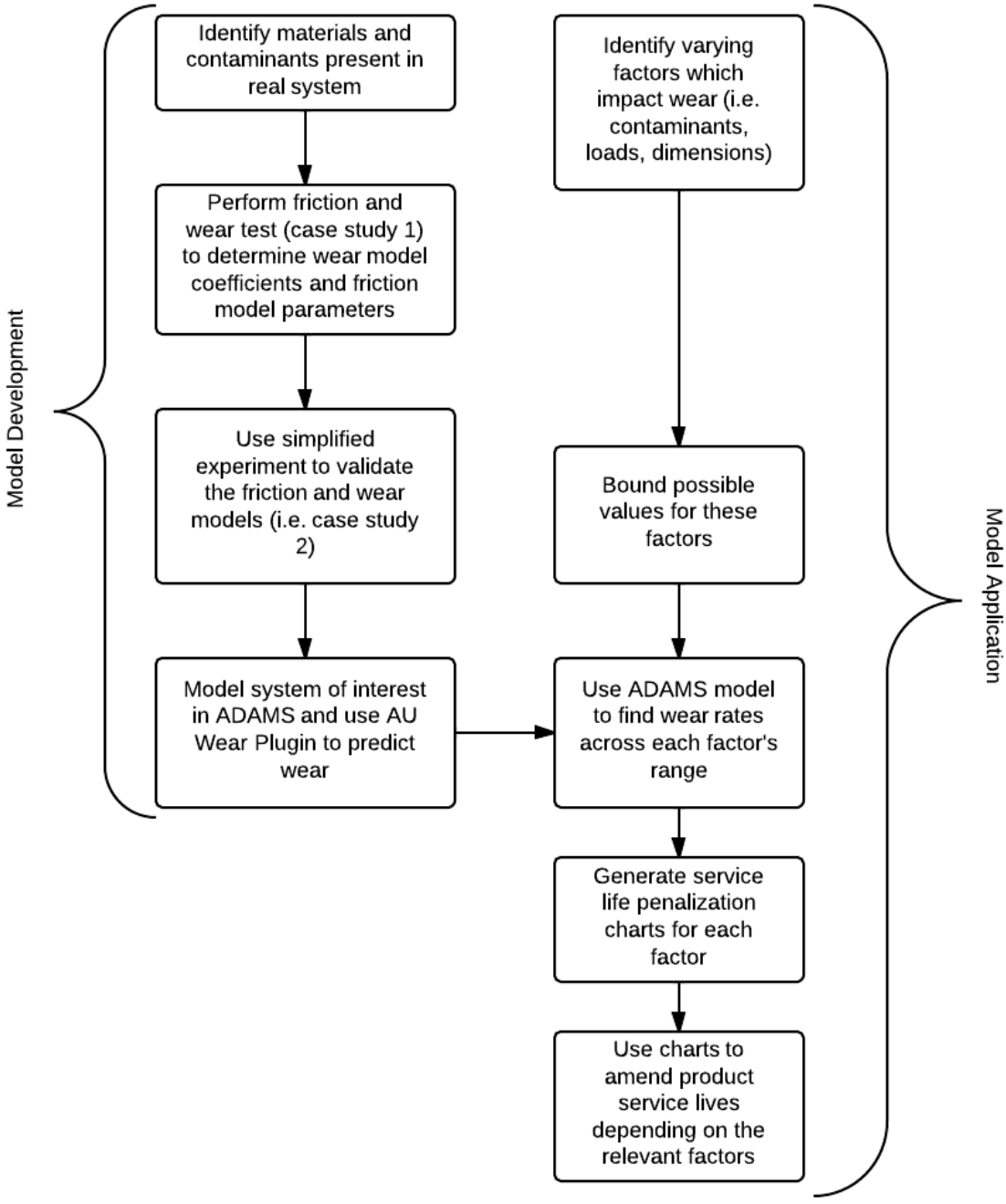

Figure 7.1: Flowchart summarizing the methodology of wear prognostics

Bibliography

[1] Department of defense test method standard: Environmental engineering considerations and laboratory tests, October 2008.

[2] Accurate simulating entire systems in motion helps deliver a better product in less time. Technical report, MSC.Software, 2010.

[3] Adams: The multibody dynamics simulation software. http://www.mscsoftware.com/product/adams, March 2014.

[4] Aluminum 7075-t73; 7075-t735x. http://www.matweb.com/search/DataSheet.aspx?MatGUID=6653b72914864cc0a0ff7adf5b720167&ckck=1, April 2014.

[5] Lms dads. http://www.lmsintl.com/DADS, March 2014.

[6] Mechanical simulation. https://www.carsim.com/, April 2014.

[7] Project chrono. http://www.chronoengine.info/chronoengine/, March 2014.

[8] Working model 2d: 2d dynamic motion simulation. http://www.design-simulation.com/WM2D/Index.php, March 2014.

[9] E. J. Abbott and F. A. Firestone. Specifying surface quality- a method based on accurate measurement and comparison. Mechanical Enginering, 55:569–572, 1933.

[10] James Ziemer ad William Herguth, Robert Bruce, Douglas Godfrey, Ray Ryason, E.R. Booser, Andrew Flaherty, Simon Tung, and Philip Guichelaar. Basics of wear. http://www.stle.org/resources/lubelearn/wear/, 2008.

