Impact Velocity, Almen Strip Curvature and Residual Stress Modelling in Vibratory Finishing

by

David Ciampini

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy

Graduate Department of Mechanical and Industrial Engineering

University of Toronto

© Copyright by David Ciampini (2008)
Abstract

The surface-normal impact velocity distributions, impact frequencies and impact power per unit area were measured using a force sensor in a vibratory finisher for two types of spherical media. These parameters control the degree, rate and character of plastic deformation of a workpiece surface in vibratory finishing. The force sensor was also used to quantify the effect of media type, finisher amplitude, and location within the finisher on the probability distribution of the particle impact velocity normal to the workpiece. It was found that reducing the total media mass in the finisher and moving closer to the wall resulted in a more aggressive process. It was also found that contacts occurred periodically within time periods that corresponded to the finisher’s driving frequency.

The Almen system was adapted to a vibratory finishing process to characterize the effect of varying process parameters for the purposes of process development and control. Saturation curves for two types of aluminum Almen strips were obtained by finishing at two distinct conditions. Comparison with the normal contact forces and effective impact velocities, measured for both these conditions, provided insight into the mechanics of the vibratory finishing process. An electromagnetic apparatus was constructed to simulate the normal impacts in the vibratory finisher. It was found that surface-normal impacts at velocities comparable to the higher range in the vibratory
finisher produced Almen saturation curves similar to those created in the vibratory finisher. This provided support for the modeling approximation of treating all contact events in a vibratory finisher as effective surface-normal impacts, and the accuracy of the effective impact velocity measurement.

A model of the process by which Almen strips were plastically deformed by media impacts in vibratory finishing was presented. The motivation was to extend the use of Almen strip measurements as a means of characterizing vibratory finishing through an improved understanding of the process parameters that controlled time-dependent curvature development. Two thicknesses of Almen strip were tested for two finishing conditions. The quantitative agreement between the model saturation curves and the experimental curves was fair, although the overall trends were predicted very well.
Acknowledgements

First and foremost I offer my sincerest gratitude to my supervisors. It is difficult to overstate my gratitude to my Ph.D. supervisors, Dr. J.K. Spelt and Dr. M. Papini. I have developed greatly as a researcher and in character under their guidance. I am honored and thankful that I was chosen to be mentored by them. They have been my role models; as such, I have been inspired by their abilities and success and will endeavor to achieve a similar career path to their own. Any future success I may have will largely be due to their involvement with me. I hope that my future work will bring them pride and will be a tribute to them.

I would like to express my sincere thanks to the Natural Sciences and Engineering Research Council of Canada. This research would not have been possible without their financial support.

In addition, I would like to express my deep and sincere gratitude to many people for their support during my studies; without their assistance this work and my graduation would have not been possible.

I extend my thanks to Shivinder Babbar for doing an excellent job as a research assistant. In addition, I express my gratitude to Professor Gang Ya for his involvement.

I must acknowledge the invaluable contributions of Len Roosman, Jeff Sansome, and Mike Smith for helping me in the student machine shop. A special thanks goes to Joseph Amankrah for manufacturing parts of an important apparatus for me. I'd also like to thank Tomas Bernreiter for making lab equipment available to me.

I thank my thesis advisory committee members Dr. B. Benhabib and Dr. J. Mostaghimi for their helpful input during my qualifying exam, seminar and departmental defense.

The debt of gratitude I owe my parents, Camillo and Maria, is so large that it can never be repaid or expressed in its entirety. Those who benefit from this work also owe my parents thanks as I would not have completed it had they not sheltered me from part of the emotional and financial burden it imposed. Their faith in me and support of my education and future has been both astonishingly generous and unwavering.
I owe my sister thanks for the support and friendship she has given me. Giulia, thanks for keeping me in touch with my inner child especially during hard times. I also owe her my deepest apologies for spelling her name incorrectly in the acknowledgements of my master’s thesis; this is a mistake I have been reminded of periodically since its occurrence. Giulia, I hope that you can forgive me now. I also would like thank you for the cheer* of support during my master’s convocation and apologize for my initial negative reaction. I will be there to cheer* for you when you graduate.

I’ve been very fortunate to have certain people in my life during my PhD studies. Their friendship and support which came in many different ways was much appreciated and I owe them my thanks: Martin Côté, Peter Bahoudian, Michael Liang, and Claudia Baldissera.

Warm thanks goes to Christine Francisco for her support and for sharing this difficult portion of my life with me even when I could not offer her much quality time; I hope that we see better days together.

I undoubtedly have failed to explicitly mention other important individuals or failed to thank others sufficiently; to those, I extend my apologies and gratitude. Your collective support and friendship have added greatly to the quality of my life and to the motivation I needed to finish this work.

* “WOOOOOOOOOOOOOOO!”
# Table of Contents

ABSTRACT .......................................................................................................................... II

ACKNOWLEDGEMENTS ................................................................................................. IV

TABLE OF CONTENTS ................................................................................................. VI

LIST OF TABLES ........................................................................................................... IX

LIST OF FIGURES ........................................................................................................ X

1. **THESIS INTRODUCTION** .................................................................................. 1
   1.1. Literature Review .............................................................................................. 5
   1.2. Thesis Objectives .............................................................................................. 19
   1.3. Structure of Thesis ............................................................................................ 19

2. **MEDIA IMPACT VELOCITY SENSOR** ............................................................ 21
   2.1. Nomenclature .................................................................................................... 21
   2.2. Introduction to the Measurement of Impact Velocities in Vibratory Finishers ... 23
   2.3. Apparatus .......................................................................................................... 24
       2.3.1. Force Sensor and Housing ........................................................................ 24
       2.3.2. Vibratory Finisher and Media ................................................................... 27
       2.3.3. Impact Velocity from Force Measurements in Ball Drop Experiments ... 28
   2.4. Results and Discussion ..................................................................................... 31
       2.4.1. Interpretation of Impact Forces .................................................................. 31
           2.4.1.1. Bursts ................................................................................................ 31
           2.4.1.2. Classification of Contact Events within a Burst ............................... 34
       2.4.2. Media Impact Velocity ................................................................................ 38
           2.4.2.1. Impact and Effective Impact Velocity ..............................................38
           2.4.2.2. Effect of Sampled Signal Duration ................................................... 38
           2.4.2.3. Effect of Sensor Location and Tub Amplitude ................................. 42
       2.4.3. Energy Flux ............................................................................................... 46
   2.5. Conclusions ....................................................................................................... 48

3. **ALMEN SYSTEM** ............................................................................................... 50
   3.1. Introduction ....................................................................................................... 50
   3.2. Apparatus .......................................................................................................... 52
       3.2.1. Almen Strips ............................................................................................. 52
       3.2.2. Vibratory Finisher and Media ................................................................... 53
       3.2.3. Vacuum Almen Strip Holder .................................................................... 53
       3.2.4. Arc-height gauge ....................................................................................... 54
       3.2.5. Vibratory Finishing Simulator .................................................................. 56
   3.3. Results and Discussion ..................................................................................... 59
       3.3.1. Vibratory Finisher Saturation Curves .......................................................59
       3.3.2. Vibratory Finishing Simulator Saturation Curves .................................... 64
   3.4. Conclusion ........................................................................................................ 66
4. MODELING OF STRESSES................................................................................. 68

4.1. Introduction........................................................................................................ 68
4.2. Model................................................................................................................ 69

4.2.1. Part 1: FEA ............................................................................................... 69

4.2.1.1. FEA Geometry .................................................................................. 69
4.2.1.2. FEA in LS-DYNA ............................................................................ 70
4.2.1.3. FEA Assumptions .............................................................................70
4.2.1.3.1. Holder compliance and boundary conditions ................................... 70
4.2.1.3.2. Aluminum sheet constitutive models................................................ 72
4.2.1.4. Model Validation .............................................................................. 72
4.2.1.4.1. Energy Storage and Dissipation........................................................ 74
4.2.1.5. Centerline Radial Stress, Curvature and Arc Height ....................... 75
4.2.1.6. Axisymmetric FEA Results .............................................................. 76
4.2.1.7. Distribution of absorbed energy........................................................ 89

4.2.2. Part 2: Process Model ............................................................................... 90

4.2.2.1. Calculating the energy absorption history ........................................ 91
4.2.2.2. Obtaining stresses and deflection from an absorbed energy state ... 95
4.2.2.3. Process Model Results and Comparison with Experiments ............. 96
4.2.2.4. Limitations of the Model ................................................................ 102
4.2.2.5. Almen strips as probes for determining vibratory finishing process contact conditions ............................................................................................... 105

4.3. Conclusions..................................................................................................... 106
4.4. Appendix......................................................................................................... 108

4.4.1. 1D process model.................................................................................... 108

5. THESIS CONCLUSIONS.................................................................................... 110

5.1. Summary of Useful Findings.......................................................................... 110
5.2. Original Contributions .................................................................................... 111
5.3. Industrial and Research Applications ............................................................. 112
5.4. Recommendations for Future Work: Improvements and Research............ 113

6. REFERENCES...................................................................................................... 116

7. APPENDIX............................................................................................................ 118

7.1. An algorithm for determining the deformation of a beam given the curvature of the elements it is comprised of......................................................................... 118
7.2. Computer Code............................................................................................... 123

7.2.1. Data Acquisition and Digital Signal Processing......................................... 123
7.2.1.1. Force Sensor Data Acquisition .........................................................125
7.2.1.2. Multiple Accelerometers Data Acquisition ..................................... 127
7.2.2. Signal Processing Code........................................................................... 129
7.2.3. FFT of force signal.................................................................................. 139
7.2.4. Bust and Peak Analysis ........................................................................... 141
7.2.5. Perform Statistics on Consolidated Data ................................................. 149
7.2.6. 1D Process Model Code.......................................................................... 153
7.2.7. Plastic and elastic work code .................................................................. 160
7.2.8. 2D process model code .......................................................................... 172
7.2.9. Moment distribution from stresses ....................................................... 217
7.2.10. Arc Height from centerline stresses .................................................... 223
7.2.11. Converting Ansys output files to comma separated values files .......... 226
7.2.12. Finite difference solution to beam bending problem .......................... 228
7.2.13. Ansys batch code for running multiple impact analysis ...................... 229
7.2.14. Ansys batch code for obtaining post processing data from Ansys ....... 231
7.2.15. Ansys code to interface to Matlab used in process model ................. 232
7.3. Force Apparatus Drawings ...................................................................... 233
7.4. Modified Almen Gauge Drawings ............................................................. 247
List of Tables

Table 2-1: Average media diameter and mass. Ten measurements (± indicates three standard deviations) ........................................................................................................................................................................ 28

Table 2-2: Analysis of force signals recorded at locations of Fig. 2-8 with 41 kg of porcelain media (41P), and either 91 kg (91S) or 112 kg (112S) of steel media. average contact frequency, \( f_{avg} \), average effective impact velocity, average, \( v_{avg} \),
maximum effective impact velocity, \( v_{max} \), with corresponding impact type, total energy flux, \( E_{Total} \), percentages of time and number of contacts for each type of contact. Data for positions ‘a’ and ‘b’ were derived from a 300 s force signal, ‘c’ and ‘d’ from a 150 s signal. ......................................................................................................................... 44

Table 3-1: Finishing impact conditions measured using the impact velocity probe of Section 2 at locations “a” and “b” of Fig. 2-8 ........................................................................................................................................... 59

Table 3-2: Average arc height and standard deviation (in brackets) of each set of five Almen strips finished in the vibratory finisher at the conditions in Fig. 3-6 ............. 61

Table 3-3: “Almen intensity” and “saturation time” obtained using Eqs. (3-2) and (3-3) with values of parameters A and B given in the table and with C=1 and D=0........ 64

Table 7-1: Angles of rotation to enforce first derivative continuity. Where \( A \) is the arc height over a 63.5 mm (2.5”) span................................................................................................................................. 120
List of Figures

Fig. 1-1: Schematic illustrations of a surface mechanical attrition (SMA) treatment to create plastic deformation of a sample surface and ultimately a nanocrystalline surface structure. Vibration generator frequency (50Hz to 20KHz). Reproduced from Tao et al. (2002). ................................................................. 2

Fig. 1-2: The Almen System, Reproduced from Cao et al. (1995)................................. 3

Fig. 1-3: Almen intensity / saturation curve. Almen arc height vs Shot Peening time Cao et al. (1995). ................................................................. 4

Fig. 1-4: Bowl type vibratory finisher. Reproduced from Yabuki et al. (2002)................. 6

Fig. 1-5: Force signal produced by a resistive strain sensor with a 10mm contact area. Reproduced from Wang et al. (2000). ........................................... 6

Fig. 1-6: Resistive strain gauge sensor. Reproduced from Wang et al. (2000). ............... 7

Fig. 1-7: Scanning electron micrographs of individual impact sites on a workpiece (AA1100-O) after 10 seconds of finishing showing evidence of normal impact in dry conditions. Reproduced from Wang et al. (2000). ............................................... 8

Fig. 1-8: Scanning electron micrographs of individual impact sites on a workpiece (AA1100-O) after 10 seconds of finishing showing evidence of sliding in wet conditions. Reproduced from Wang et al. (2000). ........................................... 8

Fig. 1-9: Resistive force sensor. The red part is the 3mm sensor pad. Reproduced from Yabuki et al. (2002)................................................................. 9

Fig. 1-10: The relationship between the impact frequency and maximum force during impact for the bowl and tub finisher in both wet and dry conditions. Reproduced from Baghbanan et al. (2003) ............................................................... 10

Fig. 1-11: Cross section of ceramic debris in the surface of AA1100-O after 90min of finishing in dry conditions. Reproduced from Baghbanan et al. (2003) ............... 11

Fig. 1-12: SEM micrograph of an AA1100-O workpiece cross-section with embedded ceramic wear particle from the media as a result of 150min of finishing in the wet conditions. Reproduced from Baghbanan et al. (2003) ............................................................... 11

Fig. 1-13: Hardness changes with depth into workpiece. (a) AA1100-O 30min Dry, (b) AA1100-O 150min Wet, (c) AA6061-T6 150min Wet . Reproduced from Baghbanan et al. (2003) ............................................................... 12

Fig. 1-14: Almen strip curvature vs. finishing time for AA1100-O, AA6061-T6 and Cu10100 in the bowl finisher under two conditions: (BD) bowl dry; (BW) bowl wet. Reproduced from Baghbanan et al. (2003) ............................................................... 13

Fig. 1-15: Almen strip curvature vs. finishing time for AA1100-O, AA6061-T6 and Cu10100 in the tub finisher under two conditions: (TD) tub dry; (TW) tub wet. Reproduced from Baghbanan et al. (2003) ............................................................... 13

Fig. 1-16: Almen strip curvature vs. finishing time for AA1100-O in the tub finisher under four conditions: (TD) tub dry; (TDG) tub dry with a grease film behind the strip; (TW) tub wet; (TWG) tub wet with grease film. Reproduced from Baghbanan et al. (2003) ............................................................... 14

Fig. 1-17: Development of compressive residual stress from shot impacts. Reproduced from Kobayashi et al. (1998) ............................................................... 18
Fig. 2-1: Sensor force output as a function of time after sudden removal of a 1.045 kg calibration mass. .......................................................................................................................... 25
Fig. 2-2: Force sensor and housing schematic. Not to scale. ................................................. 26
Fig. 2-3: Cross-section of the vibratory finisher (reproduced from [3]) showing direction of media circulation and resulting slope of the free surface. All of the dimensions are given in mm. ............................................................................................................................... 27
Fig. 2-4: (a) Solenoid actuator holding steel sphere a known height above impact force sensor. (b) Six superimposed images of a sphere falling (down) towards the sensor cap (y=0) after being released. y₁ and y₂ are the distances from the sensor to the bottom of the ball for the first and last exposures, respectively. ........................................ 28
Fig. 2-5: Superimposed force signals from drop experiments with steel and porcelain spheres at impact velocities of 0.36 and 0.50 m/s. The upper and lower curves encompass the range of measured values in ten repetitions. The central curve represents the mean of all the ten repetitions................................................................. 29
Fig. 2-6: Sensor impulse recorded for steel and porcelain media impacts with known impact velocities. ..................................................................................................................... 30
Fig. 2-7: Maximum impact force recorded by the sensor for steel and porcelain media impacts with known impact velocities. .................................................................................. 30
Fig. 2-8: Plan view of tub showing four sensor locations: a, b, c and d. Arrows indicate orientation of sensor. High wall (HW) and low wall (LW) as shown in Fig. 2-3. Sensor depth was 210 mm above bottom of tub in all cases. ........................................ 31
Fig. 2-9: (a) Typical signal recorded over 5 s when the sensor is submerged in the vibratory finisher. The portion of the signal between 0.8 and 1.5 s is expanded in (b). Sensor located at ‘a’ (Fig. 2-8), 91 kg of steel media...................................................... 33
Fig. 2-10: Overlapping bursts from a 15s force signal 41 kg of porcelain media in location ‘a’. Individual bursts are shown in several different shades and marker styles. .............................................................................................................................................. 34
Fig. 2-11: Expansion of the impact contact at approximately 1.38 s lasting 50μs in...... 35
Fig. 2-12: Expansion of a single non-impact contact (duration=0.8 ms) at approximately 1.46 s in Fig. 2-9(b). ................................................................................................................................. 36
Fig. 2-13: Interpretation of two types of contact. (a) non-impact contact showing media being pressed onto sensor cap with force P by surrounding media, (b) media impact with a velocity v. ................................................................................................................................. 36
Fig. 2-14: Peak interval distributions from 15 s force signal collected at location ‘a’. o - 91 kg steel media. □ - 41 kg porcelain media. .......................................................................................... 37
Fig. 2-15: The effect of sampling period (shown in the legend in s) on the probability of impact velocities being sensed in several velocity ranges: (a) impact velocities 0-0.18 m/s; (b) impact velocities 0.18-0.40 m/s. Steel media at position ‘a’ (Fig. 2-8). ............................................................................................................................................. 40
Fig. 2-16: Finishing duration to reach 95% surface coverage for impact velocities found at position ‘a’ (Fig. 2-8) 91 kg of steel media and a 5 min force measurement sampling duration. Each bar represents the finishing time for a range of impact velocities spanning 0.008 m/s................................................................................................................ 42
Fig. 2-17: Frequency of impacts occurring per unit area as a function of effective impact velocity, for sensor at two stand off distances (‘a’ (44 mm) and ‘b’ (95 mm) in Fig.
2-8) from the high wall. Derived from 300 s of impact force data. In legend, number gives mass of media in kg, P=porcelain media, S=steel media. ...............................45

Fig. 2-18 Frequency of impacts occurring per unit area as a function of effective impact velocity at a constant standoff distance of 95 mm for three sensor positions: facing the high wall, side wall, and low wall (‘b’, ‘c’, ‘d’ in Fig. 2-8). Derived from 150 s of impact force data. In legend, number gives mass of media in kg, P=porcelain media, S=steel media. ...............................................................................................46

Fig. 2-19: Energy flux as a function of effective impact velocity for three sensor positions: facing the high wall, side wall, and low wall (‘b’, ‘c’, ‘d’ in Fig. 2-8). In the legend, number gives mass of media in kg, P=porcelain media, and S=steel media. Results for positions ‘a’ and ‘b’ were derived from 300 s of impact force data, and from 150 s at ‘c’ and ‘d’.............................................................................47

Fig. 3-1: Schematic of Almen strip vacuum fixture immersed in tub finisher. ............53
Fig. 3-2: Photograph of Almen strip vacuum holder. .......................................................54
Fig. 3-3: Schematic of arc-height gauge. ........................................................................55
Fig. 3-4: Schematic of vibratory finisher simulator. ..............................................................56
Fig. 3-5: Velocity of dropping ball as a function displacement from its initial position: measured - open symbols representing four repeat experiments; predicted assuming gravitational acceleration – filled symbols. ..............................................................58
Fig. 3-6: Saturation curves produced using two finishing conditions and three types of Almen strip (617 µm thick 3003-O and 3003-H14; 779 µm thick 3003-O) (a) shows an expanded view of the data of Fig. (b) at low finishing durations. ...............................61
Fig. 3-7: Exponential curve fits (Eq. (3-2)) through vibratory finisher saturations curves of the O-temper strips. .............................................................................................. 63
Fig. 3-8: Vibratory finisher simulator saturation curves for. 617 µm strips (open symbols) and 779 µm strips (filled symbols). Impact velocities of 0.33 m/s (circles), 0.22 m/s (squares), and 0.18 m/s (triangles)............................................................................ 65
Fig. 4-1: Axisymmetric finite element mesh of the spherical media impacting the aluminium sheet (617 µm thickness). ....................................................................... 71
Fig. 4-2: The residual centerline radial stresses produced by shot impacts at (a) 50 m/s and (b) 75 m/s impact as reported by Meguid et al. (1999) and reproduced using the present model at different levels of mesh refinement. The stress axes are normalized by division by 600 MPa. ...........................................................................................73
Fig. 4-3: The top-left corner of the 2D axisymmetric mesh used for the energy calculation. Nodes and their surrounding rectangular or square areas are shown. The curved boundary is a cut-away line indicating that there are additional nodes and areas that repeat out of view. ..............................................................75
Fig. 4-4: Displacement of Almen strip surface after repeated impacts at 0.15 m/s (elastic perfectly-plastic 617µm thick target). ..........................77
Fig. 4-5: The radial residual stress distribution history after each impact for 100 repeated impacts against a 617 µm thick strip: (a) 0.15 m/s impact velocity against an elastic perfectly-plastic target, (b) 0.15 m/s impacts using a power law hardening target, (c) 0.30 m/s impacts using an elastic perfectly-plastic target, (d) 0.30 m/s impacts using a power law hardening target. The numbered arrows indicate the curves corresponding to the first, second, third and one-hundredth impact; the unnumbered curves continue in the same sequence. ..............................................................80
Fig. 4-6: Arc height vs. absorbed energy for 100 impacts at the velocities indicated in the legend (each data point represents a single impact): (a) 617 μm thickness, power law, (b) 617 μm thickness, elastic perfectly-plastic, (c) 779 μm thickness, power law, (d) 779 μm thickness, elastic perfectly-plastic.

Fig. 4-7: Arc heights and corresponding cumulative absorbed energy after 100 repeated impacts of different velocities for several conditions shown in legend. ....

Fig. 4-8: Coefficient of restitution vs. absorbed energy for a range of impact velocities: (a) 617 μm thickness, power law, (b) 617 μm thickness, elastic-plastic, (c) 779 μm thickness, power law, (d) 779 μm thickness, elastic-plastic. Each marker style (shown in the legends of each figure) indicates the impact velocity (m/s) for each impact in a series of 100 impacts.

Fig. 4-9: Coefficient of restitution as a function of the energy previously absorbed by the target for repeated impacts of different velocities using the elastic-plastic constitutive model against a 617 μm thick target.

Fig. 4-10: The normalized distribution of absorbed energy as a function of a normalized radial distance from the centerline, after each impact, for a several series of 100 impacts at velocities of 1.0, 2.5, 5.0, 7.5, 10.0, 12.5, 15.0, 17.5, 20.0, 22.5, 25.0, 27.5, 30.0, 32.5, 35.0, 37.5, 40.0 cm/s. Radial distance normalized by distance containing 95% of total energy, absorbed energy normalized by the maximum absorbed energy in each distribution. The hardening model and 617 μm thick strip.

Fig. 4-11: Radius about impact centre encompassing more than 95% of impact energy absorption from the axisymmetric strain hardening model for 100 repeated impacts at 0.025 to 0.4 m/s in increments of 0.025 m/s for both the 617 μm and 779 μm sheet thicknesses.

Fig. 4-12: Sequence of random impacts at different velocities with varying amounts of overlap on a small region on the surface of the Almen strip. Each dash (11 in total) in the dashed line represents a 1D element and the boxes are the 2D elements used to determine the deformation caused by the collective effect of the impacts. (a) first impact, (b) second impact, (c) third impact, (d) fourth impact. A darker shade of gray corresponds to more energy absorbed by the Almen strip.

Fig. 4-13: A 20 by 100 element mesh of ANSYS™ SHELL181 elements deflecting due to constant and equivalent bending moments in both x and y directions (coordinate system shown).

Fig. 4-14: Comparison of experiment results and process model predictions for different vibratory finishing conditions: (a) 91Sa and 617 μm thick Almen strips (b), 112Sb and 617 μm thick Almen (c), 91Sa and 779 μm thick Almen strips (d), 112Sb and 779 μm thick Almen strips. The different model options and experimental data are identified by the legends in each sub figure. The error bars on the experimental data show the range of five repetitions. The 1D model is explained in the appendix.

Fig. 4-15: Comparison of experiment results and process model predictions for different vibratory finishing simulator conditions: (a) 0.22 m/s and 617 μm thick Almen strips (b), 0.33 m/s and 617 μm thick Almen (c), 0.22 m/s and 779 μm thick Almen strips. The different model options and experimental data are identified by the legends in
each sub figure. The error bars on the experimental data show $\pm$ three standard deviations of three repetitions................................................................. 102

Fig. 7-1 (a) A simple three element domain prior to the imposition of first derivative continuity at the nodes. The arc heights are exaggerated to make the degree of curvature noticeable; i.e. the figure is not to scale..................................................... 118
1. Thesis Introduction

Vibratory finishing is a versatile surface finishing process that has been widely used in industry over the past 50 years. The process is capable of finishing metal, ceramic and plastic parts. A few of the many applications where vibratory finishing has been successfully applied are the polishing of coins, gears, plastic eyeglass frames and golf balls; smoothing of sharp edges on steel parts that have been stamped or cast for the automotive industry; texturing and hardening of surfaces of electrical connectors; part drying applications; surface hardening, surface smoothing, and cleaning applications such as the removal of rust or other contaminants.

A general description of a vibratory finisher is a machine comprised of a container filled with media that is supported by springs, and a device capable of applying time varying forces to the container such that it develops a periodic motion. This is usually accomplished by one or more rotating shafts that have an eccentric mass. The frequency of rotation is usually in the range of $40 \pm 20$ Hz. The media is a collection of solid bodies that are significantly smaller than the container, with a shape and composition tailored to suit a specific surface finishing application.

The kinetic energy of the container is transferred to the media, causing the media to move in the container and become fluidized. Within limits imposed by the container motion, the media motion is best described as random over short time scales, but having a definite pattern over long time scales. That is, over long time scales, the media motion can be approximately described by a flow field. The short time scale is defined as the period of the container vibration and the long time scale is the time it takes for media to travel distances approximately equal to the container dimensions. A workpiece placed in the finisher over a long time scale will travel at a low speed around a path that eventually approximately overlaps. The workpieces are placed inside the container with the media and the kinetic energy of the media is imparted on the workpiece; if the machine is designed well and the process is controlled properly, the work done on the workpiece will improve its surface layer in the desired manner. Vibratory finishing is often performed with the addition of a liquid detergent and water in the container.
The present study focused on the investigation of the residual stress field that develops in the vicinity of the workpiece surface due to vibratory finishing. The multitude of impacts between the workpiece and media causes the workpiece surface region to plastically deform, creating a compressive residual stress field underneath the surface. In addition, this leads to a hardening of the surface. A hardened surface layer is desirable as it improves resistance to wear in machine elements. Determining how the process parameters affect the depth and extent of the hardened layer would allow greater control over the creation of this layer. It is useful to determine this because a compressive residual stress field is beneficial; it bestows protection to the workpiece from fatigue failure. It would also be an important first step in modeling the erosion of surfaces in vibratory finishing. In addition, it has been discovered by Tao et al. (2002) that mechanical surface treatments that cause a workpiece surface to repeatedly plastically deform are capable of creating a nanocrystalline surface layer through the refinement of large grains so that they approach a nanometre in size. A nanocrystalline material has unique properties that coarse grained materials do not have; for example, a nanocrystalline material can have a higher strength, hardness and wear resistance (Tao et al. 2002). Such nanocrystalline surface layers have been created by machines that closely resemble vibratory finishers (see Fig. 1-1).

![Fig. 1-1: Schematic illustrations of a surface mechanical attrition (SMA) treatment to create plastic deformation of a sample surface and ultimately a nanocrystalline surface structure. Vibration generator frequency (50Hz to 20KHz). Reproduced from Tao et al. (2002).](image)

According to Tao et al. (2002), the principal reason for nanocrystalline layer development was plastic deformation involving high strain and high strain rates. It seems
reasonable that the media size, mass and velocity are among the parameters that will determine the degree of strain and strain rates.

Almen strip curvature is related to the residual stress field and is an inexpensive and convenient measurement that is already widely used to characterize the impact power density of a shot peening treatment; i.e. the impact energy per unit time per unit workpiece surface area. Industry has long relied on the Almen system (SAE J442 standard) for shot peening characterization (Cao et al. 1995). The Almen strip test is performed as follows: (a) Almen strip plates are attached to the Almen block as shown in Fig. 1-2. (b) The Almen strip (a standardized piece of spring steel) is shot-peened for some time, and (c) the resulting curvature of the Almen strip is measured using the Almen gauge shown in Fig. 1-2 and is known as the Almen arc height.

![Image of Almen System](image)

**Fig. 1-2: The Almen System, Reproduced from Cao et al. (1995)**

The Almen arc height can be determined as a function of time by repeated execution of steps (b) and (c) with new Almen strips and varying exposure times. A typical curve of
this kind is shown in Fig. 1-3. After some time known as the saturation time, doubling the exposure time would only cause a 10% increase in arc height. The arc height at this time is known as the Almen Intensity (Fig. 1-3).

![Fig. 1-3: Almen intensity / saturation curve. Almen arc height vs Shot Peening time Cao *et al.* (1995).](image)

The Almen system has the advantage that it combines the effects of all the process parameters into one measurement, allowing for measurement and repeatability of the process. Previously, there was no way to characterize the impact power density in a vibratory finishing machine, and for the reasons stated above it was of interest to extend the Almen system to vibratory finishing.

The process parameters, in vibratory finishing, are aspects of the process that affect the media-workpiece interaction: the media’s size, shape and density; the coefficients of restitution and friction coefficients between pieces of media and between the media and the vibratory finisher’s container; the total media mass in the finisher; the frequency and amplitude of tub vibration; the dimensions and shape of the finisher; the location and orientation in the finisher, and the finishing time.
1.1. Literature Review

There are very few scholarly works on vibratory finishing (Baghbanan et al. 2003, Fraas et al. 1996, Hashimoto 1996, Sofronas et al. 1979, Wang et al. 2000, Yabuki et al. 2002). The papers by Baghbanan et al. 2003, Wang et al. 2000, and Yabuki et al. 2002 are primarily of an experimental nature and provide the basis for the modeling of present study. They involve experiments with two types of finishers, spherical media of like composition and varying size, and standardized size and shape workpieces (a cylinder with a height of 80 mm and a diameter of 68 mm). The papers by Fraas et al. (1996), Hashimoto (1996), and Sofronas et al. (1979) are not directly relevant to the primary aims of the present study (which are presented in section 1.2); however, they are described briefly to illustrate the state of current research attempts in vibratory finishing.

Sofronas et al. (1979) established an empirical model that describes the erosion of a sharp edge as a function of different finishing conditions. Fraas et al. (1996) discussed the requirements to achieve different flow fields in a vibratory finishing machine. Hashimoto (1996) performed experimental studies that revealed the optimal processing time for parts of varying initial surface roughness.

Wang et al. (2000) measured the distribution of normal contact force over time per unit surface area of a force sensor placed in a bowl-type vibratory finisher (Fig. 1-4); a typical force signal is shown in Fig. 1-5. In addition, Wang et al. (2000) showed how roughness and hardness of the workpiece change with time under the influence of varying process parameters: spherical media radius (7.1 mm, 8.6 mm, 11 mm), degree of lubrication (wet, detergent, or dry), finishing duration, and type of aluminium alloy.
Fig. 1-4: Bowl type vibratory finisher. Reproduced from Yabuki et al. (2002).

Fig. 1-5: Force signal produced by a resistive strain sensor with a 10mm contact area. Reproduced from Wang et al. (2000).

The force sensor was constructed using a clamped-clamped beam instrumented with resistive strain gauges arranged in such a way that only normal forces were measured (see Fig. 1-6).
The magnitude of the measured contact forces and impact durations depended on the stiffness of the sensor and workpiece and therefore did not produce an absolute measure of the magnitude of the contact force between media and a workpiece. Despite this limitation, this study revealed some interesting trends. The Fourier transform of the force signal showed a strong peak at the driving frequency of the finisher and two smaller peaks at the first and second harmonics. The hardness changes were most sensitive to the lubrication condition, then media roughness and finally media diameter. Hardness generally increased during the first forty minutes of finishing; however, in at least one case, a statistically significant (95% confidence level) decrease between forty and eighty minutes appeared. The 10 mm contact sensor disk (Fig. 1-6) was in contact with the media on average 30% of the time. This was verified by video observations that indicated that the media was relatively loosely packed during finishing with large mean free paths. Scanning electron micrographs of individual impact sites on a workpiece (AA1100-O) after 10 s of finishing showed evidence of normal impact in dry conditions (Fig. 1-7) and sliding in wet conditions (Fig. 1-8). Impact force trends did not appear to vary significantly with media size or lubrication condition.
Fig. 1-7: Scanning electron micrographs of individual impact sites on a workpiece (AA1100-O) after 10 seconds of finishing showing evidence of normal impact in dry conditions. Reproduced from Wang et al. (2000).

Fig. 1-8: Scanning electron micrographs of individual impact sites on a workpiece (AA1100-O) after 10 seconds of finishing showing evidence of sliding in wet conditions. Reproduced from Wang et al. (2000).

Yabuki et al. (2002) measured the magnitude of the normal force relative to the tangential forces in the bowl-type vibratory finisher (Fig. 1-4) using a multi-axial sensor made with 4 resistive force sensors. The sensor was able to simultaneously measure the normal and tangential forces from each contact of the media with a 3 mm sensor pad. The
combined reading from three of these sensors was used for measuring the forces applied in the tangential plane and an additional sensor was used to measure force along the surface-normal axis. A schematic of this sensor is shown in Fig. 1-9.

Fig. 1-9: Resistive force sensor. The red part is the 3mm sensor pad. Reproduced from Yabuki et al. (2002)

On average the normal forces were 10 times greater than the tangential forces in the wet and dry conditions. However, in the wet condition at least a small percentage of the media had a tangential to normal force ratio approximately equal to the measured dynamic coefficient of friction between media, indicating that media sliding relative to the workpiece happened in the wet condition but not in the dry condition. In addition, Yabuki et al. (2002) showed that a force signal was measured only 1/5 of the time over the 3 mm sensor pad. This was verified by video of media motion in the vicinity of a transparent plane surface. This provides more support for the idea that the mean free path of the media is quite large, as was first asserted by Wang et al. (2000).

Baghbanan et al. (2003) used the same sensor employed in the study by Yabuki et al. (2002) to measure contact forces in a larger more energetic tub finisher. In addition, changes in surface roughness, hardness, workpiece mass, and Almen strip curvature were measured when subjected to changes in lubrication, finishing duration, aluminium alloy, and finisher type. Baghbanan et al. (2003) found that for a given impact frequency, the maximum contact force was always greater in the dry condition than in the wet condition. In addition, for a given impact frequency, the maximum contact force was always higher in the tub than in the
bowl finisher irrespective of the lubrication condition. This observed behaviour is quantified and shown in graphical form in Fig. 1-10.

![Fig. 1-10: The relationship between the impact frequency and maximum force during impact for the bowl and tub finisher in both wet and dry conditions. Reproduced from Baghbanan et al. (2003)](image)

It was found that aluminium alloys AA1100-O and AA6061-T6 developed a surface that was largely (35-50%) covered with ceramic wear debris particles, but this was more pronounced with the softer alloy (AA1100-O). This ceramic surface layer can be seen in Fig. 1-11. Some larger particles were also found embedded in the near surface region, as can be seen in Fig. 1-12. The creation of this ceramic wear particle surface was also responsible for mass increases in the workpiece during finishing; however, this only occurred in dry conditions. If the occurrence of embedded media wear particles in a workpiece is common, this would probably influence the residual stress distribution. In order to avoid the complications associated with embedded debris, the present study was restricted to smooth steel media that do not release wear debris.
Baghbanan et al. (2003) measured the hardness of finished workpieces as a function of depth from the finished surface (Fig. 1-13). In all cases, the hardness had a maximum value within 20 $\mu m$ of the surface. In the case of long finishing times, the maximum was
below the surface, while for short finishing times, the maximum was at the surface. The hardness then decreased from its maximum value as the measurements progressed deeper into the workpiece.

**Fig. 1-13: Hardness changes with depth into workpiece. (a) AA1100-O 30min Dry, (b) AA1100-O 150min Wet, (c) AA6061-T6 150min Wet. Reproduced from Baghbanan *et al.* (2003)**

Almen strips made of AA1100-O, AA6061-T6 and Cu10100 were finished for various times in the bowl and the tub finishers. Almen strip curvature increased consistently as function of time (Fig. 1-14 and Fig. 1-15). At any given time, Almen strips finished in the tub had more curvature than those finished in the bowl since the tub was more energetic.
Fig. 1-14: Almen strip curvature vs. finishing time for AA1100-O, AA6061-T6 and Cu10100 in the bowl finisher under two conditions: (BD) bowl dry; (BW) bowl wet. Reproduced from Baghbanan et al. (2003)

Fig. 1-15: Almen strip curvature vs. finishing time for AA1100-O, AA6061-T6 and Cu10100 in the tub finisher under two conditions: (TD) tub dry; (TW) tub wet. Reproduced from Baghbanan et al. (2003)
It was unexpected that the curvatures were always greater in the wet condition than the dry condition, since it had been shown that the dry condition corresponded to higher normal contact forces. Baghbanan et al. (2003) explained this as being due to a difference in the Almen strip boundary condition in the wet condition where it is possible for water to get between the Almen strip and the holding fixture. It was hypothesized that this would cause an increase in stiffness and thus an increase in contact forces relative to the dry condition, which would lead to increased plastic deformation and in turn Almen strip curvature. This was verified by an experiment where a layer of grease was sandwiched between the Almen strip and the Almen strip holding fixture and was finished in the dry and wet condition. This was compared to finishing without the grease in the dry and wet conditions. A graphical representation of the results of these experiments has been reproduced here in Fig. 1-16.

![Graph](image)

**Fig. 1-16:** Almen strip curvature vs. finishing time for AA1100-O in the tub finisher under four conditions: (TD) tub dry; (TDG) tub dry with a grease film behind the strip; (TW) tub wet; (TWG) tub wet with grease film. Reproduced from Baghbanan et al. (2003)
The curvatures in the wet condition were still the greatest; however, the application of grease in the dry condition greatly increased the curvature, confirming the aforementioned hypothesis. Baghbanan et al. (2003) recognized the need to provide a consistent boundary condition during finishing and suggested using a hot melt adhesive which could be removed by the application of low heat following the finishing operation to attach the Almen strips to the holding fixture. Baghbanan et al. (2003) did not explain why the greased-dry condition curvature was less than either of the wet condition curvatures. This suggests that a factor other than the change in boundary condition contributed to the reduced curvature in the dry condition.

Baghbanan et al. (2003) also made measurements of the speed of the media in the flow by installing a video camera into a hollow cylindrical workpiece and letting it move freely with the flow. The displacements of media flowing along the end-face of the workpiece were recorded between frames taken 1/30 seconds apart. The velocity obtained in dry conditions was $0.8 \pm 0.4$ cm/s, and $0.6 \pm 0.4$ cm/s in wet conditions ($\pm$1S.D., N=10).

Although there are no theoretical papers that discuss the modeling of the vibratory finishing process, some shot peening literature is applicable to this problem. The principal differences between vibratory finishing and shot peening relevant to a modeling effort are that shot peening media usually has a much higher velocity, smaller mass and smaller size than vibratory finishing media. However, this does not limit the application of these shot peening models to vibratory finishing, since the impact velocity and shot mass and size are always among the model input variables. In addition, the simplifying assumptions of these analyses are also applicable to vibratory finishing. Indeed, in some cases, the simplifying assumptions used in shot-peening models are more suited to vibratory finishing; for instance that the target loading is quasi-static is a better assumption in vibratory finishing, since shot velocities in shot peening are about two orders of magnitude greater than in vibratory finishing, making strain rate and other dynamic effects much more pronounced.

Two analytical shot peening models were considered for use in vibratory finishing; those of Al-Obaid 1990 and Hill et al. 1982. They provide predictions for a workpiece in its completely finished state. Changes to the workpiece over time were not considered. This was reasonable, because they assumed that these changes diminish with time and eventually stop. This state, known as saturation, is consistently observed in shot peening and vibratory
finishing (Baghbanan et al. 2003, Wang et al. 2000, Yabuki et al. 2002). These analyses generally examine the residual stress field along the centerline of an impact site, parallel to the normal axis produced of a single shot impact, and make the assumption that a workpiece has this same residual stress throughout the target material.

The model by Al-Obaid (1990) provides the residual stress field of a shot peened surface and requires several inputs: shot incident velocity ($V$), shot density ($\rho$), shot radius ($R$), Almen strip arc height ($\delta$), the depth into the target where the maximum stress exists ($\alpha$) as a fraction of plastic zone depth ($h_p$). This model neglects strain hardening, inertial body forces, friction and tangential velocity components. Unfortunately, Al-Obaid (1990) doesn’t suggest a method for determining $\alpha$, severely limiting the applicability of the model and its usefulness in the present study.

The model by Hill et al. (1982) employed all the simplifying assumptions of the Al-Obaid (1990) model. The inputs required to predict the residual stress distribution in an Almen strip are the shot diameter, shot density, shot velocity, shot impact velocity, Young’s modulus and Poisson ratio of the shot and target. Predictions of Almen strip curvature also require the thickness of the Almen strip ($t$), and the yield stress of the Almen strip in pure shear ($k$).

Guagliano (2001) performed a finite element analysis for the purpose of predicting the residual stress field in a metal workpiece induced by the shot peening process. This analysis showed the evolution of stresses in the target material due to multiple impacts. Each impact was made to occur at a different location on the surface of the finite element target, following a symmetrical pattern. The FE approach used an explicit integration scheme with a lumped matrix. The explicit solution method is more computationally efficient than implicit methods. The required time increment was proportional to the dimension of the smallest finite element and was chosen without difficulty. The solver was ABAQUS; however, in order to model kinematic work hardening, a FORTRAN user subroutine was developed by Guagliano (2001). The shot was modeled as a rigid body since preliminary experiments showed that an elastic shot produced similar results. The shot speed range considered in this study was from 20 to 110 m/s. The strain rate sensitivity of the material was neglected. It was found that the depth of the residual stress profile was directly proportional to the size and velocity of the shot used for a given Almen strip and shot material. In addition, empirical
equations that give Almen strip intensity as a function of shot velocity for a given Almen strip were derived for a series of shot sizes and shot materials.

Woytowitz and Richman (1999) performed an FEA of elastic spheres impacting normally on the surface of a kinematic hardening, elastic-plastic material for the purpose of modeling erosion. Although erosion was not of primary interest in the present study, this work does illustrate some FE techniques applicable to the present study. The use of stochastically located impacts sites with one sphere in contact with the material at a time is an example of such a technique. This approach was not taken by Guagliano (2001), because symmetry was used to reduce the computational requirements of the FEA.

Experiments have been performed that attempt to explain how a compressive surface layer of residual stress is created (Kobayashi et al. 1998). Large (5 cm) steel balls were dropped at 6.7 m/s onto thick, polished, annealed steel plates to simulate the shot-peening process. The first graph in Fig. 1-17 shows the plate, the impact site on the plate, and the distribution of stress on the plate surface surrounding the impact site as measured by X-ray diffraction. As the number of impacts increases, the tensile stress on the initial impact site and the surrounding area becomes consistently more compressive. The spheres impact the target and plastically deform its surface into a state of tension; however, the surrounding elastically deformed material compresses the surface layer and creates the layer of compressive residual stress (de los Rios 1999). Shot peening of the whole surface is known to lead to a residual stress distribution that varies with the depth, but is nearly uniform in the plane direction of the strip (Schiffner et al. 1999).
This experiment gives much insight into the creation of residual stresses by the shot peening process. It shows that there is a danger of introducing tensile stresses in the material if the distribution and possibly the nature of impacts are not correct; this would be harmful to fatigue performance. Residual tensile stress has the opposite effect of compressive stress; that is, it promotes fatigue failure. Although the experiment has the limitation of showing only the evolution of stress in the uppermost region of the surface layer, it illustrates the effect created by the cumulative affect of each impact in a multiple impact scenario.
1.2. Thesis Objectives

The design of vibratory finishing machines and processes has been the subject of relatively few scientific publications. Scientific studies of the mechanics that describe the effect of vibratory finishing on a workpiece would provide information that might result in improved vibratory finishers and process control. The focus of the present study was to make progress towards this goal.

Specifically the objectives of the present study were: (i) to measure the normal impact forces in a vibratory finisher and use these data to infer the normal impact velocity; (ii) to model the development of residual stress fields and curvature in workpieces induced by the VF process from a knowledge of the media-workpiece interaction and the VF process parameters, (iii) to determine the feasibility of using Almen saturation curves as a means of characterizing the VF process.

In order to meet the objectives it was be necessary to (a) determine the mechanisms of media-workpiece interaction that lead to the creation of residual stresses (b) to develop a method analogous to the Almen system for finishing Almen strips, measuring arc heights and obtaining saturation curves.

The model described in objective (ii) will require the normal impact velocity distribution and the average impact flux. The output of this model provided details of the mechanics of the process and the effect of process parameters useful for process design and optimization.

Objective (iii) may enable the use of Almen strips as inexpensive and simple probes capable of revealing the normal impact velocity distribution and the average impact flux which are relevant to inducing residual stress in a workpiece.

1.3. Structure of Thesis

The thesis consists of three main parts describing respectively, the measurement of media impact velocities (Ch. 2), the application of the Almen system to vibratory finishing (Ch. 3), and the modeling of Almen strip curvature development in vibratory finishing (Ch. 4). Impact velocities were recorded at the same locations in the vibratory finisher where Almen strips were placed. The measured velocity distributions were then used in the model to make predictions of Almen strip curvature as a function of exposure time in the vibratory
finisher. These model predictions were then compared with the experimental curvatures to guide model development.
2. Media Impact Velocity Sensor

Equation Chapter 2 Section 1

2.1. Nomenclature

\( v \)  Normal impact (or incident) velocity of a piece of media
\( h \)  Media drop height
\( g \)  Acceleration due to gravity, \( |g| \approx 9.81 m/s^2 \)
\( y_1 \)  Distance between bottom edge of media and impact cap in the first exposure.
\( y_2 \)  Distance between bottom edge of media and impact cap in the final exposure.
\( \Delta t \)  Time elapsed between first exposure and last exposure in Fig. 2-4
\( F(t) \)  Force signal as recorded by the sensor
\( m \)  Mass of a single piece of media
\( e \)  Coefficient of restitution in the normal direction
\( k \)  Stiffness of a contact.
\( F_{\text{max}} \)  Maximum force within a contact
\( v_{\text{max}} \)  Maximum impact velocity (for impact contacts) or effective impact velocity (for non-impact contacts) found in a given signal
\( v_{\text{avg}} \)  Average impact velocity or effective impact velocity of a given signal
\( \delta \)  Sensor deflection during a contact
\( \delta_{\text{max}} \)  Maximum sensor deflection during a contact
\( v_{\text{eff}} \)  Effective normal impact velocity of a non-impact contact
\( R \)  Correlation coefficient
\( a \)  Slope of linear fit for \( e = f(v) \)
\( b \)  \( e \) intercept of linear fit for \( e = f(v) \)
\( v_{t_2}^{\text{media}} \)  Normal rebound velocity of a piece of media where impact ends at a time equal to \( t_2 \)
\( v_{t_2}^{\text{workpiece}} \)  Normal rebound velocity of the workpiece (or sensor) following the impact at \( t_2 \)

---

\(^2\) This section was published in *J. of Materials Processing Technology*, 183, Ciampini D., Papini M., Spelt J.K, Impact velocity measurement of media in a vibratory finisher, 347-357, Copyright Elsevier (2007).
\( v_{t_1}^{\text{media}} \) Normal impact (or incident) velocity of a piece of media where impact occurs at \( t_1 \).

\( v_{t_1}^{\text{workpiece}} \) Normal impact (or incident) velocity of the workpiece (or sensor) at the onset of the impact that occurs \( t_1 \).

\( h_1 \) Distance from bottom edge of media the sensor surface.

\( h_2 \) Media rebound height from sensor surface.

\( d \) Media diameter.

\( \Delta T \) Contact period.

\( C_n \) Coverage of impact sites on a surface being finished after contact \( n \).

\( n \) Number of times that a contact with \( A_r \) has occurred on a surface.

\( A_r \) Ratio of media contact area divided by the area of surface being finished.

\( f \) Average impact frequency during finishing with a system of fixed \( A_r \).

\( T \) Finishing duration.

\( c \) Radius of contact area between spherical media and a flat surface.

\( s \) Radius of force sensor impact cap.

\( R \) Media radius.

\( v \) (Effective) Impact velocity of media.

\( \rho_1 \) Media density.

\( E_1 \) Young’s modulus of media.

\( \mu_1 \) Media Poisson ratio.

\( E_2 \) Young’s modulus of target.

\( \mu_2 \) Poisson ratio of finished surface.

HW High wall of finisher (Fig. 2-8).

LW Low wall of finisher (Fig. 2-8).

SW Side wall finisher (Fig. 2-8).
2.2. Introduction to the Measurement of Impact Velocities in Vibratory Finishers

Wang et al. (2000) measured the distribution of normal contact force over time per unit surface area of a force sensor placed in a bowl-type vibratory finisher. The force sensor was constructed using a clamped-clamped beam instrumented with resistive strain gauges arranged in a manner such that only normal forces were measured. It was recognized that the magnitude of the measured contact forces and impact durations depended on the stiffness of the sensor. The Fourier transform of the force signal showed that most of the energy occurred at the driving frequency of the finisher with lesser amounts at the second and third harmonics. It was also found that the 10 mm force sensor disk had media of similar diameter passing over it approximately 30% of the time, indicating that the media was packed relatively loosely and had large mean free paths during finishing. This was confirmed using a submerged video camera. Scanning electron micrographs of individual impact sites on an aluminium workpiece (AA1100-O) that had been finished for 10 s showed evidence of normal impact in dry conditions and sliding in wet conditions. Finally, impact force trends did not appear to vary significantly with media size or lubrication condition.

Yabuki et al. (2002) measured the magnitude of the normal force relative to the tangential forces in the same bowl-type vibratory finisher employed by Wang et al. (2000). Using a 3 mm diameter “floating” pad connected to four resistive force sensors, it was found that forces normal to the sensor were approximately 10 times greater than the tangential forces in both the wet and dry conditions. In the wet condition, a small percentage of the media had a tangential to normal force ratio approximately equal to the measured dynamic coefficient of friction between media, indicating that sliding had occurred. The force signals also confirmed the relatively loose packing of the media in the bowl finisher. Yabuki et al. (2002) hypothesized that the force signals corresponded to three distinct modes of contact: single particle impact, a single particle rolling across the sensor, and a single particle being pressed into the sensor by surrounding media. As was the case with Wang et al. (2000), the direct use of impact forces is of limited generality, because they are affected by surface compliance.

Baghbanan et al. (2003) used the same sensor employed in the study by Yabuki et al. (2002) to measure contact forces in a larger, more energetic tub finisher and to study their
relation to the erosion and work hardening of aluminium. They found that the contact forces were generally greater in the dry condition than in the wet condition when using water as a media lubricant.

In addition to the literature related to impact forces and their role in vibratory finishing, several studies have focused on measuring and modeling erosion and surface roughness changes as a function of various process parameters (Domblesky et al. 2003, Domblesky et al. 2004, Hashimoto 1996).

Although these earlier studies provided a better understanding of the mechanics of vibratory finishing, as mentioned previously, the direct use of impact force to characterize the conditions within a vibratory finisher is of limited use, because its magnitude in a given finisher is a function of the sensor or surface compliance. Therefore, the impact forces measured with a given sensor will not be those acting on an arbitrary part within the finisher. The present work presents a more general characterization of the impact conditions within a vibratory finisher by developing a means of extracting the normal impact velocity from the measured impact force signals. The normal component of the impact velocity is independent of the workpiece and sensor, and provides a useful measure of the energy imparted to a generic workpiece by a vibratory finisher. As will be demonstrated later, the impact velocity probability distribution depends strongly on the location within the finisher and on its dynamic conditions, and can be used as an input for subsequent modeling of the erosion or plastic deformation of an arbitrary workpiece surface, regardless of its compliance.

2.3. Apparatus

2.3.1. Force Sensor and Housing

A piezoelectric force sensor (Dytran Instruments Inc., Chatsworth, CA, model 1051V1) was used to measure normal forces. The factory calibration of the sensor was verified by putting a known mass on the sensor and removing it quickly, as illustrated in Fig. 2-1. Although the signal generated by a constant load falls to zero, the time constant of the decay is relatively long (i.e. several seconds) compared with the media contact times experienced in a vibratory finisher (10 $\mu$s to 10 ms); therefore, the sensor is suitable for media contact measurements.
Sensor signals were recorded using an analog-to-digital converter (National Instruments, Austin, TX PCI-6071E), capable of sampling the output signal with 12-bit resolution every \( \mu s \); a sampling frequency that is at least 10 times less than the shortest impact duration measured in the finisher. All electrical cables were shielded to minimize the electrical noise due to the finisher motor. Utilizing the NI-DAQ v7 driver (National Instruments Corporation, Austin, TX, USA), a “C language” computer program was written to save all recorded data to a file. All subsequent data analysis (see Section 2.4) was performed using Matlab v7 (The MathWorks, Inc., Natick, MA, USA).

Fig. 2-1: Sensor force output as a function of time after sudden removal of a 1.045 kg calibration mass.

The force sensor could be positioned at various locations and orientations within the finisher (Fig. 2-2). In all cases, the 4 mm diameter AA6061-T6 aluminium impact cap was the only part of the sensor to contact the media through a hole in the housing assembly (Fig. 2-2). The gap between the impact cap and the hole in the housing assembly was
approximately 0.25 mm. The sensor mount assembly and the housing assembly were electrical conductors mounted on a wall which was non-conductive. Therefore, the electrical resistance between the assemblies indicated if the gap around the impact cap was maintained. A simple circuit was employed to exploit this; it ensured that the impact cap did not contact the housing assembly.

Fig. 2-2: Force sensor and housing schematic. Not to scale.

With the sensor cap covered, and the housing submerged in the finisher, it was found that the background vibration reaching the sensor through the wall together with residual electrical noise resulted in a background signal of less than 0.3 N. Typical contacts had force maxima in the range of 1 N to 25 N; therefore, the signal is significantly above the noise floor.

The natural frequency of the sensor mount assembly was approximated using a single degree of freedom lumped mass model. It was found that the natural frequency of this assembly was 35 Hz. This is sufficiently below the driving frequency (47 Hz) of the vibratory finisher, and therefore avoids a resonance problem.
2.3.2. Vibratory Finisher and Media

The present measurements were conducted in the same tub finisher used in Baghbanan et al. (2003), (Burr Bench 2016, Brandon Industries, TX). The tub had a U-shaped cross-section with a urethane chamber liner and inner dimensions of 400 mm (x-direction) × 210 mm (y-direction) × 400 mm (z-direction) (Fig. 2-3).

Fig. 2-3: Cross-section of the vibratory finisher (reproduced from [3]) showing direction of media circulation and resulting slope of the free surface. All of the dimensions are given in mm.

Two types of media were used in the present study: 6 mm porcelain balls (Microbrite™, Abrasive Finishing Inc., Chelsea, Michigan), and 6.35 mm diameter carbon steel balls (ABCO, Abbott Ball Company Inc., West Hartford, Connecticut). Microbrite™ and steel media are designed for polishing and burnishing applications. Table 2-1 gives the average and standard deviation of the diameter and mass for each of these media.
Table 2-1: Average media diameter and mass. Ten measurements (± indicates three standard deviations)

<table>
<thead>
<tr>
<th>Material</th>
<th>Average Diameter (mm)</th>
<th>Average Mass (g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Porcelain</td>
<td>6.1 ± 0.54</td>
<td>0.29 ± 0.06</td>
</tr>
<tr>
<td>Steel</td>
<td>6.3 ± 0.015</td>
<td>1.03 ± 0.006</td>
</tr>
</tbody>
</table>

2.3.3. Impact Velocity from Force Measurements in Ball Drop Experiments

The relationship between the impact velocity and the force signal was determined by dropping porcelain and steel spheres onto the sensor cap from known heights. The left side of Fig. 2-4 shows a solenoid actuator with a small tube steel used to grip the sphere, while the right side has a similar steel tube that makes contact with the ball a small distance above the left one, and is angled slightly downward so it does not interfere with the ball once the solenoid is actuated.

Fig. 2-4: (a) Solenoid actuator holding steel sphere a known height above impact force sensor. (b) Six superimposed images of a sphere falling (down) towards the sensor cap (y=0) after being released. \(y_1\) and \(y_2\) are the distances from the sensor to the bottom of the ball for the first and last exposures, respectively.

The impact velocity, \(v\), was determined from the drop height, \(h\), by assuming that the drag force was negligible at these low impact speeds so that,
\[ v = \sqrt{2gh} \]  

Fig. 2-5 shows typical measured normal impact force signals and their variability for porcelain and steel media at two impact speeds; i.e. 0.36 m/s and 0.50 m/s. As expected, the steel media generated much larger maximum impact forces and impulses than did the porcelain media, because of their greater stiffness and mass.

Fig. 2-6 and Fig. 2-7 show the experimental correlations between the impact velocity and the impulse or maximum impact force, respectively, for the two media. Interestingly, a linear correlation provides a good fit in both cases, although the impulse regression coefficients are slightly larger than the force regression coefficients indicating a better model.

Fig. 2-5: Superimposed force signals from drop experiments with steel and porcelain spheres at impact velocities of 0.36 and 0.50 m/s. The upper and lower curves encompass the range of measured values in ten repetitions. The central curve represents the mean of all the ten repetitions.
Fig. 2-6: Sensor impulse recorded for steel and porcelain media impacts with known impact velocities.

Fig. 2-7: Maximum impact force recorded by the sensor for steel and porcelain media impacts with known impact velocities.
Previous research (Baghbanan et al. 2003, Wang et al. 2000, and Yabuki et al. 2002) suggests that the vast majority of the energy transferred to the workpiece by spherical media in the finisher is associated with impacts normal to the surface. The response of the present sensor to oblique impacts was determined by dropping balls at two angles of attack: 8° from the normal, representing what was thought to be a typical impact angle, and 45°, representing an unlikely extreme angle. The normal component of the drop velocity was compared to the normal impact velocity predicted by the impulse vs. normal impact velocity linear fit given in Fig. 2-6. At 8° from the normal, the average prediction of the normal velocity component was 6% less than the actual value (largest error was 15%), which was comparable to that observed for impacts at 90°. At 45° the average error was 19% less than the actual normal velocity (largest error was 22%). It was therefore, concluded that the sensor could accurately measure normal velocity components for oblique impacts typical of those occurring in the vibratory finisher.

2.4. Results and Discussion

2.4.1. Interpretation of Impact Forces

2.4.1.1. Bursts

Fig. 2-8 shows a plan view of the four locations at which impact forces were measured and the orientation of the sensor in each case.

Fig. 2-8: Plan view of tub showing four sensor locations: a, b, c and d. Arrows indicate orientation of sensor. High wall (HW) and low wall (LW) as shown in Fig. 2-3. Sensor depth was 210 mm above bottom of tub in all cases.
Fig. 2-9 (a) shows a typical force signal recorded over 5 s when the sensor was submerged in the vibratory finisher at location ‘a’ (shown in Fig. 2-8). Two contact types were identified in the force signals: impact and non-impact contacts (see Section 3.2), shown in different shades. As observed in Wang et al. (2000) and Yabuki et al. (2002) there are relatively long periods during which there is no contact between the media and sensor. Fig. 2-9(b) shows an expanded view of the signal between 0.8 and 1.5 s demonstrating that the contacts occur in bursts, generally every 21 ms, corresponding to the period of the finisher vibration at 47 Hz.

Yabuki et al. (2002) observed a similar trend of periodic signal bursts with a bowl finisher, and plotted all the contacts in each burst superimposed on top of each other in an effort to observe trends pertaining to how the force signal is distributed within each burst. He found that, within a burst, the envelope created by the superposition of forces associated with contacts increased monotonically from zero to a maximum, and then decreased back to zero at approximately the same rate as the increase. The rise and fall of the impact force coincides with the movement of the media toward and away from the sensor. Fig. 2-10 shows that the overlapping bursts obtained from the present measurements are very similar to those of Yabuki et al. (2002), in spite of the differences in the media, finisher, and impact force sensor. Force bursts were, however, not recorded every 21 ms (the tub period). In the 15 s force measurement using 41 kg of porcelain media at location ‘a’ (Fig. 2-8), impact forces were absent in 24% of the tub periods. The comparable figure with 91 kg of steel media was 18%. This is consistent with the observation of Wang et al. (2000) and Yabuki et al. (2002) that the media packing could be relatively loose as it flowed past surfaces in a vibratory finisher, with gaps similar to bubbles in a liquid. Wang et al. (2000) reported that there were media passing over a sensor disk only about 30% of the time; however, this may not be directly comparable with the present data since the finisher was of the bowl type, the impact force sensor had a much lower resolution and the sensor disk was larger relative to the media diameter (approximately equal diameters).
Fig. 2-9: (a) Typical signal recorded over 5 s when the sensor is submerged in the vibratory finisher. The portion of the signal between 0.8 and 1.5 s is expanded in (b). Sensor located at ‘a’ (Fig. 2-8), 91 kg of steel media.
2.4.1.2. Classification of Contact Events within a Burst

Yabuki et al. (2002) observed that a single burst consisted of several types of contact, differing mainly in the duration of the force signal. A similar behaviour was found in the present force data. Fig. 2-11 shows an expanded view of the burst consisting of a single “impact contact” at 1.38 s in Fig. 2-9(b). Such impacts were characterized by a short contact duration (0.05 ms) and a characteristic vibration or “ringing” that followed the impact with a period of about 10 ms, governed by the stiffness of the sensor. This ringing did not usually interfere with the recording of subsequent impacts, since the decay was complete within half of the finisher period of 21 ms.
Fig. 2-11: Expansion of the impact contact at approximately 1.38 s lasting 50\(\mu\)s in Fig. 2-9(b). ●Impact portion of the signal, × sensor oscillations (i.e. ringing) following the impact.

So called “non-impact contacts” (Fig. 2-12) were characterized by a much longer contact duration (0.8 ms), which was, however, always less than half the period of tub oscillation during which the media are moving toward the sensor or tub wall. Contacts of this type are believed to be due to a single piece of media being pressed into the sensor by surrounding media (Fig. 2-13), consistent with the observations of Yabuki et al. (2002).
Fig. 2-12: Expansion of a single non-impact contact (duration=0.8 ms) at approximately 1.46 s in Fig. 2-9(b).

Yabuki et al. (2002) proposed that the small individual peaks in the non-impact contact of Fig. 2-12 were the result of rough media rolling over the sensor. Fig. 2-14 shows the frequency distribution of the peak-to-peak intervals obtained from a 15 s signal at locations ‘a’ with steel and porcelain media.
The shape of the distribution using steel media in the present work matches that of a similar histogram in Yabuki et al. (2002), obtained using 9 mm diameter ceramic media; however, the frequency of occurrence of the shorter peak-peak intervals is much greater in the present case (130 per second compared to 3.5 per second in Yabuki et al. (2002)). In other words, the present sensor is detecting many more closely spaced peaks during non-impact contacts. The porcelain media histogram of Fig. 2-14 has a maximum frequency of occurrence for a peak-to-peak interval of approximately 12 ms, which is similar to that found in Yabuki et al. (2002). As mentioned previously, Yabuki et al. (2002) hypothesized that the peaks were due to the roughness of the media as it rolled across the sensor. In the present study, the steel media were much smoother than the porcelain media, and so it is expected that the steel media distribution would have a narrower distribution of peak-to-peak intervals, with a tendency toward smaller intervals compared with the porcelain distribution. This is
indeed consistent with the trends seen in Fig. 2-14. It should be noted, however, that the small local peaks in non-impact contacts (Fig. 2-12) may also be due to collisions amongst adjacent media, and that the tendency to a high frequency of occurrence of very small peak-to-peak intervals may be an artefact of ringing at the sensor’s natural frequency (period of 50 μs).

2.4.2. **Media Impact Velocity**

2.4.2.1. **Impact and Effective Impact Velocity**

In order to compare the effects of impact and non-impact contacts (Fig. 2-13) it was assumed that the stresses and plastic deformation produced by both types of contact could be modelled as if the maximum contact force were applied quasi-statically. This is justified because the impact speeds measured in the vibratory finisher were usually less than 0.5 m/s (see Table 2-2). Therefore, the extent of plastic deformation produced by impact and non-impact contacts was considered to be equivalent if the maximum force was the same in each case.

Using the relationship between maximum force and impact velocity provided in Fig. 2-7, the maximum forces generated by non-impact contact events were expressed in terms of an “effective” impact velocity. Expressing all the forces generated in non-impact contacts by their equivalent effective impact velocities allows for a more compact method of representing and comparing any of the contacts that are observed in the signal, and provides a measure of finishing potential that is independent of the sensor characteristics. In the figures and discussion below, the term, “effective impact velocity” refers to the velocity for both impact and non-impact contact events.

2.4.2.2. **Effect of Sampled Signal Duration**

It was found that the frequency distribution of the impact force in the finisher was a function of the signal duration, with relatively infrequent high-speed impacts appearing only at sufficiently long sampling periods. This is illustrated in Fig. 2-15 which shows five effective impact velocity probability distributions, calculated as the number of impact and
non-impact contacts within a certain range of effective impact velocities divided by the total number of contacts occurring in signal durations from 60 s to 300 s. Fig. 2-15(a) shows that the probability of the frequent low velocity impacts does not vary significantly with sampling duration, while Fig. 2-15(b) illustrates that the more infrequent high velocity impacts are observed only in the longer sampling periods.
Infrequent, high velocity impacts may be more important in the mechanics of the vibratory finishing process than the more frequent low velocity impacts, as they cause more plastic deformation and hardening of the impact site. The very low velocity impacts may be important or unimportant depending on the application. For example, drying with an absorptive media does not require an energetic impact since only contact is required, whereas burnishing depends on higher energy contacts to create plastic deformation.

