

**Comparison of proximal femur deformations, failures and
fractures in quasi-static and inertially-driven simulations of a
sideways fall from standing**

**An experimental study utilizing novel fall simulation, digital image
correlation, and development of digital volume correlation techniques**

by

Seth Myles Gilchrist

BSc, The University of Wyoming, 2003

MASc, The University of British Columbia, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Biomedical Engineering)

The University Of British Columbia
(Vancouver)

January 2014

© Seth Myles Gilchrist, 2014

Abstract

Knowledge of proximal femur failure mechanics has a pivotal role to play in predicting who might suffer a hip fracture. Previous researchers of sideways falls resulting in hip fracture have investigated the roles of several bone parameters such as bone mineral density and morphology, as well as different modelling boundary conditions. While important advances have been made, current models use constant displacement rates applied at the greater trochanter, which may not allow the bone to respond as it would in a sideways fall. Proximal femurs have been shown to be sensitive to displacement rate, but impacts like those in a sideways falls have never been examined. The goal of this thesis was to compare the results of constant displacement rate testing to more biofidelic, inertially driven, impact fall simulation testing and determine how these methods influence specific test outcomes. In study 1, sub-failure loads were applied to single bones at constant displacement rate, followed by impact loading in the fall simulator. Stiffnesses, energies and strains at a point were compared. In study 2, the same bones were compared using digital image correlation to examine bone strains on the anterior-superior femoral neck. In study 3, two cohorts of bones were loaded to failure, one at constant displacement rate and the other in the impact fall simulator. Initial failure locations, fracture patterns, stiffnesses and energies to failure were compared. The results of study 1 indicate that the behaviours of the bones were not affected by the change in loading. In study 2, I found that individual specimen strain maps were sensitive to the change in loading parameters; however, pooling the data from all the specimens yielded no statistical difference. In study 3, I discovered that final fracture patterns were different, but initial failure locations, stiffnesses

and energies to fracture were not. Additionally, femurs tested in the fall simulator did not show significant viscoelasticity. These data indicate that sub-failure measures of mechanical bone behaviour are insensitive to changing between constant displacement rate and impact fall simulation testing; however, strain pattern and final fracture behaviours are influence by changing the loading protocol.

Preface

Chapter 3 has been published as a design innovation paper [76]. I was responsible for conceptual design of the impact apparatus, literature review in support of the design, construction of the device, and validation of its performance. Dr. Cripton and Dr. Guy provided guidance, supervision and edited the manuscript.

Chapter 4 has been submitted for publication as an original paper [77]. I was lead investigator on this research, however, many other researchers contributed time and data to the investigation. The paper included new analyses of data from previous researchers. The specimens and data affiliated with the *slow* group were tested by myself and Dr. Nishiyama. Dr. Nishiyama prepared the specimens for testing and performed pre-test imaging on the specimens. Dr. Boyd provided supervision and funding for Dr. Nishiyama and proof reading of the paper. The specimens and data affiliated with the *fast* group were provided by Dr. de Bakker, who was supervised and funded by Dr. Oxland. I was responsible for analysing the data associated with these specimens. I collected, prepared and tested and analysed the *fall* group specimens. All statistical analyses were carried out by me. Dr. Cripton and Dr. Guy provided supervision and edited the manuscript.

I was lead investigator for the project described in Chapter 5. I was responsible for design of the tests, strain field analyses, video observations of fracture initiations, statistical analysis and writing of the manuscript. Dr. Nishiyama provided the quasi-static specimens and aided in testing them, and Dr. Boyd supervised and funded him. Dr. Guy assisted in sketching the fracture lines and classified the fractures into clinical types. He also assisted with manuscript

preparation and editing. Dr. Cripton advised me and assisted in manuscript preparation and editing.

I was lead investigator and study designer for the work presented in Appendix D, which will be submitted for publication. I performed all digital scans, and developed the methods for measuring the femoral version and the linea aspera angle and helped develop the posterior neck angle measurement. Michael Cancilla developed the posterior neck angle measurement and performed inter-observer reliability readings for other measures. Dr. Guy assisted in developing the posterior neck angle measurement, provided supervision and edited the manuscript. Dr. Cripton provided supervision and edited the manuscript generated from the chapter that is currently in submission.

I was lead investigator for the work presented in Appendix C. I developed all software and apparatus, and performed all validation testing. Dr. Liebschner provided images for the work presented in §C.4.2. Drs. Cripton and Guy provided funding, supervision and editing of posters and presentations associated with this work.

The work in Chapters 4 and 5 were approved by the University of British Columbia (UBC) Clinical Ethics review board (certificate number: H06-70337). The work in Appendix D was approved by the UBC Department of Anatomy (approval number: W0105).

Table of Contents

Abstract	ii
Preface	iv
Table of Contents	vi
List of Tables	xii
List of Figures	xiv
Glossary	xviii
Acknowledgments	xxv
1 Introduction	1
1.1 Overview	1
1.2 Hip anatomy	3
1.2.1 Femoral anatomy	3
1.2.2 Bone anatomy	6
1.3 Fractures of the proximal femur	11
1.3.1 Clinical considerations	11
1.4 Mechanical behaviour of bone	19
1.5 Determinants of bone strength	22

1.6	Age related changes in bone	27
1.7	Understanding fractures of the proximal femur	29
1.7.1	Human injury tolerance	30
1.7.2	Modelling falls to the side	32
1.8	Laboratory models of hip fracture	34
1.8.1	Development of orientation and constraint	35
1.8.2	Application of force	36
1.8.3	Lessons learned from laboratory models	38
1.8.4	Limitations of laboratory models	41
1.8.5	Summary	42
1.9	Computer models	42
1.9.1	Contributions of FE models	42
1.9.2	Limitations of FE models	43
1.9.3	Applications of models to fracture prevention	45
2	Objective and research questions	46
2.1	Research questions	46
2.2	Method of investigation	47
2.2.1	Force-displacement behaviour	47
2.2.2	Strain behaviour	47
2.2.3	Failure behaviour	48
3	Design of a fall simulator	49
3.1	Introduction	49
3.2	Materials and methods	52
3.2.1	Fall simulator apparatus	52
3.2.2	Data collection and processing	59
3.2.3	Digital image correlation verification	60

3.3	Results	62
3.4	Discussion	63
4	Impact and constant displacement rate loading change the mechanical response of the femur in sideways fall simulation	71
4.1	Introduction	71
4.2	Materials and methods	73
4.3	Results	77
4.4	Discussion	83
4.4.1	Fall group viscoelasticity	84
4.4.2	Failure and loading responses in the fall simulator	90
4.4.3	Conclusions	91
5	Impact and constant displacement rate loading change the surface strain and fractures of the femur in sideways fall simulation	93
5.1	Introduction	94
5.2	Materials and methods	96
5.3	Results	103
5.4	Discussion	107
6	Communication between mechanical and finite element modelling	114
6.1	Documentation from start to finish	114
6.2	Additional data for FE modelling	116
6.3	Data portability and transparency	118
6.4	Summary	120
7	Discussion and conclusions	121
7.1	Discussion	121
7.2	Conclusion	128

7.3 Future work	129
Bibliography	132
A Supporting Materials	162
A.1 Increased aBMD due to larger bone size	162
A.2 Femoral neck internal rotation justification	162
A.3 Calculation of Mass 1 initial velocity	164
A.4 Equation related to the response of an isolated ideal femur under constant displacement rate loading	165
A.5 Testing log sheets	166
A.6 Equipment characterization experiments	171
A.6.1 Drop tower fall velocity characterization	171
A.6.2 Potting torsion resistance	172
A.6.3 Materials testing machine compliance	174
A.6.4 Drop tower compliance	174
A.6.5 Foam compliance	177
A.6.6 Drop tower position measurement verification	179
B Additional plots	182
B.1 Quasi-stativ vs. fall simulator subfailure	183
B.2 Constant rate vs. fall simulator	188
C Digital volume correlation	191
C.1 The DVC technique	192
C.2 Digital volume correlation design	193
C.2.1 Imaging for DVC	194
C.2.2 Subdividing images for DVC	194
C.2.3 Registration of image for DVC	197

C.2.4	Error analysis	205
C.2.5	Data storage	205
C.2.6	Spatial differentiation	206
C.2.7	Coding considerations	206
C.2.8	Loading apparatus	206
C.3	Verification of DVC	214
C.3.1	Synthetically strained image	214
C.3.2	Zero strain image	215
C.4	DVC experimental results	218
C.4.1	Axial compression of a rabbit vertebra	219
C.4.2	Axial compression of a human bone core	222
C.5	DVC discussion	225
D	Anatomical correlations in the femur	226
D.1	Abstract	226
D.2	Introduction	227
D.3	Materials and methods	228
D.4	Results	231
D.5	Discussion	234
D.6	Acknowledgements	237
E	Computer Code	238
E.1	Drop tower and Instron analysis	238
E.1.1	Example input	239
E.1.2	Specimen class	241
E.1.3	Experiment class	245
E.1.4	Instron test class	248
E.1.5	Instron DAQ class	258

E.1.6	Drop tower test class	267
E.1.7	Drop tower DAQ class	287
E.1.8	Drop tower displacement class	298
E.1.9	DIC data class	303
E.2	Idealized fall simulator model	306
E.2.1	Solving the system ODE	306
E.2.2	Plotting the system response	307
E.3	DIC to DIC registration and comparison	309
E.3.1	Functions library for converting DaVis to VTK format	309
E.3.2	Main program for converting output from DaVis to VTK format	313
E.3.3	Functions library for comparing strains stored on two surfaces	314
E.3.4	Main program for comparison of strains stored on two surfaces	322
E.4	Digital volume correlation	325
E.5	Digital volume correlation verification	325

List of Tables

Table 1.1	Trabecular quantification	9
Table 1.2	WHO osteoporosis definitions	13
Table 1.3	Inputs to the FRAX® model	15
Table 1.4	Elastic properties of bone	20
Table 1.5	Failure properties of bone	24
Table 1.6	Laboratory model results summary	39
Table 4.1	Specimen groups	74
Table 4.2	Data for failure groups	78
Table 4.3	Specimen and Data Summary	81
Table 5.1	Additional study specimens	97
Table 5.2	Failure locations and fracture types for simulated group	105
Table 5.3	Failure locations and fracture types for slow group	106
Table 5.4	Initial failure and final fracture locations	106
Table 5.5	Published fracture locations for laboratory experiments	107
Table 5.6	Published fractures locations for clinical studies	108
Table 7.1	Required tests for outcomes	128
Table A.1	Drop tower apparatus mass	177
Table A.2	Foam stiffness and stress	179

Table A.3	Materials testing machine position verification	180
Table C.1	Spacial transformations	202
Table C.2	DVC cross load device calibration	211
Table D.1	Version and linea aspera data	232

List of Figures

Figure 1.1	The hip joint	3
Figure 1.2	The femur	4
Figure 1.3	Anatomical and femoral geometrical references	6
Figure 1.4	Femoral version	7
Figure 1.5	Cortical and cancellous bone images in the proximal femur	8
Figure 1.6	Cortical bone cross-section	8
Figure 1.7	Determination of MIL	10
Figure 1.8	MIL of a 2D section	10
Figure 1.9	The hip joint capsule	17
Figure 1.10	Intracapsular fractures	17
Figure 1.11	Extracapsular fractures	18
Figure 1.12	Cancellous bone failure properties	23
Figure 1.13	Cancellous bone stress-strain curve	25
Figure 1.14	Age related changes in bone	27
Figure 1.15	Response corridors and injury curves	31
Figure 1.16	Proximal femur loading and orientation constraints	36
Figure 3.1	Schematic of the fall simulator	53
Figure 3.2	Explanation of femur and lateral pelvis mass	55
Figure 3.3	Photo of the fall simulator	60

Figure 3.4	Fall simulator data compared to a human volunteer	63
Figure 3.5	Validation of digital image correlation (DIC) on the femoral neck	64
Figure 3.6	Example DIC strain and strain gauge data	65
Figure 3.7	Example surface strain data from the DIC analysis	66
Figure 4.1	Materials testing machine and impact simulator images	74
Figure 4.2	aBMD grouped by relative stiffness	79
Figure 4.3	Force-displacement behaviours grouped by relative stiffness of fall:FS and fall:QS	80
Figure 4.4	Force and displacement vs. time example	82
Figure 4.5	Force vs. displacement example	83
Figure 4.6	Stiffness vs. loading and displacement rates	84
Figure 4.7	Idealized model of the fall simulator	86
Figure 4.8	Ideal loading and compression rates vs. stiffness	86
Figure 4.9	Ideal loading and displacement rates as a function stiffness and damping .	87
Figure 4.10	Idealized Courtney et al. [46] model	88
Figure 4.11	Stiffness vs. displacement rate and damping	89
Figure 5.1	Original positions of the DIC surfaces	99
Figure 5.2	DIC surfaces aligned after ICP	100
Figure 5.3	ICP target surface extruded	101
Figure 5.4	Transfer of DIC data between surfaces	101
Figure 5.5	Surfaces to compare DIC results	102
Figure 5.6	Fall:FS group fracture lines	104
Figure 5.7	Slow group fracture lines	104
Figure 5.8	Strain differences overlaid on an example bone	108
Figure 5.9	Strain differences for specimens in the simulated group	109
Figure 6.1	Experimental and FE data flow	117

Figure 6.2	Digitizing points on the experimental apparatus	118
Figure A.1	Hip proximity related to femoral head	163
Figure A.2	Drop tower calibration	172
Figure A.3	Potting torsion resistance	173
Figure A.4	Materials testing machine compliance test	175
Figure A.5	Materials testing machine compliance results	175
Figure A.6	Drop tower compliance test	177
Figure A.7	Drop tower compliance results	178
Figure A.8	TEMA measured displacement verification	181
Figure B.1	Quasi-static vs. fall simulator strain	183
Figure B.2	Quasi-static vs. fall simulator stiffness	184
Figure B.3	Quasi-static vs. fall simulator energy	185
Figure B.4	aBMD grouped by strain	186
Figure B.5	aBMD grouped by energy	187
Figure B.6	Stiffness of groups in failure tests	188
Figure B.7	Energy to yield in failure tests	189
Figure B.8	Yield force vs. total areal bone mineral density (aBMD) in failure tests . .	190
Figure C.1	Illustration of the DVC method	192
Figure C.2	Image registration process	198
Figure C.3	Image data representation	199
Figure C.4	Interpolation of image data	200
Figure C.5	Interpolator comparison	201
Figure C.6	Transformation comparisons	203
Figure C.7	DVC axial loading device	207
Figure C.8	DVC axial loading device transparent	208
Figure C.10	DVC axial loading specimen mould	208

Figure C.9 Load cell for axial loading apparatus	209
Figure C.11 Cross loading HR-pQCT apparatus	210
Figure C.12 Exploded view of the cross loading HR-pQCT apparatus	210
Figure C.13 Cross loading apparatus calibration	212
Figure C.14 DVC cross load device calibration	213
Figure C.15 5% synthetic strain comparison	215
Figure C.16 Synthetic strain results	216
Figure C.17 Zero strain results	217
Figure C.18 Zero strain histogram	218
Figure C.19 Rabbit vertebra before testing	219
Figure C.20 Rabbit vertebra compression	220
Figure C.21 Rabbit vertebra after testing	221
Figure C.22 Rabbit vertebra growth plate	222
Figure C.23 Human bone compression	223
 Figure D.1 Constraints and orientation	227
Figure D.2 Femur scanning set up	229
Figure D.3 Digital version measurement	230
Figure D.4 Digital linea aspera measurement	230
Figure D.5 Digital posterior neck measurement	231
Figure D.6 Linea aspera, version and posterior neck angle histogram	233
Figure D.7 Posterior neck vs. version	234
Figure D.8 Linea aspera vs. version	235
Figure D.9 Separation angle vs. version	236
 Figure E.1 Data analysis hierarchy	239

Glossary

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
A_v	version angle
A_{la}	linea aspera angle
A_{pn}	posterior neck angle
A_s	separation angle
aBMD	areal bone mineral density
a.k.a.	also known as
A-P	anterior-posterior
<i>b</i>	bone
BC	boundary condition
BMC	bone mineral content
BMD	bone mineral density

BRIDGE CIHR/MSFHR Strategic Training Program Bridging Public Health, Engineering and Policy Research

BS/TV bone surface density

B-spline basis spline

BV/TV bone volume ratio

C Celsius

c damping coefficient

CHHM Centre for Hip Health and Mobility

CI confidence interval

C++ C++ programming language

ConnD connective density

CT computed tomography

cm centimetre

DAQ data acquisition system

δ uncertainty

Δ change in

DIC digital image correlation

DICOM digital image and communications in medicine

s displacement

DVC digital volume correlation

DXA dual-energy x-ray absorptiometry

e.g. for example

E energy

ϵ strain

F force

FE finite element

fps frames per second

FRAX[®] fracture risk assessment tool

FS fall simulator

Fx fracture

g gram

GB gigabyte

GHz gigahertz

HR-pQCT high resolution, peripheral, quantitative computed tomography

Hz hertz

ICP iterative closest point

ICORD International Collaboration on Repair Discoveries

I incident energy level

i.e. in other words

ITK Insight Toolkit

IS	intertrochanteric stable
IU3	intertrochanteric, unstable 3-part
IU4	intertrochanteric, unstable 4-part
J	joules
k	stiffness
kg	kilogram
kHz	kilohertz
kN	kilonewton
kPa	kilopascal
l	length
L	original length
LVDT	linear variable differential transformer
μ	x-ray attenuation coefficeint
m	meter
mat	Matlab MAT
MBD	multibody dynamic
MIL	mean intercept length
min	minute
mm	millimetres
MRI	magnetic resonance imaging

ms	milliseconds
$\mu\epsilon$	microstrain
μm	micrometer
N	newton
N/A	not available
<i>n</i>	energy level index
n	position index
N	neck
NFx	no fracture
NSERC	Natural Sciences and Engineering Research Council
ODE	ordinary differential equation
OIBG	Orthopaedics and Injury Biomechanics Group
OR	odds ratio
OREF	Orthopaedics Research and Education Fund
OP	osteoporosis
PMMA	polymethylmethacrylate
<i>p</i> _t	p-value from Student's t-test
<i>p</i> _w	p-value from Wilcoxon test
PVC	polyvinyl chloride
px	pixel

qs	quasi-static
QS	quasi-static
R^2	coefficient of determination
RAM	random access memory
RMS	root mean squared
ROI	region of interest
s	second
SAR	strength anisotropy ratio
SD	standard deviation
S-I	superior-inferior
σ	stress
sim	fall simulator
SMI	structure model index
SSSIG	sum of square of subset intensity gradients
stl	stereolithography
ST	subtrochanteric
t	tissue
TbN	trabecular number
TbTh	trabecular thickness
TbSp	trabecular spacing

THA total hip arthroplasty

V volts

vs. versus

VTK Visualization Toolkit

WHO World Health Organization

x path length in tissue

x position

Acknowledgments

The work in this thesis would not have been possible without support and assistance from a number of people and institutions.

I would like to start by thanking my supervisors and advising committee for their guidance and help throughout the process. As co-supervisors, Drs. Peter Cripton and Pierre Guy provided invaluable support in the completion of this work and in my training as a researcher. As members of my advising committee, Drs. Rizhi Wang, Dave Wilson and Thomas Oxland provided support and guidance throughout the duration of my studies. I am very appreciative of the time and effort that these researchers have given to my personal and professional development.

Secondly, I would like to acknowledge my funding sources. Four years of stipend and research funding were provided by the CIHR/MSFHR Strategic Training Program Bridging Public Health, Engineering and Policy Research (BRIDGE) fellowship program. The BRIDGE program also provided me with numerous learning and development opportunities, all of which were beneficial to my professional development and enjoyable on a personal level. Funding for purchasing specimens and other laboratory equipment was provided by an Orthopaedics Research and Education Fund (OREF) grant from the Department of Orthopaedics at the University of British Columbia. Additional funding for laboratory space and other laboratory equipment was provided by a Natural Sciences and Engineering Research Council (NSERC) operating grant. Finally, the facilities and institutes in which the tests and analysis were conducted (namely, the Blusson Spinal Cord Centre with ICORD and the Robert H. N. Ho Building

with the CHHM) were funded by the Canadian Foundation for Innovation.

There were also a number of faculty and students who assisted me both technically and emotionally over the course of my studies. Angela Melnyk, Jenn Douglas and Oscar Ariza of the Orthopaedics and Injury Biomechanics Group (OIBG), Dr. Ben Helgason of ETH-Zürich and Dr. Danmei Liu of the Centre for Hip Health and Mobility (CHHM) were all instrumental in completion of these studies.

Finally, I would like to thank my wife, Claire, my family, and all of my friends for their support over the years.

Chapter 1

Introduction

1.1 Overview

Hip fracture is a devastating injury which is becoming more common in western countries as the population ages [132, 213]. The term *hip fracture* refers to fracture of the proximal femur near the hip joint. Typically, hip fracture affects elderly individuals, results from falls [108, 188], and is sometimes associated with degenerative processes of bone, such as osteoporosis [132, 169].

The hip fracture process involves a cascade of events, each contributing to the transformation of a person into a patient. The cascade starts with a fall [92], and ends with an impact to the greater trochanter on the side of the leg [81], which can result in hip fracture. Many researchers have contributed to the understanding of the details of each stage of the process, with the most clinically measurable improvements being made in fall prevention. Fracture prediction and prevention have not made the same gains. We find ourselves in a position where the likelihood of an individual's hip fracturing due to a fall to the side cannot be determined to the level of certainty needed to justify clinical intervention.

The primary screening tool for identifying a potential hip fracture patient (besides age) is osteoporosis state (a clinical classification of bone mineral density (BMD)), which is supported by some epidemiological evidence, as well as published *ex-vivo* hip fracture models. Osteo-

porosis state is determined by BMD and is a proxy measure for a bone's fracture strength. While osteoporosis state is a useful clinical tool, examination of clinical admissions show that the majority of hip fractures happen outside of the osteoporotic population [80, 131, 212, 220]. While the fracture risk for an individual with normal bone density is low, I believe that structural factors independent of density may increase the risk of fracture from a fall to the side and could act as additional screening targets.

Previously developed experimental and computational models of hip fracture have been tremendously informative in determining the relative importance of specific structural properties. Researchers have investigated the effects of BMD, loading vector orientation, compression rate and overall femoral geometry on the strength of the proximal femur, but they have been unable to identify any screening targets for individuals besides age and BMD. Likewise, they have not established treatment options to reduce the burden of hip fracture on a population level, other than lifestyle changes and pharmaceuticals.

This thesis aims to push the state-of-the-art hip fracture testing and modelling methods to their next stage. This advancement will be made through the development of an inertially driven fall simulator that models the entire fall through fracture cascade, and through the development of tools for advanced, quantitative analysis of imaging data that allow us to look at external and internal bone strains throughout the fracture process. Additionally this thesis aims to advance the understanding of how the experimental boundary conditions affect the outcomes of fall simulation models.

In the following chapters, I will detail the work that has been done to this end. This chapter provides background knowledge regarding bone as a material; current clinical fracture risk evaluation and treatments; and current laboratory and computational models of the proximal femur. It will end with a statement of objectives that guided and informed the work presented in chapters 3 through 6 of this thesis. Finally, Chapter 7 will outline the contributions of the work to the field and make recommendations for further research to advance our understanding of, and ability to predict and prevent, hip fracture.

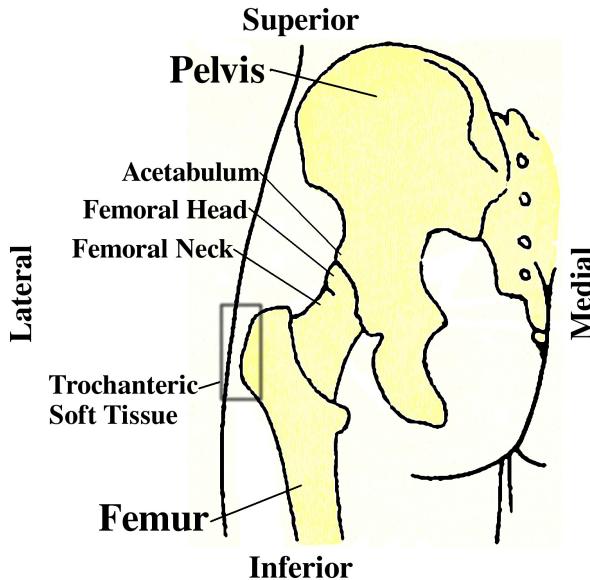


Figure 1.1: The hip joint is the junction of the trunk with the lower limb. This posterior view shows the pelvis on the medial side with the proximal femur on the lateral side of the joint. Graphic adapted from Gray and Warren Harmon Lewis [79] (copyright expired).

1.2 Hip anatomy

The hip joint is the union of the torso with the lower limb (Figure 1.1). It comprises the pelvis and proximal femur, and is a ball-and-socket joint with a large range of rotational motion. The pelvic component is the socket part of the joint and is called the acetabulum. On the femoral side is the femoral head, which is the ball of the joint, and the femoral neck, which offsets the lower limb laterally from the socket. Fractures of the hip can be on either the femoral or pelvic side of the joint. Although increasingly frequent on the acetabular side [69], the term “hip fracture” is usually reserved for clinical failure of the proximal femur.

1.2.1 Femoral anatomy

The femur is a large bone that is crucial for locomotion. It has many muscle attachments and bony landmarks associated with those attachments. The largest bony landmarks on the proximal end of the femur are the *greater* and *lesser trochanters* (Figure 1.2). The region between these two landmarks is called the *intertrochanteric region* and is relevant clinically as the site of approximately half of hip fractures [128, 145, 159, 191, 213]. The intertrochanteric region has a different morphology on the anterior and posterior aspects. On the anterior side of the intertrochanteric region is the *intertrochanteric line*, which runs from the supero-lateral greater

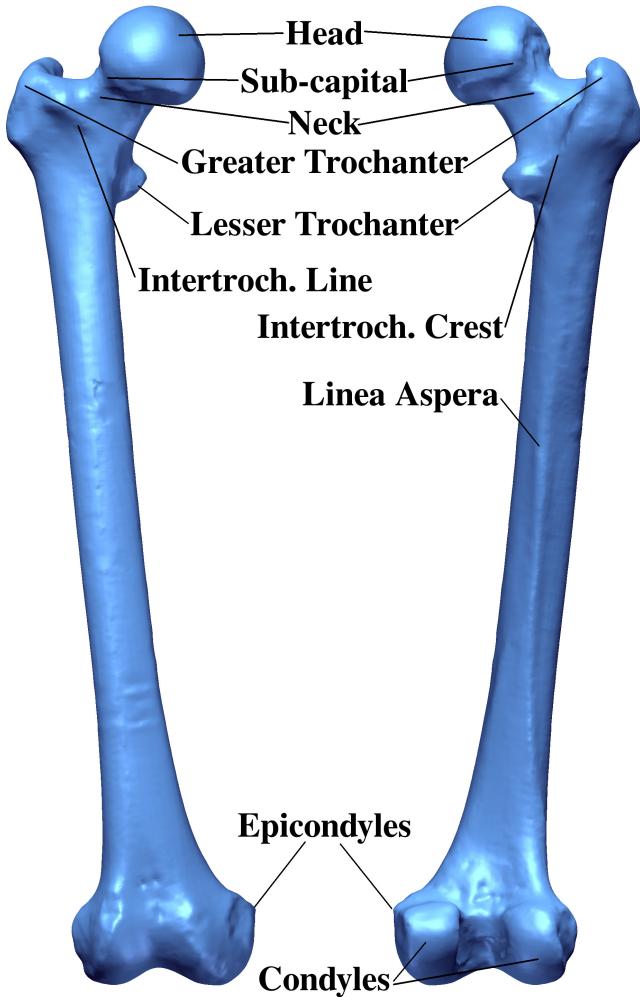


Figure 1.2: The femur, anterior (left) and posterior (right). The trochanters, linea aspera and epicondyles are the insertions of the muscles of the hip and knee joints. Graphic ©Seth Gilchrist, 2013.

trochanter to the lesser trochanter. On the posterior side is the intertrochanteric *crest*, which has a similar path to the intertrochanteric line, running from the superio-lateral grater trochanter to the lesser trochanter. The intertrochanteric crest is more prominent than the intertrochanteric line.

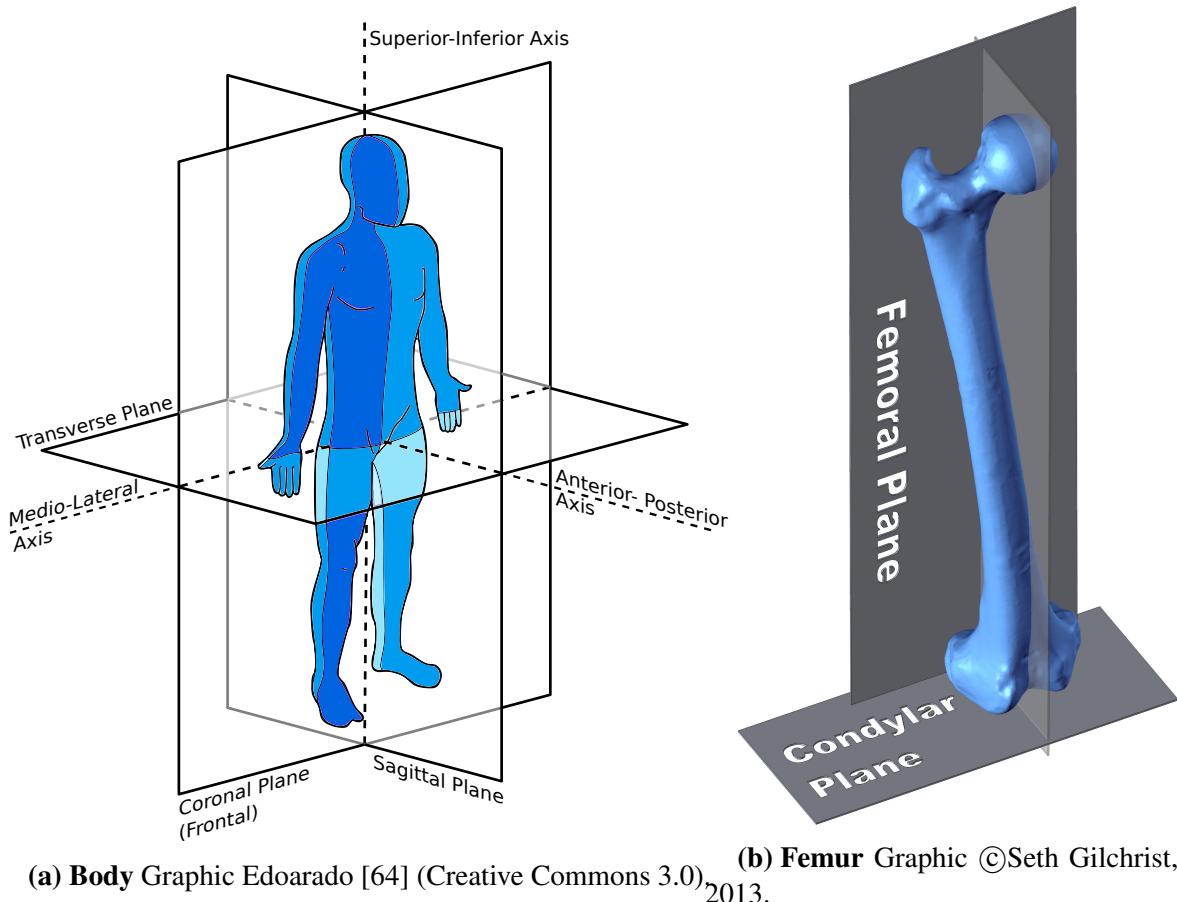
Along the posterior aspect of the femoral shaft is the *linea aspera*, which is the insertion for a number of muscles of both the knee and the hip joints. There are a number of landmarks on the distal end of the femur, but the important ones for the current discussion are the *condyles* and *epicondyles*, which are used to define the coordinate system of the femur. The condyles

of the femur are the weight bearing portions of the distal femur. The posterior and inferior condyles are approximately aligned with the coronal and transverse planes of the body, respectively in the anatomical position (Figure 1.3a). The epicondyles of the femur are associated with the collateral ligaments of the knee and are approximately aligned with the coronal plane of the body.

The coordinate system of the femur consists of two anatomical planes, with the third plane defined as orthogonal to the others (Figure 1.3) [79]. The primary plane of reference is the *femoral plane*, which is determined by the three most posterior points on the bone, typically the posterior condyles and trochanter, and is roughly parallel to the coronal plane of the body. This plane can be thought of as the surface of a table when the bone is placed, posterior down, on the table. Intraoperatively it is sometimes defined using the epicondyles rather than the posterior condyles, but the correlation between the two definitions is high and the difference small, with an average value of 2° [62].

The second plane is the *condylar plane*, and is defined as perpendicular to the femoral plane, and in contact with the inferior condyles. It is roughly parallel to the transverse plane of the body. The *mechanical plane* is defined as perpendicular to the other two planes, passing through the centre of the femoral head, and is approximately parallel to the sagittal plane of the body. This plane contains the mechanical axis of the femur, running from the centre of the femoral head to the point between the femoral condyles.

The femur has natural curves and twists that occur along its length. The most obvious curve is the anterior bow of the femur [79], which is a bending of the shaft along its length, and occurs predominantly in the third plane. A more subtle feature is the orientation of the femoral neck to the femoral plane, called *femoral twist* or *femoral version* (Figure 1.4). When the neck extends anteriorly to the femoral plane the bone is said to be anteverted, and if it extends posteriorly to the femoral plane it is said to be retroverted. The femoral version can be measured in a number of different ways using different aspects of the neck to define its orientation. Using the transverse mid-point of the lateral femoral neck and the centre of the



(a) Body Graphic Edoarado [64] (Creative Commons 3.0), 2013.

(b) Femur Graphic ©Seth Gilchrist, 2013.

Figure 1.3: Anatomical planes and directions for the body and femur. In (b), the femoral and condylar planes are labelled, and the mechanical plane is shown with transparency.

femoral head is called the Kingsley-Olmsted method [118], considered the clinical standard for version definition. Other methods of measurement particular to modern imaging systems have also been devised, one example being the use of the anterior surface of the femoral neck as observed using ultrasound [1]. Using the Kingsley-Olmsted method, the normal population has an anteversion angle of (mean(SD)) $9.73^\circ(9.82^\circ)$ [227].

1.2.2 Bone anatomy

The bones of the body are important organs, with roles in support and movement as well as physiological homoeostasis. They are composite structures with material types and properties that vary spatially and in time.

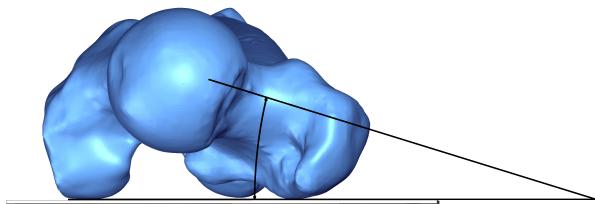


Figure 1.4: Femoral version is the acute angle between the neck and the femoral plane. It is measured by observing the angle from the superior aspect. This images shows the neck axis as defined using the Kingsley-Olmsted method [118], where the midpoint of the lateral femoral neck is identified from the supero-lateral aspect and is connected to the centre of the femoral head. This specimen is anteverted by an angle of 15.2°.

Image ©Seth Gilchrist, 2013.

At a material level, bone is made up of a collagen mesh that is mineralized by hydroxyapatite, and organized in a laminar fashion, making it strong and stiff. At a structural level bone is classified into two types: cortical and cancellous (Figure 1.5). Cortical bone (a.k.a. compact bone) is primarily found in the shaft (or diaphysis) of long bones, and also makes up the exterior shell of irregular bones. Cancellous bone (a.k.a. spongy, or trabecular bone) is a low density network, consisting of columns and plates of bone that form a cellular solid. It is found in the ends (or epiphyses) of long bones, and in the interior of most irregular bones.

There are two types of cortical bone, with names that refer to the organization of the collagen network: woven and lamellar. *Woven* bone is characterized by an unorganized, tangle of collagen Type-I fibres, and is mechanically weak. In adults, woven bone is transient, created after injury where quick reinforcement is needed, and replaced by lamellar bone as healing progresses.

Lamellar bone is characterized by long, laminarily organized, mineralized, collagen Type-I fibres. Two subtypes of lamellar bone are commonly found in the appendicular skeleton: osteonal and interstitial. Osteonal lamellar bone is contained within osteons (also called Haversian systems) which are concentric rings of lamellar bone that surround a central nutrient-transport canal (Figure 1.6). The collagen fibres in the bone of the Haversian systems are

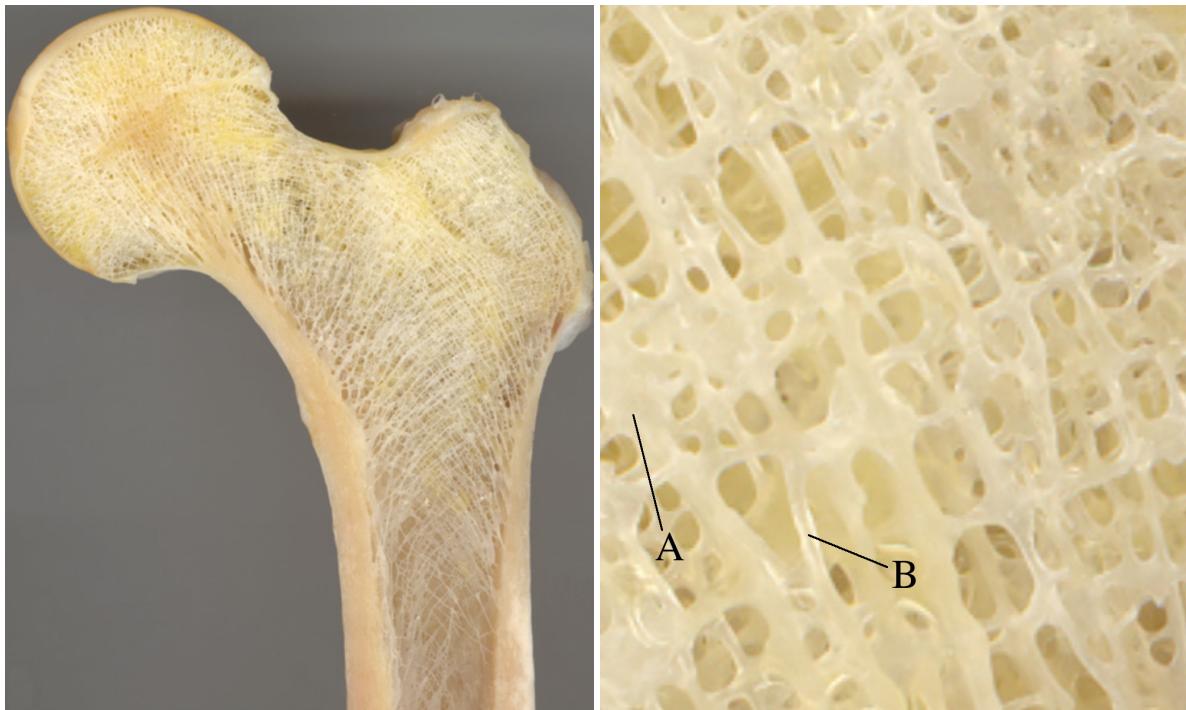


Figure 1.5: Left: A section of the proximal femur. Cortical bone serves as the exterior envelope of the organ, with cancellous bone in the interior. Right: Close up view of the cancellous bone showing plate-like (A) and rod-like (B) trabeculae. Images adapted from Ascenzi et al. [4] (with permission).

Figure 1.6: A cross-section of cortical bone at high resolution. The concentric rings are lamellar bone (each about $5\text{ }\mu\text{m}$ thick), organized into Haversian systems. The spaces between the Haversian systems contain interstitial lamellar bone.. Image courtesy of Konrad [120] (public domain).



oriented in roughly the same direction, but can be as much as 90° off from adjacent layers (both running 45° from the osteon axis). Interstitial lamellar bone consists of fragments of lamellar bone that fill the gaps between Haversian systems.

Cortical bone is quantified using either density (g/cm^3), or bone volume ratio (BV/TV) in conjunction with beam section properties such as the second area moment of inertia.

Table 1.1: Trabecular quantification measures. Many more that have been proposed, but these are the most common.

Quantity	Measure	Units	Description
Density	Bone Volume Ratio (BV/TV or V_V)	$\frac{\text{mm}^3}{\text{mm}^3}$	Ratio of bone volume to total volume in a given region of interest (ROI).
Surface Area	Surface Volume Ratio (BS/TV or S_V)	$\frac{\text{mm}^2}{\text{mm}^3}$	Ratio of bone surface area to total volume in a given ROI.
Trabecular Dimensions	Trabecular Thickness (TbTh)	mm	Average trabecular thickness calculated in a random, two-dimensional (2D) slice.
	Trabecular Spacing (TbSp)	mm	A measure of spacing between trabeculae, derived using a parallel-plate, theoretical model of trabecular morphology.
Anisotropy	Mean Intercept Length (MIL)	mm	A measure of the material fabric orientation, derived from 2D sections, which indicates the number of intersections along a projection from a randomly selected point. Provides a tensor, with the first and third Eigenvectors indicating the maximum and minimum material directions.
Topology	Trabecular Number (TbN)	$\frac{1}{\text{mm}}$	A measure indicating the number of trabeculae in a given volume. Derived using a parallel-plate, theoretical model of trabecular morphology.
	Connective Density (ConnD)	$\frac{1}{\text{mm}}$	A measure of the number of trabecular intersections.
	Structure Model Index (SMI)	unitless	A measure of morphology as more rod or plate like. An ideal rod structure has an SMI of 0 and an ideal plate structure has an SMI of 3.

Cancellous bone is also made up of lamellar bone; however, the mesioscale (i.e., between micro- and macroscale) organization of cancellous bone gives it significantly different structural properties. It has an open cell structure that is made up of discrete rod- and plate-like trabecular elements (Figure 1.5). This structure means that cancellous bone is much softer and better at absorbing energy than cortical bone. The organization of the cellular matrix is critical to the behaviour of the material, and quantification of this structure is done in a number of ways (Table 1.1).

An especially important cancellous bone measurement for the current discussion is the mean intercept length (MIL), which quantifies the orientation of the trabecular network and is important when evaluating material anisotropy. The MIL is determined from 2D sections on which lines at various angles are drawn. The average distances between trabeculae on the lines are recorded at each angle (Figure 1.7). This measurement yields a tensor which defines the

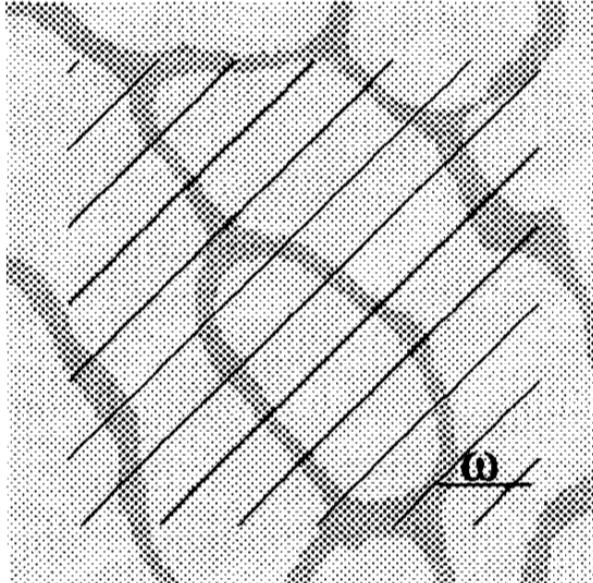
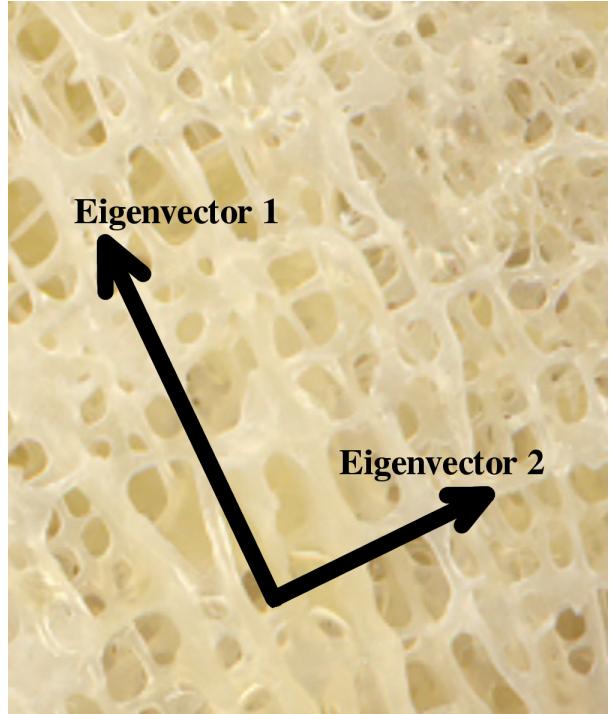


Figure 1.7: A bone cross-section showing the lines used to determine the MIL. The average distance between the trabeculae along the lines is recorded at various values of the angle ω . An image slice in a perpendicular direction will give the MIL in the out-of-image dimension. Graphic adapted from Odgaard [174] (with permission).

Figure 1.8: Eigenvectors are a mathematical tool for identifying important directions in systems described by tensors. In the case of fabric tensors, like those used for MIL, the first Eigenvector will point in the direction where most of the bone is oriented. The second and third Eigenvectors will be perpendicular to the first (and each other) and will point in the directions of the other two main material directions. In this 2D depiction, the first Eigenvector points along the strongest trabeculae and the second points in the perpendicular direction, along transverse supports. Image adapted from Ascenzi et al. [4] (with permission).



fabric properties in any given cut of the bone. The Eigenvectors and Eigenvalues of the tensor can then be used to determine the principal directions, in much the same way that principal stresses can be determined from the Eigenvectors and values of a stress tensor (Figure 1.8).

1.3 Fractures of the proximal femur

Fractures of the proximal femur are an important social and clinical problem in Canada and around the world. They constitute a large proportion of all orthopaedic injuries, with some studies reporting more than 20% of orthopaedic beds being occupied by hip fracture patients at any given time [228]. While research has shown a trend of decreasing fracture rates in recent years [128], the total incidents of fracture is increasing as the population ages [244], and the hospital admissions are becoming more complex, with increasing number of comorbidities [213].

Fractures are thought to be caused in two ways. The first, and likely the most common, is as a result of a fall to the side; the second, and less common, is a spontaneous fracture due to extreme bone fragility. While both types of fractures are important, the focus of the current work is on fractures due to falls as they are thought to constitute upwards of 95% of hip fracture cases [188].

The financial impact of the fractures on the health care system is difficult to overstate. The 1-year cost of an individual fracture in Canada has been put in the range of CDN\$29,065 – \$60,013 ([244], corrected to 2013 dollars by the Bank of Canada [8]). Lifetime costs of a single hip fracture in the US are as high as US\$107,193 ([30], corrected to 2013 dollars by the US Bureau of Labor Statistics [34]). All together, these fractures cost the Canadian system CDN\$659M/year ([225], corrected to 2013 dollars by the Bank of Canada [8]), and this value could more than double by 2040 [244].

Considerable work has been performed by the research community to understand and prevent hip fractures. This section will describe the clinical procedures and burdens of hip fracture, and the research that has been conducted to reduce that burden.

1.3.1 Clinical considerations

Hip fracture is a significant clinical problem, and an understanding of the needs and requirements of the clinical situation is key to developing a meaningful treatment or mitigation strat-

egy. Clinical interventions begin with screening and, if indicated, continue with prophylactic prescription of either drugs or biomechanical devices, such as hip protectors. If screening and prevention fail, and a hip fracture results, the most common clinical treatment is surgical repair using plates, screws or full joint components. There are a variety of surgical options depending on the details of the fracture.

Early risk assessment

The primary risk factors for hip fracture are age, falling, and bone density [54]. Risk assessment and clinical treatment targets are therefore geared towards assessing and addressing these factors. There are a number of preventative interventions that have shown some success, such as pharmacological [96], lifestyle [161] and to some degree hip protectors [85]. The first step to implementing one of these possible treatments is identification of individuals at risk of hip fracture through screening. The gold standard screening technique is dual-energy x-ray absorptiometry (DXA) to quantify bone mineral density (BMD), which is then used to evaluate osteoporosis state.

Osteoporosis is a clinical diagnosis of low BMD. It is not a disease in itself, but a condition caused by a number of possible ailments, including vitamin-D deficiency, hormone deficiency, or physiological ageing [5, Ch.12]. Two metrics exist for rating an individual's bone density. The *T-score* shows where an individual's BMD falls in comparison to a sex and race match young adult population (Equation 1.1), and the *Z-score* shows where an individual's bone density falls in relation to a sex, race and age-matched population (Equation 1.2) [179]. The definition of osteoporosis state, as given by the World Health Organization (WHO), uses the T-score (Table 1.2).

Table 1.2: WHO definition of osteoporosis state [243].

Classification	Description
Normal	BMD greater than 1 SD below the young adult average ($T > -1$).
Osteopenia	BMD between 1 and 2.5 SD below the young adult average ($-1 \geq T > -2.5$).
Osteoporosis	BMD less than 2.5 SD below the young adult average ($T \leq -2.5$).

$$T = \frac{BMD_{patient} - \overline{BMD}_{lifetime\ peak}}{\text{SD}_{lifetime\ peak}} \quad (1.1)$$

$$Z = \frac{BMD_{patient} - \overline{BMD}_{age-matched}}{\text{SD}_{age-matched}} \quad (1.2)$$

DXA is an x-ray imaging technique that provides a measure of mineralized bone content between the x-ray source and collector. Since a 2D image is created, the output of the technique is commonly called the areal bone mineral density (aBMD), in which the bone mineral content (BMC) is normalized by the projected area, and is quantified in units of g/cm². To measure BMC, the DXA method breaks the human body into two components: soft tissue and bone mineral. The soft tissues of the body all have approximately the same x-ray attenuation, which is significantly lower than that of bone mineral. With only two components, the amount of each can be found by simultaneously solving the Beer-Lambert relation at two energy levels (Equations 1.3 and 1.4) [84, 102].

$$I_1 = I_{0,1} \cdot e^{-(\mu_{1,t} \cdot x_t + \mu_{1,b} \cdot x_b)} \quad (1.3)$$

$$I_2 = I_{0,2} \cdot e^{-(\mu_{2,t} \cdot x_t + \mu_{2,b} \cdot x_b)} \quad (1.4)$$

$I_{0,n}$ and I_n are the incident and transmitted x-ray intensities at energy level n , respectively. $\mu_{n,t}$ and $\mu_{n,b}$ are the linear attenuation coefficients at energy level n for the tissue (t) and bone (b), and x_t and x_b are the path lengths in the tissue and bone. The transmitted and incident energy levels, as well as the absorption coefficients, are known *a priori*, and therefore the path

lengths can be solved. The product of the known density of bone mineral ($\text{[kg/m}^3\text{]}$) with the calculated path length ([m]) is aBMD ($\text{[kg/m}^2\text{]}$).

Modern DXA machines include internal calibrations of known path lengths to ensure accuracy [83]; however, there are a number of issues that must be considered when calculating aBMD using DXA [24]. To begin with, data from DXA machines cannot be readily compared across manufacturers [75]. Each manufacturer creates a custom database of population aBMDs, making it possible to compare T- and Z-scores. However, direct comparison of aBMD values must be corrected based on a standard calibration performed on each machine [75].

Secondly, the changes in bone quality can happen in either the cancellous or cortical compartments. Cancellous bone, by its very nature, has a lower projected density than cortical bone, and therefore makes up a smaller portion of the x_b in Equations 1.3 and 1.4. In the early stages of bone loss, cancellous bone changes can be more pronounced than cortical bone changes, but may not be detected by DXA screening [82].

Third, spurious calcification of the bones and/or adjacent structures can interfere with DXA measurements. Osteophytes (calcification of bone in non-load-bearing regions) are common in the elderly [193], and have a significant effect on aBMD readings [193]. Adjacent structures can be ossified ligaments and cartilage or calcium deposits in the aorta, which runs adjacent to the spine [215]. Calcification of these structures will increase the derived aBMD and lead to an inappropriate osteoporosis classification.

The final problem is caused by projection issues due to the size of the bone being scanned, versus the average size of the bone in the reference population. In the projection of an object, the volume increases faster than its projected area as its characteristic length increases. This can lead to bones that are significantly larger than the reference population having higher aBMD, which may be mistaken for higher BMC (see §A.1 for an example calculation).

In order to address some of these limitations, the WHO developed the fracture risk assessment tool (FRAX[®]) which can be used either in conjunction with BMD measurements, or as an independent analysis [106]. The FRAX[®] score is based on twelve risk factors and provides

Table 1.3: Inputs into the FRAX® model. The inclusion of BMD is optional [106, 231].

Risk Factor	Data
Age	Years
Gender	Male or Female
Weight	kg
Height	cm
Previous Fracture	Yes or No
Parent Fractured Hip	Yes or No
Current Smoking	Yes or No
Glucocorticoids	Yes or No
Rheumatoid Arthritis	Yes or No
Secondary Osteoporosis	Yes or No
Alcohol ≥ 3 units per day	Yes or No
Femoral Neck BMD	T-score or Z-score

a 10 year probability of spine, humerus or wrist fracture, as well as a 10 year probability of hip fracture (Table 1.3).

The FRAX® tool must be calibrated for each population for which it is used, as the weighting of each risk factor cannot be generalized. Once it has been calibrated, FRAX® can help identify those at risk of fracture. The efficacy of the tool is dependent on the probability cutoffs used. For example, if a 10 year fracture probability of 20% is used as a cutoff, nearly 30% of those who will suffer a fracture can be identified [131]. While identification of 30% of individuals is high, it is only moderately higher than the 28% that can be identified by osteoporosis score alone [220], and has been shown to be no better than using only BMD + age and neglecting the other ten risk factors [231]. A major strength of the FRAX® tool is that an initial assessment can be conducted without the need for a DXA scan, and if the individual is found to be in an intermediate risk group, a DXA scan can be performed and the person's risk level re-evaluated [107].

Keeping these limitations in mind, FRAX® and DXA provide simple, fast and useful clinical tools. DXA provides a single number (in the case of aBMD) or designation (in the case of osteoporosis state) that indicates a clinical course of action. FRAX® can be used as a pre-screening tool for DXA scanning and can refine the individual risk if DXA scanning is

indicated. The WHO recommends screening women over the age of 65, which is a significant screening burden, and while uptake is high, it is not 100% [243, p. 101]. The WHO also recognizes the fact that many individuals who are screened using these tools and found to be free from osteoporosis will still go on to fracture [243, p. 99].

When screening fails

Even for those who undergo screening, falls cannot always be avoided, and hip fracture cannot always be prevented. Depending on the fracture type, location, and any comorbidities, there are a number of treatment options available to clinicians – most of them surgical.

Patients that present with hip fracture due to fragility are typically older and female [53]. After initial assessment has indicated a possible hip fracture, anterior-posterior x-rays of the hip are used to classify of the fracture type, which guides treatment.

Fracture classification is hierarchical and helps to indicate the appropriate intervention. The highest level is intracapsular vs. extracapsular fracture, which references the fracture location in terms of whether it is located inside or outside the joint capsule (Figure 1.9). Intracapsular fractures are fractures of the femoral neck and are further classified into basicervical, cervical or subcapital fractures (Figure 1.10). Basicervical fractures occur in the femoral neck, at the junction with the trochanter; cervical fractures occur somewhere along the length of the neck; and subcapital fractures occur at the base of the femoral head.

Extracapsular fractures are sub-classified into intertrochanteric, trochanteric, or subtrochanteric (Figure 1.11). Intertrochanteric (a.k.a. peritrochanteric) fractures run between the greater and lesser trochanters, just lateral to the line of a basicervical fracture; trochanteric fractures involve only one of the trochanters; and subtrochanteric fractures involve the femoral diaphysis, distal to the lesser trochanter. The proportions of each fracture type are roughly 45% intracapsular, 55% extracapsular; however, that ratio seems to be changing, with a tendency to fewer intracapsular fractures [86].

There are two surgical strategies for hip fracture treatment: repair and replacement. Surgi-

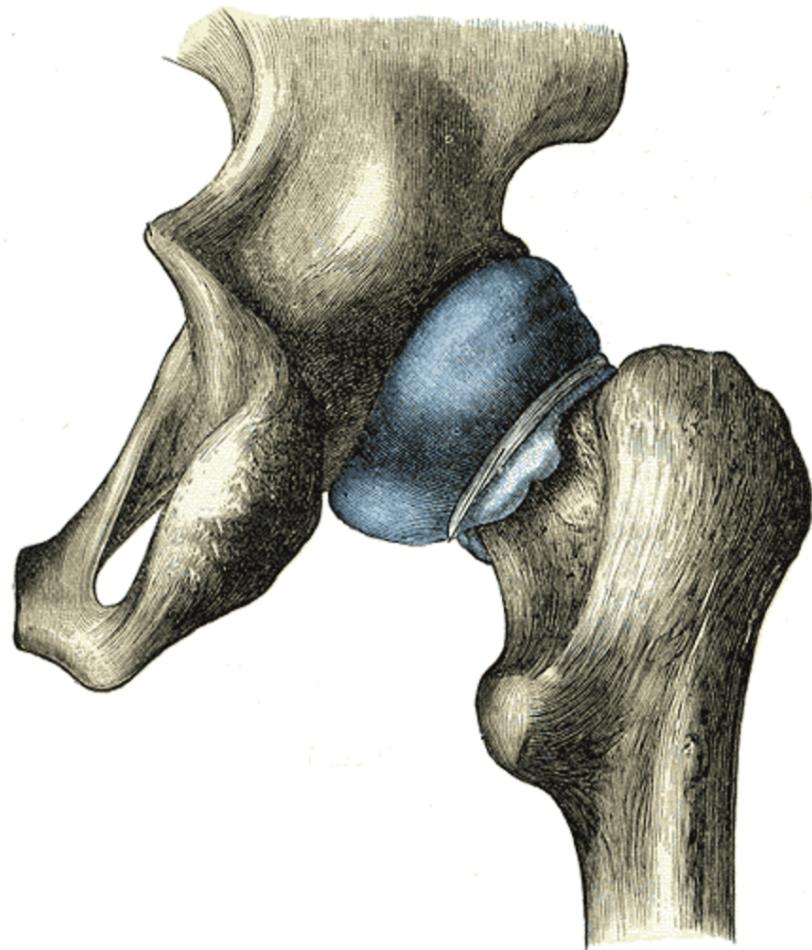


Figure 1.9: Posterior view of the hip showing the hip joint capsule. The capsule comprises the blue membrane, which has been distended in this view for visualization. Graphic adapted from Gray and Warren Harmon Lewis [79] (copyright expired).

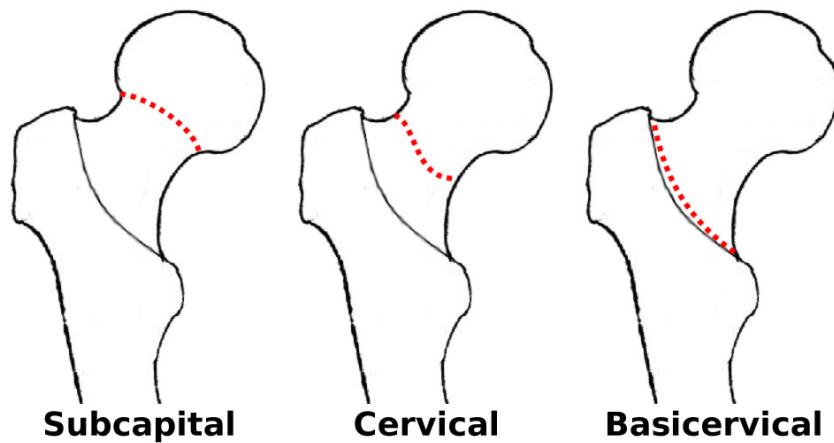


Figure 1.10: Intracapsular fractures of the proximal femur. Graphic ©Seth Gilchrist, 2013.

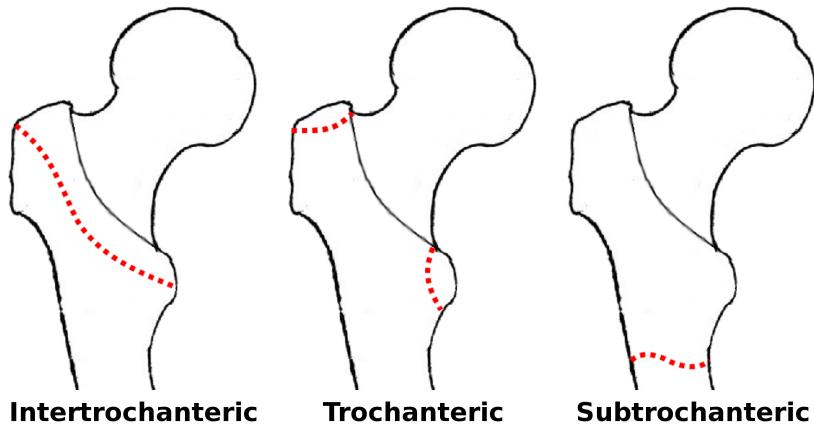


Figure 1.11: Extracapsular fractures of the proximal femur. Graphic ©Seth Gilchrist, 2013.

cal repair of proximal femur fractures involves reduction and fixation, while surgical replacement includes hemiarthroplasty, and total hip arthroplasty (THA). Reduction is the process of aligning fractured fragments, which are then secured in place using fixation, typically utilizing plates, rods screws and/or nails. Reduction can be either *open*, in which the fragments are dissected, and oriented manually, or *closed*, where external pressure and traction are used to align fragments. THA involves resection of the femoral head and neck, reaming of the acetabular cup, and replacement of these components with a stem and ball on the femoral side and an articulating cup on the pelvis side. In hemiarthroplasty, this procedure is done only on the femoral side and the acetabular anatomy is left in place.

The indications for the type of surgical solution can be quite complex [32, 210]. They rely on the location of the fracture [31, 58], the angle of the fracture line [31], the age of the patient [19, 209], time since fracture [31, 58], and any comorbidities [19, 209]. Depending on the severity and type of comorbidities, surgery may not be an option, in which case non-operative treatment may be required. Non-operative treatment of hip fracture has shown poor outcomes and is not common [93]. It requires an extended period of non-weight bearing (~6 weeks), which can exacerbate existing bone quality issues, and has been seen to have poorer long-term outcomes compared to repair or replacement strategies [93].

Hip surgery has a high success rate [19, 32, 93], and is the preferred treatment option

for hip fracture. That said, hip fracture is a very serious injury and can result in a decrease in quality of life, extended morbidity, and even mortality. Wiktorowicz et al. [244] found 38.4% of community dwelling individuals who suffered hip fractures were discharged into either assisted living (27.1%), or died (11.3%). This is corroborated by another study that found 26% of community dwelling individuals were discharged to assisted living after hip fracture [45]. Additionally, hip fracture patients spend more time in assisted living than age-matched controls – an average of 237 more days than a typical 80 year old [30].

Mortality following hip fracture is also high. While death generally isn't directly attributable to the fracture, hip fracture patients fare much worse than their age-matched controls due to a range of systemic deleterious health effects [45]. In the three months following hip fracture, patients have a mortality odds ratio (OR) of 2.8 over age-matched controls [195]. This gradually decreases back to an OR of 1.4 at two years, but mortality rates remain higher than age-matched controls for at least five years post fracture [45]. In actual percentages of patients, one year post fracture, up to 21.6% of patients have died [244].

This section has discussed the clinical screening, treatment and discharge paths of hip fracture patients. The treatment and discharge sections show just how important screening is. Research conducted to improve screening efficacy is of clear benefit to patients and society. Only with accurate screening abilities will we be able to concentrate our prevention efforts – be they pharmacological, lifestyle or surgical strengthening – on the most vulnerable people.

1.4 Mechanical behaviour of bone

Before addressing hip fracture test methods and models, the mechanical behaviour of the materials that comprise the proximal femur need to be understood. In Section 1.2 I discussed the anatomy of the femur and the bone were discussed. This section builds on that knowledge by outlining the mechanical behaviours of these materials based on anatomical and morphological differences. Following this discussion of the mechanical behaviours, the different methods for modelling hip fracture will be discussed, in which the material level behaviours will be

Table 1.4: Approximate cortical and cancellous bone material properties [141, 194, 196].

Longitudinal is along the principal material direction, which is defined by the Haversian system in cortical bone and the first Eigenvector of the MIL tensor in cancellous bone. Transverse is perpendicular to the Haversian system in cortical bone, and along the third Eigenvector of the MIL tensor in cancellous bone.

Modulus (GPa)	Direction	Bone Type	
		Cortical	Cancellous
Tensile	Longitudinal	17.9	2.0
	Transverse	10.1	0.2
Compressive	Longitudinal	18.2	9.5
	Transverse	11.7	5.5
Shear	Transverse	3.5	0.017

important.

The bones of the body are complex organs that are made up of multiple materials serving a variety of functions. The principal load bearing material is named after the organ itself and is called *bone*. Bone material is inhomogeneous, anisotropic and viscoelastic. This means that its mechanical properties vary with location, loading direction, and loading speed. These facts make determination of bone mechanical properties challenging.