[11] J.F. Archard. Contact and rubbing of flat surfaces. Journal of Applied Physics, 24:981–988, 1953.

[12] Mike Blundell. The Multibody Systems Approach to Vehicle Dynamics. Elseiver Butterwork-Heinemann, Oxford, MA, 2004.

[13] Walter Daniel. Adams methodology: Contact modeling. In Benelux ADAMS User Meeting, 2012.

[14] Renguang Dong. Approaches to incorperate large data series into adams models.

[15] Salah Faik and Holly Witteman. Modeling of impact dynamics: A literature survey. In 2000 International ADAMS User Conference, 2000.

[16] Erwin Fehlberg. Low-order classical runge-kutta formulas with step size control and their application to some heat transfer problems. Technical report, NASA Technical Report 315, 1969.

[17] P. Flores. Modeling and simulation of wear in revolute clearance joints in multibody systems. Mechanism and Machine Theory, 44:1211–1222, June 2009.

[18] A.W.J. De Gee, C.P.L. Commissaris, and J.H. Zaat. The wear of intered aluminum powder (sap) under conditions of vibrational contact. Wear, 7:535–550, 1964.

[19] Jochem Giesbers. Contact mechanics in msc adams. Master's thesis, University of Twente, 2012.

[20] G. Gilardi and I. Sharf. Literature survey of contact dynamics modeling. Mechanism and Machine Theory, 37:1213–1239, March 2002.

[21] H. R. Hertz. On the contact between elastic bodies. Gesammelte Werke, 1:1, 1895.

[22] H. R. Hertz. On the contact of rigic elastic solids. Miscellaneous Papers, 92:156, 1896.

[23] R. L. Jackson and I. Green. A finite element study of elasto-plastic hmeispherical contact. Proceedings of hte 2003 ASME/STLE International Tribology Conference, 2003.

[24] R. L. Jackson and I. Green. A statistical model of elastio-plastic asperity contact of rough surfaces. In Proceedings of 2003 STLE/ASME Joint International Tribology Conference, 2003.

[25] Robert L. Jackson and Itzhak Green. A finite element study of elasto-plastic hemispherical contact against a rigid flat. Journal of Tribology, 127:343–354, 2005.

[26] K. L. Johnson. Contact Mechanics. Cambridge University Press, 2003.

[27] L. Kogut and I. Etsion. Elastic-plastic contact analysis of a sphere and a rigid flat. ASME Journal of Applied Mechanics, 69:657–662, 2002.

[28] Captain Adam W. Lange. Hellfire: Getting the most from a lethal missile system. Armor, January-Febuary:25–30, 1998.

[29] H.M. Lankarani and P.E. Nikravesh. Continuous contact force models for impact analysis in multibody systems. Nonlinear Dynamics, 5:193–207, 1994.

[30] David Lindberg. Msc adams modelling of mechanical system in a400m crew entrance door. Master's thesis, Linkoping Institute of Technology, 2012.

[31] H.C. Meng and K.C. Ludema. Wear models and predictive equations: their form and content. Wear, 181:443–457, 1994.

[32] Forest R. Moulton. New methods in exterior balistics. University of Chicago Presss, 1926.

[33] MSC Software. About Adams/Solver.

[34] MSC Software. ADAMS 2008 r1 Release Guide, 2008.

[35] Saad M. Mukras, Nathan A. Mauntler, Nam Ho Kim, Tony L. Schmitz, and W. Gregory Sawyer. Evaluation of contact force and elastic foundation models for wear analysis of multibody systems. In Proceedings of the 34th Annual Mechanism and Robotics Conference, 2010.

[36] R. Al Nazar, T. Rantalainen, A. Heinonen, H. Sievanend, and A. Mikkola. Flexible multibody simulation approach in the anaylsis of tibial strain during walking. Journal of Biomechanics, 41:1036–1043, 2008.

[37] Adina Negrea and Mihai Valentin Predoi. The elastic contact of a sphere with an elastic half-space, a comparison between analytical and finite element solutions. U.P.B. Science Bulletin, 74:1, 2012.

[38] D. Negrut, G. Ottarsson, A. Sajdak, and R. Rampalli. On the use of the hht method in the context of index 3 differential algebraic equations of multibody dynamics. International Journal for Numerical Methods in Engineering, 00:1–6, 2000.

[39] N. M. Newmark. A method of computation for structural dynamics. Journal of Engineering Mechanics, 85:67–94, 1959.

[40] Koichi Ohtomi. Msc.adams applications to space equipment. In European MSC.ADAMS User Conference Proceedings, 2002.

[41] K. O'Riordain, P.M. Thomas, J.P. Phillips, and M.D. Gilchrist. Reconstruction of real world head injury accidents resulting from falls using multibody dynamics. Clinical Biomechanics, 18:590–600, 2003.

[42] Ernest Rabinowicz. Friction and Wear of Materials. John Wiley and Sons, Inc., 1995.

[43] Rajiv Rampalli. Recent advances in adams/solver. In MSC ADAMS Users Conference, 2001.

[44] Andrei Schaffer. Recent advances in numerical integration mmethod for complex multibody systems. In MSC Software Vehicle Dynamics Symposium, 2008.

[45] M.C. Shaw. Dimensionless analysis for wear systems. Wear, 43:263–266, 1977.