The most extreme high velocity impacts that occur very infrequently are most likely to have little effect, since they are so rare that a sufficient surface coverage to affect the workpiece would not be achieved in a practical finishing time. The average coverage \( C_n \) after \( n \) impacts at a given velocity, each producing the same fractional coverage \( A_r \) (impact area divided by target area), is given by

\[
C_n = 1 - \left(1 - A_r\right)^n
\]  

(2-2)
with $C_n = 1$ and $C_n = 0$ corresponding to, full and zero coverage, respectively. The accuracy with which Eq. (2-2) represents the average of this stochastic process increases as $A_r$ decreases. The average frequency of occurrence, $f$, of an impact at a particular velocity is related to the finishing duration, $T$, during which the $n$ impacts occur, by,

$$f = n/T$$

(2-3)

Hence, Eq. (2-2) yields,

$$T = \frac{\ln(1-C_n)}{f \ln(1-A_r)}$$

(2-4)

In the present work, the media-target Hertzian contact area was used to calculate $A_r$, and, since the diameter of the force sensor button ($s$) only allows one piece of spherical media to be in contact at any time,

$$A_r = \frac{c^2}{s^2}$$

(2-5)

where $c$ is the radius of the contact area, given by Hill et al. (1983),

$$c = R \left( \frac{1}{4} v^2 \rho_i \pi^2 \left( \frac{1-\mu_1^2}{\pi E_1} + \frac{1-\mu_2^2}{\pi E_2} \right)^{\frac{1}{5}} \right)$$

(2-6)

Here $v$ and $\rho_i$ are the impact velocity and density of the media of radius $R$, while $E$ is Young’s modulus and $\mu$ is the Poisson ratio of the media (1) and target (2). Equations (2-5) and (2-6) can be combined with Eq. (2-4) to yield the finishing duration, $T$, required to reach a chosen level of coverage, $C_n$, for a particular effective impact velocity, $v$. For example, if $C_n = 0.95$ is applied to the effective impact velocity distribution obtained from position ‘a’ (Fig. 2-8) with 91 kg of steel media,

Fig. 2-16 shows that impact velocities greater than approximately 0.2 m/s, being very infrequent (see Fig. 2-17) would reach 95% coverage only after more than 70 h of finishing. Therefore, in practical circumstances where the finishing time is likely to be much less, these infrequent impacts may not have the coverage to affect the workpiece significantly.
Fig. 2-16: Finishing duration to reach 95% surface coverage for impact velocities found at position ‘a’ (Fig. 2-8) 91 kg of steel media and a 5 min force measurement sampling duration. Each bar represents the finishing time for a range of impact velocities spanning 0.008 m/s.

2.4.2.3. Effect of Sensor Location and Tub Amplitude

During vibratory finishing, a workpiece flows within the circulating, vibrationally fluidized bed of media. In order to measure the range of impact velocities occurring in one cycle within the tub, the impact sensor was positioned at four different locations, as shown in Fig. 2-8. In all cases, the sensor was located 220 mm from the bottom of the tub and was oriented toward the nearest wall. The sensor was submerged approximately 10 cm (location ‘d’) to 25 cm (location ‘a’) below the media surface. The measurements were repeated for the following combinations of media type and mass in order to generate different finisher vibration amplitudes and hence, impact conditions: 41 kg of porcelain media; 91 and 112 kg of steel media. Increasing the media mass had the effect of reducing the amplitude of vibration without affecting the frequency, because the eccentrically weighted shaft that generated the vibration was supported by ball bearings attached to the tub. Table 2-2
summarizes the effects of amplitude and location in terms of the average \( v_{\text{avg}} \) and maximum \( v_{\text{max}} \) effective impact velocity, average contact frequency \( f_{\text{avg}} \) total energy flux \( E_{\text{Total}} \), and the percentage of both the signal duration and the total number of contacts due to each contact type. Table 2-2 also indicates the type of contact that gave \( v_{\text{max}}, f_{\text{avg}} \) was calculated by dividing the number of contacts found in a force signal by the signal duration \( T \). \( E_{\text{Total}} \) was evaluated by summing the effective kinetic energy of each contact found in a force signal and dividing by the sensor contact area, \( A_{\text{sensor}} \), and the signal duration, \( T \); i.e.

\[
E_{\text{Total}} = \frac{m_{\text{avg}} \sum_{i=1}^{N} v_i^2}{2 \cdot T \cdot A_{\text{sensor}}}
\]

where, \( N \) is the total number of contacts (of either type) occurring during \( T \), and \( m_{\text{avg}} \) and \( v_i \) are the average media mass and effective impact velocity of the \( i^{th} \) contact, respectively.

The average effective impact velocity is given by,

\[
v_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} v_i
\]

The average contact frequency is given by,

\[
f_{\text{avg}} = \frac{N}{T}
\]

The average contact frequency for each condition varies significantly from the driving frequency of the tub, although the power spectrum of the signals showed that most energy occurred at the driving frequency of the tub (47 Hz) and higher harmonics.

Fig. 2-17 shows the frequency of the effective impact velocity per unit area at sensor locations ‘a’ and ‘b’ (44 mm and 95 mm from the high wall (HW), respectively) for 41 kg of porcelain, and 91 kg and 112 kg of steel media. For the porcelain and 91 kg of steel, much higher effective impact velocities were recorded at location ‘a’, which was closer to the wall and hence the source of the media motion. For the 112 kg of steel media, location ‘b’ gave slightly greater impact frequency per unit sensor area over some of the effective impact velocity range than did location ‘a’. However, Table 2-2 shows the location ‘a’ measurements always have a higher \( E_{\text{Total}}, v_{\text{max}}, v_{\text{avg}} \) and \( f_{\text{avg}} \) than location ‘b’ measurements.
when all other factors are equal, with the exception that \( v_{\text{avg}} \) values are identical in one case.

This suggests that the finishing becomes significantly more aggressive as the part moves closer to a wall.

Table 2-2: Analysis of force signals recorded at locations of Fig. 2-8 with 41 kg of porcelain media (41P), and either 91 kg (91S) or 112 kg (112S) of steel media. average contact frequency, \( f_{\text{avg}} \), average effective impact velocity, average, \( v_{\text{avg}} \), maximum effective impact velocity, \( v_{\max} \), with corresponding impact type, total energy flux, \( E_{\text{Total}} \), percentages of time and number of contacts for each type of contact. Data for positions ‘a’ and ‘b’ were derived from a 300 s force signal, ‘c’ and ‘d’ from a 150 s signal.

<table>
<thead>
<tr>
<th>Condition</th>
<th>( f_{\text{avg}} ) (Hz)</th>
<th>( v_{\text{avg}} ) (m/s)</th>
<th>( v_{\max} ) (m/s)</th>
<th>( E_{\text{Total}} ) (W/m²)</th>
<th>% Total Non-Impact Contact Time</th>
<th>% Total Impact Contact Time</th>
<th>% Total Non-Contact Time</th>
<th>% Occurrence of Non-Impact Contacts</th>
<th>% Occurrence of Impact Contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>91Sa</td>
<td>462</td>
<td>0.014</td>
<td>0.378</td>
<td>Impact</td>
<td>12.8</td>
<td>6.02</td>
<td>0.090</td>
<td>93.8</td>
<td>94.6</td>
</tr>
<tr>
<td>91Sb</td>
<td>269</td>
<td>0.012</td>
<td>0.237</td>
<td>Impact</td>
<td>4.31</td>
<td>7.19</td>
<td>0.035</td>
<td>92.7</td>
<td>96.5</td>
</tr>
<tr>
<td>112Sa</td>
<td>351</td>
<td>0.011</td>
<td>0.179</td>
<td>Impact</td>
<td>3.82</td>
<td>13.4</td>
<td>0.025</td>
<td>86.4</td>
<td>97.9</td>
</tr>
<tr>
<td>112Sb</td>
<td>224</td>
<td>0.011</td>
<td>0.163</td>
<td>Impact</td>
<td>3.13</td>
<td>8.95</td>
<td>0.029</td>
<td>91.0</td>
<td>96.4</td>
</tr>
<tr>
<td>112Sd</td>
<td>20</td>
<td>0.007</td>
<td>0.148</td>
<td>Impact</td>
<td>0.10</td>
<td>0.22</td>
<td>0.001</td>
<td>99.7</td>
<td>98.2</td>
</tr>
<tr>
<td>112Sc</td>
<td>149</td>
<td>0.007</td>
<td>0.058</td>
<td>Impact</td>
<td>0.47</td>
<td>8.97</td>
<td>0.002</td>
<td>91.0</td>
<td>99.6</td>
</tr>
<tr>
<td>41Pa</td>
<td>481</td>
<td>0.017</td>
<td>0.175</td>
<td>Impact</td>
<td>2.49</td>
<td>13.9</td>
<td>0.012</td>
<td>86.0</td>
<td>99.1</td>
</tr>
<tr>
<td>41Pb</td>
<td>161</td>
<td>0.016</td>
<td>0.140</td>
<td>Non-Imp.</td>
<td>0.70</td>
<td>14.8</td>
<td>0.003</td>
<td>85.1</td>
<td>99.4</td>
</tr>
<tr>
<td>41Pd</td>
<td>7</td>
<td>0.009</td>
<td>0.035</td>
<td>Non-Imp.</td>
<td>0.01</td>
<td>0.28</td>
<td>0.000</td>
<td>99.7</td>
<td>100</td>
</tr>
<tr>
<td>41Pc</td>
<td>48</td>
<td>0.010</td>
<td>0.030</td>
<td>Non-Imp.</td>
<td>0.06</td>
<td>6.03</td>
<td>0.000</td>
<td>93.9</td>
<td>100</td>
</tr>
</tbody>
</table>

Fig. 2-17 also shows that the combined effect of decreasing the standoff from the high wall and increasing the tub amplitude (less mass) yielded the uppermost curve associated with the most aggressive finishing. Conversely, the combination of greatest standoff from the high wall and the smallest tub amplitude (most mass) yielded the lowest curve which would correspond to the least aggressive finishing.
Fig. 2-17: Frequency of impacts occurring per unit area as a function of effective impact velocity, for sensor at two stand off distances (‘a’ (44 mm) and ‘b’ (95 mm) in Fig. 2-8) from the high wall. Derived from 300 s of impact force data. In legend, number gives mass of media in kg, P=porcelain media, S=steel media.

Fig. 2-18 shows the frequency of impacts per unit area as a function of the effective impact velocity for sensor locations ‘b’, ‘c’ and ‘d’, each being 95 mm from the high wall, side wall and low wall, respectively. For both the porcelain and steel media, the effective impact velocity tends to be much larger at location ‘b’, and comparable at locations ‘c’ and ‘d’ for much of the impact velocity range. This suggests that the most aggressive finishing location is likely to be close to the high wall and not the low or side walls. An exception is seen with the 112Sd data, which has an unusually large occurrence of higher effective impact velocities, but relatively fewer of the more common low velocity impacts. Table 2-2 shows that position ‘d’ is always associated with very small $f_{avg}$; however, $v_{max}$ at ‘d’ is always larger than at ‘c’ if all other conditions are the same. This shows that the low wall creates fewer impacts, but the distribution is skewed more to the high velocities than for the side
wall. This is reflected in a comparison of the Table 2-2 $v_{max}$ data for 112S and 41P at locations ‘d’ and ‘c’.

![Graph showing frequency of impacts occurring per unit area as a function of effective impact velocity at a constant standoff distance of 95 mm for three sensor positions: facing the high wall, side wall, and low wall (‘b’, ‘c’, ‘d’ in Fig. 2-8). Derived from 150 s of impact force data. In legend, number gives mass of media in kg, P=porcelain media, S=steel media.]

**Fig. 2-18** Frequency of impacts occurring per unit area as a function of effective impact velocity at a constant standoff distance of 95 mm for three sensor positions: facing the high wall, side wall, and low wall (‘b’, ‘c’, ‘d’ in Fig. 2-8). Derived from 150 s of impact force data. In legend, number gives mass of media in kg, P=porcelain media, S=steel media.

### 2.4.3. Energy Flux

Fig. 2-19 shows the distribution of the energy flux of the contacts as a function of the effective impact velocity for all the experimental conditions. Although Fig. 2-17 and Fig. 2-18 show that the most frequent impacts are at lower velocities, it is apparent from Fig. 2-19 that the mid-range velocities for a given condition deliver the most energy flux. This also illustrates why $E_{Total}$ (Table 2-2), the integral of each curve in Fig. 2-17, is insufficient to quantify the effectiveness of a vibratory finishing system for a particular application. Knowledge of the distribution of the impact velocities is also required to predict the process
effectiveness, because certain applications require a high frequency of contact and do not depend on the severity of the impacts (e.g. drying using absorptive media). Other applications, such as in burnishing, depend less on the frequency of the impact and more on the velocity of the impact to cause plastic deformation. In general, conditions that create a high frequency of energetic contacts leading to large $f_{avg}$, $E_{Total}$, $v_{avg}$ and $v_{max}$ will be aggressive for any application.

![Graph showing energy flux as a function of effective impact velocity for different sensor positions.](image)

Fig. 2-19: Energy flux as a function of effective impact velocity for three sensor positions: facing the high wall, side wall, and low wall (‘b’, ‘c’, ‘d’ in Fig. 2-8). In the legend, number gives mass of media in kg, P=porcelain media, and S=steel media. Results for positions ‘a’ and ‘b’ were derived from 300 s of impact force data, and from 150 s at ‘c’ and ‘d’.

Table 2-2 shows that, all other factors being equal, when the mass of media in the finisher is decreased, and hence the amplitude of the tub vibration is increased, $f_{avg}$, $E_{Total}$, $v_{avg}$ and $v_{max}$ increase (compare 112Sa with 91Sa, and 112Sb with 91Sb). This suggests that the increased amplitude in the lower mass conditions leads to an expansion of the fluidized bed, and more energetic impacts associated with a larger mean free particle path.

Table 2-2 shows that more than 95% of all contacts are of the non-impact type, although $v_{max}$ is usually associated with impact. As well, during more than 85% of the
finishing time the sensor disk did not register a contact of any type. This is a significantly larger fraction than the 50% that might be expected as a result of media movement away from the sensor during half the vibration cycle.

2.5. Conclusions

The impact velocity of the media in a vibratory finisher is of fundamental importance in the analysis of erosion and surface plastic deformation during burnishing and work hardening. A piezoelectric impact force sensor has been calibrated to measure the spatial and temporal distributions of the normal impact velocities within a tub-type vibratory finisher. It was found that contacts occur periodically within time periods called ‘bursts’ and that the length of the burst periods correspond with the finisher’s driving frequency.

Two types of contact were recorded: short duration discrete impact contacts and longer duration non-impact contacts. The measured impact velocities were less than 0.5 m/s so that impacts could be modeled as quasi-static contacts, allowing maximum impact forces to be compared directly with the maximum forces measured in non-impact contacts. The latter were then used to find effective impact velocities for non-impact contacts.

It was found that the histogram of the effective impact velocity normalized by the total number of recorded contacts in a signal was a function of the signal duration, with relatively infrequent high-speed impacts appearing only at sufficiently long sampling periods. In general, the effective impact velocity frequency distribution was found to be a strong function of finisher amplitude and the sensor location within the finisher. It was also found that the reducing the total media mass in the finisher and moving closer to the wall results in a more aggressive process; i.e. the contact frequency, total energy flux and the average and maximum effective impact velocity were all increased with less total media mass.

This demonstrates the utility of the present impact velocity apparatus as a probe to investigate the effect of varying process parameters and finisher design in order to facilitate the optimization of vibratory finishing processes.

In earlier studies (Baghbanan et al. 2003, Wang et al. 2000, Yabuki et al. 2002), impact forces were usually measured using a sensor that was free to flow with the media rather than being fixed, as in the present case. Wang et al. (2000) did conduct a few measurements using a fixed sensor and found that the contact frequency was affected and
that the impact forces were greater than when the sensor was moving, presumably because of the increased effective stiffness of the sensor. In the present work, the use of a sensor fixed to a very stiff support provided impact velocity data that was independent of the workpiece size, shape and material. The fixed sensor was used as a probe to measure the absolute normal impact velocities of the media in the vibratory finisher rather than the impact velocities relative to a moving workpiece. Furthermore, it is hypothesized that the absolute and relative impact velocities are approximately equal for workpieces that are large compared with the media. In such cases the drag on the workpiece and its inertia will dampen the workpiece motion relative to that of the media, which are comparatively free to move. This is supported by video recordings made with cameras enclosed in moving workpieces that showed the media flowing and bouncing over the surfaces in a loosely packed arrangement (Wang et al. 2000, Yabuki et al. 2002).
3. Almen System

3.1. Introduction

The Almen system is well established (Cao et al. 1995) as an effective standardized (i.e. SAE J442) means of characterizing the shot peening process. Standardized metal strips are clamped to a rigid support and subject to the shot peening stream. On release of the clamps, residual stresses created by plastic deformation cause the strips to curve. The degree of curvature and its rate of change are related to the process parameters. The “Almen intensity” is the ultimate strip arc height reached after a certain saturation time defined such that doubling the exposure time causes only a 10% increase in arc height (Cao et al. 1995). The Almen system has the advantage that it combines the effects of all the process parameters into one measurement allowing for the characterization and determination of the repeatability of the process. These process parameters include shot density, radius, velocity, impact frequency, impact coverage, elastic modulus and yield stress, if the shot deforms plastically during impact.

Baghbanan et al. 2003 measured Almen strip curvature in a vibratory finisher as a function of lubrication, finishing duration, aluminium alloy, and finisher type. They measured the hardness of finished workpieces as a function of depth from the finished surface (Fig. 2-3). As the finishing time increased, the location of the maximum hardness moved from the surface to a depth of approximately 20 \( \mu m \). The Almen strip curvatures for strips made of AA1100-O, AA6061-T6 and Cu10100, were found to increase consistently as function of time. It was also found that at any given time, Almen strips finished in the tub had more curvature than those finished in the bowl since the tub was more energetic.

Baghbanan et al. 2003 also measured Almen strip curvature in both the wet and dry finishing conditions. The curvatures were always greater in the wet condition than the dry. This was unexpected, since it had been shown that the dry condition corresponded to higher surface-

3 This section has been accepted for publication in Wear, Ciampini D., Papini M., Spelt J.K, Characterization of vibratory finishing using the Almen system, Copyright Elsevier (2007).
normal contact forces. It was hypothesized that, in the wet condition, water could find its way between the Almen strip holding fixture and the Almen strip itself, causing an increase in the effective stiffness and thus an increase in the contact forces relative to the dry condition. This was supported by demonstrating that the curvature of an Almen strip increased when a layer of grease was sandwiched between the strip and the holding fixture. Consequently, Baghbanan et al. 2003 recognized the need to provide a consistent boundary condition during finishing and suggested using a hot melt adhesive to attach Almen strips to the holding fixture during the finishing operation. These issues are more significant in vibratory finishing than in shot peening because of the much lower impact velocities.

A number of investigators (Wang et al. 2000, Yabuki et al. 2002) measured contact forces in a vibratory finisher. Yabuki et al. 2002 concluded that tangential impact forces are approximately ten times less than normal forces. However, the measured contact forces were a function of the compliance of the sensors employed and hence were not necessarily representative of contact conditions against an arbitrary workpiece.

Section 2 related the normal component of the measured contact forces in a vibratory finisher to the impact velocity using a piezo-electric transducer. This method provided a measure of the energy availability in the contacts that was independent of the sensor compliance. Consistent with Yabuki et al. 2002, contacts could be classified as being either impact or non-impact events, depending on their duration. Both were expressed in terms of an ‘effective’ normal impact velocity. Although the piezo-electric sensor measured accurate impact velocities, it was of interest to explore further the potential of the Almen system to provide a simpler characterization of the impact conditions in vibratory finishing. The objective was to develop a tool to aid vibratory finishing process development and quality control.

In the present study, aluminium Almen strips were finished using the same vibratory finisher and media as in Section 2, providing saturation curves at two of the contact conditions that were characterized previously in terms of effective impact velocity distributions. This provided a means of relating the impact conditions to the observed changes in Almen strip curvature as a function of finishing time. The Almen strips were designed specifically for vibratory finishing applications, and a novel vacuum Almen strip attachment system was used to provide a constant boundary condition for the Almen strips
during finishing. The deflection of the Almen strips was measured using a novel gauge that applied a negligible force to the strips to provide an accurate measurement of the deflection. The relationship between normal impact velocity and Almen strip curvature saturation curves was also determined independently using an apparatus that simulated the impacts in the vibratory finisher.

3.2. **Apparatus**

3.2.1. **Almen Strips**

Almen strips 76.2 mm long and 19.05 mm wide were machined from aluminium 3003-H14 sheet, ensuring that they were not bent in the process. Two sheet thicknesses were used: 635 μm (0.025”) nominal thickness strips with an actual mean thickness of 617 μm with a standard deviation of 3.84 μm, and 813 μm (0.032”) nominal thickness strips having an actual mean thickness of 779 μm with a standard deviation of 9.76 μm. The mean and standard deviations of Almen strip thickness were based on ten strips with three measurements per strip. The 617 μm (0.025”) thicknesses was chosen to be less than that found in shot peening Almen strip specifications in order to accentuate the arc heights resulting from the much lower impact velocities in vibratory finishing.

Previous investigators have measured a small amount of anisotropy in aluminium sheet properties as a function of orientation relative to the rolling direction. Wu et al. 2003 found that the yield stress of AA3104-H19 and AA5182-0 aluminium in the cross-rolling direction was approximately 3.5% less than in the rolling direction. Very similar results were recorded for AA5019A-O by Choi et al. 2000, who demonstrated that annealing can make the crystallographic textures nearly isotropic and produce a slight reduction in yield stress anisotropy. In the present work, the Almen strips were cut so that the sheet rolling direction was normal to the long edge.

The Almen strips were annealed at 413°C for 1.5 hours to O-temper in order to relieve all the residual stresses in the samples from the manufacturing and machining processes. Comparisons with unannealed strips showed that such pre-existing residual stress can influence significantly the magnitude and variability of the measured arc heights.
3.2.2. **Vibratory Finisher and Media**

The present measurements were conducted in the same tub finisher used in Section 2. The media used were carbon steel balls (ABCO, Abbott Ball Company Inc., West Hartford, Connecticut, USA) having an average diameter and mass of 6.3 mm (standard deviation, SD=± 0.005 mm) and 0.29 g (SD=± 0.02 g).

3.2.3. **Vacuum Almen Strip Holder**

Almen strips were finished in a fixed position using a vacuum holder as shown in Fig. 3-1 and Fig. 3-2. This ensured that the strip remained flat, providing a constant boundary condition, and hence impact forces, as residual stresses accumulated during finishing.

![Fig. 3-1: Schematic of Almen strip vacuum fixture immersed in tub finisher.](image-url)
The vacuum holder consisted of a hollow aluminium cylinder with a polished, recessed rectangle (20.6x 76.7 x 0.5 mm) milled into the flat face and containing fourteen 0.5 mm diameter holes through to the vacuum chamber. Placing an Almen strip in the recess and evacuating the cylinder to approximately 3 kPa held the strip securely during finishing. To ensure repeatable flow conditions over the surface, the long direction of the Almen strip was always kept horizontal as shown in Fig. 3-2. The holder was attached to a stiff supporting column connected to an adjacent wall.

The curvature of the Almen strips may be a function of the compliance of the holder and its support. By being very stiff, the present vacuum holder and its supports provided a well defined, reproducible boundary condition. A vacuum holder that was free to flow with the media would have the advantage of providing an average response for the tub, but it would introduce the complication of a compliance and relative impact velocity that varied with its shape (drag coefficient), size and mass.

### 3.2.4. Arc-height gauge

Traditional Almen gauges use a dial gauge to measure the deflection of the finished Almen strips (the “arc height”) at their mid-span over a span of 38 mm (1.5”) while the strip rests on ball supports. The force exerted by a gauge would create a significant error with the relatively thin, flexible aluminum Almen strips used in the present experiments. It has been
estimated that typical Almen gauges exert tip forces between 0.2 and 1.5 N during typical usage (Champaigne 1992). Loads of this magnitude would result in mid-span deflections (or arc-heights) of between 40 and 300 µm for a 617 µm thickness AA3003 Almen strip over a 63.5 mm (2.5”) span, and between 20 and 150 µm for a similar 779 µm thick strip. Given that the actual maximum deflection, measured using the optical system described below, was 760 µm to 335 µm, for the 617 µm and 779 µm thick strips, respectively, it is clear that the forces exerted by a typical contact dial gauges are too high for the presently utilized thin strips. Furthermore, the error is a function of the material and thickness of the strip, thereby adding further complications when trying to compare results from different types of Almen strip. Consequently, an alternative approach was developed using the apparatus shown in Fig. 3-3. The Almen strip rested on four steel ball bearings over a span of 63.5 mm.

Fig. 3-3: Schematic of arc-height gauge.
This larger span increased the measured deflection at the mid-span and thereby the accuracy. With a strip in place, a microscope was used to view the back-lit tip of the micrometer probe as it was lowered to the Almen strip surface and stopped just as it made contact with a negligible load. Repeated measurements of arc height made in this manner were always within ±4 μm of the mean measurement.

3.2.5. **Vibratory Finishing Simulator**

In order to investigate the hypothesis that only the surface-normal component of the impact velocity had an appreciable effect on the Almen strip curvature in the vibratory finisher; an apparatus was constructed to finish an Almen strip with only normal impacts of a specified velocity. This was accomplished by using an electromagnet to pick up then drop the steel media repeatedly onto the Almen strip surface while it was secured using the vacuum holder. The vacuum holder and support beam shown in Fig. 3-1 were rotated 90° such that the Almen strip surface faced upward (Fig. 3-4).

A cage (25.4 mm by 82.55 mm) was constructed around the perimeter of the strip to contain the steel media while permitting the balls to make contact with every portion of the Almen strip surface. It was found that 33 steel balls provided enough free space to create randomized impact locations while finishing reasonably quickly. The electromagnet was
placed on the upper rim of the cage so that its flat surface was parallel to, and at a fixed height above the Almen strip.

A function generator supplying a square wave at 2 Hz was used to control a relay (Fig. 3-4) that cycled the electromagnet on and off. The frequency of 2 Hz was chosen to allow the balls enough time between drops to randomize their position and settle before they were picked up again. The diode in the circuit protected the contacts of the relay.

The constant impact speed was determined by the ball drop height (i.e. the height of the cage). Several cages of varying height were constructed to provide different maximum impact velocities. The range of possible impact velocities using the different cage heights was 0.1 to 0.37 m/s because of limitations of the electromagnet’s strength.

The speed at which a ball dropped from the electromagnet once it was turned off was measured using a high speed photographic apparatus. Several overlapping exposures of the ball were taken as it fell from the electromagnet to determine if the impact speed was being affected by lingering magnetism of the magnet or magnetization of the balls.

Fig. 3-5 demonstrates that there was no appreciable difference between the measured and predicted velocities. Nevertheless, to eliminate ball magnetization as a possible source of error, a new set of 33 balls was used for every new strip that was finished.

Although the balls struck every part of a strip, impacts tended to occur more frequently in two parallel lengthwise rows due to an uneven distribution of the magnetic field strength. This was evident mostly in the appearance of the surface during the early stages of finishing and became much less noticeable for longer times. Since this effect was the same for all strips, it did not influence comparisons among strips.
Fig. 3-5: Velocity of dropping ball as a function displacement from its initial position: measured - open symbols representing four repeat experiments; predicted assuming gravitational acceleration – filled symbols.
3.3. Results and Discussion

3.3.1. Vibratory Finisher Saturation Curves

Almen strips were finished at two of the vibratory finishing conditions that were characterized in Section 2 using an impact velocity probe. The first condition (91Sa) utilized 91 kg of steel media at location and orientation “a”, and the second (112Sb) utilized 112 kg of steel media at position and orientation “b”, shown in Fig. 2-8. Section 2 showed that the 91Sa condition corresponded to a more aggressive finishing condition than the 112Sb condition. This is evident in Table 3-1 which shows that the recorded average contact frequency \( f_{\text{avg}} \), maximum \( v_{\text{max}} \) and average \( v_{\text{avg}} \) effective impact velocities and the total energy flux \( E_{\text{Total}} \) were all greater for the 91Sa condition compared to the 112Sb condition. \( f_{\text{avg}} \) was calculated by dividing the number of contacts found in a force signal by the signal duration \( T \). \( E_{\text{Total}} \) was evaluated by summing the effective kinetic energy of each contact found in a force signal and dividing by the sensor contact area, \( A_{\text{sensor}} \), and the signal duration, \( T \), i.e.

\[
E_{\text{Total}} = \frac{m_{\text{avg}} \sum_{i=1}^{N} v_i^2}{2 \cdot T \cdot A_{\text{sensor}}} \tag{3-1}
\]

where, \( N \) is the total number of contacts that occurred during \( T \), and \( m_{\text{avg}} \) and \( v_i \) are the average media mass and effective impact velocity of the \( i^{th} \) contact, respectively.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Average contact frequency ( f_{\text{avg}} ) (Hz)</th>
<th>Average effective impact velocity ( v_{\text{avg}} ) (m/s)</th>
<th>Max. effective impact velocity ( v_{\text{max}} ) (m/s)</th>
<th>Total energy flux ( E_{\text{Total}} ) (W/m(^2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>91Sa</td>
<td>462</td>
<td>0.014</td>
<td>0.378</td>
<td>12.8</td>
</tr>
<tr>
<td>112Sb</td>
<td>224</td>
<td>0.011</td>
<td>0.163</td>
<td>3.13</td>
</tr>
</tbody>
</table>
Three types of Almen strips were finished at these two finishing conditions to generate the six saturation curves shown in Fig. 3-6. A new strip was used to determine the arc height for each finishing duration; i.e., strips were not finished again after their arc height was measured. Each data point in Fig. 3-6 corresponds to the average of five finished samples, and the error bars show the minimum and maximum arc heights. Significant variability was observed in the data from the vibratory finisher; the average standard deviation was 44 μm for each set of five strips finished at the same condition. Table 3-2 gives the average and standard deviation for the five samples at each condition. The scatter was due to the randomness of the contact conditions, the variability of the Almen strip thickness, small changes in the strip positioning during finishing, and arc height measurement error.
Fig. 3-6: Saturation curves produced using two finishing conditions and three types of Almen strip (617 µm thick 3003-O and 3003-H14; 779 µm thick 3003-O) (a) shows an expanded view of the data of Fig. (b) at low finishing durations.

Table 3-2: Average arc height and standard deviation (in brackets) of each set of five Almen strips finished in the vibratory finisher at the conditions in Fig. 3-6.

<table>
<thead>
<tr>
<th>Finishing Duration (min)</th>
<th>Al 3003-O 617 µm</th>
<th>Al 3003-H14 617 µm</th>
<th>Al 3003-O 779 µm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>91Sa 112Sb 91Sa 112Sb 91Sa 112Sb 91Sa 112Sb</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 (57) 0 (38) 0 (63) 0 (59) 0 (4) 0 (13)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>347 (32) 103 (34) 181 (67) 53 (62) 117 (29) 75 (20)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>378 (34) 121 (57) 111 (56) 59 (38) 150 (65) 76 (16)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>501 (38) 105 (36) 160 (43) 91 (33) 192 (26) 88 (21)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>585 (56) 311 (54) 212 (74) 122 (57) 206 (23) 155 (41)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>714 (46) 316 (73) 273 (46) 200 (40) 275 (42) 193 (30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>643 (29) 390 (67) 333 (61) 243 (65) 295 (40) 254 (43)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-6 shows that, for each of the three types of Almen strips, the arc height at any given finishing duration was always higher for the 91Sa condition than the 112Sb. This
was expected, because the 91Sa condition generated much higher effective impact velocities as well as a higher contact frequency (Table 3-1). Almen strips at 91Sa thus experienced more plastic deformation per unit time than those at 112SB.

It is apparent from Fig. 3-6 that the thin (617 µm) O-temper strips were the most sensitive to either finishing condition, showing the greatest change in arc height. This made them the better probes for characterizing the current vibratory finishing conditions. The arc heights of the thick (779 µm) O-temper and the thin (617 µm) H14-temper were similar. The thin H14-temper strips deflected to a lesser extent during the maximum finishing duration than the thin O-temper strips because they had a higher yield stress, 151 MPa and 41 MPa, respectively.

The arc height of each Almen strip approached a saturation value as finishing time increased (Fig. 3-6). In the case of the thin O-temper sheet, the arc height began to decrease after approximately 120 min at the 91Sa condition. A related finite element modeling study (Section 4) has indicated that the compressive residual radial stress distribution is driven below the neutral axis by repeated impacts leading to a decrease in curvature. After a single 0.225 m/s impact the plastic depth was predicted to be about half of the strip thickness, however, after one hundred overlapping impacts, this depth has extended to more than 75% of the strip thickness, resulting in a decrease in the internal bending moment. It is to be expected that this would have occurred earliest at the most aggressive finishing condition (91Sa) and with the thin O-temper Almen strip which had the lowest yield strength.

Almen saturation curves created from shot peening data are usually fit to exponential functions (Eq. (3-2)) in order to determine the Almen intensity (Kirk et al. 2005).

\[ H(T) = A\left(1 - \exp\left(B T^C\right)\right) + DT \]  

(3-2)

where \(A\), \(B\), \(C\) and \(D\) are parameters for the curve fit. Equation (3-2) also provides a good fit for saturation curves created by the vibratory finishing of Almen strips (Fig. 3-7).
The arc height $H$, and finishing duration $T$, that satisfy Eq. (3-3) are called the “Almen intensity” and “saturation time”, respectively (Cao et al. 1995)

$$H(2T) - H(T) = 0.1 \cdot H(T)$$  \hspace{1cm} (3-3)

There is no solution to Eq. (3-3) for the 617 µm thick Almen strip finished under the 91Sa condition because of the maximum observed in the arc height (Fig. 3-7). However, when the data are approximated by Eq. (3-2) using $C=1$ and $D=0$, a solution can be found.

Table 3-3 lists the Almen intensities and saturation times defined in this way for the data of Fig. 3-7 and for the AA3003-H14 strips. These parameters reflect the magnitude and frequency of the impact forces in the vibratory finisher at locations “a” and “b” with the two different media loads (91 and 112 kg).
Table 3-3: “Almen intensity” and “saturation time” obtained using Eqs. (3-2) and (3-3) with values of parameters A and B given in the table and with C=1 and D=0.

<table>
<thead>
<tr>
<th>Condition</th>
<th>A (µm)</th>
<th>B</th>
<th>Almen Intensity (µm)</th>
<th>Saturation Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA3003-O 617 µm 91Sa</td>
<td>627</td>
<td>0.533</td>
<td>564</td>
<td>4</td>
</tr>
<tr>
<td>AA3003-O 617 µm 112Sb</td>
<td>354</td>
<td>0.122</td>
<td>278</td>
<td>19</td>
</tr>
<tr>
<td>AA3003-O 779 µm 91Sa</td>
<td>253</td>
<td>0.442</td>
<td>178</td>
<td>5</td>
</tr>
<tr>
<td>AA3003-O 779 µm 112Sb</td>
<td>217</td>
<td>0.117</td>
<td>147</td>
<td>20</td>
</tr>
<tr>
<td>AA3003-H14 617 µm 91Sa</td>
<td>266</td>
<td>0.329</td>
<td>179</td>
<td>7</td>
</tr>
<tr>
<td>AA3003-H14 617 µm 112Sb</td>
<td>219</td>
<td>0.078</td>
<td>192</td>
<td>29</td>
</tr>
</tbody>
</table>

3.3.2. Vibratory Finishing Simulator Saturation Curves

The 617 µm O-temper strips were finished in the vibratory finishing simulator set to produce impact velocities of 0.18 m/s, 0.22 m/s, and 0.33 m/s, representative of the higher end of the effective velocity distributions measured for the 91Sa and 112Sb conditions. In addition, the 779 µm O-temper strip was finished at 0.22 m/s. Note that only the first impact of each drop cycle occurred at one of these target velocities; secondary impacts due to the bouncing of the media had much lower velocities. For example, a preliminary dynamic finite element analysis showed that a 0.22 m/s initial impacts had a second bounce impact velocity between 0.075 m/s and 0.15 m/s, depending on the surface hardness. Assuming a typical aluminum coefficient of restitution of 0.5, the impact velocity after the first bounce will be half of the initial velocity.

The saturation curves created by the simulator (Fig. 3-8) are similar to those resulting from the vibratory finisher (Fig. 3-6), reaching comparable maximum arc heights; i.e. 648 µm and 714 µm for the simulator and finisher, respectively. The principal differences between the curves from the simulator and those from the vibratory finisher are that curvature relaxation is more common in the simulator and the finishing durations are much longer than in the finisher. Fig. 3-8 shows that the maximum curvature in the simulator was seen at approximately 300 to 500 min at 0.22 m/s, or 2.5 to 4 times later than in the vibratory finisher. At 0.33 m/s, the curvature reached a maximum even more quickly in the simulator; i.e. between 150 and 300 minutes or 1.25 to 2.5 times later than in the vibratory finisher. In
general, the time to reach the maximum Almen curvature decreased as the impact velocity increased.

![Graph showing vibratory finisher simulator saturation curves for 617 μm strips (open symbols) and 779 μm strips (filled symbols). Impact velocities of 0.33 m/s (circles), 0.22 m/s (squares), and 0.18 m/s (triangles).]

As mentioned previously, the strip curvature begins to decrease at some point corresponding to the movement of the plastic zone below the neutral axis of the strip as a result of the cumulative deformation from many impacts. A related finite element modeling study in Section 4 has confirmed this hypothesis. Higher impact velocities create larger individual plastic zones, thereby reducing the number of impacts required to reach this point where the curvature begins to decrease. The longer finishing times seen in the simulator were expected since the operating frequency of the vibratory finisher was approximately 46 Hz.
compared with only 2 Hz in the simulator. Assuming that both the simulator and vibratory finisher had a close-packed arrangement of media and created the same impact velocity distribution, the simulator would have been expected to finish approximately 23 times slower. In reality, the media in the vibratory finisher was more closely packed than in the simulator, and therefore the vibratory finisher caused more impacts each cycle. Taking this into account, the simulator would have been expected to finish 28 times slower than the finisher. However, these packing and frequency effects were counteracted somewhat by the higher impact velocities in the simulator which were representative of maximum values in the finisher (Table 3-1).

The similarity of the Almen curves in the simulator and the vibratory finisher supports the hypothesis that the normal media velocity component is dominant (Section 2). It also indicates that, to a first order approximation, arc height is controlled by the largest impact velocities in vibratory finishing, with the more numerous lower velocities playing a relatively small role. This is consistent with the approach taken in the analytical shot peening curvature model of Hill et al. (1983).

3.4. Conclusion

The Almen system was adapted to vibratory finishing to quantify the effect of varying process parameters on the aggressiveness of the process. Saturation curves at two distinct operating conditions in a tub type vibratory finisher with steel spherical media were similar in form to those seen in shot-peening.

More aggressive finishing conditions, characterized by greater impact velocities, caused more plastic deformation and resulted in more Almen strip deflection in less finishing time. This supported the role of the Almen strip as a probe to characterize the aggressiveness of a vibratory finishing system in either process development or control.

Certain elements of the usual shot peening Almen system were modified to reflect the much lower impact velocities found in most vibratory finishers. Relatively thin AA3003 O-temper Almen strips (617 µm) displayed the greatest change in arc height, making them more sensitive probes than the thicker (779 µm) Almen strips. However these thinner strips showed no saturation point in the arc height and instead the arc height decreased after
reaching a maximum when subjected to the most aggressive finishing conditions encountered in the present study. The time to reach this maximum may be used as a characteristic of the impact conditions within a vibratory finishing system. Nevertheless, the generation of a traditional Almen curve displaying a well-defined saturation requires the selection of a strip having sufficient thickness given the impact conditions within the finisher under study.

An apparatus was constructed that showed that normal impacts alone produce saturation curves similar to those created in the vibratory finisher when impact velocities are comparable in magnitude. This provided support for normal impacts as the dominant mechanism of finishing.

Almen saturation curves in vibratory finishing may be used to show how changes in a process affect the finishing rate or the aggressiveness of the process.
4. Modeling of Stresses

4.1. Introduction

Vibratory finishing is widely used to modify the properties and microtopography of metal, ceramic and plastic parts. It is used to polish surfaces, smooth sharp edges, texture, and clean. Ductile metal surfaces experience significant plastic deformation, creating compressive residual stress that can improve fatigue performance (Gane et al. 2003). Current industrial practice relies largely on experience and experimentation to optimize the process for new parts and materials. Several authors (Domblesky et al. 2003, Domblesky et al. 2004, Hashimoto 1996) have proposed correlations and models to predict erosion rates in vibratory finishing.

In a typical configuration, a vibrating container fluidizes a bed of granular media creating a circulating bulk flow. A workpiece entrained in this flow is subject to the impacts of the vibrating media. The impact velocity of the vibrating media will control the impact force and hence the degree of plastic deformation and erosion of the workpiece surface.

The Almen system is well established (Cao et al. 1995) as an effective standardized (i.e. SAE J442) means of characterizing the shot peening process. Metal strips are clamped to a rigid support and subject to the shot peening stream. On release of the clamps, residual stresses created by plastic deformation cause the strips to curve. The degree of curvature and its rate of change are related to the process parameters. The “Almen intensity” is the ultimate strip arc height reached after a saturation time defined such that doubling this exposure time causes a 10% increase in arc height (Cao et al. 1995). The Almen system has the advantage that it combines the effects of all the process parameters into one measurement allowing for the characterization and determination of the repeatability of the shot peening process. These process parameters include shot density, radius, velocity, impact frequency, impact coverage, shot elastic modulus and yield stress, if the shot deforms plastically during impact.

---

4 This section has been submitted for publication to J. Materials Processing Technology, Ciampini D., Papini M., Spelt J.K, Modelling the development of residual stress in vibratory finishing
Section 3 described the adaptation of the Almen system to the vibratory finishing process, and demonstrated saturation curves that showed the relative aggressiveness of the finishing conditions. Section 2 described the measurement of the magnitude and frequency distribution of surface-normal impact velocities in a vibratory finisher at locations that were also used to record Almen curvatures as a function of time and process conditions. This provided a semi-quantitative relationship between Almen curvature and the impact conditions.

The objective of the present study was to develop a model of the process by which Almen strips become plastically deformed by media impacts in vibratory finishing. The motivation was to extend the use of Almen strip measurements as a means of characterizing vibratory finishing by improving the understanding of the process parameters that control time-dependent curvature development. The approach and results also have application in modelling processes such as peen forming, in which streams of shot are used to create curved metal panels.

4.2. Model

The modeling was intended to simulate aspects of the experimental Almen strip curvature data from Section 3. The Almen strips in this study were 76.20 mm long and 19.05 mm wide flat rectangular plates made of AA3003-O aluminum alloy. Two strip thicknesses were considered; i.e. 617 μm and 779 μm. The modeling began with finite element simulations of the development of residual stress and energy absorption as a function of repeated impact at a single point. The results were then used in a stochastic model that simulated the Almen curvature resulting from the impact of media having a given frequency distribution of velocities equal to that measured in Section 2.

4.2.1. Part 1: FEA

4.2.1.1. FEA Geometry

The geometry used in the finite element analysis was that of a ball of diameter 6.305 mm repeatedly impacting a circular section of aluminum sheet (10 mm diameter) that represented a section of an Almen strip. The ball was given only a normal component of
velocity between 0 and 0.4 m/s. Two sheet thicknesses were modeled, 617 and 779 µm, to match those used in experiments.

4.2.1.2. FEA in LS-DYNA

The residual stresses in the Almen strip after a series of impacts were obtained using an explicit finite element analysis with LS-DYNA from the ANSYS v8.0 release. The impacts were chosen to occur at the same location on the Almen strip, enabling the use of axisymmetric “PLANE162” elements. The element density was made to increase with proximity to the contact zone. The mesh of the ball and Almen strip are shown in Fig. 4-1(a). The results were independent of the mesh density if the number of elements was at least 1,650 for the thin Almen strip and 1,900 for the thick Almen strip. A different number of elements were used for each model to keep the mesh density approximately equivalent. The LS-DYNA ASS2D contact algorithms were used for collision detection between the ball and strip. The effect of repeated impacts was considered by restarting the analysis after rebound while retaining the stress and strain state of the target.

4.2.1.3. FEA Assumptions

4.2.1.3.1. Holder compliance and boundary conditions

The bottom nodes of the Almen strip were fixed or allowed to slide in the plane of the sheet, representing two extremes of the actual experimental boundary conditions at the interface between the Almen strip and the vacuum holder (Section 3). The FEA results were similar for both of these boundary conditions.

The effect of the compliance of the Almen vacuum holder was also investigated using an FE model of a very stiff holder and one in which the modulus of elasticity was decreased by 25%. The results showed these changes had a very small effect on the predicted residual radial stresses. This was confirmed experimentally by manufacturing a second, much more rigid vacuum Almen holder and observing that the saturation curves were not significantly different. Finally, it was shown that the inertial properties of the Almen strip holder did not have a large effect, even when the density was decreased to 25% of that of aluminum.
Fig. 4-1: (a) Axisymmetric finite element mesh of the spherical media impacting the aluminium sheet (617 μm thickness). (b) Almen strip showing compressive radial residual stress (negative) region developed directly underneath the indented side of the strip. A small tensile (positive) region is also shown. $M$ is a moment require to keep the strip flat due to these internal residual stresses, and when applied to the unstressed beam would create deflection in the sense shown.
4.2.1.3.2. Aluminum sheet constitutive models

The effect of strain rate was neglected in the FEA, because the low impact velocities were not expected to create a large dependency on strain rate with the present AA3003-O aluminum sheet.

Two constitutive relationships were used: an isotropic, elastic perfectly-plastic model (Eq. (4-1)) was used with a yield stress \( \sigma_y \) of 41 MPa, and an isotropic power law hardening model (Eq. (4-2)) with material constant \( K=188 \) MPa and hardening coefficient \( n=0.242 \) (Hatch 1984). The other aluminum properties used were a modulus of elasticity of 71.7 GPa, Poisson’s ratio of 0.33, and density of 2,800 kg/m\(^3\) (Hatch 1984).

\[
\sigma = \begin{cases} 
E \varepsilon & \sigma < \sigma_y \\
\sigma_y & \sigma = \sigma_y
\end{cases}
\]  \hspace{1cm} (4-1)

\[
\sigma = K \varepsilon^n
\]  \hspace{1cm} (4-2)

The steel media were modeled as rigid spheres, each with a density of 7,800 kg/m\(^3\). Modeling the sphere’s elasticity gave similar results, but at much higher computational cost.

The effect of friction was studied using the non-hardening material model and a static and dynamic coefficient of friction of 0.61 and 0.47, respectively (Norton 2000). The inclusion of friction between the media and Almen strip produced a significant difference in the axisymmetric FEA results. The process model used the FEA results, however, could be simplified by neglecting friction for reasons that will be discussed below.

4.2.1.4. Model Validation

The FE modeling approach was validated by reproducing results found in the literature. The residual centerline radial stresses left by a 50 m/s and a 75 m/s impact were reported by Megiud et al. (1999), and are reproduced in Fig. 4-2 along with the present FEA results assuming an elastic perfectly-plastic material model. Two different mesh densities were used (222 and 322 elements) to bound the curves given by Megiud et al. (1999). The refinement level did not have a significant effect on the stresses with the exception of those at the surface as seen in Fig. 4-2(b). Excellent agreement with Megiud et al. (1999) was observed, lending confidence to the present FEA.
Fig. 4-2: The residual centerline radial stresses produced by shot impacts at (a) 50 m/s and (b) 75 m/s impact as reported by Meguid et al. (1999) and reproduced using the present model at different levels of mesh refinement. The stress axes are normalized by division by 600 MPa.
4.2.1.4.1. Energy Storage and Dissipation

It was confirmed that the kinetic energy lost by the ball was gained by the strip in the form of plastic work and elastic strain energy stored as residual stress. This was shown for 100 impacts for each of the impact velocities and material models.

The calculation of the energy dissipated in plastic deformation and the residual elastic strain energy began with the transfer of the 2D axisymmetric FEA results from the free mesh to a 2D ordered axisymmetric mesh shown in Fig. 4-3 by interpolation. The ordered mesh consisted of 26 columns of 100 nodes 1 μm apart in the radial direction beginning at the axis of symmetry. The nodes in the 617μm and 779μm thick strips were spaced 0.617μm and 0.779 μm apart respectively in the depth direction. At each of the nodes in the ordered mesh, the equivalent (von Mises) plastic strain and the equivalent total strain were obtained along with the equivalent stress for each time step.

The equivalent elastic strain ($\varepsilon_e$) was obtained by subtracting the equivalent plastic strain ($\varepsilon_p$) from the equivalent total strain. These stresses and strains were then assumed to be constant over the area surrounding each node as is shown in Fig. 4-3. The contribution to the energy change was taken to act over the volume obtained by sweeping each area about the axis of symmetry shown in Fig. 4-3. The energy changes per unit volume were calculated for each node using Eq. (4-3), as the sum of the recoverable elastic strain energy (Eq. (4-4)) and the unrecoverable energy of the plastic strain (Eq. (4-5)).

$$dW = dW^p + dW^e \quad (4-3)$$

$$dW^e = \sigma_e d\varepsilon_e \quad (4-4)$$

$$dW^p = \sigma_e |d\varepsilon_p| \quad (4-5)$$

where $W$ is the total work, $W^p$ is the work done in plastic deformation, $W^e$ is the elastic strain energy and $\sigma_e$ is the equivalent stress. Finite approximations of the infinitesimal quantities that represent changes over time were obtained over the output time step of the FEA (4.5454x10⁻⁵ s).
4.2.1.5. Centerline Radial Stress, Curvature and Arc Height

There are several examples in the literature of using the radial stress distribution through the thickness at the center of a single impact to represent the stresses in the entire surface after a shot peening process (Guagliano 2001, Hill et al. 1982). These stresses are considered to exist everywhere in the finished surface. The moments required to keep the strip flat in the presence of these residual stresses represent the effect of the fixed boundary conditions. These moments applied to an unstressed initially flat target create deflections of the target equivalent to the deflection that residual stresses would create (4-1(b)). Guagliano (2001) used a beam approximation to model the deflection of an Almen strip and calculated the moment ($M$) in terms of the residual radial stresses ($\sigma_{r,\text{res}} (y)$) as,

$$ M = \int_d \sigma_{r,\text{res}} (y) y dA $$

(4-6)
where \( y \) is the distance from the neutral surface of the Almen strip and \( A \) is the surface area of the cross section of the Almen strip. The arc height \( (h) \) of an Almen strip can then be expressed as a function of the residual moment \( (M) \), Young’s modulus \( (E) \), the reference distance \( (l) \) over which the curvature is measured (Almen intensity), the strip width \( (b) \) and the strip thickness \( (t) \) (Guagliano 2001):

\[
h = \frac{3Ml^2}{2Ebt^2}
\]  

(4-7)

The arc height \( (h) \) in the present study was obtained from Eq. (4-7) with \( l=63.5\text{mm} \) (or \( 2.5\text{”}) \equiv l_{\text{strip}} \). If the curvature is assumed constant, the strip surface can be described as a section of a sphere, and the arc height is related to the curvature \( (\kappa) \) and the spherical radius \( (\Gamma) \) as:

\[
\kappa = \frac{1}{\Gamma} = \frac{8h}{4h + l^2}
\]  

(4-8)

A similar approach was taken in the present work, with the exception that radial stresses, moments and curvatures were not assumed to be constant over the entire Almen strip, but rather, as a first approximation, were associated with much smaller circular contact radius of radius \( (c) \) given by Eq. (2-6) (Hill et al. 1982),

\[
c = R \left[ \frac{1}{4} v^2 \rho_1 \pi \left( 1 - \frac{\mu_2}{\pi E_1} + \frac{1 - \mu_2^2}{\pi E_2} \right) \right]^{\frac{1}{5}}
\]  

(4-9)

where \( v \) and \( \rho_1 \) are the impact velocity and density of the media of radius \( R \), while \( E \) and \( \mu_1 \) are the Young’s modulus and Poisson ratio of the media (1) and target (2), respectively.

### 4.2.1.6. Axisymmetric FEA Results

Results were obtained for 100 overlapping impacts at the same location at several different velocities using both the hardening and non-hardening aluminum sheet models. As the ball approaches the surface, contact is made with the surface, and the surface displaces with the ball until it stops and changes direction. With repeated impacts, the surface of the Almen strip is progressively displaced downward due to plastic deformation. The displacement of the top surface of the strip is shown in Fig. 4-4 as a function of impact number. The increments in the depth and radius of the dent decreased with impact number, with the dent asymptotically approaching a steady-state shape. The pileup of material around the periphery of the dent increased with impact number. The notches at the top of the pile-up
zone are due to the use of finite elements to approximate a smooth curvature of the spherical indenter region. Refining the mesh reduced this phenomenon; however, it also prohibitively increased the computational time.

![Graph showing displacement of Almen strip surface](image)

**Fig. 4-4: Displacement of Almen strip surface after repeated impacts at 0.15 m/s (elastic perfectly-plastic 617μm thick target).**

During the 100 simulated impacts at 0.15 m/s with the 617 μm thick strain-hardening strip, the rebound velocity increased from 0.05 m/s to 0.13 m/s as the coefficient of restitution increased with increasing contact stiffness. Similar trends were seen with other impact velocities.

Each impact caused the distribution of the radial residual stresses along the impact centerline to move deeper into the Almen strip and to approach a steady-state configuration (Fig. 4-5). The effect of the constitutive model can be seen by comparing Figs. 4-5(a) and (b) and also comparing Figs. 4-5(c) and (d). The maximum tensile radial stresses within the Almen strip are about equal in magnitude for both models; however, the hardening model tensile stresses occur slightly deeper from the impacted surface. The tensile stresses decrease in magnitude as the number of impacts increases, becoming compressive if the impact
velocity is sufficiently high. The locations at which the maximum compressive radial stress occurs are pushed deeper into the material with additional impacts. In general, the non-hardening model predicts that the stresses change more with each impact, but more impacts are required to reach steady state than the hardening model which has an increasing yield stress, resulting in less plastic deformation.

Comparing Figs. 4-5 (b) and (d) shows that an increase in the impact velocity causes the stress distribution to become more compressive and that the compressive stress distribution is pushed deeper into the material.
Fig. 4-5: The radial residual stress distribution history after each impact for 100 repeated impacts against a 617 μm thick strip: (a) 0.15 m/s impact velocity against an elastic perfectly-plastic target, (b) 0.15 m/s impacts using a power law hardening target, (c) 0.30 m/s impacts using an elastic perfectly-plastic target, (d) 0.30 m/s impacts using a power law hardening target. The numbered arrows indicate the curves corresponding to the first, second, third and one-hundredth impact; the unnumbered curves continue in the same sequence.

The Almen strip arc height \( h \) (Eq. (4-7)) was calculated after each impact over a 63.5 mm (2.5”) reference distance \( l \). For a given ball radius, strip thickness and material constitutive model, it was found that the arc height was a unique function of the cumulative energy absorbed by the Almen strip, obtained from the summation of the kinetic energy lost by the ball after each impact (assuming frictionless contact). Figure 4-6 illustrates this for a range of impact velocities for the two strip thicknesses; each modeled using the two material models described in Section 4.2.1.3.2. In all cases, the arc height increased rapidly to a maximum and then decreased gradually to an approximately constant value for large amounts of absorbed energy.
As expected, a series of higher velocity impacts caused a greater cumulative absorbed energy and a more rapid change in the arc height through this same pattern. A series of sufficiently low velocity impacts (e.g. less than 0.075 m/s in Fig. 4-6(a)) do not generate sufficient absorbed energy to cause the strip arc height to reach its maximum value. Instead, saturation occurs (i.e. coefficient of restitution \( e = 1 \)) before reaching the maximum arc height. Only for the combination of the non-hardening model and the thinner Almen strip of Fig. 4-6(b) did the impact velocity have a significant influence on the arc height - energy relationship. For both strip thicknesses, the power law hardening model predicted that reverse bending would occur after several impacts; i.e. the arc height would become negative. This can be explained by the trends of Figure 4-5(a) and (b) which show how the radial residual stresses are driven deeper into the material. For sufficiently high impact velocities, the stresses become distributed about the strip neutral surface such that the resulting bending moment changes sign and causes the strip to bend in the opposite sense resulting in negative arc height. It was common, in experiments, to see curvature relaxation, i.e. a trend of decreasing in arc height following a maximum arc height in a saturation curve. However, negative arc heights were not observed. It was presumed that given sufficient finishing time, and in the presence of sufficient impact velocities, reverse bending would occur in experiments as predicted.

Figure 4-6 shows that the hardening and non-hardening models predicted similar maximum arc heights for the 617 \( \mu m \) and 779 \( \mu m \) thick Almen strips: for the hardening model, 717 \( \mu m \) (at \( 1.74 \times 10^{-5} \) J) and 595 \( \mu m \) (at \( 5.11 \times 10^{-5} \) J), respectively, and for the non-hardening model, 731 \( \mu m \) (at \( 3.00 \times 10^{-5} \) J) and 573 \( \mu m \) (at \( 6.71 \times 10^{-5} \) J), respectively. As expected, the non-hardening model corresponded to greater energy absorption at the maximum arc height.
Fig. 4-6: Arc height vs. absorbed energy for 100 impacts at the velocities indicated in the legend (each data point represents a single impact): (a) 617 µm thickness, power law, (b) 617 µm thickness, elastic perfectly-plastic, (c) 779 µm thickness, power law, (d) 779 µm thickness, elastic perfectly-plastic.
Figure 4-7(a) shows the arc height of the Almen strip and Fig. 4-7(b) the energy absorbed by the Almen strip after the last impact in a series of 100 impacts as a function of the impact velocity of each impact series. Figure 4-7(a) and (b) reveal that the arc height of an Almen strip and the absorbed energy is limited by the incident velocity in any series of repeated impacts, respectively. Figure 4-7(b) shows that the energy absorption depends more on the material model than the thickness of the strip; however, for a given material model and impact velocity the 617 μm strip absorbs slightly more energy than the 779 μm strip.
Figure 4-7: (a) Arc heights and corresponding (b) cumulative absorbed energy after 100 repeated impacts of different velocities for several conditions shown in legend.

Figure 4-6 gives the curvature of the Almen strip surface as a function of the energy absorbed by the strip at that location. To use these relationships in a model of the vibratory finishing process, it was necessary to calculate the energy absorbed for a range of media impact velocities. This was achieved by determining the coefficient of restitution ($e$) under conditions of repeated impact.
(a) 617 µm thickness
Power-law hardening

(b) 617 µm thickness
Elastic-plastic
Fig. 4-8: Coefficient of restitution vs. absorbed energy for a range of impact velocities: (a) 617 μm thickness, power law, (b) 617 μm thickness, elastic-plastic, (c) 779 μm thickness, power law, (d) 779 μm thickness, elastic-plastic. Each marker style (shown in the legends of each figure) indicates the impact velocity (m/s) for each impact in a series of 100 impacts.
The functional dependence of the coefficient of restitution on the impact velocity and the previously absorbed energy (Fig. 4-8) was determined using a series of 100 impacts at the same velocity, repeating this for several different velocities. It was established that the coefficient of restitution of an impact depends only on the impact velocity and the previously absorbed energy of the target; i.e. the order and magnitude of previous impacts did not affect $e$. This is illustrated in Fig. 4-9 for several series of repeated impacts, some having impacts that alternate between two impact velocities, others having fixed impact velocities, and one with several different velocities. As will be seen below, this is a useful result because it enables the total absorbed energy to be used as an independent variable characterizing the properties of the target surface after an arbitrary number of previous impacts over a range of velocities in the vibratory finishing process model.

![Graph showing the coefficient of restitution as a function of the energy previously absorbed by the target for repeated impacts of different velocities using the elastic-plastic constitutive model against a 617 μm thick target.](image)

**Fig. 4-9:** Coefficient of restitution as a function of the energy previously absorbed by the target for repeated impacts of different velocities using the elastic-plastic constitutive model against a 617 μm thick target.
As was seen previously in Figs. 4-6 and 4-8, the incremental energy absorbed by an Almen strip decreased with an increasing number of impacts, approaching zero for a given impact velocity. This limited the cumulative absorbed energy to a saturation value that was proportional to the incident velocity, and that was reached with fewer impacts for lower velocities than for higher velocities. Thus, it was expected that the Almen strip arc height would reach a saturated condition more quickly for less aggressive finishing conditions (i.e. lower impact velocities), given an equivalent average impact frequency. This is supported by experimental observations of Almen strip finishing in a vibratory finisher, discussed in the following section; i.e. a less aggressive condition causes an earlier saturation than does a more aggressive condition.

The hardening material model predicted that saturation would occur at lower energies with fewer impacts compared to the elastic-plastic model (Figs. 4-8 and 4-9) due to the increase in yield stress caused by plastic deformation. Nevertheless, both the hardening and the non-hardening models produced very similar results in the process model, as will be discussed in section 4.2.2.

4.2.1.7. Distribution of absorbed energy

The energy absorbed by plastic deformation and residual stress varied with radius over the impact zone. However, it was found that the distribution was similar for different impact velocities. This is evident in Fig. 4-10 which shows the normalized absorbed energy distribution as a function of normalized radial coordinate for a range of velocities.
Fig. 4-10: The normalized distribution of absorbed energy as a function of a normalized radial distance from the centerline, after each impact, for a several series of 100 impacts at velocities of 1.0, 2.5, 5.0, 7.5, 10.0, 12.5, 15.0, 17.5, 20.0, 22.5, 25.0, 27.5, 30.0, 32.5, 35.0, 37.5, 40.0 cm/s. Radial distance normalized by distance containing 95% of total energy, absorbed energy normalized by the maximum absorbed energy in each distribution. The hardening model and 617 μm thick strip.

4.2.2. Part 2: Process Model

Section 4.2.1 outlined a methodology to predict Almen curvature as a function of absorbed energy using FEA of repeated impacts at a single location. In order to simulate the development of the Almen strip curvature as a function of vibratory finishing time, the FEA results were used in an analytical process model that incorporated an arbitrary impact velocity distribution and varying degrees of finishing coverage. The axisymmetric FEA data were used to predict the Almen strip response to impact. In general, the model tracks energy changes to a simulated region of an Almen strip when it is subjected to impacts over time. At any given time in the simulation, the stresses can be obtained from the absorbed energy state, allowing a prediction of the strip curvature.
Both a simple one-dimensional (1D) and a two-dimensional (2D) model were developed. The 1D model was much more computationally efficient, but the 2D model was found to be less limited than the 1D model, and consequently the 2D model is discussed in detail. The 1D model is summarized and compared with the 2D model in Section 4.4.

4.2.2.1. Calculating the energy absorption history

The model located impacts stochastically in a plane region representing a portion of the Almen strip surface. The velocity of each impact was taken directly from a 5 min sampling (with a minimum of 60,000 impacts) from an impact force sensor located in a tub vibratory finisher (Section 2). If more contacts were needed in the simulation, the model would loop to the beginning of the 5 min sequence repeating the same sequence of impact velocities as many times as was required. Every point on the surface had the same probability of being struck.

The media contacts at normal velocity \( v \) in the process model were assumed to affect circular areas with a radius given by Eq. (4-10) for the hardening material model for both 617 and 779 \( \mu m \) thick Almen strips. Eq. (4-10) was obtained from a least-squares fit of the radii of the first impacts of the axisymmetric FEA data shown in Fig. 4-11. The radius of a contact at a given impact velocity corresponds to a circular area on the surface of the target that overlies the region containing at least 95% of absorbed impact energy. These areas were then used to express the absorbed energy of Figs. 4-8 (a) and (c) on a per unit area basis. Eq. (4-10) provided a good approximation of Fig. 4-11 since the radius for 95% energy absorption did not change significantly with additional impacts following the first.

\[
c = \left( 1.7v - 2v^2 \right) / 1000
\]

Regression Coefficient \( R^2 = 0.97 \)
Contact radius (mm) = 
\[
\begin{align*}
0.754(\text{absorbed energy (mJ)})^{0.2721} \\
R^2 = 0.9978
\end{align*}
\]

Fig. 4-11: Radius about impact centre encompassing more than 95% of impact energy absorption from the axisymmetric strain hardening model for 100 repeated impacts at 0.025 to 0.4 m/s in increments of 0.025 m/s for both the 617 μm and 779 μm sheet thicknesses.

The 2D model considered the intersection of the circular impacts with a 3.0x0.6 mm zone. This was found to produce results that were similar to those of a zone with an area equal to the full Almen strip. Square elements (shown in Fig. 4-12) were chosen to fill the 2D region, with 5 times the number of elements along the length as along the width. Tests with aspect ratios of 4 and 10 indicated that the predicted saturation curves were relatively insensitive to this choice.
Fig. 4-12: Sequence of random impacts at different velocities with varying amounts of overlap on a small region on the surface of the Almen strip. Each dash (11 in total) in the dashed line represents a 1D element and the boxes are the 2D elements used to determine the deformation caused by the collective effect of the impacts. (a) first impact, (b) second impact, (c) third impact, (d) fourth impact. A darker shade of gray corresponds to more energy absorbed by the Almen strip.

Contact locations in the 2D model were chosen randomly within an area extending a distance $c$ from the 3.0x0.6 mm boundary, where $c$ is the radius of the contact area (dependent on the impact velocity). This mimicked the way in which contacts were recorded by the impact velocity sensor (Section 2).

Figure 4-12 represents the progression of the impact process using a grid of square elements. Figure 4-12 (a) shows an impact whose contact area intersected 16 elements. Partial intersection of an element was treated as a full intersection if the center of the element was covered by the contact area, otherwise the element was deemed not to have been intersected. The amount of energy absorbed by an element depended on the impact velocity and the previously absorbed energy through the coefficient of restitution (Fig. 4-8) and Eq. (4-11). Both the energy of an impact and the cumulative absorbed energy were expressed on a unit area basis using Eq. (4-10) in order to attribute them to particular elements within the impact zone. Therefore, the total cumulative absorbed energy for a given impact velocity in Fig. 4-8 was divided by the impact area assuming a uniform impact energy distribution.