Mechanical characterization of bone is done using traditional materials testing methods in which specimens are formed into regular geometries and stretched, compressed or sheared. Cortical bone is typically milled into dog-bone specimens for tensile testing, and either cuboidal or cylindrical specimens for compression testing [55, 66, 194]. In cortical bone, the “orientation” of the specimen is determined by how the loading axis is aligned with the axis of the predominate Haversian system. Bone is a linearly elastic material, and as such can be characterized in the elastic loading regime using the modulus of elasticity. The modulus of elasticity of cortical bone varies depending on the orientation of loading (Table 1.4). It is highest in the longitudinal direction (aligned with the collagen fibrils), and lowest in the transverse direction.

Cancellous bone is typically milled into cuboidal or cylindrical specimens for testing [95], with the orientation often given in a site-specific manner [37, 73, 167]. For example, the testing direction for cancellous bone of the femoral neck will be given as parallel or perpendicular to

the femoral neck axis. It is becoming more common to also report the orientation with reference to fabric properties, such as MIL [196]. Using the anatomical reference, longitudinal loading is along the direction of the site-specific principal length (e.g., along the femoral neck); however, using the fabric properties, longitudinal loading is along the first Eigenvector of the fabric tensor (Figure 1.8). Transverse loading is perpendicular to the longitudinal direction. When used with site-specific reference the term *transverse* does not give any information regarding the direction in the plane perpendicular to the principal length in which the loading is taking place. For example, transverse loading of a femoral neck specimen would be radial loading of the femoral neck bone; however, the direction of the radial (e.g., anterior-posterior) would not be given. When fabric properties are used, researchers will normally give the Eigenvector along which they are loading, rather than using term transverse. For example, a specimen subjected to transverse compression would be loaded along the second or third Eigenvector.

The modulus of elasticity and strength of cancellous bone react to changes in orientation similarly to cortical bone (Table 1.4). Loading along the first Eigenvector of the MIL tensor yields the highest modulus and strength. Loading along the third Eigenvector of the MIL tensor yields the lowest modulus and strength. In addition to the material orientation variations, the density of the cancellous bone influences the strength, stiffness and post yield behaviour [37, 196].

Both cortical and cancellous bone have been shown to be viscoelastic, meaning that their material properties change with strain rate. In both types of bone, stiffness has been shown to increase with strain rate [36, 37, 52, 55, 66, 190]. In cortical bone loaded in compression, stress at a fixed strain shows a logarithmic relationship with strain rate [157], following the equation $\sigma_\varepsilon = 4200 \log(\dot{\varepsilon}) + 33,000$ where σ_ε is the stress at a given strain, and $\dot{\varepsilon}$ is the strain rate. Loaded in tension, the modulus of elasticity of cortical bone shows much less variation, being almost invariant with strain rate up to strain rates of $100\varepsilon/\text{s}$ [52].

For cancellous bone in compression, researchers have shown that stiffness follows a power-law relationship with strain rate [37, 134]. In one set of tests, the stiffness of de-fatted can-

cellous bone was proportional to $\dot{\varepsilon}^{0.06}$ [37], and in another it was found to be proportional to $\dot{\varepsilon}^{0.047}$ [134]. Importantly, this relationship broke down when the marrow was left *in situ*. When compression tests were performed on cancellous bone cores with marrow and fat still present, there was a sharp increase in stiffness at the highest strain rates tested (10/s) [37]. There were too few specimens in this group to obtain a good fit, but qualitatively an increase in stiffness of about 3x can be seen in the plots presented by Carter and Hayes [37]. High-rate tension testing of cancellous bone has never been done, so the nature of the relationship is currently unknown.

Generally speaking, cortical and cancellous bone can be said to be stiffest in compression along their principal directions, weakest in shear, with tension in the middle. Additional considerations for cancellous bone are that stiffness can vary with density and structure. Effective measures of bone behaviour will take into account the type of bone, the amount present, its orientation to the loading vector, and the rate of force or strain application.

1.5 Determinants of bone strength

Bone strength refers to the ability of the material to support a given load. While this definition may sound simple, examination of the details yields a more nuanced view. As a multiphase, polymorphic, composite material, bone has the ability to behave in a highly ductile manner, a highly brittle manner, or anywhere in between [91]. Cortical bone tends to behave in a brittle manner, and as such, ultimate load or stress is often given as the strength of the bone [91, 194]. Cancellous bone is more ductile in its failure modes, and ultimate properties would not tell as complete of a story. For this reason, cancellous bone failure information is often given as both yield and ultimate properties [141, 196, 229, 245].

The strength of cortical bone is most closely linked to loading direction (Table 1.5). Cortical bone is stronger in compression than tension, and weakest in shear [194]. As with the measurement of modulus of elasticity, the organization of the lamellar bone into linear Haversian systems gives the failure properties of cortical bone a distinct directionality. Changing orientation from longitudinal to transverse decreases the ultimate stress, but increases the ultimate

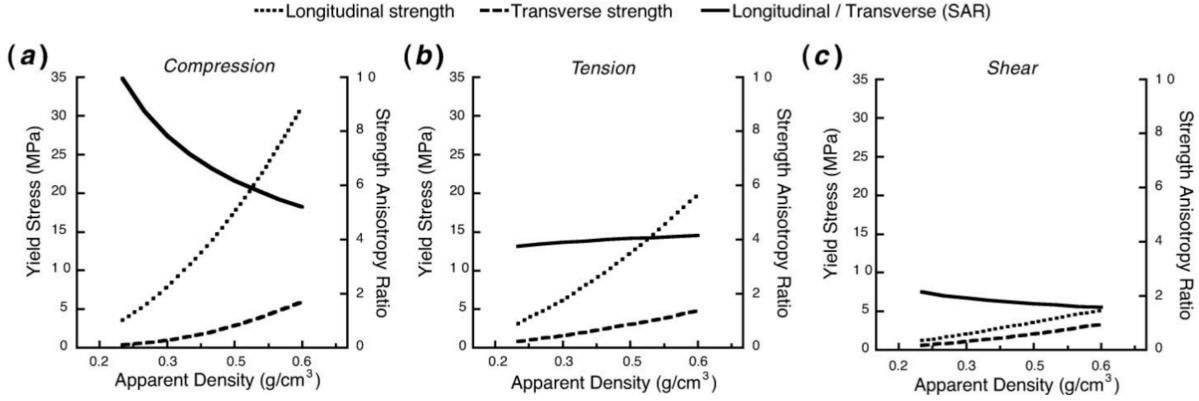


Figure 1.12: Multiaxis and multidirectional testing of trabecular bone shows the large range of properties based on direction and density. The strength anisotropy ratio (SAR) is the ratio of the longitudinal and transverse metric in each graph. An SAR different from unity indicates a different behaviour between the two directions. Graphic from Keaveny et al. [109] (with permission).

strain (note that Reilly and Burstein [194] reported difficulty in determining exact compressive failure strain in the transverse direction, which may have led to the high value shown in the Table 1.5).

The failure behaviour of cancellous bone is more complex. The primary determinants of cancellous bone failure are loading direction in relation to the principal fabric directions and density. Table 1.5 gives approximate values for failure in different modes, but differences in bone and trabecular density can change any of the numbers by a factor of 5 or more [37, 78, 175] (Figure 1.12). In addition to density, the structure and composition of the trabecular network has been shown to be significant. Bone with more plate-like trabecular members (as measured by structure model index (SMI)) has been shown to be stronger than bone with rod-like members; connective density (ConnD) has been shown to correlate with increased strength; and bone surface density (BS/TV) has also shown to be a predictor of strength [166].

When loaded in compression, cancellous bone behaves in a ductile manner. It follows a stress-strain profile that is common for cellular solids, which begins with a linear, elastic region, followed by a strain softening region, and finally densification, where the pores have been removed and the bone material itself is being compressed (Figure 1.13) [37, 73]. This

Table 1.5: Approximate cortical and cancellous bone failure properties [141, 194, 196, 229, 245].

Measure (MPa)	Direction	Bone Type		
		Cortical (Ultimate)	Cancellous (Yield)	Cancellous (Ultimate)
Tensile Stress	Longitudinal	135	3.4	4.0
	Transverse	53	2.0	—
Tensile Strain	Longitudinal	3.1	0.68	1.4
	Transverse	0.7	—	—
Compressive Stress	Longitudinal	205	9.0	10.2
	Transverse	131	1.0	—
Compressive Strain	Longitudinal	1.9	1.5	2.3
	Transverse	5.0	—	—

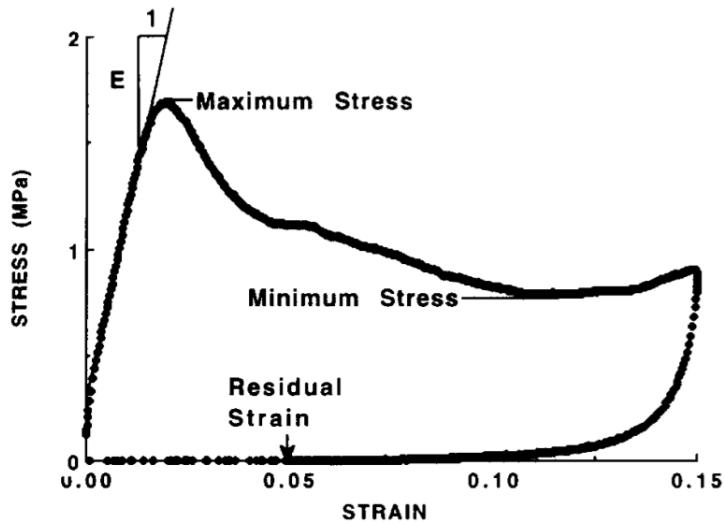


Figure 1.13: Compressive stress strain behaviour of cancellous bone showing the elastic, strain softening and densification regions. The bone samples recover to at least 96% of their original height after compression to 85% of their original height. Graphic from Fyhrie and Schaffler [73] (with permission).

same profile is not seen for tensile loading, in which separation of the trabecular members after fracture allows the stress to drop to zero [38]. Interestingly, cancellous bone has also been shown to be able to recover a large portion of its original height after compression past yield [73]. When subjected to 15% strain, human vertebral bone recovered to at least 96% of its original height – a rebound of nearly 75%.

Along with its viscoelastic properties, bone also has rate-dependent failure modes. There is some discrepancy between studies as to the effects of strain rate on the failure of cortical bone. Early research indicated that increasing strain rate increased the ultimate strength and decreased the ultimate strain [36, 52, 247], making the bone more brittle. More recent research has indicated that the situation may be more complex, with post yield behaviour showing a more quadratic shape [55, 66, 87, 251] in which ultimate stress and strain increase with strain rate at moderate rates, then decrease at high rates. Additionally, one research group found that while cortical bone brittles at high rate, it also weakens (however, they did not measure strain rate directly, reporting only test machine cross-head speed) [190].

The differences seen in the viscoelasticity tests may have to do with the changes in fracture

toughness associated with high testing rates. Fracture toughness is the ability of a material to resist crack growth and is important in brittle fracture. Materials with a high fracture toughness tend to fail in a ductile manner, rather than through crack growth. Researchers examining tensile fracture toughness of cortical bone found that yield strength increased with increasing strain rate, but at the same time, fracture toughness decreased [121]. A follow up study found that crack growth resistance (called the *R*-curve) was lower in high rate tests due to a lack of microcracking, ligament bridges and plastic zone formation [122]. The differences seen in post failure behaviour of cortical bone may indicate that there are complex reactions to loading and displacement rates; however, these results could also be confounded by the lack of a standard testing protocol. *R*-curve behaviours are known to be sensitive to specimen geometry [74, Chapter 5], which could have contributed to the differences seen in the previous literature.

Failure rate effects of cancellous bone have shown more agreement between researchers, with increased strain rates moderately increasing the ultimate stress and strain [37, 134]. The relationships were fairly weak, with ultimate strength showing power-law relationships of $\sigma_{ult} \propto \dot{\epsilon}^{0.073}$ [134] and $\dot{\epsilon}^{0.06}$ [37], and ultimate strain showing a relationship of $\varepsilon_{ult} \propto \dot{\epsilon}^{0.03}$ [134]. However, there was a significant limitation that may reduce the validity of the correlations *in-vivo*: the power-law relationships were determined using specimens with marrow removed. Other tests performed with marrow *in-situ* showed that the power law relationships, like the ones discussed in §1.4, held for low to moderate strain rates. However, the relationships diverged from the power-law fit quickly at higher strain rates, with strength increasing by approximately 4x between strain rates of 1/s and 10/s [37].

Failure behaviour of bone, similar to mechanical behaviour, is influenced by bone type, loading direction, and loading rate. The anisotropic behaviour of bone material makes the response of a structure made of multiple types of bone (e.g., the metaphysis of a long bone) too difficult to predict. To address all of these factors, accurate and complete experimental models, reproducing the loading conditions *in-vivo* as closely as possible, are needed.

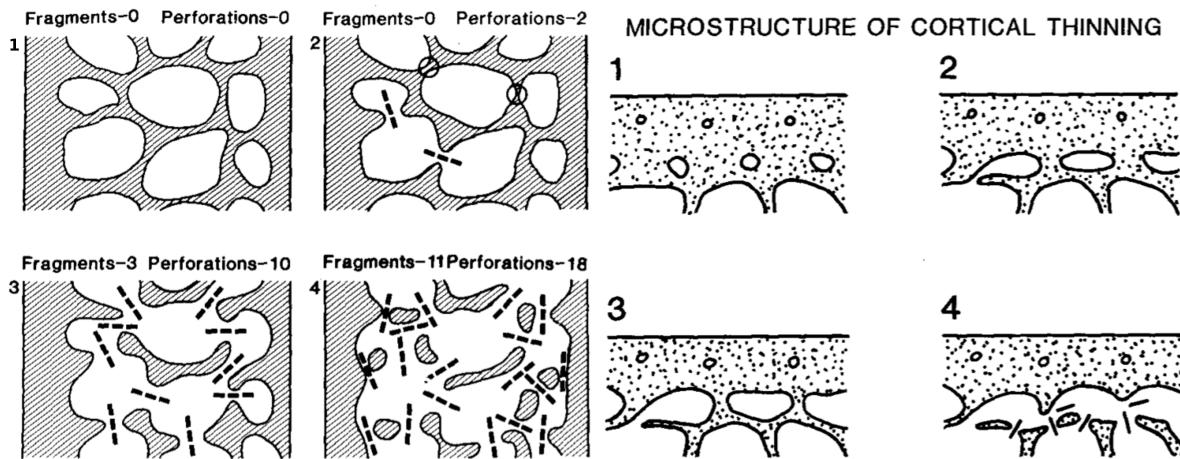


Figure 1.14: Left: Cancellous bone is progressively thinned and perforated with age, resulting in less connectivity and decreased volumetric density. Right: Cortical bone is trabecularized on the endosteal surface, which is then subjected to the same processes as cancellous bone, resulting in a thinner cortex. Graphics from Parfitt [185] (with permission).

1.6 Age related changes in bone

A final aspect of bone behaviour that is of particular importance to hip fracture are the changes that occur during the normal course of ageing. Throughout life, bone is remodelled on a continuous basis, with old bone being absorbed by cells called osteoclasts and new bone being laid down in its place by osteoblast cells. This process, while normal and healthy, can be subject to errors that decrease the quality of the bone structure. Additionally, the properties of the new bone change as a person ages, with bone from older people exhibiting lower strength and toughness. The cortex undergoes a process of thinning and trabecularization, and trabecular structures (including the trabecularized cortex) are thinned and perforated (Figure 1.14) [185]. The amount of bone (measured using DXA) changes in a site and gender specific manner [35], and the quality and type of bone changes, leading to an overall decrease in strength [156].

Research investigating the quality of bone throughout life has shown that, after reaching adulthood, cortical and cancellous bone stiffness, ultimate stress, and ultimate strain decrease with age [48, 155, 156, 162, 176, 250]. Work on cortical bone has shown that increased age correlates with a decrease in fracture energy [155, 250], fracture toughness, and energy per

unit of crack growth [250]. Yield strain was seen to be invariant with age [155, 176], but the decreases in ultimate stress and strain led to a decrease in energy to failure [155]. These data are indicative of a general decrease in bone quality with age, which may be linked to an increase in micro-porosity and degenerative remodelling [155].

Remodelling is the name given to the process of bone modification throughout life. It is a multi-step process that is thought to begin with a stimulus to remodel, such as micro-cracking or disuse, which induces osteoclast cells to remove mineralized bone (known as bone resorption). Following resorption, osteoblast cells fill in the cavities created by the osteoclast cells with new bone. In the final stages of remodelling, the osteoblasts are either enclosed by bone and turn into osteocytes, or are on the surface and turn into lining cells. When the process is proceeding normally, it is driven by mechanical stimulation and leads to a bone structure that reflects the directions of the principal stresses. This process, known as Wolf's law [246], is thought to influence the structure of many bones; however, its exact mechanism and relationship to stresses and strains remains somewhat controversial [214]. Degenerative remodelling occurs when structures that are important for load bearing are altered in such a way that their ability to resist load and absorb energy is affected.

Parfitt [184] investigated the remodelling of cancellous bone and found that there is dimensional overlap between the depths of cavities created by osteoclast cells during resorption and the thickness of a trabecular member. This means that during resorption some trabecular members will be disconnected and eventually fully resorbed, decreasing connectivity and potentially bone density. These disconnected trabeculi cannot be reconnected by osteoclast cells, and the damage becomes permanent and degenerative in nature. This process is referred to as the *resorption hypothesis* which states that bone structure quality decreases with age due to errors in the resorption portion of remodelling.

Compressive testing of cancellous bone along the weight bearing axis showed a decrease in ultimate stress with increased age [156]. Associated with this change was a decrease in apparent density, surface density, trabecular plate thickness, and trabecular plate density, as well

as an increase in trabecular separation and a decrease in ConnD (as measured using “objects per field”) [156]. These changes are consistent with the resorption hypothesis. Research on the trabecular bone of vertebral bodies showed a decrease in the number of horizontal trabeculae, and an increased thickness of vertical trabeculae with age [162]. Combining this result with the findings of Fyhrie and Schaffler [73], who showed that horizontal trabeculae fracture and absorb energy during axial loading, one can speculate that the decrease in horizontal trabeculae would result in a decrease of post yield energy absorption.

These changes in bone throughout life directly influence a person’s susceptibility to hip fracture. While we can observe the nature of the changes, researchers have yet to draw a direct causal link between any of the specific changes and hip fracture risk. Increased understanding of the roles of these physical and behavioural changes could allow researchers to devise new screening techniques which would be driven by bone structure, rather than bone quantity.

This section discussed the material and mesoscale structural changes that occur in bone with age. The way these changes manifest themselves in whole bone behaviour is somewhat unpredictable as the geometries of whole bones are complex and subject to considerable variation. The next section discusses human injury and experimental methods for quantifying its likelihood. Following that, methods for modelling hip fracture, both experimentally and computationally will, be addressed.

1.7 Understanding fractures of the proximal femur

On the surface, the problem of hip fracture appears fairly straightforward: a load is applied to the bone which exceeds its load bearing capacity. That said, one does not have to try very hard to find examples that contradict this notion. For example, in previous sections we have seen that aBMD is a strong determinant of hip fracture on a population level, yet when we look at the fracture and non-fracture clinical populations, there is considerable overlap in aBMDs [67, 80]. Additionally, clinical admission rates show that only 28% of those admitted with hip fracture had osteoporosis and 51% had osteopenia [220]. To understand what additional aspects play

a role and influence the fracture, researchers have produced models that aim to replicate the forces at the time of injury, allowing observation and study of the failure progression, and testing of bone parameters for importance. In addition to identifying important bone parameters, the models have aided in the development of fracture tolerances that are used to evaluate potential biomechanical interventions like hip protectors and compliant floorings.

1.7.1 Human injury tolerance

Evaluation of human injury tolerance is a large field, encompassing hard and soft tissues of all types, with many different failure mechanisms and definitions of tolerance. The injury tolerance of a whole bone is typically given as an ultimate force, whereas for soft organs it is more often given as a strain (in the case of connective tissues, e.g., liver), or acceleration (in the case of neural tissue). On a tissue level, injury tolerance is almost always given as a strain to failure. Regardless of what is used, research is required to identify the appropriate metric and how it should be applied.

Generally speaking, injury tolerance can be done at two levels: organ and tissue. Organ level injury tolerance is determined using an isolated organ of interest [48], or a whole cadaver from which the organ can be dissected after an injurious event [40]. In this sense, a whole or partial femur would constitute an organ. The lab environment is used to replicate the loading conditions of an injurious event, during which researchers can observe either the injury progression, or a binary injury/no-injury classification.

Tissue level injury tolerance is determined using isolated tissue samples, on which traditional mechanical tests can be performed, similar to what was discussed in §1.4 and §1.5. Examples of these tests would include making dog-bone or other geometry specimens of bone for testing in a materials testing machine [251], and dissecting liver tissue for shear testing in a rheometer [136].

The organ and tissue level methods have different applications. Organ level tolerances are typically applied to injury scenario evaluation and device design and evaluation. For example,

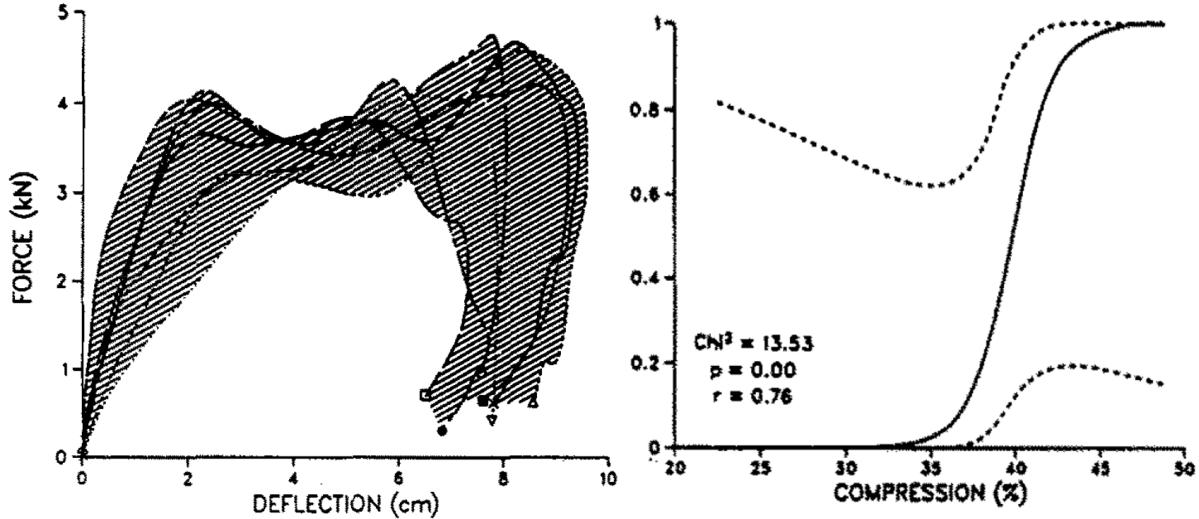


Figure 1.15: Response corridor and sigmoid injury curve for the human chest subjected to an impact on the sternum. The force-displacement corridor (left) shows the range of responses seen in the tests, and the sigmoid curve (right) shows the likelihood of injury based on percent chest compression. Graphics adapted from Viano et al. [238] (with permission).

design and evaluation of a collapsing steering column in a car to prevent chest and trunk injuries may use the frontal loading tolerances of the body [238] as a design criteria. The frontal loading tolerances are taken from laboratory tests in which the whole cadavers (i.e., organ level samples) are subjected to loading replicating a frontal collision, with observations of force and fracture. These injury data are then processed into two evaluation metrics: i) a sigmoid probability curve for failure as a function of force or deflection [15, 238], and/or ii) a corridor of force response as a function of time [238] (Figure 1.15). The sigmoid curves can be used to evaluate if a new design will prevent injury, and the corridors are used to evaluate if a new test or model is providing reasonable or similar results.

Tissue level tolerances are typically used in computational (i.e., finite element (FE)) modelling. In FE models, a load is applied to a computer simulation of a structure and determination of failure is made using the highest measured stress or strain within the structure – information that is not available in organ level laboratory testing. The material properties for the model (e.g., the elastic modulus and anisotropy) are taken from laboratory tests of isolated biological materials (such as those discussed in §1.4), as are failure metrics like stress and strain to failure

(like those discussed in §1.5). Often, the FE model force-time and failure responses are compared to the force-time or force-displacement corridors from an organ level test to evaluate the validity of the model.

The tests discussed in this thesis were organ level tests of isolated bones. They compared the mechanical responses of the whole bones in two different loading conditions and used this information to gain insight into material level behaviours in each condition. This technique is unique in that it aims to identify tissue level behaviour changes while testing at the organ level. This is justified due to the complex nature of bone tissue behaviour. As described in §1.4 and §1.5, bone tissue behaviour depends on many properties that vary with loading direction, loading rate, and density, all of which vary spatially (and not necessarily with C^2 continuity), making tissue level determination of the material response to a given organ level load very difficult.

1.7.2 Modelling falls to the side

Modelling the response of the proximal femur during a fall to the side has been done using two modelling methods, both of which were mentioned in the preceding text: laboratory and computational. Laboratory models are done in two ways: the first is using surrogate biological material such as isolated bones or cadavers; the second uses biofidelic materials, such as custom plastics, rubbers and foams, which are designed to mimic the geometry and response of a real organ under certain conditions. In bone testing, the most common biological surrogate is an isolated specimen of the bone being examined, obtained post-mortem from consenting donors. The most common biofidelic surrogates are foam-filled plastic bones, such as SawbonesTM (Vashon, WA) which were developed to have representative behaviours in the elastic stress-strain region, but are not to replicate failure behaviours.

Computational models are numerical representations of structures and loadings and there are two general forms which can be implemented depending on the information desired. The most common computational model is the finite element (FE) model, in which external loads

or displacements are applied to a discretized representation of a structure, and the stresses and strains are calculated for all points within the structure. Another kind of computational model is the multibody dynamic (MBD) model (e.g., van den Kroonenberg et al. [232]), in which forces and motions are described, but internal stresses and strains are not. Sometimes a mixed model is used, in which an MBD model is used to determine forces between bodies that are not of interest, and an integrated FE model is used to analyse a structure of interest.

While it is not the intent of this thesis to cover both laboratory and computational modelling, an understanding of each is important to understand the basis of the methods and the rational for certain decisions that I made in the course of this research. The experiments designed in the process of completing this thesis used information from both techniques, so a thorough understanding of their limitations and interactions is essential for the context of this work. Additionally, data and analyses from the experiments were documented and gathered in a way that was specifically meant to foster collaboration with the FE modelling community. More information on that process is given in Chapter 6.

One aspect that defines the interaction of laboratory and computational models is validation. Validation is the process of affirming that the model provides meaningful results in the context in which it is being used. For laboratory models, this means that the boundary conditions (BCs) replicate the loading scenario in question and produce results that are consistent with what is seen in real life. This involves a conceptual design phase, in which the BCs are evaluated and aspects such as posture, impact direction, and impact location are determined, and a verification phase in which the injury produced by the model is compared to injuries seen by clinicians. For FE and MBD models validation means that a laboratory test has been performed that has very similar – if not identical – BCs and gives similar results in terms of the desired outcome variable.

It is important to note the limits of validation for each type of model [6]. Laboratory models are valid only for the *specific scenario for which they have been validated*, and computational models are valid only for the *outcome variables that have been validated*. The first limitation

is perhaps intuitive: if one designs a model of a fall to the side, the same model cannot be used to evaluate a bicycle crash as the loading directions and magnitudes are different. The latter limitation is a bit more difficult to grasp and apply but is important to understand. An example would be a computational model designed to replicate force for a given impact. This model gives accurate force-time traces for different impacts, but may – or may not – give an accurate displacement. The displacement would have to be independently validated to determine if it is accurate.

For these reasons, laboratory models tend to be more versatile in terms of identifying what independent variables are important, because for a given input, an arbitrary response of the specimen can be examined. On the other hand, computational models are versatile in determining the effects of different inputs on a given output variable, because for a fixed output a more-or-less arbitrary input can be examined (as long as one stays within the limits of the material property validations).

1.8 Laboratory models of hip fracture

Laboratory models of hip fracture are done in two ways, testing two different fracture scenarios. The first is a standing model, in which the forces are applied in a way that reproduces the forces of walking and other mobility-related activities [97, 105, 112, 137]. These models are meant to investigate spontaneous fractures, i.e., fractures that happen due to the fragility of the bone being such that failure occurs while conducting normal daily living activities. The second type of model is one that is meant to replicate the forces on the proximal femur in a fall to the side [7, 16, 23, 28, 41, 42, 46, 47, 57, 63, 71, 94, 112, 129, 137, 138, 148, 149, 177, 189, 192, 221, 230, 242]. These models are meant to investigate traumatic fractures which occur in part due to fragility, but with the stimulus of an impact event. The model discussed in this thesis is a sideways fall model, and the discussion of laboratory models will focus on research conducted on the same.

The discussion of the model will be broken down into three aspects: (i) orientation,

(ii) constraint, and (iii) method of force application.

Orientation and constraint are geometric variables that were initially determined through trial and error, and more recently through motion tracking of human volunteers in a fall. The method of force application has seen little change over the years and has been done using either displacement control in a materials testing machine or, in three cases, inertial impact.

1.8.1 Development of orientation and constraint

Orientation of the bone in a fracture model is important because of the anisotropic and inhomogeneous nature of bone material. As discussed in §1.4 and §1.5, the mechanical behaviour and failure characteristics of bone depend on loading direction. Incorrect orientation could artificially change the load path, thereby altering the material behaviour as locations of high deformation and high strain rate change.

The first laboratory hip fracture model was developed in 1957 by Backman [7]. In this work, dried femora were subjected to displacement-controlled and impact loading from a variety of different directions, with the fracture pattern as the outcome variable. In the 30 years that followed the work of Backman [7], hip fracture research was conducted primarily using clinical data and methods. Conducting and publishing *ex-vivo* experiments resumed in the 1990's, at which time a standard orientation was proposed, and modelled of the work of Backman [7]. By selecting the orientation that resulted in the most clinically relevant fracture patterns, an initial guess of the orientation of the femur at impact was obtained. The orientation was defined such that, when viewed from the direct posterior, the femoral shaft and femoral neck both made a 30° angle with the applied load [138].

Shortly after this orientation was used for the first time, video data of volunteer fall subjects became available [233], which led to the reduction of the adduction angle from 30° to 10° , and the redefinition of neck position as 15° internal rotation from the load vector [46] (Figure 1.16).

The internal rotation angle of the femoral neck is difficult to determine from fall videos, and without a good reason to change it, it has remained at 15° . That said, there is some evidence

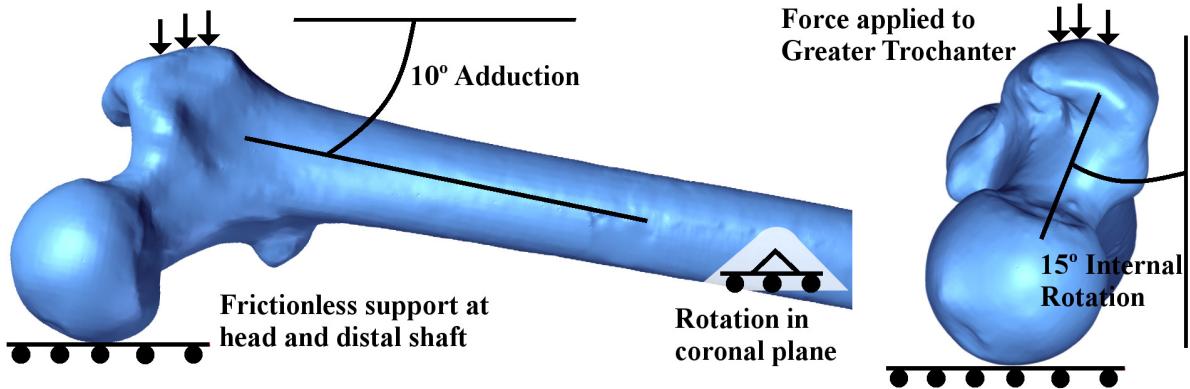


Figure 1.16: Constraints and orientation of the proximal femur in laboratory testing. Anterior view (left) and superior view (right). Graphic ©Seth Gilchrist, 2013.

that this angle is valid. Anatomical data [227] and more recent fall video analysis [68] can be used to show that the assumed internal rotation angle from the impact vector is likely to be close to the actual angle from the impact vector, or at least within 1 standard deviation (see §A.2).

Constraint of the femur has also remained the same since it was defined in detail in the early 1990's [46, 138]. The standard constraints consist of free rotation about the distal end of the shaft in the coronal plane and free motion in the sagittal plane, with all other degrees of freedom removed (Figure 1.16).

1.8.2 Application of force

The method used to apply force to the bone in a laboratory model can influence fracture because it changes loading and displacement rates. The resulting loading and displacement rates could, in turn, affect the material properties due to the viscoelastic nature of bone, as discussed in §1.4. As the material in the bone fails, a progressive change in loading path and strain rates could then change the mechanics of the failure as discussed in §1.5, such that erroneous force application could change the resulting failure and fracture patterns.

Force is applied to the lab models using either a constant displacement rate, or in some cases, an impact using a dropped mass. Backman [7] did not see any difference in fracture patterns between bones loaded at a constant compression rate and those loaded by impact. Ad-

ditionally, work examining bone tissue response to loading rate [36, 55] showed a weak influence of strain rate to mechanical properties. These two facts led early hip fracture researchers to use low, constant displacement rate at the greater trochanter [138].

Knowing that bone material is viscoelastic, researchers saw the application of the force using a low displacement rate as a possible weakness of the testing method, leading them to investigate the effects of increasing the displacement rates. Courtney et al. [46] used a materials testing machine to apply constant-rate displacements on donor-match pairs of femurs at 2 and 100 mm/s. This work showed that increasing the displacement rate significantly increased the stiffness and energy to failure, and moderately increased the load to fracture. In the years following this paper, many experiments have been carried out using a constant displacement rate of 100 mm/s at the greater trochanter [57, 148, 149, 221]. Displacement rate was not seen to influence positive correlation with aBMD, and results from lower rate tests are still considered valid. Interestingly, the effects seen by Courtney et al. [46] were much stronger than would be predicted by isolated bone testing alone (§1.4). If one assumes that the strain rate is directly proportional to displacement rate, one might expect an increase in displacement rate of 50x to increase the stiffness by approximately 20-30%, and likely decrease the energy to failure [37, 134, 157]. Courtney et al. [46] found that femurs compressed at high displacement rates were approximately 100% stiffer than those compressed at low rate and absorbed more energy to fracture, further illustrating the complexity in deriving organ level behaviours based on the behaviour of tissues.

Not all research has been conducted using constant velocity. Three researchers [7, 177, 242] have utilized impacts to apply loads to specimens. Backman [7] showed no difference in fracture pattern between impact and constant displacement rate loading. However, his experiments were performed on dried femora and lacked the ability to accurately measure force or displacement, so were unable to evaluate biomechanical response to the different loadings. Weber et al. [242] used an impact at 4 m/s with an energy of 360 J (approximately 2-3x the current estimated energy in a fall to the side [68, 201]) and found similar results to Courtney et al. [46], in

which all mechanical metrics increased as displacement rate increased. Okuzumi et al. [177] investigated the effects of different hip protectors on the failure strength of proximal femurs using a dropped mass. However, the impact mass, velocity and energy were not biomechanically justified and not compared to more traditional methods, so the influence of this protocol on the outcome is unknown.

1.8.3 Lessons learned from laboratory models

As mentioned in §1.7.1, once a laboratory model is validated for a certain scenario it can be used to examine any independent variable such as tissue displacement, force response, or yes/no injury classification. Taking advantage of this aspect of laboratory models, many different properties of the proximal femur loaded in the fall configuration have been examined (Table 1.6).

Orientation has been shown to influence femoral strength, with an increased internal rotation relating to a decrease in the strength of the specimen [71, 189]. Changing the orientation from a standing model to a side impact model showed a difference in one study [112], but not a second one [137]. However, these two studies used different orientations to model a side impact, with Keyak [112] using a rather extreme neck internal rotation of 70°, which would be predicted to significantly lower the strength of the specimens [71, 71, 111, 189].

BMD has been included as a variable in many tests, and the data have consistently shown that a higher BMD (or aBMD) leads to an increase in stiffness and ultimate load.

The independent roles of cancellous and cortical bone are known to some degree, but their relative roles are a bit more difficult to determine. Cancellous bone has been shown to be important in a fall [148, 149, 192], and less important in standing strength [97]. Researchers who have compared the relative roles of cancellous and cortical bone in failure scenarios have shown that both have important roles to play, with relative importance varying by failure location [149, 192].

A decrease in strength of specimens loaded in a fall configuration has been shown to be

Table 1.6: Important results and references for laboratory tests.

Aspect	Influence	Reference(s)
Orientation	Increased internal rotation lowers ultimate strength. Standing vs. falling orientation changes strength, but ratio of the strengths depend on the definitions of “standing” and “falling”.	[112, 137, 189, 230] [71, 111] ¹
Compression rate	Increased compression rate leads to increased stiffness, ultimate load and energy to failure.	[7, 46, 177, 242]
BMD and aBMD	Higher BMD leads to increased strength.	[23, 28, 41, 42, 129, 137, 138, 148, 149, 192, 197]
Cancellous bone	Cancellous bone important in a fall, less so in standing.	[21, 97, 148, 149, 192]
Cortical bone	Thin cortex is indicative of reduced strength.	[21, 49, 56, 57, 137, 148, 149, 154]
Age	Increased age relates to decreased BMD and strength.	[46, 47]
Side	Side-to-side strength in the same person differs on average by 19% and 16% for women and men, respectively.	[63]
Geometry	Certain geometrical features, such as a long or narrow femoral neck, relate to increased fracture risk.	[41]
Failure location	Failure typically initiates in the anterior-superior neck.	[56, 57]
Femoroplasty	Augmentation by femoroplasty (i.e., injection of bone cement to strengthen cancellous bone) increases strength of osteoporotic femurs, but makes revision surgery difficult or impossible.	[16, 56, 94, 221]

¹Computational studies

correlated with increasing age; however that change was concurrent with a decrease in BMD, which may have accounted for the decrease in strength [46, 47].

Side-to-side strength in the same donor was seen to be 19 ($\pm 13\%$) (mean (\pm standard deviation (SD))) different in females and 16 ($\pm 12\%$) different in males [63]. This result is important when considering the results of many of the comparative studies listed in Table 1.6, because results from those studies were often analysed using matched-pairs statistics on donor-matched pairs of femurs. Eckstein et al. [63] showed that matched-pairs statistics on donor-matched (left vs. right) femurs is likely insufficient for statistical significance, as there is considerable variability in strength that is unexplained by BMD.

Geometry has been thought to be a promising clinical screening tool, as it can be evaluated using planar x-rays. As such, it has been predominately studied using clinical methods [61, 65, 67, 241]. Femoral neck length, measured from the medial head to the lateral cortex of the greater trochanter, shows the highest correlation with fracture load [41], however this correlation is much lower than that of BMD with fracture load. Clinical evaluations showed that femoral neck length, and hip axis length (which is defined as the distance from the medial border of the pelvis, adjacent to the acetabulum, to the lateral cortex of the greater trochanter along the femoral neck) both correlate with hip fracture risk, but are most powerful when used in conjunction with BMD [61, 67].

Failure location has been experimentally confirmed in only one study [57] (and an associated thesis [56]). Fractures were seen to typically originate in the superior or anterior femoral neck, regardless of fracture type (e.g., cervical or trochanteric). That said, trochanteric fractures appeared to originated in the lateral neck, whereas neck fractures originated in the sub-capital region.

Femoroplasty is a technique in which polymethylmethacrylate (PMMA) (a.k.a. bone cement) is injected into the proximal femur as a way to reinforce the senile trabecular structure. Studies on femoroplasty have shown that it significantly increases strength and energy to failure [56, 94, 221]. These benefits are currently outweighed by complications in dealing with

large quantities of PMMA, which include the highly exothermic reaction as it sets [56, 94], and difficulty with revision surgery due to the hardness of the material [16].

1.8.4 Limitations of laboratory models

The data gathered in laboratory experiments are of great importance to an understanding how the proximal femur behaves during a fall to the side. However, there are limitations to the current laboratory models which must be understood in order to apply the results. One of the principal limitations is the lack of a biofidelic, dynamic model of the fall. Previously published models attempt to apply loads representative of a fall by applying a constant-rate compression of 100 mm/s. This compression rate has never been validated as a reasonable value, and therefore must be taken with caution – especially given the influence it has on the behaviour of the bone.

The majority of the models use materials testing machines to apply compression to the lateral trochanter, and continue compression until fracture is attained. This method of creating fractures ignores the fact that the bone is not loaded in constant displacement rate. Rather, it is loaded by the inertia of a falling mass (i.e., the body) and mediated by deformable elements (e.g., pelvis, skin and fat). This means that the loading conditions (such as displacement rate) will be a function of the fall characteristics and the properties of the proximal femur itself. Additionally, the energy of the mass in the fall is limited and fracture is not guaranteed. In fact, fracture is seen in only ~5% of real-world falls in the elderly [151, 163].

The results of the laboratory models are also at odds with the epidemiology. Laboratory fracture models have failed to identify any significant predictor of fracture besides BMD. Studies of fractures in the general population show a considerable overlap of BMDs between fracture and non-fracture patients [67, 80]. As few as 22% of hip fractures patients have BMDs that would classify as osteoporotic, and 42% that would classify as osteopenic or osteoporotic [220]. This could be due to the use of ultimate load as the criteria for fracture resilience, and also may be influenced by the constant displacement rate, rather than inertial

impact, testing method.

1.8.5 Summary

Laboratory models of hip fracture have been used extensively to determine the influence of different aspects of the proximal femur on its strength. The primary focus of the modelling has been to understand the strength in a fall to the side, and in this respect researchers have developed methods that have shown the influence of many physical characteristics. Even with all of this effort and knowledge, there are still gaps between what is being modelled and the real world. A potential next step in laboratory testing is to refine the model of a fall to the side, increasing its biofidelity to allow researchers to understand the inputs and results of real world falls.

1.9 Computer models

Computer models are performed using the finite element (FE) and multibody dynamic (MBD) techniques. Most of the information about the mechanism of hip fracture gleaned from computer models has come from FE models because they are capable of providing stresses, strains and potential failure locations. This section focuses on the contributions and limitations of FE models. There have been few MBD models that have provided insight into hip fracture, focusing primarily on determining the fall conditions, which are discussed more thoroughly in Chapter 3.

1.9.1 Contributions of FE models

FE models are, in most cases, based on a laboratory model that can be used as a validation. Once an FE model is validated, it can be used to examine a number of different aspects of the failure event, including the role of soft tissue [146], fall modelling [147], non-invasive strength estimation for an arbitrary bone [110, 219, 240], and fracture risk estimation [180, 219, 224].

In addition to all of these uses, FE models have been used to understand the fracture itself. Multiple researchers have used FE models to estimate the location where fractures begin and

what conditions lead to fracture [33, 113–115, 139, 140, 142, 164, 181, 224, 235, 236]. Data from these models have increased the understanding of the conditions in the femur leading up to fracture, but there are significant limitations to the models that can affect interpretation of their results.

1.9.2 Limitations of FE models

A primary limitation of FE models relates to the experimental models used to determine their validity. As discussed in §1.8, the displacement rate used to load a proximal femur can influence its mechanical and failure behaviours. It is thought that these changes in organ level behaviour originate from changes at the bone material level (see §1.4 and §1.5). The experiment used to validate an FE model should replicate the boundary and loading conditions applied in the model, and also have the similar material properties as those applied in the model. Failing to meet these requirements can lead to validation uncertainty.

The material properties applied to a model are often determined from tissue level tests conducted at low strain rate, while the validation experiment is an organ level test conducted at high displacement rate (e.g., 100 mm/s). As a result, it is possible that significant differences exist in bone properties in the tissue and organ level tests. If the FE model agrees with the organ level test, it could be due to an error in the model (such as an inappropriate constraint), rather than accurate representation of the experiment.

It is possible that the displacement rate does not affect the desired outcome variable, making the selection of a validation experiment essentially arbitrary. However, if there is a dependence on experimental method, FE researchers must be sure to select material properties that are in agreement with their validation experiments. Currently, there is no standard for what experiment should be used to validate a given FE outcome variable and therefore special attention should be paid to possible interplay between materials definitions and model validation.

In addition to validation challenges, selecting appropriate material properties for the FE model can be complicated. As we have seen in previous sections, bone is a highly complex

material at all length scales, making simplifications for computational applications necessary. The majority of FE models use relatively simple material definitions that lack the anisotropy and viscoelasticity discussed in §1.4. This lack of fidelity in the material model may influence how load is transferred and how failure is determined given that ultimate strain is a function of strain rate. Additionally, they use a single phase, neglecting the flow of bone marrow, which has been shown to be important at high strain rates [37]. A final limitation of the material definitions in current FE models is the use of a highly simplified post yield behaviour model. After yield, bone behaviour is dependent on the bone type [91], loading direction [87], and loading rate [87, 121, 122], making computational modelling very difficult. There are efforts to address this final limitation using plastic material models for post yield behaviour [14, 59, 237]. However, those models are still simple, modelling both tension and compression in the same manner, and lacking the non-linearity of post-yield strain with strain rate seen in the bone core tests described in §1.5 [14, 59].

Another limitation of FE models of biological systems is that they are deterministic in nature, i.e., each model has only one solution, even though there are many probable outcomes for a given loading scenario in the physiological situation. This limitation has been addressed using Monte Carlo simulations for material and geometry, which provide a distribution of strains or stresses for a given element, in a given loading scenario [33, 126, 223]. These statistical simulations attempt to address the uncertainty regarding material properties and geometry, but do not address uncertainty in fluid flow, or changes in material properties due to deformation, strain rate, or loading direction.

One final but important note relating to the interpretation of the results of FE models is that many predict failure in the cancellous bone [33, 113–115, 139, 142, 164, 235, 236]. This result is intriguing, but one must remember computational models are only valid for output variables that have been validated. Since the deformation of the cancellous bone cannot be directly measured in experimental tests, it has never been validated, and this necessarily limits the scientific reliability of these data. To fully trust the results that are given for internal strains,

a method needs to be devised and implemented to experimentally determine the actual internal deformations for comparison and validation.

1.9.3 Applications of models to fracture prevention

Modelling results have been applied to the development of a number of different biomechanical technologies that are intended to prevent hip fracture. The most advanced in terms of development and assessment are hip protectors, which are soft or hard pads that are positioned over the greater trochanters using special undergarments. Hip protectors have been shown to decrease the impact force [43, 123, 234], but have not proven effective in the real-world [117, 187].

Another more recent development is compliant flooring, which modifies the landing surface in order to decrease the impact force [125, 133]. This technology is currently in the development and proving stage [44], but shows promise as a passive strategy to protect individuals against fracture in a way that is effective immediately and not compromised by lack of compliance – the two main problems with pharmacological remedies (which can take years to become effective) and hip protectors (which are often not worn at the time of a fracture).

Both of these technologies use data from modelling of hip fracture to determine the target maximum forces in their simulated falls. Some tests also use knowledge of where fractures are likely to occur to determine effectiveness for a particular injury [123].

Chapter 2

Objective and research questions

The objective of this thesis is to quantitatively compare the mechanical and failure behaviours of hip fracture in constant displacement rate loading with the behaviours in impact fall simulation. Specifically, this research will determine if experimental constant displacement rate models of hip fracture produce force-displacement, strain, failure and fracture results that are similar to those observed in impact fall simulation. Any differences observed in the behaviours of the bones between these two experimental models will inform a decision matrix. This matrix can be used as a guide by *ex vivo* and FE modellers to select the most appropriate experiment for their outcome variables or validation needs.

2.1 Research questions

In pursuit of these objectives, I have identified three research questions:

1. Does the force-displacement mechanical response of a proximal femur differ when it is loaded using constant displacement rates vs. impact fall simulation?
2. Does the surface strain field of the proximal femur differ when it is loaded using constant displacement rates vs. impact fall simulation?
3. Do the locations of initial failure and final fracture change when the proximal femur is loaded using constant displacement rates vs. impact fall simulation?

2.2 Method of investigation

To answer the research questions, an inertially driven fall simulator with elements modelling the body, pelvis, and soft tissues over the greater trochanter has been built. Its design and development are discussed in Chapter 3. This fall simulator was used in conjunction with a materials testing machine to determine if the mechanical, strain and failure behaviours of the proximal femur are similar or divergent between the quasi-static and impact loading methods.

2.2.1 Does the force-displacement behaviour differ between constant displacement rate and impact fall simulation testing?

This question was addressed by comparing the sub-failure, force-displacement response of proximal femurs loaded using quasi-static, constant displacement rates in a materials testing machine and the response of the same femurs loaded to failure in the impact fall simulator. Single specimens were loaded to 50% of their aBMD predicted failure load in a materials testing machine, and the behavioural metrics of force, displacement, stiffness, and energy to maximum applied load were measured. The same bones were then transferred to the impact fall simulator and loaded to failure. The same metrics measured in the quasi-static tests were measured in the fall simulation tests and compared to the quasi-static results at the the maximum load applied in the materials testing machine. This research is presented in Chapter 4.

2.2.2 Does the surface strain differ between constant displacement rate and impact fall simulation testing?

This question was addressed using stereo digital image correlation (DIC) of the anterior-superior surface of the femoral neck. Stereo, three-dimensional (3D) DIC data were collected on the anterior-superior femoral neck of single specimens loaded to 50% of their aBMD predicted failure loads in a materials testing machine, followed by loading to failure in the impact fall simulator. The 3D surface profiles measured by the DIC software were aligned, and the minimum principal strains on the surfaces were compared to determine if there was a difference in strain magnitude or distribution between the two loading protocols. This work is presented

in Chapter 5.

2.2.3 Do the failure locations and final fracture types differ between constant displacement rate and impact fall simulation testing?

This question was addressed using two groups of specimens, one group loaded to failure in a materials testing machine under quasi-static, constant displacement rates and another group loaded to failure in the impact fall simulator. Fracture locations were determined from high-speed videos of the experiments and final fracture types were classified by an orthopaedic surgeon and compared between the two groups. This research is presented in Chapter 5.

Chapter 3

Design of a fall simulator

Before the behaviour of the proximal in a simulated fall can be compared to the behaviour in quasi-static loading, a validated impact fall simulator must be developed. Impact fall simulation has been used by the hip protector design community for many years, however, the particular requirements of testing human material makes adaptation of a pre-existing fall simulator impossible. In this chapter, the rational, design and justification for a drop tower apparatus that simulates a human fall to the side from standing, is presented.

This chapter was published as a peer reviewed paper in ASME Journal of Biomechanical Engineering [76] and is reprinted here with permission.

3.1 Introduction

Hip fracture is a devastating injury associated with high morbidity and mortality in the elderly as well as high treatment cost [30, 72, 244]. In order to prevent or improve treatment of these injuries, researchers have conducted extensively laboratory studies, with previous *ex vivo* models of hip fracture establishing the role of posture [71, 189], compression rate [46, 242], and bone density [137, 138, 149] in the strength of the femur. Bone density has been shown to correlate to the strength of a proximal femur loaded in a fall posture, and constant compression rates ranging from 0.7 mm/s to 100 mm/s [46, 242] have been used to prove that the proximal

femur's mechanical properties depend on compression rate magnitude.

In all of the previous studies, tests were conducted using materials testing machines to apply loads to either the head or lateral trochanter of the femur [23, 46, 57, 112, 137, 138, 149, 197, 242]. While compression rate varied from experiment to experiment, in each case it was fixed and constant throughout testing, and the final result was fracture of the bone. Only two researchers have conducted tests at high compression rates [7, 242], potentially representative of rates that would be experienced in a fall, but they tested the femur in isolation, neglecting the influence of surrounding tissues and structures.

Falls are by definition dynamic, inertia-driven events in which compression rates are not constant, and in some portions of the event may be as high as 3000 mm/s [68], although in individuals that are able to react quickly, fall speeds are often slowed due to an extended hand or other protective action [68, 232]. Additionally, fracture is not prescribed in a fall, with only $\sim 5\%$ of falls resulting in fracture [163]. In a fall to the side, the compression rate of the proximal femur is dependent on the contact velocity with the ground, properties of the overlying soft tissue, stiffness and mass of the pelvis, mass of the body, and the stiffness of the proximal femur itself. The stiffness of the proximal femur is of primary importance to its resulting behaviour and is itself a function of compression (strain) rate [36, 46, 52, 55, 87, 134, 157, 190, 198, 205, 242, 251]. The recursive nature of this process means that the compression rate of the femur will not be constant during the impact event and in only a subset of bones will result in fracture. Modelling the compression rate of the lateral femur as a constant, and deterministically imposing fracture, is a significant limitation of the previous experiments, and may limit their ability to inform researchers about the mechanics of the fracture and how to identify the specific bones that are susceptible to fracture.

In addition to these biomechanical observations, epidemiological research has shown that a discrepancy exists between what is expected from lab results and what is seen in the general population. Lab and epidemiology results [25, 39] have shown that bone density is an important factor that is correlated to fracture load, but we also see that more than two-thirds of fractures

occur outside the low bone density population [80, 212, 220] This indicates that other factors (e.g., structural) may be involved [27, 165, 211].

No researchers have explicitly modelled the fall and observed the resulting bone behaviour, which would depend recursively on both fall mechanics and the bone's response, as described above. Understanding these behaviours may illuminate potential screening targets, helping inform and improve biomechanical (e.g., hip protectors) and clinical (e.g., pharmaceutical) hip fracture prevention. Thus, in an effort to bridge the gap between current *ex vivo* hip fracture models and *in vivo* fractures, this research comprises two related aims: a) develop an impact-based testing apparatus to model a physiological fall to the side, including representations of the surrounding anatomic structures; b) validate quantitative methods for analysing loading mechanics using high speed imaging and surface strain analysis.

The mechanics of hip fracture are influenced by the structures and tissues (referred to as elements) surrounding the femur. These elements are, on the medial side, the pelvis and body, and on the lateral side, the soft tissue over the trochanter. Previous efforts have been made to understand the behaviour of these elements in isolation, and in some cases researchers have incorporated them into tests to evaluate the effectiveness of biomechanical devices such as hip protectors or compliant flooring [123, 125, 202]. In the following sections, research surrounding each element will be considered and representative properties will be chosen for inclusion in our laboratory model. Finally, the behaviour of the model during a simulated fall will be compared to the behaviour of published human volunteer tests to evaluate the biofidelity of the apparatus.

In addition to the above device development, a quantitative method of evaluating the strain on the bone surface will be validated. Researchers have traditionally used strain gauges to evaluate the strain state of the proximal femur during loading to fracture [51]. It is known from finite element analyses [235] and from research on the experimental error of strain gauges [50] that strain gradients on bones can be high. This has two primary consequences for their use for strain measurement: 1) it is difficult to compare two models of the same event (e.g., com-

paring a fracture experiment to a subject-specific finite element model), and 2) placement of gauges in locations where meaningful strain is "guaranteed" to occur is impossible. In an experiment complementary to the development of the fall simulator, we will directly compare strain measurements from digital image correlation (DIC), which is a method for full field strain measurement, to the gold standard strain rosette. Our goal is to validate the use of DIC for measurement of strain on the surface of a proximal femur so that future experiments can use it in lieu of strain rosettes.

3.2 Materials and methods

A drop tower-based fall simulation apparatus was developed and its loading mechanics were compared to the response of volunteer, as well as human cadaver hip and pelvis tests. In addition to the apparatus development, low compression rate, sub-failure tests were conducted on human bone specimens to validate the use of digital image correlation for surface strain measurement on the femoral neck. These techniques will be used in future experiments to determine the loading and fracture mechanics of human proximal femora loaded in a fall configuration.

3.2.1 Fall simulator apparatus

Due to the dynamic nature of a fall to the side, the tissues and structures surrounding the proximal femur must be included for accurate biofidelic modelling. These structures are, on the lateral side of the femur, skin and soft tissue over the trochanter, and on the medial side, the pelvis and body. The biological scenario was simplified to a lumped parameter model (Figure 3.1) which was characterized using a state-of-the-art plastic bone surrogate for consistency (3rd Gen. Large Femur, SawbonesTM, Vashon, WA). The following sections contain a literature review and justification for the properties selected for each element.

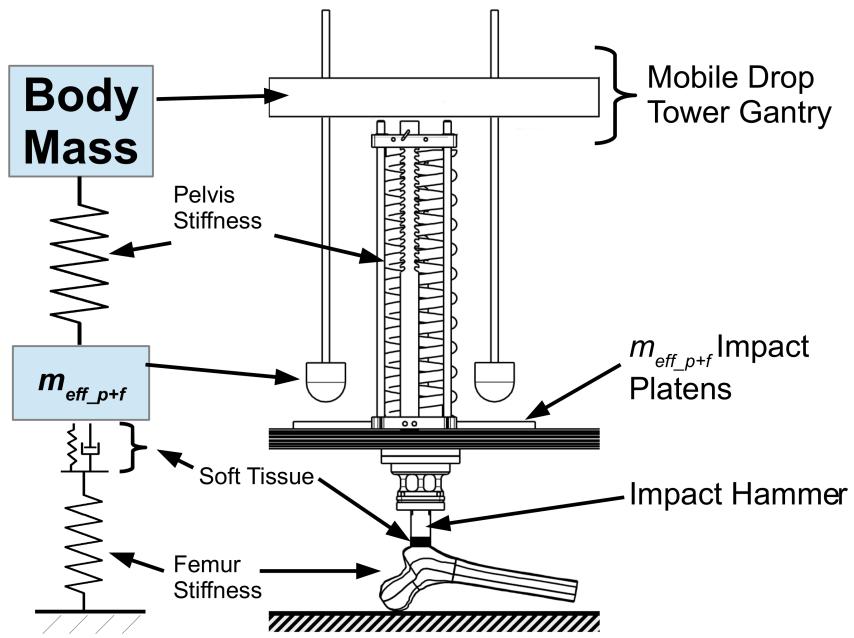


Figure 3.1: Schematic of the fall simulator showing the mass and spring of structures that influence loading a fall to the side. m_{eff_p+f} is the effective mass of the lateral pelvis and femur.

Effective body mass

The selection of the body mass determines (along with the pelvis and soft tissue stiffnesses) the peak force and (along with the impact velocity) the energy delivered during the impact. Body habitus, measured using BMI, has been shown to correlate negatively with hip fracture risk [160, 168, 169, 180, 186], indicating that increased body mass can be protective. There is, however, also some evidence to the contrary, especially that very high body mass can increase fracture risk [171]. To quantify how body mass influences the loading parameters during a fall, researchers have mathematically modelled falls using link assemblies [232], and performed pelvis drop experiments on young, university-student volunteers [124, 199].

In a fall to the side the effective body mass impacting the trochanter is less than the total body mass of the person falling. Conceptually, this is because some of the body mass would be supported by the feet and lower extremities. Experimentalists and computational modellers have put this so-called effective body mass at approximately half body weight [199, 232]. Additionally, direct measurement of the effective mass in young volunteers has provided a

range of 24.1–50 kg [123, 199, 201, 232] with a cross-study average for human tests subjects of 31.9 kg. The cross-study average for the computational models was higher, and more variable depending on the assumptions made in the model. The cross-study average of human subjects did not show the same variability due to model design, or dependency on assumptions, and was taken as the value of the effective body mass. A value of 32 kg was selected for the fall simulator, which is within one standard deviation of 50% body weight of hip fracture cases [3, 29, 170, 180].

Effective mass of the femur and lateral pelvis

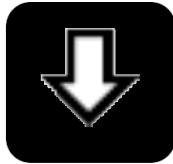
The two main mass components that govern the loading of the proximal femur in a fall are: a) the body mass, whose impact is attenuated by the compliance of the pelvis, and the selection of which was discussed above; and b) the mass of the lateral pelvis and proximal femur, whose impact is attenuated only by the compliance of the femur and overlying soft tissue (Figure 3.1). Previous work on pelvis fracture in automotive, side-impact accidents performed impacts of osteoligamentous pelvises at ~ 4.5 m/s [15]. In these experiments, an impact hammer was instrumented with a load cell and dropped on to the greater trochanter of the femur. In the resulting force-time traces a force spike was seen early in the loading history, before deformation of the pelvis started (Figure 3.2). This force spike was created by the inertia of the mass close to the impact surface.

To better understand the origin of this pulse, consider that in an impact, the velocity of the impact hammer and the contact point on the object being impacted must become equal, creating a force due to the inertia of the mass near the contact point. In the case of a fall to the side, before the pelvis can compress, the proximal femur and lateral pelvis must go from moving 3 m/s, to being stationary against the ground. This happens very quickly, requiring a high acceleration, and creating a force pulse at the interface between the ground and the trochanter. This behaviour has been documented in head-first impact studies, where a force pulse is generated as the head is stopped before compression of the neck begins [172, 204].

In the fall to the side, as one moves medially from the contact point, the pulse is attenuated due to the compliance of the tissues between the ground and the location under consideration. Because they are relatively stiff and close to the impact surface, the lateral trochanter and femoral neck will see a considerable portion of this force pulse. Since the magnitude can be quite high, as seen in Figure 3.2, it must be accounted for in the test apparatus. Once the velocity of femur and lateral pelvis have gone to zero, force in the proximal femur is generated by the body compressing the pelvis.

The effective combined mass of the lateral pelvis and proximal femur ($m_{(eff_p+f)}$) during an impact was determined using a momentum analysis of the osteoligamentous pelvis impacts [15]. The momentum equation is $\int F dt = m_{(eff_p+f)} \cdot \Delta V$ and the desired quantity is the effective mass of the lateral pelvis and femur. The force time traces were numerically integrated to determine $\int F dt$, and optical tracking data from markers attached to the greater trochanter were used to determine ΔV . The quotient of $\frac{\int F dt}{\Delta V}$ gave the effective mass of the lateral pelvis and proximal femur in these tests. This analysis yielded an effective mass (average \pm SD) of 0.97 ± 0.52 kg.

In the current apparatus, $m_{(eff_p+f)}$ impacts a platen that is rigidly attached to a component used to hold and stabilize other parts of the apparatus (Figure 3.1). The mass of this stabilizing component influences the impulse delivered to the bone, and as such, the final value of $m_{(eff_p+f)}$ needed to be tailored to the apparatus. The *ex-vivo* value determined from the momentum analysis described above was used as a starting point for this process, and the mass was increased incrementally until the force delivered by the impulse matched a published, impact-velocity scaled, human volunteer test [124]. The impact-velocity scaling was done by multiplying the human test by the ratio of (current experiment impact velocity)/(volunteer experiment impact velocity) which can be shown to be effective using impulse and vibration analysis. The final value for the pelvis mass was 1.98 kg.



Falling Mass

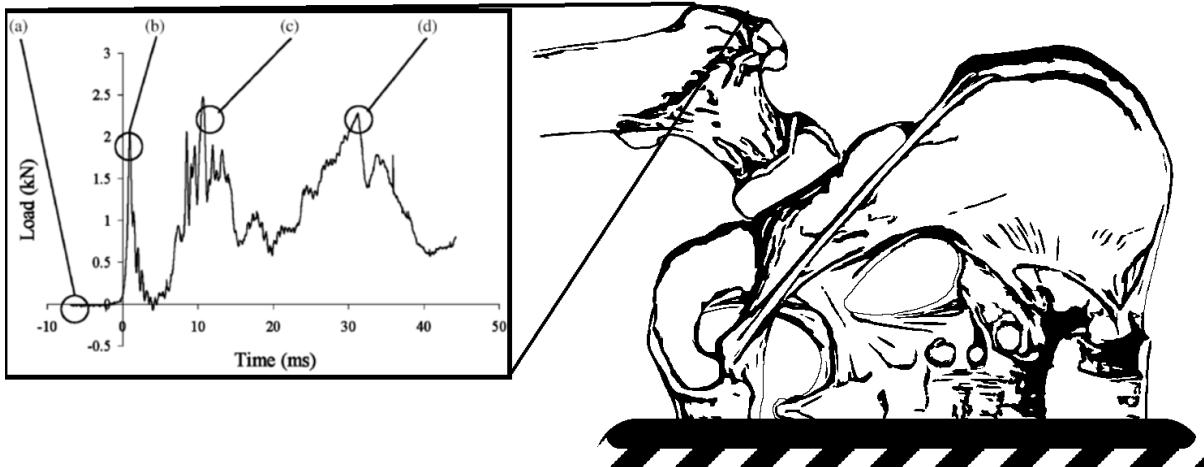


Figure 3.2: In previous tests on osteoligamentous pelvises an instrumented impactor was dropped on the greater trochanter at 4.5 m/s. When the impactor came into contact with the trochanter a force spike was seen before deformation of the pelvis had begun, as indicated by peak (b) in the inset graph (adapted from Beason et al. [15], with permission). This spike was created by the acceleration of the mass of the lateral pelvis and femur. Illustrations adapted from Gray and Warren Harmon Lewis [79], copyright expired.

Pelvis stiffness

The pelvis has a finite stiffness, the value of which determines the duration of the impulse delivered by the effective body mass. This stiffness is not only a function of bone structure, but also body position at impact [201, 233], which must be taken into account when selecting the stiffness. Human volunteer experiments to characterize the stiffness of the pelvis [124, 199, 201] have measured values ranging from 20.9 to 90 kN/m, depending on experimental protocol, and have shown its value to be constant above a compressive force of 300 N [124, 199]. Researchers used vibrational mechanics with various boundary conditions to determine the effective body mass and lateral stiffness of the pelvis [124, 201] of human volunteers. The value of the effective body mass derived from one experimental condition [201] was 32.8 kg, which is similar to the effective body mass selected for the current experiment, and the corresponding pelvic

stiffness was 35.4 ± 14.7 kN/m. This was selected as a base value for the current experiment, and was then modified to account for body position during the impact.