[46] G.W. Stachowiak and A.W. Batchelor. Enigneering Tribology. Elsevier Butterworth-Heinemann, 2005.

[47] Noise Policy Station. Assessing vibration: a technical guideline. Technical report, Department of Environment and Conservation NSW, 2006.

[48] Krishnan Suresh. SolidLab: A MATLAB interface to SolidWorks. University of Wisconson-Madison, Madison, WI 53706, 1.00 edition.

[49] Dong Zhao, Hideyuki Sakoda, W. Gregory Sawyer, Scott A. Banks, and Benjamin J. Fregly. Predicting knee replacement damage in a simulator machine using a computational model with a consistent wear factor. Journal of Biomechanical Engineering, 1:1, 2006.

Appendix A

.1 Contact Area Subroutine Source Code

The following is the FORTRAN source code for contactsub.f, the code used in Chapter 2 to track contact area.

```
      SUBROUTINE CNFSUB(ID, TIME, PAR, NPAR, LOCI, NI, LOCJ, NJ,
     & GAP, GAPDOT, GAPDOTDOT, AREA, DFLAG, IFLAG, FORCE)
C
C     ==== Type and dimension statements ==================
      IMPLICIT NONE
      INTEGER ID
      DOUBLE PRECISION TIME
      DOUBLE PRECISION PAR( * )
      INTEGER NPAR
      DOUBLE PRECISION LOCI(3)
      DOUBLE PRECISION NI(3)
      DOUBLE PRECISION LOCJ(3)
      DOUBLE PRECISION NJ(3)
      DOUBLE PRECISION GAP
      DOUBLE PRECISION GAPDOT
      DOUBLE PRECISION GAPDOTDOT
      DOUBLE PRECISION AREA
      LOGICAL DFLAG
      LOGICAL IFLAG
      DOUBLE PRECISION FORCE(3)
C
C     Input parameters
```

```
23 C
   C     ID         Identifier of calling CONTACT statement
25 C     TIME       Current time
   C     PAR        Array containing passed parameters
27 C        PAR(1) - contact stiffness coefficient
   C        PAR(2) - contact force exponent
29 C        PAR(3) - contact damping coefficient
   C        PAR(4) - contact penetration at which full damping is applied
31 C     NPAR       Number of passed parameters
   C     LOCI     contact point location on I in I coordinates
33 C     NI       contact normal on I in I coordinates
   C     LOCI     contact point location on J in J coordinates
35 C     NJ       contact normal on J in J coordinates
   C     GAP        contact penetration
37 C     GAPDOT     first time derivative of GAP
   C     GAPDOTDOT second time derivative of GAP
39 C     AREA     area of contact
   C
41 C     components returned to ADAMS
   C
43 C     FORCE Array (dimension 3) of computed CNFORC
   C
45 C     Local variable and parameter definitions
   C
47       DOUBLE PRECISION K
         DOUBLE PRECISION C
49       DOUBLE PRECISION E
         DOUBLE PRECISION D
51    LOGICAL ERRFLG
   C
53 C     ===Executable code ================================
   C
55       K = PAR(1)
```

```
      E = PAR(2)
57    C = PAR(3)
      D = PAR(4)
59
      CALL IMPACT(GAP, GAPDOT, 0.0D0, K, E, C, D, 0, FORCE, ERRFLG)
61    CALL ERRMES(ERRFLG,'ERROR CALLING IMPACT',ID,'STOP')

63    RETURN
      END
65




67


69    SUBROUTINE CFFSUB(ID, TIME, PAR, NPAR, LOCI, LOCJ, X, XDOT,
     & NFORCE, AREA, DFLAG, IFLAG, FORCE)
71 C
   C     ==== Type and dimension statements ==================
73    IMPLICIT NONE
      INTEGER ID
75    DOUBLE PRECISION TIME
      DOUBLE PRECISION PAR(*)
77    INTEGER NPAR
      DOUBLE PRECISION LOCI(3)
79    DOUBLE PRECISION LOCJ(3)
      DOUBLE PRECISION X(3)
81    DOUBLE PRECISION XDOT(3)
      DOUBLE PRECISION NFORCE
83    DOUBLE PRECISION AREA
      LOGICAL DFLAG
85    LOGICAL IFLAG
      DOUBLE PRECISION FORCE(3)
87 C
   C     Input parameters
```

```
89  C
    C     ID       Identifier of calling CONTACT statement
91  C     TIME     Current time
    C     PAR      Array containing passed parameters
93  C        PAR(1) - stiction coefficient
    C        PAR(2) - friction coefficient
95  C        PAR(3) - stiction velocity
    C        PAR(4) - friction velocity
97  C     NPAR     Number of passed parameters
    C     LOCI     contact point location on I in I coordinates
99  C     LOCI     contact point location on J in J coordinates
    C     X        sliding displacement since the beginning of contact
101 C        X(1) - translational deformation in x
    C        X(2) - translational deformation in y
103 C        X(3) - rotational deformation about z
    C     XDOT     slip velocity of contact point
105 C        XDOT(1) - slip velocity in x
    C        XDOT(2) - slip velocity in y
107 C        XDOT(3) - relative angular velocity about z
    C     NFORCE   contact normal force
109 C     AREA     area of contact
    C
111 C     components returned to ADAMS
    C
113 C     FORCE Array (dimension 3) of computed CNFORC
    C        FORCE(1) - force in x direction
115 C        FORCE(2) - force in y direction
    C        FORCE(3) - torque about z axis
117 C     Local variable and parameter definitions
    C
119       DOUBLE PRECISION CAREA
          DOUBLE PRECISION US
121       DOUBLE PRECISION UD
```

```
      DOUBLE PRECISION VS
      DOUBLE PRECISION VD
      DOUBLE PRECISION H0, H1, X0, X1, TEMP1, TEMP2
  LOGICAL ERRFLG
C
      COMMON / COMVAR / CAREA
C     ===Executable code ================================
C
      CAREA = AREA

      US = PAR(1)
      UD = PAR(2)
      VS = PAR(3)
      VD = PAR(4)

      X0 = -VS
  H0 =  -1
  X1 =  VS
  H1 =   1

  CALL STEP(XDOT(1), X0, H0, X1, H1, 0, TEMP1, ERRFLG)
      CALL ERRMES(ERRFLG,'ERROR CALLING STEP',ID,'STOP')

      X0 = VS
  H0 = US
  X1 = VD
  H1 = UD

  CALL STEP(XDOT(1), X0, H0, X1, H1, 0, TEMP2, ERRFLG)
      CALL ERRMES(ERRFLG,'ERROR CALLING STEP',ID,'STOP')

C     Friction force is function of the contact area
  FORCE(1) = -AREA*TEMP1*TEMP2
```

```
155     FORCE(2) = 0.0
        FORCE(3) = 0.0
157
              RETURN
159           END

161           SUBROUTINE VARSUB ( ID, TIME, PAR, NPAR, DFLAG,
            &                     IFLAG, VALUE)
163   C
      C ==== Type and dimension statements ==================
165   C
      C Note: For machines with 60 or more bits per word,
167   C         substitute "REAL" for "DOUBLE PRECISION".
      C
169   C --- External variable definitions ------------------
      C
171           INTEGER                       ID
              DOUBLE PRECISION              TIME
173           DOUBLE PRECISION              PAR( * )
              INTEGER                       NPAR
175           LOGICAL                       DFLAG
              LOGICAL                       IFLAG
177           DOUBLE PRECISION              VALUE
      C
179   C ID            Identifier of calling VARIABLE statement
      C TIME          Current time
181   C PAR           Array of passed statement parameters
      C NPAR          Number of passed parameters
183   C DFLAG         Differencing flag
      C IFLAG         Initialization pass flag
185   C VALUE         The VARIABLE value returned to ADAMS
      C
187   C --- Local variables -------------------------------
```

```
      C
189   C       CAREA     area of contact
              DOUBLE PRECISION CAREA
191           COMMON / COMVAR / CAREA
      C
193   C ==== Executable code ================================
      C
195
              VALUE = CAREA
197   C
              RETURN
199           END
```