The total energy absorbed ($E_a$) from an impact was calculated from the coefficient of restitution ($\epsilon$), the impact velocity ($v$), and the media mass ($m$) using:

$$E_a = \frac{1}{2} m v^2 \left(1 - \epsilon^2\right)$$

(4-11)
As mentioned above, $E_a$ was then divided by the impact area.

The replacement of the actual energy distribution, Fig. 4-10, with a uniform distribution over the impact zone was investigated for several cases and found to produce negligible differences in the predicted strip curvature; therefore, the more computationally efficient uniform energy distribution was used.

The impact frequency used in the model, $f_m$, was scaled from the impact frequency of the measured signal ($f_s$) in proportion to the ratio of the model analysis area ($A_m$) to the area of the sensor ($A_s$):

$$f_m = f_s \left( \frac{A_m}{A_s} \right) \quad (4-12)$$

The probability of an impact’s circular contact area of radius, $c$, having intersected a target area was a function of $c$ and the target area. The target area considered was rectangular with dimensions $W_1 \times L_1$ or circular with radius $r_s$. A larger rectangular area ($W_2 \times L_2$) that included the target area was taken to be subjected to a known impact velocity probability distribution. The approximate difference between actual and the apparent probability for a given impact velocity with contact radius, $c$, is given by Eq. (4-13) for a rectangular target area and by Eq. (4-14) for a circular target area. Equation (4-14) shows that the sensor of Section 2 with a diameter of 4 mm was sufficiently large to sense the actual velocity distribution of impacts falling on an Almen strip (76x19 mm) with an error of 0.6% in the velocity range $0 - 4.6 \times 10^4$ m/s. Similarly, the present model area (3.0x0.6 mm) is seen to be large enough to accurately represent the entire Almen strip.

$$P_r = \frac{(W_1 + 2c)(L_1 + 2c)}{(W_2 + 2c)(L_2 + 2c) - c^2 (4 - \pi)} \quad (4-13)$$

$$P_s = \frac{\pi (r_s + c)^2}{(W_2 + 2c)(L_2 + 2c) - c^2 (4 - \pi)} \quad (4-14)$$
4.2.2.2. Obtaining stresses and deflection from an absorbed energy state

The average energy absorbed per unit area over the model area (3.0x0.6 mm) was used with Fig. 4-6 (expressed on a per unit area basis) to obtain the arc height of the strip.

This analytical method was compared with a finite element model (FEM) using ANSYS™ SHELL181 elements corresponding to the mesh used in the energy absorption calculations (Fig. 4-13). The node at the origin was constrained from rotation about the z-axis and displacement in all directions, while two other corner nodes that were intersected by the x and y axes were constrained to the x-y plane. The absorbed energy of each element was used to find its arc height (Fig. 4-6) which was converted to an equivalent bending moment about the neutral surface of the element assuming a linear distribution of stress, with the top surface stress given by

\[
S_r = \frac{4htE}{l^2}
\] (4-15)

The stress at the bottom of the element was equal in magnitude, but with the opposite sign.

Fig. 4-13: A 20 by 100 element mesh of ANSYS™ SHELL181 elements deflecting due to constant and equivalent bending moments in both x and y directions (coordinate system shown).
In this way, the stresses predicted by the VF process model were used to initialize the stresses of the finite element model. The resulting deformation of the model area therefore included the restraining effect of elements surrounding an impact site.

Several different imposed stress states were examined in the FEM. Using only the stresses in the $x$-direction resulted in deflections that followed more closely the analytical deflection model. Specifying the moment in the $y$-direction to be equal to that in the $x$-direction created curvature in the $y$-direction that slightly reduced the overall deflection of the model area.

The above analytical and FE approaches were based on the correlations established using the axisymmetric LS-DYNA FEA between the cumulative absorbed energy and both the element arc height and the coefficient of restitution. The accuracy of this approach was compared with the use of the actual moment distribution from each impact obtained from the axisymmetric LS-DYNA FEA by using the stresses at the surface of each element in the $x$ and $y$ directions to obtain the linear stress distributions in these directions. The deflections obtained by this method were only slightly smaller than those obtained using the centerline stresses in the $x$ and $y$ directions. Since the various deflection calculation methods produced similar results, only the method that used the centerline stresses in the $x$-direction are presented below.

The radius of curvature of the model area was used to obtain the Almen strip deflection using

$$h_{\text{Strip}} \approx h_{\text{Model}} \left( \frac{l_{\text{Strip}}}{l_{\text{Model}}} \right)^2$$  \hspace{1cm} (4-16)

where $h$ and $l$ are the deflection height and length of the Almen strip and the model area as indicated. The accuracy of the scaling introduced by Eq. (4-16) was confirmed by the close agreement obtained between two model runs, one having $l_{\text{Model}} = l_{\text{Strip}}$ and the other $l_{\text{Model}} << l_{\text{Strip}}$.

### 4.2.2.3. Process Model Results and Comparison with Experiments

Two measured impact velocity distributions were used in the model to simulate the finishing of an Almen strip at a particular location in the tub finisher. The predicted arc heights as a function of finishing time were then compared with earlier experimental data.
(Section 3). The velocity distributions were obtained from Ciampini et al. (2007b) where they were denoted as 91Sa and 112Sb and were applied to Almen strip targets of two thicknesses, 617 µm and 779 µm (Section 3). Therefore, four conditions were simulated, denoted as 91Sa617, 112Sb617, 91Sa779, and 112Sb779. The 91Sa condition was more energetic than the 112Sb condition, and thus had a higher average impact frequency and greater associated impact velocities.

The process model predicted an initial rapid increase in arc height to some maximum, but the agreement with the measured values as a function of time was generally poor (“2D model unscaled” points of Fig. 4-14). The trends suggested that the model overestimated the plastic deformation per impact. The model was then calibrated to the experimental data by multiplying the incident velocity of each contact in the measured impact velocity distribution by a constant factor for each of the four conditions. These scaled model predictions are shown in Fig. 4-14 using the given factors that were determined to give the best visual fit. The much improved agreement suggests that the model did indeed overestimate the degree of plastic deformation created per impact in the experiments. As explained previously, only the FEM deformation results using x direction tensile stresses are shown because the other FE and analytical methods produced similar results.
(b) 112Sb617

- Experiment
- 1D Model
- 2D Model, Scale Factor 0.23
- 2D Mode, Unscaled

Finishing Time (min)

Arc Height (μm)

(c) 91Sa779

- Experiment
- 1D Model
- 2D Model, Scale Factor 0.125
- 2D Model, Unscaled

Finishing Time (min)
Fig. 4-14: Comparison of experiment results and process model predictions for different vibratory finishing conditions: (a) 91Sa and 617 μm thick Almen strips (b), 112Sb and 617 μm thick Almen (c), 91Sa and 779 μm thick Almen strips (d), 112Sb and 779 μm thick Almen strips. The different model options and experimental data are identified by the legends in each sub figure. The error bars on the experimental data show the range of five repetitions. The 1D model is explained in the appendix.

The velocity scaling factors for conditions 91Sa617, 112Sb617, 91Sa779, and 112Sb779, were 0.4, 0.23, 0.125 and 0.15, respectively. These had the effect of decreasing the incident power to 16%, 5.3%, 1.6%, and 2.3% of the original incident power, respectively. The model required a greater reduction in incident power when simulating the thicker strips than the thinner strips. The factors for the 91Sa and 112Sb conditions applied to the 779 μm thick strips were relatively close, differing by 20%, and an acceptable fit to the data could have been obtained using an average value for both conditions. The factors for the 617 μm strips differed by 74% between the two conditions.
The vibratory finishing simulator described in Section 3.2.5 produced saturation curves which were used to compare against the present process model’s predictions. The agreement was not expected to be good because the model was not capable of accounting for the non-uniform impact location distribution of the vibratory finishing simulator. Three vibratory simulator conditions were compared: 0.22m/s impacts on 617 \( \mu m \) and 779 \( \mu m \) thick Almen strips, and 0.33m/s impacts on 617 \( \mu m \) thick Almen strips. The model’s predicted saturation curves and the vibratory finishing simulator’s saturation curves are shown together in Fig. 4-15.

The process model predicted an initial rapid increase in arc height to some maximum, but the agreement with the measured values as a function of time, as with the vibratory finisher conditions, was poor (“2D model unscaled” points of Fig. 4-15). This provided further support that the model overestimated the plastic deformation per impact.

Unlike in the experiments, saturation was predicted to occur very quickly; i.e. around 50 min. for the two 0.22 m/s conditions, and around 20min for the 0.33 m/s condition. The saturation arc height was characterized by having more curvature for the thin strips than for the thick strips; that is, \(-450 \mu m\) and \(-230 \mu m\) respectively. This did not agree well with the data, however, the unscaled curve’s relative trends are correct; that is, the more aggressive condition reached saturation arc heights in less finishing time than the less aggressive conditions. The thick condition showed saturation with less curvature than the thin conditions.

Velocity scaling factors were applied in the model to produce saturation curves with a good visual fit to the experimental data. The velocity scaling factors were 0.14 for 0.22 m/s impacts on a 617 \( \mu m \) thick strip, 0.08 for 0.22 m/s impacts on a 779 \( \mu m \) thick strip, and, 0.1 for 0.33 m/s impacts on a 617 \( \mu m \) thick strip. The fit using scaled model curves of Fig. 4-15 was not as good as seen for the vibratory finishing conditions. This was attributed to the non-uniform impact location distribution caused by the vibratory finisher simulator which the model could not account for.
Experiment 2D Model, Scale Factor 0.14
Experiment 2D Model, Unscaled

(a) 0.22 m/s, 617 μm

(b) 0.33 m/s, 617 μm
Fig. 4-15: Comparison of experiment results and process model predictions for different vibratory finishing simulator conditions: (a) 0.22 m/s and 617 μm thick Almen strips (b), 0.33 m/s and 617 μm thick Almen (c), 0.22 m/s and 779 μm thick Almen strips. The different model options and experimental data are identified by the legends in each subfigure. The error bars on the experimental data show ± three standard deviations of three repetitions.

The discrepancy between the rate of model arc height change and that observed in the experiments may be due to either limitations in the model or uncertainties in the processing and Almen strip boundary conditions during the experiments in the vibratory finisher. These are discussed in the following section.

4.2.2.4. Limitations of the Model

One possible explanation for the overestimation of the rate of change of strip curvature is the neglect of energy losses due to friction. However, the inclusion of friction in the axisymmetric FEA (Section 2.1.3.2) did not result in significantly different scale factors in the simulation.
The model did not account for elastic wave energy absorption by the Almen strip support that held the strip fixed within the finisher; however, it has been reported that significant amounts of the impact energy may be absorbed in elastic wave propagation (Seifried et al. 2005). The neglect of such energy losses would cause the incident energy to be overestimated and the predicted arc height to change too quickly. Although this may be a contributing cause, the differences in the scaling factors between the thick and thin strips suggests that other mechanisms may also be responsible; i.e. similar elastic wave propagation would occur in both strips.

A likely source of the overestimation of the plastic deformation per impact may be related to the model assumption that the back surface of the strip remained flat during finishing. This was the assumed boundary condition created by the vacuum holder used in the experiments (Section 3). However, after an impact occurs it was shown with another axisymmetric FE simulation that the Almen strip may lift off the support surface underneath the impact site. If such small-scale deformation occurred in spite of the vacuum force holding the strip down during the experiments, subsequent impacts would occur against a much more compliant surface leading to less plastic deformation and thus less energy absorption than would be the case if the back face of the strip remained perfectly flat. An attempt was made to observe small scale dimpling of the back face using a stylus profilometer, but the results were inconclusive. Such changes in support stiffness would have to be much greater than the 25% change in support structure modulus that was found to have little effect in Section 4.2.1.3.1.

It is logical to assume that the significance of small-scale lifting of the strip back-face would diminish as the impact velocity increased and the strip became thinner. In this case, impacts would tend to elastically flatten the strip to the support structure, thereby increasing the contact stiffness during the impact. The ratio of the elastic work required to flatten the strip with respect to the incident energy defines the severity of the lifting problem. This is why Almen strip boundary conditions are less important in shot peening; a relatively small fraction of the incident kinetic energy is required to flatten the strip against the stiff holder. Comparing the velocity scaling factor for the 91Sa617 combination (scaling factor of 0.4; thin strip, high energy) and that for 112Sb779 combination (scaling factor of 0.15; thick strip, less energy), it is seen that this hypothesis has some support; i.e. the reduction in the effective
impact velocity is smaller in the thin strip, high energy case, suggesting that the assumed rigid boundary condition is closer to reality than it is in the thick strip, lower energy case. Although this means that thinner Almen strips are less sensitive to lifting and are therefore preferable in vibratory finishing, such strips would have a smaller maximum arc height which would decrease the resolution of the measurements.

The vibratory finishing simulator’s scale factors were all similar in magnitude to each other and the thick Almen strip vibratory finishing conditions. The differences in the scale factors were too small to be considered significant as the non-uniform location distribution in the simulator did not allow for a good model curve fit. In addition, the goodness of fit was obtained through a subjective method that limits the accuracy of the scale factors. Therefore, comparisons between scale factors for the vibratory finishing simulator were not meaningful.

All the vibratory finishing simulator conditions were characterized as having a higher average impact velocity than even the most aggressive vibratory finishing conditions. This suggests that the model’s predictions for the simulator conditions should not have required the use of lower scale factors than for any of the vibratory finishing conditions; however, the opposite was observed. But, the simulator’s non-uniform impact location distribution may have caused its saturation curves to change less aggressively. In the simulator, the majority of the impacts were concentrated on the centre of the Almen strip. The unfinished edges constrained the strip against deflection caused by the more extensively finished central region. Saturation occurred in the central region before the edges, thus, the non-uniformity in the degree of finishing reduced the overall deflection of the Almen strip until the entire strip was finished to saturation. This loss of aggressiveness required smaller scale factors for the simulator than for the vibratory finisher. Thus, comparisons between the scale factors of the vibratory finisher and the simulator were not meaningful. Therefore, the simulator’s predictions and associated scale factors were inconclusive with regards to supporting the lifting hypothesis.

The axisymmetric FE impact analysis was computationally efficient, but could not model the repeated impact of spheres that do not overlap perfectly. Although the effect of this neglect of impact separation distance cannot be estimated, it was assumed that it would be relatively small after many thousands of overlapping impacts. This is supported by Guagliao (2001) who used a three-dimensional one-quarter symmetry model to show the
effect of a series of five shot peening impacts on the residual radial stress distribution along
the centerline of the first impact. Three of these impacts had varying degrees of separation;
however, they were still close enough so that they were partially overlapping. Only small
changes in the first impact’s centerline residual radial stresses occurred after the other
impacts. Further support for the use of repeated axisymmetric impacts in modeling randomly
located impacts is provided by Meguid et al. (1999), who showed that there are only small
differences in the stresses caused by four concentrically located impacts and four impacts
with separation. Moreover, both of these studies used a small number of overlapping impacts
and, as seen in Fig. 4-5, relatively large changes in residual stress occur during the first few
impacts. In a vibratory finisher, many thousands of impacts occur randomly each minute
creating an average result that is relatively uniform over the entire strip.

Another possible source of error, again due to the axisymmetric nature of the FEA, is the
shape of the dent profile (Fig. 4-4) which makes the contacts more conforming as the number
of impacts increases. In the actual process, if the target surface were perfectly smooth,
partially overlapping impacts would result in a rougher surface. However, as the process
progresses, the surface roughness caused by plastic deformation would be expected to reach a
maximum then decrease as the dent ridges are pounded down into the surface. This issue of
roughness effects has also not been addressed directly in shot-peening models, where it
would be of a greater concern than in vibratory finishing.

4.2.2.5. Almen strips as probes for determining vibratory
finishing process contact conditions

For any time on the saturation curve it is possible to determine the energy absorbed by
the Almen strip through Fig. 4-6 expressed on per unit area basis (using Fig. 4-11). However,
this would be limited to the media and the Almen strip configuration used in the model.
Although the absorbed energy can be determined in this way, it is only possible to determine
the energy that was incident on the strip surface in an approximate sense using an average
coefficient of restitution with Fig. 4-8 to determine the number of repeated impacts and the
velocity at which they occur. These can then be used to determine the incident energy which
should be taken to affect an area given by the impact radius (Eq. (10)). This provides an estimate of the incident energy per unit area of Almen strip.

The role of the impact velocity distribution in determining the saturation curve was revealed by the axisymmetric FEA. For a given impact velocity probability distribution, the average impact frequency determines the speed of the process. The maximum impact velocity plays an important role in determining the maximum obtainable arc height at saturation for a given Almen strip as was seen in Fig. 4-7(a). This figure also showed that the maximum arc height obtainable by a specific Almen strip type is determined by its thickness. Therefore, the maximum arc height achieved at saturation is a good indicator of the maximum impact velocity in a vibratory finisher. Figure 4-8 shows that as the strip is plastically deformed, the impact velocities that can create plastic deformation become progressively greater. This was the reason that the rate of arc height change was largest initially and decreased with increasing finishing time (Fig. 4-14). These observations suggest that the saturation arc height can provide an indirect measure of the maximum impact velocity in a vibratory finisher if the saturation arc heights of a given Almen strip can be calibrated using a series of known media impact velocities. This might be done using the electromagnet apparatus of Section 3 or the reciprocating pneumatic media launcher of Mohaherani et al. 2007, both of which bombard Almen strips with media at a known impact velocity.

Any application of the Almen strip as a probe must have a boundary condition that can be reproduced and verified to obtain repeatable results. The stiffness of the back face support has a large effect on the impact force and plastic deformation. It is desirable to fix the bottom of the strip; however, in practice it is difficult to prevent small-scale lifting as plastic deformation proceeds. The boundary condition can also play a significant role in shaping the stress distribution when the strip thickness is sufficiently small. The effect of uncertainties in the Almen strip boundary condition is much more significant in vibratory finishing than in shot peening, because of the much lower impact velocities.

4.3. Conclusions

Finite element analysis of the repeated impact and indentation of a ball at a single location on supported aluminum sheets has been used to simulate the development of a
steady state coefficient of restitution, the centerline radial residual stress distribution, the plastic work and the dent profile. This steady state condition was achieved with fewer impacts, and thus less energy absorbed, when a power law hardening model was used for the target, rather than an elastic perfectly-plastic model. The residual stresses under conditions of repeated overlapping impact were found not to be a function of the ball’s mass, but only a function of its radius and the energy absorbed by the strip prior to a given impact.

It was established that the coefficient of restitution of an impact depends only on the impact velocity and the previously absorbed energy of the target; i.e. the order and magnitude of previous impacts did not affect $e$. This is a useful result because it enables the total absorbed energy to be used as an independent variable characterizing the properties of the target surface after an arbitrary number of previous impacts over a range of velocities in the vibratory finishing process model.

The present process model gave the deformation of an Almen strip due to varying degrees of coverage of stochastically located impacts with random velocities. Two measured impact velocity distributions were used in the model to simulate the finishing of an Almen strip; one condition delivered more power than the other condition, i.e. it was more aggressive. Two thicknesses of Almen strip were tested for each finishing condition. The overall trends in the process were predicted very well by the model. That is, the 779 µm thick Almen strip curves always had less arc height for the corresponding condition than did the 617 µm thickness, and the more aggressive 91Sa saturation curves were correctly predicted to grow faster than those for the less aggressive 112Sb condition. However, the predicted saturation curves evolved faster than those obtained in the experiments. Good agreement was obtained after scaling down the incident velocities by a fixed factor for each experimental condition, suggesting that the model overestimated the stiffness of the strip support and hence the absorbed impact energy.

Almen strip curvature can be used to reveal details of the contact conditions in vibratory finishing. The maximum arc height achieved by several strips of varying thickness would provide a good indicator of the maximum impact velocity in a vibratory finisher. The energy incident on an Almen strip can be determined approximately using an average coefficient of restitution. The strip boundary condition can play a great role in determining the resulting stress distribution when the strip thickness is sufficiently small. As well, the effect of
uncertainties in the Almen strip boundary condition is much more significant in vibratory finishing than in shot peening, because of the much lower impact velocities.

4.4. Appendix

4.4.1. 1D process model

Earlier published models of Almen strip behaviour have adopted a one-dimensional approach to the simulation of the effect of residual stresses due to shot peening (Hill et al. 1982, Al-Obaid 1990, Cao et al. 1995, Guagliano 2001). The present work began with such a simplified model, using the axisymmetric FEA results of repeated impacts to determine the deformation of an Almen strip.

The curvature of the Almen strip was approximated by that of its centerline, 63.5 mm long divided into $10^5$ linear elements. The average frequency of impacts in the 1D process model corresponded to the frequency of impacts in the measured impact velocity signal. Impacts were stochastically located; however, the 1D model recorded only the chords of the impact zones that intersected the centerline linear elements in order to sum the energy changes. Therefore, the number of elements that were contacted by each impact depended on the contact area (and hence impact velocity) and the distance of the impact center from the linear element. To correct the bias that changes the apparent (sensed) velocity distribution along the centerline due to its zero area, each contact in the measured impact velocity signal was repeated with a random location and without incrementing the simulation time until its impact area intersected the centerline elements.

Each element intersected by a contact area had its absorbed energy updated using Fig. 4-8 according to the velocity of the impact and the average previously absorbed energy. The absorbed energy of each element was translated to an arc height using Fig. 4-6, and then to an equivalent bending moment. This was equivalent to applying the bending moment caused by the appropriate axisymmetric FEA centerline stresses to the contacted elements. The deformation of the entire centerline of elements was determined from an algorithm (described in detail in Section 7.1) which is equivalent to the Bernoulli-Euler theory using a finite difference method for the small deformations experienced in this study.
The Almen strip deflection predicted in this way (shown in Fig. 4-14) were scaled by a single factor of $2 \times 10^1$ for all four conditions to provide a reasonable fit to the experimental saturation curves. The 1D model captured the overall experimental trends; i.e. the 779 µm thick Almen strip has a smaller arc height than the 617 µm thickness for the same impact condition, and the more aggressive 91Sa saturation curves were always correctly predicted to grow faster than the less aggressive 112Sb condition. The agreement between the 1D model curves and the experimental curves is good for the 112Sb condition for both thicknesses. The agreement for the 91Sa condition is fair for both thicknesses. The model predicts the rise of arc height well in the first 10 min of finishing for the 779 µm thickness and 91Sa condition; however, it over predicts the arc height at later times. The agreement for the 617 µm and 91Sa condition is the poorest of the four curves. The rise of arc height in the first 10 min is predicted reasonably well; however, for times greater than 10 min, the arc height is under predicted.
5. Thesis Conclusions

5.1. Summary of Useful Findings

The effect of several process parameters were characterized: the location and orientation in the finisher, the finishing time, the media’s size and density, the total media mass in the finisher, coefficients of restitution and friction coefficients between pieces of media and between the media and the vibratory finisher’s container. The methods presented here are thought to be useful in characterizing the effect of other vibratory finishing parameters not varied in the present study, such as the media’s shape, the frequency of tub vibration, and the dimensions and shape of the finisher.

A device to measure media impact velocity has been developed to investigate the effect of varying process parameters and finisher design in order to facilitate the optimization of vibratory finishing processes.

The Almen system was adapted to vibratory finishing to quantify the effect of varying process parameters on the aggressiveness of the process. Certain elements of the usual shot peening Almen system were modified to reflect the much lower impact velocities found in most vibratory finishers. More aggressive finishing conditions, characterized by greater impact velocities, caused more plastic deformation and resulted in more Almen strip deflection in less finishing time. This supported the role of the Almen strip as a probe to characterize the aggressiveness of a vibratory finishing system in either process development or control.

An electromagnetic apparatus was constructed that showed that normal impacts alone produce saturation curves similar to those created in the vibratory finisher when impact velocities are comparable in magnitude. This provided support for normal impacts as the dominant mechanism of finishing.

Finite element analysis of the repeated impact and indentation of a ball at a single location on supported aluminum sheets has been used to simulate the development of a steady state coefficient of restitution, the centerline radial residual stress distribution, the plastic work and the dent profile.
A process model predicted the deformation of an Almen strip due to varying degrees of coverage of stochastically located impacts with random velocities. The overall trends in the process were predicted very well by the model. However, the predicted saturation curves evolved faster than those obtained in the experiments. Good agreement was obtained after scaling down the incident velocities by a fixed factor for each experimental condition, suggesting that the model overestimated the stiffness of the strip support and hence the absorbed impact energy.

Almen strip curvature can be used to reveal details of the contact conditions in vibratory finishing. The maximum arc height achieved by several strips of varying thickness would provide a good indicator of the maximum impact velocity in a vibratory finisher. The energy incident on an Almen strip can be determined approximately using an average coefficient of restitution. The strip boundary condition can play a great role in determining the resulting stress distribution when the strip thickness is sufficiently small. As well, the effect of uncertainties in the Almen strip boundary condition is much more significant in vibratory finishing than in shot peening, because of the much lower impact velocities.

5.2. Original Contributions

Most of the findings and methodologies presented in this work were novel; however, the major contributions are summarized below.

This work presented the first impact velocity measure of media in a vibratory finisher using a force sensor. This was a considerable improvement over the previous method of force signal characterization, as impact velocity measurements are independent of the sensor properties while impact force signals are not.

Previous attempts at adapting the Almen system to vibratory finishing were refined. A unique modified Almen gauge was presented that applied no force to the Almen strip while measuring its deflection thereby removing significant measurement error. A new vacuum attachment system for securing Almen strips during finishing was designed. It provided a more consistent boundary condition than the traditional Almen block; however, this work has shown that still greater improvements are necessary for its use in vibratory finishing. Almen strips of unprecedented tolerances for use in vibratory finishing were
manufactured. The importance of annealing the Almen strips was revealed. A method of obtaining maximum impact velocities from a process using the modified Almen system was described.

A new modeling approach that relied on Axisymmetric FEA of repeated impacts and a separate program that used the FEA data to drive a stochastic impact model was presented. This approach overcame the computational difficulties, unique to vibratory finishing of modelling the very small effect of more than $10^5$ impacts. It is the first model of its kind and it provided original data such as the evolution of residual stress and Almen strip curvature as a function of finishing. In addition, it revealed many aspects of the mechanics of vibratory finishing process for residual stress creation.

Although there are many similarities of vibratory finishing to shot peening, vibratory finishing is a far more subtle process. An unprecedented level of sensitivity was achieved to successfully measure and model the sub-millimetre midspan deflections of Almen strips and the high numbers of very low velocity impacts that cause them.

5.3. Industrial and Research Applications

Although the present study was designed to investigate the vibratory finishing process, there are many related processes that involve impact and the interaction of a large number of solid bodies; e.g. ball milling, tumbling, centrifugal disk finishing, peen forming and shot peening. The present study is useful to researchers who are modeling these processes as it presents an approach for dealing with this type of multi-body impact problem.

This work is particularly applicable to peen forming since the emphasis was on the deflection of Almen strip targets which are similar to the sheet metal targets that are usual shaped by peen forming. In addition, this study was applicable to spherical media which is generally used in peen forming. The major difference is the speed range of media impact as it is much higher in peen forming.

Although surface mechanical attrition (SMA) has been used in the past to create a nanocrystalline surface layer, the findings of the present study suggest that process parameters can be controlled to encourage the generation of this nanocrystalline structure to provide the necessary high plastic strain and strain rates. Specifically, the repeated plastic
deformation of a workpiece surface is maximized using smooth spherical media with a high density, high mass, high impact velocity, high impact frequencies, and long finishing times. The present work showed that, for a given process, the maximum impact velocity in the process controlled the maximum amount of energy absorbed by the Almen strip. If the radius of the media is unchanged, then the maximum incident energy of the particles in a process control the energy absorbed by the target in plastic work. Therefore, maximizing the incident energy of the media is of major importance in creating a highly-deformed surface layer.

5.4. Recommendations for Future Work: Improvements and Research

Completing this work has made evident several possible improvements. The electromagnet apparatus of the present study was not able to produce a uniform impact distribution because of its nonuniform magnetic field. An apparatus has been designed by Mohajerani et al. (2007) to remedy this shortcoming.

An improvement in the consistency of the boundary condition provided by the vacuum attachment system was found to be desirable. An issue whereby the Almen strip locally lifts off the surface of the attachment system is thought to exist that causes a large increase in contact compliance. This introduces uncertainty in the boundary conditions that frustrates the modeling effort. Perhaps an adhesive to attach the strip and a solvent to remove it would provide a more consistent boundary condition. This solution would have the drawback of being very time consuming.

The use of long slender structures should be avoided as a means of attaching velocity sensors and Almen strips for finishing. Such structures may store incident impact energy as elastic strain energy; therefore, they introduce more uncertainty in the actual Almen boundary conditions. In practice, however, it is difficult to avoid long slender structures because they are ideal for submerging objects in the vibratory finishing flow.

Although many improvements to the Almen strip’s tolerances were made, a tightening of the thickness and initial flatness tolerances would improve the Almen system repeatability.
Greater efficiency in future projects involving measuring impact velocities and finishing Almen strips could be achieved by combining the impact velocity sensor and the Almen vacuum holder allowing these tasks to be done simultaneously. In addition, it would be beneficial to add another slot to the vacuum attachment system to finish two strips at one time. This system has the added benefit of avoiding the repositioning of the system for velocity measurement and Almen strip finishing. Also, any anomalous events that occur during finishing of the strips could be recorded by the velocity sensor. This provides an additional security that conditions in the finisher were constant throughout the finishing.

The approach of relying on the axisymmetric FEA data to drive a stochastic process model was chosen as the alternative to a full 3D stochastic FEA for several reasons. It was a reasonable starting point as it followed from the approach of some analytical models in the literature that relied on solution of analytical stress fields from one impact; that is, it was logical to refine this approach by considering additional impacts at the same location. The axisymmetric FEA approach also revealed details of the process mechanics that would have been concealed by a full 3D FEA process analysis.

Several FE shot peening models in the literature relied on symmetry that was axisymmetric, although, they chose to model relatively few impacts to obtain an achievable computational requirement. Impacts in shot peening carry much more incident energy than in vibratory finishing. Therefore, it is insufficient to model relatively few vibratory finishing impacts to simulate the process as each impact has a comparatively small effect. In this study 1700 axisymmetric impacts were modeled per experimental condition to obtain data for the process model. The process model then was able to determine the effect greatly over $10^5$ impacts on a small surface region. Although it seems close to impossible at this time, as computer power increases over time, performing a 3D FEA of the process should become economically feasible.

Although it was suggested in the shot peening literature that the effect of partially overlapping impacts was small, it would be desirable to have such a model to explicitly study their effect at the relatively high number of low velocity impacts specific to vibratory finishing. This proposed research study would study an aspect of the vibratory finishing process mechanics that was inaccessible using the present approach.
Although the 3D FEA would require much more computational power, the actual implementation of the code would be much simpler than the present code which required the use of the data of the axisymmetric FEA in a process model.
6. References


Hatch, J.E., 1984. Aluminum: Properties and physical metallurgy, ASM International


7. Appendix

7.1. An algorithm for determining the deformation of a beam given the curvature of the elements it is comprised of

The deformation of the entire line of elements and hence the deformation of the Almen strip was determined from the constant curvature of each element by enforcing displacement continuity and first derivative continuity between adjacent elements at the shared nodes. Fig. 7-1(a) shows three elements and a hypothetical curvature that they have developed by being impacted. These three elements represent a simpler system than the much larger collection of elements used in the actual process model. This simple system was used to illustrate the method of obtaining a smooth curve from a collection of elements given their curvatures.

Fig. 7-1 (a) A simple three element domain prior to the imposition of first derivative continuity at the nodes. The arc heights are exaggerated to make the degree of curvature noticeable; i.e. the figure is not to scale.
Initially the element’s nodes were all arranged in a line. A coordinate system’s origin was placed at the first node of the first element. The orientation of the coordinate system is defined such that the x-axis intersects the nodes of all the elements. The elements were designated by whole numbers starting at “1” for the first element (i.e. $e = 1$, where $e$ is the element number) and continued sequentially up to the total number of elements (i.e. $e = N_e$).

It was helpful to imagine a fictitious stationary element ($e=0$) in the derivation of the equations in this section used to anchor the other elements. The stationary anchor element had no initial arc height ($A_0 = 0$) and was oriented collinear with the $x$-axis. The $x$-$y$ position of its second node was located at the origin (i.e. $\vec{n}_1 = (0,0)$) and the position of its first node is at $\vec{n}_0 = (-L,0)$.

The constant curvature of each element was expressed in terms of the arc height ($A$) at the midspan of each element and was denoted as $A_1$, $A_2$, and $A_3$ for the element 1, 2, and 3 respectively. Fig. 7-1 (a) shows the centers of curvature for the first, second and third elements: $\Theta_1$, $\Theta_2$, and $\Theta_3$, respectively. When the elements were positioned collinear with the $x$-axis, the smallest angle between the $y$-axis and the chord that joins either node of an element and its center of curvature was defined as $\gamma_e$; where $e$ is the element number. The value of $\gamma_e$ can be obtained from eq. (7-1) which was stated in terms of the length of the element ($L$), and the element’s radius of curvature ($\Gamma_e$) (which can be obtained from eq. (4-8) in terms of the length of the element ($L$) and the arc height ($A$)).

$$\gamma_e = \sin^{-1} \left( \frac{L}{2\Gamma_e} \right)$$

(7-1)

The values of $\gamma_e$ were used to obtain $\beta_e$. $\beta_e$ was the angle of rotation the element’s second node must be rotated about its first node to make the slope at its first node equal to the slope of the second node of the previous element. The equation that was used to obtain $\beta_e$ was selected from Table 7-1. This choice was based on the arc height of the previous element ($A_{e-1}$) and of the current element ($A_e$).
Table 7-1: Angles of rotation to enforce first derivative continuity. Where \( A \) is the arc height over a 63.5 mm (2.5") span.

<table>
<thead>
<tr>
<th>( A_e )</th>
<th>( A_e&gt;0 )</th>
<th>( A_e=0 )</th>
<th>( A_e&lt;0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_{e-1} &gt; 0 )</td>
<td>( \beta_e = -\gamma_e - \gamma_{e-1} )</td>
<td>( \beta_e = -\gamma_e )</td>
<td>( \beta_e = \gamma_e - \gamma_{e-1} )</td>
</tr>
<tr>
<td>( A_{e-1} = 0 )</td>
<td>( \beta_e = -\gamma_e )</td>
<td>( \beta_e = 0 )</td>
<td>( \beta_e = \gamma_e )</td>
</tr>
<tr>
<td>( A_{e-1} &lt; 0 )</td>
<td>( \beta_e = -\gamma_e + \gamma_{e-1} )</td>
<td>( \beta_e = \gamma_{e-1} )</td>
<td>( \beta_e = \gamma_e + \gamma_{e-1} )</td>
</tr>
</tbody>
</table>

The rotation of each element about its first node (i.e. \( \beta_e \)) was expressed relative to the global coordinate system. This was defined as, \( \psi_e \), and is given by Eq. (7-2).

\[
\psi_e = \sum_{i=1}^{e} \beta_i
\]  

(7-2)

The process of calculating values of \( \psi_e \) is illustrated in Fig. 7-1 (b) for a simplified three-element mesh. The first of three elements had \( A_0 = 0 \) and \( A_1 > 0 \), therefore, Table 7-1 gives \( \beta_1 = -\gamma_1 \). \( A_1 > 0 \) and \( A_2 > 0 \) therefore, \( \beta_2 = -\gamma_2 - \gamma_1 \). \( A_2 > 0 \) and \( A_3 < 0 \) therefore, \( \beta_3 = \gamma_3 - \gamma_2 \). Thus eq. (7-2) gives \( \psi_1 = -\gamma_1 \), \( \psi_2 = -2\gamma_1 - \gamma_2 \) and \( \psi_2 = -2\gamma_1 - 2\gamma_2 + \gamma_3 \). The resulting element rotations are shown in Fig. 7-1 (b).
Fig. 7-1(b): The same elements shown in part (a) after they have been detached at the nodes and rotated about their first node by an angle $\psi_e$ to make the slopes at the shared nodes (shown in part (a)) equal.

In general, the act of rotating each element by $\psi_e$ detaches the elements at their nodes. A smooth curve was created by reattaching the elements while keeping their new orientation intact (Fig. 7-1(c)). The location of each node relative to the coordinate system shown in Fig. 7-1 (a) was given by Eq. (7-3).

$$\vec{n}_e = (L \cos \psi_e, L \sin \psi_e) + \vec{n}_{e-1} \tag{7-3}$$

Given that the curvature of each element was small, the nodes in the positions defined by eq. (7-3) give a set of points that approximated the deflected shape of an Almen strip. The arc height at approximately the center of this curve ($A_m$) over the span defined by the first node at the origin and the last node of the last element is given by eqs. (7-4) and (7-5).

$$\theta = \tan^{-1} \left( -\frac{n_{y,e=N_e}}{n_{x,e=N_e}} \right) \tag{7-4}$$

$$A_m = n_{x,e=N_e/2} \sin(\theta) + n_{y,e=N_e/2} \cos(\theta) \tag{7-5}$$

This is shown in Fig. 7-1(c), however, because it is only made with a few elements with extreme curvatures, it is only a very rough approximation. The actual simulation used $10^5$ elements each with only a small arc height, yielding an accurate prediction.
Fig. 7-1(c): The same elements shown in part (b) after they linearly translated such that shared nodes are once again coincident. The result is a smooth curve obtained from the arc heights of the elements shown in part (a). $A_m$ is shown here as the approximate overall arc height at the center of the 3 element structure.

This algorithm has been shown to provide deflection solutions very close to solutions of the Euler-Bernoulli beam theory for small deflections solved using finite difference methods. It is suspected that this algorithm provides more accurate solutions for large deflections than the Euler-Bernoulli beam theory.
7.2. Computer Code

7.2.1. Data Acquisition and Digital Signal Processing

Force signal acquisition was performed on the PCI-6071E data acquisition. In addition, NI-DAQ software driver was used to interface with the board. The NI-DAQ is a library of high and low level functions that can be called from a programming language to use the data acquisition board. The software was programmed using this NI-DAQ library and MS Visual C++ v6.

This board is capable of taking a single analog signal and converting it to a digital signal with a 12-bit resolution at a maximum sample rate of 1.25 MHz. If multiple channels are required, then the board must switch between channels (using its 64 channel multiplexer) to take samples, because it only has one analog to digital converter. It does this as quickly as possible and samples once from each of the channels then stops until the end of the “scan” cycle. At the start of a new scan cycle, the board repeats this process. Therefore, the data acquisition rate for each channel is the scan frequency.

Reading from more than one channel on the PCI-6071E has the disadvantage that the sample rate is reduced by multiplexing and by the setting time that must be allowed after switching channels.

The PCI-6071E board is capable of buffered input and unbuffered input. Unbuffered input mode reads a single value from the data acquisition board and returns it to the controlling program. Acquiring several samples successively would generate a signal; however, the time between samples would vary quite a bit because the timing is controlled by the software. Therefore, this method is only suitable for signals that change very slowly. A buffer is a piece of memory that can contain several values. In this case the buffer can only store a fixed number of these values; the actual number of values the buffer can hold is determined before the data acquisition software is executed. If buffered mode is employed for data acquisition the PCI-6071E board uses DMA (direct memory access) to write its acquired data to a buffer. The PCI-6071E board does this automatically without intervention from software once the process is initiated by software. Data acquisition is performed using hardware timing; that is, the data acquisition board very accurately controls the time between acquired samples unlike unbuffered mode. Once data acquisition is complete and the buffer
is full of data, the software is free to save the contents of the buffer to a file. The buffer size is limited by the amount of host computer memory; therefore, it is incapable of collecting data continuously. To continuously acquire data a technique called double buffering is required that is similar to buffered data acquisition except in the way the buffer is read. Double buffering also relies on a fixed length buffer. However, unlike buffered data acquisition, half of the buffer is being copied by the software to the hard drive while the other half is being filled by the data acquisition board. The board constantly fills the buffer with acquired data; when it reaches the end of the buffer it simply continues to write newly acquired data at the beginning of the buffer without any pause. The software follows behind and reads the data before it is overwritten by the data acquisition hardware. The host computer must be fast enough to keep up with the data acquisition board or the software will not run fast enough to remove data from the buffer before it is overwritten by the data acquisition board.

Three simple win32 console programs were written for the data acquisition to meet the requirements of three different applications. In each program the gain for each channel was chosen so that the full 12-bit resolution of the board would be used and so that the input signal would exceed the maximum measurable voltage. The data collected by these programs have units of the physical quantities being measured; therefore the difficulties in interpreting the signals have been eliminated.

The first program that was created made use of double-buffering to continuously retrieve data from two channels each receiving input from an accelerometer. This program outputs a file that contains the signals from both accelerometers in units of $m/s^2$. The sample frequency is also saved to a file. Thus, the file contains all the information needed to interpret the signal.

A second program was created that reads data from four channels to get input from four force sensors. This program relies on buffered input and thus has limited data acquisition capability over a fixed period of time.

The third program is for reading data from the normal force sensor. The maximum sample rate of the third program is much higher than that which the second program can obtain since it is using several channels simultaneously. This program also relies on buffered input.
7.2.1.1. Force Sensor Data Acquisition

/*******************************************************************************
 * NI-DAQ PCI-6071E Data Acquisition Program For Dytran 1051V1
 * Based on Example program:
 * DAQsingleBufAsync.c
 * Description:
 * Read a waveform from one analog input channel using internal timing (uses low-level NI-DAQ functions)
 * List of NI-DAQ Functions used in this example:
 * DAQ_Rate, DAQ_Start, NIDAQErrorHandler, DAQ_Check, NIDAQYield,
 * DAQ_VScale, DAQ_Clear
 *****************************************************************************/

#define BUFFERSIZE 15000000  //max:16777216 min:3
#define CONV_SLOPE 8.6134779240
#include "nidaqex.h"
#include <math.h>
#include <iostream.h>

double res(const int sampletimebase)
{
    double retVal=0;
    switch( sampletimebase )
    {
    case -3:
        retVal=50*pow(10,-9);
        break;
    case -1:
        retVal=200*pow(10,-9);
        break;
    case 1:
        retVal=1*pow(10,-6);
        break;
    case 2:
        retVal=10*pow(10,-6);
        break;
    case 3:
        retVal=100*pow(10,-6);
        break;
    case 4:
        retVal=1*pow(10,-3);
        break;
    case 5:
        retVal=10*pow(10,-3);
        break;
    default:
        retVal=0;
    }
    return retVal;
}

int main(void)
{
    double retVal=0;
    switch( sampletimebase )
    {
    case -3:
        retVal=50*pow(10,-9);
        break;
    case -1:
        retVal=200*pow(10,-9);
        break;
    case 1:
        retVal=1*pow(10,-6);
        break;
    case 2:
        retVal=10*pow(10,-6);
        break;
    case 3:
        retVal=100*pow(10,-6);
        break;
    case 4:
        retVal=1*pow(10,-3);
        break;
    case 5:
        retVal=10*pow(10,-3);
        break;
    default:
        retVal=0;
    }
    return retVal;
}

int main(void)
{
    double retVal=0;
    switch( sampletimebase )
    {
    case -3:
        retVal=50*pow(10,-9);
        break;
    case -1:
        retVal=200*pow(10,-9);
        break;
    case 1:
        retVal=1*pow(10,-6);
        break;
    case 2:
        retVal=10*pow(10,-6);
        break;
    case 3:
        retVal=100*pow(10,-6);
        break;
    case 4:
        retVal=1*pow(10,-3);
        break;
    case 5:
        retVal=10*pow(10,-3);
        break;
    default:
        retVal=0;
    }
    return retVal;
}

int main(void)
{
    double retVal=0;
    switch( sampletimebase )
    {
    case -3:
        retVal=50*pow(10,-9);
        break;
    case -1:
        retVal=200*pow(10,-9);
        break;
    case 1:
        retVal=1*pow(10,-6);
        break;
    case 2:
        retVal=10*pow(10,-6);
        break;
    case 3:
        retVal=100*pow(10,-6);
        break;
    case 4:
        retVal=1*pow(10,-3);
        break;
    case 5:
        retVal=10*pow(10,-3);
        break;
    default:
        retVal=0;
    }
    return retVal;
}
delete[] piBuffer;
    return -1;
}

u32 iDAQstopped = 0;
ulRetrieved = 0;
iIgnoreWarning = 0;

FILE* dataFilePtr;
int i;
double actualSamplePeriodSec = 0.0;

/* Convert sample rate (S/sec) to appropriate timebase and sample
interval values. */
/* NOTE: If you are using a DSA device, call DAQ_Set_Clock
instead. Refer to NI-DAQ Function Reference Manual for details. */
iStatus = DAQ_Clear(iDevice);

/* Acquire data from a single channel */
printf("NORMAL FORCE SENSOR HIGH SPEED DATA ACQUISITION\n");
printf("One sample will be taken every %f seconds.\n",res(iSampTB)*uSampInt);
printf("A total of %d samples will be taken.\n",BUFFERSIZE);
printf("Approximate time required for data acquisition is \%d seconds.\n",BUFFERSIZE*res(iSampTB)*uSampInt);
getchar();
printf("Press any key to abort data aquisition.\n");
iStatus = DAQ_Start(iDevice, iChan, iGain, piBuffer, ulCount, iSampTB, uSampInt);
if (0!=iStatus)
{
    iStatus = DAQ_Clear(iDevice);
delete[] piBuffer;
    delete[] piVoltBuffer;
    iRetVal = NIDAQErrorHandler(iStatus, "DAQ_Start", iIgnoreWarning);
    printf("Abnormal Exit: Problem with Dac board\n");
    exit(0);
}
while ((iDAQstopped != 1) && (iStatus == 0))
{
    /* Loop until all acquisition is complete. HINT: You can be
doing other foreground tasks during this time. */
    if (0!=iStatus)
    {
        iStatus = DAQ_Clear(iDevice);
delete[] piBuffer;
delete[] piVoltBuffer;
iRetVal = NIDAQErrorHandler(iStatus, "DAQ_Start", iIgnoreWarning);
        printf("Abnormal Exit: Problem with Dac board\n");
        exit(0);
    }
    //iRetVal = NIDAQYield(iYieldON);
    if (0 != _kbhit())
    {
        printf("ABORT REQUESTED!\n");
        iStatus = DAQ_Clear(iDevice);
delete[] piBuffer;
delete[] piVoltBuffer;
        exit(0);
    }
    iStatus = DAQ_VScale(iDevice, iChan, iGain, dGainAdjust, dOffset,
    ulCount, piBuffer, piVoltBuffer);
iRetVal = NIDAQErrorHandler(iStatus, "DAQ_VScale", iIgnoreWarning);
iStatus = DAQ_Clear(iDevice);
printf("Data aquisition complete! Dumping buffer to a file...\n");
if( (dataFilePtr = fopen( "nfdatD.csv", "w+" )) == NULL )
{
    printf( "A new data file could not be opened\n" );
delete[] piBuffer;
delete[] piVoltBuffer;
    exit(0);
}
    actualSamplePeriodSec = res(iSampTB)*uSampInt;
    while((i=0) < BUFFERSIZE)
    {
        fprintf(dataFilePtr, "%f
", -1*CONV_SLOPE*pdVoltBuffer[i] );
        if (0 != _kbhit())
        {
            printf("ABORT REQUESTED!
");
            break;
        }
    }
    /* Close file */
    if( fclose(dataFilePtr) )
}
/* OFFSET - to determine offset take the average input from 
// the file that is written by the polling code with no load on sensor. 
FILE* offsetFilePtr;
if( (offsetFilePtr = fopen( "nfOffset.csv", "w+" )) == NULL )
{
    printf( "The file 'nfOffset.csv' was not opened\n" );
    exit(0);
}
for(i=0; i < BUFFERSIZE; i++)
{
    fprintf(offsetFilePtr, "%d\n", piBuffer[i] );
}
if( fclose( dataFilePtr ) )
    printf( "The file 'nfOffset.csv' was not closed\n" );

*/

printf("Finished\n");
delete[] piBuffer;
delete[] pdVoltBuffer;
return 0;

} /* End of program */

7.2.1.2. Multiple Accelerometers Data Acquisition

#define BUFFERSIZE 5000
#define HALFBUFFERSIZE 2500   //HALFBUFFERSIZE=BUFFERSIZE/2
#define NUMBER_OF_CHANNELS 2
#define ACCEL_DUE_TO_GRAVITY 9.81
#include "nidaqex.h"
#include <stdio.h>
#include <conio.h>

int main(void)
{
    /*
    * Local Variable Declarations:
    *
    */
    i16 iStatus = 0;
    i16 iRetVal = 0;
    i16 iDevice = 1;
    i16 iChan = 1;
    i16 iGain = 1;
    f64 dSampRate = 20000.0;  //10000
    f64 dScanRate = 10000.0;  //1000
    u32 ulCount = HALFBUFFERSIZE*NUMBER_OF_CHANNELS;
    i16 iLoopCount = 0;
    i16 iHalfBufsToRead = 1;
    f64 dGainAdjust = 1.0;
    f64 dOffset = 0.0;
    i16 iUnits = 0;
    i16 iSampTh = -3;
    i16 iScanTh = -3;
    u16 uSampInt = 0;
    static i16 piBuffer[BUFFERSIZE] = {0};
    static f64 pdVoltBuffer[BUFFERSIZE] = {0.0};
    static i16 piHalfBuffer[HALFBUFFERSIZE] = {0};
    i16 iHalfReady = 0;
    i16 iDAQstopped = 0;
    u32 ulPtsTfr = 0;
    u32 ulRetrieved = 0;
    i16 iNumMuxBrds = 0;
    i16 iNumChans = 2;
    static i16 piChanVect[NUMBER_OF_CHANNELS] = {5, 6};
    static i16 piGainVect[NUMBER_OF_CHANNELS] = {20, 20};
const static f64 metersPerSecondSquaredPerVolt[NUM_CHANNELS] = \{(1000*ACCEL_DUE_TO_GRAVITY)/10.10, (1000*ACCEL_DUE_TO_GRAVITY)/10.20 \};

FILE* dataFilePtr;
int i,j,k=0;
f64 voltage=0;

acceleration = voltage * metersPerSecondSquaredPerVolt[j];
if (j!=NUMCHANS-1) {
    printf(dataFilePtr, "%f," , acceleration);
} else {
    printf(dataFilePtr, "%f\n", acceleration);
}

if ((dataFilePtr = fopen("adata.csv", "w")) == NULL) {
    printf("The file 'data2' was not opened\n");
    exit(0);
}

DAQ_DB_Config (iDevice, TRUE); /* enable double buffering */

DAQ_DB_Transfer(iDevice, piHalfBuffer, &ulPtsTfr, &iDAQstopped);
iRetVal = NIDAQErrorHandler(iStatus, "DAQ_VScale", iIgnoreWarning);

/* CLEANUP - Don't check for errors on purpose. */
iStatus = DAQ_Clear(iDevice);

/* Close file */
if( fclose( dataFilePtr ) )
    printf( "The file 'data' was not closed\n" );
return 0;

/* End of program */

7.2.2. Signal Processing Code

Mathworks Inc. Matlab v7 Code
clc;
clear;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% User Interface %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
colormap gray;
path='C:\_DOCS\PhD\THESIS\SensorDat\Force Paper Signal Figure\';
filename='nfdatD - HILL2 - Porc 6mm media - 3.75in from high wall - Centered between side walls - facing high wall - 1.csv';
mediaType = '6mm Porcelain    ';
%mediaType = '1/4" Carbon Steel';
% '6mm Porcelain    '
% '1/4" Carbon Steel'
% 'Ceramic          '
disp(mediaType);

normalforceDataSignature='nfdatD*.csv';
batchMode=false;
plotSignal=true;
allGraphs=false;
consideredDeletedEnds=false;
% noise band around zero is +-0.2 N
noiseBandHalfWidth=0.3;
datumDriftHalfWidth=0.5;
minimumForcePeakToBeConsideredContact=0.5;
minimumContactTime=3/100000; % seconds
maximumContactTime=0.02; % seconds
minimumQuasiStaticContactTime = minimumContactTime; % seconds
ringingDecayTime=0.01; % seconds
ringingDecaySecondsPerNewtonOfForceSpike=1/600; % measured
ringingDecaySecondsPerNewtonOfForceSpike=1/600;
ringingPeriodMaximum=0.00003*2;
maxImpactContactTime=55/2/1000000;
minImpactContactTime=15/(1000000*4);
minimumImpactForcePeakMagnitude=1;

secondsOfProcessingPerByteOfInputData=6.9118e-005;
secondsOfProcessingPerSampleNumber=0.0067486;

numberOfPointsPerFigure=5000000;
numOfOverLapPointsPerFig=0;
numbOfBinsInHistograms=20;

startTime = clock;
disp(['Contact Identification Program Started at: ',...
     num2str(startTime(4)),':',num2str(startTime(5))]);

% enumerations start
iStart=1;
iEnd=2;
iContactTime=3;
iImpulse=4;
iForceMaximum=5;
iContactType=6; % 0=Not Classified, 1=impact, 2=quasi-static
iImpactVelocity=7;
iIsContact=8; %iIsContact should always correspond to the highest number %since it is stripped from the data file
iTimeToNextContact=iIsContact;
iTimeToNextSameTypeContact=iTimeToNextContact+1;
cTypeNoClassification=0;
cTypeImpact=1;
cTypeQuasiStatic=2;
%enumerations end
typeNoClassificationCount=0;
typeImpactCount=0;
typeQuasiStaticCount=0;
typeNoClassificationTimeCount=0;
typeImpactTimeCount=0;
typeQuasiStaticTimeCount=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
files = dir([path,normalForceDataSignature]);
numberOfFilesInBatchQueue=length(files);
totalDataSizeBytes=0;
if batchMode==true
    disp('In Batch Mode. Processing the following input files:');
    for batch=1:numberOfFilesInBatchQueue
        ignoreFileTest1 = strfind(files(batch).name,'_Contact_Data.csv');
        ignoreFileTest2 = strfind(files(batch).name,'_Vel_Hist_Dat.csv');
        if (length(ignoreFileTest1)==0)&&(length(ignoreFileTest2)==0)
            disp(files(batch).name);
            totalDataSizeBytes = totalDataSizeBytes + files(batch).bytes;
        end
    end
    disp(['Batch Mode Completion Time is Approximately:','
    ',num2str(ceil(secondsOfProcessingPerByteOfInputData*totalDataSizeBytes/60), ' min.']);
else
    numberOfFilesInBatchQueue=1;
end
for batch=1:numberOfFilesInBatchQueue
    if batchMode==true
        filename=files(batch).name;
    end
    ignoreFileTest1 = strfind(filename,'_Contact_Data.csv');
    ignoreFileTest2 = strfind(filename,'_Vel_Hist_Dat.csv');
    if (length(ignoreFileTest1)==0)&&(length(ignoreFileTest2)==0)
        disp(['Data File Loading... ',num2str(etime(clock, startTime)/60,3),' min']);
        %for quick data set changes.
        normalForceData=0;
        normalForceData=double(dlmread([path,filename], ',', 1,0)); %read all file %file's data is already in newtons.
        numberOfSamples = length(normalForceData);
        [samplePeriod,count] = fscanf(fid,'%f',1); %%format as of July 17 2003
        minimumContactTime=samplePeriod;
        disp(['This Data File Will Take Approximately ','
        ',num2str(ceil(1/secondsOfProcessingPerSampleNumber)*numberOfSamples/60), ' min. to Process.']);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        disp(['Signal Conditioning...',num2str(etime(clock, startTime)/60,3),' min']);
        %find datum and correct for it since it drifts
        datum=0;
        n=0;
        for i=1:1:numberOfSamples
            if ((normalForceData(i)<datumDriftHalfWidth)&&(normalForceData(i) > -1*datumDriftHalfWidth))
                datum=normalForceData(i)+datum;
                n=n+1;
            end
        end
datum=datum/n;
        for i=1:1:numberOfSamples
            normalForceData(i)=normalForceData(i)-datum;
        end
        %make all points within noise band zero
        for i=1:1:numberOfSamples
            if ((normalForceData(i)<noiseBandHalfWidth*0.5)&&(normalForceData(i) > -1*noiseBandHalfWidth*0.5))
                normalForceData(i)=0;
            end
        end
        if (numberOfSamples <= 2)
            disp('Error - 7');
            return;
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        disp(['Find Zero Crossings... ','
        ',num2str(etime(clock, startTime)/60,3),' min']);
        %find datum and correct for it since it drifts
        datum=0;
        n=0;
        for i=1:1:numberOfSamples
            if ((normalForceData(i)<datumDriftHalfWidth)&&(normalForceData(i) > -1*datumDriftHalfWidth))
                datum=normalForceData(i)+datum;
                n=n+1;
            end
        end
datum=datum/n;
        for i=1:1:numberOfSamples
            normalForceData(i)=normalForceData(i)-datum;
        end
        %make all points within noise band zero
        for i=1:1:numberOfSamples
            if ((normalForceData(i)<noiseBandHalfWidth*0.5)&&(normalForceData(i) > -1*noiseBandHalfWidth*0.5))
                normalForceData(i)=0;
            end
        end
        if (numberOfSamples <= 2)
            disp('Error - 7');
            return;
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
contactData = double(0); contactToggle = false;
for i=2:numberOfSamples
    contactStart = false;
    contactEnd = false;
    normalForceDataPrev = double(normalForceData(i-1));
    normalForceDataCurr = normalForceData(i);
    if ((normalForceData(i-1)>0.0)&& (normalForceData(i)<0.0))
        contactEnd = true;
        contactStart = true;
        contactData(i,iStart) = i-1;
        contactToggle = true;
    end
    if ((normalForceData(i-1)<0.0) && (normalForceData(i)>0.0))
        contactEnd = true;
        contactStart = true;
        contactData(i,iStart) = i;
        contactToggle = true;
    end
    if ((true==contactStart) && (false==contactToggle))
        contactData(i,iEnd) = i-1;
        contactToggle = true;
    end
    if ((true==contactEnd) && (true==contactToggle))
        contactData(i,iEnd) = i-1;
        contactToggle = false;
        i = i + 1;
    end
end
numberOfContacts = length(contactData);
if (numberOfContacts==1)
    disp('ERROR: No Contacts Found');
else
    disp(['Performing Initial Contact Calculations for Contact Classification...',num2str(etime(clock, startTime)/60,3),' min']);
    firstContact = true;
    for i=1:(numberOfContacts-1)
        numOfSamplesInInt = contactData(i,iEnd)-contactData(i,iStart)+1;
        fs = contactData(i,iStart);
        fe = contactData(i,iEnd);
        forceSignalOverInt = normalForceData(contactData(i,iStart):contactData(i,iEnd)-1);
        impulse = samplePeriod*trapz(forceSignalOverInt);
        contactData(i,iImpulse) = impulse;
        contactData(i,iContactTime) = numOfSamplesInInt*samplePeriod;
        maxValue = 0;
        maxIndex = 0;
        for p=contactData(i,iStart):contactData(i,iEnd)
            if normalForceData(p) > maxValue
                maxValue = normalForceData(p);
                maxIndex = p;
            end
        end
        if maxIndex == 0
            disp('no max or min value found in contact! Program may crash!');
        end
        contactData(i,iForceMaximum) = normalForceData(maxIndex);
        clear forceSignalOverInt;
        clear timeAxisInInt;
    end
    disp(['Classifying Contacts...',num2str(etime(clock, startTime)/60,3),' min']);
    for i=1:numberOfContacts
        contactData(i,iIsContact) = 1; %assume whole signal is good
        contactData(i,iContactType) = cTypeQuasiStatic; %THIS IS WHERE THE NO-CLASSIFICATION WAS ELIMINATED
        typeNoClassificationCount = typeNoClassificationCount + 1;
        typeNoClassificationTimeCount = contactData(i,iContactTime) + typeNoClassificationTimeCount;
    end
end
for i=1:numberOfContacts
    if contactData(i,iIsContact) == 0
        if (contactData(i,iImpulse) <= 0)
contactData(i,iIsContact)=0;
end
if (contactData(i,iForceMaximum) <= noiseBandHalfWidth*2)
    contactData(i,iIsContact)=0;
end
if (contactData(i,1) <= abs(minimumForcePeakToBeConsideredContact))
    contactData(i,1)=0;
end
if (contactData(i,1,1) <= contactData(i,1,1)-1)
    contactData(i,1,1)=0;
end
if (contactData(i,1,ContactTime)<minimumContactTime)
    contactData(i,1,IsContact)=0;
end
if (contactData(i,1,ContactTime) >= minimumQuasiStaticContactTime)
    if (contactData(i,1,ContactTime) < maximumContactTime)
        contactData(i,1,ContactType)=cTypeQuasiStatic;
        typeQuasiStaticCount=typeQuasiStaticCount+1;
        typeQuasiStaticTimeCount=contactData(i,1,ContactTime)+typeQuasiStaticTimeCount;
    end
end
j=i;
startOfImpactSampleNumber=0;
impactForceMaximum=0;
if (j < numberOfContacts)
    if ((contactData(j,1,Impulse)>0)&&(contactData(j,1,ContactTime)<=maxImpactContactTime))&&...
        (contactData(j+1,1,ContactTime)<ringingPeriodMaximum))&&...
            {contactData(j+1,1,Start)=contactData(j+1,1,End)}*samplePeriod*ringingPeriodMaximum*0.5)
        startOfImpactSampleNumber = contactData(j,1,Start);
        %make sure shape of contact is correct.
        isImpact=true;
    end
    if contactData(j,1,End) < numberOfSamples
        if (contactData(j,1,End)-contactData(j,1,Start))>2
            maxValue=-1;
            maxIndex=0;
            for p=contactData(j,1,Start):contactData(j,1,End)
                if normalForceData(p) > maxValue
                    maxValue=normalForceData(p);
                    maxIndex=p;
                end
            end
            if (maxIndex > contactData(j,1,End)||[maxIndex < contactData(j,1,Start)]
                disp('ERROR!');
                return;
            end
            for p=contactData(j,1,Start):maxIndex-1
                if normalForceData(p) > normalForceData(p+1)
                    isImpact=false;
                end
            end
            for p=maxIndex+1:contactData(j,1,End)
                if normalForceData(p+1) >= normalForceData(p)
                    isImpact=false;
                end
            end
            if (isImpact==true)
                if (contactData(j,1,ForceMaximum)>= minumumImpactForcePeakMagnitude)
                    contactData(j,1,ContactType)=cTypeImpact;
                    typeImpactCount=typeImpactCount+1;
                    typeImpactTimeCount=contactData(j,1,ContactTime)+typeImpactTimeCount;
                    startOfImpactSampleNumber=contactData(j,1,End);
                    j=j+1;
                    while (j<numberOfContacts)
                        vibrationEnd=true;
                        if ((contactData(j,1,ContactTime)<ringingPeriodMaximum) && (contactData(j+1,1,ContactTime)<ringingPeriodMaximum))
                            if ((contactData(j,1,Impulse)>0) && (contactData(j+1,1,Impulse)<0)) || (contactData(j,1,Impulse)<0) && (contactData(j+1,1,Impulse)>0)
                                vibrationEnd=true;
                        end
                        timeBetweenContacts = (contactData(j+1,1,End)-contactData(j+1,1,Start))*samplePeriod;
                        if timeBetweenContacts < ringingPeriodMaximum
                            if vibrationEnd==true
                                timeBetweenContacts = (contactData(j,1,End)-contactData(j,1,Start))*samplePeriod;
                                if timeBetweenContacts < ringingPeriodMaximum
                                    vibrationEnd=false;
                                end
                            end
                        end
                        ringingTime = (contactData(j,1,End)-startOfImpactSampleNumber)*samplePeriod;
                        if (ringingTime <= ringingDecayTime)
vibrationEnd=false;
end
if (vibrationEnd==true)
  break;
else
  contactData(j,iIsContact)=0;
  contactData(j+1,iIsContact)=0;
  j=j+1;
end
end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp(['Finding Impact Velocities from contact data...',num2str(etime(clock, startTime )/60,3),' min']);
for i=1:(numberOfContacts-1)
  if mediaType=='1/4" Carbon Steel' %confirmed correct settings
    contactData(i,iContactType)==cTypeQuasiStatic)||(contactData(i,iContactType)==cTypeNoClassification))
    contactData(i,iImpactVelocity)=0.005804*contactData(i,iForceMaximum);
  end
  if (contactData(i,iContactType)==cTypeImpact)
    contactData(i,iImpactVelocity)=291.57*contactData(i,iImpulse);
  end
  elseif mediaType=='6mm Porcelain ' %confirmed correct settings
    contactData(i,iImpactVelocity)=0.01243*contactData(i,iForceMaximum);
  end
  elseif mediaType=='Ceramic          '
    disp('ERROR: No Such Media Type');
    return;
  end
  end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp(['Filtering out Garbage Contacts...',num2str(etime(clock, startTime )/60,3),' min']);
realContactsData=0;
j=0;
lastProgress=0;
numberOfColumsInRealContactDataArray=length(contactData(1,:))-1;
for i=1:numberOfContacts
  if (contactData(i,iIsContact)==1)
    j=j+1;
    realContactsData(j,1:numberOfColumsInRealContactDataArray)=...
    contactData(i,1:numberOfColumsInRealContactDataArray);
  end
end
numberOfRealContacts=length(realContactsData(:,1));
clear contactData;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp(['Generating Additional Contact Data Following Contact Classification...',num2str(etime(clock, startTime )/60,3),' min']);
realContactsData(:,iTimeToNextContact)=zeros(numberOfRealContacts,1);
realContactsData(:,iTimeToNextSameTypeContact)=zeros(numberOfRealContacts,1);
for i=1:numberOfRealContacts-1
  timeToNextContact = ( (realContactsData(i+1,iStart)-1) - (realContactsData(i,iEnd)+1) ) * samplePeriod;
  if timeToNextContact > 0
    realContactsData(i,iTimeToNextContact) = timeToNextContact;
  else
    realContactsData(i,iTimeToNextContact) = 0;
  end
  for j=(i+1):numberOfRealContacts
    if realContactsData(i,iContactType)==realContactsData(j,iContactType)
      timeToNextSameTypeContact=((realContactsData(j,iStart)-1) - (realContactsData(i,iEnd)+1))*samplePeriod;
      if timeToNextSameTypeContact > 0
        realContactsData(i,iTimeToNextSameTypeContact)=timeToNextSameTypeContact;
      else
        realContactsData(i,iTimeToNextSameTypeContact)=0;
      end
    end
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp(['Writing Contact Data to a File...',num2str(etime(clock, startTime )/60,3),' min']);
fileId=fopen([path,filename,'_Contact_Data.csv'],'w');printf(fileId,'%s,%s,%s,%s,%s,%s,%s,%s,%s
','Start index',...
numberOfRealContacts = length(realContactsData(:,1));
for i=1:numberOfRealContacts
fprintf(fileId,'%d,%d,%E,%E,%E,%d,%E,%E,%E
',realContactsData(i,:));
end
fclose(fileId);

disp(['Creating Histograms...',num2str(etime(clock, startTime )/60,3),' min']);
numberOfColsInRealContactData = length(realContactsData(1,:));
numberOfQuasiStaticContacts=0;
numberOfImpactContacts=0;
numberOfUnclassifiedContacts=0;
for i=1:numberOfRealContacts
if realContactsData(i,iContactType)==cTypeQuasiStatic
numberOfQuasiStaticContacts=numberOfQuasiStaticContacts+1;
end
if realContactsData(i,iContactType)==cTypeImpact
numberOfImpactContacts=numberOfImpactContacts+1;
end
if realContactsData(i,iContactType)==cTypeNoClassification
numberOfUnclassifiedContacts=numberOfUnclassifiedContacts+1;
end
end
quasiStaticContactData=zeros(numberOfQuasiStaticContacts,numberOfColsInRealContactData);
impactContactData=zeros(numberOfImpactContacts,numberOfColsInRealContactData);
unclassifiedContactData=zeros(numberOfUnclassifiedContacts,numberOfColsInRealContactData);
m=0;
n=0;
p=0;
for i=1:numberOfRealContacts
if realContactsData(i,iContactType)==cTypeQuasiStatic
m=m+1;
quasiStaticContactData(m,:)=realContactsData(i,:);
end
if realContactsData(i,iContactType)==cTypeImpact
n=n+1;
impactContactData(n,:)=realContactsData(i,:);
end
if realContactsData(i,iContactType)==cTypeNoClassification
p=p+1;
unclassifiedContactData(p,:)=realContactsData(i,:);
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
filenameTitle = strrep(filename,'_',' '); set(gcf,'Renderer','OpenGL'); clf reset; cla reset; tempPlot = newplot; axis tight; hist(realContactsData(:,iForceMaximum),numOfBinsInHistograms); title(filenameTitle); xlabel('Bin: Force Maximums (N)'); ylabel(['Histogram: Occurances in each Bin']); tempPlot=gcf; saveas(tempPlot,[path,filename,'_Force_Max','.fig','','fig']); saveas(tempPlot,[path,filename,'_Force_Max','.png','','png']);
if allGraphs==true
clf reset;
cla reset;
tempPlot = newplot; axis tight; hist(impactContactData(:,iForceMaximum),numOfBinsInHistograms); title(filenameTitle); xlabel('Bin: Impact Force Maximums (N)'); ylabel(['Histogram: Occurances in each Bin']); tempPlot=gcf; saveas(tempPlot,[path,filename,'_Hist_Impact_Force_Max','.fig','','fig']); saveas(tempPlot,[path,filename,'_Hist_Impact_Force_Max','.png','','png']);
clf reset;
cla reset;
tempPlot = newplot; axis tight; hist(quasiStaticContactData(:,iForceMaximum),numOfBinsInHistograms); title(filenameTitle); xlabel('Bin: Quasi-Static Force Maximums (N)'); ylabel(['Histogram: Occurances in each Bin']); tempPlot=gcf; saveas(tempPlot,[path,filename,'_Hist_Stat_Force_Max','.fig','','fig>');
saveas(tempPlot,[path,filename,'_Hist_Stat_Force_Max','.png'],'png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(impactContactData(:,iImpactVelocity),numOfBinsInHistograms);
title(filenameTitle);
xlabel('Bin: Impact Velocity of impact type (m/s)');
ylabel(['Histogram: Occurances in each Bin']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Impact_Vel','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Impact_Vel','.png'],'png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(quasiStaticContactData(:,iImpactVelocity),numOfBinsInHistograms);
title(filenameTitle);
xlabel('Bin: Quasi-Static Effective Velocity (m/s)');
ylabel(['Histogram: Occurances in each Bin']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Static_Vel','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Static_Vel','.png'],'png');
eend

clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(realContactsData(:,iImpactVelocity),numOfBinsInHistograms);
title(filenameTitle);
xlabel('Bin: Effective Impact Velocity (all types) (m/s)');
ylabel(['Histogram: Occurances in each Bin']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Eff_Impact_Vel','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Eff_Impact_Vel','.png'],'png');

binEnd = max(realContactsData(:,iImpactVelocity));
binStart = min(realContactsData(:,iImpactVelocity));
binInc = (binEnd - binStart)/numOfBinsInHistograms;
binVector = binStart:binInc:binEnd;
histogramCount = histc(realContactsData(:,iImpactVelocity),binVector);
histoGramDataForExport(1,:)=binVector;
histoGramDataForExport(2,:)=histogramCount.';
dlmwrite([path,filename,'_Vel_Hist_Dat.csv'],histoGramDataForExport,',');

if allGraphs==true

clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(impactContactData(:,iContactTime),numOfBinsInHistograms);
title(filenameTitle);
xlabel('Bin: Impact Contact Time (sec)');
ylabel(['Histogram: Occurances in each Bin']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Impact_Contact_Period','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Impact_Contact_Period','.png'],'png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(quasiStaticContactData(:,iContactTime),numOfBinsInHistograms);
title(filenameTitle);
xlabel('Bin: Quasi-Static Contact Time (sec)');
ylabel(['Histogram: Occurances in each Bin']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Static_Contact_Period','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Static_Contact_Period','.png'],'png');
eend

clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(realContactsData(:,iContactTime),numOfBinsInHistograms);
title(filenameTitle);
xlabel('Bin: Contact Time (sec)');
ylabel(['Histogram: Occurances in each Bin']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Cont_Period','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Cont_Period','.png'],'png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;

hist(realContactsData(:,iTimeToNextContact),numOfBinsInHistograms*5);
title(filenameTitle);
xlabel('Bin: time to next contact (all types) (s)');
ylabel(['Histogram: Occurances in each Bin']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Cont_Period','.fig'],"fig");
saveas(tempPlot,[path,filename,'_Hist_Cont_Period','.png'],"png");
clf reset;
cla reset;
tempPlot = newplot;
title(filenameTitle);
xlabel('Number Of Contacts by Type');
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_PIE_NumOfCon','.fig'],"fig");
saveas(tempPlot,[path,filename,'_PIE_NumOfCon','.png'],"png");

clear quasiStaticContactData;
clear impactContactData;
clear unclassifiedContactData;
if plotSignal==false
    clear normalForceData;
    clear realContactsData;
else
    disp(['Separating Signal According to Contact Type for Signal Graphing Operation...']);
    num2str(etime(clock, startTime )/60,3),' min']);

m=0;
numberOfRealContacts = length(realContactsData(:,1));

for i=1:numberOfRealContacts
    if realContactsData(i,iContactType)==0
        for j=realContactsData(i,iStart):realContactsData(i,iEnd)
            k=k+1;
        end
    end
    if realContactsData(i,iContactType)==1
        for j=realContactsData(i,iStart):realContactsData(i,iEnd)
            m=m+1;
        end
    end
    if realContactsData(i,iContactType)==2
        for j=realContactsData(i,iStart):realContactsData(i,iEnd)
            n=n+1;
        end
    end
end
end
if consideredDeletedEnds==true
    numberOfRealContacts = length(realContactsData(:,1));
    if realContactsData(i,iEnd)~=numberOfSamples
        end
    end
end
if consideredDeletedEnds==true
    for i=1:numberOfRealContacts
        if realContactsData(i,iContactType)==0
            for j=realContactsData(i,iStart):realContactsData(i,iEnd)
                k=k+1;
            end
        end
        if realContactsData(i,iContactType)==1
            for j=realContactsData(i,iStart):realContactsData(i,iEnd)
                m=m+1;
            end
        end
        if realContactsData(i,iContactType)==2
            for j=realContactsData(i,iStart):realContactsData(i,iEnd)
                n=n+1;
            end
        end
    end
end
if consideredDeletedEnds==true
    numberOfRealContacts = length(realContactsData(:,1));
    if realContactsData(i,iEnd)~=numberOfSamples
        end
    end
end
if consideredDeletedEnds==true
    numberO
for j=realContactsData(numberOfRealContacts,iEnd)+1:numberOfSamples
p=p+1;
end
end

cVForceSignalDeleted=zeros(p,2);
cVForceSignalUnclassified=zeros(k,2);
cVForceSignalImpact=zeros(m,2);
cVForceSignalQuasiStatic=zeros(n,2);

k=0;
m=0;
n=0;
p=0;