Videos of volunteers falling have shown that people flex their trunks away from the ground at impact [68, 233], which increases the stiffness of the pelvis [201]. To account for this, a spring with a rate of 50 kN/m was selected, corresponding to one standard deviation above the average stiffness for our base condition. The value of 50 kN/m also agreed with the recommendations of the international consensus statement for hip protector testing [202].

With the afore mentioned effective body mass and pelvis stiffness, the maximum anticipated deflection of the pelvis spring was 8 cm. A steel spring with an outside diameter of 10 cm, a free length of 35 cm, and a maximum linear deflection of 18.6 cm was used. A ratchet system with a tooth spacing of 9 mm was incorporated to prevent rebound, as return of the stored energy post test was undesirable.

Soft tissue model

Soft tissue over the greater trochanter is thought to protect the hip and prevent fractures in a sideways fall [147, 199, 200]. Researchers have shown that increasing the thickness of the soft tissue increases its ability to absorb and dissipate energy, thereby decreasing the potential for hip fracture. Computational work [147] showed a power-law relationship of $\text{Impact force} \propto \text{Thickness}^{-0.2011}$. Experimental results from volunteer pelvis drop tests also showed a power-law trend of decreasing peak force with increased soft-tissue thickness, with a Spearman's $\rho = -0.71$ [199], and results from impact tests of cadaveric soft tissue [200] showed that a 1 mm increase in soft tissue thickness related linearly to a decrease in peak force of 70 N. In the latter experiments, peak forces exceeded typical fracture forces for the proximal femur even for their highest tissue thickness of 45 mm.

Despite this understanding of how soft tissue attenuates impact force, epidemiological evidence of the protective nature of soft tissue is limited. Studies that have investigated the thickness of the soft tissue over the greater trochanter in fracture cases [29, 160, 170] have provided

a pooled average and standard deviation of 29.4 ± 15.6 mm. Only a single study investigated the mechanical properties of the trochanteric soft tissue [123], and showed that, while the stiffness of the tissue in the hip region of elderly individuals varied greatly with measurement location, they were typically around 20–35 kN/m over the greater trochanter when measured at a compressive stress of 100–140 kPa.

For use in our fall simulator, a thickness one standard deviation below the average of fracture cases was selected in order to model an at-risk person. Due to material thickness availability, a final thickness of 19 mm (0.75 inch) was chosen. This value also falls within the range recommended in the standardized hip protector testing protocol [202]. The material chosen had a stiffness of 26.5 kN/m at 95 kPa compressive stress (Evazote, Zotefoams Plc, Croydon, England), which was in the range of stiffnesses measured experimentally in the trochanteric region [123].

Impact velocity

Impact velocity during a fall from standing is governed by two things. Firstly, the height of the person falling [232], and secondly any reactions, such as extending an arm to break the fall [68]. Computational models [232] of falls from standing have put the impact velocity at between 3.35 and 4.34 m/s. These values are higher than those observed in experimental tests where volunteers impacted their hips at a mean \pm SD of around 3 ± 0.6 m/s [68, 233].

An impact velocity of 3.0 m/s was selected based on the average of the volunteer tests [68]. The specimen was placed stationary in the fall simulator (fall posture described below) and the effective body and pelvis+femur mass was dropped onto it. The effective body mass impacted the top of the pelvis spring, while $m_{(eff-p+f)}$ bypassed the spring and contacted loading platens connected to the specimen through the soft-tissue model. To time these events so that they occurred coincidentally, $m_{(eff-p+f)}$ was separated into two equal parts and suspended from the effective body mass by linear guide rods of length equal to the free length of the pelvis spring (Figures 3.1 and 3.3). This arrangement ensured that $m_{(eff-p+f)}$ delivered a shock load

at impact, and the effective body mass delivered a load mediated by the pelvis spring.

Specimen placement

The surrogate bone was placed in the literature standard orientation for a fall to the side from standing. This orientation was, 10° adduction of the shaft and 15° internal rotation of the neck [46, 57, 137, 149]. Polymethylmethacrylate (PMMA, Bosworth Co, Skokie, IL) was used to make caps for the femoral head and lateral greater trochanter to prevent local crushing and also to provide even mounting surfaces. Each cap consisted of 20 gram (g) of PMMA, formed by hand into a disk approximately 3.5 cm in diameter, and moulded so that the head and trochanter loading surfaces were parallel.

3.2.2 Data collection and processing

A six-axis load cell (Denton 4366J, Humanetics, Plymouth, MI), with an axial force capacity of ± 13.34 kN, and non-linearity of <130 N, recorded loads on the greater trochanter, between the pelvis spring and the specimen.

Displacements of the greater trochanter and impact hammer were measured using a high speed video camera (Phantom v9, Vision Research, Wayne, NJ) imaging at 9216 frames per second (fps) and a resolution of 576x288 pixel (px) (spatial resolution of 5 px/mm). Displacements were calculated from the video data using TEMA Automotive (v3.0, Image Systems, North Hollywood, CA). To determine the accuracy of this measurement, a validation study was carried out in which a bone surrogate (model 1130-21, Sawbones, Vashon, WA) was moved in a cyclic manner in the field of view of the displacement measurement camera. A dial gauge was recorded showing the displacement of the surrogate at any given time, and the results of the TEMA calculations were compared to the dial gauge at randomly selected times. The uncertainty of this measurement was determined to be ± 0.043 mm ($\pm SD$).

Forces were recorded using a PCI-6040E (National Instruments, Austin, TX), 12 bit data acquisition board, and custom LabView software at a sampling rate of 20 kilohertz (kHz), with hardware anti-aliasing filtering at 10 kHz. The load cell outputs were amplified such that

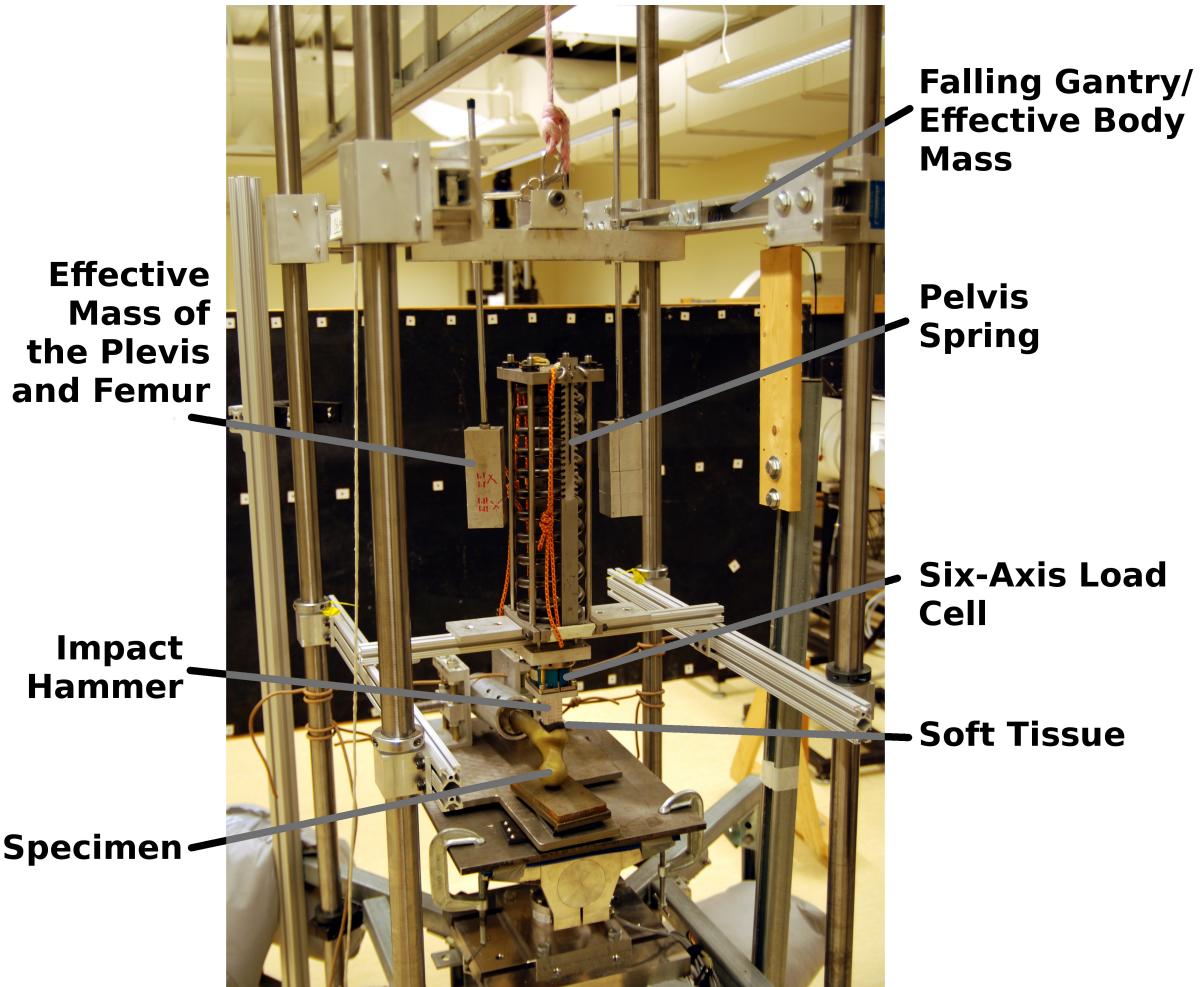


Figure 3.3: A photo of the fall simulator showing each element of the model

voltage saturation was attained at ± 6 kN, giving a force resolution of 3 N/bit. Imaging data was synchronized by recording a camera trigger signal generated by the drop tower gantry closing a switch as it fell.

All data were forward and reverse, low-pass filtered using MatLab (2010b, The Mathworks, Natick, MA) with a -3 dB cut off at 500 hertz (Hz). Force data were processed with a fourth-order Butterworth filter and displacement data were processed with a second-order Butterworth filter. The cut off frequency of 500 Hz was chosen to remove high frequency ringing of the drop tower while retaining important impact events. The final fall simulator arrangement is shown in Figure 3.3.

3.2.3 Digital image correlation verification

A separate experiment was carried out to validate the use of DIC on the bone of the proximal femur for surface strain measurement. Subsequent tests in the fall simulator will use DIC strain measurement exclusively.

Twenty human femoral specimens (3 normal, 11 osteopenic, 6 osteoporotic) were cleaned of soft tissue and periosteum, and a strain rosette (FRA-2-11-3LT, Tokyo Sokki Kenkyujo Co., Tokyo, Japan) was glued on the anterior-superior femoral neck using cyanoacrylate and a standard protocol [239, Chapter: Strain gauge analysis of hard tissues: factors influencing measurements]. The femoral neck, including the strain gauge, of each specimen was painted with a white-on-black speckle pattern using an airbrush (VL, Paasche, Chicago, IL). The specimens were placed in a materials testing machine (8874, Instron, Norwood, MA) in the fall configuration [57] discussed in §3.2.1 and loaded to 50% of their total aBMD predicted fracture load [23], at a constant compression rate of 0.5 mm/s.

Force was output by the materials testing machine using a voltage scaling to provide 4 V at the target maximum load. Displacement was output with a constant scaling of 0.35 mm/V. Three channels of strain data were collected at 20 kHz from the strain rosettes using an SCXI-1520 strain gauge input module (National Instruments, Austin, TX) connected to the PCI-6040E, with hardware anti-alias filtering at 10 Hz. Two high speed video cameras (Phantom v12.1), recording at 100 fps and a resolution of 1280x800 px (spatial resolution of approximately 17 px/mm), were used to observe the anterior-superior femoral neck including the strain gauge. The video and voltage data were synchronized by recording the camera trigger signal which was emitted by the materials testing machine. The video images were imported into commercially available DIC software (StrainMaster, LaVison, Göttingen Germany) for analysis. The DIC data were analysed using an iterative approach in which erroneous displacement vectors were identified as those that were >1.5 SD from their immediate neighbours. These vectors were replaced by an average of their neighbours and reanalysed. After this second round of analysis, the strain field was filtered using a 3x3 median filter. The strain gauge

data were taken as a gold standard measurement and the DIC data were compared to them in order to determine the DIC's accuracy and precision. Comparisons were made using minimum principal strain since it is known to correlate to compressive failure [14].

3.3 Results

The force-time profile of the fall simulator was compared to the same velocity-scaled human volunteer data discussed in §3.2.1. The fall simulator and velocity-scaled human volunteer profiles were seen to be similar in timing and magnitude (Figure 3.4). The initial force peak in the fall simulator was 40% higher than in the volunteer data and delayed by 5.2 ms. The average loading rate up to the initial peak was the same in both data sets, with a value of 122.4 kN/s in the fall simulator and 121.7 kN/s in the scaled volunteer data. The force of the initial peak was observed to be increased by internal vibration of the pelvis spring, which can be seen as a sinusoidal variation throughout loading. A Fourier analysis of the impact event showed that the amplitude of spring component was approximately 13% of the dominant frequency component, however, its timing created a discrepancy in force equal to 80% of the volunteer data at 16 ms post impact. The global peak force in the fall simulator was within 5 ms of the volunteer data. The rapid decrease in the fall simulator force after 50 ms was due to the engagement of the ratchet, preventing rebound of the spring.

In the DIC verification experiment, minimum principal strains from the DIC and the strain rosettes were well correlated (Figure 3.5). The DIC result contained image-to-image noise (Figure 3.7), the amplitude of this noise was found to be normally distributed and the frequency spectrum had no peaks in the range of 0-50 Hz (the Nyquist frequency), indicating that it was likely random in nature. Three data sets (5, 8 and 10 in Figure 3.5) were subjected to camera vibration. It is thought that one of the tripod legs was contacting the table supporting the materials testing machine, transferring vibrations associated with the hydraulic pump to the camera. Identification of the faulty data sets was trivial, as strain would change by more than 1000 microstrain ($\mu\epsilon$) from frame to frame. Excluding the datasets thus affected, the root mean

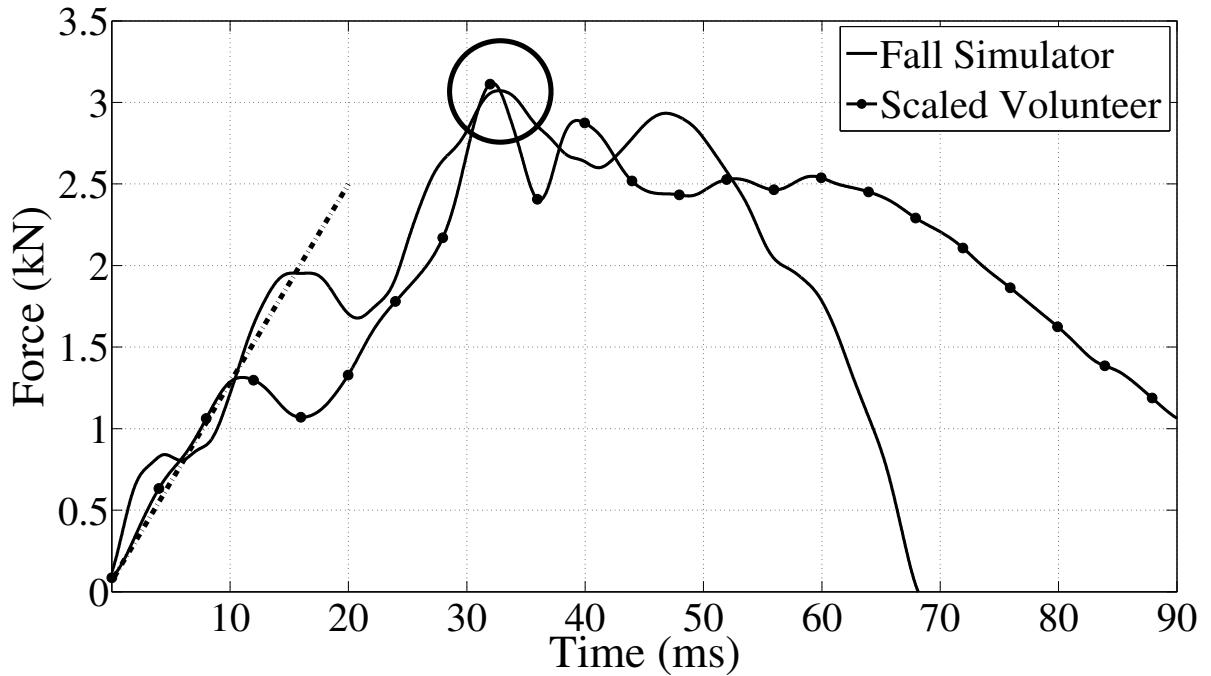


Figure 3.4: An example response of the fall simulator plotted with human pelvis drop data [124]. The dashed line indicates the initial loading slope of the scaled volunteer data and the circle indicates the location of the peak forces. The human data was scaled by the ratio of the impact velocities.

squared (RMS) average difference was $127 \mu\epsilon$ (range: [-375, 336]), and the standard deviation was $239 \mu\epsilon$.

3.4 Discussion

A method to simulate a fall to the side, impacting the greater trochanter of the femur, was developed. A lumped parameter model of the human pelvis and hip joint, with the properties of each structure represented by literature values, was used. Additionally, a method for full field strain analysis was validated on the neck of the proximal femur.

Previous researchers have developed models of the pelvis and body for use in hip protector testing [123, 199, 202], but a model appropriate for testing of cadaveric tissue has not been implemented. The selected body mass and pelvis stiffness were similar to those selected by the previous researchers, which was reflected in the time to peak force being similar between our apparatus and theirs. Robinovitch et al. [200, 202] reported a time to peak force for their

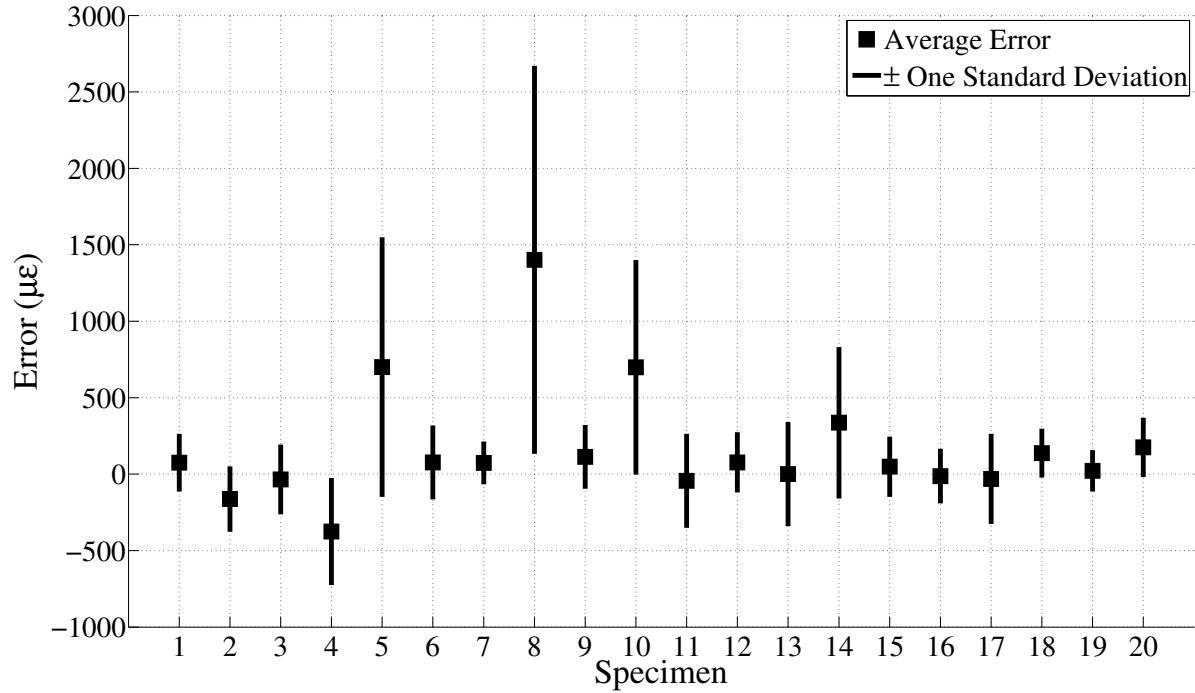


Figure 3.5: The average and standard deviation of the strain errors measured for each specimen. Three specimens, 5, 8, and 10, were subjected to camera vibration, leading to incorrect DIC strain readings.

apparatus between 30 and 45 ms post contact; our apparatus had a peak force at 32 ms, with a plateau until \sim 50 ms due to oscillations of the pelvis spring. The use of the ratchet to prevent rebound does not have an analogue in the previous literature because research on hip protectors used metal or plastic femur surrogates, which were not at risk of fracture during the test. This change represents an adaptation required for testing of human material to obtain physiological fractures.

Another adaptation for physiological testing was changing the pelvis spring from mobile to stationary. Robinovitch et al. [202] proposed two methods for fall simulation to test hip protectors, one of which used a moving linear spring, dropped in a drop tower similar to the current apparatus. Our experiments using such an arrangement, in which the pelvis spring is dropped on the specimen, during the development of the current fall simulator, showed that the inertia of the spring's free-end was large enough to fracture a proximal femur before compression of the spring even began. While the impulse due to the spring's mass may not be

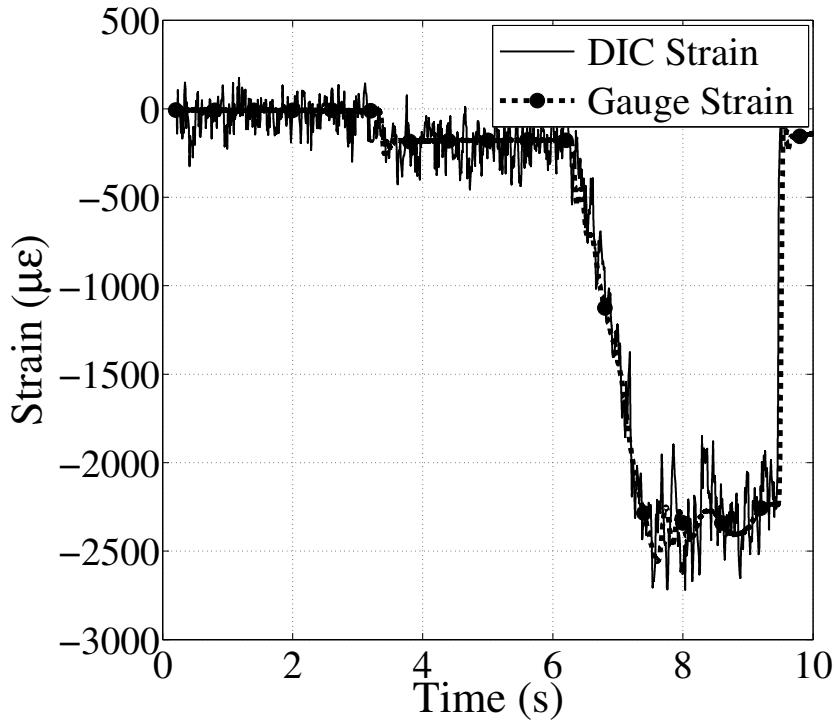


Figure 3.6: Example data from the DIC analysis plotted with the strain gauge data.
Time vs. strain plot for specimen 16 shows the character and magnitude of the random noise.

important for hip protector testing, which aims to attenuate the global force peak, it is important when testing cadaveric specimens.

One principal difference between the current fall simulator and the previous models, was the inclusion of the effective mass of the lateral pelvis and proximal femur. Previous models neglected this mass and used signal filtering to remove artefacts created by shock loading and oscillations of the pelvis spring [200, 202]. As discussed in §3.2.1, the impulse delivered by the effective mass of the lateral pelvis and proximal femur is physiologic and should be modelled. This impulse creates a situation in the initial milliseconds of the impact of a high loading rate, which may change the behaviour of a human bone specimen in ways that are difficult to predict [46, 52, 87, 157, 190, 198, 251]. The model showed that the magnitude of this impulse is determined more by the soft tissue mechanics than the stiffness of the bone. A linear mathematical model of the current apparatus showed that the sensitivity of the impulse to the selected mass was low, with a 25% increase in the mass changing the impulse magnitude by 6%. We

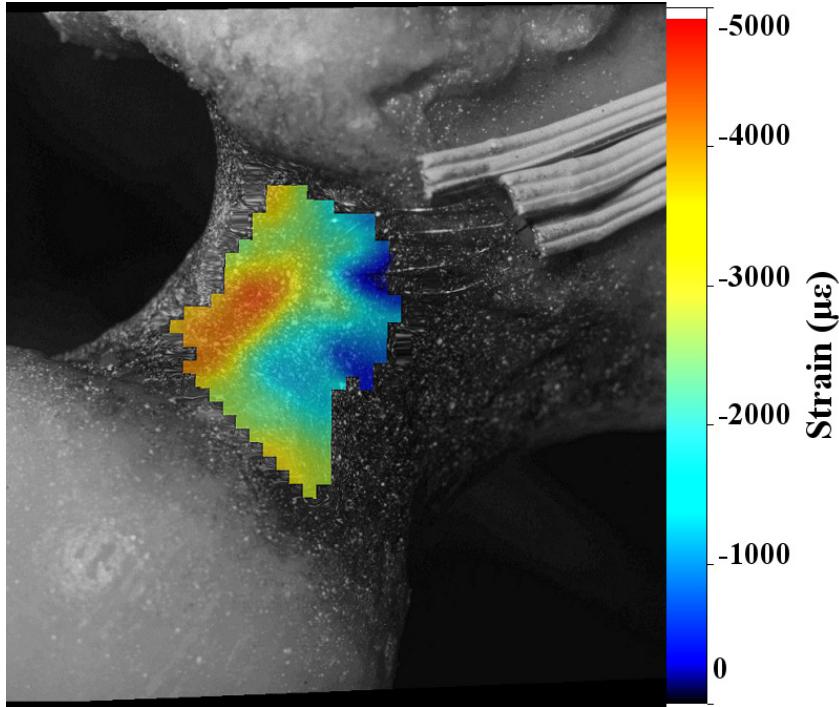


Figure 3.7: An example DIC strain contour map (b) shows how the strain varied over the surface of the bone at the maximum applied load. The bone is oriented such that superior is to the left and lateral to the top. The head of the femur is in the lower left and the trochanter occupies the upper portion of the image with the strain gauge wires visible on the right of the image.

believe that inclusion of this high loading rate portion of the curve may be important to gain an understanding of how the failure of the proximal femur is initiated and how it progresses.

The soft-tissue model in the current apparatus is minimal compared to those used in hip protector testing, in which researchers typically model the entire lateral trunk [123]. Modelling of the trunk is a requirement for accurate fitting of the hip protectors but removes the possibility of observation of the femoral deformation. The current apparatus required access to the femoral neck and trochanter for video observation. A fall from standing might allow some of the energy to be dissipated to the surrounding tissues, however, an impact on the greater trochanter in a low BMI individual would likely have little energy dissipation into the surrounding tissues due to the prominence of the bony landmark. Our fall simulator is modelling, in effect, the scenario that hip protectors are meant to address.

The impact velocity selected for the fall simulator was the average velocity from previously

publish fall volunteer experiments [68]. This value is lower than the value of 3.4 m/s recommended by the international consensus statement on hip protector testing [202]. The principal goal of the tests preformed for hip protector testing is to model a fall that would almost certainly create a fracture and see if it could be prevented through use of a biomechanical device. The goal of the current research was not to model a severe fall, but to model a physiological fall that may, or may not result in fracture. As mentioned in the introduction, previous hip fracture research tested all bones to failure. The fall simulator described in this paper has the potential to identify bones that do not fracture in a fall to the side from standing, allowing further investigation of what is protective or detrimental to bone strength.

The orientation of the bone in the apparatus was selected based on preceding literature. This placement is set by shaft angle from the vertical (adduction angle) and rotation of the neck (internal neck rotation). Previous researchers selected the orientation by reviewing fracture studies and selecting adduction and neck rotation angles that created clinical fracture types [7, 138]. The adduction angle was later modified to a more physiologic angle after video of human fall volunteers became available [46]. The neck rotation angle has remained the same as it is difficult to evaluate its value in volunteers. Feldman and Robinovitch [68] measured the angle of the contact point with the ground, around the circumference of the hip, in human volunteers who fell from standing height (the hip proximity angle). They found this angle to be an average angle of 8° posterior from directly lateral. Combination of this value with the average femoral anteversion in the human population of 9.73° [227] gives an impact angle of 17.7° , which is close to the and current value of 15° .

Data processing for the current fall simulator differs from that proposed in previous work on hip protector testing. Hip protector testing has used low pass filtering to clean unwanted artefacts from the signal by cutting off the signal at either 100 Hz [200] or 50 Hz [202]. The dynamics of the impulse delivered to the lateral trochanter, which we propose are seen by the proximal femur, are filtered out using such low cut-off frequencies. We selected a value that retains the impulse at the trochanter and also retains the oscillations of the pelvis spring. The

filter cut-off frequency of 500 Hz provided us with a meaningful, but clean force-time trace, showing dynamical events that we believe are important to biofidelic modelling of physiologic falls to the side from standing.

Strain fields were obtained using DIC and verified at one location, using a triaxial strain rosette. DIC has been used to measure the surface stains on porcine mandibles [249], mouse tibiae [222] and on the surface of a plastic surrogate femur model [60]. The magnitudes of strains measured on the mouse tibiae were compared with strain gauge measurements in similar locations on different specimens, but were not directly compared. Due to the inhomogeneity of strain on the surface of the mouse tibiae, this comparison method has limited utility as a validation. In the case of the plastic femur surrogate, DIC calculated strains were compared to a finite element model and found to agree in shape and magnitude. However, the strain field on the plastic model was more uniform and did not display the high strain gradients seen in both our tests and those conducted on the mouse tibiae. Our validation expanded upon both of these techniques by using a strain gauge, at the same time and location as DIC, to evaluate accuracy and precision of the DIC measurement on a human bone specimen. The noise level of $230 \mu\epsilon$ is only 2.0% of low rate yield strain, and 5.9% of high rate yield strain of cortical bone in compression [87]. Strain fields calculated by DIC in our experiments exhibited steep strain gradients, potentially due to the inhomogeneity of the bone in the human specimens. This is similar to the strain fields seen in previous research [222], and justifies the utility of DIC over point strain measurements.

We have developed a method to simulate a fall to the side from standing that addressed several limitations of previous fall models. Our goal was to increase the biofidelity of physiologic fall modelling and we have included many aspects of the anatomy and impact that were previously omitted. In light of these advances, there are notable limitations to discuss. Firstly, the pelvis stiffness model included a spring which had a large mass and exhibited its own vibrations, influencing the overall loading pattern. A number of different avenues for elimination of this artefact were investigated, including rubber dampening, as well as composite and elas-

tomer springs. Each of these came with its own limitations of either increasing the mass of the spring further, being difficult to characterize or being highly non-linear. The limitation of a non-ideal spring (i.e., a spring with mass) is nearly impossible to eliminate, however, it was mitigated through the use of the ratchet system and by starting the test with the spring stationary. We believe that the advantages of the drop tower method, namely free response of the specimen and biofidelic loading, outweigh the limitation imposed by the spring. Secondly, the pelvis model lacks a damping element, which in life is the consequence of the soft tissues surrounding the pelvis. Previous literature examined the potential range of damping of the pelvis and found a average value of 520 ± 340 Ns/m [201]. Using fundamental vibrational mechanics, and our chosen mass, spring and initial velocity, inclusion of this damping would change the peak force by between 0.3% and 6.3%, with an average change of 2.2%. We believe this variation is acceptable in the context of human variability. Third, the soft tissue model used in the current fall simulator is limited by the need to visualize the bone throughout the impact, and does not allow for transfer of force and energy to tissues that, in life, would surround the proximal femur. In effect, our apparatus simulates individuals that are slender and have prominent trochanters, which are a target population for hip fracture prevention [169]. We feel the limitation of only modelling a subset of individuals based on soft tissue thickness is outweighed by the benefit of fracture visualization. Fourthly, the impact model developed includes only the effective mass of the body that is active in the lateral direction. In a fall to the side there are other important masses and inertias at play, namely the superior-inferior (S-I) constraint on the femoral head by the mass of the body, and the rotational, mass, and friction constraints on the distal leg. The S-I constraint on femoral head will likely affect the final fracture lines produced more than the mechanical and failure behaviours of the bone. Relaxed constraint of the lower leg would allow the orientation of the bone to change throughout the impact and could lead to different mechanical behaviours. The added biofidelity of the latter change could lead to highly complex dynamics that would be difficult to analyse and reproduce, decreasing the consistency of the experimental boundary conditions. Finally, fixed parameters for each of the elements in

the apparatus reduces the ability to model a subject-specific fall for each femoral specimen. A subject-specific design, allowing variation of the body mass, soft tissue thickness, and possibly, pelvis stiffness and $m_{(eff_p+f)}$ would increase the biofidelity of the model. However, the current level of understanding of how the parameters would vary based on body morphology is limited, and making appropriate changes would be difficult. Additionally, changing all the experimental parameters in this manner would reduce the statistical power of a given set of experiments and greatly increase the required number of specimens. The current arrangement of a set apparatus, with the only variable between experiments being the specimen, provides a way to identify specimens of interest in a manner that is efficient and statistically powerful.

Our method, which is driven by the energy of a free falling mass representing a person falling from standing, removes artificial imposition of trochanteric displacement rate and allows each specimen to deform based on local response and failure, permitting visualization of a specimen's physiologic failure progression. In future experiments, we plan to test a multitude of specimens using this apparatus, characterize the fracture progression and compare the free response behaviour to the behaviour seen in previous fixed displacement rate experiments. We believe that understanding how the mechanics change under dynamic loading will allow us to better understand hip fracture, eventually leading to improved prevention, detection and treatment of this debilitating condition.

Chapter 4

Impact and constant displacement rate loading change the mechanical response of the femur in sideways fall simulation

To address the research questions outlined in §2.2.1 and §2.2.2, specimens were loaded to sub-failure forces using quasi-static, constant displacement rate methods, and then loaded to failure in the fall simulator. The behaviour in the sub-failure and failure loading were compared at the same force in terms of stiffness, energy, and point strain. The specimens were then compared to two previously fractured specimen groups that were tested at 2 mm/s [173] and 100 mm/s [57] in terms of stiffness, energy to failure and maximum force.

The research presented in this chapter has been submitted for publication in the Journal of Biomechanics [77].

4.1 Introduction

Hip fracture is a devastating injury associated with high mortality, morbidity, and high economic costs, on both personal and social levels [30, 244]. One of the critical avenues for reducing the burden of hip fracture has been early identification of potential hip fracture pa-

tients [104, 107, 231]. Clinically, early identification is done using areal bone mineral density (aBMD) as measured by dual-energy x-ray absorptiometry (DXA), but this technique has been shown to capture fewer than 30% of those who suffer a hip fracture [220]. One factor that may contributes to this discrepancy is a lack of understanding of the mechanism of hip fracture. Increased understanding of the fracture mechanics of the proximal femur could aid the development of specific and sensitive tools for identifying people at risk of hip fracture.

Biomechanical researchers have utilized mechanical testing and computational modelling to study how the proximal femur fails in a fall to the side [113, 114, 119, 219, 235, 236]. These researchers have identified the roles of bone density [23, 47, 57, 63, 129, 137, 138, 149, 192], posture [189], displacement rate [46, 242], geometry [41], and loading configuration [112, 114]. While these data have contributed greatly to the current understanding of hip fracture, they were conducted using constant displacement rate protocols. There may be differences in the mechanics of the proximal femur in an impact loading situation that would be important for understanding and interpreting the previous results.

In a fall to the side, the displacement rate (mm/s) of the proximal femur is not prescribed or constant. Instead, it is dictated by the dynamics of the fall and the response of the body under impact. The femur acts as a compliant member in a spring-mass system which includes the body, the pelvis, and soft tissues surrounding the femur. During the fall, the instantaneous value of a femur's compliance influences its displacement and loading rates (N/s), which in turn alter the bone's mechanical behaviour in a rapid and complex way [37, 134]. We believe that the recursive nature of this relationship makes it essential to model the fall in order to measure, and visualize, the actual progression of failure of the proximal femur. Because the femur has not been studied under these conditions previously, it is not known if the behaviour or mechanics of the femur are different between the historical laboratory simulations and real-world falls to the side.

The objectives of this research were to determine if: i) femoral deformation in constant displacement rate and impact testing are different; ii) femoral failure characteristics are dif-

ferent between the two loading methods; and iii) to characterize the free-response force and displacement curves in a biofidelic impact. We hypothesized that the proximal femur's sub-failure force-displacement and strain behaviours would be different between low displacement rate loading, and inertia-driven fall simulation. Further, we hypothesized that the proximal femurs' failure mechanics would be different between fixed displacement rate and fall simulation failure tests. We also characterized the loading and displacement rates in the fall simulation tests, and examined their relationships with specimen stiffness.

4.2 Materials and methods

Three groups of specimens were used in this analysis. These were the specimen groups used by de Bakker et al. [57] and Nishiyama et al. [173], tested in our lab under constant displacement rates, and an additional 21 specimens that were tested in the combined high sub-failure + inertial fall simulation protocol (Table 4.3). The specimens will be referenced by how displacement was applied to each; those from Nishiyama et al. [173] will be referred to as the *slow* group; those from de Bakker et al. [57] will be referred to as the *fast* group; and the specimens in the current test will be referred to as the *fall* group because they were failed using the fall simulator. Each specimen in the fall group was subjected to two test conditions, a quasi-static, constant displacement rate, sub-failure test (fall:QS) and an inertially-driven, fall simulation failure test (fall:FS).

All specimens considered in this study were fresh frozen human proximal femora. The slow and fast groups' specimen preparations were discussed in their respective publications. The specimens in the fall group were cleaned of soft tissue and periosteum, and a white-speckle on black-background pattern was painted onto the anterior-superior neck using an airbrush (VL, Paasche, Chicago, IL) to facilitate DIC measurement of surface strains [76]. The specimens were imaged using dual-energy x-ray absorptiometry (DXA) (QRD 4500W, Hologic, Bedford, MA) with 4 kg rice to simulate soft tissue [218], and high-resolution, planar X-ray (FCR Capsula X, Fujifilm, Tokyo, Japan).

Table 4.1: Specimen groups showing identifying characteristics. The slow [173] and fast [57] groups are discussed in more detail in previous publications. The specimens in Fall:QS and Fall:FS are from the same specimen group, but include different specimens based on data availability. Comparisons between these two groups were restricted to specimens tested using both methods. Numbers are shown as mean (standard deviation).

Group	Age (Years)	Number and Gender	aBMD (g/cm ²)	Displacement Rate (mm/s)	Loading Rate (kN/s)
Slow	77 (13)	4M, 14F	0.613 (0.164)	2	0.5 (0.16)*
Fast	84 (7)	6M, 5F	0.763 (0.150)	100	33 (8)*
Fall:QS	77 (10)	1M, 19F	0.707 (0.108)	0.5	0.22 (0.16)*
Fall:FS	77 (11)	2M, 15F	0.694 (0.113)	114 (53)*	150 (35)*

* These results are presented here to illustrate the differences between groups

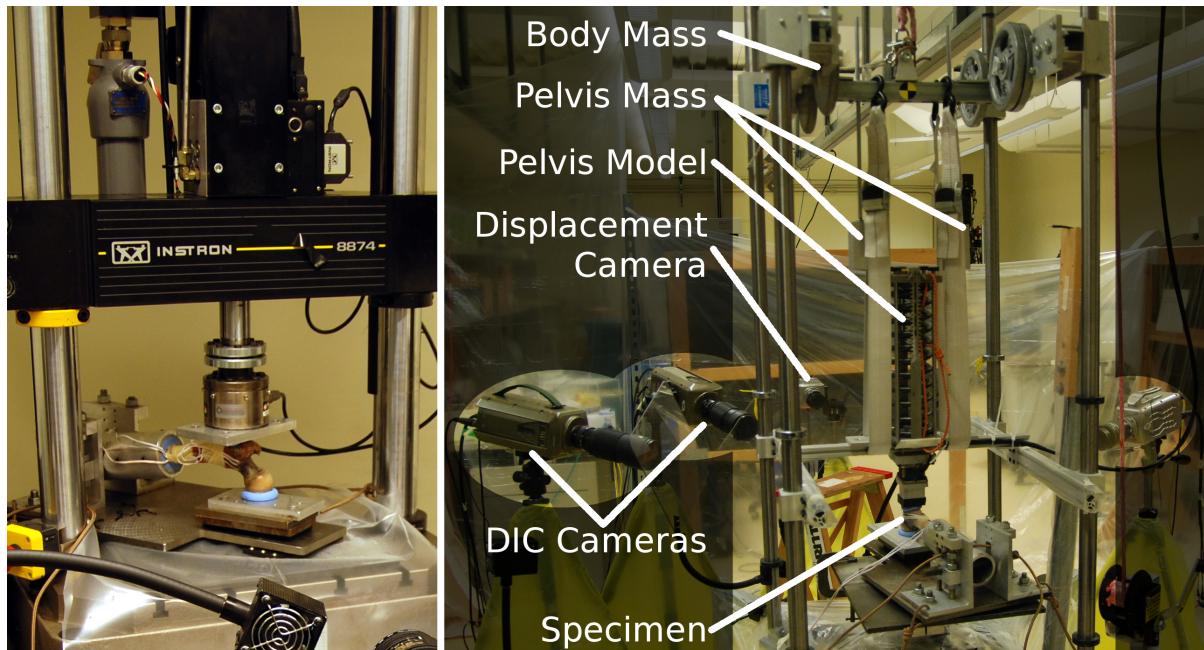


Figure 4.1: Left, the materials testing machine used for the fast, slow and fall:QS loading experiments. Right, the fall simulator used to test the fall:FS group showing the components and locations high-speed cameras used for displacement and DIC measurements. Image ©Seth Gilchrist, 2013.

The fall:QS tests were sub-failure, quasi-static, constant displacement rate tests performed to characterize the mechanical behaviours of the femurs in a scenario similar to the tests performed in the previously published literature (Figure 4.1, left). The tests were carried out in the standard fall configuration of 10° adduction, and 15° internal rotation of the femoral neck [46, 57, 149], with the distance from the pivot to the farthest point on the specimen in the range of 290–305 mm. Two polymethylmethacrylate (PMMA, Bosworth Co, Skokie, IL) caps were made, one for the lateral trochanter, and one for the medial femoral head, formed to have parallel surfaces. A materials testing machine (8874 Instron, Norwood, MA) was used to apply a 100 N preload over 5 s, which was held for 0.5 s. Then the specimens were loaded to 50% of their total-aBMD predicted failure loads [23] at 0.5 mm/s, held for 2 s, then unloaded. After testing, the bones were imaged with the planar X-ray. Force and displacement data were collected at 20 kHz (PCI-6040E, National Instruments, Austin, TX). Force was scaled to give 4 volts (V) at the target maximum load, with a constant displacement scaling of 0.35 mm/V. Two high-speed video cameras (Phantom V12.1, Vision Research, Wayne, NJ) viewed the anterior-superior femoral neck for stereo DIC. Video data were collected at 100 frames per second (fps), with an in-plane resolution of 1280x800 px (17 px/mm). The cameras were synchronized to the force and displacement data using the camera trigger signal.

The fall:FS tests were conducted to failure in the same orientation, mounting apparatus, and potting as the fall:QS tests, in a fall simulator (Figure 4.1, right) which we have previously described in detail [76]. Briefly, the simulator consisted of a drop tower with elements to replicate the mechanical effects of various anatomical elements surrounding the proximal femur including the body mass (32 kg), mass of the lateral pelvis and proximal femur (1.98 kg), pelvis stiffness (50 N/mm), and trochanteric soft tissue (19 mm foam). Force data were collected at 20 kHz from a \pm 13.34 kN, six axis load cell (Denton 4366J, Humanetics, Plymouth, MI), with a full range non-linearity of <130 N. The force was amplified such that voltage saturation would be obtained at 6 kN, giving a measurement precision of 3 N/bit. Data were collected at 20 kHz using the PCI-6040E. Displacements were obtained using a high speed camera

(Phantom V9, displacement camera in Figure 4.1) recording at 9216 fps, and an in plane resolution of 576x288 px (5 px/mm) that imaged both the impact hammer, and the potting over the greater trochanter. Two high speed video cameras (Phantom v12.1) recorded the anterior superior neck, similarly to the quasi-static tests, with a resolution of 1024x800 px (15 px/mm) and a frame rate of 10,000 fps for DIC. Finally, a single high speed camera (Phantom V9) observed the posterior aspect of the femur for qualitative fracture observation at 6006 fps, and a resolution of 480x480 px.

Specimen displacement was corrected for machine compliance. The compliances of the materials testing machine and the fall simulator were measured directly by applying known loads to the frames and measuring the resulting displacements. For slow, fast and fall:QS groups, specimen displacement was calculated by subtracting machine displacement from the loading platen displacement. For the fall:FS group, a first order, single degree of freedom, dynamic model of the drop tower was used to estimate machine displacement, which was subtracted from the (image analysis based) trochanter displacement. The dynamic model was used because static calculation overestimated the displacement in the initial milliseconds of the impact test. To compare strains between the fall:FS and fall:QS tests, minimum principal strain was measured in a location on the lateral, anterior-superior neck visible in both tests. Minimum principal strain was selected as a metric since it is known to correlate with compressive failure [14]. Strain measurement was done with DIC, using commercially available software (StrainMaster, LaVison, Göttingen Germany), and a previously validated technique [76].

In the slow, fast and fall:QS groups, stiffness was calculated as the slope of the force vs. displacement curve between 25% and 75% of the yield load. In the fall:FS group, stiffness was calculated between 25% and 90% of the yield force, where yield was defined as the point at which the force-displacement curve first deviated from linear. These ranges were chosen because they reliably captured the linear region of the force-displacement curves for all specimens. In the fall:QS group energy was calculated as the area under the force-displacement curve up to maximum applied force. In the slow, fast and fall:FS groups energy was calculated

as the area under the force-displacement curve up to the yield force. For the fall:FS group, energy was also calculated as the area under the force-displacement curve up to the maximum force applied in the fall:QS test. Loading and displacement rates were calculated as the average slopes of the force-time and displacement-time curves from the start of the impact to yield (Figure 4.5).

The energies, strains and stiffnesses of each specimen in the fall:QS and fall:FS tests were compared at the same force using matched pairs statistics. Specimens that had data for only one condition (i.e., only Fall:FS or only Fall:QS) were not analysed. Additionally, each specimen was classified by its relative behaviours in the two tests (e.g., fall:QS stiffer vs. fall:FS stiffer) and tested to determine if total aBMD was different between the classifications.

Yield force, energy to yield, and stiffness in the fall:FS group were compared to the slow and fast groups. The fall:QS group was not compared to the fast or slow groups, because the measurements for the fall:QS group were all sub-failure. The sub-failure force at which fall:FS and fall:QS were compared was selected in a specimen specific manner, and selection of a force for comparison to the fast and slow groups would not have been possible. Linear regressions of fracture force with total aBMD were examined by comparing the slope of the regression lines using Student's t-tests.

For all tests, if the data were normally distributed when examined on a Q-Q plot, Students t-tests, with $\alpha = 0.05$, were used, and are denoted by p_t . Otherwise, non-parametric, Wilcoxon tests, with $\alpha = 0.05$, were used, and denoted by p_w . All p-values were corrected for multiple comparisons within groups using Holm's method.

4.3 Results

Data for some of the specimens were unavailable (Table 4.3). There were 17 tests for which fall simulator data were available, 20 for which quasi-static data were available and 16 for which both were available. Examination of the X-rays by an orthopaedic surgeon (author PG) taken after the sub-failure tests showed no indication of fracture, and the force-displacement curves

showed low hysteresis, further indicating no damage.

Stiffness and maximum force were correlated with aBMD in the fall:FS group ($R^2=0.23$, $p_t=0.051$ and $R^2=0.39$, $p_t<0.01$, respectively), as well as the fall:QS group ($R^2=0.19$, $p_t=0.058$ and $R^2=0.94$, $p_t<0.001$, respectively).

Aggregate comparison of the fall:FS and fall:QS data showed no significant differences in energy ($p_t=1$), stiffness ($p_t=1$), or strain ($p_t=0.672$). Separating fall group specimens by relative behaviour in the FS and QS conditions showed that those that were stiffer in the FS condition had a higher total aBMD, with a difference in the medians of 0.20 g/cm^2 (Figure 4.2, $p_w=0.048$). No differences were seen in the fall group aBMDs when specimens were separated based on relative energy and strain ($p_w=1$, for both groupings).

Examining the load-displacement curves when specimens were grouped by their relative stiffness in the FS and QS conditions showed that the specimens that were stiffer in the QS condition had lower yield forces (which agrees with their lower aBMD), and displayed extended post yield, plastic deformation regions (Figure 4.3). Examination of the failures using the high speed videos showed that the specimens stiffer in the QS condition tend to fail in a progressive manner, starting with crushing of the greater trochanter, and progressing medially to the neck. Those that were stiffer in the FS condition deformed elastically before failing catastrophically in the neck or trochanteric regions.

Yield force, stiffness and energy to fracture in the fall:FS group were all significantly different from fast group (Table 4.2). The fall:FS group was 24% weaker, 160% stiffer, and absorbed only 27% of the energy before yielding.

Energy in the fall:FS group was significantly different from the slow group, absorbing only 58% of the slow group's average before yield, but no differences were observed in yield force ($p_w=0.85$) or stiffness ($p_t=0.82$).

The fall:FS group displayed a variety of loading and displacement rates, both of which correlated with specimen stiffness (Table 4.3 and Figs. 4.4, 4.6 and 4.5). As stiffness increased, loading rate increased ($p_t<0.01$) and displacement rate decreased ($p_t<0.01$). Additionally,

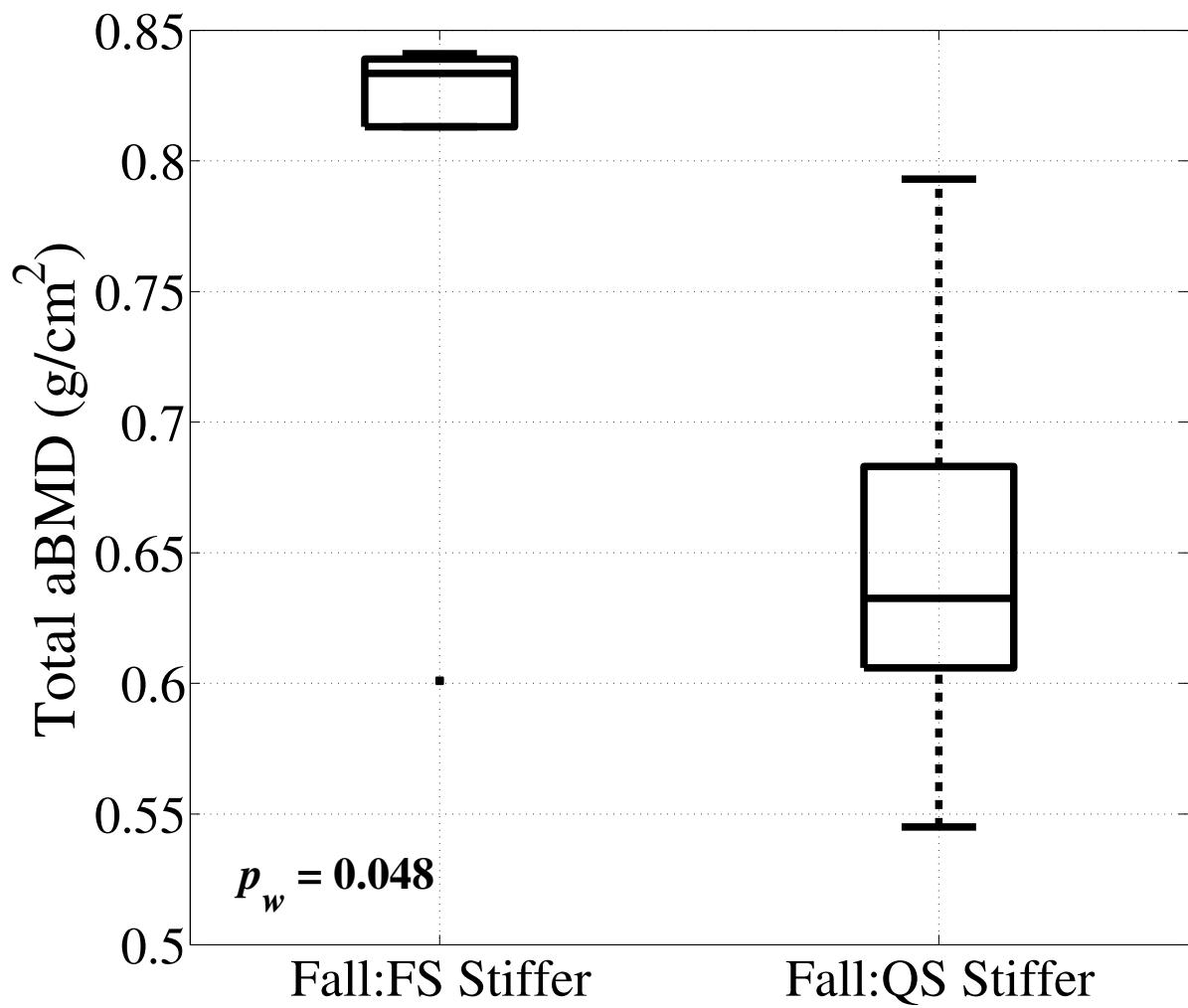


Figure 4.2: Fall group specimens that were stiffer in the fall:FS condition than the fall:QS condition had significantly higher aBMD. The cause of this phenomenon is thought to be increased hydrodynamic stiffening [37] due to decreased marrow space in bones with more trabeculi. Graphic ©Seth Gilchrist, 2013.

Table 4.2: Data from the fall:FS, fast and slow failure groups given as median (minimum, maximum) for non-normal data, and mean (standard deviation) for normally distributed data.

Measure	Fall:FS	Fast	Slow
Yield Force (N)	2.39 (1.41, 3.72)	3.15 (2.17, 5.30)*	2.42 (1.46, 4.44)
Stiffness (kN/mm)	1.92 (1.20)	0.74 (0.25)*	1.66 (0.52)
Energy (J)	2.55 (0.545, 5.39)	9.33 (5.46, 15.4)*	4.38 (1.14, 12.2)*

* $p \leq 0.01$ compared to fall:FS

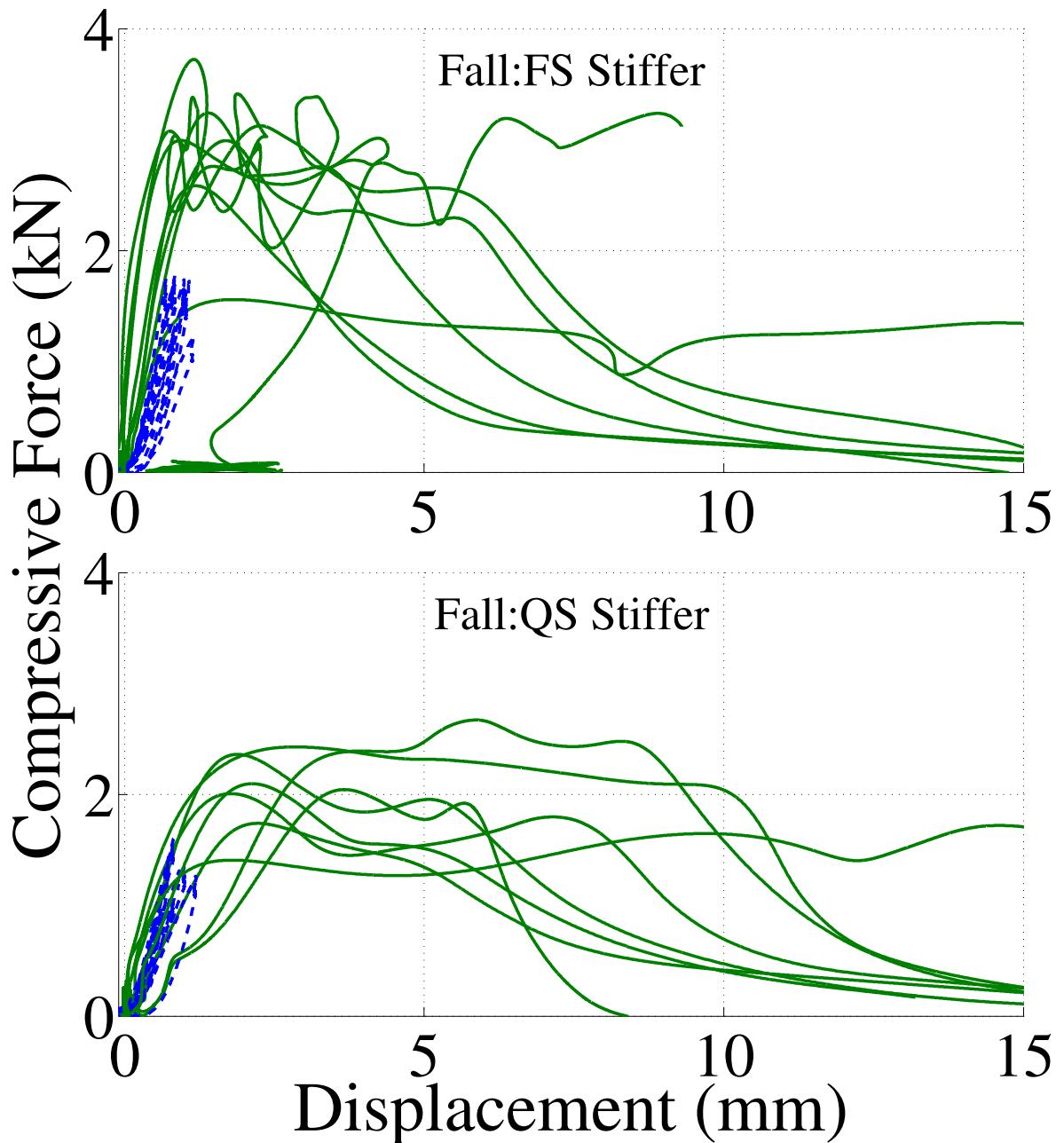


Figure 4.3: Force-displacement curves for specimens, grouped by relative behaviour in the FS and QS conditions. Those that were stiffer in the QS condition failed at a lower force, but displayed extended post yield behaviours. Dashed blue lines show the QS loading response, and solid green lines are the FS response. Graphic ©Seth Gilchrist, 2013.

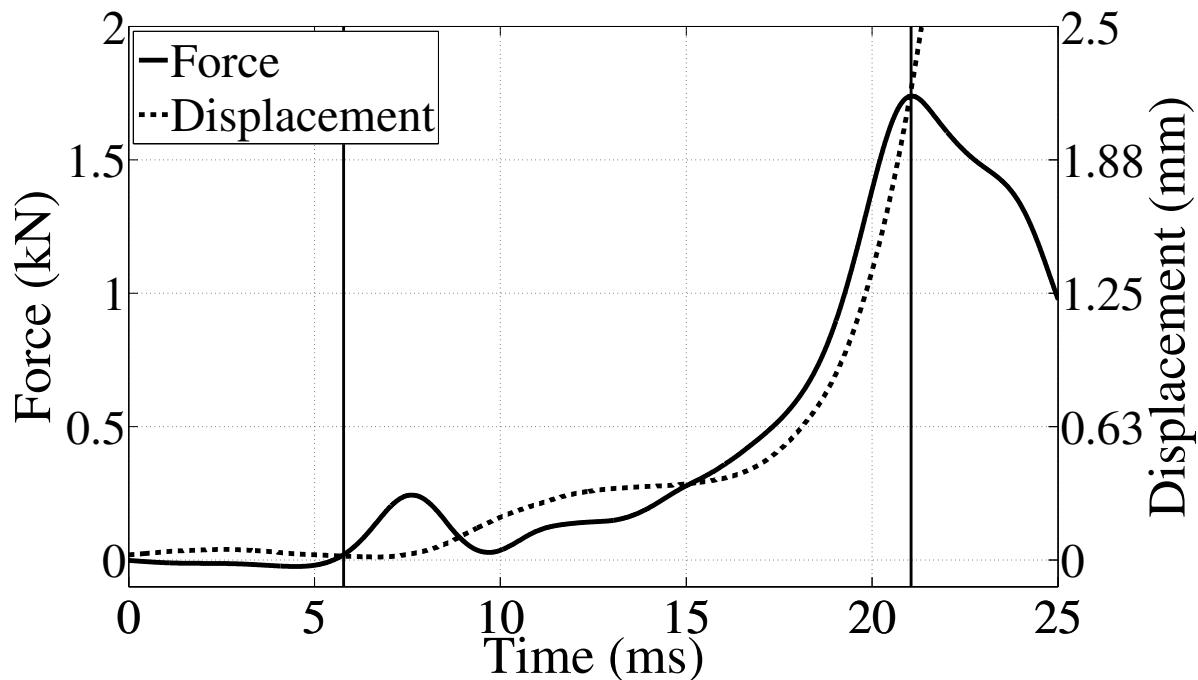


Figure 4.4: An example force and displacement vs. time plot. The vertical lines indicate the beginning of the impact (force > 20 N) and yield. Loading rate was defined as the average slope of the force-time curve from when the force exceeded 20 N to yield. Likewise, displacement rate was the average slope of the displacement-time curve from when the force exceeded 20 N to yield. These data are for specimen 7 and display the non-constant loading and displacement rates. Graphic ©Seth Gilchrist, 2013.

Table 4.3: Summary of specimen data and results. Osteoporosis status was based on WHO guidelines and calculated from aBMD.

Specimen	Side	Age	Gender	Total aBMD (g/cm ²)	OP Status	F _{qs} (N)	F _{sim,Fx} (N)	ε _{qs} (με)	ε _{sim} (με)	E _{qs} (J)	E _{sim} (J)	E _{sim,Fx} (J)	k _{qs} (kN/mm)	k _{sim} (kN/mm)	̄s _{sim} (mm/s)	̄F _{sim} (kN/s)
1	L	50	F	0.841	Normal	1793	3076	-2029	-	0.659	.166	1.96	2.42	3.86	38.8	161
2	R	73	M	0.813	Osteopenia	1693	3724	-2035	-4500	0.584	0.256	3.18	2.45	3.17	73.7	219
3	R	75	F	0.777	Osteopenia	1516	-	-2287	-	0.256	-	-	7.33	-	-	-
4	R	91	M	0.526	Osteoporosis	-	1767	-	-	-	-	0.546	-	3.31	43.5	127
5	L	86	F	0.666	Osteopenia	1362	2006	-3834	-	0.681	0.514	2.49	1.49	1.37	120	134
6	R	71	F	0.644	Osteopenia	1227	-	-3691	-	1.04	-	-	1.07	-	-	-
7	R	71	F	0.638	Osteoporosis	1312	1739	-2932	-1777	0.508	0.755	2.30	1.76	0.97	143	113
8	R	73	F	0.839	Normal	1724	3241	-2048	-2393	0.770	0.362	2.77	1.92	2.95	88.6	206
9	R	84	F	0.698	Osteopenia	1344	-	-1431	-	0.652	-	-	1.87	-	-	-
10	R	70	F	0.606	Osteoporosis	1204	1558	-2975	-1909	0.625	0.256	2.10	1.29	1.41	114	100
11	L	69	F	0.601	Osteoporosis	1230	2584	-1759	-1229	0.467	0.150	2.09	1.54	2.72	76.2	165
12	R	83	F	0.750	Osteopenia	1458	-	-2473	-	0.580	-	-	2.71	-	-	-
13	R	79	F	0.793	Osteopenia	1635	2991	-3084	-1925	0.763	0.406	3.41	1.85	2.23	106	185
14	R	76	F	0.833	Normal	1738	2759	-2328	-1703	0.784	0.601	2.55	1.89	2.33	88.6	163
15	R	78	F	0.621	Osteoporosis	1305	2361	-2369	-2046	0.409	0.503	2.92	1.97	1.33	111	144
16	L	83	F	0.782	Osteopenia	1611	2390	-3435	-3134	0.549	1.467	5.18	2.38	0.86	226	136
17	L	79	F	0.683	Osteopenia	1433	2044	-2540	-2179	0.467	1.540	3.78	2.29	0.64	218	120
18	R	80	F	0.627	Osteoporosis	1261	1407	-1571	-1764	0.669	0.597	1.86	1.23	1.20	125	102
19	R	71	F	0.834	Normal	1756	2993	-1908	-456	0.481	0.193	2.17	3.24	3.86	58.7	185
20	R	92	F	0.545	Osteoporosis	991	2428	-2105	-1372	0.322	0.131	5.39	1.48	1.20	167	142
21	L	96	F	0.546	Osteoporosis	1021	2096	-1887	-1963	0.368	0.309	2.77	1.44	1.22	132	130

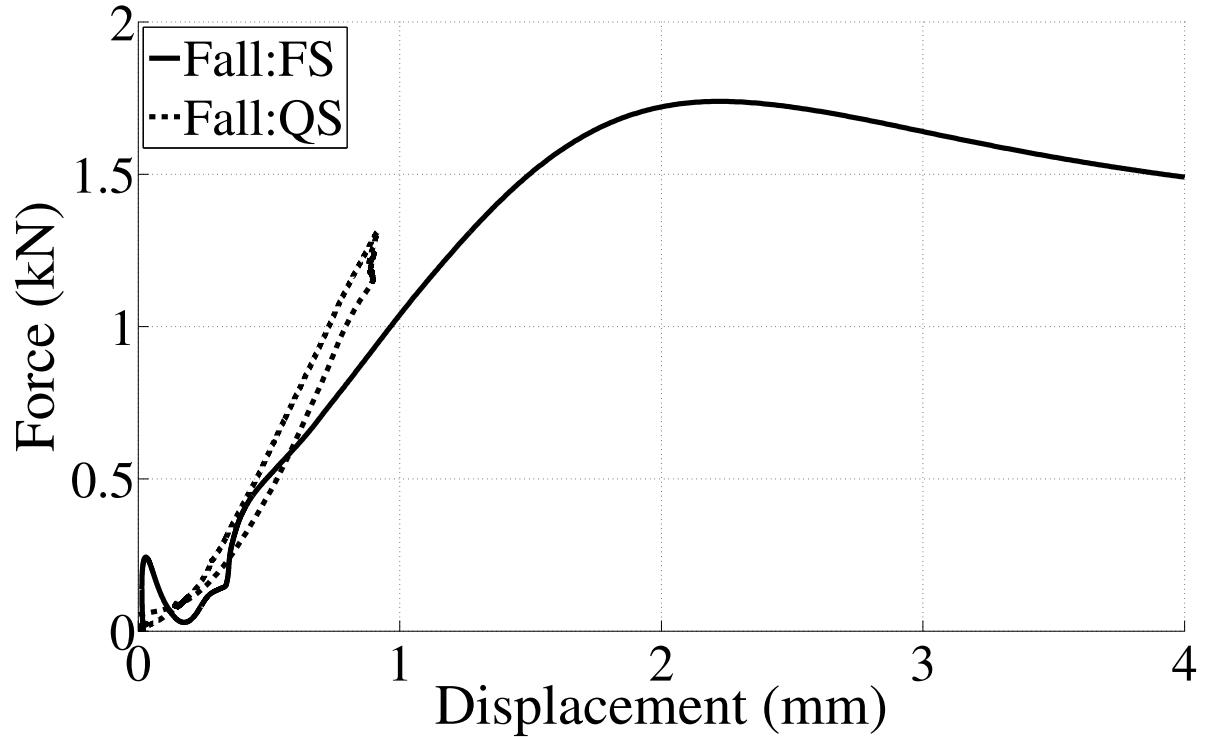


Figure 4.5: An example force vs. displacement plot. In this case the quasi-static test had a higher stiffness than the fall simulation. These data are for specimen 7.
Graphic ©Seth Gilchrist, 2013.

stiffness and yield force were proportionally related ($p_t < 0.01$, $R^2 = 0.391$). Finally, loading rate was correlated to maximum force ($R^2 = 0.92$, $p_t < 0.001$), but no correlation with displacement rate was observed ($p_t = 0.16$).