Appendix B

The following are the two peices of code required for the AU Wear Prediction plugin described in Chapter 4. The first script is CParse.C, the C language code which executes the wear prediction, the second is WearPrediction2.cmd, the ADAMS command file which creates the plugin dialog itself in ADAMS.

.2 CParse.C

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdlib.h>
/* General goal of the script:
  take in a path, a contact name, the number of "tracks" stored, and 3
      constants
  -at that path location there should be a file for each track, which has a
      bunch of data
  -all tracks should be the exact same length, and have the same time vector (
      column one)
  -need to calculate the wear volume from each track, add them all together
      into one "total
    wear volume" file with 2 columns (time, wear). And write that to file.
    */


  // COMMAND LINE ENTRIES
  //argv[1] is a basepath
```

```
16    //argv[2] is the contact name
      //argv[3] is the number of tracks
18    //argv[4] is c1
      //argv[5] is c2
20    //argv[6] is c3


22



24  char ch,ch2,ch3,file_name[100],file_name2[100], file_name3[100]; //values for
        EOL checking and the 3 file name buffers
    int k, row_iter; //loop variables
26  float wearvol=0; // wear volume of current track
    float time_vec=0; // time value of current track
28  float N, t, v, Kw, tin, wearin; //values used in various parts of wear
        calculation
    float tprev=0; //time from the previous timestep, used for integration
30  float running_sum=0; //running sum of wear (output is cumulative wear not
        instantaneous)
    char line1[100], line2[100], line3[100], line4[100], line5[100];
32
    float main( int argc, char *argv[])
34  {
       if (argc != 7)
36     {
        printf("6 Arguments Required for this model");
38      return 1;
      }
40     else
       {
42        /* basically this loop goes through each track, calculates wear from it
              , and writes that to f3.  Then f3 is copied to f2.
          when the loop repeats, the data from the new track is added to what is
              read from f2, and the sum is written to f3.
```