% considerDeletedEnds-true
if consideredDeletedEnds==true
    for i=1:1:1
    p=p+1;
cVForceSignalDeleted(p,1) = j;
cVForceSignalDeleted(p,2) = normalForceData(j);
    end
end

for i=1:numberOfRealContacts
    if realContactsData(1,iStart)==0
        for j=realContactsData(1,iStart):realContactsData(1,iEnd)
            k=k+1;
cVForceSignalUnclassified(k,1) = j;
cVForceSignalUnclassified(k,2) = normalForceData(j);
        end
    end
    if realContactsData(1,iContactType)==1
        for j=realContactsData(1,iStart):realContactsData(1,iEnd)
            m=m+1;
cVForceSignalImpact(m,1) = j;
cVForceSignalImpact(m,2) = normalForceData(j);
        end
    end
    if realContactsData(1,iContactType)==2
        for j=realContactsData(1,iStart):realContactsData(1,iEnd)
            n=n+1;
cVForceSignalQuasiStatic(n,1) = j;
cVForceSignalQuasiStatic(n,2) = normalForceData(j);
        end
    end
    if (i ~= numberOfRealContacts)
        if (realContactsData(i,iEnd)~=realContactsData(i+1,iStart)+1)
            if (realContactsData(i+1,iStart)-1)-(realContactsData(i,iEnd)+1) > 0
                for j=realContactsData(i,iEnd)+1:realContactsData(i+1,iStart)-1
                    p=p+1;
cVForceSignalDeleted(p,1) = j;
cVForceSignalDeleted(p,2) = normalForceData(j);
                end
            end
        end
    end
end

%% if mod(i,10)==0
%%    progress=floor(i*100/numberOfRealContacts);
%%    disp(['Progress ',num2str(progress),'% Complete ','num2str(etime(clock, startTime )/60,3),' min']);
%% end

if consideredDeletedEnds==true
    if realContactsData(numberOfRealContacts,iEnd)~=numberOfSamples
        for j=realContactsData(numberOfRealContacts,iEnd)+1:numberOfSamples
            p=p+1;
cVForceSignalDeleted(p,1) = j;
cVForceSignalDeleted(p,2) = normalForceData(j);
        end
    end
end

clear normalForceData;
clear realContactsData;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:(length(zerosOfForceSignal)-1)
    numOfSamplesInInt=zerosOfForceSignal(i+1)-zerosOfForceSignal(i);
    timeAxisInInt(1)=samplePeriod*zerosOfForceSignal(i);
    for j=2:(numOfSamplesInInt)
        timeAxisInInt(j)= timeAxisInInt(j-1)+ samplePeriod;
    end
    forceSignalOverInt = normalForceData(zerosOfForceSignal(i):zerosOfForceSignal(i+1)-1);
    tempPlot=plot(timeAxisInInt,forceSignalOverInt);
    title('Raw Normal Force signal');
    xlabel('Time seconds');
    ylabel('Force (N)');
    saveas(tempPlot,[path,filename,num2str(i),'_t.png'](),'png');
clear tempPlot;
clear forceSignalOverInt;
clear timeAxisInInt;
clear num2str;
disp(['Graphing Colour Coded Signal..', num2str(etime(clock, startTime) / 60, 3), ' min']);
numberOfCuts = ceil(numberOfSamples/numberOfPointsPerFigure);
for i=1:numberOfCuts
    overlapS = 0;
    if i>1
        overlapS = numOverlapPointsPerFig;
    end
    if i==numberOfCuts
        overlapE = numOverlapPointsPerFig;
    end
    if (numberOfCuts==1)
        s=1;
        e=numberOfSamples;
    else
        s=numberOfPointsPerFigure*(i-1)+1-overlapS;
        e=numberOfPointsPerFigure*i+overlapE;
    end
    clf reset;
    cla reset;
    set(gcf,'Renderer','OpenGL');
    tempPlot = newplot;
    axis tight;
    hold on;
    if length(cVForceSignalQuasiStatic(:,1))>1
        cVs=1;
        cVe=1;
        cVsNotFound=true;
        cVeNotFound=true;
        for j=1:length(cVForceSignalQuasiStatic(:,1))
            if (cVForceSignalQuasiStatic(j,1) >= s) && (cVsNotFound==true)
                cVs=j;
                cVsNotFound=false;
            end
            if (cVForceSignalQuasiStatic(j,1) >= e) && (cVeNotFound==true)
                cVe=j-1;
                cVeNotFound=false;
            end
        end
        if cVeNotFound==true
            cVe=length(cVForceSignalQuasiStatic(:,1));
        end
        if (cVs<e) && (cVs<cVe) && (cVsNotFound==false)
            plot(cVForceSignalQuasiStatic(cVs:cVe,1)*samplePeriod,cVForceSignalQuasiStatic(cVs:cVe,2),'.g','MarkerSize',1);
        end
    end
    if length(cVForceSignalImpact(:,1))>1
        cVs=1;
        cVe=1;
        cVsNotFound=true;
        cVeNotFound=true;
        for j=1:length(cVForceSignalImpact(:,1))
            if (cVForceSignalImpact(j,1) >= s) && (cVsNotFound==true)
                cVs=j;
                cVsNotFound=false;
            end
            if (cVForceSignalImpact(j,1) >= e) && (cVeNotFound==true)
                cVe=j-1;
                cVeNotFound=false;
            end
        end
        if cVeNotFound==true
            cVe=length(cVForceSignalImpact(:,1));
        end
        if (cVs<e) && (cVs<cVe) && (cVsNotFound==false)
            plot(cVForceSignalImpact(cVs:cVe,1)*samplePeriod,cVForceSignalImpact(cVs:cVe,2),'.b','MarkerSize',3);
        end
    end
    if length(cVForceSignalUnclassified(:,1))>1
        cVs=1;
        cVe=1;
        cVsNotFound=true;
        cVeNotFound=true;
        for j=1:length(cVForceSignalUnclassified(:,1))
            if (cVForceSignalUnclassified(j,1) >= s) && (cVsNotFound==true)
                cVs=j;
                cVsNotFound=false;
            end
            if (cVForceSignalUnclassified(j,1) >= e) && (cVeNotFound==true)
                cVe=j-1;
                cVeNotFound=false;
            end
        end
        if cVeNotFound==true
            cVe=length(cVForceSignalUnclassified(:,1));
        end
        if (cVs<e) && (cVs<cVe) && (cVsNotFound==false)
            plot(cVForceSignalUnclassified(cVs:cVe,1)*samplePeriod,cVForceSignalUnclassified(cVs:cVe,2),'.c','MarkerSize',1);
        end
    end
end
end
end
if length(cVForceSignalDeleted(:,1))>1
  cVs=1;
cVe=1;
cVsNotFound=true;
cVeNotFound=true;
for j=1:length(cVForceSignalDeleted(:,1))
  if (cVForceSignalDeleted(j,1) >= s) && (cVsNotFound==true)
    cVs=j;
cVsNotFound=false;
  end
  if (cVForceSignalDeleted(j,1) >= e) && (cVeNotFound==true)
    cVe=j-1;
cVeNotFound=false;
  end
end
if cVeNotFound==true
  cVe=length(cVForceSignalDeleted(:,1));
end
if (cVs<e) && (cVs<cVe) && (cVsNotFound==false)
  plot(cVForceSignalDeleted(cVs:cVe,1)*samplePeriod,cVForceSignalDeleted(cVs:cVe,2),'.r','MarkerSize',1);
end
end
filenameTitle = strrep(filename,'_',' ');%title(["Fixed Dytran Normal Force - ", num2str(i) ,' of ', num2str(numberOfCuts), filenameTitle]);
%legend('Quasi-Static','Impact','Deleted','Unclassified');
xlim([s*samplePeriod e*samplePeriod]);
title(['Force Signal: Red=Delete, Blue=Impact Green=Static Cyan=Unclassifed']);
ylabel(['Time(sec) SamplePeriod=',num2str(samplePeriod),', plot:', num2str(i) ', of',
num2str(numberOfCuts)]);
ylabel(['Force(N)   ', filenameTitle]);
saveas(tempPlot,[path,filename,'_ContactTypes',num2str(i),'.fig']);
saveas(tempPlot,[path,filename,'_ContactTypes',num2str(i),'.png'],'png');
clear tempPlot;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear cVForceSignalDeleted;
clear cVForceSignalUnclassified;
clear cVForceSignalImpact;
clear cVForceSignalQuasiStatic;
%clear normalForceData;
%clear realContactsData;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if batchMode==true
  disp(['Finished Batch! ' ,num2str(etime(clock, startTime )/60,3),' min']);
disp(['Time Per Byte to Run in sec: ', num2str(etime(clock, startTime)/totalDataSizeBytes)]);
end
%output = 'Done';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fid = fopen([path,filename]);
[samplePeriod,count] = fscanf(fid,'%f',1); %format as of July 17 2003
status = fclose(fid);
% samplePeriod=1/10000; %debug

sampleFreqHz = 1/samplePeriod;
numberOfFFTPoints = numberOfSamples;
yFFT = fft(y,numberOfFFTPoints);
clear y;

Pyy = yFFT.* conj(yFFT) / numberOfFFTPoints;
f = double(sampleFreqHz*(0:(numberOfFFTPoints/2))/numberOfFFTPoints);

if graph==true
    semilogx(f,Pyy(1:floor(numberOfFFTPoints/2)+1));
    grid on;
    title(['Power Spectrum - ', num2str(filename)]);
    xlabel('Frequency (Hz)');
    ylabel('Power (W)');
    print('-dtiff', '-r600', [path,filename,'_PowerSpectrum','.tif']);
end

disp(localMaxima);
len=length(powerSpectrum(:,1));
for k=1:len
    if powerSpectrum(k,localMaximaFreqIndex) > filterAwayPeaksBelowThisFrequency
        foundLowerLimit=false;
        for w=localMaxima(w,localMaximaFreqIndex):frequencyInHzPlusOrMinusIntegrationLimits
            if foundLowerLimit==false
                k=1;\n                powerSpectrumIndexIntervals(interval)=k-1;
                break;
            end\n        end
    end
end

powerCarriedByMaxima=0;
powerCarriedByOtherPortionsOfSignal=0;
for interval=1:numberOfLocalMaxima
    upperInterval=powerSpectrumIndexIntervals(interval+1);
    lowerInterval=powerSpectrumIndexIntervals(interval);
    power=trapz(powerSpectrum(lowerInterval:upperInterval,localMaximaPowerIndex));
    if mod(interval,2)==0 % even
        powerCarriedByMaxima=power+powerCarriedByMaxima;
    else % odd
        powerCarriedByOtherPortionsOfSignal=power+powerCarriedByOtherPortionsOfSignal;
    end
end

140
7.2.4. Bust and Peak Analysis

Mathworks Inc. Matlab v7 Code

clc;
clear all;
disp('Started Burst Isolation and Analysis');

path='C:\_DOCS\PhD\THESIS\_BurstData\Porcelain\';
%path='C:\_DOCS\PhD\THESIS\_BurstData\Steel\';
%path='C:\_DOCS\PhD\THESIS\_BurstData\Debug\';
forceSignalFilename='nfdatD Porcelain 1.75 frm HW Facing HW Flush button - 01.csv';
%forceSignalFilename='nfdatD 200 lb Steel 1.75 frm HW Facing HW Flush button - 01.csv';
%forceSignalFilename='nfdatD 200 lb Steel 1.75 frm HW Facing HW Flush button - 01.csv_CUT.csv';
contactDataFilename=[forceSignalFilename,'_Contact_Data.csv'];
%contactDataFilename='nfdatD 200 lb Steel 1.75 frm HW Facing HW Flush button - 01.csv_Contact_Data.csv_CUT.csv';
forceFFTFrequency=true;  %if 'true' use fft freq. as a starting point for trial and error algorithm to find burst length
disp(path);
disp(forceSignalFilename);
disp(contactDataFilename);

%enumerations start
iStart=1;
iEnd=2;
iContactTime=3;
iImpulse=4;
iForceMaximum=5;
iContactType=6;  % 0=Not Classified, 1=impact, 2=quasi-static
iImpactVelocity=7;
iTimeToNextContact=8;
iTimeToNextSameTypeContact=9;
cTypeNoClassification=0;
cTypeImpact=1;
cTypeQuasiStatic=2;

%FFT
numberOfLocalMaxima=10;
filterAwayPeaksBelowThisFrequency=5;
frequencyInRangePlusOrMinusIntegrationLimits=1;
graph=false;
y=double(dlmread([path,forceSignalFilename], ',', 1,0)); %read all file %file's data is already in newtons.
numberOfSamples = length(y); %number of samples
fid = fopen([path,forceSignalFilename]);
%[samplePeriod,count] = fscanf(fid,'%f',1);  %%format as of July 17 2003
status = fclose(fid);
sampleFreqHz = 1/samplePeriod;
numberOfFFTPoints = numberOfSamples;
yFFT = fft(y,numberOfFFTPoints);
clear y;
Pyy = yFFT.* conj(yFFT) / numberOfFFTPoints;

localMaximaPowerIndex=1;
localMaximaFreqIndex=2;

len=length(powerSpectrum(:,1));
k=0;
while k<=numberOfLocalMaxima
    if powerSpectrum(len,localMaximaFreqIndex) >= filterAwayPeaksBelowThisFrequency
        k=k+1;
        localMaxima(k,localMaximaPowerIndex)= powerSpectrum(len,localMaximaPowerIndex);
        localMaxima(k,localMaximaFreqIndex) = powerSpectrum(len,localMaximaFreqIndex);
    end
    len=len-1;
end

clear powerSpectrum;
clear len;
tubDrivingFrequency=localMaxima(1,localMaximaFreqIndex);
disp(tubDrivingFrequency);
clear localMaximaPowerIndex;
clear localMaximaFreqIndex;
clear len;
clear SortedPowerSpectrum;
clear localMaxima;

if notLockedIntoRightFrequency=true;
    if (tubDrivingFrequency > 48)||(tubDrivingFrequency < 45)||(forceFFTFrequency==true)
        disp('Not using Frequency from FFT');
        maxFrequency=47;
        minFrequency=45;
        approximateFinisherPeriodInSeconds=1/46; %porcelain
        numberOfFrequencyScans=1000; % should be 1000
        if forceFFTFrequency==true
            maxFrequency=tubDrivingFrequency+0.05;
            minFrequency=tubDrivingFrequency-0.05;
            approximateFinisherPeriodInSeconds=1/tubDrivingFrequency; %
            numberOfFrequencyScans=100;
        end
    end
    if minFrequency >= maxFrequency
        return;
    end
    minCycleTimeInSeconds = 1/maxFrequency;
    maxCycleTimeInSeconds = 1/minFrequency;
    scanningPeriod = (maxCycleTimeInSeconds-minCycleTimeInSeconds)/numberOfFrequencyScans;
end

else
    approximateFinisherPeriodInSeconds=1/tubDrivingFrequency;
    %maxFrequency=tubDrivingFrequency*0.01;
    %minFrequency=tubDrivingFrequency-0.01;
    %numberOfFrequencyScans=1;
    notLockedIntoRightFrequency=false;
end

% Algorithm to find start of burst as the area
numberOfSections=100;
refinementComplete=false;
lastLowestMinSum=0;

normalForceData=double(dlmread([path,forceSignalFilename], ',', 1,0));
numberOfSamplesInForceSignal=1.5*10e6;
realContactsData = csvread([path,contactDataFilename], 1, 0);
numberOfRealContacts = length(realContactsData(:,1));
fid = fopen([path,forceSignalFilename]);
[samplePeriod,count] = fscanf(fid,'%f',1); %format as of July 17 2003
status = fclose(fid);

numberOfSamplesInForceSignal/samplePeriod;

numberOfFrequencyLockingAttempts=0;
refinementComplete=False;
lastLowestMinSum=0;
lastHighestMaxSum=0;
offsetInSamples=0;

approximateFinisherPeriodInSamples = int32(round(approximateFinisherPeriodInSeconds/samplePeriod));
sectionLengthInSamples=int32(round(approximateFinisherPeriodInSamples/numberOfSections));
nNumberOfBursts=int32(floor(numberOfSamplesInForceSignal/approximateFinisherPeriodInSamples));

if notLockedIntoRightFrequency==true
approximateFinisherPeriodInSeconds=minCycleTimeInSeconds;
end

while ((notLockedIntoRightFrequency==true)||(refinementComplete==false))

% sum up impulse in each section across bursts
if notLockedIntoRightFrequency==true

sectionLengthInSamples = int32(round(approximateFinisherPeriodInSamples/numberOfSections));
approximateFinisherPeriodInSamples = int32(round(approximateFinisherPeriodInSeconds/samplePeriod));
nNumberOfBursts=int32(floor(numberOfSamplesInForceSignal/approximateFinisherPeriodInSamples));
end
clear sectionSums;
sectionSums=zeros(numberOfSections,1);
for sectionIndex=1:numberOfSections

for burstIndex=1:numberOfBursts

sectionStart=int32((burstIndex-1)*approximateFinisherPeriodInSamples + ...
(sectionIndex-1)*sectionLengthInSamples + offsetInSamples);

for contactIndex=1:numberOfRealContacts

contactStart = int32(realContactsData(contactIndex,iStart));
if (contactStart>=sectionStart)&&(contactStart<=sectionEnd)

%%contact start is in section
%%therefore add it to this section sum

sectionSums(sectionIndex)=realContactsData(contactIndex,iForceMaximum)+sectionSums(sectionIndex);
end
end
end

% find minimum sectionIndex
minSumSectionIndex=1;
maxSumSectionIndex=1;
minSum=sectionSums(minSumSectionIndex);
maxSum=sectionSums(maxSumSectionIndex);
for sectionIndex=1:numberOfSections

if sectionSums(sectionIndex)>maxSum

maxSumSectionIndex=sectionIndex;
maxSum=sectionSums(maxSumSectionIndex);
end

if sectionSums(sectionIndex)<minSum

minSumSectionIndex=sectionIndex;
minSum=sectionSums(minSumSectionIndex);
end
end

if notLockedIntoRightFrequency==true

numberOfFrequencyLockingAttempts=numberOfFrequencyLockingAttempts+1;
if numberOfFrequencyLockingAttempts==1

bestBurstPeriodFound=approximateFinisherPeriodInSeconds;
lastLowestMinSum=minSum;
lastHighestMaxSum=maxSum;
end
if ((maxSum-minSum) > (lastHighestMaxSum-lastLowestMinSum))

bestBurstPeriodFound=approximateFinisherPeriodInSeconds;
lastLowestMinSum=minSum;
lastHighestMaxSum=maxSum;

frequency=1/bestBurstPeriodFound;
disp('----------------------------------------');
disp('found better frequency');
disp(frequency);
disp(sectionSums);
disp(minSum);
disp(maxSum);
end
approximateFinisherPeriodInSeconds=approximateFinisherPeriodInSeconds+scanningPeriod;
if approximateFinisherPeriodInSeconds > maxCycleTimeInSeconds
notLockedIntoRightFrequency=false;
approximateFinisherPeriodInSeconds=bestBurstPeriodFound;
approximateFinisherPeriodInSamples = int32(round(approximateFinisherPeriodInSeconds/samplePeriod));
sectionLengthInSamples=int32(round(approximateFinisherPeriodInSamples/numberOfSections));
nNumberOfBursts=int32(floor(numberOfSamplesInForceSignal/approximateFinisherPeriodInSamples));
disp('BEST FREQUENCY: ');
frequency=1/approximateFinisherPeriodInSeconds;
disp(frequency);
disp('Power Spectrum Peak Frequency: ');
disp(tubDrivingFrequency);
end
end

if notLockedIntoRightFrequency==false

offsetInSamples = int32(offsetInSamples + (minSumSectionIndex-1)*sectionLengthInSamples);
sectionLengthInSamples=int32(round(sectionLengthInSamples/numberOfSections));
if (minSum<0)

refinementComplete=true;
end
for burstIndex=1:numberOfBursts
    burstStarts(burstIndex)=int32((burstIndex-1)*approximateFinisherPeriodInSamples - offsetInSamples);
end

%%% PLOT FORCE SIGNAL WITH BURST DIVISIONS - START %%%%%%%%%%%%%%%%%%%%%%
if drawForceSignalWithBurstDivisions==true
    clf reset;
    cla reset;
    tempPlot = newplot;
    axis tight;
    plot(normalForceData);
    set(findobj(gca,'Type','line','Color',[0 0 1]),...
        'Color','red',...
        'LineWidth',2);
    hold on;
    stemPlotBurstYAxis=zeros(numberOfBursts,1);
    for i=1:numberOfBursts
        stemPlotBurstYAxis(i)=30;
    end
    h=stem(burstStarts,stemPlotBurstYAxis );
    title('Bursts in Force Signal')
    xlabel('Time (us)');
    ylabel(['Force (N)']);
    tempPlot=gcf;
    saveas(tempPlot,[path,forceSignalFilename,'_Bursts_in_Force_Sig','.fig']);
    saveas(tempPlot,[path,forceSignalFilename,'_Bursts_in_Force_Sig','.png']);
end
%%% PLOT FORCE SIGNAL WITH BURST DIVISIONS - END %%%%%%%%%%%%%%%%%%%%%%

plotBothContactTypesInBursts=true;
shouldntHaveToUseThis=approximateFinisherPeriodInSamples+1; %% this stops a size mismatch from occuring. Don't know the root cause
burstsWithImpactsInThem=0;
burstsWithQuasiStaticsInThem=0;
burstNumber=0;
numberOfNoContactBursts=0;
numberOfBurstsWithLargeForceSignal=0;
minForForceConsideredToBeLargeInNewtons=2;
for j=2:numberOfBursts
    burstStart = int32(burstStarts(j-1));
    burstEnd = int32(burstStarts(j));
    foundImpactInBurst=false;
    foundQuasiStaticInBurst=false;
    velocityMaximumInBurst=0;
    largeForceSignalFound=false;
    for i=1:numberOfRealContacts
        contactStart = int32(realContactsData(i,iStart));
        contactEnd = int32(realContactsData(i,iEnd));
        if (contactStart<=burstStart)&&(contactEnd>=burstEnd) && (i,1) %contact in burst
            if realContactsData(i,iImpactVelocity) > velocityMaximumInBurst
                velocityMaximumInBurst = realContactsData(i,iImpactVelocity);
            end
            if realContactsData(i,iForceMaximum) >= minForForceConsideredToBeLargeInNewtons
                largeForceSignalFound=true;
            end
        end
        if iContactType==1 %impact
            if foundImpactInBurst==false
                burstsWithImpactsInThem=burstsWithImpactsInThem+1;
                burstSignalsImpact(:,burstsWithImpactsInThem)=zeros(shouldntHaveToUseThis,1);
            end
            foundImpactInBurst=true;
            for k=contactStart:contactEnd %write impact to burst Data
                burstPosition = k-burstStart+1;
                burstSignalsImpact(burstPosition, burstsWithImpactsInThem)=normalForceData(k);
            end
        end
        elseif (contactStart>burstStart) %% goto next burst
            end
        else
            if foundQuasiStaticInBurst==false
                burstsWithQuasiStaticsInThem=burstsWithQuasiStaticsInThem+1;
                burstSignalsQuasiStatic(:,burstsWithQuasiStaticsInThem)=zeros(shouldntHaveToUseThis,1);
            end
            foundQuasiStaticInBurst=true;
            for k=contactStart:contactEnd %write quasi-static contact to burst Data
                burstPosition = k-burstStart+1;
                burstSignalsQuasiStatic(burstPosition, burstsWithQuasiStaticsInThem)=normalForceData(k);
            end
        end
end
end
end

if (largeForceSignalFound==true) && (foundQuasiStaticInBurst==true)
    clf reset;
    cla reset;
    tempPlot = newplot;
end

%%% PLOT BURSTS WITH BOTH CONTACT TYPES HERE - START %%%%%%%%%%%%%%%%%%
axis tight;
xlim([0 approximateFinisherPeriodInSamples]);
hold on;
if foundImpactInBurst=true
    plot(burstSignalsImpact(:, burstsWithImpactsInThem), 'o', 'MarkerSize', 3);
    plot(burstSignalsImpact(:, burstsWithImpactsInThem), '-r');
end
if foundQuasiStaticInBurst=true
    if foundImpactInBurst=true
        hold on;
    end
    plot(burstSignalsQuasiStatic(:, burstsWithQuasiStaticsInThem), 'x', 'MarkerSize', 3);
    plot(burstSignalsQuasiStatic(:, burstsWithQuasiStaticsInThem));
end
end

%% PLOT BURSTS WITH BOTH CONTACT TYPES HERE - END %%%%%%%% %%%%% COLLECT BURST STATS - BEGIN %%%%
if ((foundImpactInBurst==true)||(foundQuasiStaticInBurst==true))
    velocityMaximumInBurstDistribution(j-1)=velocityMaximumInBurst;
    if largeForceSignalFound==true
        numberOfBurstsWithLargeForceSignal = numberOfBurstsWithLargeForceSignal+1;
    end
else
    % empty burst
    numberOfNoContactBursts = numberOfNoContactBursts+1;
end
%% COLLECT BURST STATS - END %%%%
break;
elseif ( ((contactStart>=burstStart)&&(contactStart<=burstEnd))&&(contactEnd>burstEnd))||...
    ( ((contactEnd>=burstStart)&&(contactEnd<=burstEnd))&&(contactStart<burstStart))
    disp('Error - Contact Cut by Burst');
    break;
end
end

%% OUTPUT BURST STATS TO ASCII FILE - BEGIN %%%
fileId=open([path,forceSignalFilename,'_Burst_Stats.txt'],'w');
fprintf(fileIdone,'%s

', [path,forceSignalFilename]);
fprintf(fileIdone,'%s%d
','Force Signal Duration (s): ', forceSignalDurationSeconds);
fprintf(fileIdone,'%s%d
','Burst Frequency (Hz): ', 1/approximateFinisherPeriodInSeconds);
fprintf(fileIdone,'%s%d
','Number of Bursts: ', numberOfBursts);
fprintf(fileIdone,'%s%d
','Percent of Bursts with No Contacts: ', 100*numberOfNoContactBursts/numberOfBursts);
fprintf(fileIdone,'%s%d
',
    ['Percent of Bursts with Contacts with a peak force greater than ',num2str(minForForceConsideredToBeLargeInNewtons), ' N: '
    ], 100*numberOfBurstsWithLargeForceSignal/numberOfBursts);
fclose(fileIdone);
%% OUTPUT BURST STATS TO ASCII FILE - END %%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DISTRIBUTION OF MAXIMUM VELOCITY IN EACH BURST - BEGIN %%%
[countsBurstMaxVelocity,binsBurstMaxVelocity]=hist(double(velocityMaximumInBurstDistribution),30);
for i=1:length(countsBurstMaxVelocity)
    if countsBurstMaxVelocity(i)>0
        countsBurstMaxVelocity(i)=countsBurstMaxVelocity(i)/forceSignalDurationSeconds;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
bar(binsBurstMaxVelocity,countsBurstMaxVelocity);
xlabel('Maximum Effective Impact Velocity In Bursts');
ylabel('Counts/Signal Duration');
hold off;
%title('Maximum Effective Impact Velocity In Bursts');
%saveas(tempPlot,[path,forceSignalFilename,'Max_Eff_Impact_Vel_In_Bursts_Hist','.fig'],'fig');
%saveas(tempPlot,[path,forceSignalFilename,'Max_Eff_Impact_Vel_In_Bursts_Hist','.png'],'png');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% DISTRUBTION OF MAXIMUM VELOCITY IN EACH BURST - END %%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PLOT OVERLAPPING CONTACTS - START %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
xlim([0 approximateFinisherPeriodInSamples]);
plot(burstSignalsQuasiStatic,'-x');
end

%% PLOT OVERLAPPING CONTACTS - END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ylabel('Force (N)');
tempPlot=gcf;
saveas(tempPlot,[path,forceSignalFilename,'_Overlapping_QuasiStatic_Bursts','.fig']','fig');
saveas(tempPlot,[path,forceSignalFilename,'_Overlapping_QuasiStatic_Bursts','.png']','png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
xlim([approximateFinisherPeriodInSamples]);
plot(burstSignalsImpact, 'o', 'MarkerSize', 3);
title('Overlapping Impact Bursts')
ylabel('Time (\mu s)');
tempPlot=gcf;
saveas(tempPlot,[path,forceSignalFilename,'_Overlapping_Impact_Bursts','.fig']','fig');
saveas(tempPlot,[path,forceSignalFilename,'_Overlapping_Impact_Bursts','.png']','png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
xlim([approximateFinisherPeriodInSamples]);
plot(burstSignalsQuasiStatic,'x', 'MarkerSize', 3);
hold on;
plot(burstSignalsImpact, 'o', 'MarkerSize', 3);
title('Overlapping Bursts - All Contacts')
xlabel('Time (\mu s)');
ylabel('Force (N)');
tempPlot=gcf;
saveas(tempPlot,[path,forceSignalFilename,'_Overlapping_Contacts_Bursts','.fig']','fig');
saveas(tempPlot,[path,forceSignalFilename,'_Overlapping_Contacts_Bursts','.png']','png');
hold off;

%%%%%% PLOT OVERLAPPING CONTACTS - END %%%%%%%%

%% quasi-static peak frequency.
forceRes=0.042058;
lengthOfSlopeToSkiDownInSamples=5; %4 ;yields good results  %10 is too big
firstPeakInBurstPosition=0;
timeSinceLastPeakIndex=0;
for i=1:burstsWithQuasiStaticInThem
    numberOfPeaksFound=0;
    lastSampleWasAPeak=false;
    for j=lengthOfSlopeToSkiDownInSamples:approximateFinisherPeriodInSamples-lengthOfSlopeToSkiDownInSamples
        peakFound=false;
        top=burstSignalsQuasiStatic(j,i);
        peakLocation=j;
        if ((top>forceRes)&&(lastSampleWasAPeak==false))  % Can't be a peak if it is zero
            peakFound=true;
            if (j==7962)&&(i==1)
                disp('bpoint'); %9
            end
            for s=1:lengthOfSlopeToSkiDownInSamples
                rSlope = burstSignalsQuasiStatic(j+s,i);
                lSlope = burstSignalsQuasiStatic(j-s,i);
                if (rSlope>top)
                    peakFound=false;
                    break;
                end
                priorRSlope=burstSignalsQuasiStatic(j+s-1,i); %priorRSlope=burstSignalsQuasiStatic(j-1,i);
                if ((rSlope>priorRSlope+forceRes)||(rSlope>top+forceRes))
                    peakFound=false;
                    break;
                end
                priorLSlope=burstSignalsQuasiStatic(j-s+1,i);
                if ((lSlope>priorLSlope+forceRes)||(lSlope>top+forceRes))
                    peakFound=false;
                    break;
                end
                if (lSlope<top)
                    peakFound=false;
                    break;
                end
                if (peakFound==false)
                    peakFound=true;
                    break;
                end
            end
            if (peakFound==true)
                numberOfPeaksFound=numberOfPeaksFound+1;
            end
            lastSampleWasAPeak=true;
        end
    end
end
if (numberOfPeaksFound==false)  % Can't be a peak if it is zero
    peakFound=true;

% disp('bpoint');
lastSampleWasAPeak=false;
end

%%% Remove consecutive peaks
realPeakCounter=0;
if numberOfPeaksFound >= 2
startFound=false;
for j=1:numberOfPeaksFound-1
endFound=false;
if (peakLocationsInBursts(j) == peakLocationsInBursts(j+1))$$$$
    disp('conditional breakpoint');
end
if ((peakLocationsInBursts(j+1)-peakLocationsInBursts(j)) == 1)$$$$
    startFound=true;
endFound=false;
    startLoc=peakLocationsInBursts(j);
end
if ((peakLocationsInBursts(j+1)-peakLocationsInBursts(j)) ~= 1)$$$$
    endFound=true;
    startFound=false;
    endLoc=peakLocationsInBursts(j);
if (endFound==false)&&(startFound==false) %not in flat peak. i.e. normal peak.
    realPeakCounter=realPeakCounter+1;
    realPeakLocationsInBursts(realPeakCounter,i) = peakLocationsInBursts(j);
else
    realPeakCounter=realPeakCounter+1;
    realPeakLocationsInBursts(realPeakCounter,i) = peakLocationsInBursts(numberOfPeaksFound);
end
clear peakLocationsInBursts;
if realPeakCounter>=2
for j=2:realPeakCounter
    timeSinceLastPeakIndex = timeSinceLastPeakIndex + 1;
    ssLP = realPeakLocationsInBursts(j,i) - realPeakLocationsInBursts(1,i);
    samplesSinceLastPeak(timeSinceLastPeakIndex)=ssLP;
    tslp=double(ssLP*samplePeriod*1000);
    timeSinceLastPeakMicroSeconds(timeSinceLastPeakIndex)=tslp;
end
%%% PLOT PEAKS - START %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (i<=50)
if realPeakCounter >0
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
xlim([0 approximateFinisherPeriodInSamples]);
plot(burstSignalsQuasiStatic(:,i), 'or');
hold on;
peaksYAxis=zeros(realPeakCounter,1);
for w=1:realPeakCounter
    peaksYAxis(w)=30;
end
for w=1:realPeakCounter
    tempPeakLocationForPlotting(w)=realPeakLocationsInBursts(w,i);
end
stem(tempPeakLocationForPlotting,peaksYAxis);
hold off;
clear tempPeakLocationForPlotting;
title(['Peaks - ',num2str(i)]);
xlabel(['Time (ms)']);
ylabel(['Force (N)']);
tempPlot=gcf;
saveas(tempPlot,[path,forceSignalFilename,'_Peaks_in_Static_Burst_',num2str(i),'.png']);
end
%%% PLOT PEAKS - END%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
% [countsTimeSinceLastPeak,binsTimeSinceLastPeak]=hist(timeSinceLastPeakMicroSeconds,1000);
% clf reset;
% cla reset;
% tempPlot = newplot;
% axis tight;
% bar(binsTimeSinceLastPeak,countsTimeSinceLastPeak);
% xlabel('Time Since Last Peak (ms)');
% ylabel('Frequency');
% tempPlot=gcf;
% saveas(tempPlot,[path,forceSignalFilename,'_Time_Between_Peaks','.fig'],fig');
% saveas(tempPlot,[path,forceSignalFilename,'_Time_Between_Peaks','.png'],png');

%%%%%%% PLOT QUASI-STATIC PEAK FREQUENCY HISTOGRAM - START %%%%%%%%
[countsSamplesSinceLastPeak,binsSamplesSinceLastPeak]=hist(double(samplesSinceLastPeak),100);
countsSamplesSinceLastPeak=countsSamplesSinceLastPeak.*(1/forceSignalDurationSeconds);
binsSamplesSinceLastPeak=binsSamplesSinceLastPeak.*(1/1000); %convert to ms
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
bar(binsSamplesSinceLastPeak,countsSamplesSinceLastPeak,1);
xlabel('Peak Interval (ms)');
ylabel('Peaks Per Second');
tempPlot=gcf;
saveas(tempPlot,[path,forceSignalFilename,'_Peak_Interval_Dist','.fig'],fig');
saveas(tempPlot,[path,forceSignalFilename,'_Peak_Interval_Dist','.png'],png');

%histogramDataForExport(1,:)=binsSamplesSinceLastPeak;
%histogramDataForExport(2,:)=countsSamplesSinceLastPeak.;
%dlmwrite([path,filename,'_Peak_Interval_Dist'],histogramDataForExport,','); clear histogramDataForExport;
BEGIN EXPORT TO EXCELL
fileId=fopen([path,forceSignalFilename,'_Peak_Interval_Dist','.csv'],'w');
fprintf(fileId,'%s,','Peak Interval (ms)');
fprintf(fileId,'%s','Peaks Per Second');
fprintf(fileId,'
');
for i=1:length(binsSamplesSinceLastPeak) %loop over rows
fprintf(fileId,'%d,',binsSamplesSinceLastPeak(i));
fprintf(fileId,'%d',countsSamplesSinceLastPeak(i));
fprintf(fileId, '
');
end
fclose(fileId);

%%%%%%% PLOT QUASI-STATIC PEAK FREQUENCY HISTOGRAM - END %%%%%%%%

% plot(burstSignalsImpact);
disp('FINISHED');
for j=1:5
beep;
pause(0.1);
beep;
pause(0.3);
end
clear all;
find peaks in quasi-static bursts.
Consolidate force signals
clc;

% INPUT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
path='E:\DOCS\PhD\THESIS\SensorDat\June 13 2005 results\200 lbs Steel Media, 1.75 from HW, Facing HW, 5 min\';
path='E:\DOCS\PhD\THESIS\SensorDat\June 13 2005 results\200 lbs Steel Media, 3.75 from HW, Facing HW, 5 min\';
path='E:\DOCS\PhD\THESIS\SensorDat\June 13 2005 results\250 lbs Steel Media, 1.75 from HW, Facing HW, 5 min\';
path='E:\DOCS\PhD\THESIS\SensorDat\June 13 2005 results\250 lbs Steel Media, 3.75 from HW, Facing HW, 5 min\';
path='E:\DOCS\PhD\THESIS\SensorDat\June 13 2005 results\90 lbs Porcelain Media, 1.75 from HW, Facing HW, 5 min\';
path='E:\DOCS\PhD\THESIS\SensorDat\June 13 2005 results\90 lbs Porcelain Media, 3.75 from HW, Facing HW, 5 min\';
standardInputFiles=true; % 15s at 1e-6 sample period.
signalPeriodStudyOnHistVariability=false;
limitConsolidationToThisNumberOfFiles=0;  %here 0 means no limit
% CODE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
contactDataFileSignature='*_Contact_Data.csv';
files = dir([path,contactDataFileSignature]);
numberOfFilesInBatchQueue=length(files);
if numberOfFilesInBatchQueue==0
disp(['No files in *', path]);
return;
end
if signalPeriodStudyOnHistVariability==true
rep=numberOfFilesInBatchQueue;

148
else
reps=1;
end
for counter=1:reps
if signalPeriodStudyOnHistVariability==true
consolidatedFileId = fopen(fullfile(path,files(1).name,'_','_num2str(counter),'_','_Consolidated.csv'),'w');
else
consolidatedFileId = fopen(fullfile(path,files(1).name,'_Consolidated.csv'),'w');
end
fprintf(consolidatedFileId,'%s,%s,%s,%s,%s,%s,%s,%s
','Start index','End index','Contact time s','Impulse Ns','Force max N','Contact type 0=NotClassified 1=Impact 2=Quasi-static','Impact Velocity m/s','Time Between Contacts','Time Between Contacts of Type');
indexOffset=0;
if signalPeriodStudyOnHistVariability==true
numberOfFilesToProcess=counter;
else
numberOfFilesToProcess=numberOfFilesInBatchQueue;
end
if (signalPeriodStudyOnHistVariability==false)&&(limitConsolidationToThisNumberOfFiles>0)
if numberOfFilesInBatchQueue > limitConsolidationToThisNumberOfFiles
numberOfFilesToProcess = limitConsolidationToThisNumberOfFiles;
end
end
for fileNum=1:numberOfFilesToProcess
filename = files(fileNum).name;
contactDataFileDump = csvread(fullfile(path,filename), 1, 0);
rowsOfDataInContactDataFile = length(contactDataFileDump);
if fileNum > 1
if standardInputFiles
indexOffset = (fileNum-1)*15000000;
else
indexOffset = contactDataFileDump(rowsOfDataInContactDataFile,2)+1;
end
for i=1:rowsOfDataInContactDataFile
contactDataFileDump(i,1)=contactDataFileDump(i,1)+indexOffset;
contactDataFileDump(i,2)=contactDataFileDump(i,2)+indexOffset;
end
end
for i=1:rowsOfDataInContactDataFile
fprintf(consolidatedFileId,'%d,%d,%E,%E,%E,%d,%E,%E,%E
', contactDataFileDump(i,:));
end
clear contactDataFileDump;
end
fclose(consolidatedFileId);
disp('Done');
end

7.2.5. Perform Statistics on Consolidated Data

Mathworks Inc. Matlab V7 Code

clc;
clear;
path='E:\DOCS\PhD\THEESIS\SensorDat\June 13 2005 results\90 lbs Porcelain Media, 3.75 from SW, Facing SW, 2.5 min\';
filename='90 lbs Porcelain Media, 3.75 from SW, Facing SW, 2.5 min.csv';
numOfBinsInHistograms=50;
standardSamplePeriodOfMicroSecond=true;
realContactsData = csvread(fullfile(path,filename), 1, 0);
putFileNameInTitle=false;

%enumerations start
iStart=1;
iEnd=2;
iContactTime=3;
iImpulse=4;
iForceMaximum=5;
iContactType=6; % 0=Not Classified, 1=Impact, 2=quasi-static
iImpactVelocity=7;
iTimeNextContact=8;
iTimeNextSameTypeContact=9;
cTypeNoClassification=0;

149
cTypeImpact = 1;
cTypeQuasiStatic = 2;

numberOfColsInRealContactData = length(realContactsData(:,1));
numberOfQuasiStaticContacts = 0;
numberOfImpactContacts = 0;
numberOfUnclassifiedContacts = 0;

numberOfRealContacts = length(realContactsData(:,1));

for i = 1:numberOfRealContacts
    if realContactsData(i,ContactType) == cTypeQuasiStatic
        numberOfQuasiStaticContacts = numberOfQuasiStaticContacts + 1;
    end
    if realContactsData(i,ContactType) == cTypeImpact
        numberOfImpactContacts = numberOfImpactContacts + 1;
    end
    if realContactsData(i,ContactType) == cTypeNoClassification
        numberOfUnclassifiedContacts = numberOfUnclassifiedContacts + 1;
    end
end

quasiStaticContactData = zeros(numberOfQuasiStaticContacts, numberOfColsInRealContactData);
impactContactData = zeros(numberOfImpactContacts, numberOfColsInRealContactData);
unclassifiedContactData = zeros(numberOfUnclassifiedContacts, numberOfColsInRealContactData);

m = 0;
n = 0;
p = 0;

for i = 1:numberOfRealContacts
    if realContactsData(i,ContactType) == cTypeQuasiStatic
        m = m + 1;
        quasiStaticContactData(m,:) = realContactsData(i,:);
    end
    if realContactsData(i,ContactType) == cTypeImpact
        n = n + 1;
        impactContactData(n,:) = realContactsData(i,:);
    end
    if realContactsData(i,ContactType) == cTypeNoClassification
        p = p + 1;
        unclassifiedContactData(p,:) = realContactsData(i,:);
    end
end

filenameTitle = strrep(filename,'_',' ');
set(gcf,'Renderer','OpenGL');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(realContactsData(:,iForceMaximum),numOfBinsInHistograms);
if putFileNameInTitle==true
    title(filenameTitle);
end
xlabel('Bin: Force Maximums (N)');
ylabel(['Frequency']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Force_Max','.fig'],[path,filename,'_Hist_Force_Max','.png']);
saveas(tempPlot,[path,filename,'_Hist_Stat_Force_Max','.fig'],[path,filename,'_Hist_Stat_Force_Max','.png']);
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(impactContactData(:,iForceMaximum),numOfBinsInHistograms);
if putFileNameInTitle==true
    title(filenameTitle);
end
xlabel('Bin: Impact Force Maximums (N)');
ylabel(['Frequency']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Impact_Force_Max','.fig'],[path,filename,'_Hist_Impact_Force_Max','.png']);
saveas(tempPlot,[path,filename,'_Hist_Impact_Force_Max','.fig'],[path,filename,'_Hist_Impact_Force_Max','.png']);
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(quasiStaticContactData(:,iForceMaximum),numOfBinsInHistograms);
if putFileNameInTitle==true
    title(filenameTitle);
end
xlabel('Quasi-Static Force Maximums (N)');
ylabel(['Frequency']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Stat_Force_Max','.fig'],[path,filename,'_Hist_Stat_Force_Max','.png']);
saveas(tempPlot,[path,filename,'_Hist_Stat_Force_Max','.fig'],[path,filename,'_Hist_Stat_Force_Max','.png']);
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(impactContactData(:,iImpactVelocity),numOfBinsInHistograms);
if putFileNameInTitle==true
title(filenameTitle);
end
xlabel('Impact Velocity (of impact type) (m/s)');
ylabel(['Frequency']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Impact_Vel','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Impact_Vel','.png'],'png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(quasiStaticContactData(:,iImpactVelocity),numOfBinsInHistograms);
if putFileNameInTitle==true
title(filenameTitle);
end
xlabel('Quasi-Static Effective Velocity (m/s)');
ylabel(['Frequency']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Static_Vel','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Static_Vel','.png'],'png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(realContactsData(:,iImpactVelocity),numOfBinsInHistograms);
if putFileNameInTitle==true
title(filenameTitle);
end
xlabel('Effective Impact Velocity (all types) (m/s)');
ylabel(['Frequency']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Eff_Impact_Vel','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Eff_Impact_Vel','.png'],'png');

binEnd = max( realContactsData(:,iImpactVelocity) );
binStart = min( realContactsData(:,iImpactVelocity) );
binInc = (binEnd - binStart)/numOfBinsInHistograms;
binVector = binStart:binInc:binEnd;
histogramCount = histc(realContactsData(:,iImpactVelocity),binVector);
histoGramDataForExport(1,:)=binVector;
histoGramDataForExport(2,:)=histogramCount.';
dlmwrite([path,filename,'_Vel_Hist_Dat.csv'],histogramDataForExport,',');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fileId=fopen([path,filename,'_Velocity_Histogram_Data.csv'],'w');
% fclose(fileId);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(impactContactData(:,iContactTime),numOfBinsInHistograms);
if putFileNameInTitle==true
title(filenameTitle);
end
xlabel('Impact Contact Time (sec)');
ylabel(['Frequency']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Impact_Contact_Period','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Impact_Contact_Period','.png'],'png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(quasiStaticContactData(:,iContactTime),numOfBinsInHistograms);
if putFileNameInTitle==true
title(filenameTitle);
end
xlabel('Quasi-Static Contact Time (sec)');
ylabel(['Frequency']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Static_Contact_Period','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Static_Contact_Period','.png'],'png');
clf reset;
cla reset;
tempPlot = newplot;
axis tight;
hist(realContactsData(:,iContactTime),numOfBinsInHistograms);
if putFileNameInTitle==true
title(filenameTitle);
end
xlabel('Effective Contact Time (all types) (sec)');
ylabel(['Frequency']);
tempPlot=gcf;
saveas(tempPlot,[path,filename,'_Hist_Eff_Impact_Contact_Period','.fig'],'fig');
saveas(tempPlot,[path,filename,'_Hist_Eff_Impact_Contact_Period','.png'],'png');
title(filenameTitle);
end
xlabel('Contact Time (sec)');
ylabel('Frequency');
tempPlot=gcf;
saveas(tempPlot, [path, filename, '_Hist_Cont_Period', '.fig'], 'fig');
saveas(tempPlot, [path, filename, '_Hist_Cont_Period', '.png'], 'png');
clf reset;
cia reset;
tempPlot = newplot;
axis tight;
hist(realContactsData(:, iTimeToNextContact), numOfBinsInHistograms*5);
if putFileNameInTitle==true
    title(filenameTitle);
end
xlabel('Time to Next Contact (all types) (s)');
ylabel('Frequency');
tempPlot=gcf;
saveas(tempPlot, [path, filename, '_Hist_Cont_Period', '.fig'], 'fig');
saveas(tempPlot, [path, filename, '_Hist_Cont_Period', '.png'], 'png');
clf reset;
cia reset;
tempPlot = newplot;
axis tight;
piePlotVect(1) = typeNoClassificationCount;
piePlotVect(2) = typeImpactCount;
piePlotVect(3) = typeQuasiStaticCount;
pie(piePlotVect);
title(filenameTitle);
ylabel('Number OF Contacts by Type');
tempPlot=gcf;
saveas(tempPlot, [path, filename, '_PIE_NumOfCon', '.fig'], 'fig');
saveas(tempPlot, [path, filename, '_PIE_NumOfCon', '.png'], 'png');
clf reset;
cia reset;
tempPlot = newplot;
axis tight;
piePlotVect(1) = typeNoClassificationTimeCount;
piePlotVect(2) = typeImpactTimeCount;
piePlotVect(3) = typeQuasiStaticTimeCount;
piePlotVect(4) = numberOfSamples - (typeNoClassificationTimeCount + typeImpactTimeCount + typeQuasiStaticTimeCount);
pie(piePlotVect);
title(filenameTitle);
ylabel('Portion of signal of Specific Type');
tempPlot=gcf;
saveas(tempPlot, [path, filename, '_PIE_Time', '.fig'], 'fig');
saveas(tempPlot, [path, filename, '_PIE_Time', '.png'], 'png');
clf reset;
cia reset;
tempPlot = newplot;
axis tight;
piePlotVect(1) = avgContactFrequencyByTimeToNextContact/\( \frac{1}{\text{sum(realContactsData(:,iTimeToNextContact))}/\text{length(realContactsData(:,iTimeToNextContact))}} \);
piePlotVect(2) = avgContactFrequencyByTotalTimeAndNumOfContacts = \( \frac{\text{sum(realContactsData(:,iEnd)*1e-6)} \times \text{numberOfRealContacts}}{\text{numberOfRealContacts}} \);

averageImpactVelocity = \( \frac{\text{avgContactFrequencyByTimeToNextContact}}{\text{sum(realContactsData(:,iTimeToNextContact))}/\text{length(realContactsData(:,iTimeToNextContact))}} \);
maxImpactVelocity = max(realContactsData(:,1,1,1));
mediaMassSteel = 1.031/1000;
mediaMassPorcelain = 0.286/1000;
220mm radius sensor
averageEnergyFluxSteel = \( \frac{2\times mediaMassSteel \times \text{averageImpactVelocity}^2 \times \text{avgContactFrequencyByTotalTimeAndNumOfContacts}}{(3.14159 \times (2/1000)^2)} \);
averageEnergyFluxPorcelain = \( \frac{2\times mediaMassPorcelain \times \text{averageImpactVelocity}^2 \times \text{avgContactFrequencyByTotalTimeAndNumOfContacts}}{(3.14159 \times (2/1000)^2)} \);

fprintf(fileId, '%s
', filename);
fprintf(fileId, '%s

', date);
fprintf(fileId, '%s%s
', 'Average Contact Frequency (by time to next contact) (Hz): ', num2str(avgContactFrequencyByTimeToNextContact));
fprintf(fileId, '%s
', 'Average Contact Frequency (this one is used) (Hz): ', num2str(avgContactFrequencyByTotalTimeAndNumOfContacts));
fprintf(fileId, '%s
', 'Average Energy Flux (Steel) [J/s/m^2]: ', num2str(averageEnergyFluxSteel));
fprintf(fileId, '%s
', 'Average Energy Flux (Porcelain) [J/s/m^2]: ', num2str(averageEnergyFluxPorcelain));
fprintf(fileId, '%s
', 'Maximum (Impact and Effective) Velocity (m/s): ', num2str(maxImpactVelocity));

fclose(fileId);
7.2.6. 1D Process Model Code

function hybdOneD(minutesOfFinishing, condition91Sa, Strip617umSwitch, bilinearMaterialModelSwitch, elecMagImpactVelocity)

% profile on;
overRideArguments=false;
if overRideArguments==true
minutesOfFinishing=120;
condition91Sa=false;
Strip617umSwitch=true;
bilinearMaterialModelSwitch=false;
end
useAbsNrgPerContactZoneArea=true;
simulateTransverseImpactLocationDistribution=false;
forceSequentialPickingOfEachVelocity=true;
transverseDirectionWidthEqualToContactZoneRadius=false;
fixedContactZoneRadiusSetting=false;
useLookUpTablesAndInterpolationForMatModels=true;
useLinearDecreaseInAbsorbedEnergyAsDistanceFromCenterLine=false;

debugMode=false;

%%%%%%%% Electromagnet Simulation Controls - START %%%%%%%%%
if overRideArguments==true
elecMagImpactVelocity=0;
end
elecMagConsiderBounces=false;
elecMagMaximumNumberOfBounces=0;

%%%%%%%% Electromagnet Simulation Controls - END %%%%%%%%%

versionNumber='3.24.001'; %do not put the letter v in front.
munlock('arcHeightFromAbseLKTB08');
clear arcHeightFromAbseLKTB08;
munlock('coefficientOfResGivenAbseAndVLKTB07');
clear coefficientOfResGivenAbseAndVLKTB07;
if debugMode==true
disp('WARNING! DEBUG MODE');
end

condition112Sb=~condition91Sa;
if condition91Sa==true
averageSensorCollisionFrequencyHertz=462; % nfdatD 200 lb Steel 1.75 frm HW Facing HW Flush button - 5
min.csv_Vel_Hist_Dat.csv
saveFilePath='C:\_DOCS\PhD\THESIS\_Model\_Code\1D Time Varying Curvature - Cumm abs energy, e, and arc height\Output\';
contactDataFilename = 'nfdatD 200 lb Steel 1.75 frm HW Facing HW Flush button - 5 min.csv';
contactDataPath = 'C:\_DOCS\PhD\Thesis\SensorData\June 13 2005 results\200 lbs Steel Media, 1.75 from HW, Facing HW, 5
min\';
elseif condition112Sb==true
averageSensorCollisionFrequencyHertz=224; % nfdatD Steel 3.75 frm HW Facing HW Flush button - 05.csv_Vel_Hist_Dat.csv
saveFilePath='C:\_DOCS\PhD\Thesis\SensorData\June 13 2005 results\250 lbs Steel Media, 3.75 from HW, Facing HW, 5
min\'
else
disp('ERROR: No conditions were chosen');
exit(0);
end

numberOfGridElements=round(1000000*(2.5/3));
numberOfGridElements=100000;
if debugMode==true
numberOfGridElements=round(10000*(2.5/3));
end
yieldZoneFudgeFactor=1;
PlotDeformedShape=false;
PlotCoverage=true; %setting to false causes crash - fix this
graphExperimentalDataForComparision=true;

%%%%%%%% Electromagnet Simulation - START %%%%%%%%%%%%%%%%%%%%%
elecMagFreq = 2;
elecMagNumOfBalls = 33;
elecMagPeriod = 1/elecMagFreq;
coefficientOfRestitution=0.5;
accelerationDueToGravity=9.81;
elecMagprimaryBounce=true;
forceSequentialPickingOfEachVelocity=true;

153
simulateTransverseImpactLocationDistribution=true;
transverseDirectionWidthEqualToContactZoneRadius=false;
fixedContactZoneRadiusSetting=false;

%%% Electromagnet Simulation - END %%%%
lastProgressUpdateTime = clock();
insufficientResolutionWarningCount=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Force Signal velocity distribution Data Retrieval
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if elecMagImpactVelocity==0
    realContactsData = csvread([contactDataPath,contactDataFilename], 1, 0);
%enumerations start
iStart=1;
iEnd=2;
iContactTime=3;
iImpulse=4;
iForceMaximum=5;
iContactType=6; % 0=Not Classified, 1=impact, 2=quasi-static
iImpactVelocity=7;

samplePeriod=1/1000000;
impactVelocityHistory=realContactsData(:,iImpactVelocity);
actualTimeData=realContactsData(:,iStart).*samplePeriod;
numberOfImpactVelocitiesInContactsData = length(impactVelocityHistory)-1;
impactVelocityHistoryCounter=1;

numberOfCounterResets=1;
clear realContactsData;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Force Signal velocity distribution Data Retrieval
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
numberOfNodes=numberOfGridElements+1;
coverageDescreteDomain = zeros(numberOfPointsOnCoverageAndArcheightGraphs+1,1);
arcHeightHistory = zeros(numberOfPointsOnCoverageAndArcheightGraphs+1,1);
timeAxis = zeros(numberOfPointsOnCoverageAndArcheightGraphs+1,1);
if elecMagImpactVelocity==0
    actualStripTimeAxis = zeros(numberOfPointsOnCoverageAndArcheightGraphs+1,1);
    actualSensorTimeAxis = zeros(numberOfPointsOnCoverageAndArcheightGraphs+1,1);
    impactNumberAxis = zeros(numberOfPointsOnCoverageAndArcheightGraphs+1,1);
end
nodes= zeros(numberOfGridElements+1,2); %this contains (x,y) coordinates of the strip.
arcHeightGrid= zeros(numberOfGridElements,1);
gridAbsorbedEnergy= zeros(numberOfGridElements,1);
psi= zeros(numberOfGridElements,1);
calculationCounter=0;
numberOfTimesGraphed = 1;
legend_string_matrix_index=0;
progressCounter=0;
actStripTime=0;
actSensorTime=0;
impactVelocity=0;
impactCounter=0;
outputCounter=0;
impactVelocityHistoryCounter=1;

while impactCounter<=N
    outputCounter=outputCounter+1;
    impactCounter=impactCounter+1;
    if elecMagImpactVelocity==0
        if impactCounter > numberOfImpactVelocitiesInContactsData*(numberOfCounterResets+1);
            impactVelocityHistoryCounter=1;
            numberOfCounterResets=numberOfCounterResets+1;
        end

        impactVelocityHistory=impactVelocityHistory(1:impactVelocityHistoryCounter);
        sensorPeriod = (actualTimeData(impactVelocityHistoryCounter+1)-actualTimeData(impactVelocityHistoryCounter));
        almenStripPeriod = sensorPeriod*(sensorArea/almenStripTargetArea);
        actStripTime = almenStripPeriod + actStripTime;
        actSensorTime = sensorPeriod + actSensorTime;
        impactVelocityHistoryCounter=impactVelocityHistoryCounter+1;
    else
        elecmag section
        impactVelocity=elecMagImpactVelocity;
        end

    end

    a=0;
    if fixedContactZoneRadiusSetting==true
        a=fixedContactZoneRadius;
    else
        a=fixedContactZoneRadius(impactVelocity);
        a=a*1.0030617umVsAlloysModelContactRadius(impactVelocity);
    end

    a=yieldZoneFudgeFactor*a;
    if debugMode==true
        a=a*0.1;
    end

    if debugMode==true
        if ((a/lengthOfGridElement) < 2)
            insufficientResolutionWarningCount=insufficientResolutionWarningCount+1;
            disp('Warning! Insufficient Resolution');
        end
    end