There were no differences in the slopes of the yield force vs. total aBMD regression lines between the fast, slow and fall:FS groups ($p_t = 0.88$ and $p_t = 0.46$ for de Bakker et al. [57] and Nishiyama et al. [173], respectively).

4.4 Discussion

The goals of our experiments were three fold: i) to determine if there were differences in behaviour between quasi-static, constant displacement rate and fall simulation loading conditions; ii) to determine if there were differences in failure characteristics between constant displacement rate loading and fall simulation; and iii) to characterize the variation in load-

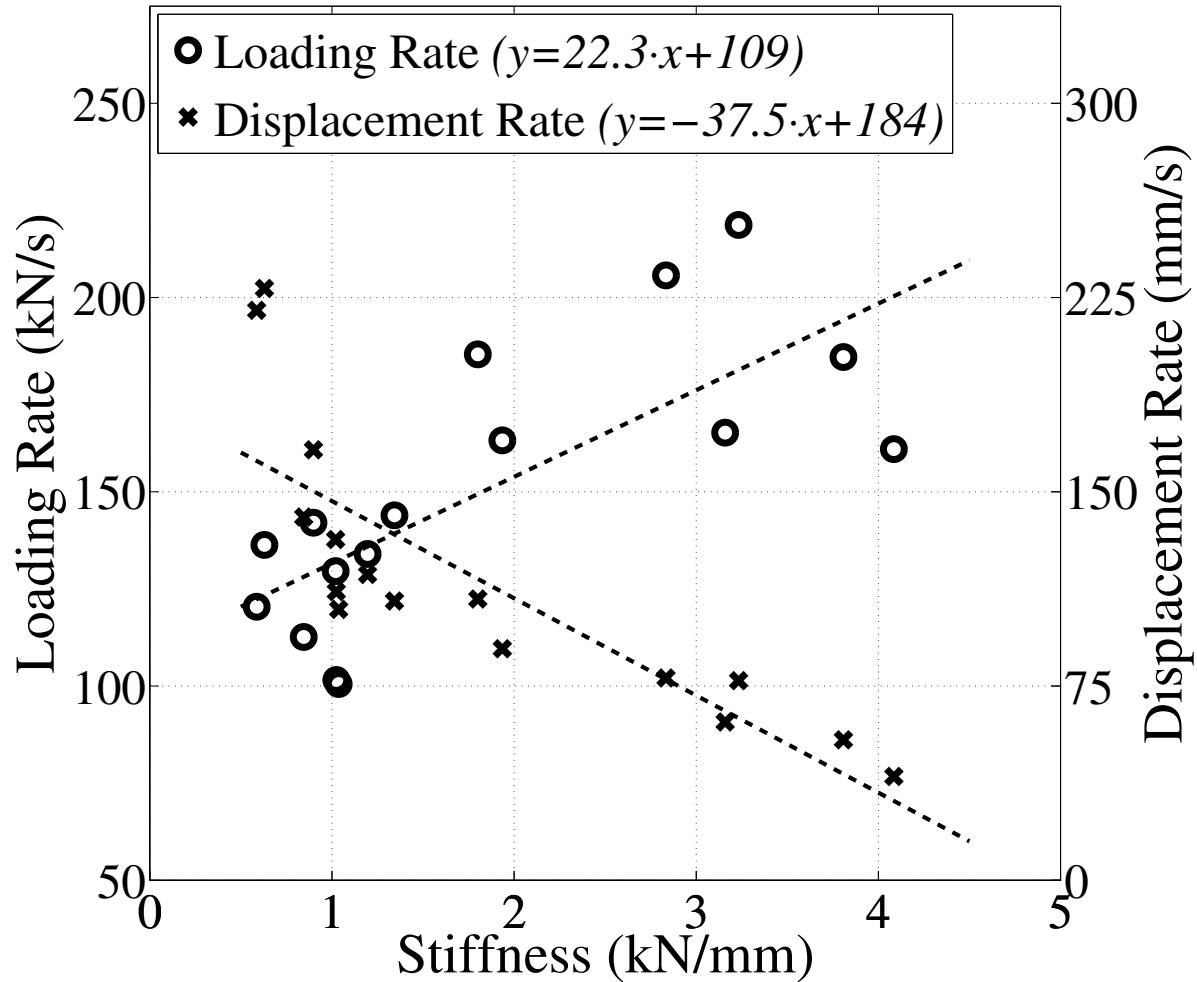


Figure 4.6: Fall:FS loading and displacement rates as functions of stiffness. Data points and regression lines for loading rate ($R^2 = .53$), and displacement rate ($R^2 = 0.66$) are shown. Graphic ©Seth Gilchrist, 2013.

ing and displacement rates as a function of specimen stiffness in the fall simulation tests. Our fall:QS vs. fall:FS results dictate that we accept the null hypothesis that the sub-failure mechanical behaviour is the same between these test conditions. However, the results of the constant displacement rate failure vs. fall simulation failure loading showed significant differences, requiring us to reject the null hypothesis of no difference between quasi-static and impact fall simulation failure characteristics. In the fall simulation tests we found that the average displacement rate decreased with specimen stiffness, while the average loading rate increased.

4.4.1 Fall group viscoelasticity

The lack of differences in the fall:QS vs. fall:FS strain, energy and stiffness data is in contrast to previous work in hip fracture [46, 242], and could be seen to run counter to studies that have shown that bone, as a material, is viscoelastic [37, 70]. However, studies examining bone have shown that the viscoelastic effects are limited, and if bone viscoelasticity alone were responsible for the increase in stiffness seen by Courtney et al. [46], the change in stiffness from 2 to 100 mm/s would be on the order of 25%, rather than the 100% observed. The result in this study of increased aBMD being predictive of changes in stiffness could indicate that bone marrow space, which is presumably less in specimens with more trabecular bone, might be responsible for the changes in stiffness. The material level studies suggest that the viscoelastic response of bone is connected to the organic matrix present [70], which is unknown for the current specimen groups, and to hydrodynamic effects of bone marrow at high strain rates [37]. In the fall simulation tests, the displacement rates were lower at the beginning of the test than at the end (Figure 4.5), but were still two orders of magnitude higher than for the fall:QS group. Therefore, our results suggest that our specimens were not exhibiting strong viscoelastic behaviour under these displacement rates, and this is the reason for the similar stiffnesses, energies and strains measured.

To examine the role of viscoelasticity, an idealized model (Figure 4.7) was used to evaluate anticipated behaviours. If the impact dynamics of the drop tower gantry and top of the spring are ignored, the initial velocity, $v(0)$, of Mass 1 can be calculated to be 2.9 m/s using energy methods (§A.3). The soft tissue model significantly influences the shape of the force pulse, $f(t)$, delivered to Mass 2 and since the soft tissue is omitted in this model, a representative sinusoidal pulse of 250 N over 5 ms ($f(t) = 250 \sin(\frac{2\pi}{0.01} \cdot t)$) was used. If the damping of the proximal femur is given a value of 0, and the pelvis spring is given its laboratory measured stiffness of 50 kN/m, the response of the system is similar to that seen in the actual tests (Figs. 4.6 and 4.8). Computer code for this model can be found in §E.2.

The question then becomes, what would the system look like if damping was increased,

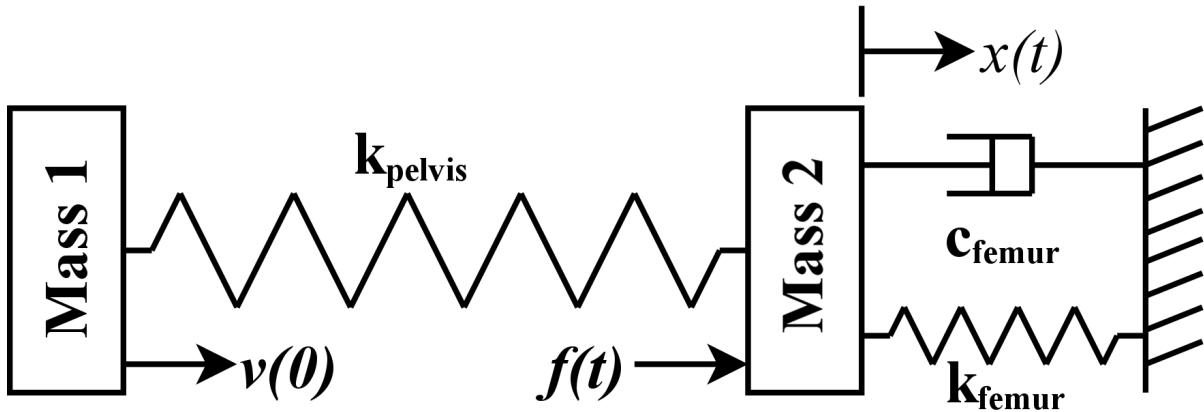


Figure 4.7: The impact fall simulator can be idealized using a series of two springs.

This model is a 2 degree of freedom, second order model. It omits the soft tissue and models the pelvis spring as an ideal spring, and the femur as a spring and damper in parallel. Mass 1 is the combination of the body mass and the top third of the spring mass, and Mass 2 is the combination of the apparatus, bottom third of the spring, and the pelvis mass. Graphic ©Seth Gilchrist, 2013.

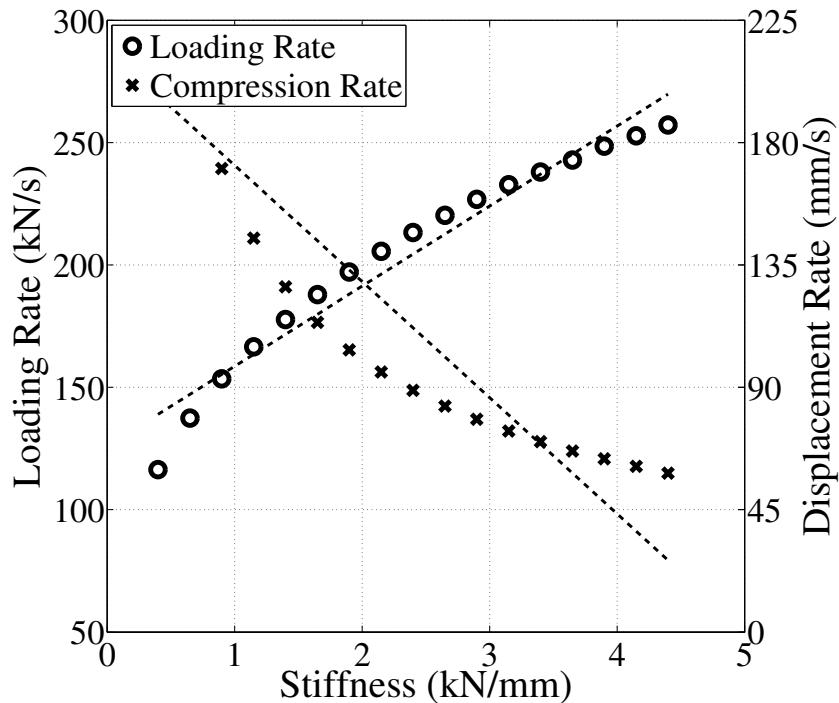


Figure 4.8: A plot for comparison to Figure 4.6 that shows the loading and compression rates vs. stiffness for an ideal spring-mass system up to 1500 N of compressive force. The trend lines in this plot behave in the same manner to those in the experimental data, indicating that the system is behaving in a spring-like fashion. Graphic ©Seth Gilchrist, 2013.

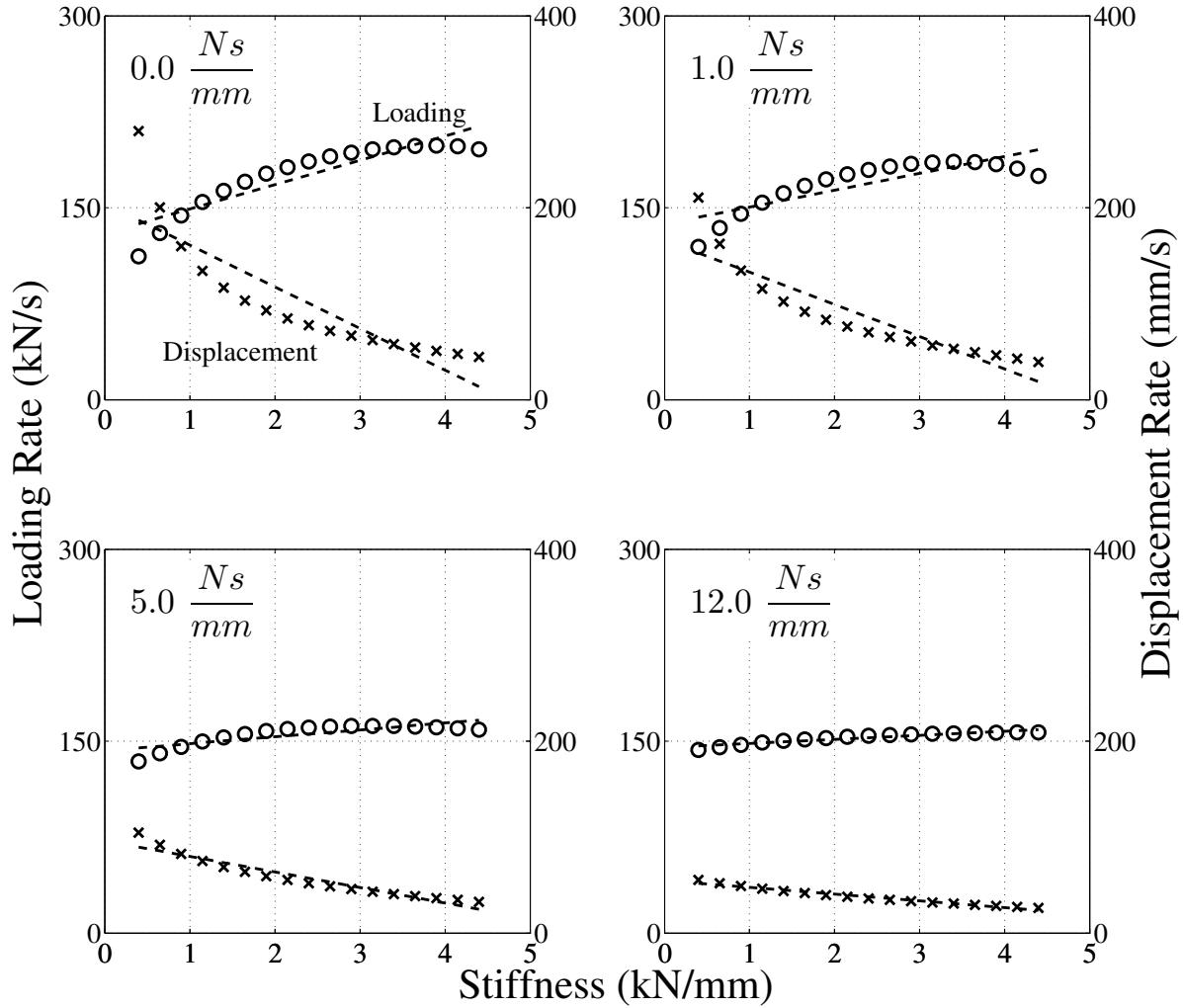


Figure 4.9: Loading and displacement rates vs. stiffness, plotted for four different damping levels. The number in the upper left of each plot gives the value of c_{femur} . Graphic ©Seth Gilchrist, 2013.

simulating high bone viscoelasticity? The same model as above was used, however the force on Mass 2 was omitted because oscillations due to the force pulse were seen to dominate loading and displacement vs. stiffness plots at high damping levels. Since none of the specimens were fractured by the initial force pulses, the artefact created by it would obfuscate the important signals in the plots. Therefore, the initial conditions were set to $v(0) = 2.9$ m/s and $f(0) = 0$, and a number of different damping values were tested (Figure 4.9).

Increased damping has a large effect on the shape of the displacement rate curve, tending to flatten it out and decrease its overall value. This shape change is due to the viscous element

preventing compression of the spring element. In specimens with low stiffness velocity tends to increase rapidly in the absence of damping, leading to high displacement rates. However, in situations where the viscous response dominates, high compressions are never achieved. Increasing the damping decrease the value of the loading rate curve and, at higher damping levels, flattens out the response. This flattening of the load and displacement rate curves at high damping levels occurs because the system is over damped. In the over damped scenario, the loading rate tends towards a constant value, determined by the compression rate of the pelvis spring, and the displacement rate tends to a value of zero.

In the test results (Figure 4.6) the loading rate fit line has a positive slope, and the displacement rate fit line has a negative slope, leading us to draw the conclusion that viscoelastic response is likely minimal.

So how does this fit in with the work of Courtney et al. [46]? The change in stiffness reported by Courtney et al. [46] for old femurs was an increase of 92% when increasing the displacement rate from 2 to 100 mm/s. These data can be used to estimate the damping coefficient of their specimens. Modelling the system used by Courtney et al. [46] as a spring and damper in parallel, driven by a constant displacement rate (Figure 4.10), we can estimate the damping coefficient needed to double the stiffness to a given compressive force (equations for this analysis cab be found in §A.4).

Using a baseline femoral stiffness of 1.5 kN/mm (which is similar to the 2 mm/s value measured by Courtney et al. [46]) and calculating the stiffness up to 2.4 kN (the average strength of the current specimens) shows that to double the stiffness when displacement rate is changed from 2 to 100 mm/s, the damping coefficient, c_{femur} would need to be approximately 12 Ns/mm (Figure 4.11). If, however, the bone was acting in accordance to the strain rate to the power of 0.047-0.06 [37, 52, 134], the corresponding damping coefficient would be approximately 5 Ns/mm.

Returning now to the model of the current experiment, it can be seen that if the bone viscoelasticity was on the order of 5 Ns/mm, as proposed by Carter and Hayes [37], Linde et al.

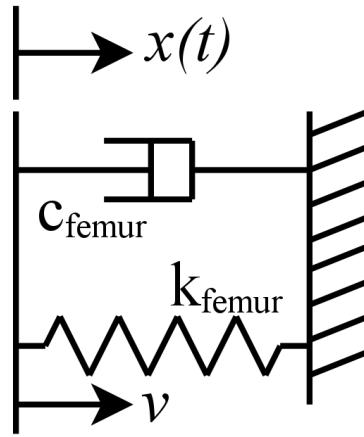


Figure 4.10: An idealized version of the test setup used by Courtney et al. [46]. The stiffness of the construct is the combination of the baseline femur stiffness and the resistance due to the damping. Graphic ©Seth Gilchrist, 2013.

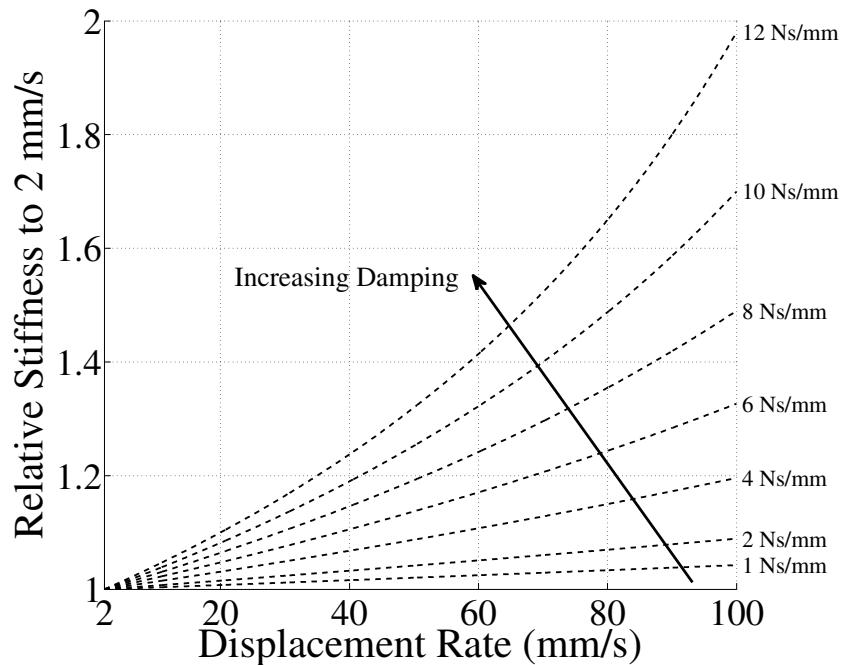


Figure 4.11: Stiffness relative to the 2 mm/s value as displacement rate increases for various damping coefficients. Relative stiffness was calculated to a force of 2.5 kN. Selection of target force or target displacement is important as it changes the nature of the relationship (§A.4). Graphic ©Seth Gilchrist, 2013.

[134] and Crowninshield and Pope [52], then it might be difficult to determine if the specimens in the current tests are acting with that level of viscoelasticity. The model's displacement rate has a distinct negative slope, as does the experiment's, and there is considerable variation in the experiment's loading rate values, which might hide a lower slope than the fit line, similar to the model's. However, if the specimens are acting with the level of viscoelasticity noted by Courtney et al. [46], then one might anticipate nearly constant displacement and loading rates for all specimens, regardless of their baseline stiffnesses. Courtney et al. [46] saw that young specimens with a higher baseline stiffness also doubled their stiffness at the higher displacement rates, indicating that their results were consistent regardless of baseline stiffness. From this, we can conclude that the specimens may have some viscoelastic behaviour, but in the current tests it is not strong enough to heavily influence the response.

4.4.2 Failure and loading responses in the fall simulator

The fall:FS group was significantly different from the fast group in every measure. In absolute value terms the yield force was not much different, but the low stiffness in the fast group meant that the energy absorbed was higher. The mechanism for these differences is not known, but the lower stiffness and higher yield force are incongruent with anticipated viscoelastic effects.

A central result of the study was observation of the variable nature of displacement and loading rates during a fall to the side. Previous investigators have evaluated the effects of displacement rate by testing specimens to failure under multiple, fixed displacement rates [46, 242]. A consequence of fixing the displacement rate is that all specimens, even the stiff ones, are exposed to high loading rates. The average stiffness from Courtney et al. [46] of 3.0 kN/mm at 100 mm/s, results in a loading rate of 300 kN/s. Previous researchers have estimated the peak loading rate in humans falling to the side to be approximately 100 kN/s [124]. The current tests include a compliant pelvis model which acted to mitigate extreme loading rates, leading to an average rate of 150 kN/s. Enforcing a high displacement rate on specimens may reduce the biofidelity when compared to free response. While loading rates of 300 and 100 kN/s are the

same order of magnitude, the differences could affect specimen behaviour in ways that are difficult to predict given the complex geometry and bone material response [37, 87, 134, 198, 251].

In the fall:FS group, displacement rate was related to stiffness, but not to yield force. While a statistical relationship with yield force could not be found, this could be due to a lack of power since the post hoc power of the current experiment is only 0.28. The inverse relationship between stiffness and displacement rate contradicts the outcomes of previous research [46, 242]. This is thought to be a result of the influence of specimen stiffness on the dynamic response. In the dynamic situation that we studied, displacement rate is related to specimen stiffness, with softer specimens experiencing higher displacement rates, which is in agreement with the fundamental mechanics of an undamped, spring-mass system. We believe that observations of previous researchers [46, 242], who noted an increase in stiffness as displacement rate increased, were potentially an artefact of the prescribed constant loading rates on the bones.

4.4.3 Conclusions

Our approach here is markedly different from previous hip fracture fall simulation tests and we believe that it incorporates important improvements over the previous test methods in terms of concordance between the loading situation in real world falls and those in the experiment. Most importantly, the new test method produces both loading and displacement rates as outcome variables that are determined by the biomechanics of the fall and bone structural material properties, rather than dictating one of these parameters *a priori*. Secondly, the use of the same specimen to test two different loading conditions improves statistical power to assess the differences in mechanical behaviours.

While the advantages of the current method are important, there are also limitations with respect to comparison to previous studies. The lack of match pairs for failure mechanics comparison reduced our ability to show differences in failure behaviours. However, comparisons were made to two populations of specimens that were similar to the current one. Addition-

ally, specimens were from a non-fracture population, which may reduce the ability to identify mechanics of bones that actually fracture due to a fall. Further, only surface strain and force-displacement behaviours were assessed, with no insight into the internal mechanics and load sharing between cancellous and cortical bone. This final point is best addressed through the use of computational modelling, which is currently on-going. Notably, the finite element models are developed and validated using data from the fall:QS group. Once the models have been validated for sub-failure loading using that data (for which no differences were seen vis-a-vis the fall:FS group), they will be extended to model the failures observed in the fall:FS group.

This chapter has shown that the mechanical behaviour of the proximal femur is the same in quasi-static, constant displacement rate loading as it is in impact fall simulation. However, it has also shown that prescription of a single displacement rate at the greater trochanter for all specimens is a simplification that could significantly alter the behaviour of the specimen and may have been responsible for the viscoelasticity observed by previous researchers. The following chapter will address the strain and failure characteristics of the proximal femur in the two loading scenarios, addressing the last two research questions outlined in Section 2.1.

Chapter 5

Impact and constant displacement rate loading change the surface strain and fractures of the femur in sideways fall simulation

Knowledge of proximal femur fracture initiation and progression could increase the ability predict who will suffer a hip fracture. Initial failure locations could be important in the effort to increase the sensitivity of hip fracture screening techniques. If failures start in one location, or in proximity to a certain type of structure, it may be possible to develop protocols that target these areas and identify if known morphologies that may lead to structural deficiency exist. Additionally, knowledge of how the initial failure location influences the final fracture type may allow screening for those who could be susceptible to the most serious types of fractures.

Previous researchers have investigated failure locations [57] and fracture patterns [7, 17, 46, 115, 137, 138, 189, 242] in proximal femurs loaded to failure in materials testing machines. These researchers used constant displacement rate loading techniques, some at low rates (<5 mm/s), and some at high rates (100 mm/s), to model a fall to the side. While it

has been shown that increasing displacement rate from 2 to 100 mm/s has only a modest effect on the ultimate strength of the proximal femur [46], there are two questions that remain unanswered. First, does 100 mm/s represent a real fall to the side, and secondly does the deformation of the bone, initial failure location or fracture type change when loading protocol is changed?

In Chapter 4 I investigated the first of these issues, and this chapter concentrates on the questions of how the bone deforms and fails in the quasi-static, constant displacement rate and the impact fall simulations protocols. Chapter 4 investigated what loading protocol was appropriate for mechanical behaviour outcomes of the proximal femur, this chapter will inform researchers of the best protocol for investigating physical damage outcomes. In this chapter, I investigate the strain developed on the surface of the proximal femur, as well as the changes in failure locations and final fracture types, with testing protocol as the independent variable.

Fracture of the proximal femur is preceded by the development of strain within the bone. To assess the difference in the strain of the bone in the quasi-static, constant displacement rate apparatus and those loaded in the fall simulator, the surface strains on the anterior-superior neck of single specimens were compared between the two test conditions using digital image correlation (DIC).

Comparison of the failure of the bones in the two tests conditions were done based on initial failure location, as well as final fracture pattern. To compare these outcomes, two groups of similar specimens were loaded to failure, one using a quasi-static, constant displacement rate protocol, and the other using impact fall simulation. The failure locations and fracture patterns were compared using categorical values and comparative diagrams.

A version of this chapter will be submitted to Osteoporosis International.

5.1 Introduction

Hip fracture is a major orthopaedic burden, with reported averages of 20% of orthopaedic beds being continuously occupied by hip fracture patients [25, 144]. For this reason, researchers

have striven to understand how fractures happen, and how those susceptible to fracture could be better identified. Many experimental and computational tools have been developed to carry out this research, searching for clues that will enable clinicians to identify bones that are susceptible to fracture.

The experimental research has lead to correlations of fracture force with a number of different physical parameters including bone mineral density [23, 57, 137], posture [71, 189], and displacement rates [46, 77, 242]. Researchers who have investigated the influence of displacement rate on the failure load of the proximal femur saw a large increase in stiffness when the rate was changed from 2 to 100 mm/s, but only a small change in failure load [46]. The small change in fracture load may be interpreted as a low dependence of failure mode on displacement rate, but failure is characterized by more than fracture load and the changes in other aspects of the failure, such as location, have never been experimentally investigated.

Bone is known to be viscoelastic [37, 52, 134, 157] and exhibit rate-dependent failure modes [37, 52, 157]. Previous research into the dependence of proximal femur failure on displacement rate showed that increasing rates increased the ultimate strength of the bone in a dose dependent manner [46, 242]. No information was available on the locations of the initial fractures, but one study noted that the fracture type changed with displacement rate stating, “there seemed to be a trend of trancervical fractures with slower loading and subtrochanteric fractures with faster loading” [242].

The dependence of strain development, initial fracture location and final fracture type on method of loading is yet unknown. Previous work has demonstrated a large difference in mechanical behaviours (i.e., maximum force and energy to failure), but has not addressed changes in fracture locations or fracture types. In this paper we aim to compare the strain fields in single specimens loaded using both quasi-static, constant displacement rate and fall simulation techniques. We will compare digital image correlation (DIC) obtained maps of the minimum principal strain on the anterior-superior neck at the same compressive force in both techniques. Additionally, fracture patterns and locations will be compared between two groups of speci-

mens loaded to failure at constant displacement rate of 2 mm/s [173], or in a fall simulator designed to replicate the force-time response of the humna body during a fall to the side [76]. We hypothesized that (i) the strain fields observed on the femoral neck will be different; (ii) the initial fracture locations will be different; and (iii) the final fracture patterns will be different between quasi-static, constant displacement rate and fall simulation testing.

5.2 Materials and methods

Twenty-four, fresh frozen human femora were acquired from consenting donors, in accordance with the ethics protocols of the University of British Columbia. The donors had an average age of 77 (SD = 10) years and consisted of 19 females and 2 males. Details of the 21 of the specimens that also had force-displacement data are available in a previously published companion paper [77]. Three additional specimens were included for this analysis and their details are listed in Table 5.1. The specimens were cleaned of soft tissue and periosteum and a white-dot on black-background speckle pattern was applied to the anterior superior neck using an airbrush (VL, Paasche, Chicago, IL). The black background was chosen to minimize glare and improve the DIC data quality. The specimens were imaged using a clinical DXA scanner (QRD 4500W, Hologic, Bedford, MA), with 4 kg rice to simulate soft tissue [218] to determine aBMD and osteoporosis classification. This group of specimens will be referred to as the *fall* group because they were fractured in the fall simulator. The fall group specimens were loaded in two ways. First, they were loaded to 50% of their total aBMD predicted failure load [23] in a materials testing machine, at a constant displacement rate of 0.5 mm/s. These test results will be referred to as the fall:QS results, due to the quasi-static nature of the loading. Following this, the fall specimens were loaded to failure in a fall simulator. The results of the latter test will be referred to as the fall:FS results, due to the use of the fall simulator.

In addition to the fall group, 20 fresh frozen human femora were loaded to failure at a constant displacement rate of 2 mm/s in a materials testing machine [173]. These specimens consisted of 20 femurs from 15 female and 5 male donors, with an average age of 77 (SD =

Table 5.1: Donor information for additional specimens included in this study that are not included in [77].

Specimen	Side	Age	Gender	Total aBMD (g/cm ²)	OP Status
22	R	73	F	0.872	Normal
23	L	72	F	0.964	Normal
24	L	85	F	N/A	N/A

13) years, and were acquired and handled in accordance with the University of Calgary’s ethics protocols. This group of specimens will be called the *slow* group because they were failed using a slow, constant displacement rate. Soft tissue and periosteum were removed and DXA scans were acquired using a clinical scanner (QDR4500, Hologic, Bedford, MA) to determine aBMD and osteoporosis state. Further details of this specimen population are available in their original publication [173].

All specimens from both groups were potted in polymethylmethacrylate (PMMA, Bosworth Co, Skokie, IL), in the literature-standard sideways fall orientation of 10° adduction and 15° internal rotation [46, 57, 137, 148, 149, 189], with 290–305 mm from the distal pivot point to the most proximal point on the femoral head. PMMA caps consisting of 20 g of material formed into 3.5 cm diameter disks, were moulded to the medial head and lateral trochanter with parallel surfaces to ensure even load application and prevent non-clinical isolated compression failures of the femoral head and lateral trochanter.

The bones in the fall group were loaded in two successive tests. First, a materials testing machine (8874, Instron, Norwood, MA) applied a sub-failure, quasi-static load at a constant displacement rate. A preloaded of 100 N was applied over 5 s and held for 0.5 s before loading the specimens to 50% of their total aBMD predicted failure load [23] at 0.5 mm/s. Two high-speed video cameras (Phantom V12.1, Vision Research, Wayne, NJ) imaging at 100 frames per second (fps), and 1280x800 px (17 px/mm), observed the anterior-superior femoral neck for DIC. Following this test, the specimens were imaged using the planar X-ray to confirm no damage and transferred into the inertial fall simulator.

The fall simulator consisted of masses and springs selected to mimic the response of the body in a fall to the side. The details of its design have described in detail elsewhere [76], but briefly, it simulated the loading in a fall to the side using an lumped parameter model consisting of a body mass (32 kg), a spring to replicate the pelvis (50 N/mm), mass to replicate the lateral pelvis and proximal femur (1.98 kg), and closed cell foam to replicate the soft tissue over the trochanter (19 mm). The simulated fall was recorded by four high speed video cameras. Two cameras observed the anterior-superior femoral neck at 10,000 fps (Phantom V12.1), and a resolution of 1024x800 px (15 px/mm) for DIC measurement. One camera (Phantom V9) recorded the impact hammer and lateral trochanter at 9,216 fps and 576x288 px (5 px/mm) for trochanter displacement measurement, and lastly, one camera (Phatom V9) qualitatively observed the posterior of the specimen at 6,006 fps and 480x480 px for fracture monitoring.

The specimens in the slow group were loaded in the same materials testing machine, at a constant displacement rate at the greater trochanter of 2 mm/s, until fracture. The tests were observed using two high speed video cameras (Phantom V12.1), aimed at the anterior and superior aspects of the specimens, recording at 500 fps and 1280x800 px. No DIC data were acquired for the slow group.

For each specimen in the fall group, the surface topologies and minimum principal strains were measured using commercially available DIC software (DaVis v8.3, LaVision, Göttingen, Germany). Each specimen in the fall group had two DIC data sets consisting of the topology and minimum principal strains on the surface of the bone in the fall:QS and fall:FS conditions. For each specimen in the fall group, DIC-derived strains and surfaces from both tests were extracted at the maximum force applied in the fall:QS test. An iterative closest point (ICP) algorithm was used to align the surface topologies, and strain data were then compared on a point-to-point basis.

Alignment of the surface topologies and analysis of the differences in strains were conducted using a custom C++ programming language (C++) algorithm (available in §E.3) utilizing the Visualization Toolkit (VTK) (5.10, Kitware, Clifton Park, NY). Data were exported

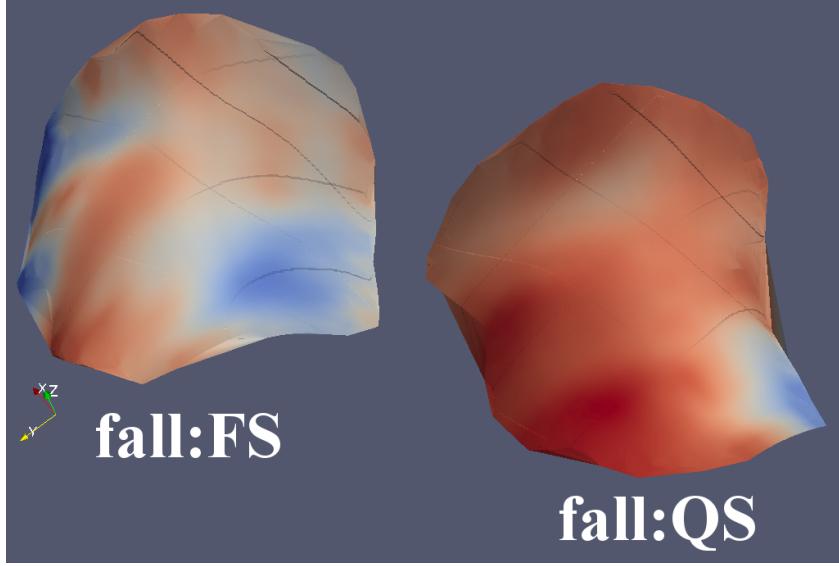


Figure 5.1: The original positions of the surfaces from the quasi-static and fall simulation tests, coloured by minimum principal strain. Image ©Seth Gilchrist, 2013

from DaVis, as two raster files, one with topology data, and the other with strain data. The topology file contained (x, y) data pairs and a z height at each location. The strain data file contained minimum principal strain at the same (x, y) locations used in the topology file. The two DaVis topology surfaces were read into memory and 3D meshes were created using Delaunay triangulation. The strain files associated with each surface were then read in and the strain data were applied to each node in the triangulated mesh. At this stage, two unaligned surfaces with strain data were contained in memory (Fig. 5.1).

Because the radius of convergence for ICP algorithms is small, a manual pre-alignment was required. The pre-alignment was performed in ParaView (3.98.1, Kitware, Clifton Park, New York) and consisted three translations and three Euler angle rotations, which were passed to the ICP alignment program at runtime.

While the profiles of the two surfaces were close, they did not match exactly, meaning that the data from one could not be readily compared to the other after the registration registration (Fig. 5.2). To overcome this issue, an extrusion method was used to transfer data from the ICP target surface (the one that was stationary throughout registration) to the transformed surface.

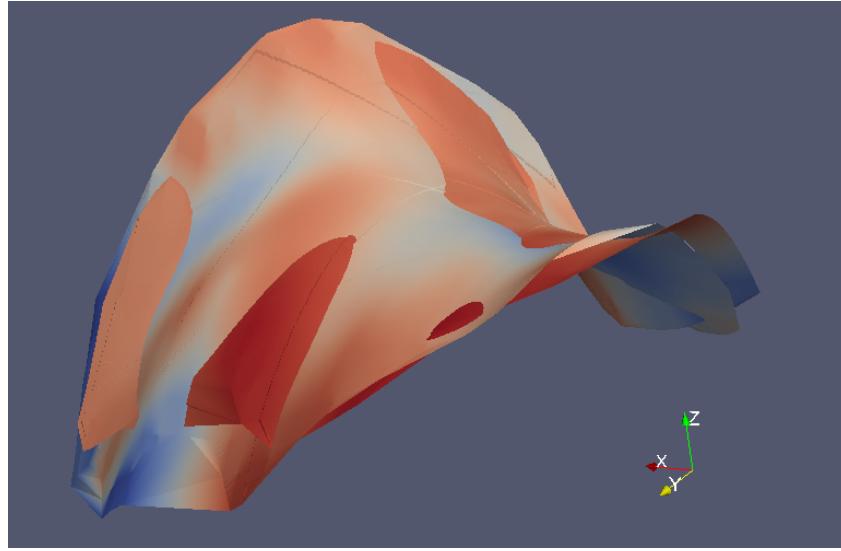


Figure 5.2: DIC surface profiles after alignment using the ICP method. Image ©Seth Gilchrist, 2013.

Since the target surface was stationary throughout the registration, it was still aligned such that there were no occluded sections when viewed along the negative z -direction. Taking advantage of this, the target surface was extruded in the z -direction to create a volume that enveloped the transformed surface (Fig. 5.3). Once the extrusion had been performed, a copy was made of the transformed surface (including the strain data). This copy was then trimmed using the bounds of the extruded target surface volume, creating a third surface (the *master* surface) that consisted of only the overlapping regions of the two original surfaces (Fig. 5.4).

After creating the master surface, the nodes of the master surface were used to probe the strain data in the extruded volume. The strains were then stored in a new, point-associated dataset on the master surface. At this stage, the master surface consisted only the overlapping regions of the original surfaces and contained all of the strain data from the original surfaces. Calculations and statistics were performed on a point-by-point basis on the master surface (Fig. 5.5).

The fall:FS and slow group videos were reviewed to identify initial failure location, which was defined as the first location where yielding or crack formation was observed. Initial failure locations were classified as one of, (a) superior neck, (b) inferior neck, (c) intertrochanteric,

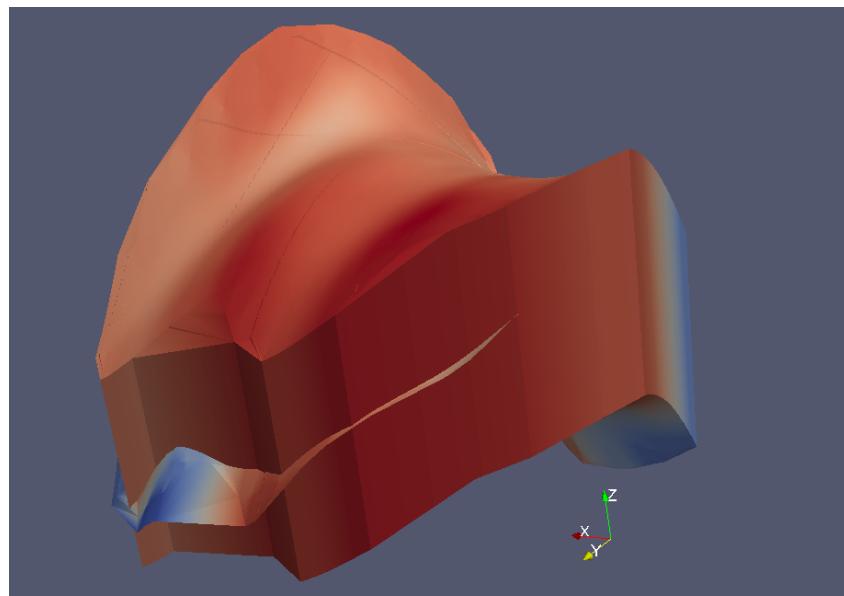


Figure 5.3: The ICP target surface was extruded in the z-direction, carrying all data with it. This ensured overlap with the transformed surface for data transfer.
Image ©Seth Gilchrist, 2013.

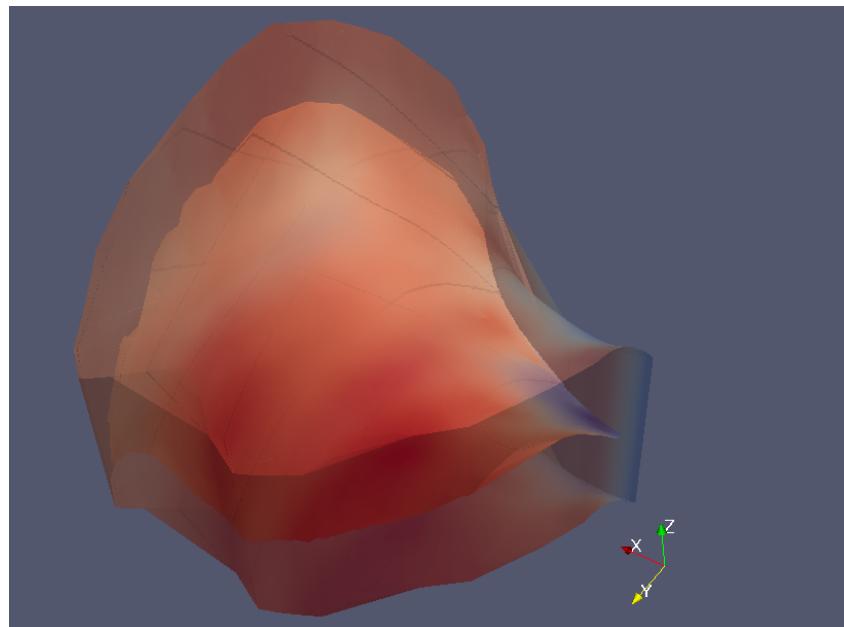


Figure 5.4: A trimmed version of the transformed surface was used as a master surface to compile all of the strain data from both the target and transformed surface in the region of overlap. Image ©Seth Gilchrist, 2013.

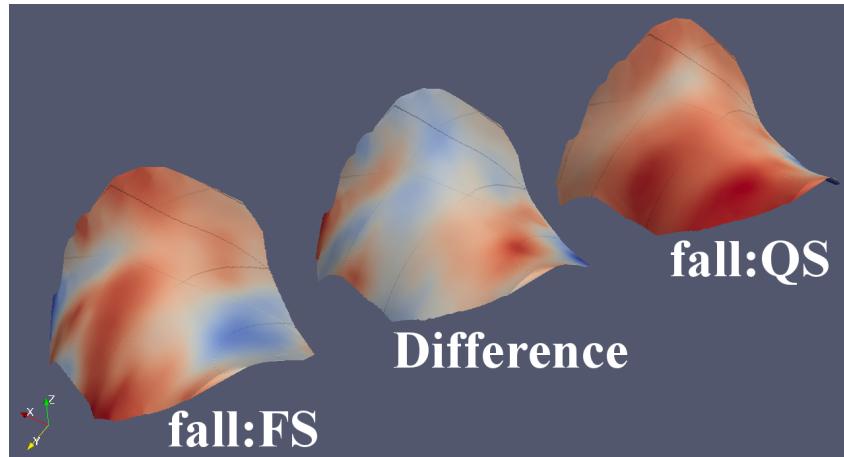


Figure 5.5: The fall simulator, quasi-static and master surfaces. In this image the master surface is showing the difference in strain between the fall:FS and fall:QS surfaces. Image ©Seth Gilchrist, 2013.

(d) trochanteric, or (e) no failure. If the failure location could not be seen in the video, the specimen was not included in the statistical analysis for initial failure location.

Fracture was differentiated from failure in the analysis. Fracture and was characterized by the loss of ability to withstand load and formation of fracture lines, whereas failure was characterized as formation of visible surface cracks, even when load was still supported. Classifications were done by reconstructing the fragments and observations of the high speed video and were carried out by a fellowship-trained, orthopaedic trauma surgeon (author PG) and classified as, (a) neck (N), (b) intertrochanteric stable (IS), (c) intertrochanteric, unstable 3-part (IU3), (d) intertrochanteric, unstable 4-part (IU4), (e) subtrochanteric (ST), or (f) no fracture (NFX).

The strain maps between the two loading conditions for the fall group were compared using match-pairs, Student's t-tests, where each point was considered to have two conditions, one for each loading scenario. The fracture types and locations between the fall:FS and slow group were compared using two-tailed Fisher Exact tests. Additionally, initial failure locations were compared to final fracture lines to see if the initial location predicted the final fracture location. Because the initial failure locations and fracture classifications were not comparable on a one-to-one basis, the analysis was simplified by grouping both classifications into either *neck* or *other* and placing the results into a 2x2 matrix with columns of initial locations and rows

of final fracture location. In the case were multiple fractures were observed, neck fractures were given precedent over trochanteric fractures, i.e., if both neck and trochanteric fractures were observed, the specimen was scored as a neck fracture. A chi-squared test was used to determine if the initial failure location predicted the final fracture location. Significance was set at $\alpha = 0.05$ for all tests, and p -values were adjusted for multiple comparisons using Holm's method.

5.3 Results

The fractures in the fall group showed significant variability (Fig. 5.6) and comminution, while the slow group (Fig. 5.7) showed relatively regular fracture lines, with little or no comminution. In both tests crushing of the greater trochanter was seen under the PMMA pad.

Fracture types and initial location data (Tables 5.2 and 5.3) showed that the two groups had significantly different fracture types ($p < 0.01$), but not significantly different initial failure locations ($p = 0.13$). The fall:FS group had more intertrochanteric fractures, which tended to be unstable. Initial failure location predicted the final fracture location in the slow group ($p = 0.037$), but not in the fall:FS group ($p = 0.40$, Table 5.4).

The slow group specimens suffered fractures in all cases. This is to be expected as the test protocol called for displacement of the greater trochanter well beyond the elastic range for the bones. In the fall group, failure was seen in all cases but fracture was seen in only 22 of the 24 specimens. In the two non-failure cases (specimens 3 and 19) cracks were observed in the video and post fracture analysis, but the specimens did not fracture, meaning that the bones retained load bearing capacity and had no open fractures.

The comparison of the surface strains in the fall:QS and fall:FS tests at the same load, and location showed that there were significant differences between the test conditions ($p < 0.01$, Fig. 5.9). When all fall group data were pooled, the fall:QS strain was $221 \mu\epsilon$ higher than the fall:FS strain ($p = 0.45$, 95% CI = (921, -477)), which was 6.9% of the average quasi-static strain of $-3205 \mu\epsilon$.

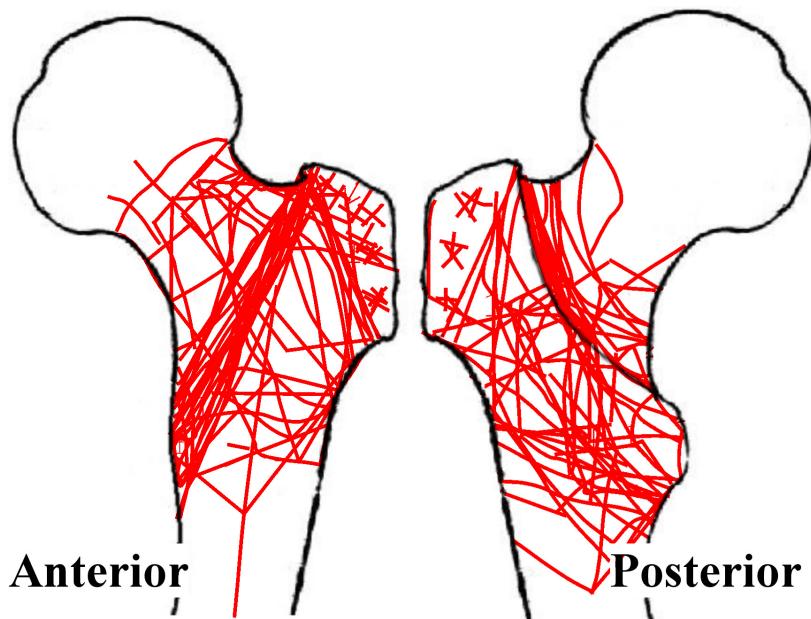


Figure 5.6: Fracture lines observed in the fall:FS group compiled onto a single femur profile. Graphic ©Seth Gilchrist, 2013.

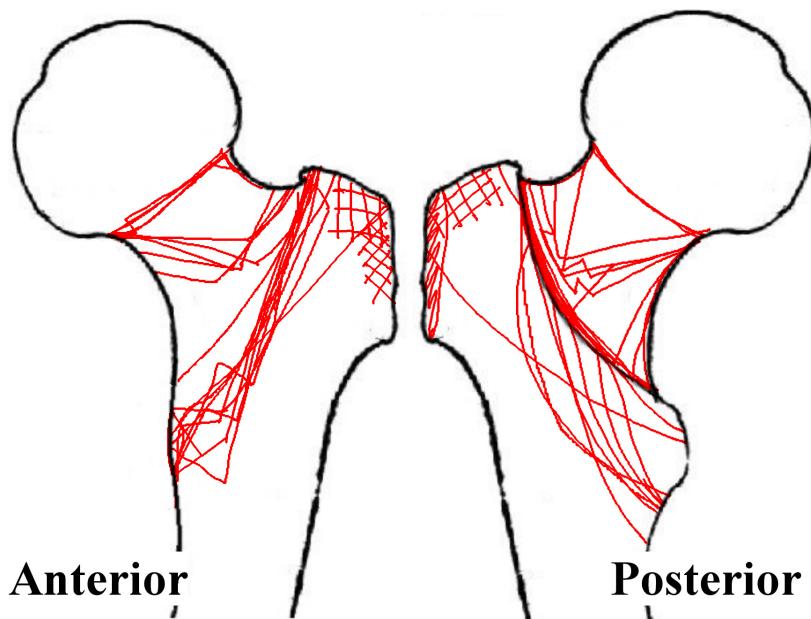


Figure 5.7: Fracture lines observed in the slow group compiled onto a single femur profile. Graphic ©Seth Gichrist, 2013.

Table 5.2: Initial failure locations and final fracture types for the specimens in the fall:FS group

Specimen	Fracture Types						Initial Failure Location
	N	IS	IU3	IU4	ST	NFx	
1	✓			✓			Intertrochanteric
2		✓					Intertrochanteric
3						✓	Superior Neck
4	✓		✓		✓		Trochanteric
5			✓				Intertrochanteric
6	✓		✓				Trochanteric
7			✓				Trochanteric
8	✓						Superior Neck
9				✓			Trochanteric
10	✓			✓			Trochanteric
11	✓						Superior Neck
12			✓				Trochanteric
13			✓				Intertrochanteric
14						✓	Trochanteric
15		✓					Intertrochanteric
16	✓		✓				Intertrochanteric
17		✓					Intertrochanteric
18	✓		✓				Intertrochanteric
19		✓					Trochanteric
20		✓					Intertrochanteric
21			✓				Trochanteric
22	✓				✓		Superior Neck
23	✓	✓					Trochanteric
24	✓						Not Available

Comparison of the strain fields showed that they were typically correlated, but often exhibited an offset in the measured strain (Fig. 5.9). The average RMS error in the surface matching was 5.3×10^{-11} m, indicating that the majority of the 50,000+ points in each surface were nearly coincident after the registration. Some of the strain comparisons displayed “branches” on the strain-strain plots (specimens 15, 24, 25 and 26), which were indicative of a redistribution of the strain field. To illustrate this, specimen 24, which showed two branches in the strain-strain plot (Fig. 5.9), was seen to have two locations where the strains were different, one on the lateral neck and the other in the sub-capital region (Fig. 5.8). The locations of the differences

Table 5.3: Initial failure locations and final fracture types for the specimens slow group

Specimen	Fracture Types						Initial Failure Location
	N	IS	IU3	IU4	ST	NFx	
1			✓				Intertrochanteric
2			✓				Intertrochanteric
3		✓					Intertrochanteric
4	✓						Superior Neck
5		✓					Unknown
6		✓					Intertrochanteric
7	✓						Superior Neck
8	✓						Superior Neck
9		✓					Intertrochanteric
10	✓						Superior Neck
11		✓					Unknown
12	✓						Superior Neck
13	✓						Intertrochanteric
14	✓						Unknown
15		✓					Intertrochanteric
16	✓						Superior Neck
17	✓						Inferior Neck
18		✓					Unknown
19		✓					Superior Neck
20		✓					Intertrochanteric

Table 5.4: Initial failure locations and final fracture locations in each group

Final	Initial		
	Neck		Other
	Neck	3	7
Final	Other	1	12

Final	Initial		
	Neck		Other
	Neck	7	1
Final	Other	1	7

Table 5.5: Published fracture locations for laboratory experiments

Author	Neck	Trochanteric
Backman [7]	36	117
Lotz and Hayes [138]	6	6
Weber et al. [242]	8	7
Courtney et al. [46] (Elderly)	5	4
Courtney et al. [46] (Young)	6	3
Pinilla et al. [189]	8	3
Keyak et al. [115]	4	10
Lochmüller et al. [137]	54	40
Totals	127	190

in strain were qualitatively examined in specimens 15, 24, 25 and 26 and the redistributions did not seem systematic, rather they appeared random.

5.4 Discussion

This experiment studied the strain distributions, failure locations and fracture patterns between quasi-static, fixed displacement rate and impact fall simulation loading protocols. The failure study showed that final fracture patterns were significantly different between the two groups, even though the initial failure locations were not. This leads us to reject the null hypotheses that fracture types are the same between quasi-static and fall simulation testing, and accept the null hypothesis that initial failure locations are the same in both testing methods. The strain between the two sub-failure tests conditions were significantly different on a point by point basis, but when pooled together there was no significant trend. Therefore, testing of the hypothesis that there would be a significant difference in strain lead to two solutions, firstly, we reject the null hypothesis that there would be no difference in strain on any given proximal femur between the two testing methods, however, we accept the null hypothesis that there is no significant difference between the testing methods on a population level.

Comparison of the fracture outcomes of the current experiments to those published in the clinical literature is complicated by the lack of post fracture x-rays in the current tests. Clinical fractures are diagnosed and classified using radiological examination, however, in the fall

Specimen 24

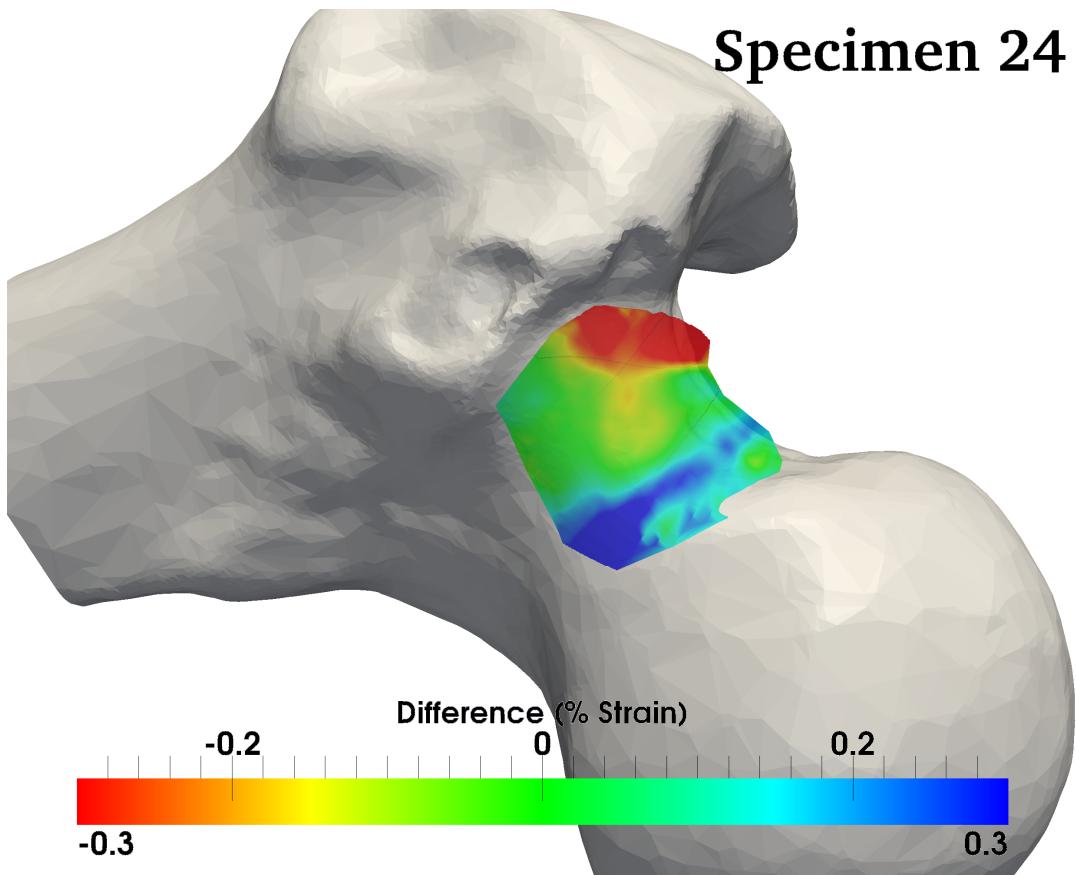


Figure 5.8: Fall:QS minus fall:FS minimum principal strains at the same load overlaid on the specimen. In this case, the fall:FS tests had higher strains in the lateral neck and lower strains along the subcapital line. The average strain difference for this specimen was $39 \mu\epsilon$, with a range of $[-770, 470] \mu\epsilon$. Graphic ©Seth Gilchrist, 2013.

Table 5.6: Published fractures locations for clinical studies

Author	Neck	Trochanteric
Michelson et al. [159]	63	83
Lyritis [145]	791	995
Sirois et al. [213]	783	1584
Lefaivre et al. [128]	22313	19677
Poole et al. [191]	36	39
Totals	23986	22378

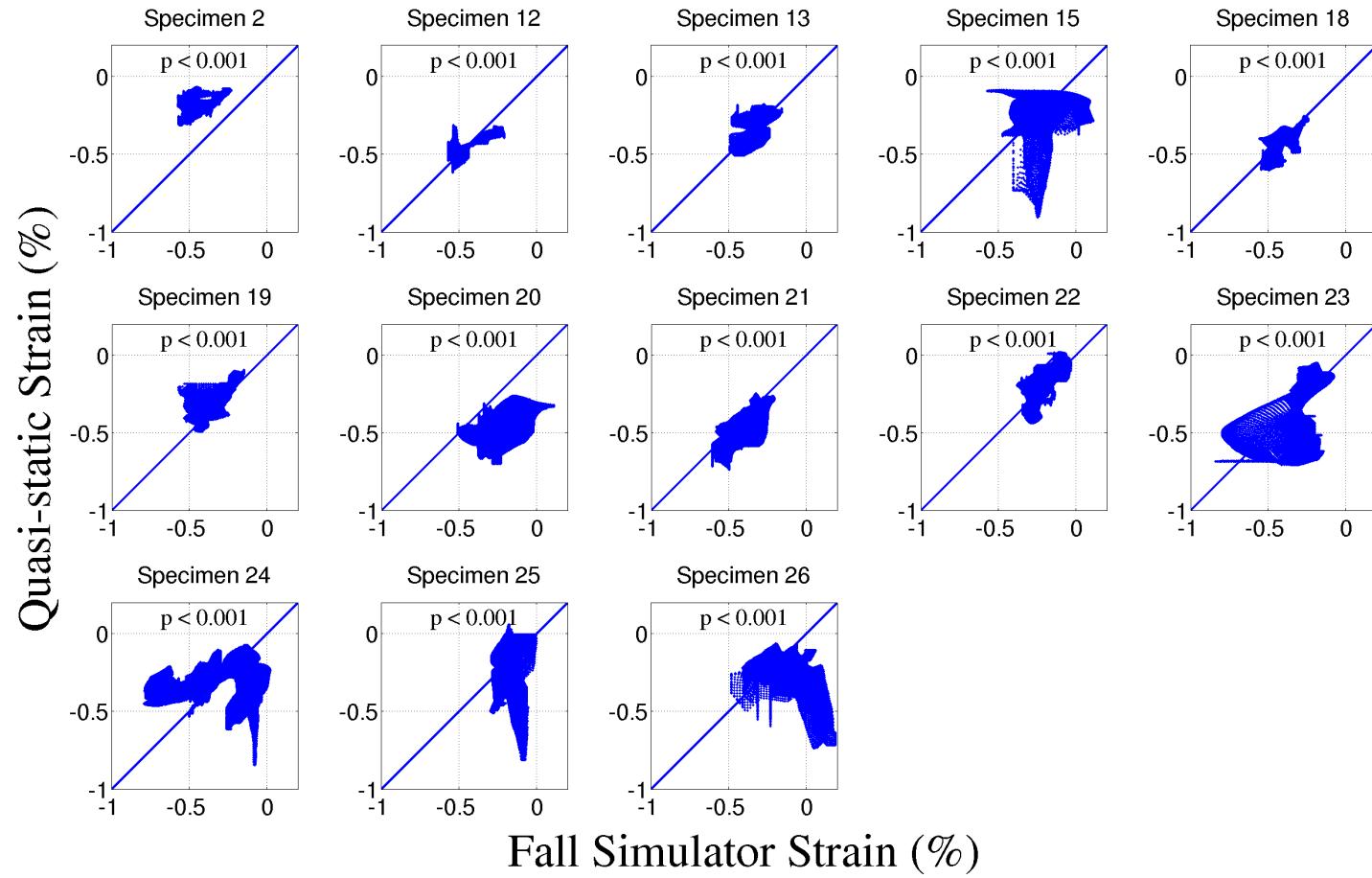


Figure 5.9: Strain-strain plots comparing minimum compressive strain between the fall:QS and fall:FS groups. Each point in each plot is at the same location and load, and the p -values are the results of a paired t-tests on all points in the point cloud. Graphic ©Seth Gilchrist, 2013.

group, the fall simulator often destroyed the greater and lesser trochanter of the specimens post fracture, making radiological examination impossible. That said, classification as either intra- or extracapsular was possible from fragment analysis and video observation and the current results are in-line with previous tests and clinical observations (Tables 5.5 and 5.6). The fall:FS fractures were split 50/50 between intra- and extracapsular fractures, while the slow group showed a slight tendency toward extracapular fractures. Both methods gave distributions of fracture types that are representative of what has been observed in the clinical literature.