```
44        Again, f3 is coppied to f2.  This is repeated until f2 and f3 eventually
              both hold the solution. Then all files other than
          f2 can be deleted.
46        */
          sprintf(file_name2,"%s%sWEARDATA.dat",argv[1],argv[2]);
48        //acutally make the file, shouldn't already exist.
          FILE *ftemp1 = fopen(file_name2, "w");
50        fclose(ftemp1);
          sprintf(file_name3,"%s%sWEARDATAtemp.dat",argv[1],argv[2]);
52        //printf("FILE2: %s\n", file_name2);
          //printf("FILE3: %s\n", file_name3);
54        // Loop itself, k iterates from 1 to NUMTRACKS, which is user-given.
           for (k=1; k<=atoi(argv[3]); k++){
56       // for debugging: for (k=1; k<=1; k++){
              /* First thing is to build the various file names used in this loop:
58            file_name: is the file name of the track file for this iteration
              file_name2: is the file name of f2
60            file_name3: is the filename of f3
          */
62            sprintf(file_name,"%sTrack%d",argv[1],k);
          //Other names are declared outside of loop
64        //printf("\n\nFILENAME ENTERED:\n\t%s\n\n",file_name);


66

          // F2 is opened for reading, nothing is written to it at first.
68        FILE *f2 = fopen(file_name2, "rt");
          if (f2 == NULL)
70        {
            printf("Error opening file 2 to read from!\n");
72          exit(1);
          }
74
```

```
76        // F3 is opened for writting, with all existing data overwritten
          FILE *f3 = fopen(file_name3, "w");
78        if (f3 == NULL)
          {
80          printf("Error opening file 3 to write to!\n");
            exit(1);
82        }


84

           //Fp is opened for reading, this is the track data
86         FILE *fp;
           fp = fopen(file_name,"rt"); // read mode
88         if( fp == NULL )
           {
90          perror("Error while opening the file.\n");
            exit(EXIT_FAILURE);
92         }


94

           /* This inner while loop iterates through the track file itself. At
              each row of the file,
96          it reads the data from the track, the data from F2 (if k>1), and
               calculates a total cumulative
            wear value, which is written to F3. To start with tprev, row_iter,
               and running_sum are
98          re-intialized to 0.
          */
100        tprev=0.0;
           row_iter=0.0;
102        running_sum=0.0;
           Kw=0.0;
104        wearvol=0.0;
```

```
106         printf("%10.9f\n", running_sum);
            while(fgets(line1,100,fp)!=NULL)
108         {
              /* the first 8 rows are garbage, so skip ahead past that, with the
                  row_iter check.
110             The k check just ensures that for the first track, nothing is read
                   from F2, which is empty at
                this point in the process.
112           */
              if (row_iter>=9){
114             //store values of time, force and velocity
                sscanf(line1, "%f %f %f\n",&t, &N, &v);
116


118             //calculate the appropriate wear coefficient for the instantaneous
                    normal load
                // be sure to check for a div by 0 error
120             if ((atof(argv[5])) != 0.0){
                  Kw=(atof(argv[4]))*pow(fabs(N),(1.0/(atof(argv[5]))))+(pow
                      (10.0,-3)*atof(argv[6]));
122               //printf("%10.5f\t", Kw);
                }
124             else {
                  Kw=0.0;
                  printf("C_2 set to 0.0\n");
126             }
128
                //calc differential wear for the past timestep
130             //not certain on orders of magnitude here or in the matlab script,
                    all needs validation.
                wearvol=(pow(10.0,-3))*(Kw*N*v)*(t-tprev); //units micrometers^3
132

                //printf("%10.9f\t", wearvol);
```

```
134
            //add the differential wear to the cumsum
136         running_sum=running_sum+wearvol;
            //printf("%10.9f\n", running_sum);
138         //step forward the tprev value
            tprev=t;
140


142         //if on the first step, dont read from file 2, just write to file 3
            if (k==1)
144         {
              if (row_iter<=30){
146             printf("%20.10f\n",running_sum);
              }
148           // write the wear and time values from track 1 to F3
              fprintf(f3, "%f,\t%f\n", t, running_sum);
150           //printf("%e %e\n",t, running_sum);
            }
152         else if(k>=2)
            {
154           //read in the data from the running save file:
              wearin=0;
156           tin=0;
              if(fgets(line2,100,f2)!=NULL)
158           {
                sscanf(line2, "%f,\t%f\n",&tin, &wearin);
160             fprintf(f3,"%f,\t%f\n",t,running_sum+wearin);
                //printf("\tRead line: %s\n",line2);
162             //printf("%f %f %f %f\n",tin, wearin, t, running_sum);
              }
164           else
              {
166             fprintf(f3, "%f,\t%f\n", t, running_sum+wearin);
```

```
            //printf("f2 line was null\n");
168         }

170         //print out some data for error checking
            if (row_iter<=30){
172           printf("%20.10f\t%20.10f\n",running_sum, wearin+running_sum);
            }
174

          }
176       }
          row_iter=row_iter+1;
178     }
        //write all of f3 into f2, clear f3.  First close files.
180     fclose(fp);
        fclose(f2);
182     fclose(f3);

184     //reopen files
        FILE *f4 = fopen(file_name2, "w");
186     if (f4 == NULL)
        {
188       //printf("Error opening file to write to!\n");
          exit(1);
190     }
        FILE *f5 = fopen(file_name3, "rt");
192     if (f5 == NULL)
        {
194       //printf("Error opening file to write to!\n");
          exit(1);
196     }
        //read from f3, write to f2
198     while (fgets(line5, 100, f5)!=NULL)
        {
```

```
200           //sscanf(line5, "%f %f \n",&tin, &wearin);
              //printf("Line to be written: %s\n",line5);
202           fprintf(f4, "%s", line5);
            }
204
             //close track file, reiterate loop
206          fclose(f4);
             fclose(f5);
208       }


210


212


214       //delete the track files and the support files
          if (remove(file_name3)!= 0){
216         perror("error deleting support file");
          }
218       for (k=1; k<=atoi(argv[3]); k++){
                sprintf(file_name,"%sTrack%d",argv[1],k);
220         if (remove(file_name)!= 0){
              perror("error deleting track file");
222         }
            }
224


226


228
          return 0;
230     }
    }
```

.3 WearPrediction2.cmd

The following is the ADAMS command file for the wear prediction plugin. Throughout the code, replace "MODELNAME" with the name of the model to be analysed, and "PATHNAME" with the file path to the folder which contains cparse2.exe.