155
if transverseDirectionWidthEqualToContactZoneRadius==true
  width=0;
end

if simulateTransverseImpactLocationDistribution==true
  randomTransverseImpactLocation=rand()*(-1*(width/2+a)-(width/2+a))+(width/2+a);
  if forceSequentialPickingOfEachVelocity==true
    contactWithCenterLine=false;
    while contactWithCenterLine==false
      % the transverse direction is measured from centre of the strip.
      if abs(randomTransverseImpactLocation)>=a
        randomTransverseImpactLocation=rand()*(-1*(width/2+a)-(width/2+a))+(width/2+a);
        impactCounter=impactCounter+1;
      end
      contactWithCenterLine=true;
  end
  lengthOfContactCircleAndGridIntersection=2*realsqrt(randomTransverseImpactLocation^2+a^2);
else
  if abs(randomTransverseImpactLocation)==a
    lengthOfContactCircleAndGridIntersection=0;
  else
    lengthOfContactCircleAndGridIntersection=2*realsqrt(randomTransverseImpactLocation^2+a^2);
  end
end

%%%%%%% Determine size and location of contact zone in terms of
%%%%%%% elements.
if lengthOfContactCircleAndGridIntersection==0
  randomLongImpactLocation=rand()*((span/2+a)-(-span/2+a))-((span/2+a)); %using coordinate system at center.
  upLimCoord = randomLongImpactLocation + a;
  lwLimCoord = randomLongImpactLocation - a;
  ub=round((upLimCoord/lengthOfGridElement) + numberOfGridElements/2);
  lb=round((lwLimCoord/lengthOfGridElement) + numberOfGridElements/2);
  if lb<=0
    lb=1;
  end
  if ub<=0
    ub=1;
  end
  if ub>=numberOfGridElements
    ub=numberOfGridElements;
  end
  if lb>=numberOfGridElements
    lb=numberOfGridElements;
  end
  if lb>ub
    lb=ub;
  end
  contactArea=pi*a^2;
  if contactArea<0
    disp('contactArea BUG');
  end
  % lengthOfContactZoneInElements=abs(ub-lb);
  numberOfElementsInContactZone=ub-lb+1;
  newCumulativeEnergy=0;
  sumOfGridEnergyInContactZone=0;
  for j=lb:ub
    if useAbsNrgPerContactZoneArea==true
      gridAbsorbedEnergy(j)=gridAbsorbedEnergy(j)+absorbedEnergy/contactArea;
    end
    sumOfGridEnergyInContactZone = sumOfGridEnergyInContactZone+gridAbsorbedEnergy(j);
  end
  if useAbsNrgPerContactZoneArea==true
    averageGridEnergyInContactZone = sumOfGridEnergyInContactZone/numberOfElementsInContactZone;
  else
    averageGridEnergyInContactZone = sumOfGridEnergyInContactZone;
  end
  if useLookUpTablesAndInterpolationForMatModels==true
    [newCumulativeEnergy, coefficientOfRestitution]  = coefficientOfResGivenAbsAndVLKTB07(Strip617umSwitch,
    bilinearMaterialModelSwitch,useAbsNrgPerContactZoneArea, averageGridEnergyInContactZone, impactVelocity);
  else
    disp('Need to finish changing this code');
    return;
  end
  vRebound=coefficientOfRestitution*impactVelocity;
  absorbedEnergy=0.5*massOfSingleMedia*(impactVelocity^2-vRebound^2);
  if absorbedEnergy<0
    disp('absorbedEnergy BUG');
  end
  for j=lb:ub
    if useAbsNrgPerContactZoneArea==true
      gridAbsorbedEnergy(j)=gridAbsorbedEnergy(j)+absorbedEnergy/contactArea;
else
    gridAbsorbedEnergy(j)=gridAbsorbedEnergy(j)+absorbedEnergy;
end

[archeightFEAMicron]=archHeightFromAbsEnergy(Strip617umSwitch, bilinearMaterialModelSwitch, useAbsNrgPerContactZoneArea, gridAbsorbedEnergy(j));
archHeightGrid(j)=archeightFEAMicron*1e-6;
end

if elecMagConsiderBounces==true
    disp('Need to finish changing this code');
    return;
end
end %if lengthOfContactCircleAndGridIntersection==0

if ((impactCounter>=calculationCounter*impactsBetweenCalculationOfCentreSpanArcHeight)

X=1; % enumeration for nodes index
Y=2; % enumeration for nodes index
beta=0;
for e=0:numberOfGridElements-1
    if e>0
        APrev=archHeightGrid(e);
    else
        APrev=0;
    end
    A=archHeightGrid(e+1);
    if (APrev==0)
        gammaPrev=0;
    else
        if APrev > 0
            gammaSignPrev=-1;
        else
            gammaSignPrev=1;
        end
        APrev=abs(APrev);
        RPrev=((span^2/4)+APrev^2)/(2*APrev);
        if abs(lengthOfGridElement/(2*RPrev))>1
            disp('Error: Out of Range for arcsin');
        end
        gammaPrev=gammaSignPrev*asin(lengthOfGridElement/(2*RPrev));
    end
    if (A==0)
        gamma=0;
    else
        if A > 0
            gammaSign=-1;
        else
            gammaSign=1;
        end
        A=abs(A);
        R=((span^2/4)+A^2)/(2*A);
        if abs(lengthOfGridElement/(2*R))>1
            disp('Error: Out of Range for arcsin');
        end
        gamma=gammaSign*asin(lengthOfGridElement/(2*R));
    end
    beta=gammaPrev+gamma;
    if e==0
        psi(e+1)=beta;
    else
        psi(e+1)=beta+psi(e);
    end
end
for e=1:numberOfGridElements
    nodes(e+1,X)= lengthOfGridElement*cos(psi(e)) + nodes(e,X);
    nodes(e+1,Y)= lengthOfGridElement*sin(psi(e)) + nodes(e,Y);
end
theta = atan(-nodes(numberOfNodes,Y)/nodes(numberOfNodes,X));
archeightOfCentreNode=nodes(round(numberOfNodes/2),X)*sin(theta)+nodes(round(numberOfNodes/2),Y)*cos(theta);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END%
% i.e. find parameters needed to calculate the height at center of strip.
arcHeightHistory(calculationCounter+1)=archeightOfCentreNode;
timeAxis(calculationCounter+1)=(impactCounter/almenStripCollisionFrequency)/60; %changed line
timeAxis(calculationCounter+1)=actStripTime/60;
actualSensorTimeAxis(calculationCounter+1)=actSensorTime/60;
impactNumberAxis(calculationCounter+1)=impactCounter;

elementsUncovered=0;
for subfI=1:numberOfGridElements
    if arcHeightGrid(subfI)==0.0
        elementsUncovered=elementsUncovered+1;
    end
end
coverageDescreteDomain(calculationCounter+1) = (numberOfGridElements-elementsUncovered)/numberOfGridElements;
calculationCounter = calculationCounter+1;

end %end of graph and record data routine

% Progress and ETA Code - START %%%%%%%%%%%%%
currentTime = clock();
if (etime(currentTime, lastProgressUpdateTime)>2)
    lastProgressUpdateTime = clock();
    percentComplete=(100*impactCounter/N);
    elapsedTime = etime(currentTime, programStartTime);
    eta = (100-percentComplete)*elapsedTime/percentComplete;
    disp([num2str(impactCounter),',','Impact Count',num2str(eta),',','Completed',num2str(impactCounter),',','Percent Complete',
    num2str((100-impactCounter/N)),',','Total Run Time in hours',num2str(elapsedTime),',','Run Time elapsed in hours',num2str(eta),',','Estimated time of arrival',
        num2str((elapsedTime/(60*60))+eta),',','Total Run Time in hours' ];)
end
% Progress and ETA Code - END %%%%%%%%%%%%%

arcHeightHistory=arcHeightHistory.*(1000*1000); %convert to micron

%BEGIN EXPORT TO EXCELL
% fileId=fopen([saveFilePath,num2str(randomInteger),'_data','.csv'],'w');
fileId=fopen([saveFilePath,fileNamePrefix,'_data','.csv'],'w');
fprintf(fileId,'%s,','Impact Number');
fprintf(fileId,'%s,','Time Derived from contact frequency');
if elecMagImpactVelocity==0
    fprintf(fileId,'%s,','Actual Sensor Time');
    fprintf(fileId,'%s,','Actual Strip Time');
end
fprintf(fileId,'%s,','Arc Height (Microns)');
fprintf(fileId,'%s','Coverage');
fprintf(fileId,'
');
lastTimeSaved=0;
for i=1:length(timeAxis) %loop over rows
    if (timeAxis(i)<30)||((timeAxis(i)-lastTimeSaved)>=2)
        lastTimeSaved=timeAxis(i);
        fprintf(fileId,'%d,',impactNumberAxis(i));
        fprintf(fileId,'%d,',timeAxis(i));
        if elecMagImpactVelocity==0
            fprintf(fileId,'%d,',actualSensorTimeAxis(i));
            fprintf(fileId,'%d,',actualStripTimeAxis(i));
        end
        fprintf(fileId,'%d,',arcHeightHistory(i));
        fprintf(fileId,'%d',coverageDescreteDomain(i));
        fprintf(fileId, '
');
    end
end
fclose(fileId);

%END EXPORT TO EXCELL

% fileId=fopen([saveFilePath,num2str(randomInteger),'_RESULTS_SETTINGS.txt'],'w');
fileId=fopen([saveFilePath,fileNamePrefix,'_SET.txt'],'w');
fprintf(fileId,'%s %s
','Random File Number: ',num2str(randomInteger));
fprintf(fileId,'%s %s
','Code Version ', versionNumber);
fprintf(fileId,'%s %s
','Completion Date & Time: ', datestr(now));
fprintf(fileId,'%s %s
','Elapsed Run-Time (Hours): ',num2str(elapsedTime/(60*60)));
fprintf(fileId,'%s %s
','Run Efficiency (Impacts per seconds of Run-Time): ',num2str(N/elapsedTime));
if elecMagImpactVelocity==0
    fprintf(fileId,'%s
','Contact Data Filename');
end
fprintf(fileId,'%s
','***MODEL*INPUTS********************************************');
fprintf(fileId,'%s %s
','Media Type:', mediaType);
fprintf(fileId,'%s %s
','Almen Strip Type:', almenStripType);
if Strip617umSwitch==true
    fprintf(fileId,'%s
','617um Thickness');
else
    fprintf(fileId,'%s %s
','617um thickness');
end
fprintf(fileId,'%s %s
','779um Thickness');
fprintf(fileId,'%s
','779um Thickness');
if bilinearMaterialModelSwitch==true
    fprintf(fid,'%s
','Bilinear Isotropic Perfectly Elastic Perfectly Plastic Material Model');
else
    fprintf(fid,'%s
','Power Law Isotropic Material Model');
end
if useAbsNrgPerContactZoneArea==true
    fprintf(fid,'%s
','Tracked the energy absorbed by the target per unit area of the contact zone.');
else
    fprintf(fid,'%s
','Tracked the energy lost by the ball.');
end
if elecMagImpactVelocity==0
    if condition91Sa==true
        fprintf(fid,'%s
','91Sa');
    else
        fprintf(fid,'%s
','112Sb');
    end
    fprintf(fid,'%s %s
','Contact Radius Scaling factor: ',num2str(yieldZoneFudgeFactor));
    fprintf(fid,'%s %s
','Minutes of Finishing: ',num2str(minutesOfFinishing));
    fprintf(fid,'%s %s
','Number of Elements: ',num2str(numberOfGridElements));
    if simulateTransverseImpactLocationDistribution==true
        fprintf(fid,'%s
','Generate Random Transverse Impact Location: ON');
    else
        fprintf(fid,'%s
','Generate Random Transverse Impact Location: OFF');
    end
end
if transverseDirectionWidthEqualToContactZoneRadius==true
    fprintf(fid,'%s
','Set Transverse Direction Width=0: ON');
else
    fprintf(fid,'%s
','Set Transverse Direction Width=0: OFF');
end
if (fixedContactZoneRadiusSetting==true)
    fprintf(fid,'%s %s
','Forced Contact Radius: ',num2str(fixedContactZoneRadius));
end
if useLinearDecreaseInAbsorbedEnergyAsDistanceFromCenterLine==true
    fprintf(fid,'%s
','Use Linear Decrease in Velocity as distance from centerline: ON');
else
    fprintf(fid,'%s
','Use Linear Decrease in Velocity as distance from centerline: OFF');
end
if elecMagImpactVelocity~=0
    % fprintf(fid,'%s
','***ELECTROMAGNET********************************************
    fprintf(fid,'%s %s
','Drop Velocity: ',num2str(elecMagImpactVelocity));
    fprintf(fid,'%s %s
','Consider Bounces: ',num2str(elecMagConsiderBounces));
    if elecMagConsiderBounces==true
        fprintf(fid,'%s %s
','Maxium Number of Bounces: ',num2str(elecMagMaximumNumberOfBounces));
    end
    fprintf(fid,'%s %s
','Number of Balls: ',num2str(elecMagNumOfBalls));
    fprintf(fid,'%s %s
','Drop Frequency: ',num2str(elecMagFreq));
    % fprintf(fid,'%s
','***END*****************************************************
else
    fprintf(fid,'%s %s
','Sensor Collision Frequency (Hz):',num2str(averageSensorCollisionFrequencyHertz));
end
fclose(fid);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

titleString = strrep(fileNamePrefix, ' ', '_');
tempPlot = newplot;
hold on;
plot(timeAxis,coverageDescreteDomain, 'linestyle', 'none', 'marker', 'x','markersize',3);
title(titleString);
xlabel(['Time (min) - Resolution: ', num2str(numberOfGridElements)]);
ylabel(['Coverage']);
saveas(tempPlot,[saveFilePath,fileNamePrefix,'_Coverage','.fig']);
saveas(tempPlot,[saveFilePath,fileNamePrefix,'_Coverage','.png'],'png');
cla reset;
clear tempPlot;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tempPlot = newplot;
hold on;
plot(actualStripTimeAxis,arcHeightHistory, 'linestyle', 'none', 'marker', '+', 'markersize',3);

if elecMagImpactVelocity==0
    % plot(actualStripTimeAxis,arcHeightHistory, 'linestyle', 'none', 'marker', 'o', 'markersize',3);
end

end
plot(timeAxis, arcHeightHistory, 'linestyle', 'none', 'marker', 'x', 'markersize', 5);
else
plot(timeAxis, arcHeightHistory, 'linestyle', 'none', 'marker', 'x', 'markersize', 5);
end
if graphExperimentalDataForComparision == true
plotExp(Strip617umSwitch, condition91Sa, elecMagImpactVelocity);
end
title(titleString);
xlabel(['Time (min) - Resolution: ', num2str(numberOfGridElements)]);
ylabel(['Arc Height (microns)']);
saveas(tempPlot, [saveFilePath, fileNamePrefix, '_ArcHeight', '.fig']);
saveas(tempPlot, [saveFilePath, fileNamePrefix, '_ArcHeight', '.png']);
cla reset;
clear tempPlot;
hold off;
clc;
clear all;

7.2.7. Plastic and elastic work code

Mathworks Inc. Matlab v7 Code

clc;
clear;
prefix='ISOEPP';
	prefix='PerLuThin';
	prefix='B1ln77um';
	prefix='PerLu77um';
	prefix='PerLu77um';
disp('Started...');
caseNumber=18;

if caseNumber == 1
    numberOfImpacts=10; % 10
    numberOfSubStepsPerLoadStepOrImpact=21;
    numberOfPaths=100;
    totalNumberOfPathsOutputed=100;
    outputPath='C:\_D\PwrLw617umMorePaths\17Output\';
    path{1}=['C:\_D\PwrLw617umMorePaths\17\'];
    prefix='PerLuThin';
    finiteElementModelLengthInMeters=(2/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
elseif caseNumber == 2
    numberOfImpacts=11; % 11
    numberOfSubStepsPerLoadStepOrImpact=2001;
    numberOfPaths=67;
    totalNumberOfPathsOutputed=100;
    outputPath='C:\_D\PwrLw617umMorePaths\timeOutput\';
    path{1}=['C:\_D\PwrLw617umMorePaths\time\'];
    prefix='PerLuThin';
    finiteElementModelLengthInMeters=(2/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
elseif caseNumber == 3
    numberOfImpacts=20; % 20
    numberOfSubStepsPerLoadStepOrImpact=60;
    numberOfPaths=50;
    totalNumberOfPathsOutputed=50;
    outputPath='C:\_D\PwrLw617umSubSteps\17Output\';
    path{1}=['C:\_D\PwrLw617umSubSteps\17\'];
    prefix='PerLuThin';
    finiteElementModelLengthInMeters=(2/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
elseif caseNumber == 4
    numberOfImpacts=12; % 12
    numberOfSubStepsPerLoadStepOrImpact=21;
    numberOfPaths=50;
    totalNumberOfPathsOutputed=50;
    outputPath='C:\_D\B1ln617um\17Output\';
    path{1}=['C:\_D\B1ln617um\17\'];
    prefix='ISOEPP';
    finiteElementModelLengthInMeters=(2/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
elseif caseNumber == 5
    numberOfImpacts=3; % 10 max
    numberOfSubStepsPerLoadStepOrImpact=61;
    numberOfPaths=50;
    totalNumberOfPathsOutputed=50;
    outputPath='C:\_D\B1ln617um\MoreSubstepsOutput\';
    path{1}=['C:\_D\B1ln617um\MoreSubsteps\'];
    prefix='ISOEPP';
    finiteElementModelLengthInMeters=(2/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
elseif caseNumber == 6
    numberOfImpacts=3; % 3 max
    numberOfSubStepsPerLoadStepOrImpact=100;
    numberOfPaths=200;
    totalNumberOfPathsOutputed=200;

    disp('Finished!')
outputPath='C:\_D\BiLn617um\TMoStepsPathsOutput\';
path{1}=[‘C:\\_D\BiLn617um\TMoStepsPaths\’];
prefix=’ISOEPP’;
finitElementModelLengthInMeters=(2/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
else if caseNumber==7
   numberOfImpacts=3; %3 max
   numberOfSubStepsPerLoadStepOrImpact=61;
   numberOfPaths=50;
   totalNumberOfPathsOutputed=50;
   outputPath='C:\\_D\BiLn617um\TMoMoreSubStepsGapOutput\';
   path{1}=[‘C:\\_D\BiLn617um\TMoMoreSubStepsGap\’];
   prefix=’ISOEPP’;
   finiteElementModelLengthInMeters=(2/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
else if caseNumber==8
   numberOfImpacts=3; %3 max
   numberOfSubStepsPerLoadStepOrImpact=61;
   numberOfPaths=50;
   totalNumberOfPathsOutputed=50;
   outputPath='C:\\_D\BiLn617um\TMoBigSubStepsGapOutput\';
   path{1}=[‘C:\\_D\BiLn617um\TMoBigSubStepsGap\’];
   prefix=’ISOEPP’;
   finiteElementModelLengthInMeters=(2/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
else if caseNumber==9
   numberOfImpacts=5; %3 max
   numberOfSubStepsPerLoadStepOrImpact=20;
   numberOfPaths=25;
   totalNumberOfPathsOutputed=25;
   outputPath='C:\\_D\BiLn617um\TMoStepsMoPathsOutput\';
   path{1}=[‘C:\\_D\BiLn617um\TMoStepsMoPaths\’];
   prefix=’ISOEPP’;
   finiteElementModelLengthInMeters=(1/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
else if caseNumber==10
   numberOfImpacts=5; %3 max
   numberOfSubStepsPerLoadStepOrImpact=20;
   numberOfPaths=25;
   totalNumberOfPathsOutputed=25;
   outputPath='C:\\_D\BiLn617um\TMoBigSubStepsOutput\';
   path{1}=[‘C:\\_D\BiLn617um\TMoBigSubSteps\’];
   prefix=’ISOEPP’;
else if caseNumber==11
   numberOfImpacts=5; %3 max
   numberOfSubStepsPerLoadStepOrImpact=20;
   numberOfPaths=25;
   totalNumberOfPathsOutputed=25;
   outputPath='C:\\_D\BiLn617um\MultiVelOutput\';
   numberOfDirectories=16;
   for i=1:numberOfDirectories
      subPath{i}=num2str(i);
   end
%   subPath{1}=’5’;
%   subPath{2}=’6’;
   for i=1:numberOfDirectories
      path{i}=[’C:\_D\BiLn617um\’,subPath{i},’\’];
   end
   prefix=’ISOEPP’;
else if caseNumber==12
   numberOfImpacts=5; %5 max
   numberOfSubStepsPerLoadStepOrImpact=20;
   numberOfPaths=25;
   totalNumberOfPathsOutputed=25;
   outputPath='C:\\_D\BiLn617um\MultiVelTestOutput\';
   numberOfDirectories=3;
   subPath{1}=num2str(3);
   subPath{2}=num2str(10);
   subPath{3}=num2str(16);
end
% for i=1:numberOfDirectories
%     subPath{i}=num2str(i);  
% end
for i=1:numberOfDirectories
    path{i}=['C:\D_BiLn617um\',subPath{i},'\'];
end
prefix='ISOEPP';
elseif caseNumber==16
    numberOfImpacts=100;  %5 max
    numberOfSubStepsPerLoadStepOrImpact=20;
    numberOfPaths=25;
    totalNumberOfPathsOutputed=(1/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
    outputPath='C:\D_BiLn617um\MultiVelOutput\';
    numberOfDirectories=1;
    for i=1:numberOfDirectories
        subPath{i}=num2str(i);
        end
        for i=1:numberOfDirectories
            path{i}=['C:\D_BiLn617um\',subPath{i},'\'];
        end
    prefix='PwrLwThin';
elseif caseNumber==17
    numberOfImpacts=100;  %5 max
    numberOfSubStepsPerLoadStepOrImpact=20;
    numberOfPaths=25;
    totalNumberOfPathsOutputed=25;
    finiteElementModelLengthInMeters=(1/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
    outputPath='C:\D_PwrLw617um\MultiVelOutput\';
    numberOfDirectories=1;
    for i=1:numberOfDirectories
        subPath{i}=num2str(i);
        end
        for i=1:numberOfDirectories
            path{i}=['C:\D_PwrLw617um\',subPath{i},'\'];
        end
    prefix='PwrLw617um';
elseif caseNumber==18
    numberOfImpacts=1;  %5 max
    numberOfSubStepsPerLoadStepOrImpact=20;
    numberOfPaths=25;
    totalNumberOfPathsOutputed=25;
    finiteElementModelLengthInMeters=(1/1000)*(numberOfPaths/totalNumberOfPathsOutputed);
    outputPath='C:\D_PwrLw617um\plstcOutput\';
    numberOfDirectories=1;
    for i=1:numberOfDirectories
        subPath{i}=num2str(i);
        end
        for i=1:numberOfDirectories
            path{i}=['C:\D_PwrLw617um\',subPath{i},'\'];
        end
    prefix='PwrLwThin';
else
    disp('Error: No such caseNumber');
    return;
end
nrgDistSig='NRGDIST';
nrNodesIntoDepth=100;
titleString=[prefix,' Impacts=',num2str(numberOfImpacts),' Paths=',num2str(numberOfPaths),'
SubSteps=',num2str(numberOfSubStepsPerLoadStepOrImpact)];
materialModelNumber=0;
materialModelElasticPerfectlyPlastic=1;
materialModelPowerLaw=2;
if (strcmp(prefix, 'PwrLwThin')||strcmp(prefix, 'PwrLw779um')||strcmp(prefix, 'PwrLw617umUy'))
    materialModelNumber=materialModelPowerLaw;
elseif (strcmp(prefix, 'BiLn779um')||strcmp(prefix, 'ISOEPP')||strcmp(prefix, 'BiLn617um'))
    materialModelNumber=materialModelElasticPerfectlyPlastic;
else
    disp('Error: Unrecognized File Prefix, Update Code.');
    return;
end
disp(path);
pathDistanceIncMeters=(finiteElementModelLengthInMeters/numberOfPaths);
modulusOfElasticity=71.7E9;  %modulus of elasticity (Pa)
materialConst_k  = 1.88e8;
hardeningCoeff_m = 0.242;
numberOfDirectories=length(path);
strainOfPointPathNumber=3;
strainOfPointNodeNumber=10;
if numberOfPaths < strainOfPointPathNumber
    disp('Reduced strain of point path Number');
end
programStartTime = clock();
lastProgressUpdateTime = clock();

col_NRGDIST_Z=1;
col_NRGDIST_EqvStress=2;
col_NRGDIST_EqvTotStrain=3;
col_NRGDIST_EqvPlasticStrain=4;

elasticEnergyDistributionPID=zeros(numberOfPaths+1,numberOfImpacts,numberOfDirectories);
plasticEnergyDissipationDistributionPID=zeros(numberOfPaths+1,numberOfImpacts,numberOfDirectories);
totalNumberOfSubSteps=(numberOfImpacts-1)*numberOfSubStepsPerLoadStepOrImpact + numberOfSubStepsPerLoadStepOrImpact;
totalElasticEnergyStateForPathPID=zeros(numberOfPaths+1,totalNumberOfSubSteps,numberOfDirectories);
pointCounter=0;

progressTarget=numberOfImpacts*numberOfDirectories*numberOfSubStepsPerLoadStepOrImpact*(numberOfPaths+1);
progressCounter=0;

for directoryNumber=1:numberOfDirectories
  for impactNumber=1:numberOfImpacts
    filename=[prefix,'_',num2str(impactNumber),'_',nrgDistSig,'_',num2str(numberOfPaths),'_substep_','.csv'];
    file = dir([path{directoryNumber},filename]);
    if (~isempty(file))
      if (file.isdir==0)
        subStepCorrectionByImpactNumber{directoryNumber}(impactNumber)=subStepCorrectionByImpactNumber{directoryNumber}(impactNumber )+1;
      end
      else
        disp([filename, ' not found in ', path{directoryNumber}]);
        return;
      end
    end
  end
end

for directoryNumber=1:numberOfDirectories
  for pathNumber=0:numberOfPaths
    if pathNumber==0
      radiusInner=0;
    else
      radiusInner=(pathNumber*pathDistanceIncMeters)-(pathDistanceIncMeters/2);
    end
    radiusOuter=(pathNumber*pathDistanceIncMeters)+(pathDistanceIncMeters/2);
    totalPlasticWorkDoneInPath=0;
    totalElasticEnergyStateForPath=0;
    totalElasticEnergyPath=0;
    for impactNumber=1:numberOfImpacts
      plasticDepth(impactNumber,directoryNumber) = 0;
      numberOfSubStepsForLoop=numberOfSubStepsPerLoadStepOrImpact + subStepCorrectionByImpactNumber{directoryNumber}(impactNumber);      if (numberOfSubStepsForLoop==1||numberOfSubStepsForLoop==11)
        disp([filename, ' not found in ', path{directoryNumber}]);
        return;
      end
      filename=[prefix,'_',num2str(impactNumber),'_',nrgDistSig,'_',num2str(pathNumber),'_substep_','.csv'];
      data=(csvread([path{directoryNumber},filename,'.csv']));
      prevData=(csvread([path{directoryNumber},filenamePrev,'.csv']));
      numberOfNodesIntoDepth=length(data(:,1));
      nodeSpacingIntoDepth=abs(data(1,col_NRGDIST_Z)-data(2,col_NRGDIST_Z));
      for i=1:numberOfNodesIntoDepth
        if nodeNumber==1
          deltaZupper=0;
          deltaZlower=abs(data(nodeNumber+1,col_NRGDIST_Z)-data(nodeNumber,col_NRGDIST_Z))/2;
        elseif nodeNumber==numberOfNodesIntoDepth
          deltaZupper=abs(data(nodeNumber,col_NRGDIST_Z)-data(nodeNumber-1,col_NRGDIST_Z))/2;
        else
          deltaZupper=abs(data(nodeNumber,col_NRGDIST_Z)-data(nodeNumber-1,col_NRGDIST_Z))/2;
          deltaZlower=abs(data(nodeNumber,nodeNumber,col_NRGDIST_Z)-data(nodeNumber,col_NRGDIST_Z))/2;
        end
        depth=deltaZupper+deltaZlower;
        volume=pi*(radiusOuter^2-radiusInner^2)*depth;
```matlab
effTotalStrainPrev = prevData(nodeNumber,col_NRGDIST_EqvTotStrain);
effTotalStrain = data(nodeNumber,col_NRGDIST_EqvTotStrain);
effPlasticStrainPrev = prevData(nodeNumber,col_NRGDIST_EqvPlasticStrain);
effPlasticStrain = data(nodeNumber,col_NRGDIST_EqvPlasticStrain);
effElasticStrainPrev = effTotalStrainPrev - effPlasticStrainPrev;
effElasticStrain = effTotalStrain - effPlasticStrain;
effectiveStressMidpoint = (data(nodeNumber,col_NRGDIST_EqvStress) + ... prevData(nodeNumber,col_NRGDIST_EqvStress))/2;

if pathNumber==0
    if nodeNumber==1
        prevNodePlasticStrain = prevData(nodeNumber-1,col_NRGDIST_EqvPlasticStrain);
        nodePlasticStrain = prevData(nodeNumber,col_NRGDIST_EqvPlasticStrain);
        if (prevNodePlasticStrain~=0)&&(nodePlasticStrain==0)
            plasticDepth(impactNumber,directoryNumber) = (nodeNumber-1)*nodeSpacingIntoDepth;
        end
    end
end

if effectiveStressMidpoint < 0
disp('Error: Effective Stress Less Than Zero! Exiting...');
return;
end
deltaEffElasticStrain = effElasticStrain - effElasticStrainPrev; %elastic strain energy can
deltaElasticEnergy = volume * (deltaEffElasticStrain*effectiveStressMidpoint);
totalElasticEnergyInPath = totalElasticEnergyInPath + deltaElasticEnergy;
deltaEffectivePlasticStrain = abs(effPlasticStrain - effPlasticStrainPrev);
deltaPlasticWorkPerUnitVolume = deltaEffectivePlasticStrain * effectiveStressMidpoint;
deltaPlasticWork = deltaPlasticWorkPerUnitVolume * volume;
totalPlasticWorkDoneInPath = totalPlasticWorkDoneInPath + deltaPlasticWork;

if numberOfSubStepsForLoop==subStepNumber
    elasticEnergyDistributionPID(pathNumber+1,impactNumber,directoryNumber)=totalElasticEnergyInPath;
    end
if (nodeNumber==strainOfPointNodeNumber)&&(pathNumber==strainOfPointPathNumber)
    if pointCounter==0
        pointCounter=pointCounter+1;
        strainOfPoint(pointCounter,1,directoryNumber)=prevData(nodeNumber,col_NRGDIST_EqvPlasticStrain);
        strainOfPoint(pointCounter,2,directoryNumber)=prevData(nodeNumber,col_NRGDIST_EqvTotStrain);
        strainOfPoint(pointCounter,3,directoryNumber)=strainOfPoint(pointCounter,2,directoryNumber)-strainOfPoint(pointCounter,1,directoryNumber);
        strainOfPoint(pointCounter,4,directoryNumber)=prevData(nodeNumber,col_NRGDIST_EqvStress);
        strainOfPoint(pointCounter,5,directoryNumber)=modulusOfElasticity*strainOfPoint(pointCounter,3);
    end
    pointCounter=pointCounter+1;
    strainOfPoint(pointCounter,1,directoryNumber)=data(nodeNumber,col_NRGDIST_EqvPlasticStrain);
    strainOfPoint(pointCounter,2,directoryNumber)=data(nodeNumber,col_NRGDIST_EqvTotStrain);
    strainOfPoint(pointCounter,3,directoryNumber)=strainOfPoint(pointCounter,2,directoryNumber)-strainOfPoint(pointCounter,1,directoryNumber);
    strainOfPoint(pointCounter,4,directoryNumber)=data(nodeNumber,col_NRGDIST_EqvStress);
    strainOfPoint(pointCounter,5,directoryNumber)=modulusOfElasticity*strainOfPoint(pointCounter,3);
end

cummulativeStepNumber=0;
for imp=1:impactNumber
    if imp < impactNumber
        cummulativeStepNumber=cummulativeStepNumber+ numberOfSubStepsPerLoadStepOrImpact + subStepCorrectionByImpactNumber(directoryNumber)(imp);
    else
        cummulativeStepNumber=cummulativeStepNumber+subStepNumber;
    end
    plasticWorkEachStepPSD(pathNumber+1,cummulativeStepNumber,directoryNumber) = totalPlasticWorkDoneInPath;
totalElasticEnergyStateForPathPSD(pathNumber+1,cummulativeStepNumber,directoryNumber) = totalElasticEnergyInPath;
end
    end
    end
end
currentTime = clock();
if (etime(currentTime, lastProgressUpdateTime)>2)
    lastProgressUpdateTime = clock();
    percentComplete=100*progressCounter/progressTarget;
    if percentComplete > 100
        percentComplete = 100;
    end
    elapsedTime = etime(currentTime, programStartTIme);
    eta = (100-percentComplete)*(elapsedTime/percentComplete)}/${60};
    if eta < 0
        eta=0;
    end
    disp([num2str(percentComplete),', ','%', 'Complete', ' ', num2str(eta), ' min left ','
    num2str(elapsedTime/60), ' min Elapsed ','
    num2str( (elapsedTime(60)+eta ) , ' Total Run Time in min' ]);}
    end
end
end %directory Loop
```

164
disp('Making graphs...');

numberOfSubsteps=length(plasticWorkEachStepPSD(1,:,1));
plasticWorkEachStep=zeros(numberOfSubsteps,numberOfDirectories);
totalElasticEnergyStateEachStep=zeros(numberOfSubsteps,numberOfDirectories);
for directoryNumber=1:numberOfDirectories
    for k=1:numberOfSubsteps
        for pathNumber=0:numberOfPaths
            plasticWorkEachStep(k,directoryNumber)=plasticWorkEachStepPSD(pathNumber+1,k,directoryNumber)+plasticWorkEachStep(k,directoryNumber);
        end
    end
end %directory Loop

pathDistanceAxisMilliMeters = pathDistanceIncMeters*1000*(0:numberOfPaths);
plasticEnergyDissipationID = zeros(numberOfImpacts,numberOfDirectories);
totalEnergyDistributionID = zeros(numberOfImpacts,numberOfDirectories);
elasticEnergyStoredID=zeros(numberOfImpacts,numberOfDirectories);
totalEnergyDistributionPID = zeros(numberOfPaths,numberOfImpacts,numberOfDirectories);
for directoryNumber=1:numberOfDirectories
    for impactNumber=1:numberOfImpacts
        for pathNumber=0:numberOfPaths
            totalEnergyDistributionPID(pathNumber+1,impactNumber,directoryNumber)=plasticEnergyDissipationDistributionPID(pathNumber+1,impactNumber,directoryNumber)+elasticEnergyDistributionPID(pathNumber+1,impactNumber,directoryNumber);
            plasticEnergyDissipationID(impactNumber,directoryNumber) = plasticEnergyDissipationDistributionPID(pathNumber+1,impactNumber,directoryNumber);
            elasticEnergyStoredID(impactNumber,directoryNumber)= elasticEnergyDistributionPID(pathNumber+1,impactNumber,directoryNumber);
        end
    end
end

for directoryNumber=1:numberOfDirectories
    cla reset;
    clear tempPlot;
tempPlot = newplot;
hold on;
plot(pathDistanceAxisMilliMeters,totalEnergyDistributionPID(:,:,directoryNumber))
xlabel('Radial Distance From Centerline (mm)');
ylabel('Energy (cumulative over impacts) (J)');
title(['(Elas+Plas) nrg Trans Dist ',titleString]);
hold off;
saveas(tempPlot,[outputPath,'totalEnergyDistributionPID','_',num2str(directoryNumber),'.fig']);
saveas(tempPlot,[outputPath,'totalEnergyDistributionPID','_',num2str(directoryNumber),'.png']);
cla reset;

    clear tempPlot;
tempPlot = newplot;
hold on;
plot(pathDistanceAxisMilliMeters,plasticEnergyDissipationDistributionPID(:,:,directoryNumber))
xlabel('Radial Distance From Centerline (mm)');
ylabel('Energy (J)');
title(['nrg Dist Plastic ',titleString]);
hold off;
saveas(tempPlot,[outputPath,'EnergyDistributionByPlasticStrain','_',num2str(directoryNumber),'.fig']);
saveas(tempPlot,[outputPath,'EnergyDistributionByPlasticStrain','_',num2str(directoryNumber),'.png']);
cla reset;

    clear tempPlot;
tempPlot = newplot;
hold on;
plot(pathDistanceAxisMilliMeters,elasticEnergyDistributionPID(:,:,directoryNumber))
xlabel('Radial Distance From Centerline (mm)');
ylabel('Energy (J)');
title(['nrg Dist Elastic ',titleString]);
hold off;
saveas(tempPlot,[outputPath,'EnergyDistributionByElasticStrain','_',num2str(directoryNumber),'.fig']);
saveas(tempPlot,[outputPath,'EnergyDistributionByElasticStrain','_',num2str(directoryNumber),'.png']);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cla reset;

    clear tempPlot;
tempPlot = newplot;
hold on;
plot(pathDistanceAxisMilliMeters,totalEnergyDistributionPID(:,:,directoryNumber))
xlabel('Radial Distance From Centerline (mm)');
ylabel('Energy (cumulative over impacts) (J)');
title(['(Elas+Plas) nrg Trans Dist ',titleString]);
hold off;
saveas(tempPlot,[outputPath,'totalEnergyDistributionPID','_',num2str(directoryNumber),'.fig']);
saveas(tempPlot,[outputPath,'totalEnergyDistributionPID','_',num2str(directoryNumber),'.png']);
cla reset;

    clear tempPlot;
tempPlot = newplot;
hold on;
plot(pathDistanceAxisMilliMeters,plasticEnergyDissipationDistributionPID(:,:,directoryNumber))
xlabel('Radial Distance From Centerline (mm)');
ylabel('Energy (J)');
title(['nrg Dist Plastic ',titleString]);
hold off;
saveas(tempPlot,[outputPath,'EnergyDistributionByPlasticStrain','_',num2str(directoryNumber),'.fig']);
saveas(tempPlot,[outputPath,'EnergyDistributionByPlasticStrain','_',num2str(directoryNumber),'.png']);
cla reset;

    clear tempPlot;
tempPlot = newplot;
hold on;
plot(pathDistanceAxisMilliMeters,elasticEnergyDistributionPID(:,:,directoryNumber))
xlabel('Radial Distance From Centerline (mm)');
ylabel('Energy (J)');
title(['nrg Dist Elastic ',titleString]);
hold off;
saveas(tempPlot,[outputPath,'EnergyDistributionByElasticStrain','_',num2str(directoryNumber),'.fig']);
saveas(tempPlot,[outputPath,'EnergyDistributionByElasticStrain','_',num2str(directoryNumber),'.png']);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cla reset;

    clear tempPlot;
tempPlot = newplot;
hold on;
plot(pathDistanceAxisMilliMeters,totalEnergyDistributionPID(:,:,directoryNumber))
xlabel('Radial Distance From Centerline (mm)');
ylabel('Energy (cumulative over impacts) (J)');
title(['(Elas+Plas) nrg Trans Dist ',titleString]);
hold off;
saveas(tempPlot,[outputPath,'totalEnergyDistributionPID','_',num2str(directoryNumber),'.fig']);
saveas(tempPlot,[outputPath,'totalEnergyDistributionPID','_',num2str(directoryNumber),'.png']);
plot(strainOfPoint(:,1,directoryNumber),strainOfPoint(:,4,directoryNumber),'-bo'); % plastic stress-strain history
plot(strainOfPoint(:,2),strainOfPoint(:,4)); % total strain
plot(strainOfPoint(:,3),strainOfPoint(:,5),'-rx'); % \Sigma = E*\varepsilon
for i=1:length(strainOfPoint(:,1,1))
    strainOfPoint(i,6)=41E6;
    if materialModelNumber==materialModelPowerLaw
        strainOfPoint(i,8,directoryNumber)=materialConst_k*strainOfPoint(i,3)^hardeningCoeff_m; % powerlaw elastic stress
    else % materialModelNumber==materialModelElasticPerfectlyPlastic
        strainOfPoint(i,8,directoryNumber)=modulusOfElasticity*strainOfPoint(i,3); % Elastic perfectly plastic elastic stress
    end
end
numberOfPointsUsedToDrawConstitutiveLaw=500;
numberOfRowsInStrainOfPoint=length(strainOfPoint(:,1,1));
strainInc=strainOfPoint(numberOfRowsInStrainOfPoint,1)/numberOfPointsUsedToDrawConstitutiveLaw;
strainConst=zeros(numberOfPointsUsedToDrawConstitutiveLaw+1);
pwrLwStressConst=zeros(numberOfPointsUsedToDrawConstitutiveLaw+1);
bilnStressConst=zeros(numberOfPointsUsedToDrawConstitutiveLaw+1);
yieldStress=41e6; % Pa
for i=1:numberOfPointsUsedToDrawConstitutiveLaw+1
    strainConst(i)=strainInc*(i-1);
    pwrLwStressConst(i)=materialConst_k*strainConst(i)^hardeningCoeff_m;
    if modulusOfElasticity*strainConst(i) <= yieldStress
        bilnStressConst(i)=modulusOfElasticity*strainConst(i);
    else
        bilnStressConst(i)=yieldStress;
    end
end
% plot(strainOfPoint(:,1),strainOfPoint(:,6),'-'); % yield stress
% plot(strainOfPoint(:,1),strainOfPoint(:,7),'--g'); % constitutive law.
plot(strainConst,pwrLwStressConst,'--g'); % constitutive law.
plot(strainConst,bilnStressConst,'--y'); % constitutive law.
plot(strainOfPoint(:,3),strainOfPoint(:,8),'-.sr'); % elastic stress-strain history
ylabel('Von Mises Effective Stress');
xlabel(['Eff Strain - ',titleString]);
title(['NodeNum=',num2str(strainOfPointNodeNumber),' PathNum= ',num2str(strainOfPointPathNumber),' plasstrn -blue o, elstrn -red x, constit law green']);
hold off;
saveas(tempPlot,'outputPath','strainTimeHistoryOfPoint','_',num2str(directoryNumber),'.fig');
saveas(tempPlot,'outputPath','strainTimeHistoryOfPoint','_',num2str(directoryNumber),'.png');
end % directory Loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
filenameMotion=[prefix,'_bllmtn.csv'];
for directoryNumber=1:numberOfDirectories
    if directoryNumber>1
        clear dataMotion;
    end
dataMotion=(csvread([path{directoryNumber},filenameMotion], 1,0));
    for i=1:length(dataMotion)-1
        if dataMotion(i+1,1)<dataMotion(i,1)
            break;
        end
    end
dataMotion=dataMotion(1:i,:);
    numberOfCurvesPlusOne=length(dataMotion(:,1));
    mediaMass=0.001031; % Kg
    kineticEnergyEveryStep{directoryNumber}=zeros(totalNumberOfStepsFromBallMotion,1);
    for k=1:totalNumberOfStepsFromBallMotion
        kineticEnergyEveryStep{directoryNumber}(k)=dataMotion(k,3)^2*mediaMass/2;
        kineticChangeEnergyEveryStep{directoryNumber}(k)=kineticEnergyEveryStep{directoryNumber}(k)-kineticEnergyEveryStep{directoryNumber}(k-1);
    end
    j=0;
    rebound=0;
    vIncident{directoryNumber}(1)=0;
    vRebound{directoryNumber}(1)=0;
    for i=2:length(dataMotion(:,1))
        if rebound==0
            if dataMotion(i-1,4)==0 && dataMotion(i,4)>0 && dataMotion(i,5)~=dataMotion(i-1,5)
                rebound=1;
            end
            vIncident{directoryNumber}(j+1)=dataMotion(i-1,3);
        end
        vRebound{directoryNumber}(j+1)=vRebound{directoryNumber}(j);
        rebound=0;
        j=j+1;
    end
end
if (dataMotion(i,4)==0)&&(dataMotion(i-1,4)==0)
    rebound=0;
    vRebound(directoryNumber)(j)=dataMotion(i,3);
end
end
i=0;
energyStateDueToAddedEnergySubStep(directoryNumber)=zeros(totalNumberOfSubSteps,1);
for impactNumber=1:numberOfImpacts
    for stepNumber=1:numberOfSubStepsPerLoadStepOrImpact
        i=i+1;
        energyStateDueToAddedEnergySubStep(directoryNumber)(i)=0.5*mediaMass*(impactNumber-1)^2;
    end
end
%incidentEnergyCum = zeros(length(vIncident),1);
%reboundEnergyCum = zeros(length(vIncident),1);
%ballsAbsorbedEnergyCum = zeros(length(vIncident),1);
for i = 1:length(vIncident{directoryNumber})
    e{directoryNumber}(i)=vRebound{directoryNumber}(i)/vIncident{directoryNumber}(i);
    incidentEnergy(i)=0.5*mediaMass*vIncident{directoryNumber}(i)^2;
    reboundEnergy(i)=0.5*mediaMass*vRebound{directoryNumber}(i)^2;
    absorbedEnergy(i)=incidentEnergy(i)-reboundEnergy(i);
    if i==1
        incidentEnergyCum(directoryNumber)(1)=incidentEnergy(i);
        reboundEnergyCum(directoryNumber)(1)=reboundEnergy(i);
        ballsAbsorbedEnergyCum(directoryNumber)(1)=absorbedEnergy(i);
    else
        incidentEnergyCum(directoryNumber)(i)=incidentEnergyCum(directoryNumber)(i-1)+incidentEnergy(i);
        reboundEnergyCum(directoryNumber)(i)=reboundEnergyCum(directoryNumber)(i-1)+reboundEnergy(i);
        ballsAbsorbedEnergyCum(directoryNumber)(i)=ballsAbsorbedEnergyCum(directoryNumber)(i-1)+absorbedEnergy(i);
    end
end
%directory Loop

for directoryNumber=1:numberOfDirectories
    for k=1:totalNumberOfSubSteps
        totalEnergyEachSubStep(k,directoryNumber)=totalElasticEnergyStateEachStep(k,directoryNumber)+plasticWorkEachStep(k,directoryNumber);
    end
    cla reset;
    clear tempPlot;
    tempPlot = newplot;
    plotData=zeros(totalNumberOfSubSteps,6);
    for k=1:totalNumberOfSubSteps
        plotData(k,1)=kineticEnergyEveryStep(directoryNumber)(k);
        plotData(k,2)=plasticWorkEachStep(k,directoryNumber);
        plotData(k,3)=totalElasticEnergyStateEachStep(k,directoryNumber);
        plotData(k,4)=totalEnergyEachSubStep(k,directoryNumber);
        %plotData(k,5)=kineticEnergyEveryStep(directoryNumber)(k)+energyStateDueToAddedEnergySubStep(directoryNumber)(k);
        %plotData(k,6)=energyStateDueToAddedEnergySubStep(directoryNumber)(k);
    end
    legendStripMatrix{1}='Ball.KineticEnergy';
    legendStripMatrix{2}='Strip.PlasticWork';
    legendStripMatrix{3}='Strip.ElasticEnergy';
    legendStripMatrix{4}='Strip.TotalEnergy';
    %legendStripMatrix{5}='Ball.AddedEnergy';
    %legendStripMatrix{6}='Ball.KineticEnergyWithAddedEnergy';
    plot(plotData);
    legend(legendStripMatrix,'location','EastOutside');
ylabel('Energy State (J)');
xlabel('Sub-Step Number');
title(titleString);
saveas(tempPlot,
      [outputPath,'EnergyState','_',num2str(directoryNumber),'.fig']);
saveas(tempPlot,
      [outputPath,'EnergyState','_',num2str(directoryNumber),'.png'])
end

for k=1:totalNumberOfSubSteps
    plotData(k,1)=kineticChangeEnergyEveryStep(directoryNumber)(k);
    for k=2:totalNumberOfSubSteps
        plotData(k-1,2)=plasticWorkEachStep(k,directoryNumber)-plasticWorkEachStep(k-1,directoryNumber);
        plotData(k-1,3)=totalEnergyEachSubStep(k,directoryNumber)-totalEnergyEachSubStep(k-1,directoryNumber);
    end
    clear legendStripMatrix;
    legendStripMatrix{1}='Ball.deltaKineticEnergy';
    legendStripMatrix{2}='Strip.deltaPlasticWork';
    legendStripMatrix{3}='Strip.deltaTotalEnergy';
end
plot(plotData);
legend(legendStripMatrix,'location','northwest');
ylabel('Energy State (J)');
xlabel('Sub-Step Number');
title(titleString);
saveas(tempPlot,[outputPath,'DeltaEnergyState','_','num2str(directoryNumber),'_','.fig']);
saveas(tempPlot,[outputPath,'DeltaEnergyState','_','num2str(directoryNumber)','_','.png']');
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
clear plotData;
dataPointsForPlotData=length(plasticEnergyDissipationID(:,1));
plotData=zeros(dataPointsForPlotData,3);
for k=1:dataPointsForPlotData
    plotData(k,1)= plasticEnergyDissipationID(k,directoryNumber);
    plotData(k,2)= elasticEnergy StoredID(k,directoryNumber) + plasticEnergyDissipationID(k,directoryNumber);
    plotData(k,3)= ballsAbsorbedEnergyCum(directoryNumber)(k);
end
plot(plotData);
legendStripMatrix{1}='Plastic Energy Dissipation';
legendStripMatrix{2}='Total Energy Transfered To Target';
legendStripMatrix{3}='Ball Absorbed Energy Cummulative';
legend(legendStripMatrix, 'Location','NorthWest');
ylabel('Energy Absorbed (j)');
xlabel('Impact Number');
title(titleString);
saveas(tempPlot,[outputPath,'absorbedEnergy','_','num2str(directoryNumber)','.fig']);
saveas(tempPlot,[outputPath,'absorbedEnergy','_','num2str(directoryNumber)','.png']');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalEnergyRatioDistributionMaximumID=zeros(numberOfImpacts, numberOfDirectories);
totalEnergyDistributionRatiosPID=zeros(numberOfPaths, numberOfImpacts, numberOfDirectories);
for directoryNumber=1:numberOfDirectories
    for impactNumber=1:numberOfImpacts
        for pathNumber=0:numberOfPaths
            totalEnergyDistributionRatiosPID(pathNumber+1,impactNumber,directoryNumber)=
                totalEnergyDistributionPID(pathNumber+1,impactNumber,directoryNumber)/
                totalEnergyDistributionID(impactNumber,directoryNumber);
        maxE=max(totalEnergyDistributionRatiosPID(:,impactNumber,directoryNumber));
        totalEnergyRatioDistributionMaximumID(impactNumber,directoryNumber)=maxE;
    end
    end
    end

totalEnergyDistributionNormalizedToMaximumPID=zeros(numberOfPaths+1, numberOfImpacts, numberOfDirectories);
for directoryNumber=1:numberOfDirectories
    for impactNumber=1:numberOfImpacts
        maxE=max(totalEnergyDistributionPID(:,impactNumber,directoryNumber));
        for pathNumber=0:numberOfPaths
            totalEnergyDistributionNormalizedToMaximumPID(pathNumber+1,impactNumber,directoryNumber)=
                totalEnergyDistributionID(pathNumber+1,impactNumber,directoryNumber)/maxE;
        end
    end
end

zoneLengthInPathsID = zeros(numberOfImpacts,numberOfDirectories);
zoneLengthMilliMetersInterpolatedID = zeros(numberOfImpacts,numberOfDirectories);
targetRatio=0.95;
for directoryNumber=1:numberOfDirectories
    for impactNumber=1:numberOfImpacts
        integratedRatio=0;
        for pathNumber=0:numberOfPaths
            currentRatio = totalEnergyDistributionRatiosPID(pathNumber+1,impactNumber,directoryNumber); %0.98
            previousRatio = integratedRatio; %0
            integratedRatio = currentRatio + integratedRatio;
            if pathNumber<numberOfPaths-1
                nextRatio = integratedRatio + totalEnergyDistributionRatiosPID(pathNumber+2,impactNumber,directoryNumber);
            end
            if integratedRatio>targetRatio
                break;
            else
                nextRatio = 1;
                disp('Warning: Consider larger path layout');
            end
        end
        previousPathCounter=(pathNumber-1); %0
        nextPathCounter=(pathNumber); %0

end

168
if pathNumber==0
    inbetweenPathCounter=0;
else
    inbetweenPathCounter = nextPathCounter + ... ((targetRatio-nextRatio)/(previousRatio-nextRatio))*(previousPathCounter-nextPathCounter); %-0.04
end
zoneLengthInPathsID(impactNumber,directoryNumber)=inbetweenPathCounter;
zoneLengthMilliMetersInterpolatedID(impactNumber,directoryNumber)=inbetweenPathCounter*(pathDistanceIncMeters*1000);
end

cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    plot(vIncident{directoryNumber}(1:numberOfImpacts),zoneLengthMilliMetersInterpolatedID(:,directoryNumber));
    ylabel(['Zone Length (mm) to include ', num2str(targetRatio*100),' of stored energy']);
    xlabel('Impact Velocity');
    title(titleString);
    hold off;
saveas(tempPlot,[outputPath,'zoneLength','.fig']);
saveas(tempPlot,[outputPath,'ZoneLength','.png']);
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    plot(vIncident{directoryNumber}(1:numberOfImpacts),zoneLengthMilliMetersInterpolatedNewID(:,directoryNumber),'
    hold off;
saveas(tempPlot,[outputPath,'zoneLengthEabs','.fig']);
saveas(tempPlot,[outputPath,'ZoneLengthEabs','.png']);
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    fprintf(fileId,'%s,', ['V (m/s) ',titleString]);
    fprintf(fileId,'%s,', ['Impact Number ',titleString]);
    fprintf(fileId,'%s,', ['Eabs (J) ',titleString]);
    fprintf(fileId,'%s,', ['95%NRG_Zone(mm) ',titleString]);
    fprintf(fileId,'%s,', ['totalEnergyDistributionMaximumID(J) ',titleString]);
    fprintf(fileId, '\n');
for impactNumber=1:numberOfImpacts
    fprintf(fileId,'%s,',num2str(vIncident{directoryNumber}(impactNumber)));
    fprintf(fileId,'%s,',num2str(impactNumber));
    fprintf(fileId,'%s,', num2str(ballsAbsorbedEnergyCum{directoryNumber}(impactNumber)));
    fprintf(fileId,'%s,', num2str( zoneLengthMilliMetersInterpolatedID(impactNumber,directoryNumber) ));
    fprintf(fileId,'%s,', num2str( totalEnergyDistributionMaximumID(impactNumber,directoryNumber) ));
end
end
fclose(fileId);

cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    plot(ballsAbsorbedEnergyCum{directoryNumber}(1:numberOfImpacts),totalEnergyDistributionMaximumID(:,directoryNumber),'
    hold off;
saveas(tempPlot,[outputPath,'totalEnergyDistributionMaximumID','.fig']);
saveas(tempPlot,[outputPath,'totalEnergyDistributionMaximumID','.png']);
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
plot(ballsAbsorbedEnergyCum{directoryNumber}(1:numberOfImpacts),totalEnergyRatioDistributionMaximumID(:,directoryNumber));
end
ylabel('Energy Transfer Distribution Ratio Maxima (J)');
xlabel('Energy Absorbed');
title(titleString);
hold off;
saveas(tempPlot,[outputPath,'totalEnergyRatioDistributionMaximumID','.fig']);
saveas(tempPlot,[outputPath,'totalEnergyRatioDistributionMaximumID','.png'],'png');

absorbedEnergyDistributionRatiosAreaStretchedAxisPID=zeros(numberOfPaths, numberOfImpacts, numberOfDirectories);
for directoryNumber=1:numberOfDirectories
    for impactNumber=1:numberOfImpacts
        for pathNumber=0:numberOfPaths
            if zoneLengthMilliMetersInterpolatedID(impactNumber,directoryNumber)==0
                absorbedEnergyDistributionRatiosAreaStretchedAxisPID(pathNumber+1,impactNumber,directoryNumber)=pathNumber/1;
            else
                absorbedEnergyDistributionRatiosAreaStretchedAxisPID(pathNumber+1,impactNumber,directoryNumber)= ...
                    pathNumber^2/zoneLengthInPathsID(impactNumber,directoryNumber)^2;
            end
        end
    end
end

absorbedEnergyDistributionRatiosLengthStretchedAxisPID=zeros(numberOfPaths, numberOfImpacts, numberOfDirectories);
for directoryNumber=1:numberOfDirectories
    for impactNumber=1:numberOfImpacts
        for pathNumber=0:numberOfPaths
            if zoneLengthMilliMetersInterpolatedID(impactNumber,directoryNumber)==0
                absorbedEnergyDistributionRatiosLengthStretchedAxisPID(pathNumber+1,impactNumber,directoryNumber)=pathNumber/1;
            else
                absorbedEnergyDistributionRatiosLengthStretchedAxisPID(pathNumber+1,impactNumber,directoryNumber)= ...
                    pathNumber/zoneLengthInPathsID(impactNumber,directoryNumber);
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c=0;
if numberOfImpacts>numberOfPaths
    numberOfMarkers=numberOfImpacts;
else
    numberOfMarkers=numberOfPaths;
end
for j=1:ceil(numberOfMarkers/17)
    for i=1:17
        c=c+1;
        if i==1 plotStylesMarkers{c}='-s';
        elseif i==2 plotStylesMarkers{c}='-+';
        elseif i==3 plotStylesMarkers{c}='-o';
        elseif i==4 plotStylesMarkers{c}='-*';
        elseif i==5 plotStylesMarkers{c}='-x';
        elseif i==6 plotStylesMarkers{c}='-.';
        elseif i==7 plotStylesMarkers{c}='-d';
        elseif i==8 plotStylesMarkers{c}='-v';
        elseif i==9 plotStylesMarkers{c}='--s';
        elseif i==10 plotStylesMarkers{c}='--+';
        elseif i==11 plotStylesMarkers{c}='--o';
        elseif i==12 plotStylesMarkers{c}='--*';
        elseif i==13 plotStylesMarkers{c}='--x';
        elseif i==14 plotStylesMarkers{c}='--. ;
        elseif i==15 plotStylesMarkers{c}='--d';
        elseif i==16 plotStylesMarkers{c}='--v';
        else plotStylesMarkers{c}=':s';
        end
    end
end
c=0;
if numberOfImpacts>numberOfPaths
    numberOfColors=numberOfImpacts;
else
    numberOfColors=numberOfPaths;
end
for j=1:ceil(numberOfColors/7)
    for i=1:7
        c=c+1;
        if i==1 colors{c}='r';
        elseif i==2 colors{c}='b';
        elseif i==3 colors{c}='g';
        elseif i==4 colors{c}='c';
        elseif i==5 colors{c}='m';
        elseif i==6 colors{c}='y';
        else colors{c}='k';
        end
    end
end

for j=1:ceil(numberOfMarkers/17)
    for i=1:17
        c=c+1;
        if i==1 plotStylesMarkers{c}='-s';
        elseif i==2 plotStylesMarkers{c}='-+';
        elseif i==3 plotStylesMarkers{c}='-o';
        elseif i==4 plotStylesMarkers{c}='-*';
        elseif i==5 plotStylesMarkers{c}='-x';
        elseif i==6 plotStylesMarkers{c}='-.';
        elseif i==7 plotStylesMarkers{c}='-d';
        elseif i==8 plotStylesMarkers{c}='-v';
        elseif i==9 plotStylesMarkers{c}='--s';
        elseif i==10 plotStylesMarkers{c}='--+';
        elseif i==11 plotStylesMarkers{c}='--o';
        elseif i==12 plotStylesMarkers{c}='--*';
        elseif i==13 plotStylesMarkers{c}='--x';
        elseif i==14 plotStylesMarkers{c}='--. ;
        elseif i==15 plotStylesMarkers{c}='--d';
        elseif i==16 plotStylesMarkers{c}='--v';
        else plotStylesMarkers{c}=':s';
        end
    end
end
for j=1:ceil(numberOfColors/7)
    for i=1:7
        c=c+1;
        if i==1 colors{c}='r';
        elseif i==2 colors{c}='b';
        elseif i==3 colors{c}='g';
        elseif i==4 colors{c}='c';
        elseif i==5 colors{c}='m';
        elseif i==6 colors{c}='y';
        else colors{c}='k';
        end
    end
end
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
xlim([0 1]);
for i=1:numberOfDirectories
    for j=1:numberOfImpacts
        styleString=[plotStylesMarkers{i},colors{i}];
        plot(absorbedEnergyDistributionRatiosLengthStretchedAxisPID(:,j,i),totalEnergyDistributionRatiosPID(:,j,i),styleString);
    end
end
ylabel('ratio of total energy');
xlabel('Normalized To Zone Length');
hold off;
saveas(tempPlot,[outputPath,'EnergyDistributionRatioAxisLengthStretch','.fig']);
saveas(tempPlot,[outputPath,'EnergyDistributionRatioAxisLengthStretch','.png']);
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
xlim([0 1]);
for i=1:numberOfDirectories
    for j=1:numberOfImpacts
        styleString=[plotStylesMarkers{i},colors{i}];
        plot(absorbedEnergyDistributionRatiosAreaStretchedAxisPID(:,j,i),totalEnergyDistributionRatiosPID(:,j,i),styleString);
    end
end
ylabel('ratio of total energy');
xlabel('Normalized To Zone Area');
hold off;
saveas(tempPlot,[outputPath,'EnergyDistributionRatioAxisAreaStretch','.fig']);
saveas(tempPlot,[outputPath,'EnergyDistributionRatioAxisAreaStretch','.png']);
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
xlim([0 1]);
for i=1:numberOfDirectories
    for j=1:numberOfImpacts
        styleString=[plotStylesMarkers{i},colors{i}];
        plot(absorbedEnergyDistributionRatiosLengthStretchedAxisPID(:,j,i),totalEnergyDistributionNormalizedToMaximumPID(:,j,i),styleString);
    end
end
ylabel('ratio of total energy');
xlabel('Normalized To Zone Length');
hold off;
saveas(tempPlot,[outputPath,'EnergyDistributionRatioMAXAxisLengthStretch','.fig']);
saveas(tempPlot,[outputPath,'EnergyDistributionRatioMAXAxisLengthStretch','.png']);

fileId=fopen([outputPath,'normalizedEnergyTransferDist.csv'],'w');
fprintf(fileId,'%s,', ['Normalized Dist. From Centre ',titleString]);
fprintf(fileId,'%s,', ['Normalized NRG Ratio ',titleString]);
fprintf(fileId, '
');
for directoryNumber=1:numberOfDirectories
    for impactNumber=1:numberOfImpacts
        for pathNumber=0:numberOfPaths
            fprintf(fileId,'%s,',num2str(absorbedEnergyDistributionRatiosLengthStretchedAxisPID(pathNumber+1,impactNumber,directoryNumber)));
            fprintf(fileId,'%s,',num2str(totalEnergyDistributionRatiosPID(pathNumber+1,impactNumber,directoryNumber)));
        end
    end
    fprintf(fileId,'\n');
end
fclose(fileId);

fileId=fopen([outputPath,'normalizedWrtMaxAndZoneLengthEnergyTransferDist.csv'],'w');
fprintf(fileId,'%s,', ['Normalized Dist. From Centre ',titleString]);
fprintf(fileId,'%s,', ['Normalized With Respect to Maximum',titleString]);
fprintf(fileId, '\n');
for directoryNumber=1:numberOfDirectories
    for impactNumber=1:numberOfImpacts
        for pathNumber=0:numberOfPaths
            fprintf(fileId,'%s,',num2str(absorbedEnergyDistributionRatiosLengthStretchedAxisPID(pathNumber+1,impactNumber,directoryNumber)));
            fprintf(fileId,'%s,',num2str(totalEnergyDistributionNormalizedToMaximumPID(pathNumber+1,impactNumber,directoryNumber)));
        end
    end
    fprintf(fileId,'\n');
end
fclose(fileId);
fileId=fopen([outputPath,'plasticDepth.csv'],"w");
fprintf(fileId,'%s,','[Impact Velocity (m/s)]
');
fprintf(fileId,'%s,','[Impact Number]
');
fprintf(fileId,'%s,','[Cumulative Absorbed Energy]
');
fprintf(fileId,'%s,','[Plastic Depth (m) ',titleString
');
fprintf(fileId, 
');
end
fclose(fileId);
disp('...Finished');