Examination of the strain values on the surface of the specimens at the same compressive load while being tested with the two protocols showed that average strain over the analysis regions was not statistically different between the two testing protocols, however, no single specimen had the same strain. These results seem contradictory, however, they are indicative of a random variation in surface strain due to the change in loading protocol. A cohort of specimens loaded quasi-statically will show, on average, the same strain as those loaded in the impact fall simulator, however, each individual specimen will have a different response in the two tests. This result has important implications for study design. If a case-series study is examining the change in strain due to an intervention in which two sufficiently sized cohorts of specimens are treated in two different ways, quasi-static, constant displacement rate simulations will be sufficient to study the effects of the intervention. However, if a study is examining changes in a specimen specific manner, e.g., trying to identify the correct intervention to decrease surface strains in a particular specimen, then impact fall simulation is warranted.

It should be noted that alignment errors in the registration process could have led to differences being detected between specimens that were not real. If the DIC-measured surfaces were not perfectly aligned, the strain gradients on the surface could manifest as a reconfiguration of strain in the strain-strain plots. However, each comparison in this study was individually inspected for two features: i) surface size, extent and off-plane features, and ii) location of strain differences. The first item was important for ensuring a reliable registration. Surfaces that were either flat or described only a portion of a cylindrical surface were eliminated from the

comparisons. Features that were considered sufficient for inclusion of a surface were extension of the surface onto the femoral head or trochanter in a way that provided an out of plane, or off-cylindrical axis registration target. The second item was a post-processing analysis aimed at identifying if the branches seen in the strain-strain plots were in close physical proximity. If the locations of strain difference were physically near each other, the surfaces were re-evaluated in terms of (i) and either eliminated if the surface was insufficient for accurate alignment, modified if an identifiable feature was driving a misalignment, or accepted if the registration was driven by a sufficiently complex surface.

A few specimens showed redistributions of strain on the anterior superior femoral neck that were, in some cases, quite large. This reasons for this are unknown, but could be related to failures of cancellous bone below the surface early in the loading which are not visible in the current tests. Failure within the structure could cause the load path to be altered, which could manifest as a change in pattern and magnitude on the surface.

To our knowledge, this study is the first of its kind to compare the failure locations, and fracture patterns in two populations tested at considerable extremes of loading protocols, as well as to examine the strain fields of individual specimens loaded to sub-failure using different protocols. Like all *ex-vivo* fracture experiments, this study has some limitations that we did our best to mitigate, but must be considered during interpretation. First of all, the small number of specimens reduces the power to identify differences between the fall and slow groups. That said, most of the comparisons between the two methods resulted in statistical results that were sufficiently powerful that more specimens would not have been required. Once exception to this may be the initial failure location. It was not seen to be different, but the *p*-value of 0.13 could be considered a trend worth further investigation. Secondly, the impact testing protocol resulted in the destruction of the specimens and consequently post fracture anterior-posterior (A-P) x-rays were not possible. Since clinical fracture evaluation is carried out using A-P x-rays, the results of this work are not directly comparable to clinical outcomes. However, the classification into intra- and extracapsular fractures should be comparable due to the distinctly

different locations of these portions of the proximal femur. Additionally, some of the fractures were such that they may not have been observed using a-p x-ray examination. These fractures were hairline fractures along the superior femoral neck, in the coronal plane and may represent failures not yet documented in the clinical literature. Finally, the strain mapping was carried out only on the superior-anterior surface of the femoral neck, which may not be representative of the proximal femur as a whole. This fact limits the applicability of the no difference when pooling data result, but it does not limit the applicability of the observations on individual specimens where we can say that in the critical location of the anterior-superior neck, the response is different between the two loading methods. Additionally, the density of the strain data are much higher than previous tests utilizing strain gauges and thus provides a rich dataset for statistical analysis.

Overall, this work indicates that more biofidelic replication of the fall and fracture scenario may benefit the examination of hip fracture progression. The changes in final fracture types and the redistribution of strain in some specimens may be indicative of internal mechanics that have heretofore gone unobserved. The examination of fracture patterns in two similar populations tested in two very different scenarios allowed us to understand how loading mechanism affects failure through rate-dependent failure changes. Our interpretation of the results lead us to recommend that experiments in which single specimens are treated in different manners with the goal of influencing strain development or fracture behaviour use fall simulation to test outcomes. However, if general changes in a cohort of specimens are to be observed, quasi-static testing at constant displacement rate, or impact fall simulation can be used. We believe that future finite element research should concentrate on incorporating strain-rate dependency in material properties, and create specimen-specific models whose fracture predictions are validated using applicable loading rates. These models could be used to identify the critical load path in a fall to the side, and investigate the effects of changing this load path due to local failures within the cancellous and cortical bone. This technique could help understand the phenomena of how an initial failure in the trochanter can lead to fracture of the neck in fall

simulation. Ultimately, we aim to understand how forces are transferred in a compromised femur, potentially giving us the tools to develop intervention and screening technologies that would reduce the burden of this devastating condition.

Chapter 6

Communication between mechanical and finite element modelling

Mechanical and finite element (FE) modelling are closely linked. Much of the work done in FE modelling is a continuation or examination of phenomena observed in experimental modelling, and many experiments are conducted in support of FE model validation. A considerable portion of the work in this thesis was done in conjunction with the FE modelling team under the supervision of Dr. Ben Helgason at ETH, Zürich, Switzerland. There was significant effort expended to ensure that the FE modelling group had all the data required to create models that could be readily compared to the experiments performed. This chapter gives a brief description of the data flow between the two modelling methods and efforts made on the experimental side to facilitate that flow. Informally, we have dubbed this the *data pipeline*. I hope that future FE-experiment collaborations can take advantage of my experience in this regard.

6.1 Documentation from start to finish

The most critical aspect in creating an experiment that can be used for FE validation is documentation and transparency. From apparatus, to data and analysis routines, all aspects of the experiment must be fully described and accessible to the modelling community (Figure 6.1).

Details of experimental apparatus are useful in defining boundary conditions (BCs) of FE models. The level of detail required in an FE model may begin modestly, but as the model is refined researchers may need to integrate very high levels of detail in some aspects of the setup. Since it cannot be easily anticipated where certain phenomena originate, experimental modellers must make an almost arbitrary level of detail available to the FE modellers. The purpose of an FE model is frequently to examine aspect of the experiment that are difficult to explain based on current knowledge, which means that current knowledge should be used cautiously in determining what to include in a model. As computational power increases, FE researchers are including more apparatus in their models in an effort to close the gap between computational and experimental models; details provided by the engineers who designed the experimental apparatus can help the two models converge.

Computer code and analysis routines is another area where concerted effort to organize and communicate can pay off. FE researchers want data packed in a manner that makes data manipulation and extraction easy to automate. Additionally, code annotation and inclusion of details like units used in a given portion of the analysis can be invaluable to an FE modeller who is working remotely from the experimental researchers. This kind of annotation and documentation of code is time consuming and can be challenging. Experimentalists often use one-off custom code, with the routines designed to evaluate specific variables of concern for a given test. Each researcher typically develops their own code, based on their own needs, that they understand well and rarely pass on to future experimentalists. Computational researchers are used to developing and documenting code with the intent of passing it to the next researcher. Indeed, very little progress would be made in the field if each FE researcher had to start from scratch. Additionally, in the computational world there is an understanding that no code is perfect, as evidenced by the continual releases of new versions of old software. Experimentalists can be overtly aware of limitations, coding discrepancies and the nuances of running their software, which can make them reluctant to open-source it for fear of criticism [9]. To address the latter fear, documentation must be an integral part of the analysis code development process, with

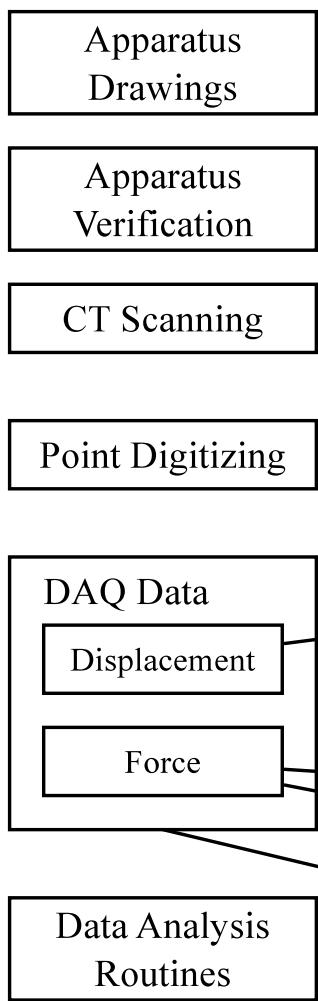
the expectation that it will be released, along with the data, to another research group. This was handled in the current work by developing a series of analysis classes that were portable and self contained (§E.1). Each class object contained the raw data, the final data, filtering and interpolation routines that yielded the final data, and routines to analyse and extract meaningful data, as well as variables concerned with those analyses. This transparency and documentation allowed sending the data to researchers who were unfamiliar with the tests, and allowing them to access the full analysis, as well as documentation (including information like units) for each test and specimen.

6.2 Additional data for FE modelling

As mentioned in the previous section, the origins of phenomena observed in an experiment are often difficult to predict. One potential source of obfuscation is poorly or incorrectly applied BCs. This can be addressed by careful recording of experimental setup using technologies like computed tomography (CT), surface digitization and landmark point digitization. The majority of experimental models do not need for CT imaging and surface topology of specimens. A-P radiographs and DXA images are used by researchers to give a clinical context to the specimens being tested. When working with FE models, detailed geometric, densitometric and orientation data are needed, requiring collection of CT images, as well as bone and test equipment landmarks.

Once the drawings of the apparatus, and geometry and density of the bones are obtained, an FE model can be constructed and appropriate fixed boundary conditions applied. The effector of an FE model is typically a displacement or force boundary condition, the alignment of which is critical to the output of the model. To align the force or displacement application in the computational model, the orientation of the bone during the actual experiment must be obtained. This was done by collecting the (x,y,z) coordinates of points on the surface of the bone and loading apparatus using an Optotrak (Certus, NDI, Ontario, Canada) with a digitization wand (Figure 6.2). Points were collected on each part of the specimen and apparatus and saved

Experimental Steps



Finite Element Steps

Figure 6.1: Flow of data between experimental and FE modelling techniques. Documentation from the beginning is key and access to analysis routines is important for comparing data on an even footing. Graphic ©Seth Gilchrist, 2013.



Figure 6.2: In this image, taken from a video of the setup procedure, the Optotrak digitizing wand is being used to digitize the location of the pivot at the distal end of the femoral shaft. Image ©Seth Gilchrist, 2013.

in a text file. Approximately 30 points were taken on the femoral head, 30 on the femoral neck, 30 on the trochanter, 20 on the shaft, 20 on interface between each PMMA cup and the bone, 4 on the bottom platen, 6 on each side of the pivot, 8 on the impact hammer and finally 2 on the strain gauge. The organization of the points was recorded in a log sheet (given in full in §A.5) which was provided to the FE modellers for reference.

6.3 Data portability and transparency

Data from the experiment are used to drive the FE model and validate the output. The models of the current tests, created by Oscar Ariza in Dr. Ben Helgason’s facilities, used a velocity derived from the displacement data as the affecting BC. Other models may use a force BC, but in any case, the modellers need to have access to the data and any filtering or post processing that was conducted.

The sources of the data, while important to a mechanical modeller, are of less importance to the computational community. Instead, computational modellers are more concerned with consistency in data preparation and organization, allowing them to automate the process of

extracting needed BCs. For example, in the tests conducted in this thesis, displacement in the materials testing machine was recorded by a data acquisition system (DAQ) from a voltage signal emitted by the materials testing machine, but in the fall simulator displacement was extracted from optical measurements of the trochanter and impact hammer. Computational modellers would like the data to be accessed in a similar way, regardless of the fact that it was collected using different techniques.

Post processing of the data is important to both computational and mechanical modellers. FE modellers obtain force and displacement as outputs of their models, and process those data into stiffnesses or energies which should be directly comparable to the experiment. This requires knowledge of how the experiment was conducted and how the data was filtered and/or interpolated.

To meet these needs, I developed nine MatLab (2010b, The Mathworks, Natick, MA) classes that contained all raw data, processed data, processing, filtering and analysis functions as members and methods. This code was fully documented and released under the GNU General Public License, and is included as an appendix to this thesis (§E.1).

A major advantage to using classes such as those detailed in §E.1 is data portability. Once the “experiment” class has been given all the information and all the data has been loaded into the respective subclasses, the class object can be saved as a Matlab MAT (*mat*) file and sent to any researcher with the source code for analysis. The *mat* file contains all the data listed in the previous paragraph and there will be no ambiguity as to how the analysis was performed.

I recommended that data storage and processing be carried out in this way when collaborating with other research groups. It allows trouble shooting by any researcher familiar with MatLab, removes the need for complex directory systems to contain raw, filtered and processed data and incorporates a built in record of the settings used during processing.

6.4 Summary

Validation of FE model data using experimental data is critical for ongoing development of computational techniques. This type of collaboration requires a significant amount of planning and effort on both sides of the problem. Experimenters must be amenable to the collection of additional data, transparent with data analysis routines and make efforts to store their data in such a way that it can be accessed using automated, batch processes. Finally, modellers from both teams should meet regularly to critically evaluate both mechanical and FE model results.

Chapter 7

Discussion and conclusions

7.1 Discussion

Hip fracture research, in the larger sense, has one overall goal: to prevent fractures. There are a number of different methods for attaining this goal, from life-style interventions to pharmacology, but before any of these can be tried, an individual's risk of fracture must be estimated. This is where current knowledge is limited. The majority of hip fractures occur in individuals who would not have been identified using osteoporosis state, which is the current clinical standard for beginning treatment [220]. On top of that, there is considerable overlap in BMDs of those who suffer fracture and those who do not [80]. From these data, it appears that the current understanding of hip fracture risk based primarily on osteoporosis state and BMD does not give sufficient sensitivity to help clinicians reduce fracture rates below their current levels.

One way to increase the screening sensitivity might be through improved identification of physical parameters of the proximal femur that predispose a person to fracture. In the past, gross geometric properties of the proximal femur have been proposed as supplemental data to osteoporosis state to increase the rates of identification. In the more recent past, the FRAX[®] score was developed which included medical and family history that has been linked to hip fracture through epidemiological studies. While gross anatomy of the proximal femur showed promise [67], it has never been generally adopted into clinical use, and while there

are considerable advantages to the use of the FRAX® score (namely, pre-screening without using expensive and potentially dangerous DXA scanning [107]), the sensitivity of FRAX® is only slightly better than BMD and age alone [231], and the specificity of FRAX® is highly dependent on the selection of risk bounds [131].

The fact that BMD accounts for only a portion of those that suffer hip fractures leads us to believe that there may be other important features of the bone which could be used by clinicians to screen for those at risk of a fracture. A number of studies have indicated that the configuration of the bone itself could be important in determining hip fracture risk. Cortical bone has been seen to become thinner and trabecularized with age, which could result in a decrease in strength [21, 49, 154], and cancellous bone has been seen to have a different morphology in fracture patients than in controls [21, 149]. While these data indicate that changes on a bone material level may in part be responsible for fracture susceptibility, the hip fracture research community has not been able to definitely identify what features or configurations are culpable for this change in fracture risk.

Hip fracture modelling, both experimental and computational, has been carried out (or validated) using proximal femurs loaded in materials testing machines (see Table 1.6 on page 39 for experimental references). These machines apply forces by applying a defined velocity at the greater trochanter of the femur. The selection of the velocity has been shown to be important [46, 242], but the actual value that should be used on a specimen specific basis has never been determined. It is possible that the use of a prescribed displacement rate introduces artefacts that may influence how fractures are initiated, how they propagate, and what features influence them. If this is indeed the case, the previous data must be evaluated with this in mind and conclusions should be limited to those that are appropriate for the boundary conditions used. This work was intended to address this final point – what is the effect of a constant velocity at the greater trochanter and what parameters are influenced by this boundary condition? Knowledge of which parameters are valid using this boundary condition and what parameters require more biofidelic testing will allow researchers to design and develop experiments and

validations to address the specific questions they have. Additionally, previous research can be reinterpreted and applied in a way that is consistent with the validity of its output variables.

The objective of the research, outlined in §2, was to determine if current test methods capture the force-displacement, strain, failure and fracture behaviours of the proximal femur in a fall to the side. This objective was met by comparing and contrasting three aspects of the proximal femur when the fall was modelled using either constant displacement rate tests or impact fall simulation utilizing a lumped parameter model of the human body. The aspects that were examined were the (i) mechanical response as characterized by force and displacement and their integral (energy) and derivative (stiffness) measures; (ii) failure response as characterized by maximum force, initial cortical failure location and final fracture pattern; and (iii) surface strain. Comparisons showed that although the quasi-static, constant displacement rate and impact fall simulations methods were equivalent in many cases, there were also differences for some specific outcome variables.

Quasi-static, constant displacement rate testing used in many previous fracture protocols [7, 16, 23, 47, 63, 94, 112, 137, 138] was hypothesized to be a limitation of these experiments. While there is evidence that increased displacement rate significantly changes the response of the bone [46, 242], to our knowledge no research has been conducted that confirmed if this change is indeed representative of the situation *in-vivo*, or if it is an artefact of the increased (but still constant) displacement rate.

The roots of this uncertainty are in the complexity of bone as a material, and the complex arrangement of the bone in structures like the proximal femur. It is known that bone is viscoelastic [37, 52, 134, 157] and there is evidence that the strength of the viscoelastic response is influenced by the presence of bone marrow [37], however, this relationship is not well understood and anticipating its influence in mechanical testing is difficult. On top of this, bone properties are anisotropic and inhomogeneous [109], with values and principal directions that vary continuously within the bone structure [4, 167]. Knowing how the specific arrangement of the bone in a given specimen will be influenced by a given boundary condition is currently out-

side the scope of our understanding. We believe that the best way to address these challenges is to allow the proximal femur freedom to respond to an inertially driven system, similar to a fall. This allows changes in bone material properties and configurations of bone in the specimen to influence the applied loading profile throughout the failure process. The bone then determines the force and displacement time history during a test, in a recursive way.

In order to grant these freedoms and answer the research questions in this thesis, we designed an inertially driven impact fall simulator that utilized a lumped parameter model of the human body to reproduce the loading in a fall to the side. This work was detailed in Chapter 3. The advantage of a fall simulator was that it allowed the bone to respond as a compliant, potentially viscous member in a spring mass system and influence the overall behaviour of the system, in the aforementioned recursive way. This technique removes artificial loading constraints that may influence the behaviour of the bone leading up to and at fracture. The method was limited in its ability to model a specimen specific fall because the parameters of the lumped parameter model were fixed, i.e., the stiffness of the spring, mass of the body and pelvis, and thickness of the tissue over the greater trochanter could not be modified based on knowledge of the donor's age, BMD or body habitus. This limitation could be addressed in future experiments, but the background knowledge required to make informed changes to these parameters currently does not exist. Additionally, increasing the biofidelity of an experiment will not always increase the power of the experiment to be able to address specific questions. While a realistic test is desired, if too many variables are adjusted between experiments, the statistical burden of proof can become large and make it difficult to obtain the power required to test a hypothesis in a meaningful way. It is therefore the responsibility of the researcher to identify the important parameters which drive the outcome under investigation and find realistic values for the other parameters. We believe that the advantage of reduced constraint of the loading rate on the response of the proximal femur during testing outweighs the potential limitation of a single body being simulated for all specimens.

The question relating to the mechanical response in fall simulator testing vs. materials test-

ing machine testing was addressed using two tests and is contained in Chapter 4 in which one set of femurs was tested in two ways. This experiment was used to determine if the mechanical behaviours of the proximal femurs were affected by the change in loading technique and hypothesized that there would be changes in stiffness, strain at a point and energy to the failure. We found that there was no difference in the behaviours of the bones in terms of any of our mechanical metrics. There were no statistically significant differences in the stiffness, energy or strain on the anterior superior femoral neck if the proximal femur is loaded using quasi-static, constant displacement rate or impact fall simulation. These data indicate that if the outcomes of experimental testing or FE validation are sub-failure force-displacement response, then mechanical testing in a materials testing machine is adequate.

We also compared the results of this test to those of Courtney et al. [46], who had identified a strong viscoelastic response when displacement rate at the greater trochanter was changed from 2 to 100 mm/s. Our results indicate that the viscoelastic response in the fall simulator was much lower than that observed by Courtney et al. [46], even though our average displacement rate was nearly the same as the 100 mm/s rate used by Courtney et al. [46]. This indicates that artificially imposing a high displacement rate at the greater trochanter affects outcomes and may not represent the true behaviour of a specimen in a fall to the side.

The question of whether failure initiation and fracture patterns would change due to loading technique was discussed in Chapter 5, and addressed by hypothesizing that there would be different initial failure locations and final fracture patterns in the constant displacement rate and impact fall simulation testing. The initial failure locations were identified using high-speed video of the tests and final fracture types were classified using a clinical classification system by an orthopaedic surgeon. The initial failure locations were not significantly different between the two tests, but the final fracture types were. Additionally, in the fall simulation tests, the initial failure locations were not predictive of the final fracture types, whereas in the quasi-static tests they were.

These data, combined with the data in the previous paragraphs indicating that the mechan-

ical behaviours are the same for sub-failure loading, indicate that the sub-failure response of the proximal femur is relatively insensitive to the type of loading used. However, once failure begins and there is a re-definition of the primary load path, the type of testing used becomes important. The fact that the initial failure observed using high speed video occurred at nearly the same instant as the force-displacement identified yield indicates that the primary load path, regardless of test method, in the sub-failure regime, is through the cortex of the bone. Once the failure begins, the primary load path changes and observations of the exterior of the bone are no longer sufficient to determine its course.

The final tests conducted were to determine if the sub-failure strains on the surface of the proximal femur were different between quasi-static, constant displacement rate and impact fall simulation tests and is detailed in Chapter 5. The results of these tests indicate that the surface strains were different in any given specimen, but taken as a whole (i.e., pooling all the data) showed that there was no systematic difference between the two test methods. The majority of the specimens showed a shift in strain levels, either to higher or lower strain, when tested in fall simulation vs. quasi-static testing. In these cases, the patterns were similar, but there was a “DC” offset in magnitude. Some specimens showed a redistribution of strain on the bone cortex, however the magnitude of the average strain was not significantly higher in one test or the other.

These data seem counter to the *no difference in sub-failure mechanical behaviour* result that was identified in Chapter 4. The reason for this may be due to lack of statistical power to determine differences in loading behaviours in the previous tests. Those tests had only five data points per specimen, in contrast to the DIC strain measurements which obtains full strain fields containing hundreds of data points per specimen. However, it is important to note that the differences observed in the DIC results were statistically significant, but may not be physically significant. The average difference in strain was only about 7% of the strain value, a result that may be due to variations in the experimental setup. Regardless, the measured differences are real, and based on our data indicating a diverging behaviour after initial failure,

the affect on strain would likely increase at higher loads (remember that the response is being compared at 50% of the aBMD predicted failure load). It is therefore our recommendation that if changes in surface strain in a specimen-specific manner is the desired outcome, fall simulation be employed. However, if the outcome of a series of tests is a generalized behaviour of strain that will be averaged over a cohort, then either quasi-static, constant displacement rate or fall simulation may be employed.

It is important to note that two specimens that were tested in the fall simulator failed but did not fracture. Additionally, the failures occurred in the same place and had the same morphologies: cracks running along the superior femoral neck, originating in the greater trochanter and terminating in the lateral neck. Due to their location and size it is unlikely that these failures would have been detected using clinical tools like anterior-posterior x-rays or CT scanning. It is not impossible that failures like these are common after a fall, and go undetected in the population. Given that our results suggest that cancellous bone mechanics may influence fracture behaviour in a fall after the cortex has failed, understanding the loading paths in these specimens might help inform researchers what made them resistant to fracture.

FE analysis, which is on-going in our FE collaborator's lab at ETH Zürich, is one technique that has specific advantages for detailed exploration of the behaviour of a specific specimen. The non-destructive nature and ability to examine many different outcomes as a function of time can illuminate events that are difficult to observe experimentally leading up to failure. That said, the secret to these specimens may lie in the behaviour of their cancellous bone which, as I have pointed out, has not been validated in any FE studies. The work on digital volume correlation (DVC) presented in Appendix C of this thesis may therefore be critical to understanding how specimens that are resilient to fracture behave. It appears from the quasi-static and fall simulation work that cancellous bone behaviour is critically important after initial cortical failure, and therefore accurate modelling is crucial for understanding the behaviour of these specimens. The DVC technique and tools developed herein could be used to validate quasi-static strain behaviour of cancellous bone and allow FE researchers to advance their

Table 7.1: Required tests for given outcome variables in mechanical testing and FE validation

Outcome Measure	Testing Options
Sub-failure mechanical behaviour	Quasi-static, constant displacement rate or impact fall simulation
Initial fracture location	Quasi-static, constant displacement rate or impact fall simulation
Final fracture pattern	Impact fall simulation
Specimen specific surface strain	Impact fall simulation
Cohort surface strain	Quasi-static, constant displacement rate or impact fall simulation

understand the role of cancellous bone. While the nature of the DVC technique requires that tests be conducted in a quasi-static manner, the validation is no less important since initial failure location (which may be the critical feature of these specimens) can be predicted in quasi-static tests.

Overall, the results of these tests indicate that different tests are appropriate for different outcomes in proximal femur fracture research (Table 7.1). In general, sub-failure testing that does not rely on surface strain profiles can be conducted using either quasi-static, constant displacement rates or fall simulation. If post failure mechanics (such as fracture mechanics) or detailed surface strain maps are to be considered, then fall simulation is the best choice. The implication of the changing post failure mechanics is that dynamics that cannot be observed using high-speed video of tests are important, in which case it is likely that observation of the cancellous bone mechanics and changing loading paths may be informative.

7.2 Conclusion

This research addresses one of the most basic questions in hip fracture testing: how biofidelic do our tests need to be? It was conducted using a first-of-its-kind impact fall simulator and

pioneered using DIC to assess strain on femoral bone. The experiments resulted in a framework for identifying what tests are appropriate for a given outcome variable and showed that full field strain measurement on the surface of the bone can be used to detect small differences in response to a given loading. Specimens which did not fracture under an impact loading were identified, which could lead to new insights into the mechanics of resilient bones, and the source of that resilience.

We found that sub-failure mechanical behaviours of the proximal femur are insensitive to the vastly different methods we used to apply loads representing a fall to the side. Additionally, surface strains of individual specimens were different, but overall strain behaviour of all the specimens were consistent between the two testing methods. Finally, initial failure locations were not different between the two testing methods, but final fracture patterns were.

These results indicated that there is need for careful consideration of the outcomes of a given test before selecting a test method. Additionally should a biofidelic, impact test be used, the tools developed herein can be applied to detect small differences in behaviour as well as identify those that are resistant to hip fracture. This knowledge will allow fracture researchers to make better decisions in design, interpretation and application of tests to identify those at risk of fracture and hopefully prevent the onset of the devastating cascade that can follow.

7.3 Future work

As with many studies, the work presented in this thesis has illuminated new holes in the knowledge, and helped determine which previously known deficiencies require further investigation. Recommendations for future work that will, in my opinion, assist researchers in their efforts to understand proximal femur fracture are given below.

Increase tissue thickness to identify resilient specimens. In the current study, two specimens were impacted and did not fracture (Chapter 4). In real world falls, only about 5% of falls result in fracture and one of the likely reasons for the difference between our fracture rates and those in the real world is that the current test used a lower bound on soft tissue thickness

over the greater trochanter. In the proposed study, the soft tissue thickness would be increased to one standard deviation above the average for fracture cases. The results from previous researchers have indicated that this increase in soft tissue thickness would likely result in a lower peak force by about 1700 N [170, 200] and generate more differentiation between strong and weak specimens. The identification of strong and weak specimens could allow for analysis of cancellous and cortical bone parameters which may indicate why some bones are more resilient to fracture than others.

Regional high resolution, peripheral, quantitative computed tomography (HR-pQCT) examination of fracture initiation location and non-fracture specimens. The location of fracture initiation in the proximal femur is neither regular, nor 100% predictive of the final fracture type (Chapter 5). In this study, specimens that failed would have regional analyses of the cortical and cancellous bone performed on HR-pQCT scans that were acquired before fracture testing. Additionally, bones that did not fracture in fall simulation testing would have critical regions like the superio-lateral femoral neck and anterior intertrochanteric region evaluated. HR-pQCT outcomes would be compared between the fracture and non-fracture groups to identify those that are predictive of initial fracture location.

Comparison of fall simulation generated fractures to clinical fractures. After fracture in the current fall simulator, the drop tower gantry continues moving, compressing the remaining bone fragments. This creates many secondary and tertiary fracture lines and makes post fracture x-ray impossible (Chapter 5). In this study, a method, such as stop-blocks, would be fitted in to the fall simulator to prevent further compression of the specimen after a certain compression level, likely in the range of 10 mm, had been attained. Specimens from this group would be imaged using planar x-ray and read by an orthopaedic surgeon or radiologist for comparison to clinical fracture groups.

Inclusion of an acetabular mass to create compaction injuries. In the current fall simulator, the head of the femur has no constraint, and is often separated fully from the femoral neck after fracture (Chapters 3 and 4). In a real fall, the head is contained in the acetabulum which is

immobilized by the mass of the body. Inertial constraint of the femoral head, as would be the case in a real fall to the side, may increase the number of intrecapsular fractures and impactions of the femoral neck, increasing the biofidelity of the apparatus. Continued improvement of the apparatus biofidelity could lead to more clinically relevant fracture types and potentially more relevant fracture locations and patters.

Bibliography

- [1] A. Aamodt, T. Terjesen, J. Eine, and K. A. Kvistad. Femoral anteversion measured by ultrasound and CT: a comparative study. *Skeletal Radiology*, 24(2):105–109, Feb. 1995. ISSN 0364-2348. URL <http://www.ncbi.nlm.nih.gov/pubmed/7747174>. [Accessed on 2010-10-24]. → pages
- [2] B. Abrahamsen, T. van Staa, R. Ariely, M. Olson, and C. Cooper. Excess mortality following hip fracture: a systematic epidemiological review. *Osteoporosis International*, 20(10):1633–1650, Oct. 2009. ISSN 1433-2965. doi:10.1007/s00198-009-0920-3. URL <http://www.ncbi.nlm.nih.gov/pubmed/19421703>. [Accessed on 2010-12-14]. → pages
- [3] M. E. Armstrong, E. A. Spencer, B. J. Cairns, E. Banks, K. Pirie, J. Green, F. L. Wright, G. K. Reeves, V. Beral, and f. t. M. W. S. Collaborators. Body mass index and physical activity in relation to the incidence of hip fracture in postmenopausal women. *Journal of Bone and Mineral Research*, 26(6):1330–1338, 2011. ISSN 1523-4681. doi:10.1002/jbmr.315. [Accessed on 2013-02-18]. → pages
- [4] M.-G. Ascenzi, N. Hetzer, A. Lomovtsev, R. Rude, A. Nattiv, and A. Favia. Variation of trabecular architecture in proximal femur of postmenopausal women. *Journal of Biomechanics*, 44(2):248–256, Jan. 2011. ISSN 0021-9290. doi:10.1016/j.jbiomech.2010.10.017. URL <http://www.sciencedirect.com/science/article/B6T82-51D26DP-4/2/1524104cf0f9646b51a822c15c95b296>. [Accessed on 2011-02-04]. → pages
- [5] L. V. Avioli and S. M. Krane. *Metabolic bone disease and clinically related disorders*. Academic Press, San Diego, 3rd ed edition, 1998. ISBN 0120687003. → pages
- [6] I. Babuska and J. T. Oden. Verification and validation in computational engineering and science: basic concepts. *Computer Methods in Applied Mechanics and Engineering*, 193(36–38):4057–4066, Sept. 2004. doi:10.1016/j.cma.2004.03.002. URL <http://www.sciencedirect.com/science/article/pii/S0045782504001781>. → pages
- [7] S. Backman. The proximal end of the femur: investigations with special reference to the etiology of femoral neck fractures; anatomical studies; roentgen projections; theoretical stress calculations; experimental production of fractures. *Acta Radiologica Supplementum*, 146:1–166, 1957. ISSN 0365-5954. URL <http://www.ncbi.nlm.nih.gov/pubmed/13435002>. [Accessed on 2010-11-03]. → pages

- [8] Bank of Canada. Inflation calculator - bank of canada, 2013. URL <http://www.bankofcanada.ca/rates/related/inflation-calculator/>. [Accessed on 2013-07-25]. → pages
- [9] N. Barnes. Publish your computer code: it is good enough. *Nature*, 467(7317): 753–753, Oct. 2010. ISSN 0028-0836. doi:10.1038/467753a. URL <http://www.nature.com/doifinder/10.1038/467753a>. [Accessed on 2011-05-03]. → pages
- [10] B. K. Bay. Texture correlation: a method for the measurement of detailed strain distributions within trabecular bone. *Journal of Orthopaedic Research*, 13(2):258–267, Mar. 1995. ISSN 0736-0266. doi:10.1002/jor.1100130214. URL <http://www.ncbi.nlm.nih.gov/pubmed/7722763>. [Accessed on 2010-10-28]. → pages
- [11] B. K. Bay. Methods and applications of digital volume correlation. *The Journal of Strain Analysis for Engineering Design*, 43(8):745–760, Aug. 2008. ISSN 0309-3247. doi:10.1243/03093247JSA436. URL <http://journals.pepublishing.com/content/r12586p6v04j1100/>. [Accessed on 2010-11-03]. → pages
- [12] B. K. Bay, T. S. Smith, D. P. Fyhrie, and M. Saad. Digital volume correlation: Three-dimensional strain mapping using x-ray tomography. *Experimental Mechanics*, 39(3):217–226, Sept. 1999. ISSN 0014-4851. doi:10.1007/BF02323555. URL <http://www.springerlink.com/content/2j3454k102314700/>. [Accessed on 2010-11-03]. → pages
- [13] B. K. Bay, S. A. Yerby, R. F. McLain, and E. Toh. Measurement of strain distributions within vertebral body sections by texture correlation. *Spine*, 24(1):10–7, 1999. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=9921585. [Accessed on 2014-01-11]. → pages
- [14] H. H. Bayraktar, E. F. Morgan, G. L. Niebur, G. E. Morris, E. K. Wong, and T. M. Keaveny. Comparison of the elastic and yield properties of human femoral trabecular and cortical bone tissue. *Journal of Biomechanics*, 37(1):27–35, Jan. 2004. ISSN 0021-9290. doi:10.1016/S0021-9290(03)00257-4. URL <http://www.sciencedirect.com/science/article/B6T82-494HK7J-8/2/0ffa5b5cee1703da936060e713f6af42>. [Accessed on 2010-12-13]. → pages
- [15] D. P. Beason, G. J. Dakin, R. R. Lopez, J. E. Alonso, F. A. Bandak, and A. W. Eberhardt. Bone mineral density correlates with fracture load in experimental side impacts of the pelvis. *Journal of Biomechanics*, 36(2):219–27, 2003. URL <http://www.ncbi.nlm.nih.gov/pubmed/12547359>. [Accessed on 2014-01-11]. → pages
- [16] J. Beckmann, S. J. Ferguson, M. Gebauer, C. Luering, B. Gasser, and P. Heini. Femoroplasty–augmentation of the proximal femur with a composite bone cement–feasibility, biomechanical properties and osteosynthesis potential. *Medical Engineering and Physics*, 29(7):755–764, Sept. 2007. ISSN 1350-4533.

- doi:10.1016/j.medengphy.2006.08.006. URL
<http://www.ncbi.nlm.nih.gov/pubmed/17023189>. [Accessed on 2011-12-02]. → pages
- [17] K. L. Bell, N. Loveridge, J. Power, N. Garrahan, M. Stanton, M. Lunt, B. F. Meggitt, and J. Reeve. Structure of the femoral neck in hip fracture: Cortical bone loss in the inferoanterior to superoposterior axis. *Journal of Bone and Mineral Research*, 14(1):111–119, 1999. ISSN 1523-4681. doi:10.1359/jbmr.1999.14.1.111. URL
<http://dx.doi.org/10.1359/jbmr.1999.14.1.111>. [Accessed on 2014-01-11]. → pages
- [18] A. Benoit, S. Guérard, B. Gillet, G. Guillot, F. Hild, D. Mitton, J.-N. Périé, and S. Roux. 3D analysis from micro-MRI during in situ compression on cancellous bone. *Journal of Biomechanics*, 42(14):2381–2386, Oct. 2009. ISSN 00219290.
doi:10.1016/j.jbiomech.2009.06.034. [Accessed on 2010-11-03]. → pages
- [19] M. Bhandari, P. J. Devereaux, M. F. Swiontkowski, P. TornettaIII, W. Obremskey, K. J. Koval, S. Nork, S. Sprague, E. H. Schemitsch, and G. H. Guyatt. Internal fixation compared with arthroplasty for displaced fractures of the femoral neck a meta-analysis. *The Journal of Bone and Joint Surgery*, 85(9):1673–1681, 2003. URL
<http://jbjs.org/article.aspx?articleID=26075>. [Accessed on 2013-07-25]. → pages
- [20] P. Bing, X. Hui-min, X. Bo-qin, and D. Fu-long. Performance of sub-pixel registration algorithms in digital image correlation. *Measurement Science and Technology*, 17(6):1615–1621, June 2006. ISSN 0957-0233. doi:10.1088/0957-0233/17/6/045. URL
<http://iopscience.iop.org/0957-0233/17/6/045>. [Accessed on 2010-11-03]. → pages
- [21] H. Blain, P. Chavassieux, N. Portero-Muzy, F. Bonnel, F. Canovas, M. Chammas, P. Maury, and P. D. Delmas. Cortical and trabecular bone distribution in the femoral neck in osteoporosis and osteoarthritis. *Bone*, 43(5):862–868, 2008. → pages
- [22] J. M. Bland and D. Altman. Statistical methods for assessing agreement between two methods of clinical measurement. *The Lancet*, 327(8476):307–310, Feb. 1986. ISSN 0140-6736. doi:10.1016/S0140-6736(86)90837-8. URL
<http://www.sciencedirect.com/science/article/pii/S0140673686908378>. [Accessed on 2013-01-23]. → pages
- [23] H. Boehm, A. Horng, M. Notohamiprodjo, F. Eckstein, D. Burklein, A. Panteleon, J. Lutz, and M. Reiser. Prediction of the fracture load of whole proximal femur specimens by topological analysis of the mineral distribution in DXA-scan images. *Bone*, 43(5):826–831, Nov. 2008. ISSN 87563282. doi:10.1016/j.bone.2008.07.244.
[Accessed on 2011-01-24]. → pages
- [24] H. Bolotin, H. Sievänen, and J. Grashuis. Patient-specific DXA bone mineral density inaccuracies: Quantitative effects of nonuniform extraosseous fat distributions. *Journal of Bone and Mineral Research*, 18(6):1020–1027, 2003. ISSN 1523-4681.
doi:10.1359/jbmr.2003.18.6.1020. [Accessed on 2013-12-16]. → pages
- [25] A.-M. Borissova, R. Rashkov, M. Boyanov, A. Shinkov, P. Popivanov, N. Temelkova, J. Vlahov, and M. Gavrilova. Femoral neck bone mineral density and 10-year absolute

- fracture risk in a national representative sample of bulgarian women aged 50 years and older. *Archives of Osteoporosis*, 6(1-2):189–195, Dec. 2011. ISSN 1862-3522, 1862-3514. doi:10.1007/s11657-011-0064-x. URL <http://link.springer.com/article/10.1007/s11657-011-0064-x>. [Accessed on 2013-01-30]. → pages
- [26] M. Bornert, N. Lenoir, P. Bésuelle, Y. Pannier, S. Hall, G. Viggiani, and J. Desrues. Discrete and continuum analysis of localised deformation in sand using x-ray μ CT and volumetric digital image correlation. *Géotechnique*, 60(5):315–322, Jan. 2010. ISSN 0016-8505, 1751-7656. doi:10.1680/geot.2010.60.5.315. URL <http://www.icevirtuallibrary.com/content/article/10.1680/geot.2010.60.5.315>. [Accessed on 2013-08-22]. → pages
- [27] M. L. Bouxsein. Bone quality: where do we go from here? *Osteoporosis International*, 14(5S):118–127, Sept. 2003. ISSN 0937-941X, 1433-2965. doi:10.1007/s00198-003-1489-x. URL <http://www.springerlink.com/content/ltrcvq9uuqba1j15/>. [Accessed on 2011-10-28]. → pages
- [28] M. L. Bouxsein, A. C. Courtney, and W. C. Hayes. Ultrasound and densitometry of the calcaneus correlate with the failure loads of cadaveric femurs. *Calcified Tissue International*, 56(2):99–103, 1995. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=7736330. [Accessed on 2014-01-11]. → pages
- [29] M. L. Bouxsein, P. Szulc, F. Munoz, E. Thrall, E. Sornay-Rendu, and P. D. Delmas. Contribution of trochanteric soft tissues to fall force estimates, the factor of risk, and prediction of hip fracture risk. *Journal of Bone and Mineral Research*, 22(6):825–831, 2007. ISSN 1523-4681. doi:10.1359/jbmr.070309. [Accessed on 2013-02-18]. → pages
- [30] R. S. Braithwaite, N. F. Col, and J. B. Wong. Estimating hip fracture morbidity, mortality and costs. *Journal of the American Geriatrics Society*, 51(3):364–370, Mar. 2003. ISSN 00028614. doi:10.1046/j.1532-5415.2003.51110.x. [Accessed on 2011-02-02]. → pages
- [31] T. J. Bray. Femoral neck fracture fixation: clinical decision making. *Clinical Orthopaedics and Related Research*, 339:20–31, 1997. URL http://journals.lww.com/corr/Abstract/1997/06000/Femoral_Neck_Fracture_Fixation__Clinical_Decision.4.aspx. [Accessed on 2013-07-25]. → pages
- [32] B. D. Browner. *Skeletal Trauma*, volume 2. W.B. Saunders, Philadelphia, Pa.; London, 3rd edition, 2002. ISBN 0721691757 9780721691756 0721694810 9780721694818. → pages
- [33] R. Bryan, P. B. Nair, and M. Taylor. Use of a statistical model of the whole femur in a large scale, multi-model study of femoral neck fracture risk. *Journal of Biomechanics*,

- 42(13):2171–2176, Sept. 2009. ISSN 0021-9290. doi:10.1016/j.jbiomech.2009.05.038. URL <http://www.sciencedirect.com/science/article/pii/S0021929009003170>. [Accessed on 2013-08-08]. → pages
- [34] Bureau of Labor Statistics. CPI inflation calculator, 2013. URL <http://data.bls.gov/cgi-bin/cpicalc.pl>. [Accessed on 2013-07-25]. → pages
- [35] H. Burger, P. Van Daele, D. Algra, F. Van den Ouwehand, D. Grobbee, A. Hofman, C. Van Kuijk, H. Schutte, J. Birkenhager, and H. Pols. The association between age and bone mineral density in men and women aged 55 years and over: The rotterdam study. *Bone and Mineral*, 25(1):1–13, 1994. → pages
- [36] D. R. Carter and W. C. Hayes. Bone compressive strength: the influence of density and strain rate. *Science*, 194(4270):1174–1176, Dec. 1976. ISSN 0036-8075. URL <http://www.ncbi.nlm.nih.gov/pubmed/996549>. [Accessed on 2011-10-24]. → pages
- [37] D. R. Carter and W. C. Hayes. The compressive behavior of bone as a two-phase porous structure. *The Journal of Bone and Joint Surgery American*, 59(7):954–62, 1977. → pages
- [38] D. R. Carter, G. H. Schwab, and D. M. Spengler. Tensile fracture of cancellous bone. *Acta Orthopaedica Scandinavica*, 51(5):733–741, Oct. 1980. ISSN 0001-6470. PMID: 7468167. → pages
- [39] J. A. Cauley, L.-Y. Lui, H. K. Genant, L. Salamone, W. Browner, H. A. Fink, P. Cohen, T. Hillier, D. C. Bauer, and S. R. Cummings. Risk factors for severity and type of the hip fracture. *Journal of Bone and Mineral Research*, 24(5):943–955, May 2009. ISSN 1523-4681. doi:10.1359/jbmr.081246. URL <http://www.ncbi.nlm.nih.gov/pubmed/19113930>. [Accessed on 2011-09-01]. → pages
- [40] J. M. Cavanaugh, T. J. Walilko, A. Malhotra, Y. Zhu, and A. I. King. Biomechanical response and injury tolerance of the pelvis in twelve sled side impacts. In *SAE Transactions*, Warrendale, PA, Nov. 1990. SAE International. URL <http://papers.sae.org/902305>. [Accessed on 2011-07-12]. → pages
- [41] X. G. Cheng, G. Lowet, S. Boonen, P. H. F. Nicholson, P. Brys, J. Nijs, and J. Dequeker. Assessment of the strength of proximal femur in vitro: Relationship to femoral bone mineral density and femoral geometry. *Bone*, 20(3):213–218, Mar. 1997. ISSN 8756-3282. doi:10.1006/jbm.1997.0038. URL <http://www.sciencedirect.com/science/article/pii/S8756328296003833>. [Accessed on 2011-09-01]. → pages
- [42] X. G. Cheng, G. Lowet, S. Boonen, P. H. Nicholson, G. Van der Perre, and J. Dequeker. Prediction of vertebral and femoral strength in vitro by bone mineral density measured at different skeletal sites. *Journal of Bone and Mineral Research*, 13(9):1439–1443, Sept. 1998. ISSN 0884-0431. doi:10.1359/jbmr.1998.13.9.1439. URL <http://www.ncbi.nlm.nih.gov/pubmed/9738516>. [Accessed on 2011-09-01]. → pages

- [43] W. J. Choi, J. A. Hoffer, and S. N. Robinovitch. Effect of hip protectors, falling angle and body mass index on pressure distribution over the hip during simulated falls. *Clinical Biomechanics*, 25(1):63–69, Jan. 2010. ISSN 1879-1271. doi:10.1016/j.clinbiomech.2009.08.009. URL <http://www.ncbi.nlm.nih.gov/pubmed/19766363>. [Accessed on 2010-10-27]. → pages
- [44] ClinicalTrials.gov. Flooring for injury prevention trial - full text view - ClinicalTrials.gov, 2012. URL <http://clinicaltrials.gov/ct2/show/NCT01618786>. [Accessed on 2013-08-08]. → pages
- [45] C. Cooper. The crippling consequences of fractures and their impact on quality of life. *The American Journal of Medicine*, 103(2A):12S–17S; discussion 17S–19S, Aug. 1997. ISSN 0002-9343. URL <http://www.ncbi.nlm.nih.gov/pubmed/9302893>. [Accessed on 2011-09-01]. → pages
- [46] A. C. Courtney, E. F. Wachtel, E. R. Myers, and W. C. Hayes. Effects of loading rate on strength of the proximal femur. *Calcified Tissue International*, 55(1):53–58, July 1994. ISSN 0171-967X. doi:10.1007/BF00310169. URL <http://www.springerlink.com/content/rgg88811473015r8/>. [Accessed on 2010-10-27]. → pages
- [47] A. C. Courtney, E. F. Wachtel, E. R. Myers, and W. C. Hayes. Age-related reductions in the strength of the femur tested in a fall-loading configuration. *The Journal of Bone and Joint Surgery American*, 77(3):387–395, Mar. 1995. ISSN 0021-9355. URL <http://www.ncbi.nlm.nih.gov/pubmed/7890787>. [Accessed on 2010-10-25]. → pages
- [48] A. C. Courtney, W. C. Hayes, and L. J. Gibson. Age-related differences in post-yield damage in human cortical bone. experiment and model. *Journal of Biomechanics*, 29 (11):1463–1471, Nov. 1996. ISSN 0021-9290. doi:10.1016/0021-9290(96)84542-8. URL <http://www.sciencedirect.com/science/article/B6T82-3W3NCG2-8/2-58557393958d1a327c875cfbb34067ed>. [Accessed on 2011-03-17]. → pages
- [49] N. Crabtree, N. Loveridge, M. Parker, N. Rushton, J. Power, K. L. Bell, T. J. Beck, and J. Reeve. Intracapsular hip fracture and the region-specific loss of cortical bone: Analysis by peripheral quantitative computed tomography. *Journal of Bone and Mineral Research*, 16(7):1318–1328, July 2001. ISSN 08840431. doi:10.1359/jbmr.2001.16.7.1318. [Accessed on 2011-02-03]. → pages
- [50] L. Cristofolini, B. P. McNamara, A. Freddi, and M. Viceconti. In vitro measured strains in the loaded femur: quantification of experimental error. *The Journal of Strain Analysis for Engineering Design*, 32(3):193–200, Jan. 1997. ISSN 0309-3247. doi:10.1243/0309324971513337. URL <http://journals.pepublishing.com/content/p411n7v6t1701368/>. [Accessed on 2010-11-08]. → pages
- [51] L. Cristofolini, M. Juszczyk, S. Martelli, F. Taddei, and M. Viceconti. In vitro replication of spontaneous fractures of the proximal human femur. *Journal of*

- Biomechanics, 40(13):2837–2845, 2007. ISSN 0021-9290.
doi:10.1016/j.jbiomech.2007.03.015. URL
<http://www.ncbi.nlm.nih.gov/pubmed/17475269>. [Accessed on 2011-06-04]. → pages
- [52] R. D. Crowninshield and M. H. Pope. The response of compact bone in tension at various strain rates. Annals of Biomedical Engineering, 2(2):217–225, 1974. URL
<http://www.springerlink.com/index/m832u7373275m81t.pdf>. [Accessed on 2012-12-06]. → pages
- [53] S. R. Cummings and L. J. Melton. Epidemiology and outcomes of osteoporotic fractures. The Lancet, 359(9319):1761–1767, May 2002. ISSN 0140-6736.
doi:10.1016/S0140-6736(02)08657-9. URL
<http://www.sciencedirect.com/science/article/pii/S0140673602086579>. [Accessed on 2011-07-27]. → pages
- [54] S. R. Cummings, M. C. Nevitt, W. S. Browner, K. Stone, K. M. Fox, K. E. Ensrud, J. Cauley, D. Black, and T. M. Vogt. Risk factors for hip fracture in white women. study of osteoporotic fractures research group. N Engl J Med, 332(12):767–73, 1995. URL
http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=7862179. → pages
- [55] J. Currey. The effects of strain rate, reconstruction and mineral content on some mechanical properties of bovine bone. Journal of Biomechanics, 8(1):81–86, 1975. ISSN 0021-9290. doi:10.1016/0021-9290(75)90046-9. URL
<http://www.sciencedirect.com/science/article/pii/0021929075900469>. [Accessed on 2012-12-05]. → pages
- [56] P. M. de Bakker. Hip Fractures: Understanding the Mechanism and Seeking Prevention Through Prophylactic Augmentation of the Proximal Femur. Master of applied science, University of British Columbia, Vancouver, Canada, July 2006. → pages
- [57] P. M. de Bakker, S. L. Manske, V. Ebacher, T. R. Oxland, P. A. Cripton, and P. Guy. During sideways falls proximal femur fractures initiate in the superolateral cortex: evidence from high-speed video of simulated fractures. Journal of Biomechanics, 42(12):1917–1925, Aug. 2009. ISSN 1873-2380. doi:10.1016/j.jbiomech.2009.05.001. URL
<http://www.ncbi.nlm.nih.gov/pubmed/19524929>. [Accessed on 2010-10-25]. → pages
- [58] S. N. DeLaMora and M. Gilbert. Introduction of intracapsular hip fractures: Anatomy and pathologic features. Clinical Orthopaedics and Related Research, pages 9–16, June 2002. ISSN 0009-921X. URL
<http://ovidsp.ovid.com/ovidweb.cgi?T=JS&CSC=Y&NEWS=N&PAGE=fulltext&D=ovftf&AN=00003086-200206000-00003>. [Accessed on 2013-07-26]. → pages
- [59] L. C. Derikx, R. Vis, T. Meinders, N. Verdonschot, and E. Tanck. Implementation of asymmetric yielding in case-specific finite element models improves the prediction of femoral fractures. Computer Methods in Biomechanics and Biomedical Engineering,

- 14:183–193, Feb. 2011. ISSN 1025-5842, 1476-8259.
doi:10.1080/10255842.2010.542463. URL
<http://www.tandfonline.com/doi/abs/10.1080/10255842.2010.542463>. [Accessed on 2011-10-24]. → pages
- [60] A. S. Dickinson, A. C. Taylor, H. Ozturk, and M. Browne. Experimental validation of a finite element model of the proximal femur using digital image correlation and a composite bone model. *Journal of Biomechanical Engineering*, 133(1):014504, Jan. 2011. doi:10.1115/1.4003129. URL <http://link.aip.org/link/?JBY/133/014504/1>. [Accessed on 2011-03-15]. → pages
- [61] V. E. Dinçel, M. Sengelen, V. Sepici, T. Cavusoglu, and B. Sepici. The association of proximal femur geometry with hip fracture risk. *Clinical Anatomy*, 21(6):575–580, Sept. 2008. ISSN 08973806, 10982353. doi:10.1002/ca.20680. URL
<http://doi.wiley.com/10.1002/ca.20680>. [Accessed on 2013-08-07]. → pages
- [62] L. D. Dorr, Z. Wan, A. Malik, J. Zhu, M. Dastane, and P. Deshmane. A comparison of surgeon estimation and computed tomographic measurement of femoral component anteversion in cementless total hip arthroplasty. *The Journal of Bone and Joint Surgery*, 91(11):2598–2604, Nov. 2009. ISSN 0021-9355. doi:10.2106/JBJS.H.01225. URL
<http://dx.doi.org/10.2106/JBJS.H.01225>. [Accessed on 2013-03-08]. → pages
- [63] F. Eckstein, C. Wunderer, H. Boehm, V. Kuhn, M. Priemel, T. M. Link, and E.-M. Lochmüller. Reproducibility and side differences of mechanical tests for determining the structural strength of the proximal femur. *Journal of Bone and Mineral Research*, 19(3):379–385, Mar. 2004. ISSN 0884-0431. doi:10.1359/JBMR.0301247. URL
<http://www.ncbi.nlm.nih.gov/pubmed/15040825>. [Accessed on 2010-10-25]. → pages
- [64] Edorardo. Anatomical planes and directions, Nov. 2011. URL
http://commons.wikimedia.org/wiki/File:Anatomical_Planes-en.svg. [Accessed on 2014-01-11]. → pages
- [65] S. El-Kaissi, J. A. Pasco, M. J. Henry, S. Panahi, J. G. Nicholson, G. C. Nicholson, and M. A. Kotowicz. Femoral neck geometry and hip fracture risk: the geelong osteoporosis study. *Osteoporosis International*, 16(10):1299–1303, Aug. 2005. ISSN 0937-941X. doi:10.1007/s00198-005-1988-z. URL
<http://www.springerlink.com/content/tt38660011527221/>. [Accessed on 2011-09-01]. → pages
- [66] G. P. Evans, J. C. Behiri, L. C. Vaughan, and W. Bonfield. The response of equine cortical bone to loading at strain rates experienced in vivo by the galloping horse. *Equine Veterinary Journal*, 24(2):125–128, 1992. ISSN 2042-3306.
doi:10.1111/j.2042-3306.1992.tb02796.x. [Accessed on 2012-12-05]. → pages
- [67] K. G. Faulkner, S. R. Cummings, D. Black, L. Palermo, C. Glüer, and H. K. Genant. Simple measurement of femoral geometry predicts hip fracture: The study of osteoporotic fractures. *Journal of Bone and Mineral Research*, 8(10):1211–1217, Oct.