```
!
interface dialog_box create &
   dialog_box_name = .gui.WearPrediction2 &
   location = 1219.0, 415.0 &
   height = 200.0 &
   width = 250.0 &
   units = pixel &
   horiz_resizing = scale_all &
   vert_resizing = scale_all &
   title = "AU Wear Prediction Plugin" &
   iconifiable = no &
   decorate = yes &
   resizable = no &
   grab_all_input = no
!
interface push_button create &
   push_button_name = .gui.WearPrediction2.button_1 &
   location = 176.0, 173.0 &
   height = 25.0 &
   width = 72.0 &
   units = pixel &
   horiz_resizing = attach_right &
   vert_resizing = attach_bottom &
   label = "Cancel" &
   commands = "interface dialog undisplay dialog=$_parent"
!
```

```
interface push_button create  &
   push_button_name = .gui.WearPrediction2.button_2  &
   location = 104.0, 173.0  &
   height = 25.0  &
   width = 70.0  &
   units = pixel  &
   horiz_resizing = attach_right  &
   vert_resizing = attach_bottom  &
   label = "Apply"  &
   commands = "analysis collate_contacts analysis_name=.MODELNAME.Last_Run
      contact_name=($field_1) ", &
            "variable create variable_name=iter integer_value=0", &
            "while condition=(iter<1000)", &
            "    variable modify variable_name=iter integer_value=(eval(iter
               +1))", &
            "    ! variable create variable_name=test string=(eval(\".
               MODELNAME.Last_Run.\"//$field_1//\".TRACK_\"//iter))", &
            "    if condition=(eval(DB_EXISTS(eval(\".MODELNAME.Last_Run
               .\"//$field_1//\".TRACK_\"//iter))))", &
            "        if cond=(\"numeric\" == \"analysis\" || \"numeric\" ==
               \"request\" || \"numeric\" == \"results\" || \"numeric\" ==
               \"graphics\")", &
            "    interface container execute container = .gui.
               ppt_file_export.c_analysis undisplay = no", &
            "        else", &
            "    interface container execute container = .gui.
               ppt_file_export.c_numeric undisplay = no", &
            "            numeric_results write  &", &
            "        result_set_component_name = (eval($field_1//\".TIME\")
               ), (eval($field_1//\".TRACK_\"//iter//\".I_Normal_Force.Mag
               \")), (eval($field_1//\".TRACK_\"//iter//\".Slip_Velocity.
               Speed\"))  &", &
            "        sort_by = by_time  &", &
```

```
49              "          order = ascending  &", &
                "          write_to_terminal = off  &", &
51              "          file_name = (eval(\"PATHNAME/Track\"//iter))  & ", &
                "                    & ", &
53              "  ", &
                "        end", &
55              "    else", &
                "        variable modify variable_name=iter integer_value=(eval(
                    iter-1))", &
57              "        break", &
                "    end", &
59              "end  ", &
                "if cond=(\"numeric\" == \"analysis\" || \"numeric\" == \"
                    request\" || \"numeric\" == \"results\" || \"numeric\" == \"
                    graphics\")", &
61              "    interface container execute container = .gui.
                    ppt_file_export.c_analysis undisplay = no", &
                "else", &
63              "    interface container execute container = .gui.
                    ppt_file_export.c_numeric undisplay = no", &
                "    numeric_results write  &", &
65              " result_set_component_name = (eval(\".MODELNAME.Last_Run.\"//
                    $field_5//\".Q\")) &", &
                " sort_by = by_time  &", &
67              "        order = ascending  &", &
                " write_to_terminal = off  &", &
69              " file_name = (eval(\"PATHNAME/CArea\"))  & ", &
                "             & ", &
71              " ", &
                "end", &
73              "system command=(eval(\"PATHNAME/cparse2.exe PATHNAME/\"//
                    $field_1//\" \"//iter//\" \"//$field_2//\" \"//$field_3//\"
                    \"//$field_4)) &", &
```