7.2.8. 2D process model code

Mathworks Inc. Matlab v7 Code

function HybdSquare(minutesOfFinishing, condition91Sa, strip617umSwitch, bilinearMaterialModelSwitch, frictionSwitch,
elecMagImpactVelocity)
clc;
%elecMagImpactVelocity=0;
%clear all;
% profile on;

versionNumber='4.36'; %do not put the letter v in front.
overRideArguments=true;
if overRideArguments==true
minutesOfFinishing=120;
condition91Sa=true;
strip617umSwitch=true;
bilinearMaterialModelSwitch=false;
frictionSwitch=false;
elecMagImpactVelocity=0;
end
dbugMode=false;
%% Plot of Energy absorbed by the target surface .................................
graphGridEnergy=true;
saveFilePath='C:\_DOCS\PhD\THESIS\_Model\_Code\VF Process Model\Output\'

contactAreaScalingFactor=1; %does not affect energy absorbed
energyTransferScalingFactor=1;
energyIncidentScalingFactor=1;
velocityHiPass=0;
automaticallySetImpactVelocityScalingFactorToOptimalForEachCon=false;
automaticallySetImpactVelocityScalingFactorToOptimalForEachCon=false;

gridSpanLength=0.0635;
gridSpanLength=3/1000;
numberOfGridElementsLength=round(500); %odd only 100

graphStressDistribution=true;
matchOutputTimesWithExperiments=true;
% forceSequentialPickingOfEachVelocity=true;
%%% ANSYS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

useAnsysToCalculateArcHeight=true;
lenToMidRatioOfAnsysPlate=5;
useOnlyLengthStressesInAnsysFEA=true;

useTransformation=true;
useActualStressDistributionInsteadOfCenterlineStresses=false;
useEntireActualStressDistributionRadiusNotContactRadius=false;
usePartialActualStressDistributionRadiusScaleByContactRadius=false;
usePartOfActualStressDistributionRadiusScaleByContactRadiusScaleFactor=2;
actualStressDistributionExtentRadiusInMeters=1/1000;

numberOfGridElementsWidth=round(numberOfGridElementsLength/lenToMidRatioOfAnsysPlate);

ansysCodeAndOutputPath='C:\_D\IStress\'
workingDirectory = cd;
lenNumOfAnsysElements=numberOfGridElementsLength;
widNumOfAnsysElements=round(numberOfGridElementsLength/lenToMidRatioOfAnsysPlate);
gridSpanWidth=gridSpanLength/lenToMidRatioOfAnsysPlate;
if widNumOfAnsysElements > numberOfGridElementsWidth
    widNumOfAnsysElements = numberOfGridElementsWidth;
end
lenNumOfAnsysElements = round(widNumOfAnsysElements / lenToWidRatioOfAnsysPlate);
disp('Warning: Width is longer than Length. This goes against the convention used');
end
numberOfAnsysElem = lenNumOfAnsysElements * widNumOfAnsysElements;
midNode = 0;
ansysBgnYIndex = round((numberOfGridElementsWidth - widNumOfAnsysElements) / 2) + 1;
ansysEndYIndex = widNumOfAnsysElements - ansysBgnYIndex + 1;
disp('Warning: Width is longer than Length. This goes against the convention used');
return;
end
if useAnsysToCalculateArcHeight == false
    graphStressDistribution = false;
end

if automaticallySetImpactVelocityScalingFactorToOptimalForEachCon == true
    if condition91Sa == true && strip617umSwitch == true
        impactVelocityScalingFactor = 0.60;
    elseif condition91Sa == false && strip617umSwitch == true
        impactVelocityScalingFactor = 0.23;
    elseif condition91Sa == true && strip617umSwitch == false
        impactVelocityScalingFactor = 0.13;
    else % (condition91Sa == false && strip617umSwitch == false)
        impactVelocityScalingFactor = 0.15;
    end
end

if automaticallySetIncidentEnergyScalingFactorToOptimalForEachCon == true
    if condition91Sa == true && strip617umSwitch == true
        energyIncidentScalingFactor = 0.42^2;
    elseif condition91Sa == false && strip617umSwitch == true
        energyIncidentScalingFactor = 0.25^2;
    elseif condition91Sa == true && strip617umSwitch == false
        energyIncidentScalingFactor = 0.15^2;
    else % (condition91Sa == false && strip617umSwitch == false)
        energyIncidentScalingFactor = 0.17^2;
    end
end

if useActualStressDistributionInsteadOfCenterlineStresses == true
    if automaticallySetImpactVelocityScalingFactorToOptimalForEachCon == true
        if condition91Sa == true && strip617umSwitch == true
            impactVelocityScalingFactor = 0.75;
        elseif condition91Sa == false && strip617umSwitch == true
            impactVelocityScalingFactor = 0.45; % 0.45 good fit
        elseif condition91Sa == true && strip617umSwitch == false
            impactVelocityScalingFactor = 0.30;
        else % (condition91Sa == false && strip617umSwitch == false)
            impactVelocityScalingFactor = 0.30;
        end
    end
end

if automaticallySetIncidentEnergyScalingFactorToOptimalForEachCon == true
    if condition91Sa == true && strip617umSwitch == true
        energyIncidentScalingFactor = 0.63^2;
    elseif condition91Sa == false && strip617umSwitch == true
        energyIncidentScalingFactor = 0.375^2;
    elseif condition91Sa == true && strip617umSwitch == false
        energyIncidentScalingFactor = 0.225^2;
    else % (condition91Sa == false && strip617umSwitch == false)
        energyIncidentScalingFactor = 0.26^2;
    end
end

matchOutputTimesWithExperimentsCounter = 1; % set to 1
matchOutputAtTimeCoincidingWithExperiments(1) = 0.001;
matchOutputAtTimeCoincidingWithExperiments(2) = 0.003;
matchOutputAtTimeCoincidingWithExperiments(3) = 0.005;
matchOutputAtTimeCoincidingWithExperiments(4) = 0.008;
matchOutputAtTimeCoincidingWithExperiments(5) = 0.01;
matchOutputAtTimeCoincidingWithExperiments(6) = 0.02;
matchOutputAtTimeCoincidingWithExperiments(7) = 0.9;
matchOutputAtTimeCoincidingWithExperiments(8) = 1;
matchOutputAtTimeCoincidingWithExperiments(9) = 2;
matchOutputAtTimeCoincidingWithExperiments(10) = 5;
matchOutputAtTimeCoincidingWithExperiments(11) = 15;
matchOutputAtTimeCoincidingWithExperiments(12) = 30;
matchOutputAtTimeCoincidingWithExperiments(13) = 60;
matchOutputAtTimeCoincidingWithExperiments(14) = 90;
matchOutputAtTimeCoincidingWithExperiments(15) = 120;
numberOfOutputPoints = length(matchOutputAtTimeCoincidingWithExperiments(1,:));

if elecMagImpactVelocity == 0
matchOutputTimesWithExperimentsCounter=1; % set to 1
matchOutputAtTimesCoincidingWithExperiments(1)=0.1;
matchOutputAtTimesCoincidingWithExperiments(2)=2;
matchOutputAtTimesCoincidingWithExperiments(3)=5;
matchOutputAtTimesCoincidingWithExperiments(4)=10;
matchOutputAtTimesCoincidingWithExperiments(5)=20;
matchOutputAtTimesCoincidingWithExperiments(6)=50;
matchOutputAtTimesCoincidingWithExperiments(7)=90;
matchOutputAtTimesCoincidingWithExperiments(8)=150;
matchOutputAtTimesCoincidingWithExperiments(9)=300;
matchOutputAtTimesCoincidingWithExperiments(10)=510;
matchOutputAtTimesCoincidingWithExperiments(11)=720;

numberOfOutputPoints=length(matchOutputAtTimesCoincidingWithExperiments(1,:));
end
if minutesOfFinishing > matchOutputAtTimesCoincidingWithExperiments(numberOfOutputPoints)
    minutesOfFinishing = matchOutputAtTimesCoincidingWithExperiments(numberOfOutputPoints);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
useAbsNrgPerBall=0;
useAbsNrgPerContactZoneArea=1;
useAbsNrgPerContactZoneVolume=2;
useAbsNrgPerContactZoneBallWithArcHeightStorage=3;
useAbsNrgPer=useAbsNrgPerContactZoneArea;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nrgDistUniform=0;
nrgDistLinear=1;
nrgDistReal=2;
nrgDistOption=nrgDistUniform;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
contactRadiusHills=0;
contactRadiusFEA=1;
contactRadiusFixed=2;
contactRadiusOption=contactRadiusFEA;
fixedContactZoneRadius=0.001;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
useElectroMagnetCalibrationForArcheightDetermination=false;
if useElectroMagnetCalibrationForArcheightDetermination==true
disp('Using Electromagnet Calibration');
    contactRadiusOption=contactRadiusFEA;
    useAbsNrgPer=useAbsNrgPerContactZoneArea;
end
% useLinearDecreaseInAbsorbedEnergyAsDistanceFromCenterLine=false;
elecMagConsiderBounces=false;
elecMagConsiderBounces=false;
elecMagConsiderBounces=false;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elecMagImpactVelocity=0;
end
elecMagConsiderBounces=false;
elecMagMaximumNumberOfBounces=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if overRideArguments==true
    elecMagImpactVelocity=0;
end
elecMagConsiderBounces=false;
elecMagMaximumNumberOfBounces=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elecMagImpactVelocity=0;
elecMagPeriod = 1/elecMagFreq;
accelerationDueToGravity=9.81;
elecMagprimaryBounce=false;
% forceSequentialPickingOfEachVelocity=false;
% simulateTransverseImpactLocationDistribution=true;
% transverseDirectionWidthEqualToContactZoneRadius=false;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if debugMode==true
disp('WARNING! DEBUG MODE');
end
condition112Sb=~condition91Sa;
if condition91Sa==true
    averageSensorCollisionFrequencyHertz=462; % nfdatD 200 lb Steel 1.75 frm HW Facing HW Flush button - 5 min.csv; Vel_Hist_Dat.csv
    contactDataFilename =`nfdatD 200 lb Steel 1.75 frm HW Facing HW Flush button - 5 min.csv';
    contactDataPath = 'C:\_DOCS\PhD\THESIS\SensorDat\June 13 2005 results\200 lbs Steel Media, 1.75 from HW, Facing HW, 5 min\';
else
    averageSensorCollisionFrequencyHertz=224; % nfdatD Steel 3.75 frm HW Facing HW Flush button - 5 min.csv; Vel_Hist_Dat.csv
    contactDataFilename =`nfdatD Steel 3.75 frm HW Facing HW Flush button - 5 min.csv';
    contactDataPath = 'C:\_DOCS\PhD\THESIS\SensorDat\June 13 2005 results\250 lbs Steel Media, 3.75 from HW, Facing HW, 5 min\';
end
disp('ERROR: No conditions were chosen');
exit(0);
if overRideArguments==true
    disp('Warning: Argument Override enabled.');
end

RUNS=1;
almenStripLengthMeters=3*(25.4/1)*(1/1000); %3"=0.0762m
almenStripWidthMeters=0.75*(25.4/1)*(1/1000);
% gridLength=0.0762*(2.5/3);
span=0.0635;
almenAreaMetersSquared=almenStripLengthMeters*almenStripWidthMeters;
% if centerElementIndex*2 ~= numberOfGridElementsLength
%    disp('Consider even max elem');
end

numberOfGridElements = numberOfGridElementsLength*numberOfGridElementsWidth;
gridSpanLength=sqrt(pi*(2/1000)^2); %same as sensor area
lengthOfGridElement=gridSpanLength/numberOfGridElementsLength;

sensorDiameter=0.004;
sensorArea=pi()*(sensorDiameter/2)^2;
collisionFlux=averageSensorCollisionFrequencyHertz/sensorArea;

gridPhysicalArea=gridSpanLength*gridSpanWidth;
gridCollisionFrequency = collisionFlux*gridPhysicalArea;
N=round(gridCollisionFrequency*60*minutesOfFinishing); %number of impacts

disp(['Number of impacts: ',num2str(N)]);

if elecMagImpactVelocity~=0
    collisionFlux = (elecMagNumOfBalls*elecMagFreq)/almenAreaMetersSquared;
end

disp(['Length of Grid Element: ', num2str(lengthOfGridElement*1000*1000),' micron']);
disp(['Length of Sensitive Area: ', num2str(gridSpanLength*1000),' mm']);
saturationEnergyForPlot=1.6e-3; %J
peakArcHeightEnergyForPlot=5.11e-5; %J
end
else
disp('Error: Energy tracking graph for this mode not supported yet');
graphGridEnergy=false;
end
saturationEnergyForPlot=saturationEnergyForPlot*0.5; %DEBUG
cMapNrg=zeros(numberOfColorsInMap,3);
if graphGridEnergy==true
%saturationEnergyForPlot=saturationEnergyForPlot/10;
joulesPerColor=saturationEnergyForPlot/numberOfColorsInMap;

% highlightMaximumArcHeightColor==true
numberOfColorsToPeakArcHeightEnergy=ceil(peakArcHeightEnergyForPlot/joulesPerColor);
for i=1:numberOfColorsToPeakArcHeightEnergy
    cMapNrg(i,1)=i/numberOfColorsToPeakArcHeightEnergy;
end
for i=numberOfColorsToPeakArcHeightEnergy+1:numberOfColorsInMap
    cMapNrg(i,1)=0;
    cMapNrg(i,2)=((i-1)/(numberOfColorsInMap-numberOfColorsToPeakArcHeightEnergy));
    cMapNrg(i,3)=1;
end
peakColor=[1,1,1];
peakColorHalfWidthInteger=1;
for j=1:-1:peakColorHalfWidthInteger
    cMapNrg(i,1)=peakColor;
end
else
for i=1:numberOfColorsInMap
    cMapNrg(i,1)=(i-1)/(numberOfColorsInMap);
    cMapNrg(i,2)=(i-1)/(numberOfColorsInMap);
    cMapNrg(i,3)=(i-1)/(numberOfColorsInMap);
end
colormap(cMapNrg);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Force Signal velocity distribution Data Retrieval
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Section of the VELOCITY GENERATOR CODE - START %%%%%%%%%%%%%%%%%%%%%%%%
if elecMagImpactVelocity==0
    realContactsData = csvread([contactDataPath,contactDataFilename], 1, 0);
    % enumerations start
    iStart=1;
    iEnd=2;
    iContactTime=3;
    iImpulse=4;
    iForceMaximum=5;
    iContactType=6; % 0=Not Classified, 1=impact, 2=quasi-static impactVelocity=?;
    samplePeriod=1/1000000;
    impactVelocityHistory=realContactsData(:,iImpactVelocity);
    actualTimeData=realContactsData(:,iStart).*samplePeriod;
    numberOfImpactVelocitiesInContactsData = length(impactVelocityHistory)-1;
    impactVelocityHistoryCounter=1;
    numberOfCounterResets=1;
    clear realContactsData;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Section of the VELOCITY GENERATOR CODE - END %%%%%%%%%%%%%%%%%%%%%%%%

%numberAnsysElem=numberOfGridElements+1;
averageEntireGridEnergyHistory = zeros(numberOfOutputPoints+1,1);
totalEntireGridEnergyHistory = zeros(numberOfOutputPoints+1,1);
coverageDescreteDomain = zeros(numberOfOutputPoints+1,1);
arcHeightHistoryByAnyNGC = zeros(numberOfOutputPoints+1,1);
arcHeightHistoryByDeformation = zeros(numberOfOutputPoints+1,1);
arcHeightHistoryByAnyMaxElem=zeros(numberOfOutputPoints+1,1);
arcHeightHistoryByAnyMidElem=zeros(numberOfOutputPoints+1,1);
arcHeightHistoryByAnyTest=zeros(numberOfOutputPoints+1,1);
% arcHeightHistoryMomentDistroForcedToCentre=zeros(numberOfOutputPoints+1,1);
timeAxis = zeros(numberOfOutputPoints+1,1);
if elecMagImpactVelocity==0
gridTimeAxis = zeros(numberOfOutputPoints+1,1);
sensorTimeAxis = zeros(numberOfOutputPoints+1,1);
end
impactNumberAxis = zeros(numberOfOutputPoints+1,1);
nodes= zeros(numberOfGridElementsLength+1,2); %this contains (x,y) coordinates of the strip.
Vfxy = coordinates go up in increments of span/numberOfGridElements
arcHeightGrid= zeros(numberOfGridElementsLength,1);
gridAbsorbedEnergy= zeros(numberOfGridElementsLength,numberOfGridElementsWidth);
psi= zeros(numberOfGridElementsLength,numberOfGridElementsWidth,1);
gridAbsNrgImage = zeros(numberOfGridElementsLength, numberOfGridElementsWidth);

if (useActualStressDistributionInsteadOfCenterlineStresses==true)
    stressDistribution=zeros(lenNumOfAnsysElements, widNumOfAnsysElements, 2);
    stressDistStXIndx=1;
    stressDistStYIndx=2;
end

if (useAbsNrgPer==useAbsNrgPerContactZoneBallWithArcHeightStorage)
    arcHeightRectGrid=zeros(numberOfGridElementsLength,numberOfGridElementsWidth);
end

disp(["Simulation Time: ", num2str(minutesOfFinishing)]);
if strip617umSwitch==true
    disp('617um Thickness');
    stripThicknessString='617um_';
    almenStripThickness=617/1e6;
else
    disp('779um Thickness');
    stripThicknessString='779um_';
    almenStripThickness=779/1e6;
end

if frictionSwitch==true
    disp('Friction ON');
    frictionString='F1_';
else
    disp('Friction OFF');
    frictionString='F0_';
end
if bilinearMaterialModelSwitch==true
    disp('Bilinear, Isotropic, Elastic Perfectly Plastic Material Model');
    materialModelString='Bile_';
else
    disp('Power Law Isotropic Material Model');
    materialModelString='PwL_';
end

if useAbsNrgPer==useAbsNrgPerBall
    disp('Absorbed Energy Tracking: BALL');
    useAbsNrgPerString='EB_';
else
    disp('Error: no such Absorbed Energy Tracking');
    return;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nrgDistOption==nrgDistLinear
    disp('Energy Distribution: LINEAR');
    nrgDistOptionString='DL_';
else
    disp('Error: no such Energy Distribution Option');
    return;
end

if contactRadiusOption==contactRadiusHills
disp('Contact Radius Option: HILLS');
contactRadiusOptionString = 'CH_';
elseif contactRadiusOption==contactRadiusFEA
disp('Contact Radius Option: FEA');
contactRadiusOptionString = 'CE_';
elseif contactRadiusOption==contactRadiusFixed
disp('Contact Radius Option: FIXED');
contactRadiusOptionString = 'CX_';
else
disp('Error: no such Contact Radius Option');
return;
end
if useActualStressDistributionInsteadOfCenterlineStresses==true
stressDistString='ST_';
disp('Using Actual Stress Distribution');
else
stressDistString='SC_';
disp('Using Centerline Stresses');
end
velocityHiPassString=['VF',num2str(velocityHiPass)];
verboseConditionsString=['contactAreaScalingFactorString,energyTransferScalingFactorString,
energyIncidentScalingFactorString,...
impactVelocityScalingFactorString,useAbsNrgPerString,nrgDistOptionString,...
contactRadiusOptionString,frictionString,stressDistString, velocityHiPassString];
fileNamePrefix=['PM_',versionNumber,'_', materialModelString, stripThicknessString, conditionString,
num2str(minuteOfFinishing), ' _',verboseConditionsString, '_',num2str(randomInteger)];
fileNamePrefix = strrep(fileNamePrefix, '"', '');
fileNamePrefix = strrep(fileNamePrefix, ' ', '_');
fileNamePrefix = strrep(fileNamePrefix, '/', '');
titleString = strrep(fileNamePrefix, '_', ' ');
% if debugMode==true
% a=a*0.1;
% end
if (debugMode~=true)&&(elementSizeWarnings==true)
if ((contactArea/(lengthOfGridElement^2)) < 1)
    insufficientResolutionWarningCount=insufficientResolutionWarningCount+1;
    disp('Warning! Element resolution was insufficient.\');
end
if (gridSpanLength/(2*a)) < 1
    insufficientSensitiveAreaSize=insufficientSensitiveAreaSize+1;
    disp('Warning! A contact area was larger than the sensitive area.\');
end
if (insufficientSensitiveAreaSize*100/impactCounter)>2
    disp('ERROR! Many contact areas were larger than the sensitive area. exiting...\');
    return;
end
if (insufficientResolutionWarningCount*100/impactCounter)>2
    disp('ERROR: Many impacts encountered insufficient element resolution. exiting...\');
    return;
end
end
aElem=ceil(a/lengthOfGridElement); %
%aElem=(a/lengthOfGridElement);
randomNumberStartTime = clock();

%%%%%%%%%%%%%%%%%%%%%%%% RANDOM NUMBER GENERATION %%%%%%%%%%%%%%%%%%%%%%%% 
randImpLocInElemCentreDatumX=0;
randImpLocInElemCentreDatumY=0;
impactCentreGridIndexX = 0;
impactCentreGridIndexY = 0;
randMaxNumLen=floor(centerElementIndexLength + aElem)-1;
randMaxNumWid=floor(centerElementIndexWidth + aElem)-1;
goodRandomNumbers=false;
condition1=false;
condition2=false;
condition3=false;
while goodRandomNumbers==false
    randImpLocInElemCentreDatumY = floor((2*rand()-1)*randMaxNumWid);
    randImpLocInElemCentreDatumX = floor((2*rand()-1)*randMaxNumLen);
    impactCentreGridIndexX = randImpLocInElemCentreDatumX + centerElementIndexLength;
    impactCentreGridIndexY = randImpLocInElemCentreDatumY + centerElementIndexWidth;
    if abs(randImpLocInElemCentreDatumX) < centerElementIndexLength + aElem
        condition1=true;
    end
    if abs(randImpLocInElemCentreDatumY) < centerElementIndexWidth + aElem
        condition2=true;
    end
    if (condition1==true)&&(condition2==true)
        for xIndex=-aElem:aElem
            for yIndex=-aElem:aElem
                gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                if (sqrt(xIndex^2+yIndex^2))<=aElem
                    condition3=true;
                end
            end
        end
        if (etime(clock(), randomNumberStartTime)>120)
            disp('random number generator stalled...\');
            return;
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%% COUNT CONTACTED ELEMENTS %%%%%%%%%%%%%%%%%%%%%%%% 
numberOfElementsInContactZone=0;
numberOfElementsInContactZoneAndInGrid=0;
sumOfGridEnergyInContactZone=0;
x=0;
for xIndex=-aElem:aElem
    for yIndex=-aElem:aElem
        gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
        gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
        numberOfElementsInContactZone=numberOfElementsInContactZone+1;
        numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
        x=0;
        for xIndex=-aElem:aElem
            for yIndex=-aElem:aElem
                if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                    gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                    gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                    numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                    numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                    x=0;
                    for xIndex=-aElem:aElem
                        for yIndex=-aElem:aElem
                            if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                x=0;
                                for xIndex=-aElem:aElem
                                    for yIndex=-aElem:aElem
                                        if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                            gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                            gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                            numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                            numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                            x=0;
                                            for xIndex=-aElem:aElem
                                                for yIndex=-aElem:aElem
                                                    if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                        gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                        gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                        numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                        numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                        x=0;
                                                        for xIndex=-aElem:aElem
                                                            for yIndex=-aElem:aElem
                                                                if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                    gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                    gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                    numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                    numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                    x=0;
                                                                    for xIndex=-aElem:aElem
                                                                        for yIndex=-aElem:aElem
                                                                            if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                                gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                                gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                                numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                                numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                                x=0;
                                                                                for xIndex=-aElem:aElem
                                                                                    for yIndex=-aElem:aElem
                                                                                        if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                                            gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                                            gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                                            numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                                            numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                                            x=0;
                                                                                            for xIndex=-aElem:aElem
                                                                                                for yIndex=-aElem:aElem
                                                                                                    if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                                                        gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                                                        gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                                                        numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                                                        numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                                                        x=0;
                                                                                                        for xIndex=-aElem:aElem
                                                                                                            for yIndex=-aElem:aElem
                                                                                                                if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                                                                    gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                                                                    gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                                                                    numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                                                                    numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                                                                    x=0;
                                                                                                                    for xIndex=-aElem:aElem
                                                                                                                        for yIndex=-aElem:aElem
                                                                                                                            if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                                                                                gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                                                                                gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                                                                                numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                                                                                numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                                                                                x=0;
                                                                                                                                for xIndex=-aElem:aElem
                                                                                                                                    for yIndex=-aElem:aElem
                                                                                                                                        if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                                                                                            gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                                                                                            gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                                                                                            numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                                                                                            numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                                                                                            x=0;
                                                                                                                                            for xIndex=-aElem:aElem
                                                                                                                                                for yIndex=-aElem:aElem
                                                                                                                                                    if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                                                                                                        gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                                                                                                        gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                                                                                                        numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                                                                                                        numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                                                                                                    x=0;
                                                                                                                                                    for xIndex=-aElem:aElem
                                                                                                                                                        for yIndex=-aElem:aElem
                                                                                                                                                            if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                                                                                                                gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                                                                                                                gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                                                                                                                numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                                                                                                                numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                                                                                                                x=0;
                                                                                                                                                                for xIndex=-aElem:aElem
                                                                                                                                                                    for yIndex=-aElem:aElem
                                                                                                                                                                        if (ceil(sqrt(xIndex^2+yIndex^2))<=aElem)
                                                                                                                                                                            gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
                                                                                                                                                                            gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
                                                                                                                                                                            numberOfElementsInContactZone=numberOfElementsInContactZone+1;
                                                                                                                                                                            numberOfElementsInContactZoneAndInGrid=numberOfElementsInContactZoneAndInGrid+1;
                                                                                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
if ((gridIndexX<=numberOfGridElementsLength)&&(gridIndexX>0))
if ((gridIndexY<=numberOfGridElementsWidth)&&(gridIndexY>0))
numberOfElementsInContactZoneAndInGrid+=numberOfElementsInContactZoneAndInGrid;
sumOfGridEnergyInContactZone = gridAbsorbedEnergy(gridIndexX,gridIndexY)+sumOfGridEnergyInContactZone;
end
end

%%%%%%%%%%%%%%%%%%%%%%%% DETERMINE ABSORBED ENERGY %%%%%%%%%%%%%%%%%%%%%%%%

if contactArea<0
disp('contactArea BUG');
end

coefficientOfRestitution=0;
if useAbsNrgPer==useAbsNrgPerContactZoneArea
averageGridEnergyPerAreaInContactZone = sumOfGridEnergyInContactZone/numberOfElementsInContactZoneAndInGrid;
[newCumulativeEnergy, coefficientOfRestitution] = coefficientOfRestGivenAbsAndVLKTB11(strip67umSwitch,
bilinearMaterialModelSwitch,frictionSwitch ,useAbsNrgPer, averageGridEnergyPerAreaInContactZone, impactVelocity);
elseif useAbsNrgPer==useAbsNrgPerBall
averageGridEnergyInContactZone = sumOfGridEnergyInContactZone/numberOfElementsInContactZoneAndInGrid;
[newCumulativeEnergy, coefficientOfRestitution] = coefficientOfRestGivenAbsAndVLKTB11(strip67umSwitch,
bilinearMaterialModelSwitch,frictionSwitch ,useAbsNrgPer, averageGridEnergyInContactZone, impactVelocity);
elseif useAbsNrgPer==useAbsNrgPerContactZoneBallWithArcHeightStorage
averageGridEnergyPerElementInContactZone = sumOfGridEnergyInContactZone/numberOfElementsInContactZoneAndInGrid;
previouslyAbsorbedEnergyInContactLocation=numberOfElementsInContactZone*averageGridEnergyPerElementInContactZone;
[newCumulativeEnergy, coefficientOfRestitution] = coefficientOfRestGivenAbsAndVLKTB11(strip67umSwitch,
bilinearMaterialModelSwitch,frictionSwitch ,useAbsNrgPer, previouslyAbsorbedEnergyInContactLocation, impactVelocity);
else
disp('Invalid Option: useAbsNrgPer');
return;
end

%averageGridEnergyInContactZone = sumOfGridEnergyInContactZone/numberOfElementsInContactZoneAndInGrid;

% averageGridEnergyPerUnitAreaInContactZone=sumOfGridEnergyInContactZone/contactArea;
% option=useAbsNrgPer;
% if useAbsNrgPer==
% option=

%[newCumulativeEnergy, coefficientOfRestitution] = coefficientOfRestGivenAbsAndVLKTB11(strip67umSwitch,
bilinearMaterialModelSwitch,frictionSwitch ,useAbsNrgPer, averageGridEnergyInContactZone, impactVelocity);
% [newCumulativeEnergyBall, coefficientOfRestitutionBall] = coefficientOfRestGivenAbsAndVLKTB11(strip67umSwitch, bilinearMaterialModelSwitch,frictionSwitch ,useAbsNrgPerBall, averageGridEnergyInContactZone, impactVelocity);
% vReboundBall=coefficientOfRestitutionBall*impactVelocity;
% absorbedEnergy=0.5*m*abs(vRebound^2)*energyTransferScalingFactor;

%%%%%%%%%%%%%%%%%%%%%%%% UPDATE ELEMENT ENERGY %%%%%%%%%%%%%%%%%%%%%%%%

absorbedEnergy=newCumulativeEnergy-averageGridEnergyInContactZone;

k=0;
if k<0;
    aFromElem=aElem*lengthOfGridElement;
    StAbsorbedEnergy=0;
    for xIndex=-aElem:aElem
        for yIndex=-aElem:aElem
            gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
            gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
            if ((gridIndexX<=numberOfGridElementsLength)&&(gridIndexX>0))
                if ((gridIndexY<=numberOfGridElementsWidth)&&(gridIndexY>0))
                    distanceOfElementFromCenterElementInElementLengthUnits = ceil(sqrt(gridIndexX^2+gridIndexY^2));
                    if cell(distanceOfElementFromCenterElementInElementLengthUnits)<aElem
                        radialDistanceMeters = distanceOfElementFromCenterElementInElementLengthUnits*lengthOfGridElement;
                        if 0 == distanceOfElementFromCenterElementInElementLengthUnits
                            disp('sdffdsf');
                        end
                        if useAbsNrgPer==useAbsNrgPerContactZoneArea
                            energyAtRadius = nrgDistributionLinear01(radialDistanceMeters, aFromElem,
                            absorbedEnergy);
                        end
                    end
                end
            end
        end
    end
    for xIndex=-aElem:aElem
        for yIndex=-aElem:aElem
            gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
            gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
            if ((gridIndexX<=numberOfGridElementsLength)&&(gridIndexX>0))
                if ((gridIndexY<=numberOfGridElementsWidth)&&(gridIndexY>0))
                    distanceOfElementFromCenterElementInElementLengthUnits = ceil(sqrt(gridIndexX^2+gridIndexY^2));
                    if cell(distanceOfElementFromCenterElementInElementLengthUnits)<aElem
                        radialDistanceMeters = distanceOfElementFromCenterElementInElementLengthUnits*lengthOfGridElement;
                        if 0 == distanceOfElementFromCenterElementInElementLengthUnits
                            disp('sdffdsf');
                        end
                        if useAbsNrgPer==useAbsNrgPerContactZoneArea
                            energyAtRadius = nrgDistributionLinear01(radialDistanceMeters, aFromElem,
                            absorbedEnergy);
                        end
                    end
                end
            end
        end
    end
end

%averageGridEnergyPerUnitAreaInContactZone=sumOfGridEnergyInContactZone/contactArea;
% option=useAbsNrgPer;
% if useAbsNrgPer==
% option=

%[newCumulativeEnergy, coefficientOfRestitution] = coefficientOfRestGivenAbsAndVLKTB11(strip67umSwitch,
bilinearMaterialModelSwitch,frictionSwitch ,useAbsNrgPer, averageGridEnergyInContactZone, impactVelocity);
% [newCumulativeEnergyBall, coefficientOfRestitutionBall] = coefficientOfRestGivenAbsAndVLKTB11(strip67umSwitch, bilinearMaterialModelSwitch,frictionSwitch ,useAbsNrgPerBall, averageGridEnergyInContactZone, impactVelocity);
% vReboundBall=coefficientOfRestitutionBall*impactVelocity;
% absorbedEnergy=0.5*m*abs(vRebound^2)*energyTransferScalingFactor;

%%%%%%%%%%%%%%%%%%%%%%%% UPDATE ELEMENT ENERGY %%%%%%%%%%%%%%%%%%%%%%%%

absorbedEnergy=newCumulativeEnergy-averageGridEnergyInContactZone;

k=0;
if k<0;
    aFromElem=aElem*lengthOfGridElement;
    StAbsorbedEnergy=0;
    for xIndex=-aElem:aElem
        for yIndex=-aElem:aElem
            gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
            gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
            if ((gridIndexX<=numberOfGridElementsLength)&&(gridIndexX>0))
                if ((gridIndexY<=numberOfGridElementsWidth)&&(gridIndexY>0))
                    distanceOfElementFromCenterElementInElementLengthUnits = ceil(sqrt(gridIndexX^2+gridIndexY^2));
                    if cell(distanceOfElementFromCenterElementInElementLengthUnits)<aElem
                        radialDistanceMeters = distanceOfElementFromCenterElementInElementLengthUnits*lengthOfGridElement;
                        if 0 == distanceOfElementFromCenterElementInElementLengthUnits
                            disp('sdffdsf');
                        end
                        if useAbsNrgPer==useAbsNrgPerContactZoneArea
                            energyAtRadius = nrgDistributionLinear01(radialDistanceMeters, aFromElem,
                            absorbedEnergy);
                        end
                    end
                end
            end
        end
    end
    for xIndex=-aElem:aElem
        for yIndex=-aElem:aElem
            gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
            gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
            if ((gridIndexX<=numberOfGridElementsLength)&&(gridIndexX>0))
                if ((gridIndexY<=numberOfGridElementsWidth)&&(gridIndexY>0))
                    distanceOfElementFromCenterElementInElementLengthUnits = ceil(sqrt(gridIndexX^2+gridIndexY^2));
                    if cell(distanceOfElementFromCenterElementInElementLengthUnits)<aElem
                        radialDistanceMeters = distanceOfElementFromCenterElementInElementLengthUnits*lengthOfGridElement;
                        if 0 == distanceOfElementFromCenterElementInElementLengthUnits
                            disp('sdffdsf');
                        end
                        if useAbsNrgPer==useAbsNrgPerContactZoneArea
                            energyAtRadius = nrgDistributionLinear01(radialDistanceMeters, aFromElem,
                            absorbedEnergy);
                        end
                    end
                end
            end
        end
    end
end

%averageGridEnergyPerUnitAreaInContactZone=sumOfGridEnergyInContactZone/contactArea;
% option=useAbsNrgPer;
% if useAbsNrgPer==
% option=

%[newCumulativeEnergy, coefficientOfRestitution] = coefficientOfRestGivenAbsAndVLKTB11(strip67umSwitch,
bilinearMaterialModelSwitch,frictionSwitch ,useAbsNrgPer, averageGridEnergyInContactZone, impactVelocity);
% [newCumulativeEnergyBall, coefficientOfRestitutionBall] = coefficientOfRestGivenAbsAndVLKTB11(strip67umSwitch, bilinearMaterialModelSwitch,frictionSwitch ,useAbsNrgPerBall, averageGridEnergyInContactZone, impactVelocity);
% vReboundBall=coefficientOfRestitutionBall*impactVelocity;
% absorbedEnergy=0.5*m*abs(vRebound^2)*energyTransferScalingFactor;
gridAbsorbedEnergy(gridIndexX, gridIndexY) = energyAtRadius + gridAbsorbedEnergy(gridIndexX, gridIndexY);

elseif nrgDistOption==nrgDistReal
energyAtRadius = nrgDistribution01(radiolDistanceMeters, aFromAElem, absorbedEnergy, almenStripThickness);
gridAbsorbedEnergy(gridIndexX, gridIndexY) = energyAtRadius + gridAbsorbedEnergy(gridIndexX, gridIndexY);

elseif nrgDistOption==nrgDistUniform
energyPerUnitArea=absorbedEnergy/contactArea;
energyPerElementArea=(absorbedEnergy/numberOfElementsInContactZone)/elementArea;

gridAbsorbedEnergy(gridIndexX, gridIndexY)=gridAbsorbedEnergy(gridIndexX, gridIndexY) + absorbedEnergy/contactArea; %the input Data would have to be modified to make this correct.
% else
% disp('Error: no such Energy Distribution Option');
% return;
% end
% else
% disp(num2str(gridAbsorbedEnergy(gridIndexX,gridIndexY)));
% end
% E=energyAtRadius+Et;
% EtAbsorbedEnergy=EtAbsorbedEnergy+absorbedEnergy/contactArea;
}

elseif useAbsNrgPer==useAbsNrgPerBall
if nrgDistOption==nrgDistLinear
energyTransferToElement = energyAtRadius*elementArea;
gridAbsorbedEnergy(gridIndexX,gridIndexY)= energyTransferToElement + gridAbsorbedEnergy(gridIndexX,gridIndexY);

tgridAbsorbedEnergy(gridIndexX,gridIndexY)=elementArea*energyAtRadius*gridAbsorbedEnergy(gridIndexX,gridIndexY);
tgridAbsorbedEnergy(gridIndexX,gridIndexY)=energyAtRadius*pi*(aElem*lengthOfGridElement)^2+gridAbsorbedEnergy(gridIndexX,gridIndexY);
%check:
elseif nrgDistOption==nrgDistReal
energyAtRadius = nrgDistribution01(radiolDistanceMeters, aFromAElem, numberOfElementsInContactZone*absorbedEnergy);
energyTransferToElement = energyAtRadius*elementArea;
gridAbsorbedEnergy(gridIndexX,gridIndexY)= energyTransferToElement + gridAbsorbedEnergy(gridIndexX,gridIndexY);

gridAbsorbedEnergy(gridIndexX,gridIndexY)=gridAbsorbedEnergy(gridIndexX,gridIndexY) + absorbedEnergy; % /numberOfElementsInContactZone; %the input Data would have to be modified to make this correct.
% else
% disp('Error: no such Energy Distribution Option');
% return;
% end
% else
% disp(num2str(gridAbsorbedEnergy(gridIndexX,gridIndexY)));
% end
% if
% E=energyTransferToElement+Et;
% E=absorbedEnergy+absorbedEnergy; % the input Data would have to be modified to make this correct.
% else
% disp('Error: no such Energy Distribution Option');
% return;
% end

elseif useAbsNrgPer==useAbsNrgPerContactZoneBallWithArcHeightStorage
for xIndex=-aElem:aElem
for yIndex=-aElem:aElem
gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
if (gridIndexX<numberOfGridElementsLength+1) && (gridIndexX>=1) && (gridIndexY<numberOfGridElementsWidth+1) && (gridIndexY>=1)
distanceOfElementFromCenterElementInElementLengthUnits=ceil(sqrt(xIndex^2+yIndex^2));
if (distanceOfElementFromCenterElementInElementLengthUnits <= aElem)
energyStateOfContactZoneAfterImpact=previouslyAbsorbedEnergyInContactLocation+absorbedEnergy;

temp=arcHeightFromAbseLKTB11( strip617umSwitch, bilinearMaterialModelSwitch, frictionSwitch,
useAbsNrgPerBall, energyStateOfContactZoneAfterImpact);
arcHeightRectGrid(gridIndexX,gridIndexY)=temp;
arcHeightRectGrid(gridIndexX,gridIndexY)=arcHeightRectGrid(gridIndexX,gridIndexY)*1e-6;
end
end
end
end

% Update Actual Stress Distribution - Begin % % % % Update Actual Stress Distribution - End % %
%image(arcHeightRectGrid);

if (useActualStressDistributionInsteadOfCenterlineStresses==true)
    maxStressRadiusInElementLengths =...
        usePartOfActualStressDistributionScaledByContactRadiusScalingFactor*aElem;
    maxStressRadiusInElementLengths =...
        floor(actualStressDistributionExtentRadiusMeters/lengthOfGridElement);
end

if (useEntireActualStressDistributionRadiusNotContactRadius==true)&&
    (usePartialActualStressDistributionRadiusScaledByContactRadius==false)
    maxStressRadiusInElementLengths = aElem;
end

for xIndex=-maxStressRadiusInElementLengths:maxStressRadiusInElementLengths
    for yIndex=-maxStressRadiusInElementLengths:maxStressRadiusInElementLengths
        gridIndexX = xIndex + randImpLocInElemCentreDatumX + round(numberOfGridElementsLength/2);
        gridIndexY = yIndex + randImpLocInElemCentreDatumY + round(numberOfGridElementsWidth/2);
        if ((gridIndexX<=numberOfGridElementsLength)&&(gridIndexX>0))
            if ((gridIndexY<=numberOfGridElementsWidth)&&(gridIndexY>0))
                elementDistanceFromCenterComponentX=gridIndexX-impactCentreGridIndexX;
                elementDistanceFromCenterComponentY=gridIndexY-impactCentreGridIndexY;
                elementRadialDistanceInElementLengths=sqrt(elementDistanceFromCenterComponentX^2+elementDistanceFromCenterComponentY^2);
                if elementRadialDistanceInElementLengths <= maxStressRadiusInElementLengths
                    radialDistanceMeters=elementRadialDistanceInElementLengths*lengthOfGridElement;
                    elementTheta=atan(elementDistanceFromCenterComponentY/elementDistanceFromCenterComponentX);  %theta [-90,90] degrees
                    if elementDistanceFromCenterComponentX == 0
                        if elementDistanceFromCenterComponentY > 0
                            elementTheta = pi/2;
                        elseif elementDistanceFromCenterComponentY < 0
                            elementTheta = -pi/2;
                        end
                    elseif elementDistanceFromCenterComponentX > 0
                        if elementDistanceFromCenterComponentY == 0
                            elementTheta = 0;
                        elseif elementDistanceFromCenterComponentY > 0
                            elementTheta = atan(elementDistanceFromCenterComponentY/elementDistanceFromCenterComponentX);  %theta [0,360] degrees
                        elseif elementDistanceFromCenterComponentY < 0
                            elementTheta = pi+atan(elementDistanceFromCenterComponentY/elementDistanceFromCenterComponentX);  %theta [-360,0] degrees
                        end
                    end
                else
                    radialDistanceMeters=elementRadialDistanceInElementLengths*lengthOfGridElement;
                    elementTheta=atan2(elementDistanceFromCenterComponentY,elementDistanceFromCenterComponentX);  %theta [0,360] degrees
                end
            end
        end
    end
end

if useTransformation==true
    MxInN=MrOut*abs(sin(elementTheta))+MthetaOut*abs(cos(elementTheta));
    MyInN=MrOut*abs(cos(elementTheta))+MthetaOut*abs(sin(elementTheta));
    StX=(equivalentLinearStressDistribFromMomentPerUnitLength(MxInN, almenStripThickness));
    if useOnlyLengthStressesInAnsysFEA==false
        if elementDistanceFromCenterComponentX <= 0
            if elementDistanceFromCenterComponentY == 0
                elementTheta = 0;
            elseif elementDistanceFromCenterComponentY > 0
                elementTheta = pi/2;
            elseif elementDistanceFromCenterComponentY < 0
                elementTheta = -pi/2;
            end
        elseif elementDistanceFromCenterComponentY > 0
            if elementDistanceFromCenterComponentX == 0
                elementTheta = pi/2;
            elseif elementDistanceFromCenterComponentX > 0
                elementTheta = pi-atan(elementDistanceFromCenterComponentX/elementDistanceFromCenterComponentY);  %theta [-90,0] degrees
            elseif elementDistanceFromCenterComponentX < 0
                elementTheta = -pi+atan(elementDistanceFromCenterComponentX/elementDistanceFromCenterComponentY);  %theta [0,90] degrees
            end
        end
    end
else
    elementTheta=atan(elementDistanceFromCenterComponentY/elementDistanceFromCenterComponentX);  %theta [-90,90] degrees
end

MrOut, MthetaOut = momentsPerMeterFromAbsAbsFunctionOfRadialDistanceLKTB004(strip67umSwitch, bilinearMaterialModelSwitch, frictionSwitch, useAbsNrgPer , averageGridEnergyPerAreaInContactZone, radialDistanceMeters);
%         image(stressDistribution(:,:,1));
%         quiver(stressDistribution(:,:,1),stressDistribution(:,:,2));

%%%%%%%%%%% %%% % Update Actual Stress Distribution - End % %%% %%%%%%%%%%%%

if numberOfElementsInContactZoneAndInGrid==0
    disp('bug');
end
%         disp(['Actual Absorbed Energy: ', num2str(EtAbsorbedEnergy)]);
%         disp(['Dist Absorbed Energy: ', num2str(Et)]);
%         disp('    ');

%%%%%%%%%%%%%%%%%%%%%%% OUTPUT DATA - START %%%%%%%%%%%%%%%%%%%%%%%%

currentTime=(impactCounter/gridCollisionFrequency)/60;
produceOutput=false;
produceOutputEnd=false;
if matchOutputTimesWithExperiments==true
    lenMatchedOutput=length(matchOutputAtTimesCoincidingWithExperiments);
    if matchOutputTimesWithExperimentsCounter==lenMatchedOutput+1
        if produceOutputEnd==false
            produceOutput=true;
            produceOutputEnd=true;
        else
            produceOutput=false;
        end
    else
        if currentTime >= matchOutputAtTimesCoincidingWithExperiments(matchOutputTimesWithExperimentsCounter);
            matchOutputTimesWithExperimentsCounter=matchOutputTimesWithExperimentsCounter+1;
            produceOutput=true;
        end
    end
end
else
    nextTime = (simulationTimeMinutesBetweenOutputEventsIfNotMatchingExp*calculationCounter +...
                 simulationTimeGrowthMinutesBetweenOutputEventsIfNotMatchingExp^(calculationCounter-1));
    produceOutput=(currentTime>=nextTime)||(currentTime==minutesOfFinishing);
end
if produceOutput==true
    currentTime=num2str(currentTime,'%-5.2f');
    elementsUncovered=0;
    sumOfEntireGridEnergy=0;
    sumOfArcHeightOfGrid=0;
    for gridIndexX=1:numberOfGridElementsLength
        for gridIndexY=1:numberOfGridElementsWidth
            if gridAbsorbedEnergy(gridIndexX,gridIndexY)==0
                elementsUncovered=elementsUncovered+1;
            end
        end
    end
    averageEntireGridEnergy=mean(mean(gridAbsorbedEnergy));
    totalEntireGridEnergy=sum(sum(gridAbsorbedEnergy));
    if (useAbsNrgPer==useAbsNrgPerContactZoneBallWithArcHeightStorage)
        archeightOfCentreNodeByAvgNRG=archHeightFromAbseLKTB11( strip617umSwitch, bilinearMaterialModelSwitch,
                                    frictionSwitch,useAbsNrgPerBall ,averageEntireGridEnergy);
    else
        if useElectroMagnetCalibrationForArcheightDetermination==true
            archeightOfCentreNodeByAvgNRG=getArcHeightforThinStripFromElectroCalibForAreaMode(averageEntireGridEnergy);
        else
            archeightOfCentreNodeByAvgNRG=archHeightFromAbseLKTB11( strip617umSwitch, bilinearMaterialModelSwitch,
                                    frictionSwitch,useAbsNrgPer ,averageEntireGridEnergy);
        end
        archeightOfCentreNodeByAvgNRG=archeightOfCentreNodeByAvgNRG*1e-6;
    end
    if (useAbsNrgPer==useAbsNrgPerContactZoneBallWithArcHeightStorage)
halfWidth=round(numberOfGridElementsWidth/2);
[archeightOfCentreNodeByDeformation]=
arcHeightAtCenterFromArcHeightOfElements01(arcHeightRectGrid(:,halfWidth), nodes, lengthOfGridElement);
else
    for gridIndexX=1:numberOfGridElementsLength
        if useElectroMagnetCalibrationForArcheightDetermination==true
            [arcHeightGrid(gridIndexX)]=getArcHeightforThinStripFromElectroCalibForAreaMode(gridAbsorbedEnergy(gridIndexX,centerElementIndexWidth));
        else
            [arcHeightGrid(gridIndexX)]=arcHeightFromAbsNrgPer(gridIndexX,centerElementIndexWidth);
        end
        arcHeightGrid(gridIndexX)=arcHeightGrid(gridIndexX)*1e-6;
    end
[archeightOfCentreNodeByDeformation]= arcHeightAtCenterFromArcHeightOfElements01(arcHeightGrid, nodes, lengthOfGridElement);
end
[archeightOfCentreNodeByDeformation]= arcHeightOverDifferentSpan(archeightOfCentreNodeByDeformation, gridSpanLength, span);

%     sign=0;
%     if archeightOfCentreNodeByDeformation==0
%     
%     else
%         if archeightOfCentreNodeByDeformation>0
%             sign=1;
%         
%         else
%             archeightOfCentreNodeByDeformation=abs(archeightOfCentreNodeByDeformation);
%             sign=-1;
%         end
%     end
%     radius=((gridSpanLength^2/4)+archeightOfCentreNodeByDeformation^2)/(2*archeightOfCentreNodeByDeformation);
%     %archeightOfCentreNodeByDeformation = radius +- sqrt(radius^2 - (1/4)*span^2);
%     archeightOfCentreNodeByDeformation = radius - sqrt(radius^2 - (1/4)*span^2);
%     archeightOfCentreNodeByDeformation = sign*archeightOfCentreNodeByDeformation;
% end

if useAnsysToCalculateArcHeight==true
    %
    lenNumOfElem=length(stressDistrib(1,:));
    %
    widNumOfElem=length(stressDistrib(:,1));

    if madeMeshBefore==false
        madeMeshBefore=true;
        if useActualStressDistributionInsteadOfCenterlineStresses==true
            midNode=isMesher007(ansysCodeAndOutputPath, strip617umSwitch, stressDistribution, lengthOfGridElement);
        else
            stressDistrib=zeros(lenNumOfAnsysElements,widNumOfAnsysElements);
            midNode=isMesher007(ansysCodeAndOutputPath, strip617umSwitch, stressDistrib, lengthOfGridElement);
        end
        disp(['Middle Node : ', num2str(midNode)]);
    end
    if useActualStressDistributionInsteadOfCenterlineStresses==true
        isMakeAnsysInitialStressFileUsingMxAndMy002(ansysCodeAndOutputPath, stressDistribution, useOnlyLengthStressesInAnsysFEA);
    else
        for j=1:lenNumOfAnsysElements
            for i=ansysBgnYIndex:widNumOfAnsysElements+ansysBgnYIndex-1
                if useAbsNrgPer==useAbsNrgPerContactZoneBallWithArcHeightStorage
                    archeightAnsys=arcHeightRectGrid(j,i);
                else
                    if useElectroMagnetCalibrationForArcheightDetermination==true
                        archeightAnsys=getArcHeightforThinStripFromElectroCalibForAreaMode(gridAbsorbedEnergy(j,i));
                    else
                        archeightAnsys=arcHeightFromAbsNrgPer(gridAbsorbedEnergy(j,i));
                    end
                    archeightAnsys=archeightAnsys*1e-6;
                end
                stressDistrib(j,i-ansysBgnYIndex+1)=equivalentLinearStressDistribFromArcHeight(archeightAnsys, 0.0635, almenStripThickness, 71.7E9);
            end
        end
        isMakeAnsysInitialStressFile004(ansysCodeAndOutputPath, stressDistrib, useOnlyLengthStressesInAnsysFEA);
    end
    cd(ansysCodeAndOutputPath);
    %currentTime=num2str(round(currentTime));
    if debugMode==false

184
!isRunfea.bat

if (isRunfea.bat)
    copyfile('isDat.txt', [saveFilePath, fileNamePrefix, '_', currentTime, '_isDat.txt'], 'f');
    copyfile('file000.png', [saveFilePath, fileNamePrefix, '_', currentTime, '_UZ.png'], 'f');
    copyfile('file001.png', [saveFilePath, fileNamePrefix, '_', currentTime, '_SX.png'], 'f');
    copyfile('file002.png', [saveFilePath, fileNamePrefix, '_', currentTime, '_SY.png'], 'f');
    copyfile('file003.png', [saveFilePath, fileNamePrefix, '_', currentTime, '_SXY.png'], 'f');
    copyfile('file004.png', [saveFilePath, fileNamePrefix, '_', currentTime, '_SZ.png'], 'f');
    copyfile('file005.png', [saveFilePath, fileNamePrefix, '_', currentTime, '_SYZ.png'], 'f');
    copyfile('file006.png', [saveFilePath, fileNamePrefix, '_', currentTime, '_SEQV.png'], 'f');
    copyfile('file007.png', [saveFilePath, fileNamePrefix, '_', currentTime, '_DIST.png'], 'f');

    cd(workingDirectory);
    [maxArcHeight, midNodeArcHeight] = isReadMaxAndMidNodeValue002(ansysCodeAndOutputPath, 'nodesUZ.lis', midNode);
    cd(ansysCodeAndOutputPath);

    % if debugMode == false
    %     !reset.bat
    % end

    [status, message, messageid] = copyfile('nodesUZ.lis', [saveFilePath, fileNamePrefix, '_', currentTime, '_nodesUZ.lis'], 'f');
    cd(workingDirectory);
    arcHeightHistoryByAnsysMidElem(calculationCounter + 1) = arcHeightOverDifferentSpan(midNodeArcHeight, gridSpanLength, span);
    arcHeightHistoryByAnsysMaxElem(calculationCounter + 1) = arcHeightOverDifferentSpan(maxArcHeight, gridSpanLength, span);
    arcHeightHistoryByAnsysTest(calculationCounter + 1) = (midNodeArcHeight / gridSpanLength) * span;
    arcHeightHistoryByAnsysTest(calculationCounter + 1) = almenArcHeightOverDifferentSpanTest(gridSpanLength, midNodeArcHeight);

    end

averageEntireGridEnergyHistory(calculationCounter + 1) = averageEntireGridEnergy;
totalEntireGridEnergyHistory(calculationCounter + 1) = totalEntireGridEnergy;
arcHeightHistoryByAvgNRG(calculationCounter + 1) = arcHeightOfCentreNodeByAvgNRG;
arcHeightHistoryByDeformation(calculationCounter + 1) = arcHeightOfCentreNodeByDeformation;
timeAxis(calculationCounter + 1) = (impactCounter / gridCollisionFrequency) / 60; % changed line
gridTimeAxis(calculationCounter + 1) = gridTime / 60;
sensorTimeAxis(calculationCounter + 1) = sensorTime / 60;
impactNumberAxis(calculationCounter + 1) = impactCounter;
coverageDescreteDomain(calculationCounter + 1) = (numberOfGridElements - elementsUncovered) / numberOfGridElements;

%% Graph incomplete arc height - BEGIN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
clf;
tempPlot = newplot;
hold on;
plot(timeAxis(1:length(arcHeightHistoryByAvgNRG)), arcHeightHistoryByAvgNRG, '*1000*1000', 'bo', 'markersize', 6);
plot(timeAxis(1:length(arcHeightHistoryByDeformation)), arcHeightHistoryByDeformation, '*1000*1000', 'em', 'markersize', 8);
if useAnsysToCalculateArcHeight == true
    plot(timeAxis(1:length(arcHeightHistoryByAnsysMidElem)), arcHeightHistoryByAnsysMidElem, '*1000*1000', 'b+', 'markersize', 10);
end
plot(timeAxis(1:length(arcHeightHistoryByAnsysMaxElem)), arcHeightHistoryByAnsysMaxElem, '*1000*1000', 'k+', 'markersize', 8);
plot(timeAxis(1:length(arcHeightHistoryByAnsysTest)), arcHeightHistoryByAnsysTest, '*1000*1000', 'kd', 'markersize', 4);
end

if graphExperimentalDataForComparision == true
    plotExp(strip617umSwitch, condition91Sa, elecMagImpactVelocity);
end

title(titleString);
xlabel('Time (min) from - Resolution: ', num2str(numberOfGridElements));
ylabel('Arc Height (microns)');
saveas(tempPlot, [saveFilePath, fileNamePrefix, '_ArcHeight_incomplete', '.fig']);
saveas(tempPlot, [saveFilePath, fileNamePrefix, '_ArcHeight_incomplete', '.png']);
tola reset;
tola tempPlot;
hold off;
%% Graph incomplete arc height - END %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if graphGridEnergy == true
    clf; end
cla reset;
clear tempPlot;
tempPlot = newplot;
hold off;

for ic = 1:numberOfGridElementsLength
for jc=1:numberOfGridElementsWidth
    elemVal=gridAbsorbedEnergy(ic,jc);
    if elemVal > 0
        gridAbsNrgImageShadow(ic,jc)=1;
    else
        gridAbsNrgImageShadow(ic,jc)=0;
    end
end
if elemVal>saturationEnergyForPlot
    gridAbsNrgImage(ic,jc)=numberOfColorsInMap;
else
    gridAbsNrgImage(ic,jc)=round((elemVal/saturationEnergyForPlot)*numberOfColorsInMap);
end
end
if (graphStressDistribution==true)&&(useActualStressDistributionInsteadOfCenterlineStresses==true)
    subplot(4,1,1);
    contourf(gridAbsorbedEnergy.');
end
if (graphStressDistribution==true)&&(useActualStressDistributionInsteadOfCenterlineStresses==false)
    subplot(3,1,1);
    imagesc(gridAbsNrgImage.');
    daspect([1 1 1]);
    axis tight;
    subplot(3,1,2);
    imagesc(gridAbsNrgImageShadow.');
    daspect([1 1 1]);
    axis tight;
    saveas(tempPlot,saveFilePath,fileNamePrefix,'_contourf_',num2str(calculationCounter,'%05d'),'.fig');
end
if (graphStressDistribution==false)
    imhandle=image(gridAbsNrgImage.');
    colorbar('EastOutside');
    % xlim([1 numberOfGridElementsLength]);
    % ylim([1 numberOfGridElementsWidth]);
    title(titleString);
    xlabel('Time (min) - Resolution: ', num2str(numberOfGridElements));
    ylabel(['Grid Energy Plot']);
    %ylabel(['Arc Height (microns)']);
    saveas(tempPlot,saveFilePath,fileNamePrefix,'_contourf_',num2str(calculationCounter,'%05d'),'_',currentTime,'.png');
    saveas(tempPlot,saveFilePath,fileNamePrefix,'_contourf_',num2str(calculationCounter,'%05d'),'_',currentTime,'.fig');
    cla reset;
    clear tempPlot;
    hold off;

if (graphStressDistribution==true)&&(useActualStressDistributionInsteadOfCenterlineStresses==true)
    subplot(4,1,2); imagesc(stressDistribution(:,:,1).');
    subplot(4,1,3); imagesc(stressDistribution(:,:,2).');
    subplot(4,1,4); quiver(stressDistribution(:,:,1).', stressDistribution(:,:,2).');
    subplot(4,1,2); contourf(stressDistribution(:,:,1).');
    subplot(4,1,3); contourf(stressDistribution(:,:,2).');
    subplot(4,1,4); quiver(stressDistribution(:,:,1).', stressDistribution(:,:,2).');
end
if (graphStressDistribution==true)&&(useActualStressDistributionInsteadOfCenterlineStresses==false)
    subplot(3,1,3); imagesc(stressDistrib.');
    subplot(3,1,2); imagesc(stressDistrib.');
    subplot(3,1,1); imagesc(stressDistrib.');
end

saveas(tempPlot,saveFilePath,fileNamePrefix,'_contourf_',num2str(calculationCounter,'%05d'),'_',currentTime,'.png');
saveas(tempPlot,saveFilePath,fileNamePrefix,'_contourf_',num2str(calculationCounter,'%05d'),'_',currentTime,'.fig');
clear all;
calculationCounter = calculationCounter+1;
% %PLOT DEFORMED SHAPE
% if (PlotDeformedShape==true)
% for k=1:numberOfGridElements
%     nodes(k,Y)=-1*(nodes(k,Y)-(slope*k+yint));
% end
% plot(nodes(:,X) ,nodes(:,Y));
% numberOfTimesGraphed = numberOfTimesGraphed+1;
% end
end  %end of graph and record data routine

%%%%%%%%%%%%%%%%%%% Progress and ETA Code - START %%%%%%%%%%%%%%%
currentTime = clock();
elapsedTime=0;
if (etime(currentTime, lastProgressUpdateTime)>2)
    lastProgressUpdateTime = clock();
    percentComplete=(100*impactCounter/N);
    elapsedTime = etime(currentTime, programStartTime);
    eta = ((100-percentComplete)*(elapsedTime/percentComplete))/(60*60);
    fprintf(['
', num2str(eta), ', ', num2str((elapsedTime/(60*60))+eta), ', Total Run Time in hours']);
end
%%%%%%%%%%%%%%%%%%% Progress and ETA Code - START %%%%%%%%%%%%%%%
end %end of simulation

arcHeightHistoryByAvgNRG = arcHeightHistoryByAvgNRG.*(1000*1000); %convert to micron
arcHeightHistoryByDeformation=arcHeightHistoryByDeformation.*(1000*1000); %convert to micron
arcHeightHistoryByAnsysMidElem=arcHeightHistoryByAnsysMidElem.*(1000*1000); %convert to micron
arcHeightHistoryByAnsysMaxElem=arcHeightHistoryByAnsysMaxElem.*(1000*1000); %convert to micron
arcHeightHistoryByAnsysTest=arcHeightHistoryByAnsysTest.*(1000*1000); %convert to micron

%BEGIN EXPORT TO EXCEL
 fileId=fopen([saveFilePath,num2str(randomInteger),'_data','.csv'],"w");
 if fileId==-1
     disp('ERROR: Cannot open file for writing');
 else
     fprintf(fileId,'%s,','Impact Number');
     fprintf(fileId,'%s,','Time Derived from contact frequency');
     fprintf(fileId,'%s,','Actual Sensor Time');
     fprintf(fileId,'%s,','Actual Grid Time');
     fprintf(fileId,'%s,','Arc Height from AvgNRG (Microns)');
     fprintf(fileId,'%s,','Arc Height from Beam Deformation (Microns)');
     fprintf(fileId,'%s,','Arc Height from Ansys Mid Node (Microns)');
     fprintf(fileId,'%s,','Arc Height from Ansys Max Height (Microns)');
     fprintf(fileId,'%s,','Arc Height from Ansys Test (Microns)');
     fprintf(fileId,'%s,','Coverage');
     fprintf(fileId,'%s,','Average Grid Energy Value, units depend on mode. Area Mode: J/m^2');
     fprintf(fileId,'%s,','Total Grid Energy Value, units depend on mode. Area Mode: J/m^2');
     fprintf(fileId,'
');
 end
lastTimeSaved=0;
for i=1:length(timeAxis) %loop over rows
    if (timeAxis(i)<30)||(timeAxis(i)-lastTimeSaved)>=2
        lastTimeSaved=timeAxis(i);
        fprintf(fileId,'%d,',impactNumberAxis(i));
        fprintf(fileId,'%d,',timeAxis(i));
        fprintf(fileId,'%d,',sensorTimeAxis(i));
        fprintf(fileId,'%d,',gridTimeAxis(i));
        fprintf(fileId,'%d,',arcHeightHistoryByAvgNRG(i));
        fprintf(fileId,'%d,',arcHeightHistoryByDeformation(i));
        fprintf(fileId,'%d,',arcHeightHistoryByAnsysMidElem(i));
        fprintf(fileId,'%d,',arcHeightHistoryByAnsysMaxElem(i));
        fprintf(fileId,'%d,',arcHeightHistoryByAnsysTest(i));
        fprintf(fileId,'%d,',coverageDescreteDomain(i));
        fprintf(fileId,'"n"');
    end
end
fprintf(fid,'%d,'); averageEntireGridEnergyHistory(i));
fprintf(fid,'%d,'); totalEntireGridEnergyHistory(i));
fprintf(fid, '\n');
end
fclose(fid);

%END EXPORT TO EXCEL
fprintf(fid,'%',saveFilePath, num2str(randomInteger), '_RESULTS_SETTINGS.txt','%', 'w');
fprintf(fid,'%', saveFilePath, fileNamePrefix, '_SET.txt','%', 'w');
fprintf(fid,'%s %s
', 'Random File Number: ', num2str(randomInteger));
fprintf(fid,'%s %s
', 'Code Version ', versionNumber);
fprintf(fid,'%s %s
', 'Completion Date & Time: ', datestr(now));
fprintf(fid,'%s %s
', 'Elaspsed Run-Time (Hours): ', num2str(elapsedTime/(60*60)));
fprintf(fid,'%s %s
', 'Run Efficiency (Impacts per seconds of Run-Time): '
num2str(N/(elapsedTime+0.0000000000000000000001));
fprintf(fid,'%s %s
', '% of Impacts that triggered a Sensitive Area Size warning: '
num2str(insufficientSensitiveAreaSize*100/N));
fprintf(fid,'%s %s
', '% of Impacts that triggered a element resolution warning: '
num2str(insufficientResolutionWarningCount*100/N));
fprintf(fid,'%s %s
', 'Length of Grid Element (micron): ', num2str(lengthOfGridElement*1000*1000));
if elecMagImpactVelocity==0
fprintf(fid,'%s
',contactDataFilename);
end
if frictionSwitch==true
fprintf(fid,'%s 
','Friction ON');
else
fprintf(fid,'%s 
','Friction OFF');
end
if bilinearMaterialModelSwitch==true
fprintf(fid,'%s 
','Bilinear Isotropic Perfectly Elastic Perfectly Plastic Material Model');
else
fprintf(fid,'%s 
','Power Law Isotropic Material Model');
end
if condition91Sa==true
fprintf(fid,'%s 
','91Sa');
else
fprintf(fid,'%s 
','112Sb');
end
fprintf(fid,'%s %s
','Contact Area Scaling Factor: ',num2str(contactAreaScalingFactor));
fprintf(fid,'%s %s
','Absorbed Energy Scaling Factor: ',num2str(energyTransferScalingFactor));
fprintf(fid,'%s %s
','Incident Energy Scaling Factor: ',num2str(energyIncidentScalingFactor));
fprintf(fid,'%s %s
','Velocity Scaling Factor: ',num2str(impactVelocityScalingFactor));
fprintf(fid,'%s %s
','Velocity Hi-Pass Cutoff: ',num2str(velocityHiPass));
fprintf(fid,'%s %s
','Minutes of Finishing: ',num2str(minutesOfFinishing));
fprintf(fid,'%s %s
','Number of Elements: ',num2str(numberOfGridElements));
if useAbsNrgPer==useAbsNrgPerBall
fprintf(fid,'%s 
','Absorbed Energy Tracking: BALL');
else if useAbsNrgPer==useAbsNrgPerContactZoneArea
fprintf(fid,'%s 
','Absorbed Energy Tracking: PER AREA (J/m^2)');
else if useAbsNrgPer==useAbsNrgPerContactZoneVolume
fprintf(fid,'%s 
','Absorbed Energy Tracking: PER VOLUME (J/m^3)');
else disp('Error: no such Absorbed Energy Tracking');
return;
end
fprintf(fid,'%s %s
','Energy Distribution: LINEAR');
elseif nrgDistOption==nrgDistReal
fprintf(fid,'%s 
','Energy Distribution: REAL');
else nrgDistOption==nrgDistUniform
fprintf(fid,'%s 
','Energy Distribution: UNIFORM');
end

fprintf(fid,'%',saveFilePath, num2str(randomInteger), '_RESULTS_SETTINGS.txt','%', 'w');
fprintf(fid,'%', saveFilePath, fileNamePrefix, '_SET.txt','%', 'w');
fprintf(fid,'%s %s
', 'Random File Number: ', num2str(randomInteger));
fprintf(fid,'%s %s
', 'Code Version ', versionNumber);
fprintf(fid,'%s %s
', 'Completion Date & Time: ', datestr(now));
fprintf(fid,'%s %s
', 'Elaspsed Run-Time (Hours): ', num2str(elapsedTime/(60*60)));
fprintf(fid,'%s %s
', 'Run Efficiency (Impacts per seconds of Run-Time): '
num2str(N/(elapsedTime+0.0000000000000000000001));
fprintf(fid,'%s %s
', '% of Impacts that triggered a Sensitive Area Size warning: '
num2str(insufficientSensitiveAreaSize*100/N));
fprintf(fid,'%s %s
', '% of Impacts that triggered a element resolution warning: '
num2str(insufficientResolutionWarningCount*100/N));
fprintf(fid,'%s %s
', 'Length of Grid Element (micron): ', num2str(lengthOfGridElement*1000*1000));
if elecMagImpactVelocity==0
fprintf(fid,'%s
',contactDataFilename);
end
if frictionSwitch==true
fprintf(fid,'%s 
','Friction ON');
else
fprintf(fid,'%s 
','Friction OFF');
end
if bilinearMaterialModelSwitch==true
fprintf(fid,'%s 
','Bilinear Isotropic Perfectly Elastic Perfectly Plastic Material Model');
else
fprintf(fid,'%s 
','Power Law Isotropic Material Model');
end
if condition91Sa==true
fprintf(fid,'%s 
','91Sa');
else
fprintf(fid,'%s 
','112Sb');
end
fprintf(fid,'%s %s
','Contact Area Scaling Factor: ',num2str(contactAreaScalingFactor));
fprintf(fid,'%s %s
','Absorbed Energy Scaling Factor: ',num2str(energyTransferScalingFactor));
fprintf(fid,'%s %s
','Incident Energy Scaling Factor: ',num2str(energyIncidentScalingFactor));
fprintf(fid,'%s %s
','Velocity Scaling Factor: ',num2str(impactVelocityScalingFactor));
fprintf(fid,'%s %s
','Velocity Hi-Pass Cutoff: ',num2str(velocityHiPass));
fprintf(fid,'%s %s
','Minutes of Finishing: ',num2str(minutesOfFinishing));
fprintf(fid,'%s %s
','Number of Elements: ',num2str(numberOfGridElements));
if useAbsNrgPer==useAbsNrgPerBall
fprintf(fid,'%s 
','Absorbed Energy Tracking: BALL');
else if useAbsNrgPer==useAbsNrgPerContactZoneArea
fprintf(fid,'%s 
','Absorbed Energy Tracking: PER AREA (J/m^2)');
else if useAbsNrgPer==useAbsNrgPerContactZoneVolume
fprintf(fid,'%s 
','Absorbed Energy Tracking: PER VOLUME (J/m^3)');
else disp('Error: no such Absorbed Energy Tracking');
return;
end
fprintf(fid,'%s %s
','Energy Distribution: LINEAR');
elseif nrgDistOption==nrgDistReal
fprintf(fid,'%s 
','Energy Distribution: REAL');
else nrgDistOption==nrgDistUniform
fprintf(fid,'%s 
','Energy Distribution: UNIFORM');
end

fprintf(fid,'%d,averageEntireGridEnergyHistory(i));
fprintf(fid,'%d,totalEntireGridEnergyHistory(i));
fprintf(fid, \n');
end
fclose(fid);