1993. ISSN 1523-4681. doi:10.1002/jbmр.5650081008. [Accessed on 2011-09-19]. → pages
- [68] F. Feldman and S. N. Robinovitch. Reducing hip fracture risk during sideways falls: evidence in young adults of the protective effects of impact to the hands and stepping. *Journal of Biomechanics*, 40(12):2612–2618, 2007. ISSN 0021-9290. doi:10.1016/j.jbiomech.2007.01.019. URL <http://www.ncbi.nlm.nih.gov/pubmed/17395188>. [Accessed on 2010-10-27]. → pages
- [69] T. A. Ferguson, R. Patel, M. Bhandari, and J. M. Matta. Fractures of the acetabulum in patients aged 60 years and older AN EPIDEMIOLOGICAL AND RADIOLOGICAL STUDY. *Journal of Bone & Joint Surgery, British Volume*, 92-B(2):250–257, Feb. 2010. ISSN 2049-4394, 2049-4408. doi:10.1302/0301-620X.92B2.22488. URL <http://www.bjj.boneandjoint.org.uk.ezproxy.library.ubc.ca/content/92-B/2/250>. [Accessed on 2013-12-16]. → pages
- [70] M. Fois, A. Lamure, M. J. Fauran, and C. Lacabanne. Study of human cortical bone and demineralized human cortical bone viscoelasticity. *Journal of Applied Polymer Science*, 79(14):2527–2533, 2001. ISSN 1097-4628. doi:10.1002/1097-4628(20010401)79:14<2527::AID-APP1061>3.0.CO;2-J. [Accessed on 2013-08-27]. → pages
- [71] C. M. Ford, T. M. Keaveny, and W. C. Hayes. The effect of impact direction on the structural capacity of the proximal femur during falls. *Journal of Bone and Mineral Research*, 11(3):377–383, Mar. 1996. ISSN 1523-4681. doi:10.1002/jbmр.5650110311. [Accessed on 2011-07-05]. → pages
- [72] L. Forsén, A. J. Søgaard, H. E. Meyer, T.-H. Edna, and B. Kopjar. Survival after hip fracture: Short- and long-term excess mortality according to age and gender. *Osteoporosis International*, 10(1):73–78, Aug. 1999. ISSN 0937-941X. doi:10.1007/s001980050197. URL <http://www.springerlink.com/content/mktcer4tclde8v48/>. [Accessed on 2011-02-02]. → pages
- [73] D. Fyhrie and M. Schaffler. Failure mechanisms in human vertebral cancellous bone. *Bone*, 15(1):105–109, Jan. 1994. ISSN 8756-3282. doi:10.1016/8756-3282(94)90900-8. URL <http://www.sciencedirect.com/science/article/B6T4Y-4BWV5HH-M/2/d254ae7ad958084c26695537b89dd02b>. [Accessed on 2011-03-17]. → pages
- [74] E. E. Gdoutos. *Fracture mechanics an introduction*. Springer ; Distributed in North, Central and South America by Springer, Dordrecht; Norwell, MA, 2005. ISBN 9781402031533 140203153X 1402028636 9781402028632. URL <http://public.eblib.com/EBLPublic/PublicView.do?ptID=303140>. [Accessed on 2013-07-23]. → pages
- [75] H. K. Genant, S. Grampp, C. C. Glüer, K. G. Faulkner, M. Jergas, K. Engelke, S. Hagiwara, and C. van Kuijk. Universal standardization for dual x-ray

- absorptiometry: Patient and phantom cross-calibration results. *Journal of Bone and Mineral Research*, 9(10):1503–1514, 1994. ISSN 1523-4681.
doi:10.1002/jbm.5650091002. [Accessed on 2013-07-22]. → pages
- [76] S. Gilchrist, P. Guy, and P. A. Cripton. Development of an inertia-driven model of sideways fall for detailed study of femur fracture mechanics. *Journal of Biomechanical Engineering*, 135(12):121001–121001, Oct. 2013. ISSN 0148-0731.
doi:10.1115/1.4025390. URL <http://dx.doi.org/10.1115/1.4025390>. [Accessed on 2013-10-23]. → pages
- [77] S. Gilchrist, K. K. Nishiyama, P. de Bakker, P. Guy, S. K. Boyd, T. Oxland, and P. A. Cripton. Impact and constant displacement rate loading changes the response of the femur in fall simulation. *Journal of Biomechanics*, (Conditionally Accepted), 2013. → pages
- [78] R. Goulet, S. Goldstein, M. Ciarelli, J. Kuhn, M. Brown, and L. Feldkamp. The relationship between the structural and orthogonal compressive properties of trabecular bone. *Journal of Biomechanics*, 27(4):375–377, 1994. → pages
- [79] H. Gray and Warren Harmon Lewis. *Anatomy of the Human Body*. Lea & Febiger, Philadelphia, 20th edition, 1918. ISBN 1-58734-102-6. → pages
- [80] S. L. Greenspan, E. R. Myers, L. A. Maitland, N. M. Resnick, and W. C. Hayes. Fall severity and bone mineral density as risk factors for hip fracture in ambulatory elderly. *JAMA: The Journal of the American Medical Association*, 271(2):128 –133, Jan. 1994.
doi:10.1001/jama.1994.03510260060029. [Accessed on 2011-08-29]. → pages
- [81] J. A. Grisso, J. L. Kelsey, B. L. Strom, G. Y. Ghiu, G. Maislin, L. A. O'Brien, S. Hoffman, and F. Kaplan. Risk factors for falls as a cause of hip fracture in women. *New England Journal of Medicine*, 324(19):1326–1331, May 1991. ISSN 0028-4793.
doi:doi:10.1056/NEJM199105093241905. URL
<http://dx.doi.org/10.1056/NEJM199105093241905>. [Accessed on 2014-01-11]. → pages
- [82] G. Guglielmi and S. K. Grimston. Osteoporosis: diagnosis with lateral and posteroanterior dual x-ray absorptiometry compared with quantitative CT. *Radiology*, 192(3):845–50, 1994. ISSN 0033-8419. → pages
- [83] H. Guillemot, B. Besnault, S. Robin, C. Got, J. Y. Le Coz, F. Lavaste, and J.-P. Lassau. Pelvic injuries in side impact collisions: A field accident analysis and dynamic tests on isolated pelvic bones. In *SAE Transactions*, Warrendale, PA, Nov. 1997. SAE International. URL <http://digitallibrary.sae.org/searchresults/>. [Accessed on 2013-07-19]. → pages
- [84] L. Gustafsson, B. Jacobson, and L. Kusoffsky. X ray spectrophotometry for bone-mineral determinations. *Medical and Biological Engineering*, 12(1):113–119, Jan. 1974. ISSN 0025-696X, 1741-0444. doi:10.1007/BF02629842. URL

<http://link.springer.com/article/10.1007/BF02629842>. [Accessed on 2013-07-22]. → pages

- [85] T. P. Haines, K. D. Hill, K. L. Bennell, and R. H. Osborne. Hip protector use amongst older hospital inpatients: compliance and functional consequences. *Age and Ageing*, 35(5):520 –523, Sept. 2006. doi:10.1093/ageing/afl020. URL <http://ageing.oxfordjournals.org/content/35/5/520.short>. → pages
- [86] S. Haleem, L. Lutchman, R. Mayahi, J. Grice, and M. Parker. Mortality following hip fracture: Trends and geographical variations over the last 40 years. *Injury*, 39(10): 1157–1163, Oct. 2008. ISSN 0020-1383. doi:16/j.injury.2008.03.022. URL <http://www.sciencedirect.com/science/article/pii/S0020138308001423>. [Accessed on 2011-09-01]. → pages
- [87] U. Hansen, P. Ziopoulos, R. Simpson, J. D. Currey, and D. Hynd. The effect of strain rate on the mechanical properties of human cortical bone. *Journal of Biomechanical Engineering*, 130(1):011011, 2008. ISSN 01480731. doi:10.1115/1.2838032. URL <http://Biomechanical.asmedigitalcollection.asme.org/article.aspx?articleid=1475452>. [Accessed on 2013-06-04]. → pages
- [88] M. R. Hardisty and C. M. Whyne. Whole bone strain quantification by image registration: A validation study. *Journal of Biomechanical Engineering*, 131(6): 064502–6, June 2009. doi:10.1115/1.3127249. URL <http://link.aip.org/link/?JBY/131/064502/1>. [Accessed on 2010-12-08]. → pages
- [89] M. R. Hardisty, M. Akens, A. J. Yee, and C. M. Whyne. Image registration demonstrates the growth plate has a variable affect on vertebral strain. *Annals of Biomedical Engineering*, 38(9):2948–2955, May 2010. ISSN 0090-6964. doi:10.1007/s10439-010-0052-0. URL <http://www.springerlink.com/content/640185lm83451h83/>. [Accessed on 2011-01-13]. → pages
- [90] T. P. Harrigan, M. Jasty, R. W. Mann, and W. H. Harris. Limitations of the continuum assumption in cancellous bone. *Journal of Biomechanics*, 21(4):269–275, 1988. ISSN 0021-9290. doi:10.1016/0021-9290(88)90257-6. URL <http://www.sciencedirect.com/science/article/pii/0021929088902576>. [Accessed on 2012-02-10]. → pages
- [91] W. C. Hayes and M. L. Bouxsein. Biomechanics of cortical and trabecular bone: Implications for assessment of fracture risk. In *Basic Orthopaedic Biomechanics*, page 43. Lippincott Williams & Wilkins, 2 edition, June 1997. ISBN 978-0397516841. → pages
- [92] W. C. Hayes, E. R. Myers, S. N. Robinovitch, A. Van Den Kroonenberg, A. C. Courtney, and T. A. McMahon. Etiology and prevention of age-related hip fractures. *Bone*, 18(1 Suppl):77S–86S, 1996. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=8717551. [Accessed on 2014-01-11]. → pages

- [93] M. Heim, A. Adunski, and A. Chechick. Nonoperative treatment of intracapsular fractures of the proximal femur. *Clinical Orthopaedics and Related Research*, pages 35–41, June 2002. ISSN 0009-921X. URL <http://ovidsp.ovid.com/ovidweb.cgi?T=JS&CSC=Y&NEWS=N&PAGE=fulltext&D=ovftf&AN=00003086-200206000-00006>. [Accessed on 2013-07-25]. → pages
- [94] P. F. Heini, T. Franz, C. Fankhauser, B. Gasser, and R. Ganz. Femoroplasty-augmentation of mechanical properties in the osteoporotic proximal femur: a biomechanical investigation of PMMA reinforcement in cadaver bones. *Clinical Biomechanics*, 19(5):506–12, 2004. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=15182986. [Accessed on 2014-01-11]. → pages
- [95] B. Helgason, E. Perilli, E. Schileo, F. Taddei, S. Brynjólfsson, and M. Viceconti. Mathematical relationships between bone density and mechanical properties: A literature review. *Clinical Biomechanics*, 23(2):135–146, Feb. 2008. ISSN 0268-0033. doi:10.1016/j.clinbiomech.2007.08.024. URL <http://www.sciencedirect.com/science/article/B6T59-4PVY2YM-1/2/e4abdc04bc4af35816ee129cc8d224fe>. [Accessed on 2010-12-08]. → pages
- [96] A. B. Hodsman, D. A. Hanley, and R. Josse. Do bisphosphonates reduce the risk of osteoporotic fractures? an evaluation of the evidence to date. *CMAJ: Canadian Medical Association Journal*, 166(11):1426–1430, May 2002. ISSN 0820-3946. → pages
- [97] G. Holzer, G. von Skrbensky, L. A. Holzer, and W. Pichl. Hip fractures and the contribution of cortical versus trabecular bone to femoral neck strength. *Journal of Bone and Mineral Research*, 24(3):468–474, Mar. 2009. ISSN 0884-0431. doi:10.1359/jbmr.081108. URL <http://doi.wiley.com/10.1359/jbmr.081108>. [Accessed on 2011-07-05]. → pages
- [98] L. Ibanez. ImageOriginAndSpacing.fig, 2003. URL http://itk.org/gitweb?p=ITKSoftwareGuide.git;a=blob_plain;f=SoftwareGuide/Art/ImageOriginAndSpacing.fig;hb=HEAD. [Accessed on 2014-01-11]. → pages
- [99] L. Ibanez. RegistrationComponentsDiagram.fig, 2003. URL http://itk.org/gitweb?p=ITKSoftwareGuide.git;a=blob_plain;f=SoftwareGuide/Art/RegistrationComponentsDiagram.fig;hb=HEAD. [Accessed on 2014-01-11]. → pages
- [100] L. Ibanez, W. Schroeder, L. Ng, J. Cates, and Insight Software Consortium. *ITK Software Guide : updated for version 2.4*. Kitware Inc., 2003. ISBN 9781930934153. URL <http://www.itk.org/ItkSoftwareGuide.pdf>. [Accessed on 2014-01-11]. → pages
- [101] Inversitus. Prostatic stone, May 2012. URL http://upload.wikimedia.org/wikipedia/commons/3/34/Prostatic_stone.jpg. [Accessed on 2014-01-11]. → pages
- [102] B. Jacobson. X-ray spectrophotometry in vivo. *American Journal of Roentgenology, Radium Therapy and Nuclear Medicine*, Vol: 91, Jan. 1964. URL

http://www.osti.gov/energycitations/product.biblio.jsp?osti_id=4143053. [Accessed on 2013-07-22]. → pages

- [103] A. K. Jain, A. V. Maheshwari, S. Nath, M. P. Singh, and M. Nagar. Anteversion of the femoral neck in indian dry femora. *Journal of Orthopaedic Science: Official Journal of the Japanese Orthopaedic Association*, 8(3):334–340, 2003. ISSN 0949-2658. doi:10.1007/s10776-003-0648-5. URL <http://www.ncbi.nlm.nih.gov/pubmed/12768475>. [Accessed on 2010-10-24]. → pages
- [104] O. Johnell, J. A. Kanis, A. Oden, H. Johansson, C. De Laet, P. Delmas, J. A. Eisman, S. Fujiwara, H. Kroger, D. Mellstrom, P. J. Meunier, L. J. Melton III, T. O'Neill, H. Pols, J. Reeve, A. Silman, and A. Tenenhouse. Predictive value of BMD for hip and other fractures. *Journal of Bone and Mineral Research*, 20(7):1185–1194, July 2005. ISSN 1523-4681. doi:10.1359/JBMR.050304. [Accessed on 2011-09-01]. → pages
- [105] M. M. Juszczyk, L. Cristofolini, and M. Viceconti. The human proximal femur behaves linearly elastic up to failure under physiological loading conditions. *Journal of Biomechanics*, 44(12):2259–2266, Aug. 2011. ISSN 00219290. doi:10.1016/j.jbiomech.2011.05.038. URL <http://linkinghub.elsevier.com/retrieve/pii/S0021929011004325>. [Accessed on 2013-12-14]. → pages
- [106] J. A. Kanis, O. Johnell, A. Oden, H. Johansson, and E. McCloskey. FRAX(TM) and the assessment of fracture probability in men and women from the UK. *Osteoporosis International*, 19(4):385–397, Apr. 2008. ISSN 0937-941X. doi:10.1007/s00198-007-0543-5. URL <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2267485/>. [Accessed on 2013-11-05]. → pages
- [107] J. A. Kanis, A. Oden, H. Johansson, F. Borgström, O. Ström, and E. McCloskey. FRAX® and its applications to clinical practice. *Bone*, 44(5):734–743, May 2009. ISSN 8756-3282. doi:10.1016/j.bone.2009.01.373. URL <http://www.sciencedirect.com/science/article/pii/S8756328209004049>. [Accessed on 2011-09-01]. → pages
- [108] P. Kannus, P. Leiponen, J. Parkkari, M. Palvanen, and M. Järvinen. A sideways fall and hip fracture. *Bone*, 39(2):383–384, Aug. 2006. ISSN 8756-3282. doi:10.1016/j.bone.2006.01.148. URL <http://www.sciencedirect.com/science/article/B6T4Y-4JCCG4J-1/2/06c19bc989590cb1ad8afb05ab295ab0>. [Accessed on 2011-05-17]. → pages
- [109] T. M. Keaveny, E. F. Morgan, G. L. Niebur, and O. C. Yeh. Biomechanics of trabecular bone. *Annual Review of Biomedical Engineering*, 3:307–333, 2001. ISSN 1523-9829. doi:10.1146/annurev.bioeng.3.1.307. URL <http://www.ncbi.nlm.nih.gov/pubmed/11447066>. [Accessed on 2011-04-18]. → pages
- [110] T. M. Keaveny, P. F. Hoffmann, M. Singh, L. Palermo, J. P. Bilezikian, S. L. Greenspan, and D. M. Black. Femoral bone strength and its relation to cortical and

- trabecular changes after treatment with PTH, alendronate, and their combination as assessed by finite element analysis of quantitative CT scans. *Journal of Bone and Mineral Research*, 23(12):1974–1982, Dec. 2008. ISSN 0884-0431, 0884-0431. doi:10.1359/jbmbr.080805. URL <http://doi.wiley.com/10.1359/jbmbr.080805>. [Accessed on 2012-05-15]. → pages
- [111] J. Keyak, H. B. Skinnera, and J. A. Flemingc. Effect of force direction on femoral fracture load for two types of loading conditions. *Journal of Orthopaedic Research*, 19 (4):539–544, July 2001. doi:10.1016/S0736-0266(00)00046-2. → pages
- [112] J. H. Keyak. Relationships between femoral fracture loads for two load configurations. *Journal of Biomechanics*, 33(4):499–502, Apr. 2000. ISSN 0021-9290. doi:16/S0021-9290(99)00202-X. URL <http://www.sciencedirect.com/science/article/pii/S002192909900202X>. [Accessed on 2011-09-01]. → pages
- [113] J. H. Keyak and S. A. Rossi. Prediction of femoral fracture load using finite element models: an examination of stress- and strain-based failure theories. *Journal of Biomechanics*, 33(2):209–214, Feb. 2000. ISSN 0021-9290. doi:10.1016/S0021-9290(99)00152-9. URL <http://www.sciencedirect.com/science/article/B6T82-3Y6HBCC-9/2/4b9a3716a85be522355fdd99457ed6f5>. [Accessed on 2010-11-04]. → pages
- [114] J. H. Keyak, S. A. Rossi, K. A. Jones, and H. B. Skinner. Prediction of femoral fracture load using automated finite element modeling. *Journal of Biomechanics*, 31(2): 125–133, May 1998. ISSN 0021-9290. doi:10.1016/S0021-9290(97)00123-1. URL <http://www.sciencedirect.com/science/article/B6T82-3VCCR77-3/2/4eb0c4aae90fddb4c84e52c8337298e4>. [Accessed on 2010-11-04]. → pages
- [115] J. H. Keyak, S. A. Rossi, K. A. Jones, C. M. Les, and H. B. Skinner. Prediction of fracture location in the proximal femur using finite element models. *Medical Engineering and Physics*, 23(9):657–664, 2001. → pages
- [116] G. Khang, K. Choi, C.-S. Kim, J. S. Yang, and T.-S. Bae. A study of korean femoral geometry. *Clinical Orthopaedics and Related Research*, (406):116–122, Jan. 2003. ISSN 0009-921X. doi:10.1097/01.blo.0000030502.43495.c1. URL <http://www.ncbi.nlm.nih.gov/pubmed/12579009>. [Accessed on 2010-10-24]. → pages
- [117] D. P. Kiel, J. Magaziner, S. Zimmerman, L. Ball, B. A. Barton, K. M. Brown, J. P. Stone, D. Dewkett, and S. J. Birge. Efficacy of a hip protector to prevent hip fracture in nursing home residents. *JAMA: The Journal of the American Medical Association*, 298 (4):413 –422, July 2007. doi:10.1001/jama.298.4.413. [Accessed on 2011-09-01]. → pages
- [118] P. C. Kingsley and K. L. Olmsted. A study to determine the angle of anteversion of the neck of the femur. *The Journal of Bone and Joint Surgery. American Volume*, 30A(3): 745–751, July 1948. ISSN 0021-9355. URL <http://www.ncbi.nlm.nih.gov/pubmed/18109784>. [Accessed on 2010-10-21]. → pages

- [119] J. E. Koivumäki, J. Thevenot, P. Pulkkinen, V. Kuhn, T. M. Link, F. Eckstein, and T. Jämsä. CT-based finite element models can be used to estimate experimentally measured failure loads in the proximal femur. *Bone*, 50(4):824–829, Apr. 2012. ISSN 8756-3282. doi:10.1016/j.bone.2012.01.012. URL <http://www.sciencedirect.com/science/article/pii/S8756328212000221>. [Accessed on 2013-08-05]. → pages
- [120] L. o. Konrad. Compact bone, 2009. URL http://commons.wikimedia.org/wiki/File:Compact_bone.png. [Accessed on 2014-01-11]. → pages
- [121] R. M. Kulin, F. Jiang, and K. S. Vecchio. Effects of age and loading rate on equine cortical bone failure. *Journal of the Mechanical Behavior of Biomedical Materials*, 4(1):57–75, Jan. 2011. ISSN 17516161. doi:10.1016/j.jmbbm.2010.09.006. URL <http://linkinghub.elsevier.com/retrieve/pii/S1751616110001293>. [Accessed on 2013-06-04]. → pages
- [122] R. M. Kulin, F. Jiang, and K. S. Vecchio. Loading rate effects on the r-curve behavior of cortical bone. *Acta Biomaterialia*, 7(2):724–732, Feb. 2011. ISSN 1742-7061. doi:10.1016/j.actbio.2010.09.027. URL <http://www.sciencedirect.com/science/article/pii/S174270611000437X>. [Accessed on 2011-10-24]. → pages
- [123] A. C. Laing and S. N. Robinovitch. The force attenuation provided by hip protectors depends on impact velocity, pelvic size, and soft tissue stiffness. *Journal of Biomechanical Engineering*, 130(6):061005–9, Dec. 2008. doi:10.1115/1.2979867. URL <http://link.aip.org/link/?JBY/130/061005/1>. [Accessed on 2011-02-03]. → pages
- [124] A. C. Laing and S. N. Robinovitch. Characterizing the effective stiffness of the pelvis during sideways falls on the hip. *Journal of Biomechanics*, 43(10):1898–1904, July 2010. ISSN 1873-2380. doi:10.1016/j.jbiomech.2010.03.025. URL <http://www.ncbi.nlm.nih.gov/pubmed/20398905>. [Accessed on 2010-10-27]. → pages
- [125] A. C. Laing, I. Tootoonchi, P. A. Hulme, and S. N. Robinovitch. Effect of compliant flooring on impact force during falls on the hip. *Journal of Orthopaedic Research*, 24(7):1405–1411, July 2006. ISSN 0736-0266. doi:10.1002/jor.20172. URL <http://www.ncbi.nlm.nih.gov/pubmed/16705716>. [Accessed on 2010-10-28]. → pages
- [126] P. J. Laz, J. Q. Stowe, M. A. Baldwin, A. J. Petrella, and P. J. Rullkoetter. Incorporating uncertainty in mechanical properties for finite element-based evaluation of bone mechanics. *Journal of Biomechanics*, 40(13):2831–2836, 2007. ISSN 0021-9290. doi:10.1016/j.jbiomech.2007.03.013. URL <http://www.sciencedirect.com/science/article/pii/S002192900700125X>. [Accessed on 2013-08-08]. → pages
- [127] D. Lecompte, A. Smits, S. Bossuyt, H. Sol, J. Vantomme, D. Van Hemelrijck, and A. Habraken. Quality assessment of speckle patterns for digital image correlation.

- Optics and Lasers in Engineering, 44(11):1132–1145, Nov. 2006. ISSN 0143-8166.
doi:10.1016/j.optlaseng.2005.10.004. URL
<http://www.sciencedirect.com/science/article/pii/S0143816605001727>. [Accessed on 2013-06-07]. → pages
- [128] K. A. Lefavre, A. R. Levy, B. Sobolev, S. Y. Cheng, L. Kuramoto, and P. Guy. Changes in first hip fracture rates in british columbia canada, 1990–2004. *Osteoporosis International*, 22(11):2817–2827, Jan. 2011. ISSN 0937-941X, 1433-2965.
doi:10.1007/s00198-010-1488-7. URL
<http://www.springerlink.com/content/g46gp40740235114/>. [Accessed on 2011-09-13]. → pages
- [129] I. Leichter, A. Simkin, V. Neeman, C. Jabschinsky, D. Schoenfeld, A. Foldes, and M. Liebergall. Optical processing of radiographic trabecular pattern versus bone mineral density of proximal femur as measures of bone strength. *Journal of Clinical Densitometry*, 4(2):121–129, 2001. doi:10.1385/JCD:4:2:121. → pages
- [130] N. Lenoir, M. Bornert, J. Desrues, P. Bésuelle, and G. Viggiani. Volumetric digital image correlation applied to x-ray microtomography images from triaxial compression tests on argillaceous rock. *Strain*, 43(3):193–205, Aug. 2007. ISSN 00392103.
doi:10.1111/j.1475-1305.2007.00348.x. [Accessed on 2010-12-08]. → pages
- [131] W. D. Leslie and S. Morin. Fracture burden in relation to low bone mineral density and FRAX® probability. *Journal of Clinical Densitometry*, 14(3):279–285, July 2011. ISSN 1094-6950. doi:10.1016/j.jocd.2011.04.010. URL
<http://www.sciencedirect.com/science/article/pii/S1094695011001090>. [Accessed on 2011-08-30]. → pages
- [132] W. D. Leslie, S. O'Donnell, S. Jean, C. Lagacé, P. Walsh, C. Bancej, S. Morin, D. A. Hanley, and A. Papaioannou. Trends in hip fracture rates in canada. *JAMA: The Journal of the American Medical Association*, 302(8):883 –889, 2009.
doi:10.1001/jama.2009.1231. [Accessed on 2011-09-01]. → pages
- [133] N. Li, E. Tsushima, and H. Tsushima. Comparison of impact force attenuation by various combinations of hip protector and flooring material using a simplified fall-impact simulationdevice. *Journal of Biomechanics*, 46(6):1140–6, Apr. 2013. ISSN 0021-9290. doi:10.1016/j.jbiomech.2013.01.007. URL
<http://www.sciencedirect.com/science/article/pii/S0021929013000353>. [Accessed on 2013-03-16]. → pages
- [134] F. Linde, P. Nørgaard, I. Hvid, A. Odgaard, and K. Søballe. Mechanical properties of trabecular bone. dependency on strain rate. *Journal of Biomechanics*, 24(9):803–809, 1991. ISSN 0021-9290. doi:10.1016/0021-9290(91)90305-7. URL
<http://www.sciencedirect.com/science/article/pii/0021929091903057>. [Accessed on 2011-10-24]. → pages
- [135] C. Liu. On the minimum size of representative volume element: An experimental investigation. *Experimental Mechanics*, 45(3):238–243, June 2005. ISSN 0014-4851.

- doi:10.1007/BF02427947. URL
<http://www.springerlink.com/content/m736734145t80829/>. [Accessed on 2010-12-08]. → pages
- [136] Z. Liu and L. E. Bilston. Large deformation shear properties of liver tissue. *Biorheology*, 39(6):735–742, 2002. ISSN 0006-355X. → pages
- [137] E. M. Lochmüller, O. Groll, V. Kuhn, and F. Eckstein. Mechanical strength of the proximal femur as predicted from geometric and densitometric bone properties at the lower limb versus the distal radius. *Bone*, 30(1):207–216, Jan. 2002. ISSN 8756-3282. doi:10.1016/S8756-3282(01)00621-4. URL
<http://www.ncbi.nlm.nih.gov/pubmed/11792587>. [Accessed on 2010-10-25]. → pages
- [138] J. Lotz and W. Hayes. The use of quantitative computed tomography to estimate risk of fracture of the hip from falls. *Journal of Bone and Joint Surgery American*, 72(5):689–700, June 1990. [Accessed on 2010-11-03]. → pages
- [139] J. C. Lotz, E. J. Cheal, and W. C. Hayes. Fracture prediction for the proximal femur using finite element models: Part II–nonlinear analysis. *Journal of Biomechanical Engineering*, 113(4):361–5, 1991. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=1762431. [Accessed on 2014-01-11]. → pages
- [140] J. C. Lotz, E. J. Cheal, and W. C. Hayes. Fracture prediction for the proximal femur using finite element models: Part I–linear analysis. *Journal of Biomechanical Engineering*, 113(4):353–60, 1991. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=1762430. [Accessed on 2014-01-11]. → pages
- [141] J. C. Lotz, T. N. Gerhart, and W. C. Hayes. Mechanical properties of metaphyseal bone in the proximal femur. *Journal of Biomechanics*, 24(5):317–29, 1991. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=2050708. [Accessed on 2014-01-11]. → pages
- [142] J. C. Lotz, E. J. Cheal, and W. C. Hayes. Stress distributions within the proximal femur during gait and falls: Implications for osteoporotic fracture. *Osteoporosis International*, 5:252–261, July 1995. ISSN 0937-941X, 1433-2965. doi:10.1007/BF01774015. URL <http://www.springerlink.com/content/p6629864w5404g51/>. [Accessed on 2011-09-06]. → pages
- [143] H. Lu and P. D. Cary. Deformation measurements by digital image correlation: Implementation of a second-order displacement gradient. *Experimental Mechanics*, 40(4):393–400, Dec. 2000. ISSN 0014-4851. doi:10.1007/BF02326485. URL
<http://www.springerlink.com/content/v763526k4887616r/>. [Accessed on 2010-11-04]. → pages
- [144] G. Lyritis. Epidemiology and socioeconomic cost of osteoporotic fractures in greece. *Calcified tissue international*, 51(2):93–94, 1992. URL

<http://www.springerlink.com/index/J502856841762433.pdf>. [Accessed on 2013-03-13]. → pages

- [145] G. P. Lyritis. Epidemiology of hip fracture: The MEDOS study. *Osteoporosis International*, 6(3):11–15, May 1996. ISSN 0937-941X, 1433-2965. doi:10.1007/BF01623757. URL <http://link.springer.com/article/10.1007/BF01623757>. [Accessed on 2013-03-13]. → pages
- [146] S. Majumder, A. Roychowdhury, and S. Pal. Simulation of hip fracture in sideways fall using a 3D finite element model of pelvis-femur-soft tissue complex with simplified representation of whole body. *Medical Engineering and Physics*, 29(10):1167–1178, Dec. 2007. ISSN 1350-4533. doi:16/j.medengphy.2006.11.001. URL <http://www.sciencedirect.com/science/article/pii/S1350453306002323>. [Accessed on 2011-07-27]. → pages
- [147] S. Majumder, A. Roychowdhury, and S. Pal. Effects of trochanteric soft tissue thickness and hip impact velocity on hip fracture in sideways fall through 3D finite element simulations. *Journal of Biomechanics*, 41(13):2834–2842, Sept. 2008. ISSN 0021-9290. doi:10.1016/j.jbiomech.2008.07.001. URL <http://www.ncbi.nlm.nih.gov/pubmed/18718597>. [Accessed on 2010-11-01]. → pages
- [148] S. L. Manske, T. Liu-Ambrose, P. M. Bakker, D. Liu, S. Kontulainen, P. Guy, T. R. Oxland, and H. A. McKay. Femoral neck cortical geometry measured with magnetic resonance imaging is associated with proximal femur strength. *Osteoporosis International*, 17(10):1539–1545, July 2006. ISSN 0937-941X. doi:10.1007/s00198-006-0162-6. URL <http://www.springerlink.com/content/x306v5v06728n141/>. [Accessed on 2010-11-03]. → pages
- [149] S. L. Manske, T. Liu-Ambrose, D. M. L. Cooper, S. Kontulainen, P. Guy, B. B. Forster, and H. A. McKay. Cortical and trabecular bone in the femoral neck both contribute to proximal femur failure load prediction. *Osteoporosis International*, 20(3):445–453, July 2008. ISSN 0937-941X. doi:10.1007/s00198-008-0675-2. URL <http://www.springerlink.com/content/7k63844871667185/>. [Accessed on 2011-09-01]. → pages
- [150] I. Masoud, F. Shapiro, R. Kent, and A. Moses. A longitudinal study of the growth of the new zealand white rabbit: cumulative and biweekly incremental growth rates for body length, body weight, femoral length, and tibial length. *Journal of Orthopaedic Research: Official Publication of the Orthopaedic Research Society*, 4(2):221–231, 1986. ISSN 0736-0266. doi:10.1002/jor.1100040211. URL <http://www.ncbi.nlm.nih.gov/pubmed/3712130>. [Accessed on 2011-01-06]. → pages
- [151] T. Masud and R. O. Morris. Epidemiology of falls. *Age and Ageing*, 30(suppl 4):3–7, Nov. 2001. ISSN 0002-0729, 1468-2834. doi:10.1093/ageing/30.suppl_4.3. URL http://ageing.oxfordjournals.org/content/30/suppl_4/3. [Accessed on 2013-08-07]. → pages

- [152] S. Matsuda, H. Matsuda, T. Miyagi, K. Sasaki, Y. Iwamoto, and H. Miura. Femoral condyle geometry in the normal and varus knee. *Clinical Orthopaedics and Related Research*, pages 183–188, Apr. 1998. ISSN 0009-921X. URL <http://ovidsp.ovid.com/ovidweb.cgi?T=JS&CSC=Y&NEWS=N&PAGE=fulltext&D=ovftc&AN=00003086-199804000-00022>. [Accessed on 2013-09-09]. → pages
- [153] S. Matsuda, H. Miura, R. Nagamine, T. Mawatari, M. Tokunaga, R. Nabeyama, and Y. Iwamoto. Anatomical analysis of the femoral condyle in normal and osteoarthritic knees. *Journal of Orthopaedic Research*, 22(1):104–109, 2004. ISSN 1554-527X. doi:10.1016/S0736-0266(03)00134-7. [Accessed on 2013-09-09]. → pages
- [154] P. M. Mayhew, C. D. Thomas, J. G. Clement, N. Loveridge, T. J. Beck, W. Bonfield, C. J. Burgoyne, and J. Reeve. Relation between age, femoral neck cortical stability, and hip fracture risk. *The Lancet*, 366(9480):129–135, 2005. → pages
- [155] R. McCalden, J. McGeough, M. Barker, and C. Court-Brown. Age-related changes in the tensile properties of cortical bone. the relative importance of changes in porosity, mineralization, and microstructure. *Journal of Bone and Joint Surgery American*, 75 (8):1193–1205, Aug. 1993. [Accessed on 2011-03-17]. → pages
- [156] R. W. McCalden, J. A. McGeough, and C. M. Court-Brown. Age-related changes in the compressive strength of cancellous bone. the relative importance of changes in density and trabecular architecture. *Journal of Bone and Joint Surgery American*, 79 (3):421–7, Mar. 1997. [Accessed on 2011-03-17]. → pages
- [157] J. H. McElhaney. Dynamic response of bone and muscle tissue. *Journal of Applied Physiology*, 21(4):1231–1236, July 1966. ISSN 8750-7587, 1522-1601. URL <http://jap.physiology.org/content/21/4/1231>. [Accessed on 2013-07-23]. → pages
- [158] J. C. Meza. Steepest descent. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(6):719–722, 2010. ISSN 1939-0068. doi:10.1002/wics.117. [Accessed on 2013-09-03]. → pages
- [159] J. D. Michelson, A. S. Myers, R. Jinnah, Q. Cox, and M. M. Van Natta. Epidemiology of hip fractures among the elderly: Risk factors for fracture type. *Clinical Orthopaedics and Related Research*, pages 129–135, Feb. 1995. ISSN 0009-921X. URL <http://ovidsp.ovid.com/ovidweb.cgi?T=JS&CSC=Y&NEWS=N&PAGE=fulltext&D=ovftb&AN=00003086-199502000-00016>. [Accessed on 2013-02-25]. → pages
- [160] R. J. Minns, A.-M. Marsh, A. Chuck, and J. Todd. Are hip protectors correctly positioned in use? *Age and Ageing*, 36(2):140 –144, Mar. 2007. doi:10.1093/ageing/afl186. [Accessed on 2010-11-25]. → pages
- [161] A. Moayyeri. The association between physical activity and osteoporotic fractures: A review of the evidence and implications for future research. *Annals of Epidemiology*, 18(11):827–835, Nov. 2008. ISSN 1047-2797. doi:16/j.annepidem.2008.08.007. URL <http://www.sciencedirect.com/science/article/pii/S104727970800166X>. [Accessed on 2011-09-01]. → pages

- [162] L. Mosekilde. Age-related changes in vertebral trabecular bone architecture—assessed by a new method. *Bone*, 9(4):247–250, 1988. → pages
- [163] N. M. Nachreiner, M. J. Findorff, J. F. Wyman, and T. C. McCarthy. Circumstances and consequences of falls in community-dwelling older women. *Journal of Women's Health*, 16(10):1437–1446, Dec. 2007. ISSN 1540-9996. doi:10.1089/jwh.2006.0245. URL <http://www.ncbi.nlm.nih.gov/pubmed/18062759>. [Accessed on 2011-09-01]. → pages
- [164] S. Nawathe, H. Akhlaghpour, M. L. Bouxsein, and T. M. Keaveny. Microstructural failure mechanisms in the human proximal femur for sideways fall loading. *Journal of Bone and Mineral Research*, 2013. ISSN 1523-4681. doi:10.1002/jbmr.2033. [Accessed on 2013-07-16]. → pages
- [165] K. E. Naylor, E. V. McCloskey, R. Eastell, and L. Yang. The use of DXA based finite element analysis of the proximal femur in a longitudinal study of hip fracture. *Journal of Bone and Mineral Research*, 28(5):1014–1021, 2012. ISSN 1523-4681. doi:10.1002/jbmр.1856. [Accessed on 2013-01-31]. → pages
- [166] A. Nazarian, M. Stauber, D. Zurakowski, B. D. Snyder, and R. Müller. The interaction of microstructure and volume fraction in predicting failure in cancellous bone. *Bone*, 39(6):1196–1202, Dec. 2006. ISSN 8756-3282. doi:10.1016/j.bone.2006.06.013. URL <http://www.ncbi.nlm.nih.gov/pubmed/16920051>. [Accessed on 2011-12-15]. → pages
- [167] A. Nazarian, J. Muller, D. Zurakowski, R. Müller, and B. D. Snyder. Densitometric, morphometric and mechanical distributions in the human proximal femur. *Journal of Biomechanics*, 40(11):2573–2579, 2007. ISSN 0021-9290. doi:10.1016/j.jbiomech.2006.11.022. URL <http://www.sciencedirect.com/science/article/B6T82-4MX4W12-3/2/a2d9ebb037e5bb95aaa239ff47579d4a>. [Accessed on 2011-02-04]. → pages
- [168] N. D. Nguyen, C. Pongchaiyakul, J. R. Center, J. A. Eisman, and T. V. Nguyen. Abdominal fat and hip fracture risk in the elderly: The dubbo osteoporosis epidemiology study. *BMC Musculoskeletal Disorders*, 6(1):11, Feb. 2005. ISSN 1471-2474. doi:10.1186/1471-2474-6-11. [Accessed on 2013-02-18]. → pages
- [169] N. D. Nguyen, C. Pongchaiyakul, J. R. Center, J. A. Eisman, and T. V. Nguyen. Identification of High-Risk individuals for hip fracture: A 14-Year prospective study. *Journal of Bone and Mineral Research*, 20(11):1921–1928, Nov. 2005. ISSN 1523-4681. doi:10.1359/JBMR.050520. [Accessed on 2011-09-01]. → pages
- [170] C. M. Nielson, M. L. Bouxsein, S. S. Freitas, K. E. Ensrud, and E. S. Orwoll. Trochanteric soft tissue thickness and hip fracture in older men. *Journal of Clinical Endocrinology and Metabolism*, 94(2):491–496, Feb. 2009. doi:10.1210/jc.2008-1640. [Accessed on 2010-11-25]. → pages
- [171] C. M. Nielson, L. M. Marshall, A. L. Adams, E. S. LeBlanc, P. M. Cawthon, K. Ensrud, M. L. Stefanick, E. Barrett-Connor, and E. S. Orwoll. BMI and fracture risk

- in older men: The osteoporotic fractures in men study (MrOS). *Journal of Bone and Mineral Research*, 26(3):496–502, 2011. ISSN 1523-4681. doi:10.1002/jbmr.235. [Accessed on 2013-02-18]. → pages
- [172] R. W. Nightingale, J. H. McElhaney, D. L. Camacho, M. Kleinberger, B. A. Winkelstein, and B. S. Myers. The dynamic responses of the cervical spine: buckling, end conditions, and tolerance in compressive impacts. In *SAE Conference Proceedings* P, page 451–472, 1997. → pages
- [173] K. K. Nishiyama, S. Gilchrist, P. Guy, P. Cripton, and S. K. Boyd. Proximal femur bone strength estimated by a computationally fast finite element analysis in a sideways fall configuration. *Journal of Biomechanics*, 46(7):1231–1236, Apr. 2013. ISSN 0021-9290. doi:10.1016/j.biomech.2013.02.025. URL <http://www.sciencedirect.com/science/article/pii/S002192901300105X>. [Accessed on 2013-05-13]. → pages
- [174] A. Odgaard. Three-dimensional methods for quantification of cancellous bone architecture. *Bone*, 20(4):315–328, Apr. 1997. ISSN 8756-3282. doi:10.1016/S8756-3282(97)00007-0. URL <http://www.sciencedirect.com/science/article/pii/S8756328297000070>. [Accessed on 2012-01-18]. → pages
- [175] A. Odgaard, J. Kabel, B. van Rietbergen, M. Dalstra, and R. Huiskes. Fabric and elastic principal directions of cancellous bone are closely related. *Journal of Biomechanics*, 30 (5):487–495, May 1997. ISSN 0021-9290. doi:10.1016/S0021-9290(96)00177-7. URL <http://www.sciencedirect.com/science/article/pii/S0021929096001777>. [Accessed on 2011-12-15]. → pages
- [176] C. Öhman, M. Baleani, C. Pani, F. Taddei, M. Alberghini, M. Viceconti, and M. Manfrini. Compressive behaviour of child and adult cortical bone. *Bone*, 49(4): 769–776, Oct. 2011. ISSN 8756-3282. doi:10.1016/j.bone.2011.06.035. URL <http://www.sciencedirect.com/science/article/pii/S8756328211010763>. [Accessed on 2013-07-11]. → pages
- [177] H. Okuzumi, A. Harada, H. Iwata, and N. Konishi. Effect on the femur of a new hip fracture preventive system using dropped-weight impact testing. *Journal of Bone and Mineral Research: The Official Journal of the American Society for Bone and Mineral Research*, 13(12):1940–1945, Dec. 1998. ISSN 0884-0431. doi:10.1359/jbmr.1998.13.12.1940. URL <http://www.ncbi.nlm.nih.gov/pubmed/9844113>. [Accessed on 2010-11-04]. → pages
- [178] Omega Engineering. 19mm diameter stainless steel compression load cell, metric, 0-100 to 0-5000 newtons, 2013. URL <http://www.omega.com/pptst/LCM302.html>. [Accessed on 2013-09-05]. → pages
- [179] E. S. Orwoll and SpringerLINK eBooks - English/International Collection (pre-2005). *Osteoporosis Pathophysiology and Clinical Management*. Contemporary

- Endocrinology Ser. Humana Press Springer [Distributor], New York : Secaucus, 2011.
URL http://GW2JH3XR2C.search.serialssolutions.com/?sid=sersol&SS.jc=TC_012881014&title=Osteoporosis. [Accessed on 2013-07-26]. → pages
- [180] E. S. Orwoll, L. M. Marshall, C. M. Nielson, S. R. Cummings, J. Lapidus, J. A. Cauley, K. Ensrud, N. Lane, P. R. Hoffmann, D. L. Kopperdahl, and T. M. Keaveny. Finite element analysis of the proximal femur and hip fracture risk in older men. *Journal of Bone and Mineral Research*, 24(3):475–483, 2009. ISSN 1523-4681.
doi:10.1359/jbmр.081201. [Accessed on 2013-02-18]. → pages
- [181] T. Ota, I. Yamamoto, and R. Morita. Fracture simulation of the femoral bone using the finite-element method: How a fracture initiates and proceeds. *Journal of Bone and Mineral Metabolism*, 17(2):108–112, May 1999. ISSN 0914-8779.
doi:10.1007/s007740050072. URL <http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s007740050072>. [Accessed on 2014-01-11]. → pages
- [182] B. Pan, H. Xie, Z. Wang, K. Qian, and Z. Wang. Study on subset size selection in digital image correlation for speckle patterns. *Optics Express*, 16(10):7037–7048, May 2008. doi:10.1364/OE.16.007037. [Accessed on 2013-06-07]. → pages
- [183] B. Pan, K. Qian, H. Xie, and A. Asundi. Two-dimensional digital image correlation for in-plane displacement and strain measurement: a review. *Measurement Science and Technology*, 20(6):062001, June 2009. ISSN 0957-0233.
doi:10.1088/0957-0233/20/6/062001. URL <http://iopscience.iop.org/0957-0233/20/6/062001>. [Accessed on 2010-11-09]. → pages
- [184] A. Parfitt. Trabecular bone architecture in the pathogenesis and prevention of fracture. *The American Journal of Medicine*, 82(1):68–72, 1987. → pages
- [185] A. M. Parfitt. Age-related structural changes in trabecular and cortical bone: Cellular mechanisms and biomechanical consequences. *Calcified Tissue International*, 36(S1):S123–S128, Mar. 1984. ISSN 0171-967X. doi:10.1007/BF02406145. URL <http://www.springerlink.com/content/5r41p47616264053/>. [Accessed on 2011-03-17]. → pages
- [186] E. D. Parker, M. A. Pereira, B. Virnig, and A. R. Folsom. The association of hip circumference with incident hip fracture in a cohort of postmenopausal women: The Iowa women's health study. *Annals of Epidemiology*, 18(11):836–841, Nov. 2008. ISSN 1047-2797. doi:10.1016/j.annepidem.2008.07.007. URL <http://www.sciencedirect.com/science/article/pii/S1047279708001646>. [Accessed on 2013-02-18]. → pages
- [187] M. J. Parker, W. J. Gillespie, and L. D. Gillespie. Effectiveness of hip protectors for preventing hip fractures in elderly people: systematic review. *BMJ*, 332(7541):571–574, Mar. 2006. doi:10.1136/bmj.38753.375324.7C. [Accessed on 2011-09-02]. → pages

- [188] J. Parkkari, P. Kannus, M. Palvanen, A. Natri, J. Vainio, H. Aho, I. Vuori, and M. Järvinen. Majority of hip fractures occur as a result of a fall and impact on the greater trochanter of the femur: A prospective controlled hip fracture study with 206 consecutive patients. *Calcified Tissue International*, 65(3):183–187, Sept. 1999. ISSN 0171-967X. doi:10.1007/s002239900679. URL <http://www.springerlink.com/content/ar3w48gkteh2ylkaf/>. [Accessed on 2011-05-17]. → pages
- [189] T. P. Pinilla, K. C. Boardman, M. L. Bouxsein, E. R. Myers, and W. C. Hayes. Impact direction from a fall influences the failure load of the proximal femur as much as age-related bone loss. *Calcified Tissue International*, 58(4):231–235, Apr. 1996. ISSN 0171-967X. doi:10.1007/BF02508641. URL <http://www.ncbi.nlm.nih.gov/pubmed/8661953>. [Accessed on 2010-11-04]. → pages
- [190] M. Pithioux, D. Subit, and P. Chabrand. Comparison of compact bone failure under two different loading rates: experimental and modelling approaches. *Medical Engineering and Physics*, 26(8):647–653, Oct. 2004. ISSN 1350-4533. doi:10.1016/j.medengphy.2004.05.002. URL <http://www.sciencedirect.com/science/article/pii/S1350453304000918>. [Accessed on 2012-12-05]. → pages
- [191] K. E. S. Poole, G. M. Treece, P. M. Mayhew, J. Vaculík, P. Dungl, M. Horák, J. J. Štěpán, and A. H. Gee. Cortical thickness mapping to identify focal osteoporosis in patients with hip fracture. *PLoS ONE*, 7(6):e38466, June 2012. ISSN 1932-6203. doi:10.1371/journal.pone.0038466. URL <http://dx.plos.org/10.1371/journal.pone.0038466>. [Accessed on 2012-06-17]. → pages
- [192] P. Pulkkinen, T. Jämsä, E.-M. Lochmüller, V. Kuhn, M. T. Nieminen, and F. Eckstein. Experimental hip fracture load can be predicted from plain radiography by combined analysis of trabecular bone structure and bone geometry. *Osteoporosis International*, 19(4):547–558, Apr. 2008. ISSN 0937-941X. doi:10.1007/s00198-007-0479-9. URL <http://www.ncbi.nlm.nih.gov/pubmed/17891327>. [Accessed on 2011-09-02]. → pages
- [193] T. Rand, G. Seidl, F. Kainberger, A. Resch, K. Hittmair, B. Schneider, C. C. Glüer, and H. Imhof. Impact of spinal degenerative changes on the evaluation of bone mineral density with dual energy x-ray absorptiometry (DXA). *Calcified Tissue International*, 60(5):430–433, May 1997. ISSN 0171-967X, 1432-0827. doi:10.1007/s002239900258. URL <http://link.springer.com/article/10.1007/s002239900258>. [Accessed on 2013-07-26]. → pages
- [194] D. T. Reilly and A. H. Burstein. The elastic and ultimate properties of compact bone tissue. *Journal of Biomechanics*, 8(6):393–405, 1975. ISSN 0021-9290. doi:10.1016/0021-9290(75)90075-5. URL <http://www.sciencedirect.com/science/article/pii/0021929075900755>. [Accessed on 2013-07-21]. → pages

- [195] J. Richmond, G. B. Aharonoff, J. D. Zuckerman, and K. J. Koval. Mortality risk after hip fracture. *Journal of Orthopaedic Trauma*, 17(1):53–56, Jan. 2003. ISSN 0890-5339. URL <http://www.ncbi.nlm.nih.gov/pubmed/12499968>. [Accessed on 2011-09-02]. → pages
- [196] L. Rincón-Kohli and P. K. Zysset. Multi-axial mechanical properties of human trabecular bone. *Biomechanics and Modeling in Mechanobiology*, 8(3):195–208, June 2009. ISSN 1617-7959, 1617-7940. doi:10.1007/s10237-008-0128-z. URL <http://link.springer.com/article/10.1007/s10237-008-0128-z>. [Accessed on 2013-07-22]. → pages
- [197] B. J. Roberts, E. Thrall, J. A. Muller, and M. L. Bouxsein. Comparison of hip fracture risk prediction by femoral aBMD to experimentally measured factor of risk. *Bone*, 46(3):742–746, Mar. 2010. ISSN 8756-3282. doi:10.1016/j.bone.2009.10.020. URL <http://www.sciencedirect.com/science/article/pii/S8756328209019887>. [Accessed on 2013-02-18]. → pages
- [198] D. M. Robertson and D. C. Smith. Compressive strength of mandibular bone as a function of microstructure and strain rate. *Journal of Biomechanics*, 11(10–12):455–471, 1978. ISSN 0021-9290. doi:10.1016/0021-9290(78)90057-X. URL <http://www.sciencedirect.com/science/article/pii/002192907890057X>. [Accessed on 2012-12-07]. → pages
- [199] S. N. Robinovitch, W. C. Hayes, and T. A. McMahon. Prediction of femoral impact forces in falls on the hip. *Journal of Biomechanical Engineering*, 113(4):366–374, Nov. 1991. ISSN 0148-0731. doi:10.1115/1.2895414. URL <http://www.ncbi.nlm.nih.gov/pubmed/1762432>. [Accessed on 2010-11-04]. → pages
- [200] S. N. Robinovitch, T. A. McMahon, and W. C. Hayes. Force attenuation in trochanteric soft tissues during impact from a fall. *Journal of Orthopaedic Research*, 13(6):956–962, Nov. 1995. ISSN 0736-0266. doi:10.1002/jor.1100130621. [Accessed on 2010-11-23]. → pages
- [201] S. N. Robinovitch, W. C. Hayes, and T. A. McMahon. Distribution of contact force during impact to the hip. *Annals of Biomedical Engineering*, 25(3):499–508, May 1997. ISSN 0090-6964. doi:10.1007/BF02684190. URL <http://www.springerlink.com/content/u884ww2507l660h2/>. [Accessed on 2011-02-03]. → pages
- [202] S. N. Robinovitch, S. L. Evans, J. Minns, A. C. Laing, P. Kannus, P. A. Cripton, S. Derler, S. J. Birge, D. Plant, I. D. Cameron, D. P. Kiel, J. Howland, K. Khan, and J. B. Lauritzen. Hip protectors: recommendations for biomechanical testing—an international consensus statement (part I). *Osteoporosis International*, 20(12):1977–1988, Oct. 2009. ISSN 0937-941X. doi:10.1007/s00198-009-1045-4. URL <http://www.springerlink.com/index/10.1007/s00198-009-1045-4>. [Accessed on 2011-05-15]. → pages

- [203] S. Roux, F. Hild, P. Viot, and D. Bernard. Three-dimensional image correlation from x-ray computed tomography of solid foam. *Composites Part A: Applied Science and Manufacturing*, 39(8):1253–1265, Aug. 2008. ISSN 1359-835X.
doi:10.1016/j.compositesa.2007.11.011. URL <http://www.sciencedirect.com/science/article/B6TWN-4R8NBHH-1/2/c2ba098fc26341f1552198691b9a422f>. [Accessed on 2011-01-25]. → pages
- [204] A. Saari, E. Itshayek, and P. A. Cripton. Cervical spinal cord deformation during simulated head-first impact injuries. *Journal of Biomechanics*, 44(14):2565–2571, Aug. 2011. ISSN 1873-2380. doi:10.1016/j.jbiomech.2011.06.015. URL <http://www.ncbi.nlm.nih.gov/pubmed/21840526>. [Accessed on 2011-09-05]. → pages
- [205] S. Saha and W. C. Hayes. Instrumented tensile-impact tests of bone. *Experimental Mechanics*, 14(12):473–478, 1974. URL <http://www.springerlink.com/index/MW7L27777J1L2RH0.pdf>. [Accessed on 2012-12-05]. → pages
- [206] S. Schlesinger, R. Crosbie, R. Gagne, G. Innis, C. Lalwani, J. Loch, R. Sylvester, R. Wright, N. Kheir, and D. Bartos. Terminology for model credibility. *SIMULATION*, 32(3):103–104, Mar. 1979. ISSN 0037-5497, 1741-3133.
doi:10.1177/003754977903200304. URL <http://sim.sagepub.com/content/32/3/103>. [Accessed on 2013-08-22]. → pages
- [207] H. W. Schreier and M. A. Sutton. Systematic errors in digital image correlation due to undermatched subset shape functions. *Experimental Mechanics*, 42(3):303–310, Sept. 2002. ISSN 0014-4851. doi:10.1007/BF02410987. URL <http://www.springerlink.com/content/yt8r554823n04328/>. [Accessed on 2010-11-09]. → pages
- [208] H. W. Schreier, J. R. Braasch, and M. A. Sutton. Systematic errors in digital image correlation caused by intensity interpolation. *Optical Engineering*, 39(11):2915–2921, Nov. 2000. URL <http://link.aip.org/link/?JOE/39/2915/1>. [Accessed on 2010-11-09]. → pages
- [209] A. K. Shah, J. Eissler, and T. Radomisli. Algorithms for the treatment of femoral neck fractures. *Clinical Orthopaedics and Related Research*, pages 28–34, June 2002. ISSN 0009-921X. URL <http://ovidsp.ovid.com/ovidweb.cgi?T=JS&CSC=Y&NEWS=N&PAGE=fulltext&D=ovftf&AN=00003086-200206000-00005>. [Accessed on 2013-07-26]. → pages
- [210] R. R. Simon. *Emergency orthopedics*. McGraw-Hill Medical, New York, 2011. ISBN 9780071625920 0071625925 9780071625937 0071625933 9780071625944 0071625941 9780071632522 0071632522. → pages
- [211] M. Singh, A. R. Nagrath, and P. S. Maini. Changes in trabecular pattern of the upper end of the femur as an index of osteoporosis. *The Journal of Bone and Joint Surgery American*, 52(3):457–467, Apr. 1970. ISSN 0021-9355. URL <http://www.ncbi.nlm.nih.gov/pubmed/5425640>. [Accessed on 2011-02-03]. → pages

- [212] E. S. Siris, Y.-T. Chen, T. A. Abbott, E. Barrett-Connor, P. D. Miller, L. E. Wehren, and M. L. Berger. Bone mineral density thresholds for pharmacological intervention to prevent fractures. *Archives of Internal Medicine*, 164(10):1108–1112, May 2004. doi:10.1001/archinte.164.10.1108. [Accessed on 2011-05-06]. → pages
- [213] M.-J. Sirois, M. Côté, and S. Pelet. The burden of hospitalized hip fractures: Patterns of admissions in a level i trauma center over 20 years. *The Journal of Trauma: Injury, Infection, and Critical Care*, 66(5):1402–1410, May 2009. ISSN 0022-5282. doi:10.1097/TA.0b013e31818cc1cc. [Accessed on 2011-02-02]. → pages
- [214] J. G. Skedros and S. L. Baucom. Mathematical analysis of trabecular ‘trajectories’ in apparent trajectory structures: The unfortunate historical emphasis on the human proximal femur. *Journal of Theoretical Biology*, 244(1):15–45, Jan. 2007. ISSN 0022-5193. doi:10.1016/j.jtbi.2006.06.029. URL <http://www.sciencedirect.com/science/article/pii/S0022519306002773>. [Accessed on 2012-02-25]. → pages
- [215] J.-A. Smith, J. A. Vento, R. P. Spencer, and B. E. Tendler. Aortic calcification contributing to bone densitometry measurement. *Journal of Clinical Densitometry*, 2 (2):181–183, 1999. ISSN 1094-6950. doi:10.1385/JCD:2:2:181. URL <http://www.sciencedirect.com/science/article/pii/S1094695006600966>. [Accessed on 2013-08-09]. → pages
- [216] T. S. Smith, B. K. Bay, and M. M. Rashid. Digital volume correlation including rotational degrees of freedom during minimization. *Experimental Mechanics*, 42(3): 272–278, Sept. 2002. ISSN 0014-4851. doi:10.1007/BF02410982. URL <http://www.springerlink.com/content/e8804wl826518920/>. [Accessed on 2010-12-08]. → pages
- [217] X. Song, N. Shi, E. Badamgarav, J. Kallich, H. Varker, G. Lenhart, and J. R. Curtis. Cost burden of second fracture in the US health system. *Bone*, 48:828–836, Apr. 2011. ISSN 8756-3282. doi:10.1016/j.bone.2010.12.021. URL <http://www.sciencedirect.com/science/article/B6T4Y-51W058M-1/2/66064e3863ec02e19313e580f66cb9a0>. [Accessed on 2011-02-02]. → pages
- [218] M. M. Sran, K. M. Khan, K. Keiver, J. B. Chew, H. A. McKay, and T. R. Oxland. Accuracy of DXA scanning of the thoracic spine: cadaveric studies comparing BMC, areal BMD and geometric estimates of volumetric BMD against ash weight and CT measures of bone volume. *European Spine Journal*, 14(10):971–976, Dec. 2004. ISSN 0940-6719, 1432-0932. doi:10.1007/s00586-004-0836-8. URL <http://www.springerlink.com/index/10.1007/s00586-004-0836-8>. [Accessed on 2013-02-12]. → pages
- [219] B. Srinivasan, D. L. Kopperdahl, S. Amin, E. J. Atkinson, J. Camp, R. A. Robb, B. L. Riggs, E. S. Orwoll, L. J. Melton, T. M. Keaveny, and S. Khosla. Relationship of femoral neck areal bone mineral density to volumetric bone mineral density, bone size, and femoral strength in men and women. *Osteoporosis International*, 23(1):155–162,

- Nov. 2011. ISSN 0937-941X, 1433-2965. doi:10.1007/s00198-011-1822-8. URL <http://www.springerlink.com/index/10.1007/s00198-011-1822-8>. [Accessed on 2012-01-11]. → pages
- [220] K. L. Stone, D. G. Seeley, L.-Y. Lui, J. A. Cauley, K. Ensrud, W. S. Browner, M. C. Nevitt, and S. R. Cummings. BMD at multiple sites and risk of fracture of multiple types: long-term results from the study of osteoporotic fractures. *Journal of Bone and Mineral Research*, 18(11):1947–1954, Nov. 2003. ISSN 0884-0431. doi:10.1359/jbmr.2003.18.11.1947. URL <http://www.ncbi.nlm.nih.gov/pubmed/14606506>. [Accessed on 2011-02-03]. → pages
- [221] E. G. Sutter, S. C. Mears, and S. M. Belkoff. A biomechanical evaluation of femoroplasty under simulated fall conditions. *Journal of Orthopaedic Trauma*, 24(2):95–99, Feb. 2010. ISSN 1531-2291. doi:10.1097/BOT.0b013e3181b5c0c6. URL <http://www.ncbi.nlm.nih.gov/pubmed/20101133>. [Accessed on 2011-12-02]. → pages
- [222] P. Sztefek, M. Vanleene, R. Olsson, R. Collinson, A. A. Pitsillides, and S. Shefelbine. Using digital image correlation to determine bone surface strains during loading and after adaptation of the mouse tibia. *Journal of Biomechanics*, 43(4):599–605, Mar. 2010. ISSN 1873-2380. doi:10.1016/j.jbiomech.2009.10.042. URL <http://www.ncbi.nlm.nih.gov/pubmed/20005517>. [Accessed on 2010-10-27]. → pages
- [223] F. Taddei, S. Martelli, B. Reggiani, L. Cristofolini, and M. Viceconti. Finite-element modeling of bones from CT data: Sensitivity to geometry and material uncertainties. *IEEE Transactions on Biomedical Engineering*, 53(11):2194–2200, 2006. ISSN 0018-9294. doi:10.1109/TBME.2006.879473. → pages
- [224] E. Tanck, A. Bakker, S. Kregting, B. Cornelissen, J. Klein-Nulend, and B. Van Rietbergen. Predictive value of femoral head heterogeneity for fracture risk. *Bone*, 44(4):590–595, Apr. 2009. ISSN 87563282. doi:10.1016/j.bone.2008.12.022. URL <http://linkinghub.elsevier.com/retrieve/pii/S8756328208009848>. [Accessed on 2013-08-07]. → pages
- [225] J.-E. Tarride, R. B. Hopkins, W. D. Leslie, S. Morin, J. D. Adachi, A. Papaioannou, L. Bessette, J. P. Brown, and R. Goeree. The burden of illness of osteoporosis in Canada. *Osteoporosis International*, 23(11):2591–2600, Nov. 2012. ISSN 0937-941X. doi:10.1007/s00198-012-1931-z. URL <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3483095/>. [Accessed on 2013-07-25]. → pages
- [226] W. Tong. An evaluation of digital image correlation criteria for strain mapping applications. *Strain*, 41(4):167–175, Nov. 2005. ISSN 00392103. doi:10.1111/j.1475-1305.2005.00227.x. [Accessed on 2010-11-09]. → pages
- [227] P. A. Toogood, A. Skalak, and D. R. Cooperman. Proximal femoral anatomy in the normal human population. *Clinical Orthopaedics and Related Research*, 467(4):876–885, Apr. 2009. ISSN 1528-1132. doi:10.1007/s11999-008-0473-3. URL <http://www.ncbi.nlm.nih.gov/pubmed/18758876>. [Accessed on 2010-10-24]. → pages

- [228] D. J. Torgerson and P. Dolan. Letter: The cost of treating osteoporotic fractures in the united kingdom female population. *Osteoporosis International*, 11(6):551–552, July 2000. ISSN 0937-941X, 1433-2965. doi:10.1007/s001980070101. URL <http://link.springer.com/article/10.1007/s001980070101>. [Accessed on 2013-07-25]. → pages
- [229] C. Turner, S. Cowin, J. Rho, R. Ashman, and J. Rice. The fabric dependence of the orthotropic elastic constants of cancellous bone. *Journal of Biomechanics*, 23(6): 549–561, 1990. → pages
- [230] C. H. Turner. The biomechanics of hip fracture. *The Lancet*, 366:98–99, July 2005. doi:10.1016/S0140-6736(05)66842-0. → pages
- [231] J. P. W. van den Bergh, T. A. C. M. van Geel, W. F. Lems, and P. P. Geusens. Assessment of individual fracture risk: FRAX and beyond. *Current Osteoporosis Reports*, 8(3):131–137, Sept. 2010. ISSN 1544-2241. doi:10.1007/s11914-010-0022-3. URL <http://www.ncbi.nlm.nih.gov/pubmed/20563901>. [Accessed on 2011-09-02]. → pages
- [232] A. J. van den Kroonenberg, W. C. Hayes, and T. A. McMahon. Dynamic models for sideways falls from standing height. *Journal of Biomechanical Engineering*, 117(3): 309–318, 1995. doi:10.1115/1.2794186. URL <http://link.aip.org/link/?JBY/117/309/1>. [Accessed on 2010-11-22]. → pages
- [233] A. J. van den Kroonenberg, W. C. Hayes, and T. A. McMahon. Hip impact velocities and body configurations for voluntary falls from standing height. *Journal of Biomechanics*, 29(6):807–811, June 1996. ISSN 0021-9290. URL <http://www.ncbi.nlm.nih.gov/pubmed/9147979>. [Accessed on 2010-10-27]. → pages
- [234] N. van Schoor, A. van der Veen, L. Schaap, T. Smit, and P. Lips. Biomechanical comparison of hard and soft hip protectors, and the influence of soft tissue. *Bone*, 39(2):401–407, Aug. 2006. ISSN 8756-3282. doi:16/j.bone.2006.01.156. URL <http://www.sciencedirect.com/science/article/pii/S8756328206002572>. [Accessed on 2011-09-02]. → pages
- [235] E. Verhulp, B. van Rietbergen, and R. Huiskes. Comparison of micro-level and continuum-level voxel models of the proximal femur. *Journal of Biomechanics*, 39(16): 2951–2957, 2006. ISSN 0021-9290. doi:10.1016/j.jbiomech.2005.10.027. URL <http://www.sciencedirect.com/science/article/B6T82-4HTBM51-1/2/997ef2f3a5ce0a0374514ee7b82e530d>. [Accessed on 2010-11-04]. → pages
- [236] E. Verhulp, B. van Rietbergen, and R. Huiskes. Load distribution in the healthy and osteoporotic human proximal femur during a fall to the side. *Bone*, 42(1):30–35, Jan. 2008. ISSN 8756-3282. doi:10.1016/j.bone.2007.08.039. URL <http://www.sciencedirect.com/science/article/B6T4Y-4PKPGX2-3/2/716ca39b04011153b93d8a4d1e1fb238>. [Accessed on 2010-11-08]. → pages

- [237] E. Verhulp, B. Van Rietbergen, R. Müller, and R. Huiskes. Micro-finite element simulation of trabecular-bone post-yield behaviour – effects of material model, element size and type. *Computer Methods in Biomechanics and Biomedical Engineering*, 11(4):389, 2008. ISSN 1025-5842. doi:10.1080/10255840701848756. URL <http://www.informaworld.com/10.1080/10255840701848756>. [Accessed on 2010-11-08]. → pages
- [238] D. C. Viano, I. V. Lau, C. Asbury, A. I. King, and P. Begeman. Biomechanics of the human chest, abdomen, and pelvis in lateral impact. *Accident Analysis and Prevention*, 21(6):553–574, Dec. 1989. ISSN 0001-4575. doi:16/0001-4575(89)90070-5. URL <http://www.sciencedirect.com/science/article/pii/0001457589900705>. [Accessed on 2011-07-27]. → pages
- [239] M. Viceconti, A. Toni, and A. Giunti. *Experimental Mechanics: Technology Transfer Between High Tech Engineering and Biomechanics (Clinical Aspects of Biomedicine)*. Elsevier Science, Amsterdam, 1992. ISBN 0444895809. → pages
- [240] M. Viceconti, M. Davinelli, F. Taddei, and A. Cappello. Automatic generation of accurate subject-specific bone finite element models to be used in clinical studies. *Journal of Biomechanics*, 37(10):1597–605, 2004. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=15336935. [Accessed on 2014-01-11]. → pages
- [241] Q. Wang, J. W. Teo, A. Ghasem-Zadeh, and E. Seeman. Women and men with hip fractures have a longer femoral neck moment arm and greater impact load in a sideways fall. *Osteoporosis International*, 20(7):1151–1156, 2009. ISSN 0937-941X, 1433-2965. doi:10.1007/s00198-008-0768-y. URL <http://link.springer.com/10.1007/s00198-008-0768-y>. [Accessed on 2013-08-07]. → pages
- [242] T. Weber, K. Yang, R. Woo, and R. J. Fitzgerald. Proximal femur strength: Correlation of the rate of loading and bone mineral density. *ASME Advances in Bioengineering*, 22:111–114, 1992. → pages
- [243] WHO Study Group. *Assessment of fracture risk and its application to screening for postmenopausal osteoporosis*. WHO Technical Report Series. World Health Organization, Geneva, 1994. ISBN 9241208430 9789241208437. → pages
- [244] M. E. Wiktorowicz, R. Goeree, A. Papaioannou, J. D. Adachi, and E. Papadimitropoulos. Economic implications of hip fracture: Health service use, institutional care and cost in canada. *Osteoporosis International*, 12(4):271–278, May 2001. ISSN 0937-941X. doi:10.1007/s001980170116. URL <http://www.springerlink.com/content/62r75ac3tk39132v/>. [Accessed on 2011-02-02]. → pages
- [245] J. L. Williams and J. L. Lewis. Properties and an anisotropic model of cancellous bone from the proximal tibial epiphysis. *Journal of Biomechanical Engineering*, 104(1):

50–56, Feb. 1982. doi:10.1115/1.3138303. URL <http://dx.doi.org/10.1115/1.3138303>. [Accessed on 2012-01-18]. → pages

- [246] J. Wolff, P. Maquet, and R. Furlong. Law of Bone Remodelling. Springer-Verlag, 1986. ISBN 354016281X, 9783540162810. URL <http://books.google.ca/books?id=XolsAAAAMAAJ>. [Accessed on 2014-01-11]. → pages
- [247] T. M. Wright and W. C. Hayes. Tensile testing of bone over a wide range of strain rates: effects of strain rate, microstructure and density. Medical and Biological Engineering, 14(6):671–80, 1976. URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&dopt=Citation&list_uids=994579. [Accessed on 2014-01-11]. → pages
- [248] S. Yaofeng and J. H. Pang. Study of optimal subset size in digital image correlation of speckle pattern images. Optics and Lasers in Engineering, 45(9):967–974, Sept. 2007. ISSN 0143-8166. doi:10.1016/j.optlaseng.2007.01.012. URL <http://www.sciencedirect.com/science/article/pii/S0143816607000425>. [Accessed on 2013-06-07]. → pages
- [249] M. Yasuyuki, U. Masakazu, T. Mitsugu, M. Yasuyuki, A. Kazuo, and K. Kiyoshi. Relationship between the load-displacement curve and deformation distribution in porcine mandibular periodontium. Journal of Biomechanical Science and Engineering, 4(3):336–344, 2009. doi:10.1299/jbse.4.336. URL <http://dx.doi.org/10.1299/jbse.4.336>. [Accessed on 2014-01-11]. → pages
- [250] P. Ziopoulos and J. Currey. Changes in the stiffness, strength, and toughness of human cortical bone with age. Bone, 22(1):57–66, Jan. 1998. ISSN 8756-3282. doi:10.1016/S8756-3282(97)00228-7. URL <http://www.sciencedirect.com/science/article/pii/S8756328297002287>. [Accessed on 2012-12-22]. → pages
- [251] P. Ziopoulos, U. Hansen, and J. D. Currey. Microcracking damage and the fracture process in relation to strain rate in human cortical bone tensile failure. Journal of Biomechanics, 41(14):2932–2939, Oct. 2008. ISSN 0021-9290. doi:10.1016/j.jbiomech.2008.07.025. URL <http://www.sciencedirect.com/science/article/pii/S0021929008003977>. [Accessed on 2011-09-23]. → pages

Appendix A

Supporting Materials

A.1 Increased aBMD due to larger bone size

This section explains the effect of larger bone size on aBMD discussed in §1.3.1.

Areal bone mineral density (aBMD) can be artificially increased if the bone being scanned in a dual-energy x-ray absorptiometry (DXA) scanner is larger than assumed (i.e., larger than the reference population). For example, consider a vertebral body to be cylindrical in shape. If the diameter and height of the reference population vertebra are 2 cm each, the volume would be 12.56 cm^3 , and the projected area would be 4 cm^2 . Now consider that the reference population BMD is 2 g/cm^3 , so the total BMC is 12.56 g , and the aBMD would be $(12.56 \text{ g}/4 \text{ cm}^2) = 3.14 \text{ g/cm}^2$. If we now consider a large patient whose vertebra has a diameter and height of 3 cm each, they would have a total volume of 21.2 cm^3 , and a projected area of 9 cm^2 . If this patient has the same BMD of 2 g/cm^3 , they would have a total BMC of 42.4 g , and an aBMD of 4.71 g/cm^2 , indicating that the larger patient has 50% more dense bone, even though their BMD was the same.

A.2 Femoral neck internal rotation justification

Femoral orientation in the fall protocol defines the position of the femoral neck as 15° internal rotation. This value can be shown to be approximately correct using data from Toogood et al.