```
                "  send_output_to_info_window=off &", &
75              "  echo_to_logfile=off", &
                "! variable delete variable_name=test", &
77              "variable delete variable_name=iter", &
                "interface dialog display dia=.gui.ppt_file_import parameter=\"
                    testdata\"", &
79              "if condition = (str_find(str_remove_whitespace(\"\"), \" \") >
                    0)", &
                "   if condition = (str_find(str_remove_whitespace(\"\"), \",\")
                    == 0)", &
81              "     mdi gui_utl_alert_box_1 type=\"Error\" text=\"Units must
                    be comma seperated. \"", &
                "     return", &
83              "  end ! IF", &
                "end ! IF", &
85              "if condition=(eval(DB_EXISTS(\".\"//$field_1//\"WEARDATA.dat\")
                    ))", &
                "    entity delete entity_name=(eval(\".\"//$field_1//\"WEARDATA
                    .dat\"))", &
87              "end", &
                "file testdata read measures model_name=(eval(\".MODELNAME.\"//
                    $field_1//\"_Wear_Estimate\")) use_file_column_labels=no
                    independent_column_index=1 file_name=(eval(\"PATHNAME/\"//
                    $field_1//\"WEARDATA.dat\"))", &
89              "", &
                "interface plot panel mode_set mode=measure ", &
91              "interface plot panel reload", &
                "interface tree_navigator refresh=TRUE", &
93              "interface dia undisp dia=.gui.ppt_file_import", &
                "interface dialog undisplay dialog=.gui.msg_box  ", &
95              "interface dialog execute dialog=$_parent undisp=no"
!
97 interface push_button create  &
```

```
    push_button_name = .gui.WearPrediction2.button_3  &
 99 location = 34.0, 173.0  &
    height = 25.0  &
101 width = 66.0  &
    units = pixel  &
103 horiz_resizing = attach_right  &
    vert_resizing = attach_bottom  &
105 label = "OK"  &
    commands = "analysis collate_contacts analysis_name=.MODELNAME.Last_Run
        contact_name=($field_1) ", &
107             "variable create variable_name=iter integer_value=0", &
                "while condition=(iter<1000)", &
109             "    variable modify variable_name=iter integer_value=(eval(iter
                    +1))", &
                "    ! variable create variable_name=test string=(eval(\".
                    MODELNAME.Last_Run.\"//$field_1//\".TRACK_\"//iter))", &
111             "    if condition=(eval(DB_EXISTS(eval(\".MODELNAME.Last_Run
                    .\"//$field_1//\".TRACK_\"//iter))))", &
                "        if cond=(\"numeric\" == \"analysis\" || \"numeric\" ==
                    \"request\" || \"numeric\" == \"results\" || \"numeric\" ==
                    \"graphics\")", &
113             "    interface container execute container = .gui.
                    ppt_file_export.c_analysis undisplay = no", &
                "        else", &
115             "    interface container execute container = .gui.
                    ppt_file_export.c_numeric undisplay = no", &
                "            numeric_results write  &", &
117             "        result_set_component_name = (eval($field_1//\".TIME\")
                    ), (eval($field_1//\".TRACK_\"//iter//\".I_Normal_Force.Mag
                    \")), (eval($field_1//\".TRACK_\"//iter//\".Slip_Velocity.
                    Speed\"))  &", &
                "        sort_by = by_time  &", &
119             "        order = ascending  &", &
```

```
                "          write_to_terminal = off  &", &
121             "          file_name = (eval(\"PATHNAME/Track\"//iter))  & ", &
                "                      & ", &
123             "  ", &
                "        end", &
125             "    else", &
                "        variable modify variable_name=iter integer_value=(eval(
                    iter-1))", &
127             "        break", &
                "    end", &
129             "end  ", &
                "if cond=(\"numeric\" == \"analysis\" || \"numeric\" == \"
                    request\" || \"numeric\" == \"results\" || \"numeric\" == \"
                    graphics\")", &
131             "    interface container execute container = .gui.
                    ppt_file_export.c_analysis undisplay = no", &
                "else", &
133             "    interface container execute container = .gui.
                    ppt_file_export.c_numeric undisplay = no", &
                "    numeric_results write  &", &
135             " result_set_component_name = (eval(\".MODELNAME.Last_Run.\"//
                    $field_5//\".Q\")) &", &
                " sort_by = by_time  &", &
137             "        order = ascending  &", &
                " write_to_terminal = off  &", &
139             " file_name = (eval(\"PATHNAME/CArea\"))  & ", &
                "             & ", &
141             " ", &
                "end", &
143             "system command=(eval(\"PATHNAME/cparse2.exe PATHNAME/\"//
                    $field_1//\" \"//iter//\" \"//$field_2//\" \"//$field_3//\"
                    \"//$field_4)) &", &
                "  send_output_to_info_window=off &", &
```

```
145                "  echo_to_logfile=off", &
                   "! variable delete variable_name=test", &
147                "variable delete variable_name=iter", &
                   "interface dialog display dia=.gui.ppt_file_import parameter=\"
                       testdata\"", &
149                "if condition = (str_find(str_remove_whitespace(\"\"), \" \") >
                       0)", &
                   "   if condition = (str_find(str_remove_whitespace(\"\"), \",\")
                       == 0)", &
151                "     mdi gui_utl_alert_box_1 type=\"Error\" text=\"Units must
                       be comma seperated. \"", &
                   "     return", &
153                "  end ! IF", &
                   "end ! IF", &
155                "if condition=(eval(DB_EXISTS(\".\"//$field_1//\"WEARDATA.dat\")
                       ))", &
                   "    entity delete entity_name=(eval(\".\"//$field_1//\"WEARDATA
                       .dat\"))", &
157                "end", &
                   "file testdata read measures model_name=(eval(\".MODELNAME.\"//
                       $field_1//\"_Wear_Estimate\")) use_file_column_labels=no
                       independent_column_index=1 file_name=(eval(\"PATHNAME/\"//
                       $field_1//\"WEARDATA.dat\"))", &
159                "", &
                   "interface plot panel mode_set mode=measure ", &
161                "interface plot panel reload", &
                   "interface tree_navigator refresh=TRUE", &
163                "interface dia undisp dia=.gui.ppt_file_import", &
                   "interface dialog undisplay dialog=.gui.msg_box   ", &
165                "interface dialog execute dialog=$_parent undisp=yes"
    !
167 interface field create &
       field_name = .gui.WearPrediction2.field_1 &
```