%END EXPORT TO EXCEL
fclose(fid);
fprintf(fileId,'%s 
', 'Energy Distribution: UNIFORM');
else
disp('Error: no such Energy Distribution Option');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if contactRadiusOption==contactRadiusFEA
fprintf(fileId,'%s 
','FEA Contact Radii');
elseif contactRadiusOption==contactRadiusHills
fprintf(fileId,'%s 
','Hills Contact Radii');
elseif contactRadiusOption==contactRadiusFixed
fprintf(fileId,'%s %s\n','Fixed Contact Radii: ',num2str(fixedContactZoneRadius));
else
disp('ERROR: No such contact radius option');
end
if elecMagImpactVelocity~=0
%    fprintf(fileId,'%s
','***ELECTROMAGNET********************************************
fprintf(fileId,'%s %s
','Drop Velocity: ',num2str(elecMagImpactVelocity));
fprintf(fileId,'%s %s
','Consider Bounces: ',num2str(elecMagConsiderBounces));
if elecMagConsiderBounces==true
fprintf(fileId,'%s %s
','Maxium Number of Bounces: ',num2str(elecMagMaxNumberofBounces));
end
fprintf(fileId,'%s %s
','Number of Balls: ',num2str(elecMagNumOfBalls));
fprintf(fileId,'%s %s
','Drop Frequency: ',num2str(elecMagFreq));
%fprintf(fileId,'%s
','***END*****************************************************
else
fprintf(fileId,'%s %s
','Sensor Collision Frequency (Hz):',num2str(averageSensorCollisionFrequencyHertz));
end
if useAnsysToCalculateArcHeight==true
fprintf(fileId,'%s 
','ANSYS Deformation Calculation: ON');
else
fprintf(fileId,'%s 
','ANSYS Deformation Calculation: OFF');
end
if useActualStressDistributionInsteadOfCenterlineStresses==true
fprintf(fileId,'%s 
','Using Actual Stress Distribution');
else
fprintf(fileId,'%s 
','Using Centerline Stresses');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf;
tempPlot = newplot;
%hold on;
plot(timeAxis,coverageDescreteDomain, 'linestyle', 'none', 'marker', 'x','markersize',3);
title(titleString);
xlabel(['Time (min) - Resolution: ', num2str(numberOfGridElements)]);
ylabel(['Coverage']);
saveas(tempPlot,[saveFilePath,fileNamePrefix,'_Coverage','.fig']);
saveas(tempPlot,[saveFilePath,fileNamePrefix,'_Coverage','.png'],'png');
cla reset;
clear tempPlot;
hold off;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if elecMagImpactVelocity==0
    tempPlot = newplot;
hold on;
plot(gridTimeAxis, arcHeightHistoryByAvgNRG, 'bo', 'markersize', 6);
plot(gridTimeAxis, arcHeightHistoryByDeformation, 'mx', 'markersize', 6);
if useAnsysToCalculateArcHeight == true
    plot(gridTimeAxis, arcHeightHistoryByAnsysMidElem, 'b+', 'markersize', 12);
    plot(gridTimeAxis, arcHeightHistoryByAnsysMaxElem, 'k+', 'markersize', 8);
    plot(gridTimeAxis, arcHeightHistoryByAnsysTest, 'kd', 'markersize', 4);
end
if graphExperimentalDataForComparision == true
    plotExp(strip617umSwitch, condition91Sa, elecMagImpactVelocity);
end
title(titleString);
xlabel(['Time from signal impact periods (min) - Resolution: ', num2str(numberOfGridElements)]);
ylabel(['Arc Height (microns)']);
saveas(tempPlot, [saveFilePath, fileNamePrefix, '_ArcHeight_GridTime','.fig']);
saveas(tempPlot, [saveFilePath, fileNamePrefix, '_ArcHeight_GridTime','.png']);
cla reset;
clear tempPlot;
hold off;
etempPlot = newplot;
end

function [newArcHeight] = arcHeightOverDifferentSpan(originalArcHeight, originalSpan, newSpan)
    sign = 0;
    if originalArcHeight == 0
        newArcHeight = 0;
    else
        if originalArcHeight > 0
            sign = 1;
            originalArcHeight = abs(originalArcHeight);
            sign = -1;
        end
        radius = ((originalSpan^2/4) + originalArcHeight^2) / (2 * originalArcHeight);
        arcHeightOfCentreNodeByDeformation = radius - sqrt(radius^2 - (1/4)*span^2);
        newArcHeight = sign * (radius - sqrt(radius^2 - (1/4)*newSpan^2));
    end
end

function [almenArcHeight] = almenArcHeightOverDifferentSpanTest(modelSpan, modelArcheight)
    almenMeasurementSpan = 0.0635;
    almenArcheight = modelArcheight * (almenMeasurementSpan/modelSpan)^2;
end
function stressTopLinear = equivalentLinearStressDistribFromArcHeight(beamArcHeight, beamLength, beamThickness, beamE)
    stressTopLinear = 4*beamArcHeight*beamE*beamThickness/beamLength^2;
end

function stressTopLinear = equivalentLinearStressDistribFromMomentPerUnitLength(momentPerUnitLengthInNewtons, beamThickness)
    stressTopLinear = 6*momentPerUnitLengthInNewtons/beamThickness^2;
end

function arcHeightInMicron = beamArcHeightFromMomentPerUnitLength(momentPerUnitLength, span, beamThickness)
    E=71.7E9; % modulus of elasticity (Pa)
    arcHeightInMicron=(3*momentPerUnitLength*span^2)/(2*E*beamThickness^3);
end

function arcHeight = getArcHeightforThinStripFromElectroCalibForAreaMode(AbsE)
    arcHeight=1.7247*AbsE;
end

function arcHeight = getArcHeightforThinStripFromElectroCalibForAreaModeLKUP(AbsE)
    persistent arcHeightTableThinStrip
    persistent arcHeightTableThinStripLength
    if isempty(arcHeightTableThinStrip)
        arcHeightTableThinStrip(1,1)=0; %absorbed Energy J/m^2
        arcHeightTableThinStrip(1,2)=0; %Arc Height Microns
        arcHeightTableThinStrip(2,1)=317.7;
        arcHeightTableThinStrip(2,2)=336;
        arcHeightTableThinStrip(3,1)=335.6;
        arcHeightTableThinStrip(3,2)=455;
        arcHeightTableThinStrip(4,1)=339.0; %absorbed Energy J/m^2
        arcHeightTableThinStrip(4,2)=518; %Arc Height Microns
        arcHeightTableThinStrip(5,1)=340.2; %absorbed Energy J/m^2
        arcHeightTableThinStrip(5,2)=545; %Arc Height Microns
        arcHeightTableThinStrip(6,1)=341.3; %absorbed Energy J/m^2
        arcHeightTableThinStrip(6,2)=563; %Arc Height Microns
        arcHeightTableThinStrip(7,1)=341.7; %absorbed Energy J/m^2
        arcHeightTableThinStrip(7,2)=565; %Arc Height Microns
        arcHeightTableThinStrip(8,1)=341.9; %absorbed Energy J/m^2
        arcHeightTableThinStrip(8,2)=525; %Arc Height Microns
        arcHeightTableThinStrip(9,1)=341.9; %absorbed Energy J/m^2
        arcHeightTableThinStrip(9,2)=565; %Arc Height Microns
        arcHeightTableThinStripLength=length(arcHeightTableThinStrip(:,1));
    end
    foundArcHeight=false;
    for i=1:arcHeightTableThinStripLength-1
        if AbsE==arcHeightTableThinStrip(i,1)
            arcHeight=arcHeightTableThinStrip(i,2);
            foundArcHeight=true;
            break;
        end
        if AbsE==arcHeightTableThinStrip(i+1,1)
            arcHeight=arcHeightTableThinStrip(i+1,2);
            foundArcHeight=true;
            break;
        end
        if (AbsE < arcHeightTableThinStrip(i+1,1)) && (AbsE > arcHeightTableThinStrip(i,1))
            % arcHeight = arcHeightVsEabs(i+1,2) * ... (
            %     (ArcE-arcHeightVsEabs(i+1,1)) / (ArcE-arcHeightVsEabs(i,1))
            % ) + arcHeightVsEabs(i,2);
            arcHeight=ArcA; % arcHeight = arcHeightVsEabs(1+1,2) *...
            % (ArcE-arcHeightVsEabs(i+1,1)) / (ArcE-arcHeightVsEabs(i,1))
            % ) + arcHeightVsEabs(i,2);
            if (foundArcHeight==true) % check this!
                arcHeight=arcHeightTableThinStrip(length(arcHeightTableThinStrip(:,1)),2);
            end
        end
    end
end

% function contactRadius = AA3003O617um91aHillsModelContactRadius(impactVelocity)
E1=206.8E9;  % modulus of elasticity (Pa)
poissonsRatio1=0.28;
density1=7800; %(kg/m^3)
Radius1=3.175E-3; %(m) check this!
E2=71.7E9; % modulus of elasticity (Pa)
poissonsRatio2=0.333;

E1=206.8E9;  % modulus of elasticity (Pa)
poissonsRatio1=0.28;
density1=7800; %(kg/m^3)
Radius1=3.175E-3; %(m) check this!
E2=71.7E9; % modulus of elasticity (Pa)
poissonsRatio2=0.333;
% almenStrapThickness=6.17E-4; %(m) 0.025" = 6.35E-4 m
Determine actual thickness
% yieldStrengthInShear2=41E6/2; %(Pa)
k1=(1-poissonsRatio1^2)/(pi*E1);
k2=(1-poissonsRatio2^2)/(pi*E2);
contactRadius=Radius1*((5/4)*(impactVelocity^2)*density1*pi^2*(k1+k2))^(1/5);

function [AbsE, coefficientOfRestitution] = coefficientOfResGivenAbseAndVLTB05( Strip617umSwitch, bilinearMaterialModel, pAbsE, impactVelocity)

dbg=true;
if dbg==true
    clear all;
cld;
    Strip617umSwitch=false;
    bilinearMaterialModel=false;
    pAbsE=0;
    impactVelocity=0.006;
end

mlock;

function [AbsE, coefficientOfRestitution] = coefficientOfResGivenAbseAndVLTB05( Strip617umSwitch, bilinearMaterialModel, pAbsE, impactVelocity)

if isempty(archHeightVsEabs)
    % Persistent Variables to improve speed:
    persistent lowerPage;
    persistent upperPage;
    persistent minUpperVdiff;
    persistent minLowerVdiff;
    persistent firstLowerPage;
    persistent firstUpperPage;
    persistent impactVelocityLowerThanELookupMinVel;
    persistent upperV;
    persistent lowerV;
    persistent noVelocityInterpolationRequired;
    persistent testVel;
    persistent velDiff;
    persistent lowerPabsIndx;
    persistent upperPabsIndx;
    persistent pages;
    persistent numberOfPages;
    persistent overMaximumAbsorbedEnergy;
    persistent firstLowerpAbsIndx;
    persistent minLowerpAbsEdiff;
    persistent upperpAbsEdiff;
    persistent lowerpAbsEdiff;
    persistent minLowerpAbsEdiff;
    persistent lowerpAbsEdiff;
    persistent lowerpAbsIndx;
    persistent upperpAbsIndx;
    persistent firstUpperpAbsIndx;
    persistent minUpperpAbsIndx;
    persistent upperpAbsIndx;
    persistent lowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent lowerLowerpAbsEdiff;
    persistent upperLowerpAbsEdiff;
    persistent lowerLowerpAbsEdiff;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
    persistent upperLowerpAbsIndx;
    persistent lowerLowerpAbsIndx;
persistent eLowerVelLower;
persistent eUpperVelLower;
persistent eLower;
persistent reboundVelocity;
persistent currentlyAbsorbedEnergy;
persistent foundArcHeight;

persistent eLookupVeloIndx=1;
persistent eLookupAbsIndx=2;
persistent eLookupCResIndx=3;
persistent eLookupMaxVel=0;
persistent eLookupMinVel=0;

massOfSingleMedia = 1.03E-03; %kg

if ((Strip617umSwitch==true)&&(bilinearMaterialModel==true))
    exit;
elseif ((Strip617umSwitch==true)&&(bilinearMaterialModel==false))
    feaLabel='PwrLw617um';
elseif ((Strip617umSwitch==false)&&(bilinearMaterialModel==true))
    exit;
elseif ((Strip617umSwitch==false)&&(bilinearMaterialModel==false))
    feaLabel='PwrLw779um';
else
    disp('error');
    exit(0);
end

eLookupTableFilename=['eLookUp_',feaLabel];
arcHeightVsEabsFilename=['aLookUp_',feaLabel];

disp(eLookupTableFilename);
disp(arcHeightVsEabsFilename);
load(eLookupTableFilename, 'eLookupTable');
load(arcHeightVsEabsFilename, 'arcHeightVsEabs');

if (isempty(arcHeightVsEabs)||isempty(eLookupTable))
    disp('ERROR: Could not load material model files');
    return;
end

arcHeightVsEabsImpacts=length(arcHeightVsEabs(:,1));
eLookupImpacts=length(eLookupTable(:,1,1));
eLookupPages=length(eLookupTable(1,:,:));

for i=1:eLookupPages
    testVel = eLookupTable(1,eLookupVeloIndx,i);
    if i==1
        eLookupMinVel=testVel;
eLookupMinVelPage=i;
    else
        if (testVel>eLookupMaxVel)
            eLookupMaxVel=testVel;
            end
        end
        if (testVel<eLookupMinVel)
            eLookupMinVel=testVel;
eLookupMinVelPage=i;
        end
        end
maxEabsArch = arcHeightVsEabs(arcHeightVsEabsImpacts,1);
numberOfPointsInSectionArch = 20;

for i=1:floor(arcHeightVsEabsImpacts/numberOfPointsInSectionArch)+1;
    k=1;
    for j=1:numberOfPointsInSectionArch
        sectionsArch(i,j)=arcHeightVsEabs(j,1);
        sectionsArch(i,1)=sectionsArch(i,1)+1;
    end
end

disp(feaLabel);

if (impactVelocity > eLookupMaxVel)
    disp('ERROR: Velocity out of range');
end
return;
end
if impactVelocity < 0
impactVelocity = abs(impactVelocity);
end
if impactVelocity == 0
coefficientOfRestitution = 1;
AbsE = pAbsE;
else

if impactVelocity < 0
impactVelocity = abs(impactVelocity);
end
if impactVelocity == 0
coefficientOfRestitution = 1;
AbsE = pAbsE;
else

if impactVelocityLowerThanELookupMinVel == false;
impactVelocityLowerThanELookupMinVel = true;
lowerVelocity = 0;
else
for page = 1:eLookupPages
testVelocity_Lower = eLookupTable(1,eLookupVeloIndex,page);
testVelocity_High = eLookupTable(1,eLookupVeloIndex,page+1);
if (((impactVelocity > testVelocity_Lower) && (impactVelocity < testVelocity_High))
upperPage = page+1;
lowerPage = page;
upperVelocity = testVelocity_High;
lowerVelocity = testVelocity_Lower;
end
if impactVelocity > testVelocity_High
noVelocityInterpolationRequired = true;
upperPage = page;
lowerPage = page;
upperVelocity = testVelocity_Lower;
break;
end
if impactVelocity < testVelocity_Lower
noVelocityInterpolationRequired = true;
upperPage = page+1;
lowerPage = page+1;
upperVelocity = testVelocity_High;
lowerVelocity = testVelocity_High;
break;
end
end
end
end

cosine = zeros(2,1);
lowerPAbsE = 0;
lowerPAbsIndex = 0;
upperPAbsIndex = 0;
upperPAbsE = 0;
if (lowerPage == upperPage) || (impactVelocityLowerThanELookupMinVel == true)
numberOfPages = 1;
else
numberOfPages = 2;
end
if impactVelocityLowerThanELookupMinVel == true
pages(1) = eLookupMinVelPage;
upperPage = eLookupMinVelPage;
else
pages(1) = upperPage;
end
pages(2) = lowerPage;
overMaximumAbsorbedEnergy(1) = false;
overMaximumAbsorbedEnergy(2) = false;
for j = 1:numberOfPages
maxPAbsE = eLookupTable(eLookupImpacts,eLookupEAbsIndex,pages(j));
if pAbsE > maxPAbsE
overMaximumAbsorbedEnergy(j) = true;
% upperPAbsE(j) = maxPAbsE;
% upperPAbsIndex(j) = eLookupImpacts;
% lowerPAbsE(j) = maxPAbsE;
% lowerPAbsIndex(j) = eLookupImpacts;
else
    firstUpperpAbsE=true;
    firstLowerpAbsE=true;
    for i=1:eLookupImpacts
        testPAbsE = eLookupTable(i,eLookupEAbsIndx,pages(j));
        pAbsEDiff = testPAbsE - pAbsE;
        if pAbsEDiff > 0
            if firstUpperpAbsE==true
                firstUpperpAbsE=false;
                minUpperpAbsDiff=abs(pAbsEDiff);
                upperpAbsE(j)=testPAbsE;
                upperPabsIndx(j)=i;
            else
                if (abs(pAbsEDiff) < minUpperpAbsDiff)
                    minUpperpAbsDiff=abs(pAbsEDiff);
                    upperpAbsE(j)=testPAbsE;
                    upperPabsIndx(j)=i;
                end
            end
        elseif pAbsEDiff < 0
            if firstLowerpAbsE==true
                firstLowerpAbsE=false;
                minLowerpAbsDiff=abs(pAbsEDiff);
                lowerpAbsE(j)=testPAbsE;
                lowerPabsIndx(j)=i;
            else
                if (abs(pAbsEDiff) < minLowerpAbsDiff)
                    minLowerpAbsDiff=abs(pAbsEDiff);
                    lowerpAbsE(j)=testPAbsE;
                    lowerPabsIndx(j)=i;
                end
            end
        else
            upperpAbsE(j)=testPAbsE;
            upperPabsIndx(j)=i;
            lowerpAbsE(j)=testPAbsE;
            lowerPabsIndx(j)=i;
            break;
        end
    end
end

if overMaximumAbsorbedEnergy(1)==false
    if upperpAbsE(1)==lowerpAbsE(1)
        eUpper=eLookupTable(upperPabsIndx(1),eLookupCResIndx,upperPage);
    else
        upperpAbsEvelUpper=upperpAbsE(1);
        lowerpAbsEvelUpper=lowerpAbsE(1);
        upperPabsIndxVelUpper=upperPabsIndx(1);
        lowerPabsIndxVelUpper=lowerPabsIndx(1);
        eLowerVelUpper=eLookupTable(lowerPabsIndxVelUpper,eLookupCResIndx,upperPage);
        eUpperVelUpper=eLookupTable(upperPabsIndxVelUpper,eLookupCResIndx,upperPage);
        eUpper = eUpperVelUpper + ((pAbsE-upperpAbsEvelUpper)/(lowerpAbsEvelUpper-upperpAbsEvelUpper))*eLowerVelUpper-
        eUpperVelUpper;
    end
    else
        eUpper=1;
    end
end

if (impactVelocityLowerThanELookupMinVel==true)
    if noVelocityInterpolationRequired==false
        if overMaximumAbsorbedEnergy(2)==false
            if upperpAbsE(2)==lowerpAbsE(2)
                eLower=eLookupTable(upperPabsIndx(2),eLookupCResIndx,lowerPage);
            else
                upperpAbsEvelLower=upperpAbsE(2);
                lowerpAbsEvelLower=lowerpAbsE(2);
                upperPabsIndxVelLower=upperPabsIndx(2);
                lowerPabsIndxVelLower=lowerPabsIndx(2);
                eLowerVelLower=eLookupTable(lowerPabsIndxVelLower,eLookupCResIndx,lowerPage);
                eUpperVelLower=eLookupTable(upperPabsIndxVelLower,eLookupCResIndx,lowerPage);
                eLower = eUpperVelLower + ((pAbsE-upperpAbsEvelLower)/(lowerpAbsEvelLower-upperpAbsEvelLower))*eLowerVelLower-
                eUpperVelLower;
            end
        else
            eLower=1;
        end
    end
end

if impactVelocityLowerThanELookupMinVel==true
    eLower=1;
    lowerVel=0;
end
if (lowerPage==upperPage)
    coefficientOfRestitution = eUpper;
else
    coefficientOfRestitution = eUpper + ((impactVelocity-lowerVel)/(lowerVel-upperVel))*eLower-eUpper;
end
\[ \text{reboundVelocity} = \text{impactVelocity} \times \text{coefficientOfRestitution}; \]
\[ \text{currentlyAbsorbedEnergy} = 0.5 \times \text{massOfSingleMedia} \times \text{impactVelocity}^2 - \text{reboundVelocity}^2; \]
\[ \text{AbsE} = \text{pAbsE} + \text{currentlyAbsorbedEnergy}; \]
\[ \text{end} \quad \text{if} \quad \text{impactVelocity} = 0 \]

\[ \text{if} \ \text{AbsE} < \text{arcHeightVsEabs(} \text{arcHeightVsEabsImpacts,1)} \]
\[ \text{for} \ j=1: \text{numberOfSectionsArcH} \]
\[ \text{if} \ \text{AbsE} < \text{sectionsArcH(j,1)} \quad \text{or} \quad \text{AbsE} > \text{sectionsArcH(j,2)} \]
\[ \text{break}; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{for} \ i= \text{sectionsArcH(j,3)}: \text{sectionsArcH(j,4)} \]
\[ \text{if} \ \text{AbsE} < \text{arcHeightVsEabs(i,1)} \quad \text{or} \quad \text{AbsE} > \text{arcHeightVsEabs(i,2)} \]
\[ \text{foundArcHeight=true}; \]
\[ \text{break}; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{else} \]
\[ \text{arcHeight=arcHeightVsEabs(} \text{arcHeightVsEabsImpacts,2)}; \]
\[ \text{foundArcHeight=true}; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{if} \ \text{foundArcHeight=false} \]
\[ \text{disp('Error!')}; \]
\[ \text{end} \]

\[ \text{function [archeightOfCentreNode]=arcHeightAtCenterFromArcHeightOfElements01(} \text{arcHeightGrid, nodes, lengthOfGridElement)} \]
\[ \text{span=0.0635} \quad \text{meters \ (i.e. 2.5"=0.0635m)} \]
\[ \text{X=1; \ enumeration \ for \ nodes \ index} \]
\[ \text{Y=2; \ enumeration \ for \ nodes \ index} \]
\[ \beta=0; \]
\[ \text{numberOfGridElements}=\text{length(} \text{arcHeightGrid)}; \]
\[ \text{numberOfNodes}=\text{length(} \text{nodes)}; \]
\[ \text{if} \ \text{numberOfNodes} \neq \text{numberOfGridElements}+1 \]
\[ \text{disp('ERROR: The dimensions of the input arguments are incorrect')} \]
\[ \text{return}; \]
\[ \text{end} \]
\[ \psi=\text{zeros(} \text{numberOfGridElements,1)}; \]
\[ \text{for} \ e=0: \text{numberOfGridElements}-1 \]
\[ \text{if} \ e>0 \]
\[ \text{APrev=arcHeightGrid(e)}; \]
\[ \text{else} \]
\[ \text{APrev=0}; \]
\[ \text{end} \]
\[ \text{A=arcHeightGrid(e+1)}; \]
\[ \text{if} \ \text{APrev}=0 \]
\[ \text{gammaPrev=0}; \]
\[ \text{else} \]
\[ \text{if} \ \text{APrev} > 0 \]
\[ \text{gammaSignPrev=-1}; \]
\[ \text{else} \]
\[ \text{gammaSignPrev=1}; \]
\[ \text{end} \]
\[ \text{APrev=abs(APrev)}; \]
\[ \text{RPrev=} \text{((span}^2/4)+\text{APrev}^2)/(2*\text{APrev}); \]
\[ \text{if} \ \text{abs(lengthOfGridElement)}/(2*RPrev)>1 \]
\[ \text{disp('Error: Out of Range for arcsin')}; \]
\[ \text{end} \]
\[ \text{gammaPrev=gammaSignPrev*asin(lengthOfGridElement)/(2*RPrev)}; \]
\[ \text{end} \]
\[ \text{if} \ \text{A}=0 \]
\[ \text{gamma=0}; \]
\[ \text{else} \]
\[ \text{if} \ A > 0 \]
\[ \text{gammaSign=1}; \]
\[ \text{else} \]
\[ \text{gammaSign=-1}; \]
\[ \text{end} \]

196
A=abs(A);
R=((span^2/4)+A^2)/(2*A);
if abs(lengthOfGridElement/(2*R))>1
    disp('Error: Out of Range for arcsin');
    end
    gamma=gammaSign*asin(lengthOfGridElement/(2*R));
end
beta=gammaPrev+gamma;
if e==0
    psi(e+1)=beta;
else
    psi(e+1)=beta+psi(e);
end
end
for e=1:numberOfGridElements
    nodes(e+1,X)= lengthOfGridElement*cos(psi(e)) + nodes(e,X);
    nodes(e+1,Y)= lengthOfGridElement*sin(psi(e)) + nodes(e,Y);
end
theta = atan(-nodes(numberOfNodes,Y)/nodes(numberOfNodes,X));

function [arcHeight] = arcHeightFromAbseLKTB( Strip617umSwitch, bilinearMaterialModel, frictionSwitch,useAbsNrgPer ,AbsE)

depbug=false;
if debug==true
    clear all;
clear;
    Strip617umSwitch=false;
    bilinearMaterialModel=false;
    AbsE=0.0000001;
    useAbsNrgPer=false;
end

% mlock;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%these should exist in the hybrid sin as well.
useAbsNrgPerBall=0;
useAbsNrgPerContactZoneArea=1;
useAbsNrgPerContactZoneVolume=2;
useAbsNrgPerContactZoneAreaAndRealNrgDist=3;
useAbsNrgPerContactZoneVolumeAndRealNrgDist=4;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Persistent Variables for algorithm
persistent arcHeightVsAbsAbs;
persistent arcHeightVsAbsImpacts;
persistent bilinearMaterialModelOrig;
persistent Strip617umSwitchOrig;
persistent frictionSwitchOrig;
persistent currentlyAbsorbedEnergy;
persistent foundArcHeight;
% Persistent Variables for algorithms that improve
persistent sections;
persistent maxAbsArcH;
persistent numberOfSections;
%persistent currentlyAbsorbedEnergy;
persistent foundArcHeight;
%Persistent ffeLabel;
persistent sectionUpperAbsNrgIndx;
persistent sectionLowerAbsNrgIndx;
persistent sectionUpperSecNumIndx;
persistent sectionLowerSecNumIndx;
loadNewLookUpTable=false;
if (~isequal(bilinearMaterialModelOrig))
    if (bilinearMaterialModelOrig == bilinearMaterialModel) || ...
        (Strip617umSwitchOrig == Strip617umSwitch) || ...
        (useAbsNrgPerOrig == useAbsNrgPer) || ...
        (frictionSwitchOrig == frictionSwitchOrig)

197
disp('Loading new Look-up Table in arcHeightFromAbseLKTB10.m - switched material model');
if loadNewLookUpTable==true;
    arcHeightVsEabs=[];
end
end

if isempty(arcHeightVsEabs) || loadNewLookUpTable==true)

    if ((Strip617umSwitch==true)&&(bilinearMaterialModel==true)&&(frictionSwitch==false))
        felaLabel='Biln617um';
    elseif ((Strip617umSwitch==true)&&(bilinearMaterialModel==false)&&(frictionSwitch==false))
        felaLabel='PwrLw617um';
    elseif ((Strip617umSwitch==false)&&(bilinearMaterialModel==true)&&(frictionSwitch==false))
        felaLabel='Biln779um';
    elseif ((Strip617umSwitch==false)&&(bilinearMaterialModel==false)&&(frictionSwitch==false))
        felaLabel='PwrLw779um';
    elseif ((Strip617umSwitch==true)&&(bilinearMaterialModel==true)&&(frictionSwitch==true))
        felaLabel='Biln617umF';
    elseif ((Strip617umSwitch==true)&&(bilinearMaterialModel==false)&&(frictionSwitch==true))
        felaLabel='PwrLw617umF';
    elseif ((Strip617umSwitch==false)&&(bilinearMaterialModel==true)&&(frictionSwitch==true))
        felaLabel='Biln779umF';
    elseif ((Strip617umSwitch==false)&&(bilinearMaterialModel==false)&&(frictionSwitch==true))
        felaLabel='PwrLw779umF';
    else
        disp('ERROR in arcHeightFromAbseLKTB10.m');
        return;
end

bilinearMaterialModelOrig = bilinearMaterialModel;
Strip617umSwitchOrig = Strip617umSwitch;
frictionSwitchOrig = frictionSwitch;
useAbsNrgPerOrig=useAbsNrgPer;

if (useAbsNrgPer==useAbsNrgPerContactZoneArea)||(useAbsNrgPer==useAbsNrgPerContactZoneAreaAndRealNrgDist)
    arcHeightVsEabsFilename=['ALookUpArea_','felaLabel'];
else useAbsNrgPer==useAbsNrgPerBall
    arcHeightVsEabsFilename=['ALookUp_','felaLabel'];
else useAbsNrgPer==useAbsNrgPerContactZoneVolume)||(useAbsNrgPer==useAbsNrgPerContactZoneVolumeAndRealNrgDist)
    arcHeightVsEabsFilename=['ALookUpVol_','felaLabel'];
end

disp(arcHeightVsEabsFilename);
    if loadNewLookUpTable==true
        clear arcHeightVsEabs;
    end
load(arcHeightVsEabsFilename, 'arcHeightVsEabs');
    if isempty(arcHeightVsEabs)
        disp('ERROR: Could not load material model files');
        return;
    end

arcHeightVsEabsImpacts=length(arcHeightVsEabs(:,1));
sectionLowerAbsNrgIndx=1;
sectionUpperAbsNrgIndx=2;
sectionLowerSecNumIndx=3;
sectionUpperSecNumIndx=4;

maxEabsArcH = arcHeightVsEabs(arcHeightVsEabsImpacts,1);
%absEIncArcH = maxEabsArcH/arcHeightVsEabsImpacts;
numberOfPointsInSectionArcH = 20;
numberOfSections=floor(arcHeightVsEabsImpacts/numberOfPointsInSectionArcH);

for i=1:floor(arcHeightVsEabsImpacts/numberOfPointsInSectionArcH)
    j=[1:numberOfPointsInSectionArcH + 1]';
    k=[1:(numberOfPointsInSectionArcH + 1)]';
    sections(:,i,sectionLowerAbsNrgIndx)=arcHeightVsEabs(j,1);
    sections(:,i,sectionUpperAbsNrgIndx)=arcHeightVsEabs(k,1);
    sections(:,i,sectionLowerSecNumIndx)=j;
    sections(:,i,sectionUpperSecNumIndx)=k;
end

foundArcHeight=false;
if AbsE < arcHeightVsEabs(arcHeightVsEabsImpacts,1)
    sectionContainingAbsE=0;
    for sectionContainingAbsE=1:numberOfSections
        if (AbsE>sections(sectionContainingAbsE,sectionLowerAbsNrgIndx))
            (AbsE<sections(sectionContainingAbsE,sectionUpperAbsNrgIndx))
                break;
        end
    end
    for i=sections(sectionContainingAbsE,sectionLowerSecNumIndx):sections(sectionContainingAbsE,sectionUpperSecNumIndx)-1
        if AbsE==arcHeightVsEabs(i,1)
            arcHeight=arcHeightVsEabs(i,2);
            foundArcHeight=true;
        end
    end
end
foundArcHeight=true;
break;
end
if AbsE==arcHeightVsEabs(i+1,1)
    arcHeight=arcHeightVsEabs(i+1,2);
    foundArcHeight=true;
    break;
end
if (AbsE < arcHeightVsEabs(i+1,1))&&(AbsE > arcHeightVsEabs(i,1))
    E=AbsE;
    Eu=arcHeightVsEabs(i+1,1);
    Al=arcHeightVsEabs(i+1,2);
    Au=arcHeightVsEabs(i,2);
    El=arcHeightVsEabs(i,1);
    A=A+(x-Eu)*(Al-Au)/(El-Eu);
    % arcHeight = arcHeightVsEabs(i+1,2) +...
    % (AbsE-arcHeightVsEabs(i+1,1))/(arcHeightVsEabs(i+1,1)-
    arcHeightVsEabs(i+1,1)))*(arcHeightVsEabs(i+1,2)-arcHeightVsEabs(i+1,2));
    foundArcHeight=true;
    break;
end
if (foundArcHeight==false)&&(i==(arcHeightVsEabsImpacts-1))
    arcHeight=arcHeightVsEabs(arcHeightVsEabsImpacts,2);
    foundArcHeight=true;
end
end
else
    arcHeight=arcHeightVsEabs(arcHeightVsEabsImpacts,2);
    foundArcHeight=true;
end
if foundArcHeight==false
    disp('Error!');
end

function [AbsE, coefficientOfRestitution]  = coefficientOfResGivenAbseAndVLKTB( Strip617umSwitch, bilinearMaterialModel, frictionSwitch, useAbsNrgPer ,pAbsE, impactVelocity)

persistent bilinearMaterialModelOrig;
persistent Strip617umSwitchOrig;
persistent useAbsNrgPerContactZoneAreaOrig;
persistent frictionSwitchOrig;
persistent eLookupTable;
persistent eLookupMaxVel;
persistent eLookupMinVel;
persistent eLookupPages;
persistent eLookupImpacts;
persistent eLookupVeloIndx;
persistent eLookupEAbsIndx;
persistent eLookupCResIndx;
persistent eLookupMinVelPage;
persistent massOfSingleMedia;

persistent lowerPage;
persistent upperPage;
%persistent minUpperVdiff;
%persistent minLowerVdiff;
%persistent firstLowerPage;
%persistent firstUpperPage;
persistent impactVelocityLowerThanELookupMinVel;
persistent upperVel;
persistent lowerVel;
persistent noVelocityInterpolationRequried;
persistent testVel;
%persistent velDiff;
%persistent lowerPAbsE;
%persistent lowerPabsIndx;
persistent upperPabsIndx;
%persistent upperPAbsE;
persistent pages;
persistent numberOfPages;
persistent overMaximumAbsorbedEnergy;
persistent firstUpperAbsE;
persistent minUpperPabsEdiff;
persistent upperpAbsE;
%persistent upperPabsIndx;
persistent firstLowerpAbsE;
persistent minLowerpAbsEdiff;
persistent lowerpAbsE;
%persistent lowerPabsIndx;
persistent upperpAbsEvelUpper;
persistent lowerpAbsEvelUpper;
persistent upperpAbsIndxVelUpper;
persistent lowerpAbsIndxVelUpper;
persistent eLowerVelUpper;
persistent eUpperVelUpper;
persistent upperpAbsEvelLower;
persistent lowerpAbsEvelLower;
persistent upperpAbsIndxVelLower;
persistent lowerpAbsIndxVelLower;
persistent eLowerVelLower;
persistent eUpperVelLower;
persistent reboundVelocity;
persistent currentlyAbsorbedEnergy;

these should exist in the hybrid sim as well.
useAbsNrgPerBall=0;
useAbsNrgPerContactZoneArea=1;
useAbsNrgPerContactZoneVolume=2;
useAbsNrgPerContactZoneAreaAndRealNrgDist=3;
useAbsNrgPerContactZoneVolumeAndRealNrgDist=4;

debug=false;
if debug==true
    clear all;
    clf;
    Strip617umSwitch=false;
bilinearMaterialModel=false;
pAbsE=0;
impactVelocity=0.006;
useAbsNrgPer=useAbsNrgPerContactZoneArea;
end

if isempty(eLookupTable)%||(loadNewLookUpTable==true)

if isempty(eLookupTable) || (loadNewLookUpTable==true)

if isempty(eLookupTable) || (loadNewLookUpTable==true)

if isempty(eLookupTable) || (loadNewLookUpTable==true)
elseif useAbsNrgPer==useAbsNrgPerBall
  eLookupTableFilename=['eLookUp_',feaLabel]; %%% CHECK!!! SHOULD 'eLookUp_' BE 'eLookUpBall_'
elseif useAbsNrgPer==useAbsNrgPerContactZoneVolume||useAbsNrgPer==useAbsNrgPerContactZoneVolumeAndRealNrgDist
  eLookupTableFilename=['eLookUpVol_',feaLabel];
else
end
disp(eLookupTableFilename);

% if loadNewLookUpTable==true
% clear eLookupTable;
% end
load(eLookupTableFilename, 'eLookupTable');

if isempty(eLookupTable)
  disp('ERROR: Could not load material model files');
  return;
end
eLookupImpacts=length(eLookupTable(:,:,1));
eLookupPages=length(eLookupTable(1,:,:));
for i=1:eLookupPages
  testVel = eLookupTable(1,eLookupVeloIndx,i);
  if i==1
    eLookupMinVel=testVel;
    eLookupMinVelPage=i;
  else
    if (testVel>eLookupMaxVel)
      eLookupMaxVel=testVel;
    end
    if (testVel<eLookupMinVel)
      eLookupMinVel=testVel;
      eLookupMinVelPage=i;
    end
  end
end

if (impactVelocity > eLookupMaxVel)
  disp('ERROR: Velocity out of range');
  return;
end
if impactVelocity <0
  impactVelocity=abs(impactVelocity);
end
if impactVelocity==0
  coefficientOfRestitution=1;
  AbsE=pAbsE;
else
  lowerPage=0;
  upperPage=0;
  impactVelocityLowerThanELookupMinVel=false;
  upperVal=0;
  lowerVal=0;
  noVelocityInterpolationRequired=false;
  if impactVelocity < eLookupMinVel
    impactVelocityLowerThanELookupMinVel=true;
    lowerVal=0;
    upperVel=eLookupMinVel;
  else
    for page=1:eLookupPages-1
      testVelLw = eLookupTable(1,eLookupVeloIndx,page);
      testVelHi = eLookupTable(1,eLookupVeloIndx,page+1);
      if ((impactVelocity>testVelLw)&&(impactVelocity<testVelHi))
        upperPage=page+1;
        lowerPage=page;
      end
    end
    testVelLw = eLookupTable(1,eLookupVeloIndx,lowerPage);
    testVelHi = eLookupTable(1,eLookupVeloIndx,upperPage);
    if ((impactVelocity>testVelLw)&&(impactVelocity<testVelHi))
      upperPage=page+1;
      lowerPage=page;
    end
  end
end
upperVel=testVelHi;
lowerVel=testVelLw;
end
if impactVelocity==testVelHi
noVelocityInterpolationRequired=true;
upperPage=page;
lowerPage=page;
upperVel=testVelLw;
lowerVel=testVelLw;
brea;}
lowerPAbsE(j) = testPAbsE;
lowerPabsIndx(j) = i;
break;
end
end
end
end
if overMaximumAbsorbedEnergy(1) == false
if upperpAbsE(1) == lowerpAbsE(1)
eUpper = eLookupTable(upperPabsIndx(1), eLookupCResIndx, upperPage);
else
upperPabsEVelUpper = upperpAbsE(1);
lowerPabsEVelUpper = lowerpAbsE(1);
upperPabsIndxVelUpper = upperPabsIndx(1);
lowerPabsIndxVelUpper = lowerPabsIndx(1);
eLowerVelUpper = eLookupTable(lowerPabsIndxVelUpper, eLookupCResIndx, upperPage);
eUpperVelUpper = eLookupTable(upperPabsIndxVelUpper, eLookupCResIndx, upperPage);
eUpper = eUpperVelUpper + ((pAbsE - upperPabsEVelUpper)/(lowerPabsEVelUpper - upperPabsEVelUpper)) * (eLowerVelUpper - eUpperVelUpper);
else
eUpper = 1;
end
else
endif
if (impactVelocityLowerThanELookupMinVel == true)
if noVelocityInterpolationRequired == false
if overMaximumAbsorbedEnergy(2) == false
if upperPabsE(2) == lowerPabsE(2)
eLower = eLookupTable(upperPabsIndx(2), eLookupCResIndx, lowerPage);
else
upperPabsEVelLower = upperpAbsE(2);
lowerPabsEVelLower = lowerpAbsE(2);
upperPabsIndxVelLower = upperPabsIndx(2);
lowerPabsIndxVelLower = lowerPabsIndx(2);
eLowerVelLower = eLookupTable(lowerPabsIndxVelLower, eLookupCResIndx, lowerPage);
eUpperVelLower = eLookupTable(upperPabsIndxVelLower, eLookupCResIndx, lowerPage);
eLower = eUpperVelLower + ((pAbsE - upperPabsEVelLower)/(lowerPabsEVelLower - upperPabsEVelLower)) * (eLowerVelLower - eUpperVelLower);
else
eLower = 1;
end
endif
else
eLower = 1;
else
endif
if impactVelocityLowerThanELookupMinVel == true
eLower = 1;
else
endif
if (lowerPage == upperPage)
coefficientOfRestitution = eUpper;
else
coefficientOfRestitution = eUpper + ((impactVelocity - upperVel)/(lowerVel - upperVel)) * (eLower - eUpper);
endif
reboundVelocity = impactVelocity * coefficientOfRestitution;
currentlyAbsorbedEnergy = 0.5 * massOfSingleMedia * (impactVelocity^2 - reboundVelocity^2);
AbsE = pAbsE + currentlyAbsorbedEnergy;
end % impactVelocity==0

clear all;
clc;

feLabel = 'BiLn617um';
feLabel = 'BiLn617umF';
feLabel = 'BiLn779um';
feLabel = 'PwrLw617um';
feLabel = 'PwrLw617umC';
feLabel = 'PwrLw779um';
feLabel = 'PwrLw779umUy';
feLabel = 'PwrLw617umUy';
inputPaths = ['C:\_DOCS\PhD\TREIS\_Model\FE\FEA Output\', feLabel, '\'];
useMultiPointStressesInsteadOfCenterline = false; this attempts to free the centerline stress restriction
normalizeEnergyLostByBallToContactZoneArea = false;
normalizeEnergyLostByBallToContactZoneVolume = true;
normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion = false;
tryToSmoothTheDataAndReduceTheNumberOfPoints = false;
if normalizeEnergyLostByBallToContactZoneVolume==true
  if ~isempty(strfind(feaLabel, '617um'))
    almenStripThickness=617/1e6;
  elseif ~isempty(strfind(feaLabel, '779um'))
    almenStripThickness=779/1e6;
  else
    disp('Error: Unrecognized feaLabel, Update Code.');
    return;
  end
end

if (normalizeEnergyLostByBallToContactZoneVolume==true) && (normalizeEnergyLostByBallToContactZoneArea==true)
  normalizeEnergyLostByBallToContactZoneVolume=false;
end

%this is simple however, it is less accurate.
outputPath=inputPath;
fileNameSignature='HvsH';
fileExtSignature='csv';

if (normalizeEnergyLostByBallToContactZoneVolume==true) || (normalizeEnergyLostByBallToContactZoneArea==true) || (normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==true)
  if normalizeEnergyLostByBallToContactZoneVolume==true
    saveFileSig='VolS';
  end
  if (normalizeEnergyLostByBallToContactZoneArea==true)
    saveFileSig='AreaS';
  end
  else
    zoneSizeDataFilename='zone_95%NRG_Boundary.csv';
    zoneSize=csvread([inputPath,zoneSizeDataFilename], 1, 0);
    zoneSize_Col_Velocity=1;
    zoneSize_Col_ImpactNumber=2;
    zoneSize_Col_CumAbsNRG=3;
    zoneSize_Col_RadiusMm=4;
    zoneSize_Length=length(zoneSize(:,1));
    if (normalizeEnergyLostByBallToContactZoneVolume==true)
      saveFileSig='Vol';
    end
    if (normalizeEnergyLostByBallToContactZoneArea==true)
      saveFileSig='Area';
    end
    else
      saveFileSig='Ball';
  end
end

eLookupTableFilename = ['eLookUp',saveFileSig,'_',feaLabel];
files = dir(inputPath);\nnumberOfFiles=length(files);

stressTensorFileSuffix='SxSy';
if useMultiPathStressesInsteadOfCenterline==true
  mltPthMmntFileIndexImpactNum=1;
  mltPthMmntFileIndexVi=2;
  mltPthMmntFileIndexVr=3;
  mltPthMmntFileIndexE=4;
  mltPthMmntFileIndexEnergy=6;
  mltPthMmntFileIndexCumEnergy=7;
  mltPthMmntFileIndexStressDir=3;
  numberOfStressDirections=3;
  mltPthMmntFilename=['MultiPathMomentsInNewtons.csv'];
  mltPthMmntData=csvread([inputPath,mltPthMmntFilename], 1, 0);
  mltPthMmntNumCol=length(mltPthMmntData(1,:));
  % Get File Header %
  fileId = fopen([inputPath,mltPthMmntFilename]);
  if fileId==-1
    disp(['ERROR: file could not be opened: ',inputPath,mltPthMmntFilename]);
    return;
  end
  tline = fgetl(fileId);
  commas = strfind(tline, ',');
  for i=1:length(commas)
    if i==1
      mltPthMmntFileHeader{i}=tline(1:commas(i)-1);
    else
      mltPthMmntFileHeader{i}=tline(commas(i-1)+1:commas(i)-1);
    end
  end
  numberOfPaths=(mltPthMmntNumCol-mltPthMmntFileIndex1stM)/numberOfStressDirections;
  numberOfMomentCols=numberOfPaths*numberOfStressDirections;
  numberOfImpacts=length(mltPthMmntData(:,1));
  else
    arcHeightVsEabsFilename = ['ALookUp',saveFileSig,'_',feaLabel];
end
arcHeightVsEabs(1,1)=0;
arcHeightVsEabs(1,2)=0;
%% HvsH ennumerations - Start %%
Archeight=1;
Vin=2;
Vr=3;
e=4;
Ein=5;
Eabs=6;
%% HvsH ennumerations - End %%
end
eLookupVeloIndx=1;
eLookupEAbsIndx=2;
eLookupCNesIndx=3;

if useMultiPathStressesInsteadOfCenterline==false
k=1;
enumerations
totalNumberOfImpacts=0;
page=0;
for i=1:numberOfFiles
    if ~isempty(findstr(fileNameSignature, files(i).name))
        if ~isempty(findstr(fileExtSignature, files(i).name))
            page=page+1;
            disp(files(i).name);
            pathAndFilename=[inputPath,files(i).name];
            data = csvread(pathAndFilename);
            lengthOfDataFile=length(data(:,1));
            if ((normalizeEnergyLostByBallToContactZoneArea==true)||(normalizeEnergyLostByBallToContactZoneVolume==true))&&... (normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==false)
                for m=2:lengthOfDataFile
                    numData=m-1;
                    zoneSizeRow=0;
                    for n=1:zoneSize_Length
                        velZone=zoneSize(n,zoneSize_Col_Velocity);
                        velData=data(m,Vin);
                        numZone=zoneSize(n,zoneSize_Col_ImpactNumber);
                        if (velZone==velData)&&(numZone==numData) %check
                            zoneSizeRow=n;
                            break;
                        end
                    end
                    if zoneSizeRow~=0
                        radiusMeters=zoneSize(zoneSizeRow,zoneSize_Col_RadiusMm)/1000;
                        contactAreaMetersSquared=pi*radiusMeters^2;
                        if normalizeEnergyLostByBallToContactZoneVolume==true
                            data(m,Eabs)=data(m,Eabs)/(contactAreaMetersSquared*almenStripThickness);
                        else
                            data(m,Eabs)=data(m,Eabs)/contactAreaMetersSquared;
                        end
                        else
                            disp('ERROR: Zone Size Data not found. Check to see if all data is present in the file');
                            return;
                    end
                end
            else
                disp('ERROR: Zone Size Data not found. Check to see if all data is present in the file');
                return;
            end
            impactVelocity=data(2,Vin);
            for m=3:lengthOfDataFile
                impactVelocityOther=data(m,Vin);
                if (impactVelocity~=impactVelocityOther)
                    disp('Error in HvsH file!');
                    break;
                end
            end
        end
    end
end
end
arcHeightVsEabs(k,1)=data(j,Eabs);
arcHeightVsEabs(k,2)=data(j,Archeight);

eV(k-1)=impactVelocity;
eEabs(k-1)=data(j-1,Eabs);
eE(k-1)=data(j,e);
end
for j=1:lengthOfDataFile-1
eLookupTable(j,1,page)=impactVelocity;
eLookupTable(j,2,page)=data(j,Eabs);
eLookupTable(j,3,page)=data(j+1,e);
end
end
end
clear data;
clear lengthOfDataFile;
clear impactVelocity;
end
end
if useMultiPathStressesInsteadOfCenterline==true
if ((normalizeEnergyLostByBallToContactZoneArea==true)||(normalizeEnergyLostByBallToContactZoneVolume==true))&&...(normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==false)
for mi=1:numberOfImpacts
zoneSizeRow=0;
for zi=1:zoneSize_Length
velZone=zoneSize(zi,zoneSize_Col_Vel)
velData=mltPthMmntData(mi,mltPthMmntFileIndxVi)
numZone=zoneSize(zi,zoneSize_Col_ImpactNumber)
numData=mltPthMmntData(mi,mltPthMmntFileIndxImpactNum)
if (velZone==velData)&&(numZone==numData) %check
zoneSizeRow=zi;
break;
end
end
if zoneSizeRow~0
radiusMeters=zoneSize(zoneSizeRow,zoneSize_Col_RadiusMm)/1000;
contactAreaMetersSquared=pi*radiusMeters^2;
if normalizeEnergyLostByBallToContactZoneVolume==true
mltPthMmntData(mi,mltPthMmntFileIndxAbsEnergyCum)=...
mltPthMmntData(mi,mltPthMmntFileIndxAbsEnergyCum)/...(contactAreaMetersSquared*almenStripThickness);
else
mltPthMmntData(mi,mltPthMmntFileIndxAbsEnergyCum)=...
mltPthMmntData(mi,mltPthMmntFileIndxAbsEnergyCum)/...(contactAreaMetersSquared);
end
else
disp('ERROR: Zone Size Data not found. Check to see if all data is present in the file');
return;
end
end
if ((normalizeEnergyLostByBallToContactZoneArea==true)||(normalizeEnergyLostByBallToContactZoneVolume==true))&&...(normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==true)
for mi=1:numberOfImpacts
impactVelocity=mltPthMmntData(mi,mltPthMmntFileIndxVi)
radiusMeters=feaZoneRadius(impactVelocity)
contactAreaMetersSquared=pi*radiusMeters^2;
if normalizeEnergyLostByBallToContactZoneVolume==true
mltPthMmntData(mi,mltPthMmntFileIndxAbsEnergyCum)=...
mltPthMmntData(mi,mltPthMmntFileIndxAbsEnergyCum)/...(contactAreaMetersSquared*almenStripThickness);
else
mltPthMmntData(mi,mltPthMmntFileIndxAbsEnergyCum)=...
mltPthMmntData(mi,mltPthMmntFileIndxAbsEnergyCum)/...(contactAreaMetersSquared);
end
end
end
page=0;
for mi=1:numberOfImpacts
velData=mltPthMmntData(mi,mltPthMmntFileIndxVi)
numData=mltPthMmntData(mi,mltPthMmntFileIndxImpactNum)
absorbedEnergyBeforeImpact=0;
if numData>1
page=page+1;
else
absorbedEnergyBeforeImpact=mltPthMmntData(mi-1,mltPthMmntFileIndxAbsEnergyCum);
end
eLookupTable(numData,1,page)=velData;
eLookupTable(numData,2,page)=absorbedEnergyBeforeImpact;
eLookupTable(numData,3,page)=mltPthMmntData(mi,mltPthMmntFileIndxE);
%for graphing purposes:
eV(mi)=velData;
eAbs(mi)=absorbedEnergyBeforeImpact;
e(mi)=mltPthMmntData(mi,mltPthMmntFileIndxE);
end

%for graphing purposes:
eV(mi)=velData;
eAbs(mi)=absorbedEnergyBeforeImpact;
e(mi)=mltPthMmntData(mi,mltPthMmntFileIndxE);
end

% Centerline Arc height or multi path moment distribution

for i=1:eLookupPages
  testVel = eLookupTable(1,eLookupVeloIndx,i);
  if i==1
    eLookupMaxVel=testVel;
eLookupMaxVelPage=i;
  else
    if (testVel>eLookupMaxVel)
      eLookupMaxVel=testVel;
eLookupMaxVelPage=i;
    end
  end
end

for j=1:eLookupPages
  eLookupMinVel=eLookupMaxVel;
eLookupMinVelPage=eLookupMaxVelPage;
  for i=1:eLookupPages
    testVel = eLookupTable(1,eLookupVeloIndx,i);
    if (testVel<eLookupMinVel)&&(testVel>lastFoundMinVel)
      eLookupMinVel=testVel;
eLookupMinVelPage=i;
    end
  end
  sortedELookup(:,:,j)=eLookupTable(:,:,eLookupMinVelPage);
  lastFoundMinVel=eLookupMinVel;
end

clear eLookupTable;
eLookupTable=sortedELookup;
clear sortedELookup;
csvwrite([outputPath,eLookupTableFilename,'.csv'],eLookupTable);
save([outputPath,eLookupTableFilename,'.mat'], 'eLookupTable');
i=i+1;
tempMltPthMmntData(i,1)=avgE;
for k=2:numberOfMomentCols+1
    tempMltPthMmntData(i, k)=avgM(k-1);
end
end

clear mltPthMmntData;
arcHeightVsEabs = tempMltPthMmntData;
end
csvwrite([outputPath,mltPthMmntFilename,'Sorted','.csv'],mltPthMmntData(:,mltPthMmntFileIndxAbsEnergyCum:mltPthMmntNumCol));
else

    arcHeightVsEabs = sortrows(arcHeightVsEabs,1);
    if tryToSmoothTheDataAndReduceTheNumberOfPoints==true
        arcHeightVsEabsImpacts=length(arcHeightVsEabs(:,1));
        arcHeightVsEabsIncrement=arcHeightVsEabs[arcHeightVsEabsImpacts,1]/arcHeightVsEabsImpacts;
        tempArcHeightVsEabs(1,1)=0;
        tempArcHeightVsEabs(1,2)=0;
        i=1;
        lastE=0;
        while j <= arcHeightVsEabsImpacts
            sumE=0;
            sumA=0;
            num=0;
            sumEst=0;
            while sumEst < 3*arcHeightVsEabsIncrement
                j=j+1;
                if j==arcHeightVsEabsImpacts
                    break;
                end
                sumEst = sumEst + arcHeightVsEabs(j,1)-arcHeightVsEabs(j-1,1);
                sumE = sumE + arcHeightVsEabs(j,1);
                sumA = sumA + arcHeightVsEabs(j,2);
                num = num+1;
            end
            if (j <= arcHeightVsEabsImpacts)&&(num>0)
                avgA=sumA/num;
                avgE=sumE/num;
                i=i+1;
                tempArcHeightVsEabs(i,1)=avgE;
                tempArcHeightVsEabs(i,2)=avgA;
            end
        end
    end
end

%%% PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tempPlot = newplot;
plot3(eV, eEabs, eE, 'linestyle', 'none', 'marker', 'o','markersize',5);
xlabel('Velocity (m/s)');
if (normalizeEnergyLostByBallToContactZoneArea==true)&&(normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==true)
    ylabel('Absorbed Energy (J/m²) - Simple');
end
if (normalizeEnergyLostByBallToContactZoneArea==true)&&(normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==false)
    ylabel('Absorbed Energy (J/m²)');
end
if (normalizeEnergyLostByBallToContactZoneVolume==true)&&(normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==false)
    ylabel('Absorbed Energy (J/m³)');
end
if (normalizeEnergyLostByBallToContactZoneVolume==true)&&(normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==true)
    ylabel('Absorbed Energy (J/m³) - Simple');
end
if (normalizeEnergyLostByBallToContactZoneArea==false)&&(normalizeEnergyLostByBallToContactZoneVolume==false)
    ylabel('Absorbed Energy (J)');
end
saveas(tempPlot,[outputPath,'e3d_'.saveFileSig,'.fig']);
saveas(tempPlot,[outputPath,'e3d_'.saveFileSig,'.png']);

tempPlot = newplot;
hold on;
if useMultiPathStressesInsteadOfCenterline==false
    if tryToSmoothTheDataAndReduceTheNumberOfPoints==true
        yy=smooth([arcHeightVsEabs(:,1),arcHeightVsEabs(:,2)],50,'lowess');
        plot([arcHeightVsEabs(:,1),yy],g');
    end
end
plot(tempArcHeightVsEabs(:,1),tempArcHeightVsEabs(:,2),'linestyle', 'none', 'marker', 'o','markersize',8);
else
    plot(arcHeightVsEabs(:,1),arcHeightVsEabs(:,2) , 'linestyle', 'none', 'marker', 'x','markersize',5);
end
if (normalizeEnergyLostByBallToContactZoneArea==true) &&...
(normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==true)
    xlabel('Absorbed Energy (J/m^2) - Simple');
    end
if (normalizeEnergyLostByBallToContactZoneArea==true) &&...
(normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==false)
    xlabel('Absorbed Energy (J/m^2)');
    end
if (normalizeEnergyLostByBallToContactZoneVolume==true) &&...
(normalizeEnergyLostByBallToContactZoneAreaOrVolumeSimpleVersion==false)
    xlabel('Absorbed Energy (J/m^3)');
    end
if (normalizeEnergyLostByBallToContactZoneVolume==true) &&...
(normalizeEnergyLostByBallToContactZoneArea==false)
    xlabel('Absorbed Energy (J/m^3) - Simple');
    end
if (normalizeEnergyLostByBallToContactZoneArea==false) &&...
(normalizeEnergyLostByBallToContactZoneVolume==false)
    xlabel('Absorbed Energy (J)');
    end
ylabel('Arc Height (micron)');
    if tryToSmoothTheDataAndReduceTheNumberOfPoints==true
        clear arcHeightVsEabs;
        arcHeightVsEabs=tempArcHeightVsEabs;
    end
    save([outputPath,arcHeightVsEabsFilename,'.mat'], 'arcHeightVsEabs');
    csvwrite([outputPath,arcHeightVsEabsFilename,'.csv'],arcHeightVsEabs);
end

disp('Finished');

function contactRadius = feaZoneRadius(impactVelocity)
%this is based on pwrlw model for both thicknesses R^2=0.97
contactRadius=(-2*impactVelocity^2+1.7*impactVelocity)/1000;
tresult is in mm;

function isMakeAnsysInitialStressFile(saveFilePath,stressDistrib,useOnlyLengthStressesInAnsysFEA)
% if lenNumOfElem~=widNumOfNodes*lenToWidRatio-1
%     disp('Error!!  ! ! !   ! !  ! !  !  !  !  !  !
%     return;
% end
% if widNumOfElem~=widNumOfNodes-1
%     disp('Error!!  ! ! !   ! !  ! !  !  !  !  !  !
%     return;
% end
if debug==true
    for i=1:widNumOfElem %incremented along y-axis
        for j=1:lenNumOfElem%incremented along x-axis  %width should be in inner loop. width is smaller than length.
            stressDistrib(i,j)= 40e6*2*(rand()-1/2);
        end
    end
end
for i=1:widNumOfElem %incremented along y-axis
    for j=1:lenNumOfElem%incremented along x-axis  %width should be in inner loop. width is smaller than length.
        stressDistrib(i,j)= 40e6*2*(rand()-1/2);
    end
end
if debug==true
    for i=1:widNumOfElem %incremented along y-axis
        for j=1:lenNumOfElem%incremented along x-axis  %width should be in inner loop. width is smaller than length.
            stressDistrib(i,j)= 40e6*2*(rand()-1/2);
        end
    end
end
valFclose=fclose(fileId);
if valFclose==-1
    disp('Error: Cannot close isDat.txt');
    return;
end
%%%%%%%%%%%%%%%%%%%
function isMakeAnsysInitialStressFileUsingMxAndMy(saveFilePath, stressDistribution, useOnlyLengthStressesInAnsysFEA)
    debug=false;
    if debug==true
        saveFilePath='C:\_D\IStress\';
        lenToWidRatio=5;
        widNumOfElem=1;
        lenNumOfElem=widNumOfElem*lenToWidRatio;
        stressDistribution=zeros(widNumOfElem,lenNumOfElem,2);
    end
    %#Pass to operating system: ! (bang), computer, dos, perl, unix, winopen,
    %#system
end

function isMakeAnsysInitialStressFileUsingMxAndMy(saveFilePath, stressDistribution, useOnlyLengthStressesInAnsysFEA)
% if lenNumOfElem ~= widNumOfNodes*lenToWidRatio+1
%     disp('Error!!  ! ! !   ! !  ! !  !  !  !  !  !');
%     return;
% end

% if widNumOfElem ~= widNumOfNodes-1
%     disp('Error!!  ! ! !   ! !  ! !  !  !  !  !  !');
%     return;
% end

if debug==true
    for i=1:widNumOfElem
        for j=1:lenNumOfElem %incremented along x-axis, i.e. columns %width should be in inner loop. width is smaller than
        %length.
            stressDistribution(i,j,stResStXIndx)= 40e6*2*(rand()-1/2);
            stressDistribution(i,j,stResStYIndx)= 40e6*2*(rand()-1/2);
        end
    end
    for i=1:widNumOfElem
        for j=1:lenNumOfElem %incremented along x-axis, i.e. columns %width should be in inner loop. width is smaller than
        %length.
            e=e+1;
            fprintf(fileId,'%s,%s
', 'eis',num2str(e));
            StXTop=num2str(stressDistribution(i,j,stResStXIndx));
            StXTot=num2str(stressDistribution(i,j,stResStYIndx));
            StXBot=num2str(-1*stressDistribution(i,j,stResStXIndx));
            StYBot=num2str(-1*stressDistribution(i,j,stResStYIndx));
            if useOnlyLengthStressesInAnsysFEA==true
                if lenNumOfElem >= widNumOfElem
                    fprintf(fileId,'%s,%s,%s,%s,%s,%s
', StXTop,'0','0','0','0','0');
                    fprintf(fileId,'%s,%s,%s,%s,%s,%s
', '0','0','0','0','0','0');
                    fprintf(fileId,'%s,%s,%s,%s,%s,%s
', StXBot,'0','0','0','0','0');
                else
                    fprintf(fileId,'%s,%s,%s,%s,%s,%s
', '0',StXTop,'0','0','0','0');
                    fprintf(fileId,'%s,%s,%s,%s,%s,%s
', '0','0','0','0','0','0');
                    fprintf(fileId,'%s,%s,%s,%s,%s,%s
', '0',StXBot,'0','0','0','0');
                end
            else
                fprintf(fileId,'%s,%s,%s,%s,%s,%s
', StXTop,StYTop,'0','0','0','0');
                fprintf(fileId,'%s,%s,%s,%s,%s,%s
', '0','0','0','0','0','0');
                fprintf(fileId,'%s,%s,%s,%s,%s,%s
', StXBot,StYBot,'0','0','0','0');
            end
        end
    end
valFclose=fclose(fileId);
if valFclose==-1
    disp('Error: Cannot close isDat.txt');
    return;
end