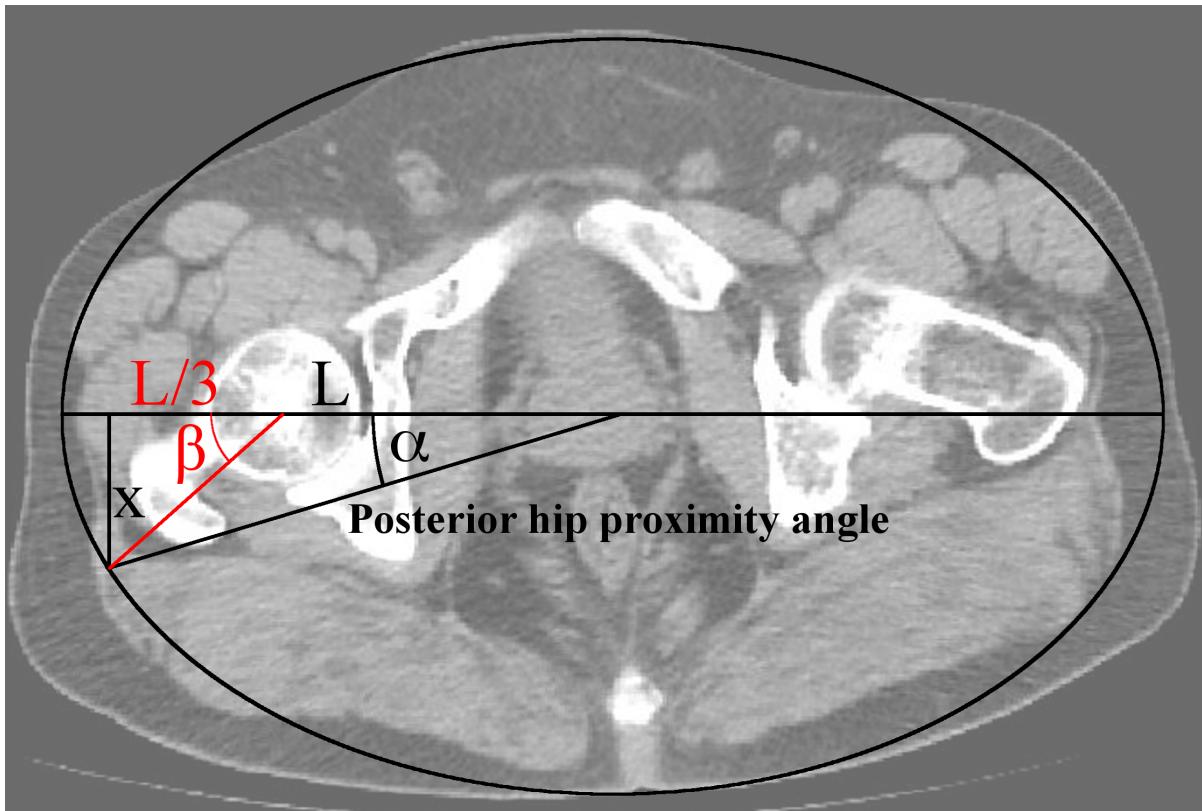


Figure A.1: The hip proximity angle is referenced to the centre of the pelvis. Changing the reference to the centre of the femoral head roughly doubles the angle.
Graphic adapted from Inversitus [101], permission not required.

[227] and Feldman and Robinovitch [68]. Feldman and Robinovitch [68] reported a “hip proximity angle” of approximately 8° posterior. This angle is referenced to the centre of the pelvis, and is a different angle when referenced to the centre of the femoral head (Figure A.1).

Assuming that the femoral head lies approximately one third of the way from the centre of the pelvis to the surface of the hip, a hip proximity angle of 8° gives an angle of approximately 22.9° degrees when referenced to the femoral head (Equation A.1).

$$\begin{aligned}
\tan(\alpha) &= \frac{x}{L} \\
\Leftrightarrow x &= L \cdot \tan(\alpha) \\
\tan \beta &= \frac{x}{L/3} \\
\Leftrightarrow \tan \beta &= \frac{L \cdot \tan(\alpha)}{L/3} \\
\therefore \beta &= \arctan(3 \cdot \tan(\alpha)) \\
\Leftrightarrow \beta &= \arctan(3 \cdot \tan(8^\circ)) = 22.9^\circ
\end{aligned} \tag{A.1}$$

The impact angle of 22.9° internal rotation about the femoral head is independent of femoral twist, which tends to decrease the value of the impact angle. The femoral twist in the normal population is 9.73° [227], so the resulting average impact angle would be 13.1° . This is not the value of 15° used in the research, but given the impact angle standard deviation of 15° , and femoral twist angle standard deviation of 9.28° , the standard orientation is well within the expected normal variation.

A.3 Calculation of Mass 1 initial velocity

The velocity of Mass 1 in the idealized model discussed in §4.4 can be calculated as shown below.

$$\begin{aligned}
E_{start} &= E_{finish} \\
E_{start} &= \frac{1}{2} \cdot m_{st} \cdot V_{st,start}^2 + \frac{1}{2} \cdot m_{body} \cdot V_{body,start}^2 \\
E_{finish} &= \frac{1}{2} \cdot m_{st} \cdot V^2 + \frac{1}{2} \cdot m_{body} \cdot V^2 \\
m_{st} \cdot V_{st,start}^2 + m_{body} \cdot V_{body,start}^2 &= (\cdot m_{st} + \cdot m_{body}) \cdot V^2 \\
\therefore V &= \sqrt{\frac{m_{st} \cdot V_{st,start}^2 + m_{body} \cdot V_{body,start}^2}{m_{st} + m_{body}}}
\end{aligned} \tag{A.2}$$

Where m_{st} is the stationary mass of the top plate (1 kg) and top $\frac{1}{3}$ of the spring (1.17 kg), and V_{st} is the initial velocity of the stationary mass, namely, zero. Filling in the missing numbers

and solving.

$$\begin{aligned} V &= \sqrt{\frac{2.17 \cdot 0^2 + 32 \cdot 3^2}{2.17 + 32}} \\ &= 2.90 [m/s] \end{aligned} \quad (\text{A.3})$$

A.4 Equation related to the response of an isolated ideal femur under constant displacement rate loading

The response of the idealized model of the loading performed by Courtney et al. [46], discussed in §4.4.1, can be calculated using a fixed velocity, and selecting either a maximum displacement or maximum force. The use of either maximum displacement or force has an affect on the shape of the displacement rate vs. relative stiffness graph, so explicit selection of this parameter is important.

$$\begin{aligned} F &= ks + c\dot{s} \\ k_{total} = \frac{F}{s} &= \frac{ks + c\dot{s}}{s} \end{aligned}$$

Calculating the stiffness up to a set displacement gives a linear relationship with damping coefficient (c) (Equation A.4). Note that when c is high the force at even small displacements can be very high, dominated by the damping portion of the relationship. This makes calculation of the total stiffness up to a given force, rather than given displacement, preferable.

$$\begin{aligned} k_{Total} = \frac{F_{max}}{s_{max}} &= \frac{ks_{max} + c\dot{s}}{s_{max}} \\ \therefore k_{Total} &= k + \frac{c\dot{s}}{s_{max}} \end{aligned} \quad (\text{A.4})$$

Calculating the stiffness to a set force gives an inverse relationship with $(1 - c)$ (Equation A.5).

$$\begin{aligned}
k_{Total} &= \frac{F_{max}}{s_{max}} = \frac{ks_{max} + c\dot{s}}{s_{max}} \\
s_{max} &= \frac{F_{max} - c\dot{s}}{\frac{k}{k}} \\
\therefore k_{Total} &= \frac{k\left(\frac{F_{max} - c\dot{s}}{k}\right) + c\dot{s}}{\frac{F_{max} - c\dot{s}}{k}} = \frac{F_{max}}{\frac{F_{max} - c\dot{s}}{k}} \\
k_{Total} &= \frac{k}{1 - c\frac{\dot{s}}{F_{max}}}
\end{aligned} \tag{A.5}$$

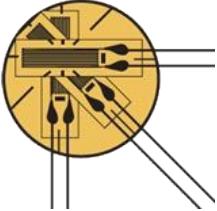
A.5 Testing log sheets

The following four pages contain the test day log and checklist used to ensure data integrity and communicate with FE modellers.

Experimenters		Date	
Specimen	ID	Side	
Potting			
Exposed Length			
Strain Gauges	Number	Locations	

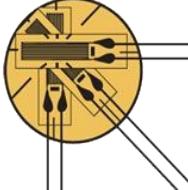
Specimen	ID	Side	Date

INSTRON TESTING

Wavemaker File Name							
Wavemaker Diagram							
Load	P	I	D	Position	P	I	D
Outputs							
Load	N/V	Channel	Position	mm/V	Channel	Trigger	Channel
Strain Gauge							
							
Cameras							
8244		8243		4622		4623	
FPS		FPS		FPS		FPS	
Resolution		Resolution		Resolution		Resolution	
Exposure		Exposure		Exposure		Exposure	
EDR		EDR		EDR		EDR	
Post Trigg		Post Trigg		Post Trigg		Post Trigg	
Timing	Intern/Extern	Timing	Intern/Extern	Timing	Intern/Extern	Timing	Intern/Extern
Lens		Lens		Lens		Lens	
Target		Target		Target		Target	
Extender		Extender		Extender		Extender	
Focus		Focus		Focus		Focus	
f-Stop		f-Stop		f-Stop		f-Stop	
Focal Lngth		Focal Ingth		Focal Ingth		Focal Ingth	
Notes - Max Load:				Fracture: YES NO			

Specimen	ID	Side	Date

DROP TOWER

Digitizing Points								
File Name	Head Cup	Neck/Trochanter	Trochanter Cup	Shaft	Pivot	Impactor Axis	Grnd	Strain Gauge
Strain Gauge								
								
DAQ Settings								
Ch 0	Ch 1	Ch 2	Ch 3	Ch 4	Ch 5	Ch 6	Ch 7	
1000 (A,D)	1000 (A,D)	500 (A,C)	250 (B,C)	250 (B,C)	200 (A,B)	250 (B,C)	1 (D,A)	
10 kHz (-3dB)	10 kHz (-3dB)	10 kHz (-3dB)	10 kHz (-3dB)	10 kHz (-3dB)	10 kHz (-3dB)	10 kHz (-3dB)	10 kHz (-3dB)	
Fx-6x	Fy-6x	Fz-6x	Mx-6x	My-6x	Mz-6x	Fz-1x	Trigger	
Physical Parameters								
Target Eng				Target Velo				
Drop Mass				Height				
Cameras								
8244		8243		4622		4623		
FPS		FPS		FPS		FPS		
Resolution		Resolution		Resolution		Resolution		
Exposure		Exposure		Exposure		Exposure		
EDR		EDR		EDR		EDR		
Post Trigg		Post Trigg		Post Trigg		Post Trigg		
Timing	Intern/Extern	Timing	Intern/Extern	Timing	Intern/Extern	Timing	Intern/Extern	
Lens		Lens		Lens		Lens		
Target		Target		Target		Target		
Extender		Extender		Extender		Extender		
Focus		Focus		Focus		Focus		
f-Stop		f-Stop		f-Stop		f-Stop		
Focal Lngth		Focal Ingth		Focal Ingth		Focal Ingth		
Notes - Max Load:					Fracture: YES NO			

Check Lists

INSTRON

Runup

Limits
Position Zeroed
Max Load Set
Lighting
Camera Views
Camera Triggers
DAQ File Name
DAQ Working

Pin

Lights ON
Cameras Pre-Trigger
DAQ ON
DAQ START
Test

Post

Lights OFF
DAQ STOP
Video Data (V12 V12 V9)
Specimen Secure
Calibration Data (V12 V12 V9)

DROP TOWER

Runup

Mass Attached
Lighting
Camera Views
Camera Triggers
DAQ File Name
DAQ Working

Pin

Lights ON
Camera Pre-Trigger
DAQ ON
DAQ START
Area Secure
Test

Post

Lights OFF
DAQ STOP
Video Data (V12 V12 V9)
Specimen Secure
Calibration Data (V12 V12 V9)

A.6 Equipment characterization experiments

This section details validation and characterization experiments conducted to ensure data integrity and understand measurement uncertainty. Each subsection will detail an experiment and the results of that experiment.

A.6.1 Drop tower fall velocity characterization

The drop tower is a highly constrained, four post design with four in-plane bearings used to direct the gantry. Due to the constraint of the bearings and posts, the drop tower does not act in free-fall during a drop, but instead must be characterized by a height vs. impact velocity relationship. Once this relationship is known the height can be used to anticipate the velocity at impact.

Method

The drop tower gantry was loaded with five different masses and dropped from three heights. Each mass and height was repeated a minimum of two times. Drops with 32.4 kg and 34 kg masses were repeated three times at 0.6 m as these were close to the test condition. The gantry was filmed using a high speed camera (V12.1, Vision Research, Wayne, NJ) at a frame rate of 6200 fps, and resolution of 1280x800 px (4.8 px/mm). The velocity was measured using the time for the drop tower gantry to travel the last inch before contact. Velocity vs. drop height was plotted for each mass, additionally, the data at each drop height were combined to generate and average curve for the 29.5-41 kg mass range.

Results

The drop tower displayed a linear relationship between drop height and velocity (Figure A.2). The mass had a much smaller influence than the height, however, the single mass below 30 kg displayed a lower velocity profile than the other masses. The average line was highly linear (Equation A.6, $R^2 = 0.9$, $p = 0.015$). The target velocity for the drops in the experiments was 3.0 m/s, and from the averaged equation, this lead to a drop height of 62 cm.

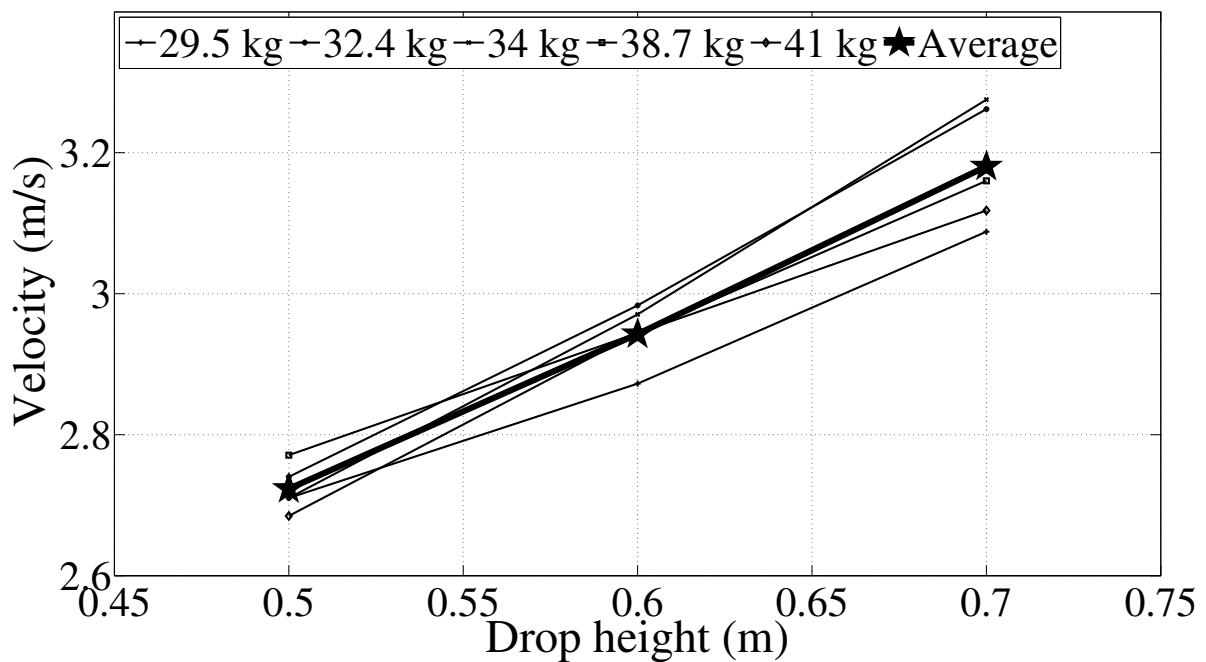


Figure A.2: The drop tower velocity was sensitive to height, and to some degree to mass loaded. At mass >30 kg, the variation due to mass was irregular. Graphic ©Seth Gilchrist, 2013.

$$Velocity = 2.288 \cdot Height + 1.575 \quad (\text{A.6})$$

A.6.2 Potting torsion resistance

The method used to pot the specimens was changed to make the potting easier and independent of the mounting apparatus. Previously, the specimens were potted directly into an aluminium tube that was part of the mounting apparatus. The new method places the specimens in a 2 inch, schedule 40 polyvinyl chloride (PVC) pipe which is then secured in the testing apparatus using six set screws. This test was to confirm that the set screws could resist the anticipated torque loads applied during testing. An instructional video for how to pot the specimens can be found at <http://youtu.be/qK967DE0y-Q>.

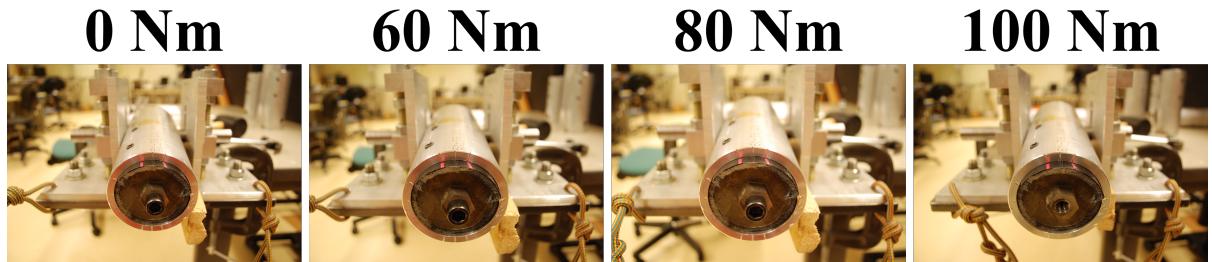


Figure A.3: A 3/8 inch bolt was potted in the PVC pipe that would be used for testing. A moment was applied to the bolt with a torque wrench and photographs were used to determine if any rotation had taken place. Graphic ©Seth Gilchrist, 2013.

Methods

The torque loads were estimated by mounting a plastic bone replica (v3 large composite femur, Sawbones, Vashon, WA) and taking a measurement from the superior, in the transverse plane from the most lateral point on the greater trochanter to the most medial point on the femoral head. This measurement was 1.2 cm. The maximum expected load for the femurs was 6000 N, leading to a maximum expected torsion of 72 Nm. A factor of safety of 1.3 was considered acceptable for this torque value, leading to a maximum testing torque of 93.6 Nm, which was rounded up to 100 Nm.

A piece of 1 inch, hexagonal bar with a 3/8 inch tapped hole was potted in the same PVC pipe that was going to be used for the testing. A torque wrench was used to apply torsions of 60, 80 and 100 Nm to a 3/8 inch steel bolt placed in the tapped hole in the hexagonal bar (Figure A.3). Photographs were taken after each torsion to check for rotation of the PVC pipe.

Results

Torsion values of 60, 80 and 100 Nm were successfully withstood by the set screws. Notably, the 3/8 inch bolt yielded at about 98 Nm and 100 Nm was attained only for a short time before the test had to be completed in order to avoid torquing the head off the bolt.

A.6.3 Materials testing machine compliance

Displacements were measured in the materials testing machine using the linear variable differential transformer (LVDT) incorporated in the machine, which measures the ram displacement. This value will consist of the displacement of the trochanter due to the compression of the bone, as well as the displacement of the trochanter due to translation of the head as the loading plates and bearing plates below the specimen compress. This test was conducted to measure the degree that the loading and bearing plates below the femoral head compress during testing.

Methods

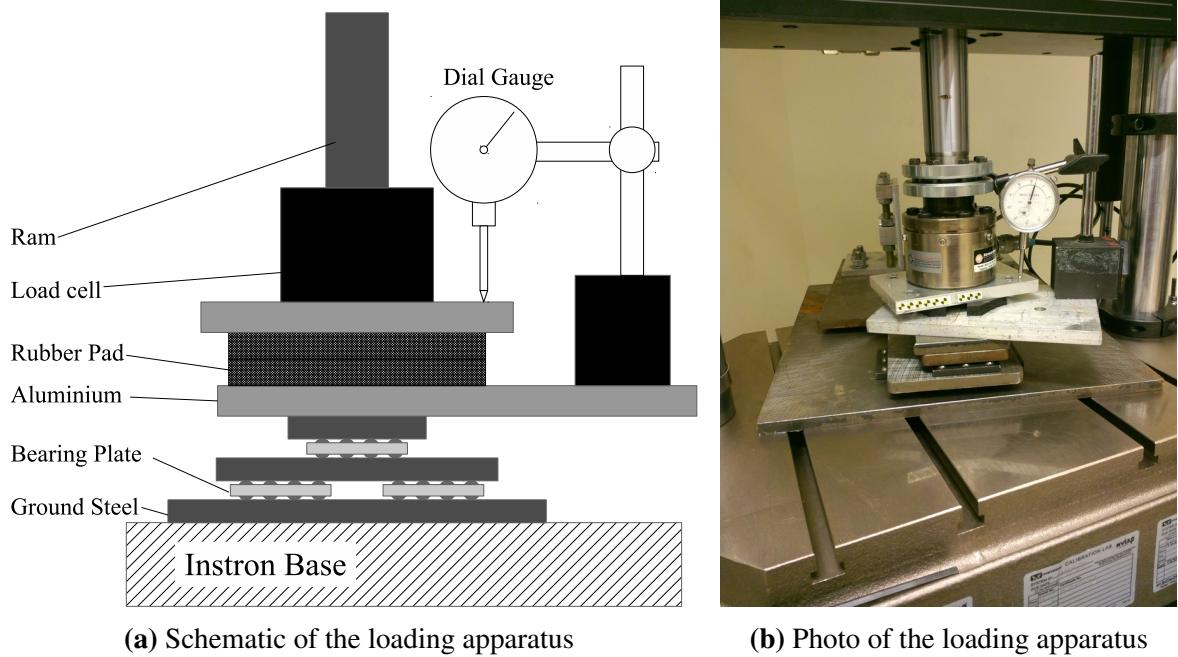
The test apparatus consists of three ground steel plates with bearing plates between them and two aluminium spacer plates on either side of the specimen (Figure A.4). Two pieces of rubber were placed in the space normally occupied by a specimen. These rubber mats were there to increase compliance and prevent machine overload. A dial gauge was used to measure the compression of the rubber. A laser level was used to ensure that the dial gauge probe was perpendicular to the testing machine base. The cross head of the materials testing machine (8874, Instron, Norwood, MA) was moved down manually in displacement control until a compressive force of 500 N was achieved. The force was allowed to settle until it was steady within 1 N for 60 s and the displacement was read. The compressive force was increased by 500 N and the process repeated until a compressive force of 2000 N had been achieved.

Results

The compression of the apparatus increased linearly with increasing force (Figure A.5). The stiffness of the bearing plates was found to be approximately 30 kN/mm, which is about 10-30x that observed for a typical specimen.

A.6.4 Drop tower compliance

The drop tower is a linear impact device consisting of four vertical rails and a horizontal gantry. It is constructed almost entirely out of $1\frac{5}{8}$ inch strut-lock channel. This design gives it great



(a) Schematic of the loading apparatus

(b) Photo of the loading apparatus

Figure A.4: In the materials testing machine compliance test a rubber mat was compressed in the test apparatus. A dial gauge was used to read the compression of the rubber which was subtracted from the displacement of the load cell to get the compression of the lower plates. Graphic ©Seth Gilchrist, 2013.

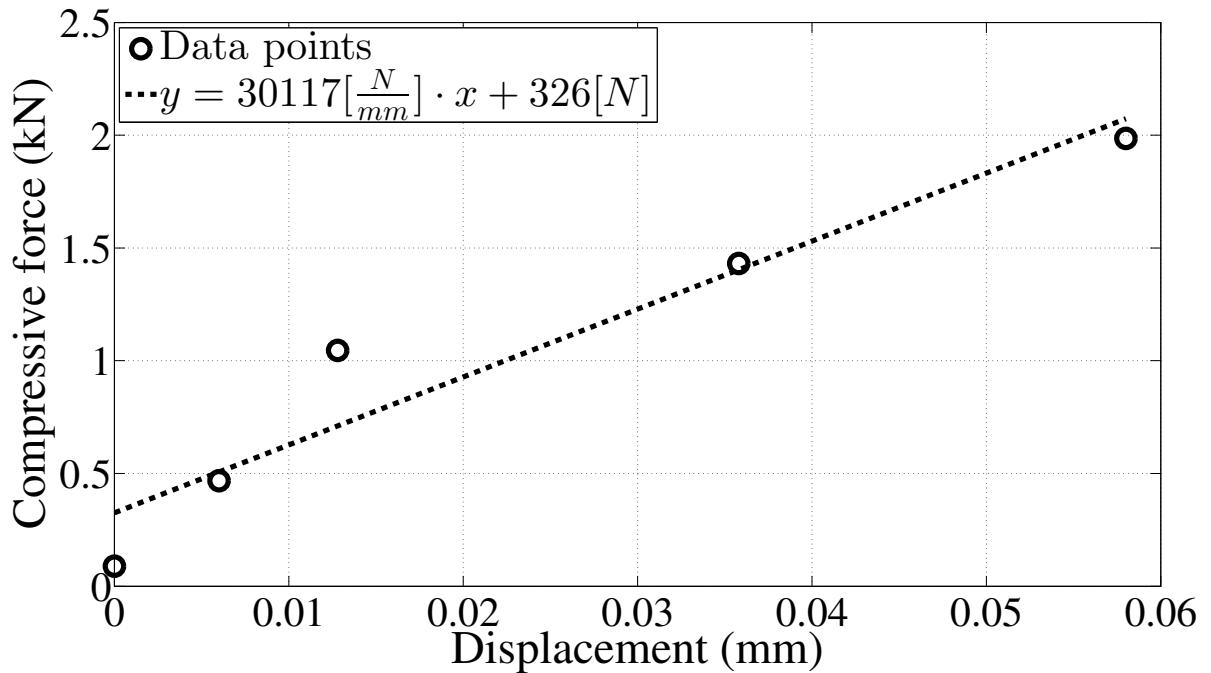


Figure A.5: The materials testing machine compliance tests showed a linear increase in displacement with force. The stiffness was calculated by fitting a line to the data. Graphic ©Seth Gilchrist, 2013.

flexibility to be reconfigured, but the open cross section and friction-lock joints mean that as a foundation material it is quite compliant. However using the static response is likely to overestimate the displacement in the initial seconds after an impact. For this reason, a dynamic model of the structure was used to estimate the displacement-time profile during the impact. This compliance could lead to potentially large displacements of the loading platform during impact. To perform these calculations, two things needed to be known, i) the compliance of the structure; ii) the mass mounted to the top of the structure.

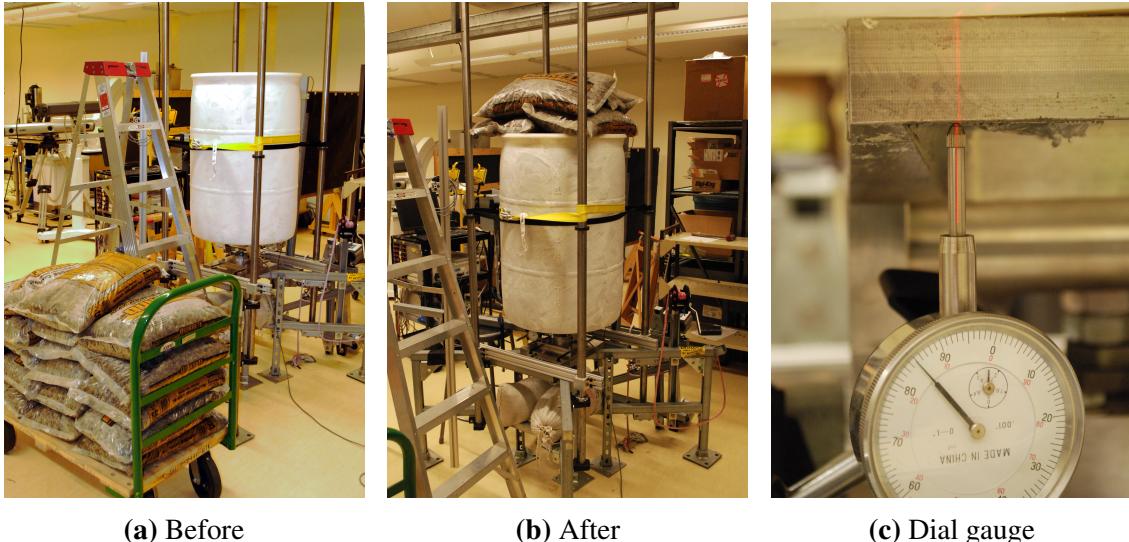
Methods

To measure the compliance of the drop tower, a significant load needed to be applied and the displacement of the loading platform needed to be measured. Unlike the materials testing machine, the drop tower does not have a way to apply a constant, high force to the platform, therefore, an external means of applying the force was used. A 50 gallon barrel was placed on top of the loading platform and filled with rocks to apply the load (Figure A.6). A dial gauge was fixed to a platform sitting on the floor and oriented to measure vertical displacement using a laser level. The drop tower's single axis load cell (LC 402, Omega Engineering, Stamford, CT) was used to measure compressive loads. Rocks were used from dead weight and were transferred into the barrel and load and displacement readings were taken at 500 N intervals. Measurements were taken during loading and unloading and the stiffness of the machine was determined by averaging the slopes of the loading and unloading curves.

After unloading the drop tower, the loading platform was removed along with the T-slot plate, load cell and all other mounting apparatus down to the strut-lock channel foundation. These items were weighed using a digital scale for items less than 20 kg and an analogue bathroom-style scale for those over 20 kg.

Results

The drop tower showed linear behaviour during loading and unloading (Figure A.7). The stiffness varied by 13% between loading and unloading. This change in stiffness is likely



(a) Before

(b) After

(c) Dial gauge

Figure A.6: The drop tower was loaded using a barrel filled with rocks. The displacement of the loading platform was measured using a dial gauge which was referenced to the ground. Graphic ©Seth Gilchrist, 2013.

Table A.1: Mass of all test apparatus used in the drop tower.

Item	Mass (kg)
Loading platen	23.18
Load cell	6.419
T-slot plate	21.36
Bearing plates	1.896
Head support plates	2.327
Mount plate	13.66
Bottom plate	14.54
Pivot rod	0.507
Total	83.89

due to the hysteresis of the friction joints at junctions in the strut-lock channel. The average stiffness for combined loading and unloading was 5637 N/mm.

A.6.5 Foam compliance

The foam used to pad the greater trochanter was bought from a local Evazote® supplier. Foams can be prone to variations in material properties because of the complexity of the manufacturing process. For this reason, an experiment was performed to measure the foam's compliance at different levels of compression.

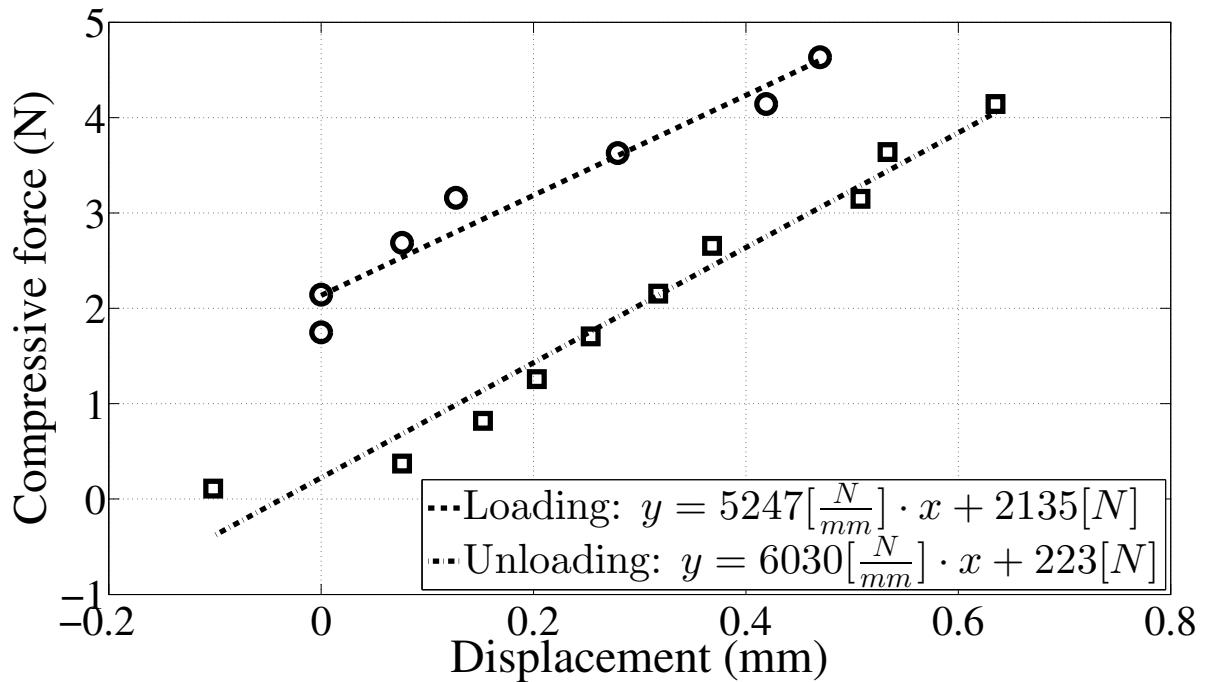


Figure A.7: The drop tower showed linear behaviour during both loading and unloading. Graphic ©Seth Gilchrist, 2013.

Methods

A 2 inch square piece of foam was cut from the same sheet of foam used in the fall simulation tests. The foam was loaded into the materials testing machine (8874, Instron, Norwood, MA) and compressed to 75%, 50% and 25% of its original height in displacement control. Compressive force measurements were taken at each level of compression. Stiffness was defined as the average stiffness to the compression level under inspection.

Results

The stiffness was seen to increase by a power-law relationship as compression increased (Table A.2). The stiffness was relatively constant between the 25% and 50% compression measurements, but as the foam pores collapsed a sharp increase in stiffness was observed.

Table A.2: Nominal stiffness and stress in the foam at various compression levels.

Compression(%)	Stress (kPa)	Stiffness (kN/m)
25	36	19.92
50	95	26.45
75	308	57.39

A.6.6 Drop tower position measurement verification

The positions of the greater trochanter and impact hammer were measured using image tracking techniques. This technique involved videoing the impact hammer and the greater trochanter during the impact event and using TEMA Automotive (v3.0, Image Systems, North Hollywood, CA) to measure the displacements. Frames from the video were loaded input TEMA and an image of a calibration target was used to correct the images for lens distortion. Tracking was done using a 1 cm square target on the impact hammer, and the junction of the bone and polymethylmethacrylate (PMMA) potting cap.

Methods

To determine the accuracy of this measurement, the impact hammer used in the fall simulation tests was secured in a materials testing machine (8874, Instron, Norwood, MA). A piece of surrogate bone with a similar colour to human bone was secured to the impact hammer using the same PMMA used in the fall simulation testing. The camera and lens used in the fall simulation tests were set up at the same distance, focal length, focus and frame rate as in the fall simulation tests. Lighting was also adjusted to be similar to the fall simulation tests. A dial gauge was arranged to measure the displacement of the impact hammer to confirm the displacement reported by the TEMA and Instron software. A second camera was arranged to record the dial gauge during the test and spot comparisons were made between the Instron output and dial gauge reading. A data acquisition system was used to record the output of the materials testing machine and the camera trigger signal to synchronize the readings. The materials testing machine was programmed to move the cross head according to the equation

Table A.3: Values and differences of the materials testing machine and dial gauge at various times

Time post trigger (ms)	Dial gauge (mm)	Instron (mm)	Difference (mm)
64.643	4.928	5.080	-0.152
114.771	9.906	10.025	-0.119
164.790	4.851	4.942	-0.091
189.528	1.397	1.389	0.008
253.435	3.175	3.334	-0.159
276.437	6.731	6.870	-0.139
292.821	8.763	8.918	-0.155
364.214	4.953	5.030	-0.077
381.140	2.413	2.440	-0.027
Average			-0.101
Standard Deviation			0.060

$5 \sin(10\pi t)$ mm, which gives a maximum velocity of 157 mm/s.

Results

The Instron output and dial gauge showed good correlation. Ten spot checks showed an average(SD) difference of -0.10(0.06) mm (Table A.3). The comparison of the TEMA with the Instron also showed good correlation, with an average(SD) error of -0.032(.043) mm (Figure A.8).

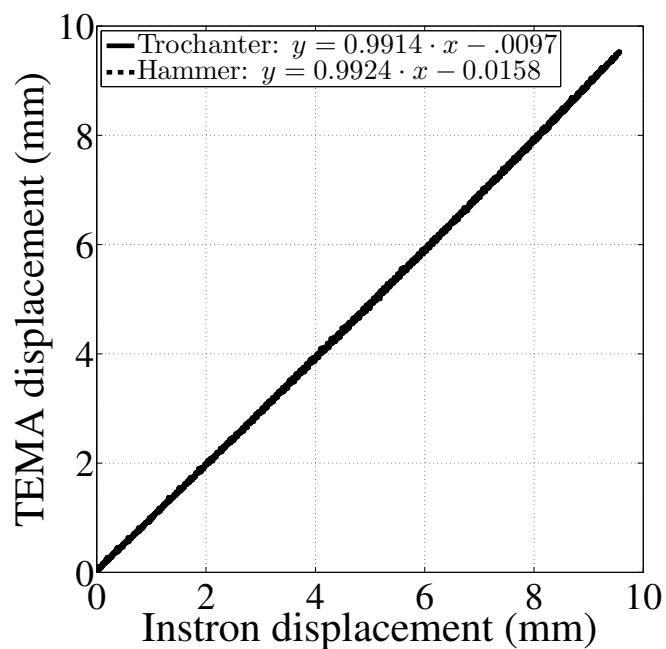


Figure A.8: The displacement measured by TEMA for both the trochanter and the impact hammer agreed well with each other and with the output of the materials testing machine. In this plot the lines are coincident and as such impossible to differentiate. Graphic ©Seth Gilchrist, 2013.

Appendix B

Additional plots

This appendix contains extra plots that may be useful, but were not included in the main document.

B.1 Quasi-stativ vs. fall simulator subfailure

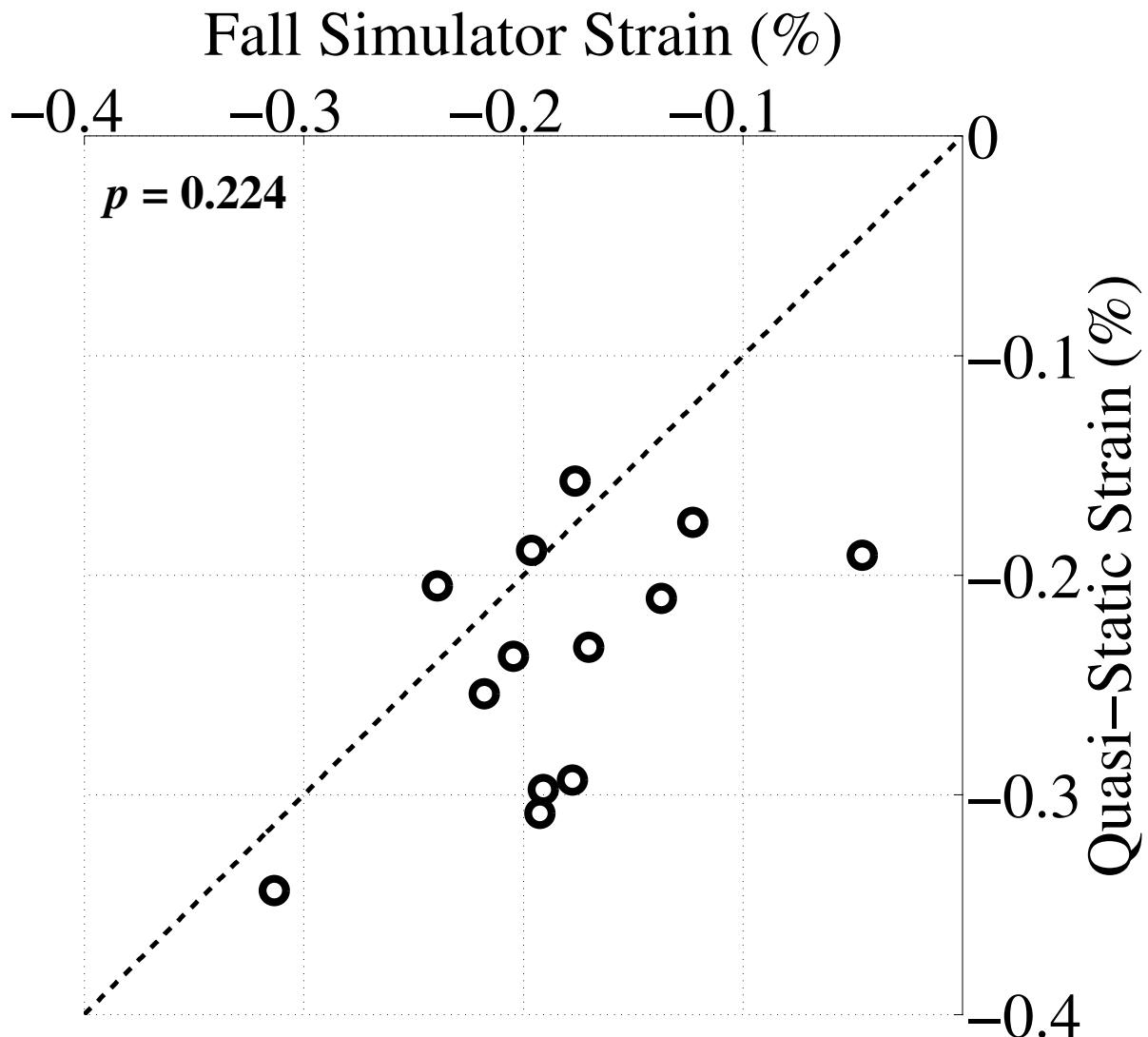


Figure B.1: Minimum principal strain at the location of the strain gauge in the fall simulator and in the quasi-static testing at the maximum force applied in the quasi-static test. Graphic ©Seth Gilchrist, 2013.

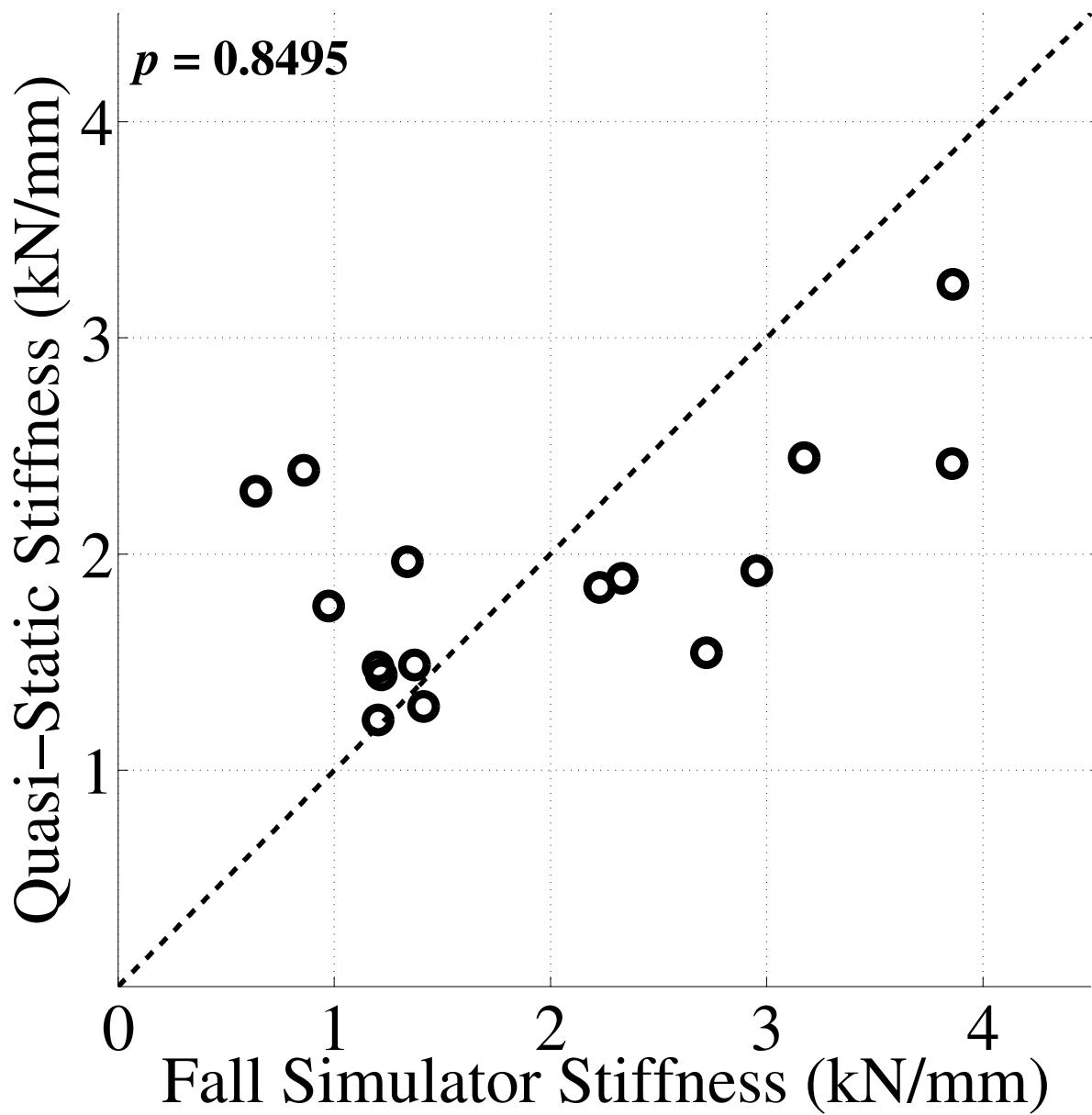


Figure B.2: Average stiffness in the quasi-static and fall simulator tests up to the maximum load applied in the quasi-static testing. Graphic ©Seth Gilchrist, 2013.

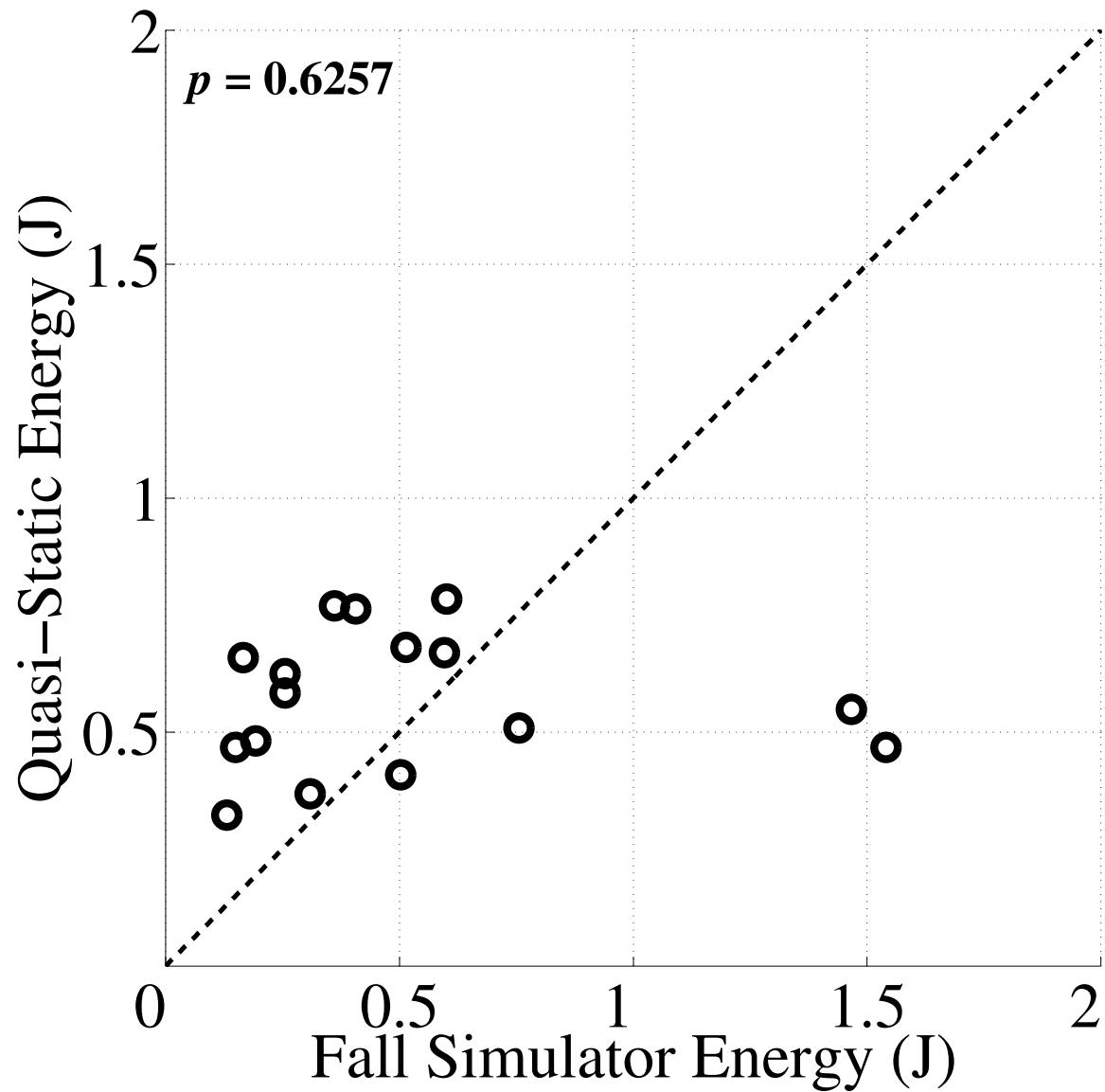


Figure B.3: Energy absorbed by the fall simulator and the quasi-static tests upto the maximum force applied in the quasi-static tests. Graphic ©Seth Gilchrist, 2013.

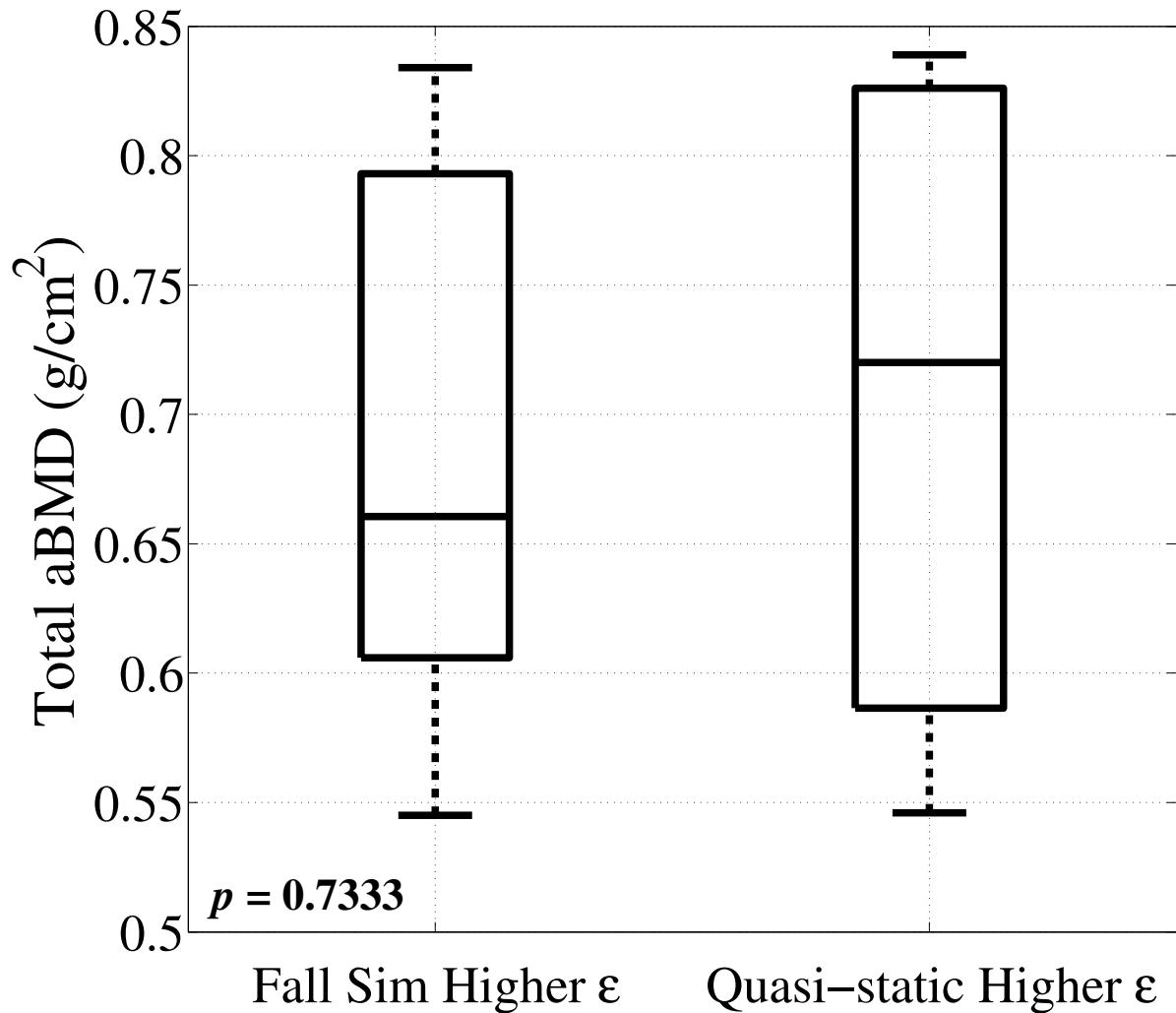


Figure B.4: aBMD of fall specimens grouped by their relative strains up to the maximum force applied in the QS condition, in the FS and QS conditions. Graphic ©Seth Gilchrist, 2013.

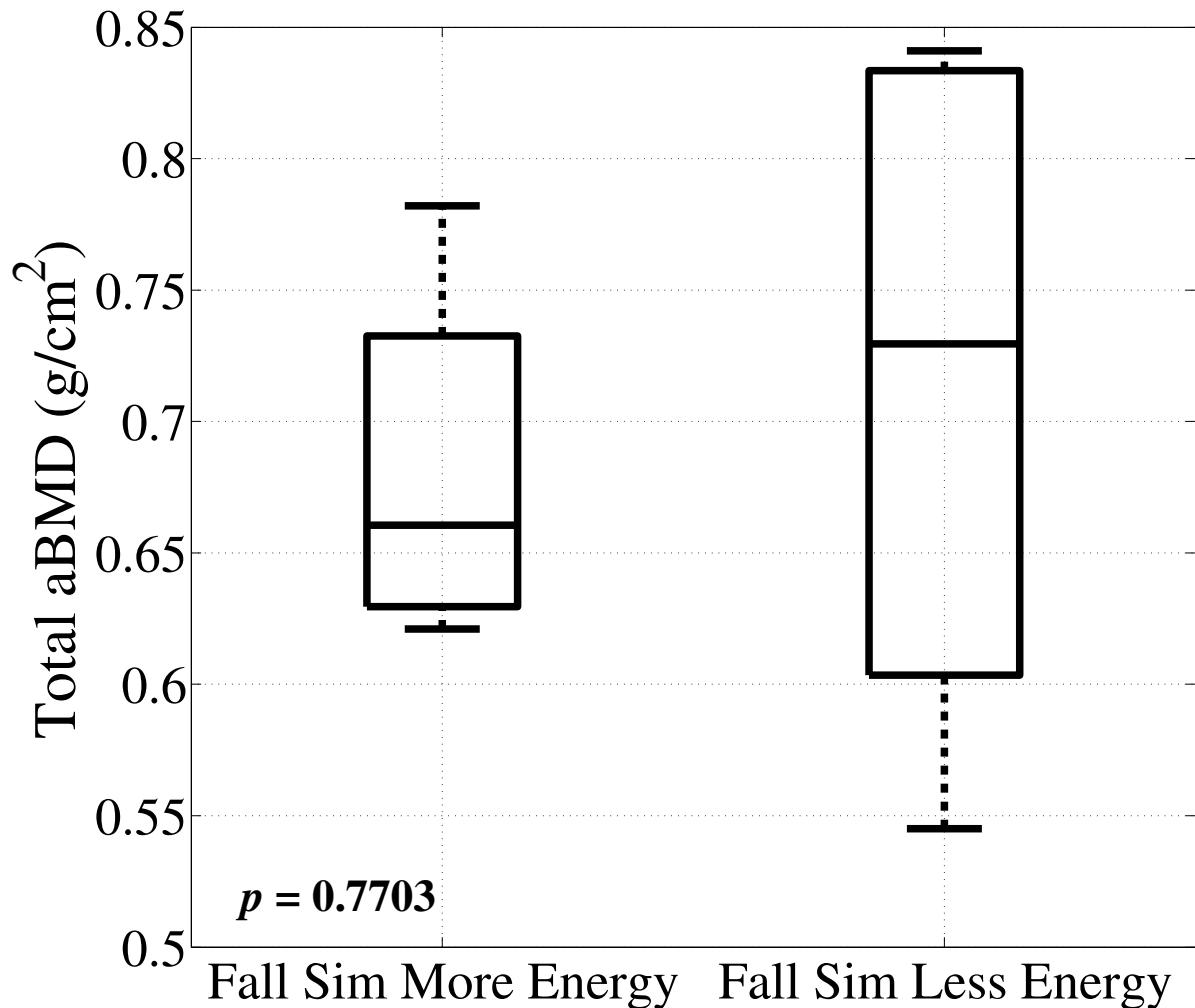


Figure B.5: aBMD of fall specimens grouped by the relative energy up to the maximum force applied in the QS condition, in the FS and QS conditions. Graphic ©Seth Gilchrist, 2013.

B.2 Constant rate vs. fall simulator

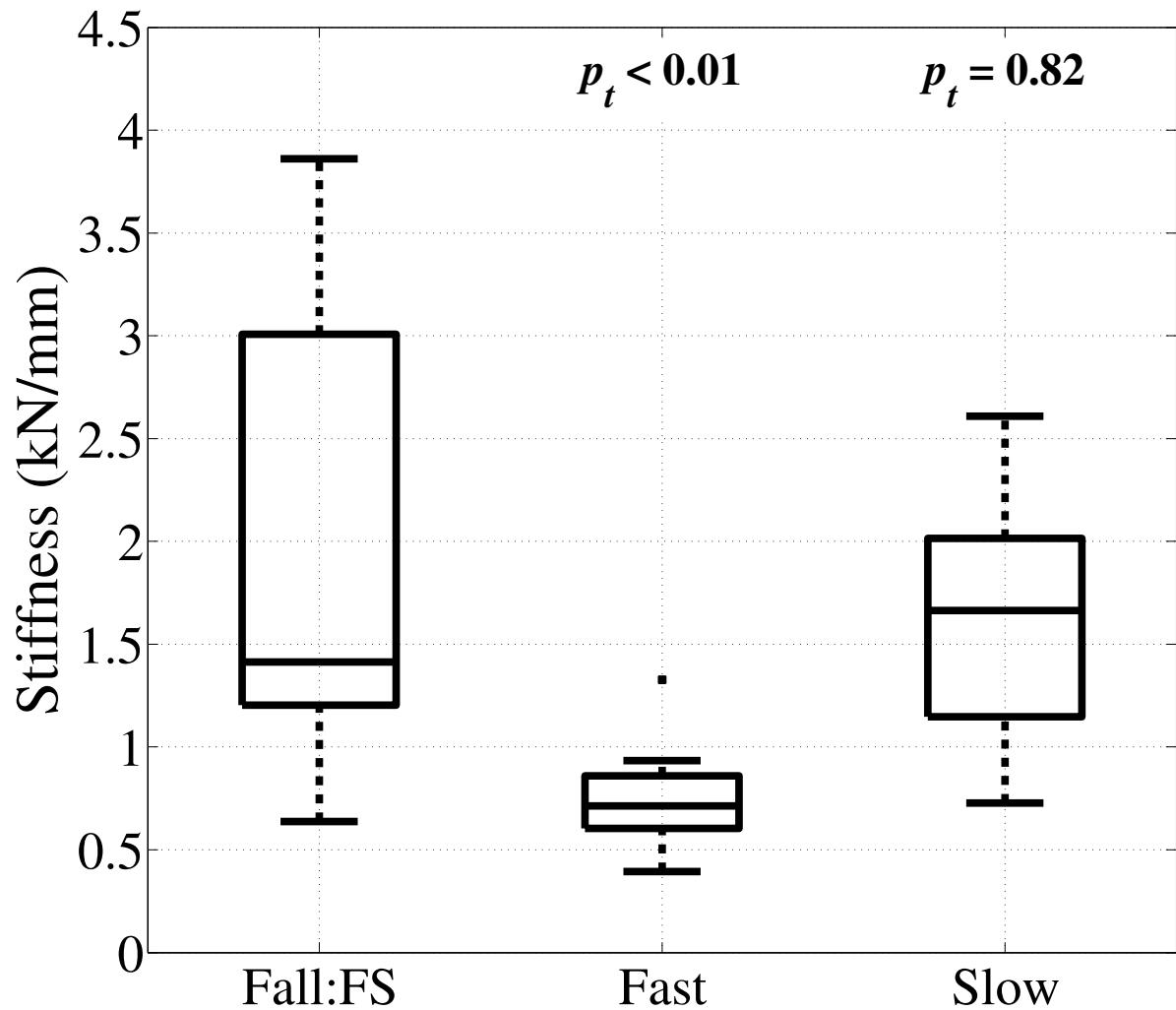


Figure B.6: Stiffness of the fall:FS, fast and slow groups in the failure tests. Graphic
©Seth Gilchrist, 2013.

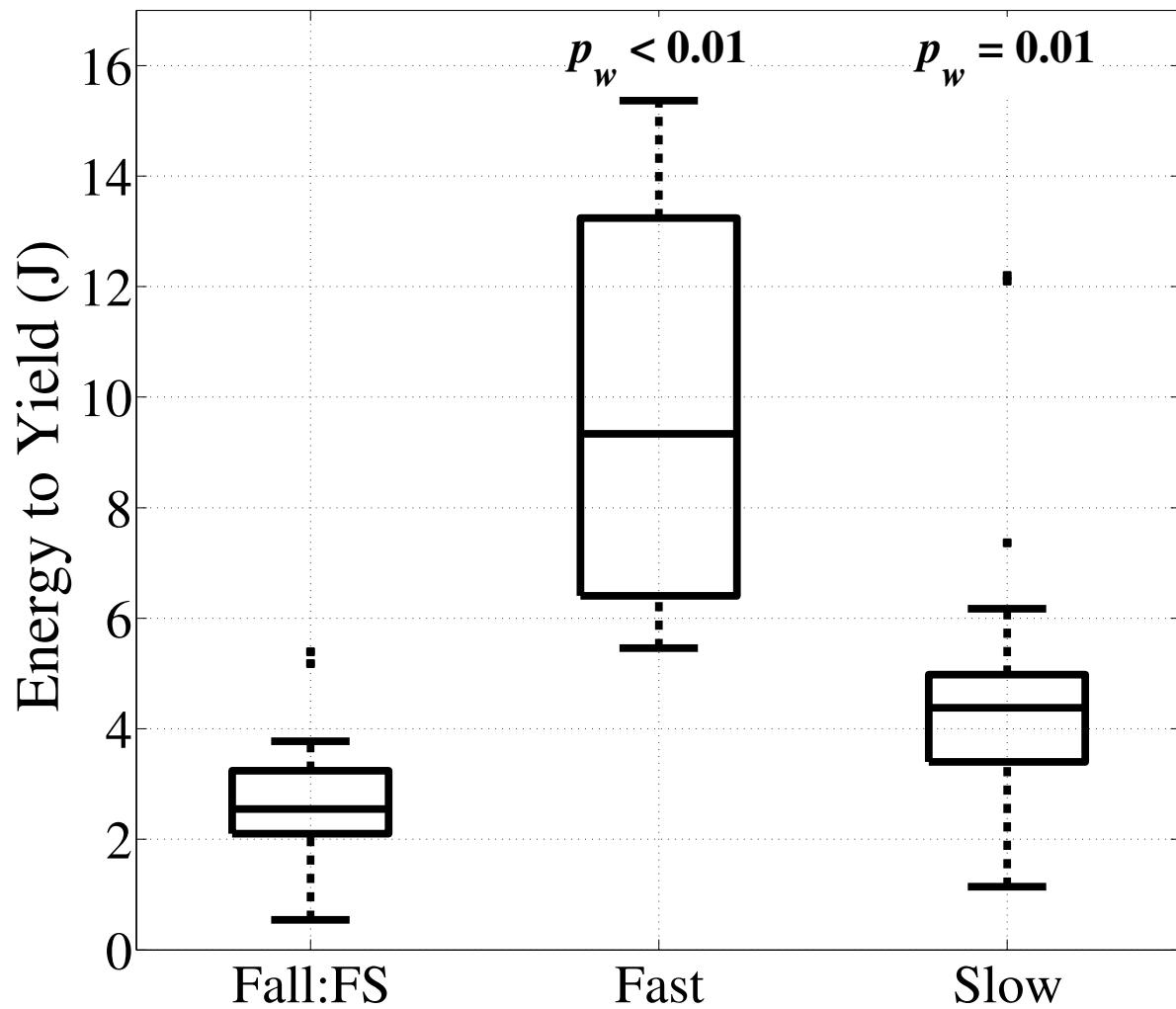


Figure B.7: Energy to yield in the fall:FS, fast and slow groups in the failure tests.

Graphic ©Seth Gilchrist, 2013.