```
169     location = 90.0, 2.0  &
        height = 27.0  &
171     width = 158.0  &
        units = pixel  &
173     horiz_resizing = attach_left  &
        vert_resizing = attach_top  &
175     scrollable = no  &
        editable = yes  &
177     required = yes  &
        execute_cmds_on_exit = no  &
179     number_of_values = 1  &
        string_type = alpha_numeric  &
181     add_quotes = yes
    !
183 interface label create  &
        label_name = .gui.WearPrediction2.label_1  &
185     help_text =  &
                    "Just the name of the contact you wish to examine, e.g.
                       CONTACT_1"  &
187     location = 2.0001, 2.0  &
        height = 27.0  &
189     width = 87.0  &
        units = pixel  &
191     horiz_resizing = attach_left  &
        vert_resizing = attach_top  &
193     justified = left  &
        text = "Contact Name"
195 !
    interface field create  &
197     field_name = .gui.WearPrediction2.field_2  &
        location = 92.0, 68.0  &
199     height = 29.0  &
        width = 154.0  &
```

```
    units = pixel  &
    horiz_resizing = attach_left  &
    vert_resizing = attach_top  &
    scrollable = no  &
    editable = yes  &
    preload_strings = "0.00000000000000236850"  &
    required = yes  &
    execute_cmds_on_exit = no  &
    number_of_values = 1  &
    numeric_type = real  &
    upper_check = none  &
    lower_check = none
!
interface label create  &
    label_name = .gui.WearPrediction2.label_2  &
    help_text =  &
                "First coefficient of the Modified Archard Equation, for
                    Traditional Archard, set to 0."  &
    location = 2.0001, 70.0  &
    height = 27.0  &
    width = 87.0  &
    units = pixel  &
    horiz_resizing = attach_left  &
    vert_resizing = attach_top  &
    justified = left  &
    text = "C_1"
!
interface field create  &
    field_name = .gui.WearPrediction2.field_3  &
    location = 92.0, 100.0  &
    height = 29.0  &
    width = 154.0  &
    units = pixel  &
```

```
233     horiz_resizing = attach_left  &
        vert_resizing = attach_top  &
235     scrollable = no  &
        editable = yes  &
237     preload_strings = "6.3197059745759692"  &
        required = yes  &
239     execute_cmds_on_exit = no  &
        number_of_values = 1  &
241     numeric_type = real  &
        upper_check = none  &
243     lower_check = none
    !
245 interface field create  &
        field_name = .gui.WearPrediction2.field_4  &
247     location = 92.0, 132.0  &
        height = 31.0  &
249     width = 154.0  &
        units = pixel  &
251     horiz_resizing = attach_left  &
        vert_resizing = attach_top  &
253     scrollable = no  &
        editable = yes  &
255     preload_strings = "-0.00000000000000095089"  &
        required = no  &
257     execute_cmds_on_exit = no  &
        number_of_values = 1  &
259     numeric_type = real  &
        upper_check = none  &
261     lower_check = none
    !
263 interface label create  &
        label_name = .gui.WearPrediction2.label_3  &
265     help_text =  &
```

```
                "Second coefficient of the Modified Archard Equation, cannot be
                    0, can be anything if using Traditional Archard." &
    location = 2.0001, 100.0 &
    height = 29.0 &
    width = 87.0 &
    units = pixel &
    horiz_resizing = attach_left &
    vert_resizing = attach_top &
    justified = left &
    text = "C_2"
!
interface label create &
    label_name = .gui.WearPrediction2.label_4 &
    help_text = &
                "Third coefficient of the Modified Archard Equation, or the
                    dimensional wear coefficient if using Traditional Archard
                    model." &
    location = 2.0001, 132.0 &
    height = 31.0 &
    width = 87.0 &
    units = pixel &
    horiz_resizing = attach_left &
    vert_resizing = attach_top &
    justified = left &
    text = "C_3"
!
interface label create &
    label_name = .gui.WearPrediction2.label_5 &
    location = 2.0, 28.0 &
    height = 39.0 &
    width = 89.0 &
    units = pixel &
    horiz_resizing = attach_left &
```

```
    vert_resizing = attach_top  &
297     justified = left  &
    text = "CArea Var. "
299 !
interface field create  &
301     field_name = .gui.WearPrediction2.field_5  &
    location = 91.0, 34.0  &
303     height = 29.0  &
    width = 157.0  &
305     units = pixel  &
    horiz_resizing = attach_left  &
307     vert_resizing = attach_top  &
    scrollable = no  &
309     editable = yes  &
    required = no  &
311     execute_cmds_on_exit = no  &
    number_of_values = 1  &
313     string_type = literal  &
    add_quotes = yes
```