%%%%%%%%%%%%%%%%%%

function midNode = isMesher(saveFilePath, strip617umSwitch, stressDistrib, elementWidth)

    clc;
    clear all;
    debug=false;

    if debug==true
        len=0.0635;
        wid=0.0127;
        saveFilePath='C:\_D\ISTress\';
        lenNumOfElements=10;
        lenToWidRatio=5;
        elementWidth=1;
        strip617umSwitch=true;
    end
    lenNumOfNodes=10;

    lenNumOfElements=length(stressDistrib(1,:));
    widNumOfElements=length(stressDistrib(:,1));
    [widNumOfElements,lenNumOfElements, stressDirections] = size(stressDistrib);
    if stressDirections==1
        temp=widNumOfElements;
        widNumOfElements=lenNumOfElements;
        lenNumOfElements=temp;
    end
widNumOfNodes=widNumOfElements+1;
% lenNumOfElements=widNumOfElements*lenToWidRatio;
lenNumOfNodes=lenNumOfElements+1;
elementLength=elementWidth;

fileId=fopen([saveFilePath,'isThickness.txt'],'w');
if fileId==-1
  disp('ERROR: isMesher005.m -- cannot open isThickness.txt');
end
if strip617umSwitch==true
  fprintf(fileId, '%s
','R,1,617e-6, , , , , ,');
else
  fprintf(fileId, '%s
','R,1,779e-6, , , , , ,');
end
if fclose(fileId)==-1
  disp('ERROR: isMesher005.m -- Cannot close isThickness.txt');
end

fileId=fopen([saveFilePath,'ismesh.txt'],'w');
if fileId==-1
  disp('ERROR: isMesher005.m -- cannot open ismesh.txt');
end

% make nodes
nodeNumber=0;
for j=1:widNumOfNodes %incremented along y-axis
  for i=1:lenNumOfNodes%incremented along x-axis  %width should be in inner loop. width is smaller than length.
    nodeNumber=nodeNumber+1;
    if ((i-1)==round(lenNumOfNodes/2))&&((j-1)==round(widNumOfNodes/2)) %output Node that is closest to centre
      midNode=nodeNumber;
    end
    fprintf(fileId,'%s,%s,%s,%s
', 'n',num2str(nodeNumber), num2str(x), num2str(y));
  end
end
for j=1:widNumOfElements
  for i=1:lenNumOfElements %incremented along x-axis, i.e. columns  %width should be in inner loop. width is smaller than
                             %length.
    fprintf(fileId,'%s,%s,%s,%s,%s
', 'e', num2str(nodeNumberFromIJ(i,j,lenNumOfNodes)),...
               num2str(nodeNumberFromIJ(i+1,j,lenNumOfNodes)),...
               num2str(nodeNumberFromIJ(i+1,j+1,lenNumOfNodes)),...
               num2str(nodeNumberFromIJ(i,j+1,lenNumOfNodes)));
  end
end
fprintf(fileId,'%s,%s,%s,%s
', 'd',num2str(nodeNumberFromIJ(1,1,lenNumOfNodes)),'UX','0');
fprintf(fileId,'%s,%s,%s,%s
', 'd',num2str(nodeNumberFromIJ(1,1,lenNumOfNodes)),'UY','0');
fprintf(fileId,'%s,%s,%s,%s
', 'd',num2str(nodeNumberFromIJ(1,1,lenNumOfNodes)),'UZ','0');
fprintf(fileId,'%s,%s,%s,%s
', 'd',num2str(nodeNumberFromIJ(1,widNumOfNodes,lenNumOfNodes)),'UZ','0');
fprintf(fileId,'%s,%s,%s,%s
', 'd',num2str(nodeNumberFromIJ(lenNumOfNodes,1,lenNumOfNodes)),'UZ','0');
fprintf(fileId,'%s,%s,%s,%s
', 'd',num2str(nodeNumberFromIJ(lenNumOfNodes,widNumOfNodes,lenNumOfNodes)),'UZ','0');
if fclose(fileId)==-1
  disp('ERROR: isMesher005.m -- Cannot close isMesh.txt');
end

function [nodeNumber] = nodeNumberFromIJ(i,j,lenNumOfNodesInDirection) %i,j,
nodeNumber=1+lenNumOfNodesInDirection*(j-1);

end
function [maxHeight, midNodeHeight] = readMaxAndMidNodeValue(path, fileName, midnode)

debug=false;
if debug=true
    path='C:\_D\IStress\';
    fileName='nodesUZ.lis';
    midnode=500;
end

fidLis = fopen([path,fileName]);
if fidLis==-1
    disp('ERROR: ',fileName,' could not be found. Did Ansys run?');
    return;
end

patternMax=' VALUE ';
patternMid=[' ',num2str(midnode),'] ');
patternLengthMax=length(patternMax);
patternLengthMid=length(patternMid);
tempStrMax='';
tempStrMid='';
foundMax=false;
foundMid=false;
while (0==feof(fidLis))
tline = fgetl(fidLis);
tlineLength = length(tline);
startOfStringPos=strfind(tline, patternMax);
if ~isempty(startOfStringPos)
    foundMax=true;
    for c=startOfStringPos(1)+patternLengthMax:tlineLength
        tempStrMax=[tempStrMax,tline(c)];
    end
end

startOfStringPos=strfind(tline, patternMid);
if (~isempty(startOfStringPos))&&(foundMid==false)
    foundMid=true;
    for c=startOfStringPos(1)+patternLengthMid:tlineLength
        tempStrMid=[tempStrMid,tline(c)];
    end
end
end

if debug=true
    disp('Max: ', tempStrMax);
    disp('Mid: ', tempStrMid);
end

maxHeight=str2num(tempStrMax);
midNodeHeight=str2num(tempStrMid);
fcloseVal=fclose(fidLis);
if fcloseVal==-1
    disp('Error: Cannot close ', fileName);
end

end

function [MxOut, MyOut]  = momentsPerMeterFromAbsAsFunctionOfRadialDistanceLKTB( Strip617umSwitch, bilinearMaterialModel,
frictionSwitch, useAbsNrgPer ,AbsE, radialDistanceMeters)

debug=false;
if debug=false
    % mlock;
else
    clear all;
    clc;
    debug=true;
end

fakeACenterLineDistribution=false; %for testing only

%%%%%%%%%%%%%%%%%% Persistant Variables for algorithm

if debug=true
    Strip617umSwitch=true;
    bilinearMaterialModel=false;
    AbsE=0.0000001;
    useAbsNrgPer=false;
    useAbsNrgPerContactZoneArea=false;
    frictionSwitch=false;
radiolDistanceMeters=1200/1000;
end
persistent bilinearMaterialModelOrig;  
persistent Strip617umSwitchOrig;  
persistent useAbsNrgPerOrig;  
persistent frictionSwitchOrig;  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Persistent Variables for algorithms that improve  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% speed  
persistent sections;  
 persistent maxEabs;  
persistent numberOfSections;  
persistent foundAbsNrg;  
 persistent feaLabel;  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Persistent Variables for algorithms that improve  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% speed  
persistent numberOfStressDirections;  
persistent distanceBetweenPathsMeters;  
persistent mltPthMmntData;  
persistent numberOfImpacts;  
persistent pathsRadialDistance;  
 persistent sectionUpperAbsNrgIndx;  
persistent sectionLowerAbsNrgIndx;  
persistent sectionUpperSecNumIndx;  
persistent sectionLowerSecNumIndx;  
%loadNewLookUpTable=false;  
if (~isempty(bilinearMaterialModelOrig))  
  if (bilinearMaterialModelOrig ~= bilinearMaterialModel) || ...  
    (Strip617umSwitchOrig ~= Strip617umSwitch) ||...  
      (useAbsNrgPerOrig ~= useAbsNrgPer) |...  
        (frictionSwitch Orig == frictionSwitch)  
    disp('Loading new Look-up Table in momentsPerMeterFromAbseAsFunctionOfRaidalDistanceLKTB001.m - switched material  
model');  
%loadNewLookUpTable==true;  
  mltPthMmntData=[];  
end  
end  
if isempty(mltPthMmntData) || (loadNewLookUpTable==true)  
  if (Strip617umSwitch==true) && (bilinearMaterialModel==true) && (frictionSwitch==false)  
    feaLabel='Biln617um';  
  elseif (Strip617umSwitch==true) && (bilinearMaterialModel==false) && (frictionSwitch==false)  
    feaLabel='PwrLw617um';  
  elseif (Strip617umSwitch==false) && (bilinearMaterialModel==true) && (frictionSwitch==false)  
    feaLabel='Biln779um';  
  elseif (Strip617umSwitch==false) && (bilinearMaterialModel==false) && (frictionSwitch==false)  
    feaLabel='PwrLw779um';  
  elseif (Strip617umSwitch==true) && (bilinearMaterialModel==true) && (frictionSwitch==true)  
    feaLabel='Biln617umF';  
  elseif (Strip617umSwitch==true) && (bilinearMaterialModel==false) && (frictionSwitch==true)  
    feaLabel='PwrLw617umF';  
  elseif (Strip617umSwitch==false) && (bilinearMaterialModel==true) && (frictionSwitch==true)  
    feaLabel='Biln779umF';  
  elseif (Strip617umSwitch==false) && (bilinearMaterialModel==false) && (frictionSwitch==true)  
    feaLabel='PwrLw779umF';  
  else  
    disp('ERROR in momentsPerMeterFromAbseAsFunctionOfRaidalDistanceLKTB001.m');  
    return;  
  end  
nolinearMaterialModelOrig = bilinearMaterialModel;  
 Strip617umSwitchOrig = Strip617umSwitch;  
 frictionSwitchOrig = frictionSwitch;  
 useAbsNrgPerOrig = useAbsNrgPer;  
 if (useAbsNrgPer==useAbsNrgPerContactZoneArea) || (useAbsNrgPer==useAbsNrgPerContactZoneAreaAndRealNrgDist)  
  tarcHeightVsEabsFilename=['ALookUpArea_','.feaLabel'];  
  mltPthMmntFilename=['FeaLabel', '_multiPathMomentsInNewtons.csvSorted.csv'];  
elseif useAbsNrgPer==useAbsNrgPerBall  
  tarcHeightVsEabsFilename=['ALookUp_','.feaLabel'];  
  mltPthMmntFilename=['FeaLabel', '_multiPathMomentsInNewtons.csvSorted.csv'];  
end  
 disp(mltPthMmntFilename);  
 mltPthMmntData = csvread([mltPthMmntFilename], 0, 0);  
 if isempty(mltPthMmntData)  
    disp('ERROR: Could not load material model files: ', mltPthMmntFilename);  
    return;  
end  
numberOfStressDirections=3;  
mltPthMmntFileIndexAtM=2;
mltPthMmntNumCol = length(mltPthMmntData(1,:));

numberOfPaths = (mltPthMmntNumCol - mltPthMmntFileIndx1stM) / numberOfStressDirections;
distanceBetweenPathsMeters = (1/1000) / numberOfPaths;

for r = 1:numberOfPaths
    pathsRadialDistance(r) = (r-1) * distanceBetweenPathsMeters;
end

numberOfImpacts = length(mltPthMmntData(:,1));

maxAbs = mltPthMmntData(numberOfImpacts,1);
absEInc = maxAbs / numberOfImpacts;
numberOfPointsInSection = 50;
numberOfSections = floor(numberOfImpacts / numberOfPointsInSection);
sectionLowerAbsNrgIndx = 1;
sectionUpperAbsNrgIndx = 2;
sectionLowerSecNumIndx = 3;
sectionUpperSecNumIndx = 4;

for i = 1:numberOfSections
    j = (i-1) * numberOfPointsInSection + 1;
    k = (i) * numberOfPointsInSection + 1;
    sections(j, sectionLowerAbsNrgIndx) = mltPthMmntData(j,1);
    sections(j, sectionUpperAbsNrgIndx) = mltPthMmntData(k,1);
    sections(i, sectionLowerSecNumIndx) = j;
    sections(i, sectionUpperSecNumIndx) = k;
end

sections(numberOfSections, sectionUpperSecNumIndx) = numberOfImpacts;
sections(numberOfSections, sectionUpperAbsNrgIndx) = mltPthMmntData(numberOfImpacts,1);
end

foundAbsE = false;

if absE < mltPthMmntData(1,1)
    absE = mltPthMmntData(1,1);
end

for sectionContainingAbsE = 1:numberOfSections
    if (absE >= sections(sectionContainingAbsE, sectionLowerAbsNrgIndx)) &&
        (absE <= sections(sectionContainingAbsE, sectionUpperAbsNrgIndx))
        break;
    end
end

% here the value of j gives the section containing the abs E

for i = sections(sectionContainingAbsE, sectionLowerSecNumIndx):sections(sectionContainingAbsE, sectionUpperSecNumIndx) - 1
    if absE == mltPthMmntData(i,1)
        absHeight = mltPthMmntData(i,2);
        subSecNumUpper = i;
        subSecNumLower = i;
        foundAbsE = true;
        break;
    end
    if absE == mltPthMmntData(i+1,1)
        absHeight = mltPthMmntData(i+1,2);
        subSecNumUpper = i+1;
        subSecNumLower = i+1;
        foundAbsE = true;
        break;
    end
    if (AbsE < mltPthMmntData(i+1,1)) && (AbsE > mltPthMmntData(i,1))
        subSecNumUpper = i+1;
        subSecNumLower = i;
        foundAbsE = true;
        break;
    end
    if (foundAbsE == false) && (i == (numberOfImpacts-1))
        subSecNumUpper = numberOfImpacts;
        subSecNumLower = numberOfImpacts;
        absHeight = mltPthMmntData(numberOfImpacts,2);
        foundAbsE = true;
        break;
    end
end

else
    absHeight = mltPthMmntData(numberOfImpacts,2);
    foundAbsE = true;
    subSecNumUpper = numberOfImpacts;
    subSecNumLower = numberOfImpacts;
end

if foundAbsE == false
    error('Error: Cannot find AbsE within section that should contain it.');
    return;
end

hiAbsE = mltPthMmntData(subSecNumUpper,1);
loAbsE=mltPthMmntData(subSecNumLower, 1);

[pathNumberLower, pathNumberUpper]=findBoundingIndicesForValueInArrayOfAscendingValues(radialDistanceMeters, pathsRadialDistance);

if fakeACenterLineDistribution==true  %for testing only
    pathNumberLower=1;
    pathNumberUpper=1;
end

loR=pathsRadialDistance(pathNumberLower);
hiR=pathsRadialDistance(pathNumberUpper);
pathNumberLower=pathNumberLower+1; %correct path number so that they match mltPthMmntData
pathNumberUpper=pathNumberUpper+1; %correct path number so that they match mltPthMmntData
hiRMxPath=pathNumberUpper;
loRMxPath=pathNumberLower;
hiRMyPath=pathNumberUpper;
loRMyPath=pathNumberLower;

Mx=0; %this is needed so that the output variables are written to properly.
My=0; %this is needed so that the output variables are written to properly.
if subSecNumLower==subSecNumUpper
    if pathNumberLower==pathNumberUpper
        Mx=mltPthMmntData(subSecNumUpper, hiRMxPath);
        My=mltPthMmntData(subSecNumUpper, hiRMyPath);
    else
        MxloR=mltPthMmntData(subSecNumUpper, loRMxPath);
        hiRMxPath=mltPthMmntData(subSecNumUpper, hiRMxPath);
        Mx=interpolate(loR, radialDistanceMeters, hiR, MxloR, hiRMxPath);
        MyloR=mltPthMmntData(subSecNumUpper, loRMyPath);
        hiRMyPath=mltPthMmntData(subSecNumUpper, hiRMyPath);
        My=interpolate(loR, radialDistanceMeters, hiR, MyloR, hiRMyPath);
    end
else
    hiMeioR=mltPthMmntData(subSecNumUpper, hiRMxPath);
    loMeioR=mltPthMmntData(subSecNumUpper, loRMxPath);
    Mx=interpolate(loAbsE, AbsE, hiAbsE, loMeioR, hiMeioR);
    hiMyloR=mltPthMmntData(subSecNumUpper, hiRMyPath);
    loMyloR=mltPthMmntData(subSecNumUpper, loRMyPath);
    Mu=interpolate(loAbsE, AbsE, hiAbsE, loMyloR, hiMyloR);
    end
end

else
    foundAbsE=true;
    Mx=0;
    My=0;
end

if debug==true
    disp(['Mx = ', num2str(Mx)]);
    disp(['My = ', num2str(My)]);
end
MxOut=Mx;
MyOut=My;
if fakeACenterLineDistribution==true  %for testing only
    MxOut=Mx;
    MyOut=My;
end
end  %end of main function

%%% other functions %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);

function [iY] = interpolate(xL,x,xU,iL,iU)
    iY=iU+(x-xU)*(iL-iU)/(xL-xU);
function [lowerIndex, upperIndex] = findBoundingIndicesForValueInArrayOfAscendingValues(value, array)
    foundIndices=false;
    minimumValue=array(1);
    if value < minimumValue
        disp(["ERROR: value < ", num2str(minimumValue)]);
        return;
    end
    maximumIndex=length(array);
    if value > array(maximumIndex)
        upperIndex=maximumIndex;
        lowerIndex=maximumIndex;
        foundIndices=true;
    else
        for i=1:maximumIndex-1
            if value==array(i)
                upperIndex=i;
                lowerIndex=i;
                foundIndices=true;
                break;
            end
            if value==array(i+1)
                upperIndex=i+1;
                lowerIndex=i+1;
                foundIndices=true;
                break;
            end
            if (value>array(i))&&(value<array(i+1))
                upperIndex=i+1;
                lowerIndex=i;
                foundIndices=true;
                break;
            end
        end
        if foundIndices==false
            disp('ERROR: foundIndices==false');
            return;
        end
    end
end

7.2.9. Moment distribution from stresses

Mathworks Inc. Matlab v7 Code

%% Relating Almen intensity to residual stresses induced by shot peening.pdf
clc;
clear;
outputPath='C:\_D\PwrLw779um\MomentDistribution\';
j=0;
for i=1:17
    j=j+1;
    path{i}=['C:\_D\PwrLw779um\',num2str(i),'\'];
end
unitSystemLTM='mm s g';
filenamePrefix='PwrLwThin';
filenamePrefix='PwrLw779um';
filenamePrefix='BiLn617um';
filenamePrefix='BiLn779um';
filenamePrefix='PwrLw617um';
filenamePrefix='PwrLw779um';
filenamePrefix='PwrLw779umUy';
filenamePrefix='PwrLw617umUy';
unitSystemLTM='m s Kg';
mediaMass=0.001031; %Kg
E2=71.7E9; %modulus of elasticity (Pa)
distanceOverWhichArcHeightIsMeasured=6.35E-2; %m
distanceOverWhichArcHeightIsMeasuredLongitudinal=0.5*(25.4/1000); %m
stripWidth=0.75*(25.4/1000);
stripLength=3*(25.4/1000);
numberOfImpacts=100; %100
numberOfPaths=25;
finiteElementModelLengthInMeters=1/1000;
pathDistanceIncMilliMeter=((finiteElementModelLengthInMeters/numberOfPaths)*1000);

stressTensorFileSuffix='SxSy';
stressTensorFileIndxPosition=1;
stressTensorFileIndxSx=2;
stressTensorFileIndxSy=3;
stressTensorFileIndxSz=4;
stressTensorFileIndxSxy=5;

if ~isempty(strfind(filenamePrefix, 'PwrLwThin'))
    feaLabel='PwrLwThin';
elseif ~isempty(strfind(filenamePrefix, 'ISOEPP'))
    feaLabel='BiLn617um';
else
    feaLabel=filenamePrefix;
end

filenamePrefix='rising';
suffixMotion = 'bllmnt.csv';

fileIdA=fopen([outputPath, feaLabel, '_multiPathArcHeightsMicron.csv'],'w');
if fileIdA==-1
    disp('Error creating output file');
    return;
end
fprintf(fileIdA,'%s,', 'Impact#');
fprintf(fileIdA,'%s,', 'vi');
fprintf(fileIdA,'%s,', 'vr');
fprintf(fileIdA,'%s,', 'Ei');
fprintf(fileIdA,'%s,', 'Eabs');
fprintf(fileIdA,'%s,', 'EabsCum');
for i=1:numberOfPaths+1
    fprintf(fileIdA,'%s,', ['Ax@d=',num2str(pathDistanceIncMilliMeter*(i-1)),'mm']);
    fprintf(fileIdA,'%s,', ['Ay@d=',num2str(pathDistanceIncMilliMeter*(i-1)),'mm']);
    fprintf(fileIdA,'%s,', ['Axy@d=',num2str(pathDistanceIncMilliMeter*(i-1)),'mm']);
end
fprintf(fileIdA, '\n');

fileIdM=fopen([outputPath,feaLabel, '_multiPathMomentsInNewtons.csv'],'w');
if fileIdM==-1
    disp('Error creating output file');
    return;
end
fprintf(fileIdM,'%s,', 'Impact#');
fprintf(fileIdM,'%s,', 'vi');
fprintf(fileIdM,'%s,', 'vr');
fprintf(fileIdM,'%s,', 'Ei');
fprintf(fileIdM,'%s,', 'Eabs');
fprintf(fileIdM,'%s,', 'EabsCum');
for i=1:numberOfPaths+1
    fprintf(fileIdM,'%s,', ['Mx@d=',num2str(pathDistanceIncMilliMeter*(i-1)),'mm']);
    fprintf(fileIdM,'%s,', ['My@d=',num2str(pathDistanceIncMilliMeter*(i-1)),'mm']);
    fprintf(fileIdM,'%s,', ['Mxy@d=',num2str(pathDistanceIncMilliMeter*(i-1)),'mm']);
end
fprintf(fileIdM, '\n');

%sxLegend(1) = {'sdlfkjsf'}
%sxLegend(2) = {'sdlfsdfkjsf'}
numberOfDirectories=length(path);
arcHeightsX(1)=0;
for directoryNumber=1:numberOfDirectories
    for impactNumber=1:numberOfImpacts
        for pathNumber=0:numberOfPaths
            %filename=[num2str(velocities(velocityIndex),['%3.',numberOfDigitsInVelocityNumber,'f']),prefix,'_',num2str(impactNumber),'_',stressTensorFileSuffix];
            %filenameSurf=[num2str(velocities(velocityIndex),['%3.',numberOfDigitsInVelocityNumber,'f']),prefix,'_',num2str(impactNumber),'_',suffixSurf];
        end
    end
end
%filename=[velocities, prefix, '_', num2str(impactNumber), '_', stressTensorFileSuffix];
filenameStress=[filenamePrefix, '_', num2str(impactNumber), '_', stressTensorFileSuffix, '_', num2str(pathNumber), '.csv'];
disp([path{directoryNumber}, filenameStress]);
data=(csvread([path{directoryNumber}, filenameStress], 1, 0));

numberOfDataPoints=length(data);
if (strcmp(unitSystemLTM, 'mm s g'))
    for i=1:numberOfDataPoints % convert to 'm s kg' unit system
        data(i,stressTensorFileIndxPosition)=data(i,stressTensorFileIndxPosition)/1000.00;
    end
end
topPosition=double(data(1,stressTensorFileIndxPosition));
bottomPosition=double(data(numberOfDataPoints,stressTensorFileIndxPosition));
almenStripThickness=double(bottomPosition-topPosition);

for i=1:numberOfDataPoints
    zAxis(i)=double(data(i,stressTensorFileIndxPosition))-double((almenStripThickness/2.0)); %reset datum to neutral axis
end
MxPerMeter=0;
MyPerMeter=0;
MxyPerMeter=0;
r=pathNumber*pathDistanceIncMilliMeter/1000;
dz=almenStripThickness/(double(numberOfDataPoints-1));

for i=1:numberOfDataPoints %double ofDataPoints
    MxPerMeter=MxPerMeter+double(data(i,stressTensorFileIndxSx)*dz*zAxis(i));
    MyPerMeter=MyPerMeter+double(data(i,stressTensorFileIndxSy)*dz*zAxis(i));
    MxyPerMeter=MxyPerMeter+double(data(i,stressTensorFileIndxSxy)*dz*zAxis(i));
end

arcHeightX=(3*MxPerMeter*distanceOverWhichArcHeightIsMeasured^2)/(2*E2*almenStripThickness^3);
arcHeightY=(3*MyPerMeter*distanceOverWhichArcHeightIsMeasured^2)/(2*E2*almenStripThickness^3);
arcHeightXY=(3*MxyPerMeter*distanceOverWhichArcHeightIsMeasured^2)/(2*E2*almenStripThickness^3);

fileIdone=fopen([path{directoryNumber},filenameStress,'_StArcHght_',num2str(pathNumber),'.txt'],'w');
fprintf(fileIdone,'%s
', [path{directoryNumber},filenameStress,'_StArcHght_']);
fprintf(fileIdone,'%s
', unitSystemLTM);
fprintf(fileIdone,'%s
', 'Saturation Arc Height Transverse (micron): ');
fprintf(fileIdone,'%s
', round(arcHeightX*1000*1000));
fprintf(fileIdone,'%s
', 'Saturation Arc Height Longitudinal (micron): ');
fprintf(fileIdone,'%s
', round(arcHeightY*1000*1000));
fprintf(fileIdone,'%s
', 'Saturation Arc Height Both (micron): ');
fprintf(fileIdone,'%s
', round(arcHeightXY*1000*1000));
fclose(fileIdone);

arcHeightsX(impactNumber,pathNumber+1,directoryNumber)=round(arcHeightX*1000*1000);
arckeyHeightsY(impactNumber,pathNumber+1,directoryNumber)=round(arcHeightY*1000*1000);
arcHeightsXY(impactNumber,pathNumber+1,directoryNumber)=round(arcHeightXY*1000*1000);
momentsX(impactNumber,pathNumber+1,directoryNumber)=MxPerMeter;
momentsY(impactNumber,pathNumber+1,directoryNumber)=MyPerMeter;
momentsXY(impactNumber,pathNumber+1,directoryNumber)=MxyPerMeter;
end
filenameMotion=[filenamePrefix,'_',suffixMotion];
dataMotion=(csvread([path{directoryNumber},filenameMotion], 1,0));
for i=1:length(dataMotion)-1
    if dataMotion(i+1,1)<dataMotion(i,1)
        break;
    end
end
dataMotion=dataMotion(1:i,:);
numberOfCurvesPlusOne=length(dataMotion(1,1));

j=0;
rebound=0;
vIncident(1,directoryNumber)=0;
vRebound(1,directoryNumber)=0;
for i=2:length(dataMotion(1,:))
    if rebound==0
        if (dataMotion(i-1,4)==0)&&(dataMotion(i,4)>0)&&(dataMotion(i,5)~=dataMotion(i-1,5))
            rebound=1;
            j=j+1;
vIncident(j,directoryNumber)=dataMotion[i-1,3];
        end
    else
        if (dataMotion(i,4)==0)&&(dataMotion(i-1,4)~=0)
            rebound=0;
vRebound(j,directoryNumber)=dataMotion(i,3);
        end
    end
end
%incidentEnergyCum = zeros(length(vIncident),1);
%reboundEnergyCum = zeros(length(vIncident),1);
%ballsAbsorbedEnergyCum = zeros(length(vIncident),1);
for i = 1:length(vIncident)
e(i,directoryNumber)=vRebound(i,directoryNumber)/vIncident(i,directoryNumber);
incidentEnergy(i)=0.5*mediaMass*vIncident(i,directoryNumber)^2;
reboundEnergy(i)=0.5*mediaMass*vRebound(i,directoryNumber)^2;
absorbedEnergy(i,directoryNumber)=incidentEnergy(i)-reboundEnergy(i);
if i==1
    incidentEnergyCum(1,directoryNumber)=incidentEnergy(i);
    reboundEnergyCum(1,directoryNumber)=reboundEnergy(i);
    cumulativeAbsorbedEnergy=absorbedEnergy(i,directoryNumber);
else
    incidentEnergyCum(1,directoryNumber)=incidentEnergy(i)+incidentEnergyCum(1,directoryNumber);
    reboundEnergyCum(1,directoryNumber)=reboundEnergy(i)+reboundEnergyCum(1,directoryNumber);
    cumulativeAbsorbedEnergy=absorbedEnergy(i,directoryNumber)+cumulativeAbsorbedEnergy;
end
end

for j=1:1:numberOfImpacts
    fprintf(fileIdA,'%s,', num2str(j));
    fprintf(fileIdA,'%s,', num2str(vIncident(j,directoryNumber)));
    fprintf(fileIdA,'%s,', num2str(vRebound(j,directoryNumber)));
    fprintf(fileIdA,'%s,', num2str(e(j,directoryNumber)));
    fprintf(fileIdA,'%s,', num2str(incidentEnergyCum(j,directoryNumber)));
    fprintf(fileIdA,'%s,', num2str(absorbedEnergy(j,directoryNumber)));
    fprintf(fileIdA,'%s,', num2str(cumulativeAbsorbedEnergy)));
    for i=1:numberOfPaths+1
        fprintf(fileIdA,'%s,', num2str(arcHeightsX(j,i,directoryNumber)));
        fprintf(fileIdA,'%s,', num2str(arcHeightsY(j,i,directoryNumber)));
        fprintf(fileIdA,'%s,', num2str(arcHeightsXY(j,i,directoryNumber)));
    end
    fprintf(fileIdA, '\n');
end

for j=1:1:numberOfImpacts
    fprintf(fileIdM,'%s,', num2str(j));
    fprintf(fileIdM,'%s,', num2str(vIncident(j,directoryNumber)));
    fprintf(fileIdM,'%s,', num2str(vRebound(j,directoryNumber)));
    fprintf(fileIdM,'%s,', num2str(e(j,directoryNumber)));
    fprintf(fileIdM,'%s,', num2str(incidentEnergyCum(j,directoryNumber)));
    fprintf(fileIdM,'%s,', num2str(absorbedEnergy(j,directoryNumber)));
    fprintf(fileIdM,'%s,', num2str(cumulativeAbsorbedEnergy)));
    for i=1:numberOfPaths+1
        fprintf(fileIdM,'%s,', num2str(momentsX(j,i,directoryNumber)));
        fprintf(fileIdM,'%s,', num2str(momentsY(j,i,directoryNumber)));
        fprintf(fileIdM,'%s,', num2str(momentsXY(j,i,directoryNumber)));
    end
    fprintf(fileIdM, '\n');
end
fprintf(fidM, '\n');
end
fclose(fidA);
fclose(fidM);

fprintf(fileIdM, '
');
end
fclose(fileIdM);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c=0;
if numberOfImpacts>numberOfPaths
    numberOfMarkers=numberOfImpacts;
else
    numberOfMarkers=numberOfPaths;
end
for j=1:ceil(numberOfMarkers/17)
    for i=1:17
        c=c+1;
        if i==1 plotStylesMarkers{c}='s';
        elseif i==2 plotStylesMarkers{c}='t';
        elseif i==3 plotStylesMarkers{c}='c';
        elseif i==4 plotStylesMarkers{c}='o';
        elseif i==5 plotStylesMarkers{c}='x';
        elseif i==6 plotStylesMarkers{c}='.';
        elseif i==7 plotStylesMarkers{c}='d';
        elseif i==8 plotStylesMarkers{c}='v';
        elseif i==9 plotStylesMarkers{c}='*';
        elseif i==10 plotStylesMarkers{c}='*';
        elseif i==11 plotStylesMarkers{c}='x';
        elseif i==12 plotStylesMarkers{c}='x';
        elseif i==13 plotStylesMarkers{c}='*';
        elseif i==14 plotStylesMarkers{c}='*';
        elseif i==15 plotStylesMarkers{c}='*';
        elseif i==16 plotStylesMarkers{c}='*';
        else plotStylesMarkers{c}='s';
        end
    end
    c=0;
    if numberOfImpacts>numberOfPaths
        numberOfColors=numberOfImpacts;
    else
        numberOfColors=numberOfPaths;
    end
    for j=1:ceil(numberOfColors/7)
        for i=1:7
            c=c+1;
            if i==1 colors{c}='r';
            elseif i==2 colors{c}='b';
            elseif i==3 colors{c}='g';
            elseif i==4 colors{c}='c';
            elseif i==5 colors{c}='m';
            elseif i==6 colors{c}='y';
            else colors{c}='k';
            end
        end
    end

% Determine Energy Distribution from archeight vs ball absorbed energy.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j=1:ceil(numberOfMarkers/17)
    for i=1:17
        if i==1 plotStylesMarkers{c}='s';
        elseif i==2 plotStylesMarkers{c}='t';
        elseif i==3 plotStylesMarkers{c}='c';
        elseif i==4 plotStylesMarkers{c}='o';
        elseif i==5 plotStylesMarkers{c}='x';
        elseif i==6 plotStylesMarkers{c}='.';
        elseif i==7 plotStylesMarkers{c}='d';
        elseif i==8 plotStylesMarkers{c}='v';
        elseif i==9 plotStylesMarkers{c}='*';
        elseif i==10 plotStylesMarkers{c}='*';
        elseif i==11 plotStylesMarkers{c}='x';
        elseif i==12 plotStylesMarkers{c}='x';
        elseif i==13 plotStylesMarkers{c}='*';
        elseif i==14 plotStylesMarkers{c}='*';
        elseif i==15 plotStylesMarkers{c}='*';
        elseif i==16 plotStylesMarkers{c}='*';
        else plotStylesMarkers{c}='s';
        end
    end

    c=0;
    if numberOfImpacts>numberOfPaths
        numberOfColors=numberOfImpacts;
    else
        numberOfColors=numberOfPaths;
    end
    for j=1:ceil(numberOfColors/7)
        for i=1:7
            c=c+1;
            if i==1 colors{c}='r';
            elseif i==2 colors{c}='b';
            elseif i==3 colors{c}='g';
            elseif i==4 colors{c}='c';
            elseif i==5 colors{c}='m';
            elseif i==6 colors{c}='y';
            else colors{c}='k';
            end
        end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Determine Energy Distribution from archeight vs ball absorbed energy.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for j=1:ceil(numberOfMarkers/17)
    for i=1:17
        if i==1 plotStylesMarkers{c}='s';
        elseif i==2 plotStylesMarkers{c}='t';
        elseif i==3 plotStylesMarkers{c}='c';
        elseif i==4 plotStylesMarkers{c}='o';
        elseif i==5 plotStylesMarkers{c}='x';
        elseif i==6 plotStylesMarkers{c}='.';
        elseif i==7 plotStylesMarkers{c}='d';
        elseif i==8 plotStylesMarkers{c}='v';
        elseif i==9 plotStylesMarkers{c}='*';
        elseif i==10 plotStylesMarkers{c}='*';
        elseif i==11 plotStylesMarkers{c}='x';
        elseif i==12 plotStylesMarkers{c}='x';
        elseif i==13 plotStylesMarkers{c}='*';
        elseif i==14 plotStylesMarkers{c}='*';
        elseif i==15 plotStylesMarkers{c}='*';
        elseif i==16 plotStylesMarkers{c}='*';
        else plotStylesMarkers{c}='s';
        end
    end

    c=0;
    if numberOfImpacts>numberOfPaths
        numberOfColors=numberOfImpacts;
    else
        numberOfColors=numberOfPaths;
    end
    for j=1:ceil(numberOfColors/7)
        for i=1:7
            c=c+1;
            if i==1 colors{c}='r';
            elseif i==2 colors{c}='b';
            elseif i==3 colors{c}='g';
            elseif i==4 colors{c}='c';
            elseif i==5 colors{c}='m';
            elseif i==6 colors{c}='y';
            else colors{c}='k';
            end
        end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Determine Energy Distribution from archeight vs ball absorbed energy.
arcHeightsForAllPaths(k,2,pathCounter)=arcHeightsForPathSorted(k,2);
end
clear arcHeightsForPathSorted;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    plot(ballsAbsorbedEnergyCum(:,directoryNumber),arcHeightsX(:,1,directoryNumber),plotStylesMarkers{directoryNumber});
end
ylabel('Arc Height X Direction over ', num2str(distanceOverWhichArcHeightIsMeasured), ' m');
xlabel('Absorbed Energy (i.e. energy lost by ball)');
hold off;
saveas(tempPlot,[outputPath,'archeightsX','.fig'])
saveas(tempPlot,[outputPath,'archeightsX','.png'])
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    plot(ballsAbsorbedEnergyCum(:,directoryNumber),arcHeightsY(:,1,directoryNumber),plotStylesMarkers{directoryNumber});
end
ylabel('Arc Height Y Direction over ', num2str(distanceOverWhichArcHeightIsMeasured), ' m');
xlabel('Absorbed Energy (i.e. energy lost by ball)');
hold off;
saveas(tempPlot,[outputPath,'archeightsY','.fig'])
saveas(tempPlot,[outputPath,'archeightsY','.png'])
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    plot(ballsAbsorbedEnergyCum(:,directoryNumber),arcHeightsXY(:,1,directoryNumber),plotStylesMarkers{directoryNumber});
end
ylabel('Arc Height SXY over ', num2str(distanceOverWhichArcHeightIsMeasured), ' m');
xlabel('Absorbed Energy (i.e. energy lost by ball)');
hold off;
saveas(tempPlot,[outputPath,'archeightsXY','.fig'])
saveas(tempPlot,[outputPath,'archeightsXY','.png'])
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    plot(ballsAbsorbedEnergyCum(:,directoryNumber),momentsX(:,directoryNumber),plotStylesMarkers{directoryNumber});
end
ylabel('Radial Moment per Meter X-Direction');
xlabel('Absorbed Energy (i.e. energy lost by ball)');
hold off;
saveas(tempPlot,[outputPath,'momentsX','.fig'])
saveas(tempPlot,[outputPath,'momentsX','.png'])
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    plot(ballsAbsorbedEnergyCum(:,directoryNumber),momentsY(:,directoryNumber),plotStylesMarkers{directoryNumber});
end
ylabel('Radial Moment per Meter Y-Direction');
xlabel('Absorbed Energy (i.e. energy lost by ball)');
hold off;
saveas(tempPlot,[outputPath,'momentsY','.fig'])
saveas(tempPlot,[outputPath,'momentsY','.png'])
cla reset;
clear tempPlot;
tempPlot = newplot;
hold on;
for directoryNumber=1:numberOfDirectories
    plot(ballsAbsorbedEnergyCum(:,directoryNumber),momentsXY(:,directoryNumber),plotStylesMarkers{directoryNumber});
end
ylabel('Radial Moment per Meter XY-Direction');
xlabel('Absorbed Energy (i.e. energy lost by ball)');
hold off;
saveas(tempPlot,[outputPath,'momentsXY','.fig'])
saveas(tempPlot,[outputPath,'momentsXY','.png'])
7.2.10. Arc Height from centerline stresses

Mathworks Inc. Matlab v7 Code

%% Relating Almen intensity to residual stresses induced by shot peening.pdf
clc;
clear;

unitSystemLTM='mm s g';
unitSystemLTM='m s Kg';
mediaMass=0.001031; %Kg
E2=71.7E9; %modulus of elasticity (Pa)
distanceOverWhichArcHeightIsMeasured=6.35E-2; %(m) % 2.5"
distanceOverWhichArcHeightIsMeasuredLongidntal=0.5*(25.4/1000); %(m) % 0.5"
position=1;
sx=2;
velocityIndex=1;

numberOfImpacts=100;
path='C:\_D\PwrLw617umC\17\';
velocities(velocityIndex)=0.15;
numberOfDigitsInVelocityNumber='2';
prefix='mps';
suffix='SxCntrLn.csv';
suffixSurf='Srfc.csv';
useNames=true;
thing='ISOEPP';
thing='PwrLwThin';
thing='PwrLw779um';
thing='PwrLw779umUy';
thing='PwrLw779umUy';
thing='rising';
suffixMotion = 'blmtn.csv';

%sxLegend(1) = {'sdlfkjsf'}
%sxLegend(2) = {'sdlfsdfkjsf'}
arcHeights(1)=0;
for impactNumber=1:numberOfImpacts
    filename=[num2str(velocities(velocityIndex),['%3.',numberOfDigitsInVelocityNumber,'f']),prefix,'_',num2str(impactNumber),'_',suffix];
    filenameSurf=[num2str(velocities(velocityIndex),['%3.',numberOfDigitsInVelocityNumber,'f']),prefix,'_',num2str(impactNumber),'_',suffixSurf];
    if useNames==true
        filename=[thing,'_',num2str(impactNumber),'_',suffix];
        filenameSurf=[thing,'_',num2str(impactNumber),'_',suffixSurf];
    end
    disp(filename);
    data=(csvread([path,filename], 1,0));
dataSurf=(csvread([path,filenameSurf], 1,0));
    if impactNumber==1
        AllSurfDat(:,impactNumber)=dataSurf(:,1);
        AllAlmenSx(:,impactNumber)=data(:,1);
    end
    AllSurfDat(:,impactNumber+1)=dataSurf(:,2);
    AllAlmenSx(:,impactNumber+1)=data(:,2);
    numberOfDataPoints=length(data);
    if strcmp(unitSystemLTM, 'mm s g')
        for i=1:numberOfDataPoints % convert to 'm s kg' unit system
            data(i,position)=data(i,position)/1000.00;
        end
    end

if strcmp(unitSystemLTM, 'm s Kg')
    for i=1:length(data) % convert to 'm s kg' unit system
        data(i,position)=data(i,position)/1000.00;
    end
end
topPosition = double(data(1, position));
bottomPosition = double(data(numberOfDataPoints, position));
almenStripThickness = double(bottomPosition - topPosition);
dz = almenStripThickness / (double(numberOfDataPoints - 1));
dA = dz * stripWidth;
M = 0;
for i = 1: numberOfDataPoints
    data(i, position) = double(data(i, position)) - double((almenStripThickness / 2.0)); % reset datum to neutral axis
    M = double(data(i, sx) * dA * data(i, position) + M);
end
disp(M);
arcHeightTransverse = (3 * M * distanceOverWhichArcHeightIsMeasured^2) / (2 * stripWidth * E2 * almenStripThickness^3);
arcHeightLongitudinal = (3 * M * distanceOverWhichArcHeightIsMeasuredLongitudinal^2) / (2 * distanceOverWhichArcHeightIsMeasuredLongitudinal * E2 * almenStripThickness^3);
arHeight = arcHeightTransverse + arcHeightLongitudinal;
format short g;
disp('Unit System');
disp(unitSystemLTM);
disp('Transverse Arc Height: ');
disp(arcHeightTransverse * 1000 * 1000);
disp('Longitudinal Arc Height: ');
disp(arcHeightLongitudinal * 1000 * 1000);
disp('Both Arc Heights: ');
disp(arcHeight * 1000 * 1000);
disp('Units are in Microns');

%% OUTPUT TO ASCII FILE - BEGIN %%%
fileIdDone = fopen(fullfile([path, filename, '_Sat_Arc_Height.txt']), 'w');
fprintf(fileIdDone, '%s
', [path, filename]);
fprintf(fileIdDone, '%s

', unitSystemLTM);
fprintf(fileIdDone, '%s%d
', 'Saturation Arc Height Transverse (micron): ', round(arcHeightTransverse * 1000 * 1000));
fprintf(fileIdDone, '%s%d
', 'Saturation Arc Height Longitudinal (micron): ', round(arcHeightLongitudinal * 1000 * 1000));
fprintf(fileIdDone, '%s%d
', 'Saturation Arc Height Both (micron): ', round(arcHeight * 1000 * 1000));
fclose(fileIdDone);
%% OUTPUT TO ASCII FILE - END %%%
if (dataMotion(i,4)==0) && (dataMotion(i+1,4)>0) && (dataMotion(i+1,5)=dataMotion(i,5))
    rebound=1;
    vIncident(j)=dataMotion(i,3);
end
else
    if (dataMotion(i+1,4)==0) && (dataMotion(i,4)~=0)
        rebound=0;
        vRebound(j)=dataMotion(i+1,3);
    end
end
incidentEnergyCum = zeros(length(vIncident),1);
reboundEnergyCum = zeros(length(vIncident),1);
absorbedEnergyCum = zeros(length(vIncident),1);
for i = 1:length(vIncident)+1
    if i==1
        e(i)=0;
        incidentEnergy(i)=0;
        reboundEnergy(i)=0;
        absorbedEnergy(i)=0;
    else
        e(i)=vRebound(i-1)/vIncident(i-1);
        incidentEnergy(i)=0.5*mediaMass*vIncident(i-1)^2;
        reboundEnergy(i)=0.5*mediaMass*vRebound(i-1)^2;
        absorbedEnergy(i)=incidentEnergy(i)-reboundEnergy(i);
    end
    if i==1
        incidentEnergyCum(i)=incidentEnergy(i);
        reboundEnergyCum(i)=reboundEnergy(i);
        absorbedEnergyCum(i)=absorbedEnergy(i);
    else
        incidentEnergyCum(i)=incidentEnergyCum(i-1)+incidentEnergy(i);
        reboundEnergyCum(i)=reboundEnergyCum(i-1)+reboundEnergy(i);
        absorbedEnergyCum(i)=absorbedEnergyCum(i-1)+absorbedEnergy(i);
    end
end

%header = {'Arc Height', 'Incident Velocity', 'Incident E', 'Rebound Velocity'};
dlmwrite(path,'HvsN.csv',header,'-append');
tgraymon;

%dlmwrite([path, 'HvsN.csv'],outputData,'-append');
%plot(arcHeights);
dlmwrite([path, 'HvsN.csv'],arcHeights,'-append');
dataCols=6;
outputData=zeros(numberOfImpacts+1,dataCols);
for j=2:numberOfImpacts+1
    outputData(j,1)=arcHeights(j);
    outputData(j,2)=vIncident(j-1);
    outputData(j,3)=vRebound(j-1);
    outputData(j,4)=e(j);
    outputData(j,5)=incidentEnergyCum(j);
    outputData(j,6)=absorbedEnergyCum(j);
end
dlmwrite([path, 'HvsN.csv'],outputData,'-');

plot(incidentEnergyCum);
axis tight;
xlabel('Impact Number');
ylabel('Cumulative Incident Energy (J)');
title(unitSystemLTM);
saveas(gcf,[path,'ICEIN','.fig'],'fig');
saveas(gcf,[path,'ICEIN','.png'],'png');
dlmwrite([path, 'ICEIN.csv'],incidentEnergyCum);

plot(absorbedEnergyCum);
axis tight;
xlabel('Impact Number');
ylabel('Absorbed Energy (J)');
title(unitSystemLTM);
saveas(gcf,[path,'EAIN','.fig'],'fig');
saveas(gcf,[path,'EAIN','.png'],'png');
dlmwrite([path, 'EAIN.csv'],absorbedEnergyCum);

plot(incidentEnergyCum,absorbedEnergyCum);
axis tight;
xlabel('Cumulative Incident Energy (J)');
ylabel('Cumulative Absorbed Energy (J)');
title(unitSystemLTM);
saveas(gcf,[path,'EAICE','.fig'],'fig');
saveas(gcf,[path,'EAICE','.png'],'png');
dlmwrite([path, 'EAICE.csv'],absorbedEnergyCum);

plot(incidentEnergyCum,incidentEnergyCum(1:length(arcHeights)),arcHeights);
axis tight;

225
xlabel('Cumulative Incident Energy (J)');
ylabel(['Arc Height (Micron)']);
title(unitSystemLTM);
saveas(gcf, [path,'AHICE','.fig'], '.fig');
saveas(gcf, [path,'AHICE','.png'], '.png');

plot(absorbedEnergyCum(1:length(arcHeights)),arcHeights);
axis tight;
xlabel('Absorbed Energy (J)');
ylabel(['Arc Height (Micron)']);
title(unitSystemLTM);
saveas(gcf, [path,'AHAE','.fig'], '.fig');
saveas(gcf, [path,'AHAE','.png'], '.png');

for impactNumber=1:length(arcHeights)-1
    sxLegend(impactNumber) = {'N=',num2str(impactNumber), ' Vi=', num2str(vIncident(impactNumber)), ' A=',num2str(arcHeights(impactNumber+1)), ' e=',num2str(e(impactNumber+1),2)};
end

numberOfCurvesPlusOne=length(AllAlmenSx(1,:));
plot(AllAlmenSx(:,1),AllAlmenSx(:,2:numberOfCurvesPlusOne));
axis tight;
xlabel('Depth (m)');
ylabel(['Radial Stress (Pa)']);
title(unitSystemLTM);
legend(sxLegend,'Location','EastOutside');
saveas(gcf, [path,'ResSX','.fig'], '.fig');
saveas(gcf, [path,'ResSX','.png'], '.png');
dlmwrite([path, 'ResSX_Col1Depth.csv'],AllAlmenSx);

plot(AllSurfDat(1:50,1),AllSurfDat(1:50,2:numberOfCurvesPlusOne));
axis tight;
xlabel('Distance from impact centre (m)');
ylabel(['Displacement (m)']);
title(unitSystemLTM);
legend(sxLegend,'Location','EastOutside');
saveas(gcf, [path,'Surf','.fig'], '.fig');
saveas(gcf, [path,'Surf','.png'], '.png');
dlmwrite([path, 'Surf_Col1D.csv'],AllSurfDat);

beep;

7.2.11. Converting Ansys output files to comma separated values

files
clc;
clear all;
J=0;

for i=17:17
    %
    % path[i]=['C:\_D\PwrLw617um\', num2str(i), '\'];
    %
    % end
    J=J+1;
    path(i)=['C:\_D\BiLn617um\', num2str(i), '\'];
end

numberOfDirectories=length(path);
% path[i]=['C:\_D\BiLn617um\IS\'];

% numberOfDirectories=1;
for i=1:numberOfDirectories
    disp(['Converting... ',path(i)]);
    files = dir([path(i), '*.lis']);
    lastFileCounter=0;
    numberOfFiles=length(files);
    lastProgressUpdateTime = clock();
    programStartTime = clock();
    for j=1:numberOfFiles
        convertLisToCsv07(path(i),files(j).name);
        delete([path(i),files(j).name]);
        currentTime = clock();
        timeSinceLastUpdate=etime(currentTime, lastProgressUpdateTime);
        if timeSinceLastUpdate>progressInterval
            lastProgressUpdateTime = clock();
            disp(['... Converting... ',path(i)]);
        end
    end
end

226
if (timeSinceLastUpdate>10)
    lastProgressUpdateTime = clock();
    percentComplete=100*j/numberOfFiles;
    if percentComplete > 100
        percentComplete = 100;
    end
    elapsedTime = etime(currentTime, programStartTime);
    eta = ((100-percentComplete)*elapsedTime/percentComplete)/(60);
end
    if eta <0
        eta=0;
    end
    filesPerSec=(j-lastFileCounter)/timeSinceLastUpdate;
    disp([percentComplete,'% Complete', '   ', eta,' min left', ' ',elapsedTime,' min Elapsed', ' ', (elapsedTime+eta), ' Total Run Time in min', ' ',filesPerSec,' Files Converted Per Second']);
    disp(path{i},files(j).name);
    disp('');
    lastFileCounter=j;
end
clear files;

disp('Finished');

function convertLisToCsv(path,fileName)
    path='C:\_D\IStress\';
    fileName='nodesUZ.lis';
    fidLis = fopen([path,fileName]);
    outputFilename= strrep(fileName, '.lis', '.csv');
    fidOutput = fopen([path,outputFilename],'w');
    deleteMode = true;
endOfLine=false;
firstS=true;
skipLine=false;
lineNumber=0;
while (0==feof(fidLis))
    lineNumber=lineNumber+1;
    tline = fgetl(fidLis);
    tlineLength = length(tline);
    if tlineLength>=1
        skipLine=false;
    if tlineLength>=6
        possibleS=tline(6);
        if ((strcmp(possibleS,'S'))&&(strcmp(tline(5),' ')))
            deleteMode = false;
            if firstS=false
                skipLine=true;
            firstS=false;
        end
    end
    if skipLine==false
        possible1=tline(1);
        if strcmp(possible1,'1')
            deleteMode = true;
        end
    deleteMode -- false
    tline = strtrim(tline);
    tlineLength= length(tline);
    resetTempStrAndPrint=false;
    tempStr='';
    foundFieldSeparator=false;
    for c=1:tlineLength
        if c==tlineLength
            tempStr=[tempStr,tline(c)];
        resetTempStrAndPrint=true;
        tempStr='';
        if resetTempStrAndPrint==true
            resetTempStrAndPrint=false;
        foundFieldSeparator=false;
        fprintf(fidOutput, '%s,', tempStr);
    if c<tlineLength
        tempStr=[tempStr,tline(c)];
    resetTempStrAndPrint=true;
        tempStr='';
    end
end
7.2.12. Finite difference solution to beam bending problem

Clc;
clear all;

% INPUTS
N=3000;
L=0.0635; %m
v=0.33;
E=71.7e9; %Pa
t=617e-6; %m
M=1.935177; %Nm/m

%s=M*12;
dx=L/N;

% CALCULATED CONSTANTS
C=(12*(1-v)/(E*t^3));

% MEMORY
momGrid=zeros(N,1); %moments per unit length
dspGrid=zeros(N,1); %displacement
solGridA=zeros(N,N);
solGridB=zeros(N,1);

% INPUT ERROR CHECKING
N=ceil(N);
if N<5
disp('Need more than 5 nodes');
return;
end

% X = linsolve(A,B) solves the linear system A*X = B
for i=1:N
    momGrid(i)=M*(i/N)^2;
end

% for i=1:1 %forward difference equations
%     solGridB(i)=momGrid(i)*(-1)*C*dh^2;
%     solGridB(i+1)=0;
%     solGridDA(i,i+3)=1;
%     solGridDA(i,i+2)=4;
%     solGridDA(i,i+1)=-5;
%     solGridDA(i,i)=2;
% end

for i=2:N-1 %central difference equations
    solGridB(i)=momGrid(i)*(-1)*C*dh^2;
    solGridA(i,i+1)=1;
    solGridA(i,i)=-2;
    solGridA(i,i-1)=1;
end

% for i=N:N %Backward difference equations
%     solGridB(i)=momGrid(i)*(-1)*C*dh^2;
%     solGridB(i)=0;
%     solGridA(i,i)=2;
%     solGridA(i,i-1)=-5;
%     solGridA(i,i-2)=4;

fclose('all');
fclose(fidDis);
fclose(fidOutput);
disp(['Converted ',fileName, ' to ', outputFilename]);
7.2.13. **Ansys batch code for running multiple impact analysis**

Ansys 8 Code

Contents of Script.bat:
set ANSYS80_PRODUCT=structds
set ANS_CONSEC=YES
del d3dump*
del d3hsp*
"C:\Program Files\Ansys Inc\V80\ANSYS\bin\intel\ansys80" -b -i script.txt -o script.out.txt
/BATCH
!/Config,NPROC,2 !for SMP
!/Config,NRES,10000 !set results file max size

Contents of Script.txt:
AR20='PwrLwThin' !filename
AR22=-0.01 !impact velocity
AR23=0.0002 !time increment after restart
AR40='centerLinePathThinAvg'
AR41='surfacePath'
AR21='C:\_DOCS\PhD\THESIS\_Model\FE\BB' !30 character limit
AR24=100 !number of repeats
AR30=992 !almen strip node
AR31=105 !ball node
RESUME,AR20,db,AR21,0,0
/CWD,AR21
/Filename,AR20,1
/SOLU
EDRST,20, !output file frequency. higher number bigger file.
EDDUMP,0, ,7,
AR26=0
ESTART,1
AR26=AR23+AR26 !increment time value
TIME,AR26, !set time
EDPV,VELO, 2,0,AR22,0,0,0,0,
SOLVE

AR34=1
AR32='d3dump'
*DO, AR7, 5, AR24-1, 1
/SYS, call rD3DUMP.bat
AR33='Latest3dump'
/SOLU
EXISTAT,2, .7,AR33
AR26=AR23+AR26
TIME,AR26,
EDPV,VELO, 2,0,AR22,0,0,0,0,
SOLVE
*ENDDO

"MSG, UI,'Starting path operations'
"DO, AR27, 1, AR24, 1
"MSG, UI,'The index is:'
"MSG, UI,'Index ',%AR27%

/POST1
SET, AR27, LAST, 1
PARESU, AR40, ', ', AR21
PATH, STAT
*
AVPRIN, 0, ,
*
PDEF, .S.X, AVG
/PBC, PATH, , 1
*
AVPRIN, 0, ,
*
PDEF, .S.Y, AVG
/PBC, PATH, , 1
*
/OUTPUT, '%AR20%_%AR27%_SxCntrLn', 'lis'
PLPATH, SX, SY
PRTPATH, SX, SY
/OUTPUT
PARESU, AR41, ', ', AR21
PATH, STAT
*
AVPRIN, 0, ,
*
PDEF, .U.Y, AVG
/PBC, PATH, , 1
*
/OUTPUT, '%AR20%_%AR27%_Srfc', 'lis'
PLPATH, UY
PRTPATH, UY
/OUTPUT

*ENDDO
"MSG, UI,'end of lis output'

/POST26
FILE, AR20, 'rst', '.',
/ UI, COLL, 1
NUMVAR, 200
SOLU, 191, NCMIT
STORE, MERGE
FILLDATA, 191, , 1, 1
REALVAR, 191, 191
/AUTO, 1
/REP, FAST
*
NSOL, 2, AR31, U, Y, UY_2
STORE, MERGE
FILLDATA, 192, , 0, 0
FILLDATA, 193, , 1, 0
FILLDATA, 194, , -1, 0
FILLDATA, 195, , 1, 1
VARNAME, 195, NSET
!
! Name: VY_2
! ID:  3
! Function: nsol(105,V,Y)
NSOL, 3, AR31, V, VY_2
!
STORE, MERGE
FILLDATA, 192, , 0, 0
FILLDATA, 193, , 1, 0
FILLDATA, 194, , -1, 0
FILLDATA, 195, , 1, 1
VARNAME, 195, NSET
!
! Name: AY_2
! ID:  4
! Function: nsol(105,A,Y)
NSOL, 4, AR31, A, AY_2
!
STORE, MERGE
*
NSOL, 5, AR30, U, Y, UY_5
STORE, MERGE
! Save time history variables to file AR20 ballmotion.csv
*CREATE, scratch, gui
*DEL, _P26_EXPORT
*BIM, _P26_EXPORT, TABLE, 10000, 4
VGET, _P26_EXPORT(1,0), 1
7.2.14. Ansys batch code for obtaining post processing data from Ansys

Ansys 8 Code

Contents of SCRIPT_Zone16.bat

set ANSYS80_PRODUCT=structds
set ANS_CONSEC=YES
del d3dump*
del d3hsp*
"C:\Program Files\Ansys Inc\V80\ANSYS\bin\intel\ansys80" -b -i script_impact_1.txt -o script_impact.out.txt
"C:\Program Files\Ansys Inc\V80\ANSYS\bin\intel\ansys80" -b -i script_impact_2.txt -o script_impact.out.txt
"C:\Program Files\Ansys Inc\V80\ANSYS\bin\intel\ansys80" -b -i script_impact_3.txt -o script_impact.out.txt
.
.
"C:\Program Files\Ansys Inc\V80\ANSYS\bin\intel\ansys80" -b -i script_impact_98.txt -o script_impact.out.txt
"C:\Program Files\Ansys Inc\V80\ANSYS\bin\intel\ansys80" -b -i script_impact_99.txt -o script_impact.out.txt
"C:\Program Files\Ansys Inc\V80\ANSYS\bin\intel\ansys80" -b -i script_impact_100.txt -o script_impact.out.txt

Contents of SCRIPT_script_impact_1.txt
*USE,Script_Zone16.txt,1

Contents of SCRIPT_script_impact_2.txt
*USE,Script_Zone16.txt,2

Contents of SCRIPT_script_impact_3.txt
*USE,Script_Zone16.txt,3
.
.
.

Contents of Script_Zone16.txt

/BATCH
!version 1-16
!this version of the code does one load step at a time to work around the ansys memory leak
AR21='C:\D\PwrLw617um\17\' !30 character limit
AR20='PwrLwThin'  !filename
AR31=25 !number of paths to make
AR32=0.001 !distance from centre of furthest path (in meters)
AR33=AR32/AR31  !path spacing
AR34=-0.000617 !y-coordinate of bottom of strip. i.e. strip thickness in meters
AR22=ARG1
AR29=25  !number of substeps... or a number greater than the biggest expected substep number
AR2=25 !number of substeps... or a number greater than the biggest expected substep number

/CWD,AR21

/POST1
FILE,AR20,'rst','.'
/OUTPUT,SET,1,1a
SET,LIST
/OUTPUT
*DO,AR36,1,AR29,1
SET,AR22,AR36
*DO,AR35,0,AR31, 1 !path loop
PATH, res,2,30,100,
PPATH,1,0,AR30*AR33,0,0,0,
PPATH,2,0,AR30*AR33,AR34,0,0,
ETABLE, ETS,EPT0,EQV !total equivalent strain
PDEF, ,ETS,ETS,NG
ETABLE,EFEQ,NMEI!!, 1 !equivalent plastic strain
7.2.15. Ansys code to interface to Matlab used in process model

Ansys 8 Code

Contents of isRunFwa.bat:

@echo off
set ANSYS80_PRODUCT=structds
set ANS_CONSEC=YES
call reset.bat
"C:\Program Files\Ansys Inc\V80\ANSYS\bin\intel\ansys80" -b -i isMain.txt -o isMain.out.txt

Contents of isMain.txt:
/BATCH
/Config,NPROC,3
/CWD,'C:\_D\IStress'
*USE,isMatElem.txt ! This file is generated using matlab code; it defines mesh.
*USE,isSol.txt
*USE,isOut.txt

Contents of isMatElem.txt:
/BATCH
/CWD,'C:\_D\IStress'
/COM, ANSYS RELEASE  8.0
/PREP7
!*
ET,1,SHELL181
!*
KEYOPT,1,1,0
KEYOPT,1,2,0
KEYOPT,1,3,0
KEYOPT,1,4,0
KEYOPT,1,5,0
!*
!R,1,617e-6, , , , , ,
*use,isThickness.txt ! this is generated by Matlab to specify the thickness
R MORE, , , , , , ,
*
* MPTEMP, , ,
MPTEMP,1,0
MPDATA,EX,1,71.7e9
MPDATA,PRXY,1,0.33
sect,1,shell,,
secdatas,617e-6,1,0,3
secoffset,MID
seccontrol,0,0,0, 0, 1, 1, 1

Contents of isSol.txt:
/SOLI
ISFILE, Option, Fname, Ext, -->, LOC, MAT1, MAT2, MAT3, MAT4, MAT5, MAT6, MAT7, MAT8, MAT9, MAT10
ISFILE, READ, isDat, txt, , 1
outres,all,all
SOLVE
FINISH

Contents of isOut.txt
/PLOT1
/OUTPUT,'nodesUZ', 'lis'
/PNSOL,U,Z
/OUTPUT
/EFACE,1
AVPRIN,0,

/SHOW,PNS
PCHR,COMP,1,-1
PCHR,ORIENT,Horizontal
PCHR,COLOR,2
PCHR,ENMOD,1
/GFILE,1024,
/PLOPTS, DATE ,0
PLNSOL,U,Z,0,1
PLNSOL, S, X, 0, 1
PLNSOL, S, Y, 0, 1
PLNSOL, S, Z, 0, 1

232
7.3. Force Apparatus Drawings

All dimensions in this section are given in inches.

Assembly drawing of entire sensor apparatus with detailed views of different sections.

U-bolt assembly used to secure housing assembly tube.
U-bolt assembly used to secure sensor beam.

L-channel used as part of sensor beam U-bolt assembly
L-channel used as part of housing U-bolt assembly
Sensor Housing Pipe Fitting
Dytran transducer shield
Dytran transducer shield and Sensor Housing Pipe Fitting assembly, held together using fasteners.
Housing tube
Sensor beam. Threaded hole is used to mount the Dytran transducer.
Impact Cap for Dytran force sensor
Housing assembly boom
Housing assembly L-Brackets used to fasten the housing boom to the wall.
Sensor assembly L-Brackets used to fasten the sensor boom to the wall.
Wall Hole Positions
Sensor Assembly Boom

7.4. Modified Almen Gauge Drawings
Photographs of the modified Almen gauge

Assembly drawings of the modified Almen gauge
Almen Gauge - Base

Diameters for holes are given only for one the first of each series. There are three hole sizes here. Machine out of Aluminum 6061-T6.

SECTION A-A

The top must be very flat around these holes. Precision Fluid Bearings with a slightly larger diameter will fit in these holes and their tops must intersect a plane that is parallel to this page with a tolerance of plus or minus a micron.

SECTION B-B

Grade 1
1/4" x 1.5" Dowel Pins

SECTION J-J

SECTION K-K

Balloon Description
1. Tolerance for small holes
2. Tolerance for large holes for dowel pins.

Wed. July 21, 2004 - 5:51pm
Almen Gauge - Micrometer Stand Clamp

All dimensions are in inches
The default tolerance is plus or minus 0.001"
Al 6061-T6

3/16" x 0.75" Dowel Pin

1/4"-20 x 1.75" Socket Head Cap Screw
3/16" x 0.75" Dowel Pin

Insert Socket Head Cap Screw
1.75" Long

.2 ±.01
.2 ±.01
.4 ±.005
.6 ±.005
1.0 ±.001
.65 ±.001
.82 ±.01
1.00 ±.01
.41 ±.01

.1840
.1825
.1840
.1825

Ream.
Insert ANSI B18.8.2-1995
Ground Dowel Pin 3/16"

.201 THRU ALL
1/4-20 UNC - 2B .500
Ream.
Insert ANSI B18.8.2-1995
Ground Dowel Pin 3/16"
All dimensions are in inches. The default tolerance is plus or minus 0.001". The material is Aluminum 6061-T6.
Ball Bearing Placement on the Almen Gauge Base

Ball Bearings are glued on to Almen Gauge Base with epoxy glue. The top of the balls must be within 0.00005" of a plane that is parallel to the top surface of the base.

Almen Gauge Base "Top Surface"
The "Almen Gauge" will use this to perform measurements

Mitutoyo
No. 293-766-30

This does not need to be manufactured. It is just for reference. All dimensions are in inches. Measurements were made with calipers and should be accurate to within a 0.001"
This should be purchased 1 is required.

3/8-24 UNF-2B 3/4" Long
Socket Set Screw Cup Point
Dowel Pins

15.0\"  1.50

5 are required. The highest precision grade should be used, i.e. Grade 1

1/4" x 1.5" Dowel Pins
Almen Gauge - Precision Ball Bearings

Inches

This should be bought and does not need to be manufactured.

8 of these are required.
Dowel Pins
2 are required

3/16" x 0.75"