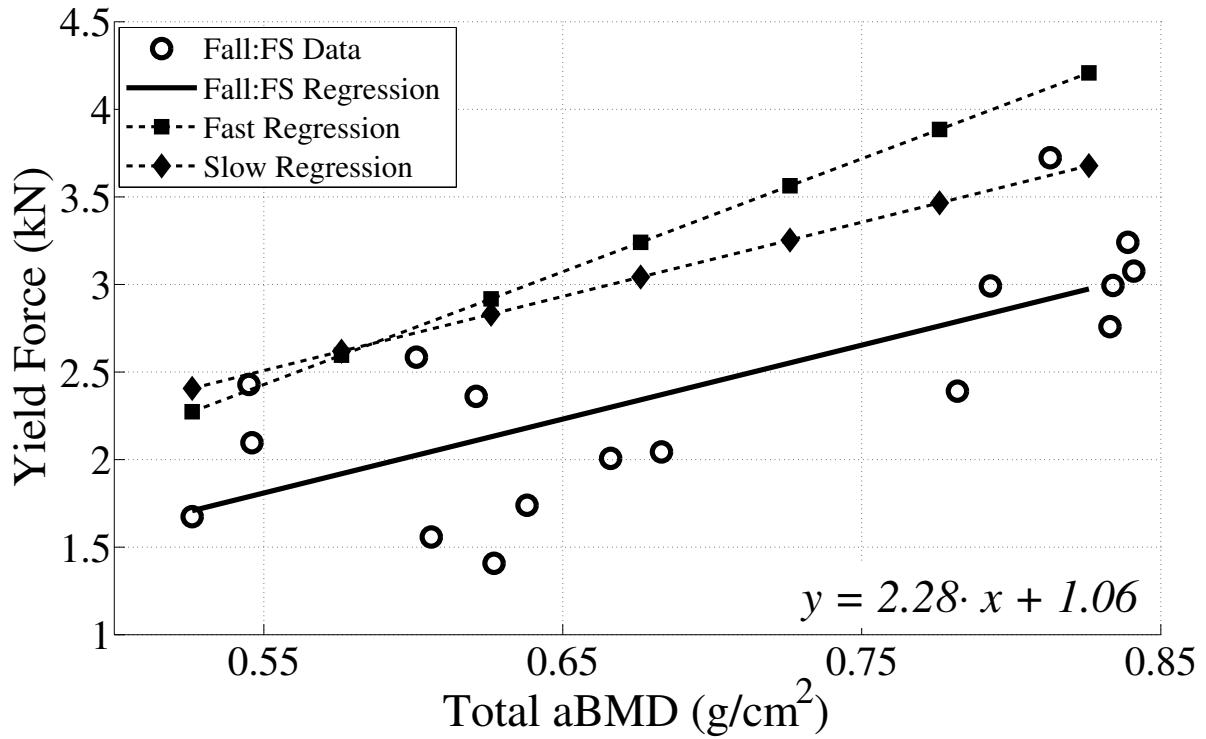


Figure B.8: The yield force as a function of total aBMD in the failure tests. The equation is the regression for the current results. Graphic ©Seth Gilchrist, 2013.

Appendix C

Digital volume correlation

As discussed in the introduction to this thesis, validation of computational data is key to its interpretation. Finite element code is often used to predict the deformations of the cancellous bone – a measure for which it has never been validated. There are a number of FE studies that are indicating that the cancellous bone compartment may play a critical role in hip fracture, but until the deformations measured in the cancellous bone can be validated, these results should be interpreted with a degree of scepticism.

Digital volume correlation (DVC) is a technique similar to digital image correlation (DIC) which uses digital image tracking to determine the motion of a texture in two sequential images of the same thing under different loading conditions. The tracking is done based on registration of subregions of the first image with the second image (Figure C.1). Once a number of subregions have been tracked, the relative displacement of each region can be used to calculate strain. This strain value can then be compared to FE calculated strain fields as a validation.

Similarly to FE, this technique is computational in nature, but does not use any a prior knowledge of the potential deformations, making it an independent measurement technique. While it may not be possible to conclude that it is a fully qualified measurement method (there remains no way to take objective measures of the occluded cancellous bone compartment), comparison with FE using Bland-Altman plots would provide a source of verification [206].

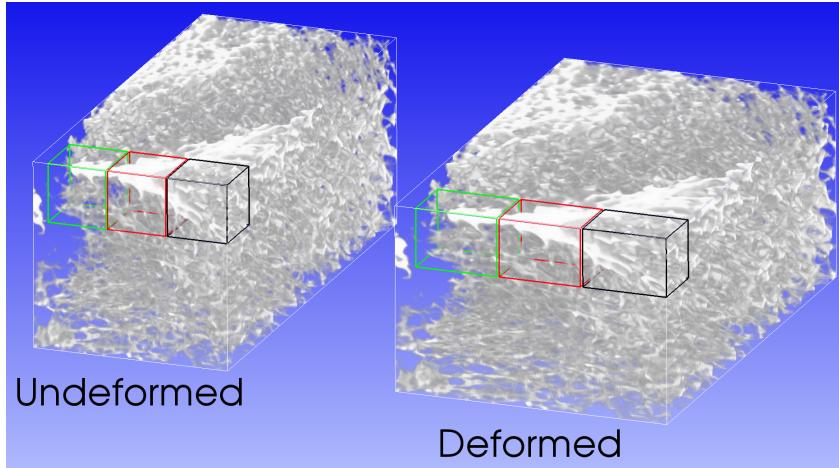


Figure C.1: In DVC, regions of the undeformed image are transformed and registered to a second image of the same object after deformation. Graphic ©Seth Gilchrist, 2013.

C.1 The DVC technique

DVC works by comparing two images on a subregion basis and computing the motion of each subregion between the images. To acquire these images, an object is imaged multiple times while undergoing loading. The first image will be of the unloaded object, the second image will be at some finite load, and each successive image will be at incrementally higher load. The images can then be compared to the first, unloaded image, or to the previously acquired image in order to determine the deformation between the two load levels. The common terminology for the two images are the *fixed* and *floating* or *moving* images. The fixed image is used as the reference image and the floating image is registered to the fixed image. In the present DVC algorithm, the fixed image is the one from the higher load condition, and the floating image is the one from the lower load condition.

In order to register to images, there must be a semi-random image texture that allows for determination of a unique match between the two images. Cancellous bone imaged using CT is a good candidate because it contains a texture, which while being regular in nature, is unique at any given location.

The DVC techniques described in this section were carried out in custom software pro-

grammed in C++ and utilizing the Visualization Toolkit (VTK) and the Insight Toolkit (ITK).

The source code can be found in §E.4.

C.2 Design and implementation of a digital volume correlation algorithm

On its face, the DVC method is relatively simple. The steps of the DVC method are:

1. CT a bone in an unloaded state.
2. CT the bone in a loaded state.
3. Read in the loaded and unloaded images.
4. Globally register the two images to ensure overlap and good initial guess.
5. Break the images into subregions. The fixed image subregions will be large, and the moving image subregions will be smaller.
6. Register a moving subregion within the corresponding fixed subregion.
7. Check for possible errors in the calculation and label regions for reanalysis.
8. Perform secondary registration on regions identified as erroneous.
9. Store the translation data of the registration in a new image with a voxel located at the centre of the fixed subregion.
10. Spatially differentiate the translation data to obtain strain.

The parameters used while performing these tasks are related to the methods of each task. There are three main tasks being performed: imaging, subdividing and registration. The decisions made about how each of these are performed influence the overall reliability, robustness and speed of the process.

C.2.1 Imaging for DVC

Imaging for DVC can be done in multiple ways. The only requirements are that it provide texture at a resolution sufficient to realize the texture. Previous researchers who have used DVC, or a variant of, have used magnetic resonance imaging (MRI) [18] and CT [10–13, 88, 89, 130, 203, 216]. CT is the modality of choice as the machines are more readily available in the resolution ranges required, and they work on differences in density which makes imaging in materials like conglomerate rock possible [26, 130].

The general rule for the imaging portion is to acquire images at as high a resolution possible. This is because the end result of the registration process, discussed in §C.2.3, is highly dependent on the outcome resolution. That said, if the resolution is too fine, the image sizes become cumbersome, especially when imaging whole bones which can be many centimetre in length.

C.2.2 Subdividing images for DVC

Subdividing images into subregions for DVC requires knowledge of your texture dimensions. There are two dimensions to chose when subdividing an image: i) the size of each subregion, and ii) the spacing between subregions.

Since the registration relies on matching texture, each region must contain enough texture information to differentiate it from the surrounding image, and this is used to determine the size of each region. In the case of DVC the texture is the trabecular structure, so knowledge of aspects like trabecular spacing and trabecular thickness allow for selection of this number. For registration, the larger the subregions, the more accurate the registration due to the increased data volume. However, in terms of strain measurement, smaller subregions are preferable as the registration steps will tend to return an average displacement over the region and larger regions lead to more averaging. Additionally, if the regions become too large, holding them in memory while performing registration can be challenging. In the presented method, both subregions are loaded into the computer’s RAM in their entirety. Modifying the code to stream

only the sections currently under consideration could make the random access memory (RAM) requirements more manageable.

Due to a dearth of information regarding what an appropriate volume dimension is for DVC, dimensions were taken from the DIC literature to justify the selection. Researchers in DIC have determined that speckle size, subregion dimensions and pattern entropy all influence the outcome of the calculation [127, 182, 248]. The most robust method for determination of subset size is subset entropy [248] and the sum of square of subset intensity gradients (SSSIG) [182], both of which are summations of gradients over the subregion. Other researchers have shown that subregion size can be chosen based on the mean texture size [127], which in the case of DIC is a dot speckle. The experiments on three different speckle patterns utilizing multiple subregions sizes showed that a region that was 2.3x to 4.3x the size of the texture was appropriate. While using the gradient based measures would lead to the most robust subregion dimensions on a subregion-by-subregion basis (it would be evaluated dynamically during subdivision), a value based on the trabecular dimensions was desired to make implementation easy and intuitive. For this reason, a selection of 3x trabecular spacing (TbSp) was selected. This was used as it would ensure that at least two trabeculae were included in each direction, but kept the subregion size to a minimum.

Spacing of the subregions, along with the accuracy of the registration, influences the overall uncertainty of strain. Equation C.1 gives the definition of engineering strain in a single dimension, where ϵ is strain, Δl is the change in length, and L is the original length. Equation C.2 expands that definition to the average strain between two points, where x_n is the original position at location n , and Δx_n is the change position at location n . If we assume that we know the starting positions with absolute certainty, the uncertainty of the strain measure reduces to Equation C.3, in which $\delta(\Delta x_n)$ is the uncertainty of the change in position at location n . Calculating the partial derivatives, and substituting into Equation C.3 gives the strain uncertainty in terms of the position change uncertainty. What we see is that the strain error is inversely related to the original distance between the measurement points (Equation C.5), i.e., the larger

the distance between the subregions, the lower the strain error, regardless of the registration error. This also shows that the strain error is directly related to the registration, which is an intuitive result.

$$\varepsilon = \frac{\Delta\ell}{L} \quad (\text{C.1})$$

$$= \frac{\Delta x_2 - \Delta x_1}{x_2 - x_1} \quad (\text{C.2})$$

$$\delta\varepsilon = \delta(\Delta x_2) \cdot \frac{\partial\varepsilon}{\partial\Delta x_2} + \delta(\Delta x_1) \cdot \frac{\partial\varepsilon}{\partial\Delta x_1} \quad (\text{C.3})$$

$$\frac{\partial\varepsilon}{\partial\Delta x_2} = \frac{1}{x_2 - x_1} = -\frac{\partial\varepsilon}{\partial\Delta x_1} \quad (\text{C.4})$$

$$\therefore \delta\varepsilon = \frac{\delta(\Delta x_2) - \delta(\Delta x_1)}{x_2 - x_1} \quad (\text{C.5})$$

This shows that increasing the distance between subregions is advantageous from an accuracy perspective, however, increasing the spacing also has the unfortunate consequence of decreasing the resolution of the resulting strain field. The upper bound on the spacing is therefore determined by desired accuracy and is normally determined using a set of validation experiments. The lower bound can be set by considering the limits of the continuum assumption in the material that you are considering. Work done to identify the limits of continuum theory [135] on inhomogeneous solids, as well as work in cancellous bone to determine the limits of continuity in FE modelling [90] can provide insights that may guide us to a reasonable solution. Theoretical work on cancellous bone suggests that continuum level analyses break down when FE element sizes are less than 3-5 trabecular separations [90]. The work in inhomogeneous solids suggests that a square volume with edges about 15x the crystal size in the solid gives consistent results [135]. Based on the results from these experiments, it was determined that a region spacing between 5x and 15x TbSp would be appropriate. The final value would depend on the specimen, the variation of TbSp across the specimen, and the smoothness of the results obtained at each spacing (e.g., erratic results would require reanalysis with larger separations). In general, the first analysis of any specimen was conducted at 5x TbSp and increased from there if results appeared erratic.

In summary, DVC images were broken up in to square regions that were 3x TbSp on a side, and separated by a minimum of 5x TbSp. These values could be increased if the strain fields appeared erratic, because increasing region size would tend to decrease the registration error, and increasing separation would decreases strain error according to Equation C.5.

C.2.3 Registration of image for DVC

The registration of the subregions is the crux of the DVC technique. It has a number of sub-processes, each of which require decisions on the methods used to minimize error and computation time – frequently competing interests. The process (Figure C.2) has two data structures, and four processes. The data required for the process are the two images to be registered, in the case of the DVC program these are the subregion images taken from the loaded and unloaded CT images. The processes are:

1. Interpolator: interpolates the moving image pixel data into the grid of the fixed image.
2. Metric: compares the interpolated moving image and the fixed image and provides a single value representing their agreement.
3. Optimizer: computes the most likely position of the moving image that will result in good agreement with the fixed image.
4. Transform: manipulates the grid of the moving image into a new space.

The registration proceeds can be described as follows: (i) The two images are loaded into memory and the moving image is interpolated into the grid of the fixed image. (ii) The metric computes the agreement between the two images and also in a neighbourhood of the current position. This can be quite time consuming as the “current position” is defined in terms of the transform parameters which can be numerous. For a rigid transform there are six parameters (x , y , z translations and rotations), and for an affine transform in 3D there are 12 parameters. (iii) The optimizer examines the metric value and local gradient and makes a decision on how to transform the moving image to better match the fixed image. (iv) The moving image is

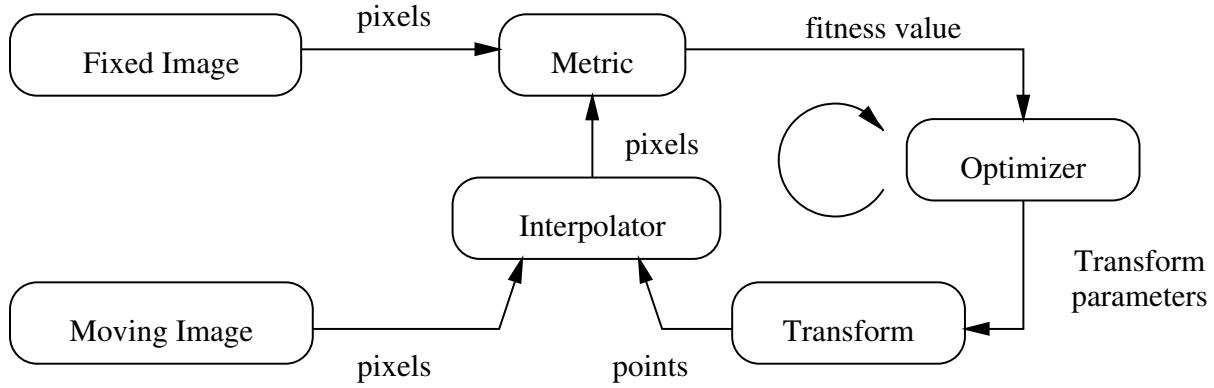


Figure C.2: The image registration process operates in an iterative fashion. Graphic from Ibanez [99], BSD-2 license, no permission required.

modified based on the optimizer-determined transform and the interpolator interpolates the data of the transformed moving image into the grid of the fixed image. (v) Once the metric has reached a user-set criteria indicating a suitable match, the optimizer terminates the loop and returns the transform parameters. Each of these processes can be done in a number of ways, and the relative strengths of each are discussed below.

Interpolator

Interpolation is the method used to determine the value of the pixels between data points in a discrete dataset. Images are a discrete representation of analogue information in which each pixel, or voxel as they are known in 3D, has a defined location and value. The location of the voxel can typically be thought of as accurate to the level of precision provided by the CT machine used to generate the image. The value of the voxel is an average value of the analogue data in the region around the location. This 3D the region of averaging will be an ovoid, but for simplicity it is often thought of as being a cuboid (Figure C.3).

When comparing two images, it is very unlikely that the grid of each image will line up exactly, especially after an arbitrary transform has been applied to one of the images. This means that one would be comparing the data in the fixed image at a specific point, to the data in a moving image Delaunay region. In order to perform the comparison, one must interpolate the value of the moving image in the Delaunay region.

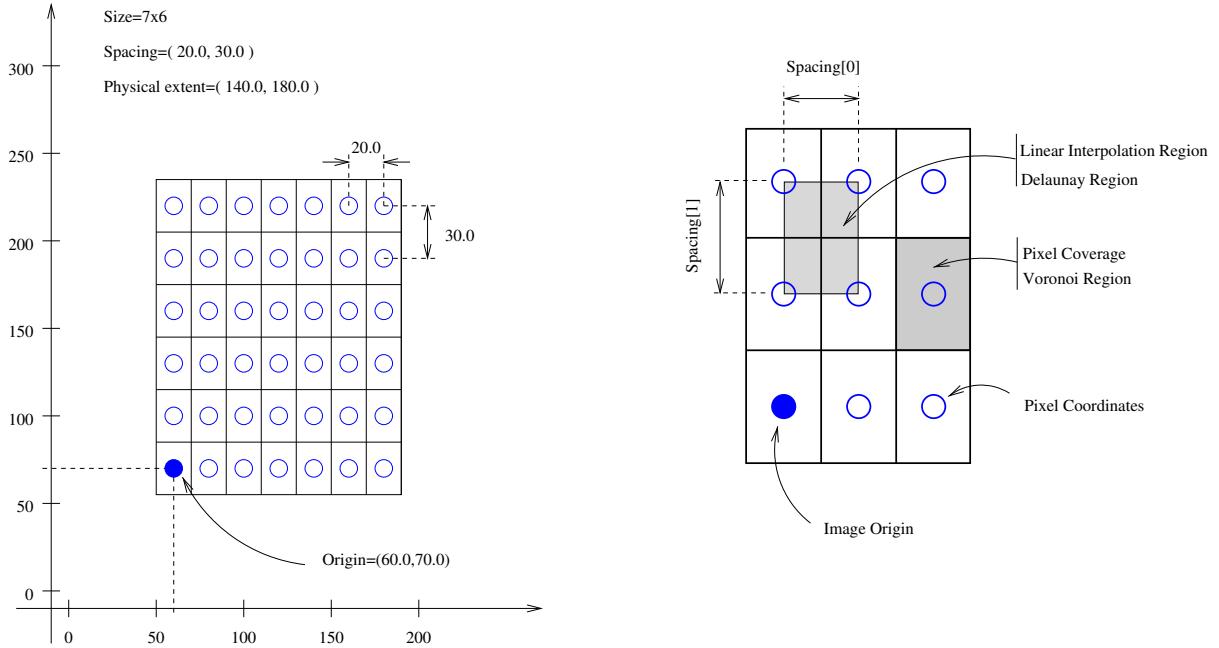


Figure C.3: Image data has point locations that are precise, and point values that are averages over the pixel Voronoi region. The Delaunay region is the space in which interpolation is necessary. In this example, the pixels of the image are not square, resulting in a rectangular Voronoi region. In 3D medical images, voxels are often defined with one dimension larger than the other two (the out-of-plane dimension). Graphic from Ibanez [98], BSD-2 license, no permission required.

The typical image interpolation techniques are (Figure C.4): (i) nearest neighbour, in which locations in the pixel's Voronoi region all take the value of the pixel; (ii) linear, in which a linear function is used to determine the value in the Delaunay region; and (iii) basis spline (B-spline), in which splines of order ≥ 2 are constructed and used to determine the value in the Delaunay region.

Nearest neighbour interpolation is used in image registration when aligning a label map to an image or another label map. Interpolation of a label map has no meaning, e.g., if one has a label map of the cortical bone, cancellous bone and bone marrow, there is no meaning to interpolating between them, pixels are either one or the other, not a mixture. This type of interpolation is rarely used when interpolating image data, and is not used in the DVC algorithm described here. Linear interpolation is frequently used with image data because it gives reasonable results and is computationally inexpensive and fast. B-spline interpolation

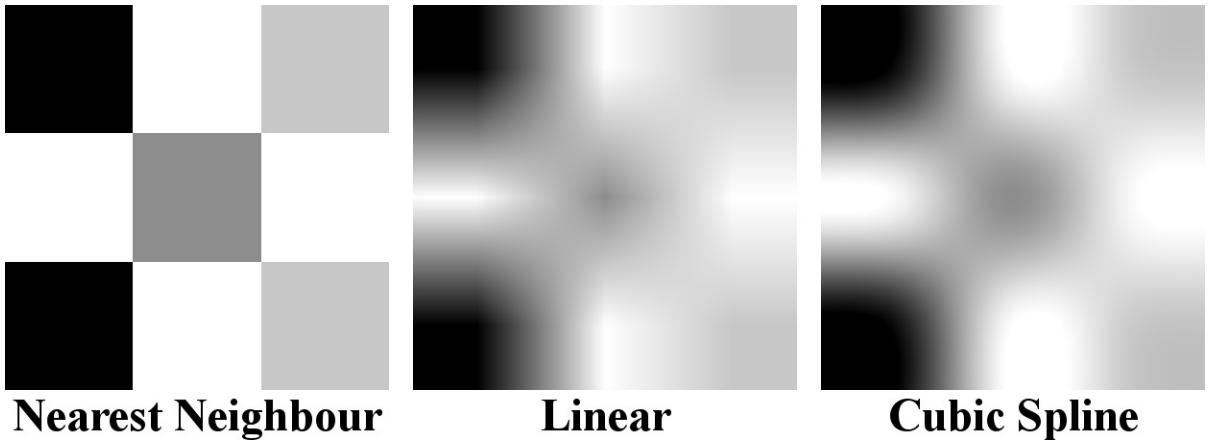


Figure C.4: Image data interpolated using nearest neighbour (left), linear (middle) and cubic B-spline (right) interpolation. It can be seen that linear and cubic spline give similar results. Graphic ©Seth Gilchrist 2013.

is used when increased accuracy is needed, with the degree of computation being determined by the order of the spline used. Investigators who have researched higher order interpolation routines [208] have shown that for grey scale images (as opposed to binary images) a fourth order B-spline provides significantly lower error (up to $1/5^{th}$) than a third order B-spline.

Testing of the DVC algorithm using both quintic B-spline and linear interpolators showed that at small strain values there is an advantage to the B-spline interpolators (Figure C.5). When a test image was synthetically strained to 1.6% strain (as described in §C.3) the B-spline interpolator had a median error of approximately half that of the linear interpolator (0.0255 and 0.0488 voxels, respectively). This advantage diminished when strains were larger, with the B-spline interpolator showing only an 8% improvement when a maximum strain of 10% was applied (0.2489 and 0.2708 voxels for the B-spline and linear interpolators, respectively).

The DVC algorithm uses both linear and quintic B-spline interpolation. In the global registration (Item 4 on page 193) linear interpolation is used. This step is meant to give a good approximate starting position for the subregion registrations. It operates on a large data set so needs to be fast and require little memory, making linear interpolation a good candidate. During the registration of the subregions, the increased accuracy of the quintic B-splines is worth the computational cost to ensure accurate strain output.

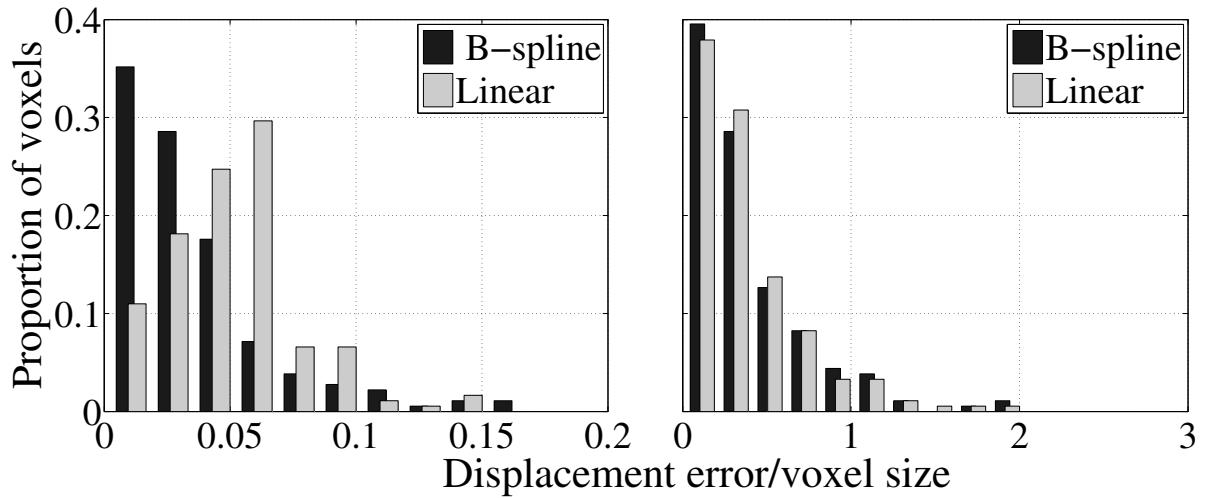


Figure C.5: Comparison of linear and 4th order B-spline interpolation. When a synthetic strain with a maximum value of 1.6% ε was applied to an image, the B-spline interpolator showed a clear advantage (left). This was not true when a synthetic strain with a maximum value of 10% ε was applied (right). Transformations were performed using affine transforms for this test. Graphic ©Seth Gilchrist 2013.

Metric

The metric is the most computationally expensive part of the registration. That said, the performance of the metric directly affects the results of the registration, and the number of iterations required by the optimizer to reach the best match. Metrics are either correlation or sum-squared difference functions, both of which share many fundamental characteristics and are closely related [226].

In order to reduce the computational expense of the process, we can take advantage of certain aspects of the HR-pQCT imaging technique. The most notable of these aspects is that image parameters such as lighting intensity are stable over time and between images. This fact means that normalization of the images is not required, greatly reducing the computational cost [183].

For this reason, the non-normalized, sum-squared difference metric was used in the current DVC technique.

Table C.1: Different DIC transformations and their number of parameters in 3D [100].

Rigid body	Six parameters: three translations and three rotations.
Euler transform	Seven parameters: three translations, three rotations and a scaling.
Affine	12 parameters: nine for shear, rotation and scaling, and three for translations.
B-spline deformable	Many parameters that describe an arbitrary warping of the image space.

Transform

The transform is used to manipulate the moving image subregion such that it resembles the fixed image, making it possible to find a suitable match. There are a number of different kinds of transforms with varying levels of complexity and varying numbers of parameters to characterize the deformation (Table C.1).

Each of these transforms can describe different kinds and degrees of deformation. The simplest of them is the Euler rigid body transform which describes only translations and rotations and therefore cannot account for actual changes in shape. The affine transform is very versatile and is used in many image processing algorithms. It has 12 parameters and is able to describe scaling, shear, translation and rotation in each of the three dimensions independently, it allows for significantly more deformation, with only five more parameters than the Euler transform.

Because the number of transform parameters dictate the number of calculations that must be made by the metric when evaluating the gradient surrounding the current point, it is computationally advantageous to minimize the number of parameters. In DIC the selection of the transform is often termed the shape-function selection [143, 183, 207]. In order to be accurate, the deformation allowed for each subregion must be able to capture the change in shape of that subregion. For small strains and small subregions, one can assume that the deformation across the region is linear, but for larger strains and larger regions this assumption begins to break down. For subfailure strain levels, first order shape functions have been shown to be as effective as higher order functions in sub-pixel registration [20].

For the current DVC algorithm Euler and affine transforms were considered the best two

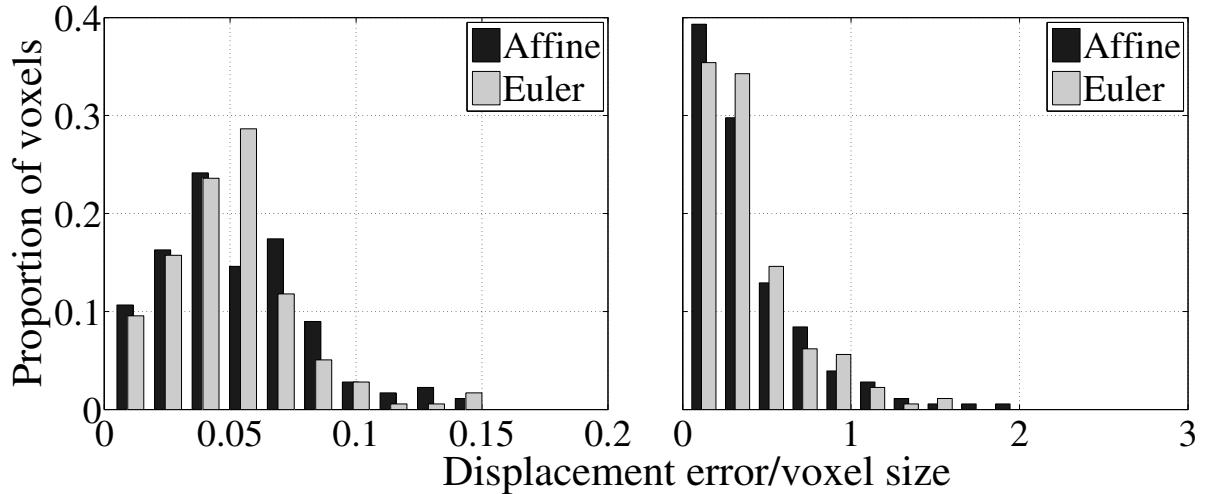


Figure C.6: Comparison of affine and Euler transformations. When a synthetic strain with a maximum value of 1.6% ϵ was applied to an image, the affine transform was seen to be moderately better (left). The comparison improved for the affine transform when a synthetic strain with a maximum value of 10% ϵ was applied (right). Transformations were performed using linear interpolations for this test. Graphic ©Seth Gilchrist 2013.

candidates. Tests performed using synthetically strained images (described in §C.3) were conducted using linear interpolation and changing the transformation between affine and Euler (Figure C.6). When a linear strain with a maximum value of 1.6% was applied, the median displacement error measured by the affine and Euler transforms differed by 2.2% (0.0474 and 0.0485 voxels for affine and Euler, respectively). When the deformation was increased such that the maximum strain was 10% the affine transform was 3.7% better than the Euler transform (0.2708 and 0.2808 voxels for affine and Euler, respectively). Even though the difference in measured displacement errors was low in both cases, affine transformations showed a consistent reduction in error across all strain values.

The DVC algorithm described here was designed to use two different transforms. An initial, global alignment of the two images is performed using a Euler rigid body transform. This ensures that if the two images were separated in space due to placement in the CT scanner, they are first roughly aligned so that they are fully overlapping. The subregions are then mapped using affine transforms. The decision to use the affine transform was made because the con-

sistent improvement in displacement measurement across all strains would improve the overall results of the method.

Optimizer

The optimizer makes decisions on how to deform the moving image such that it will correlate well with the fixed image. Since registration is performed in real-world coordinates (rather than pixel-space), there are an infinite number of different possible solution locations. This means that not all possible solutions, or even probable ones, can be tested in a reasonable time span, therefore a method to find the solution with the minimum number of iterations must be used. This is the job of the optimizer.

Most classical optimizers work using the gradient of the metric field. The metric provides a single value describing the agreement between the two images at a given transformation location. The optimizer evaluates the metric in a neighbourhood around the current location and follows the gradient of the metric to either the maximum (in the case of correlation coefficient metrics) or the minimum (for sum-square difference metrics).

Newtonian optimizers have been shown to work well for DIC methods [20], however their assumption of a quadratic optimization problem can lead to a small radius of convergence. Gradient descent methods are known to be robust, with a radius of convergence that can be tailored by selecting an appropriate initial step size. That said, gradient descent optimizers are known to be inefficient, suffering from the “ping-pong” effect where the optimizer will use ever smaller steps as one approaches the optimal solution [158]. One way to help with this issue is to use a regular step, gradient descent optimizer which uses one step size until the sign of the gradient reverses, at which time it reduces the step size by a set relaxation factor and continues. In this way, the step size is not determined by the magnitude of the gradient, but by the passing of the minimum, reducing the number of steps to convergence.

Previous researchers have examined the accuracy of different optimizers for DIC and found that while Newtonian methods gave the best results, there was a significant computational

penalty for using them [20]. In fact, the Newtonian method took two orders of magnitude longer to solve than the gradient based method, even with the known inefficiency of the gradient methods.

To allow for the largest radius of convergence and highest speed, a regular step, gradient descent algorithm was used for the current DVC algorithm.

C.2.4 Error analysis

The error analysis routine for the DVC was used to detect and reanalyse subregion registrations that were determined to be likely erroneous. The search for erroneous registrations was based on parametric statistics of the region under consideration and the connected analysis regions. The results image was iterated through and for each region the average and standard deviation of the connected neighbours of the displacements in the x , y and z directions were calculated. The tolerance for an erroneous result is set by the user at the start of the analysis in number of standard deviations from the connected neighbour mean. For example, if point n has a z -displacement of 0.51 mm and the average and standard deviation of the z -displacement for n 's neighbours are .48 mm and .02 mm, respectively, this region would be reanalysed if the user set a tolerance of one standard deviation, but not if two standard deviations were used. If a region is flagged for reanalysis, an initial guess of its solution was set as the average of the valid neighbours displacements.

C.2.5 Data storage

The data for the DVC algorithm is stored in a finite element style mesh. Initial versions of the algorithm used a regular grid (similar to an image), but this method required too many analyses for large datasets and also didn't have the flexibility to represent complex shapes while maintaining proper subregion spacing. Other advantages of the FE mesh is that algorithms for calculating the derivatives (i.e., strains) have been developed and did not have to be implemented from scratch. The open-source mesh generator Gmsh (<http://geuz.org/gmsh/>) was used to generate the meshes and the DVC algorithm could read in either a Gmsh mesh, or a VTK

mesh with or without data. Regardless of the mesh provided, a VTK mesh was used internally, as there are a number of predefined algorithms and iterators that make working with them easy.

C.2.6 Spatial differentiation

Strain is determined using spatial differentiation. Equation C.2 gives the strain definition in 1D, but the idea is the same in three dimensions. The VTK mesh used to store the displacement data has built-in methods for taking spatial derivatives to determine ε_x , ε_y , and ε_z , and for calculating the Eigenvalues and Eigenvectors to give principal strains and their directions.

C.2.7 Coding considerations

The code to perform the DVC calculations was implemented in C++ utilizing ITK (3.20.1, Kitware, Clifton Park, NY) for the registration and VTK (5.10.1, Kitware, Clifton Park, NY) for data storage, and manipulation. It was done in three classes and one main file. A generic DVC class called *DIC* contained methods and virtual functions for performing DVC on any 3D image. A child class of DIC, specific for DVC in which data is stored and manipulated using a VTK mesh was written and called *DICMesh*. The last class was designed as a helper class to setup and run the DVC calculations. It was called *AnalyzeDVC* and contained methods for reading input images, logging and saving data. Finally, the main program which called on all of these classes to run the DVC analysis was called *AnalyzeImages*. Code for all of these calculations can be found in §E.4.

The code was compiled using g++ (4.7.2, Free Software Foundation, Boston, MA) configured by cmake (2.8.9, Kitware, Clifton Park, NY) on a computer (Vostro 430, Intel Core i7 860 @2.8 GHz x8, 8 GB RAM, Dell, Toronto, ON) running Linux (3.5.0-40 x86 64bit, Linux.org). The cmake files can be found with the other DVC code in §E.4.

C.2.8 Loading apparatus

Two loading apparatuses were designed and constructed to obtain images from the HR-pQCT scanner for use with the DVC algorithm. The first one is intended to load small specimens

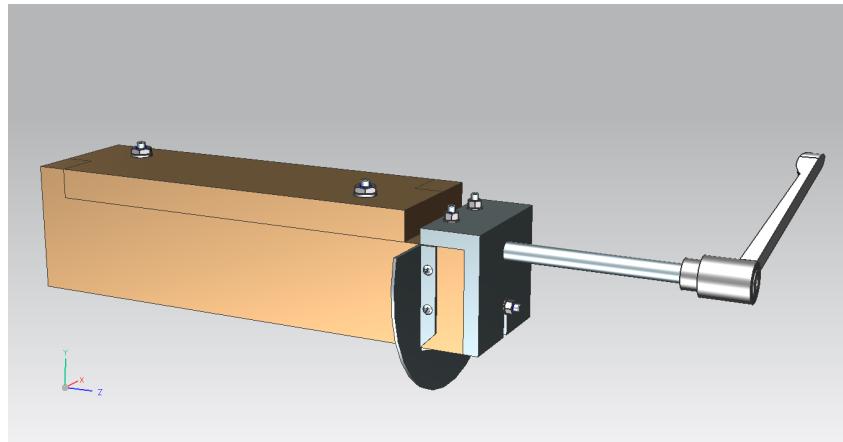


Figure C.7: The DVC axial loading device used to apply loads to small specimens.

Graphic ©Seth Gilchrist, 2013.

loaded axially in the scanner, and the second loads full proximal femurs cross-wise in the scanner. Both use manual displacement control and output load using either a discrete load cell or a built in load cell.

Axial loading of small specimens

Loading of small specimens allowed for compression testing of bone cores or small bones in the HR-pQCT. For x-ray compatibility the device was constructed of wood. A screw was used to push a keyed plunger to prevent rotations of the screw being transmitted to the specimen. The plunger compressed the specimen in line with a single axis load cell (Figures C.7 and C.8).

The load cell used in the compression apparatus was a 0–2 kN compression load cell from Omega Engineering (Figure C.9).

In addition to the compression device, a mould was constructed that allowed the specimen to be mounted in a pot conforming to the loading device's interior dimensions (Figure C.10).

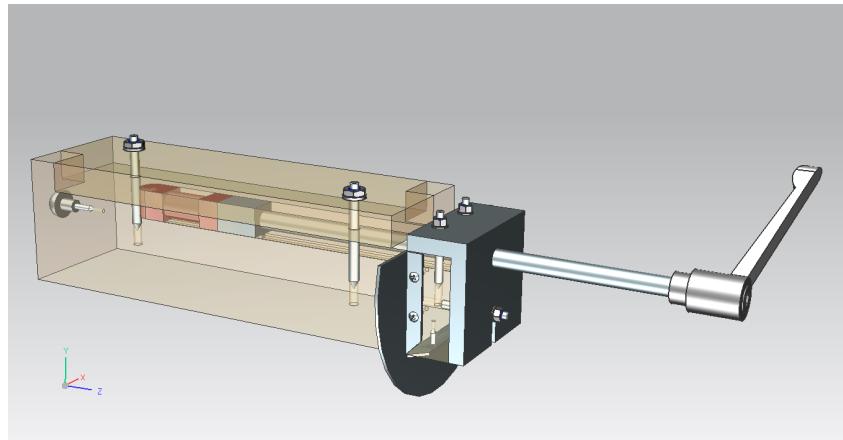


Figure C.8: A transparent rendering of the DVC axial loading device. The keyed plunger can be seen with the specimen rendered in red. Graphic ©Seth Gilchrist, 2013.

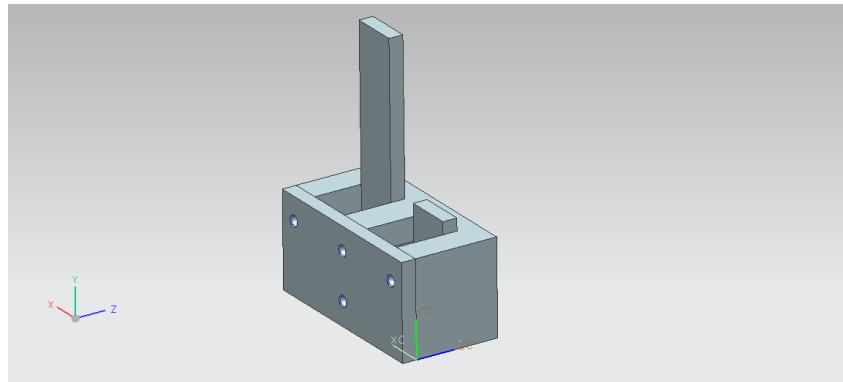


Figure C.10: The specimen mould for the axial loading of the DVC apparatus allowed for moulding of the specimen potting to conform to the internal dimensions of the apparatus. Graphic ©Seth Gilchrist, 2013.

Cross loading of whole proximal femurs

The end goal of the DVC project is to be able to load whole proximal femurs in the literature standard fall configuration [76]. To this end, a device that is large enough to accept a full proximal femur, while being small enough to fit into the bore of the HR-pQCT was developed (Figures C.11 and C.12). The apparatus had four posts which were driven independently by screws to apply compressive loads to the bone. Additionally, each post had a strain gauge glued on to measure the load passing through the post.

The apparatus was calibrated by securing it in the Instron and applying tensile loads to the

19 mm (0.75") DIAMETER STAINLESS STEEL COMPRESSION LOAD CELL

STANDARD AND METRIC MODELS

LC302/LCM302 Series

Compression

0-25 lb to 0-1000 lb

0-100 to 0-5000 N

1 Newton = 0.2248 lb

1 daNewton = 10 Newtons

1 lb = 454 g

1 t = 1000 kg = 2204 lb

All Models
\$295



- ✓ Small 19 mm (0.75") Dia.
- ✓ Low 13 mm (0.5") Profile
- ✓ Robotic and Test Bench Applications
- ✓ Built-In Load Button for Easy Installation
- ✓ 5-Point Traceable Calibration Provided

The LC302/LCM302 Series load cells are only 19 mm ($\frac{4}{5}$ ") in diameter and fit in the smallest places. Their high $\pm 0.5\%$ accuracy and small package solve many industrial force measurement applications. The all stainless steel construction and high-quality strain gage ensure long-term reliability.

SPECIFICATIONS

Excitation: 5 Vdc, 15 Vdc max

Output: 1 mV/V (nominal)

Accuracy: $\pm 0.5\%$ FSO linearity, hysteresis, repeatability combined

5-Point Calibration:

0%, 50%, 100%, 50%, 0%

Zero Balance: $\pm 2\%$ FSO

Operating Temp Range:

-54 to 125°C (-65 to 250°F)

Compensated Temp Range:

16 to 71°C (60 to 160°F)

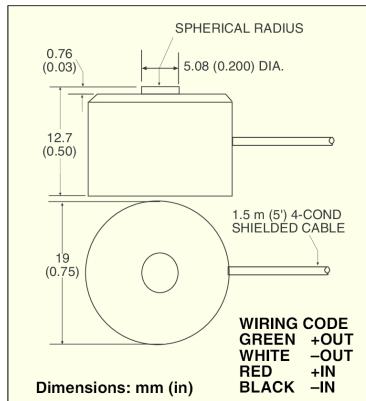
Thermal Effects:

Zero: 0.009% FSO/°F

Span: 0.036% FSO/°F

Safe Overload: 150% of capacity

Protection Class: IP54



LC302-1K, \$295,
shown actual size.

Deflection: 0.03 to 0.08 mm
(0.001 to 0.003")

Ultimate Overload: 300% of capacity

Bridge Resistance: 350 Ω minimum

Construction: Stainless steel

Electrical Connection: 1.5 m (5')
4-conductor insulated cable
with temperature compensation board

STANDARD MODELS **MOST POPULAR MODELS HIGHLIGHTED!**

To Order (Specify Model Number)

CAPACITY lb	N	MODEL NO.	PRICE	COMPATIBLE METERS*
				DPI-S, DP41-S, DP25B-S
25	111	LC302-25	\$295	DPI-S, DP41-S, DP25B-S
50	222	LC302-50	\$295	DPI-S, DP41-S, DP25B-S
100	445	LC302-100	\$295	DPI-S, DP41-S, DP25B-S
250	1112	LC302-250	\$295	DPI-S, DP41-S, DP25B-S
500	2224	LC302-500	\$295	DPI-S, DP41-S, DP25B-S
1000	4448	LC302-1K	\$295	DPI-S, DP41-S, DP25B-S

METRIC MODELS

CAPACITY N	lb	MODEL NO.	PRICE	COMPATIBLE METERS*
				iSeries, DP41-S, DP25B-S
100	22.5	LCM302-100N	\$295	iSeries, DP41-S, DP25B-S
200	45	LCM302-200N	\$295	iSeries, DP41-S, DP25B-S
500	112	LCM302-500N	\$295	iSeries, DP41-S, DP25B-S
1000	225	LCM302-1KN	\$295	iSeries, DP41-S, DP25B-S
2000	450	LCM302-2KN	\$295	iSeries, DP41-S, DP25B-S
5000	1124	LCM302-5KN	\$295	iSeries, DP41-S, DP25B-S

Comes complete with 5-point NIST-traceable calibration and 59 k Ω shunt data.

* See section D for compatible meters.

Ordering Examples: LC302-25, 25 lb capacity load cell, \$295.

LC302-100, 100 lb capacity load cell, \$295.

LCM302-100N, 100 Newton capacity load cell, \$295.

LCM302-2KN, 2000 N capacity load cell, \$295.

Recommended Reference Book: Mechanical Design Handbook, ME-1804, \$130.
See Section Y For Additional Books



LOAD CELLS



Figure C.9: The data sheet for the 2 kN load cell used in the axial loading apparatus.
[178], public domain, no permission required.

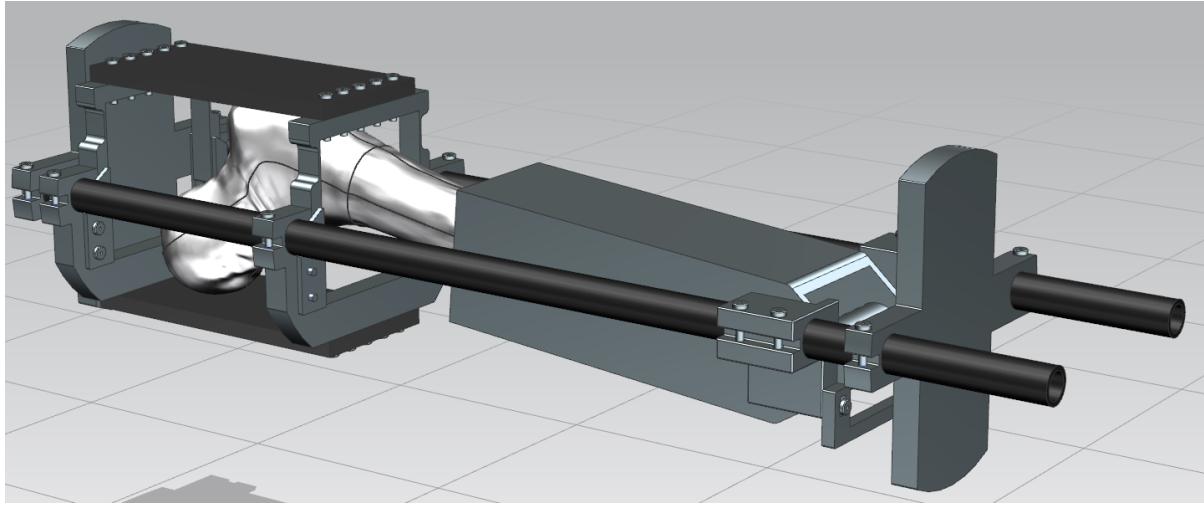


Figure C.11: The cross loading of whole proximal femurs HR-pQCT apparatus.
Graphic ©Seth Gilchrist, 2013.

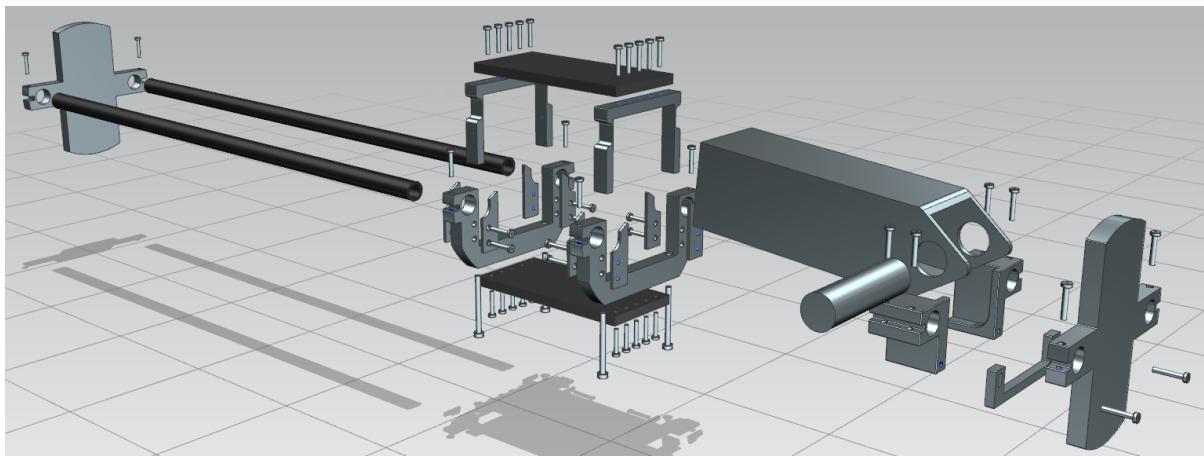


Figure C.12: An exploded view of the cross loading of whole proximal femurs HR-pQCT apparatus. There are two small differences between this rendering and the final device. One was the doubling of the loading platens, each of which consisted of two plates of carbon glued together. The second was the use of washer plates between the carbon platens and the five screws to prevent local crushing of the carbon plates. Graphic ©Seth Gilchrist, 2013.

Table C.2: Calibration constants for the DVC cross loading device. The calibration is linear of the form $Force(N) = Microstrain \cdot a + b$.

Post	a	b
Left front	0.5673	51.0
Left back	0.5568	46.7
Right front	0.5822	12.0
Right back	0.5298	57.2
Combined	0.5590	41.7

frame, simulating compressing a proximal femur (Figure C.13). The results of the calibration showed that the device was linear to a total applied load of 2.5 kN (Table C.2 and Figure C.14). The slight non-linearity observed in the graph is typical for aluminium tensile loading which does not have a perfectly linear stress strain curve.

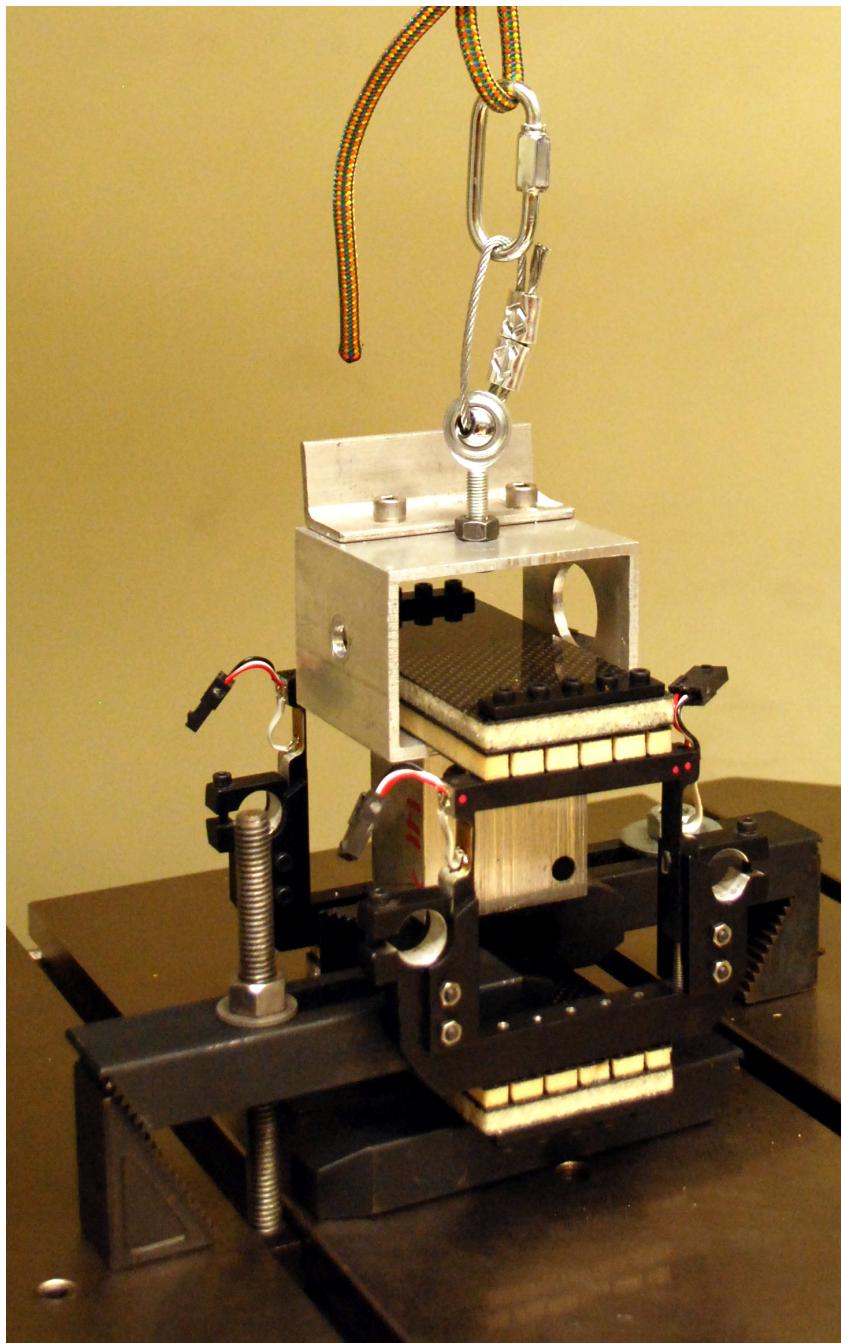


Figure C.13: The cross loading apparatus was calibrated by mounting it in the Instron and applying tensile loads to the frame, which simulated compressing a bone internally. Graphic ©Seth Gilchrist, 2013.

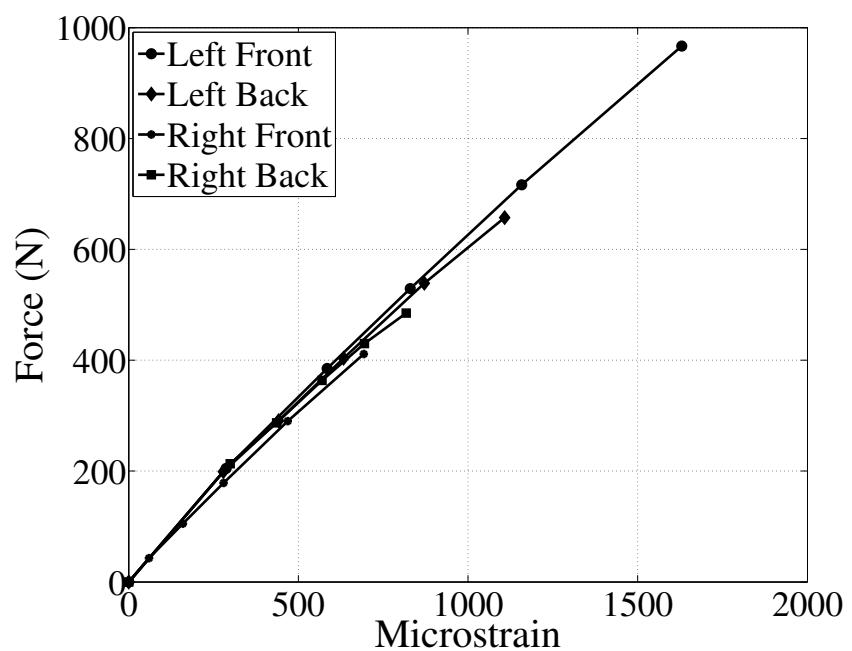


Figure C.14: Calibration curves for the DVC cross load calibration device. Every post has approximately the same calibration value. Graphic ©Seth Gilchrist, 2013.

C.3 Verification of DVC

The DVC algorithm was tested in two ways. First a verification of the algorithm was conducted by synthetically warping an image and then using the DVC algorithm to calculate the known, ground truth strain. The second was a validation of the HR-pQCT in which a bone was scanned twice in the same position under no load and the DVC algorithm was used to measure the strain, which should result in a uniform value of zero strain.

C.3.1 Synthetically strained image

The DVC algorithm was characterized using synthetically strained images. A CT image of a bone sample was deformed using a quadratic deformation, resulting in a linearly increasing strain across the image.

A single human femoral bone specimen was scanned in the Centre for Hip Health and Mobility (CHHM) HR-pQCT (XtremeCT, Scanco, Brüttisellen, Switzerland) at an isotropic resolution of 0.041 mm. Following scanning, a section of cancellous bone was isolated and exported to DICOM format. The DICOM images were read using ITK and synthetically strained using a quadratic deformation, resulting in a linearly increasing strain (Figure C.15). The code used to perform the deformation can be found with the other DVC code in §E.4. Two strain fields were investigated. In the first, strain varied linearly from 0 to -1% strain and in the second, the strain varied linearly from 0 to -5% strain.

The strain measured by the DVC algorithm tracked the applied strain closely (Figure C.16). Analysis of the 0 to -1% image had low strain errors, with an average difference from the applied strain of 0.0047% strain. Analysis of the 0 to -5% image had larger error magnitudes, but the average error was still low (0.057%). Both of these errors are below cancellous bone yield strain (Table 1.5).

Both the -1% and -5% tests showed a sensitivity high enough to detect strains below yield strain in cancellous bone. The -5% tests showed less agreement than the -1% test in the region of overlap, and it is thought that this might be due to the higher strain gradient present in the -

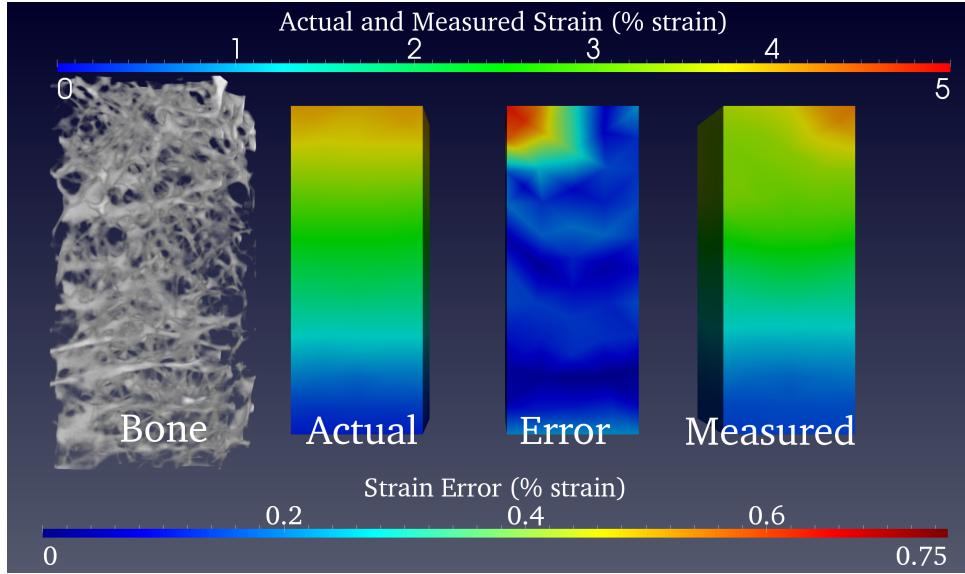


Figure C.15: The linear strain to 5% maximum is shown. The bone image (far left) that was deformed using a quadratic field to produce a linearly increasing strain from 0 to -5% (middle left). The DVC algorithm was used to measure the strain (far right) and the difference between the true strain and the measured strain could be evaluated (middle right). Image ©Seth Gilchrist, 2013.

5% results. The affine transform used by the DVC is linear in nature and would under represent the shape functions required to capture the quadratic deformation used to create the linearly increasing strain. Additionally, the quality of the results decreased at compressive strains of greater than -3%. This could be due to decreased region size in the end of the measurement volume. The DVC algorithm selects a subregion with dimensions set by the user (determined by the TbSp), however, if the desired region is outside of the image, the subregion will be clipped by the image bounds. This reduction in subregion size at the edges of a measurement volume could affect the accuracy of any strain measurements near the edge of the image in a systematic way. It is therefore recommended that analyses be conducted in images that are much larger than the measurement region.

C.3.2 Zero strain image

In this experiment a single human proximal femur was scanned in the HR-pQCT using the same protocol as in §C.3.1. Instead of taking a single image, two images were taken in succes-

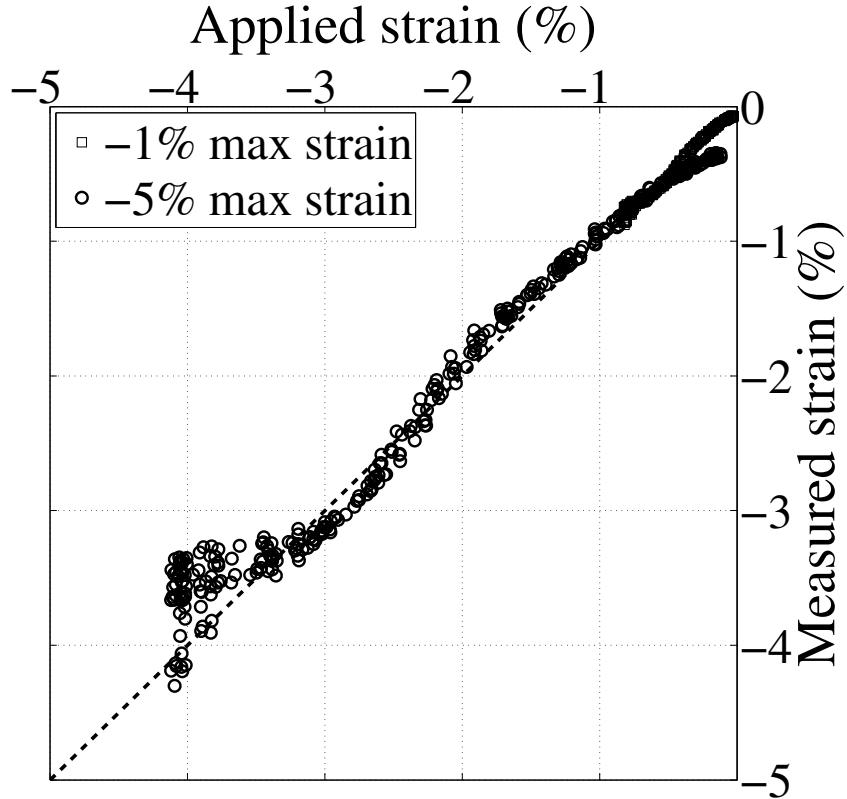


Figure C.16: The compiled results of the synthetic strain test. Two tests are shown here, one with linearly increasing strain from 0 to -1% ($DVC = 1.024 \cdot actual + 0.015$, $R^2 = .988$) and a separate test with linearly increasing strain from 0 to -5% ($DVC = 1.081 \cdot actual + 0.113$, $R^2 = .979$). Graphic ©Seth Gilchrist, 2013.

sion. Overlapping regions of the two images were extracted and saved in DICOM format and analysed using the DVC algorithm.

DVC of the two image should show no strain, however, the results showed a band of non-zero strain running through the middle of the measurement volume (Figure C.17). The majority of the measurement volume showed low strain values (Figure C.18). The bimodal distribution seen in the histogram displays that the majority of the measurement volume enjoyed low error values (median of strains $<0.25\% = 0.12\%$). The central band of high strain created a second, normally distributed peak, with an value of (average (SD)) 0.45(.07%).

The central band of high strain was seen to be on the junction of two HR-pQCT slice blocks. The XtremeCT takes images in blocks of 220 slices when acquiring at 0.041 mm resolution.

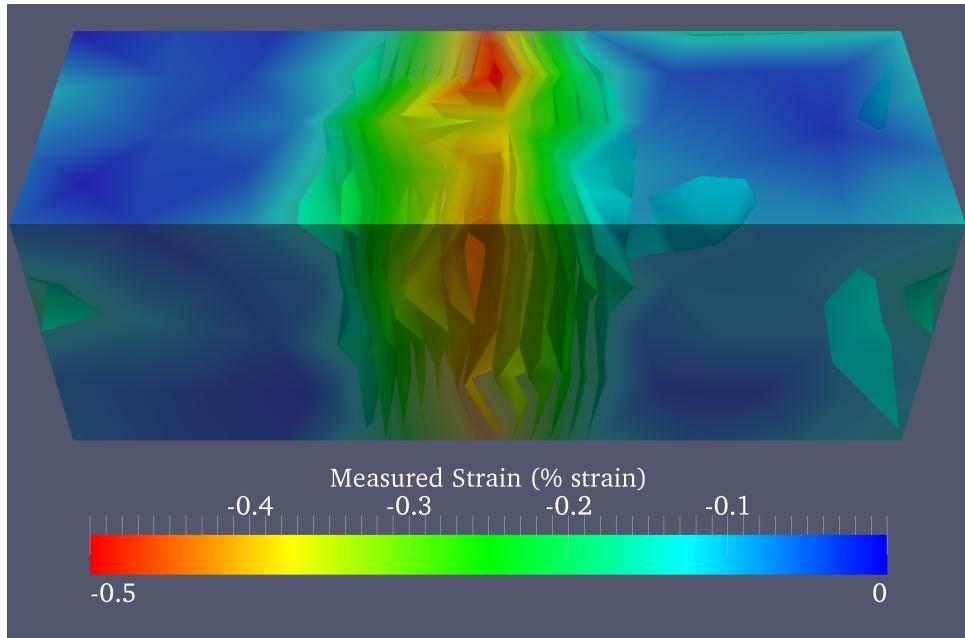


Figure C.17: Two images were taken in succession with no applied strain between them. The resulting DVC analysis showed a band of non-zero strain that was located at a junction of two HR-pQCT slice blocks. Misalignment of the slice blocks in the reconstruction of the CT data resulted in a 0.5 voxel discontinuity in the image, creating the linear strain shown. The image shows the analysis region, rendered transparently, with contours of constant compressive strain greater than -0.15%. Graphic ©Seth Gilchrist, 2013

The blocks of slices are axially registered together during reconstruction using data from the machine stepper motors, which have a finite accuracy. While the accuracy varies, this junction displayed an approximately 0.5 voxel discontinuity, resulting in the elevated stains across it. Indeed, only one of the two images used for this analysis showed evidence of a slice block discontinuity, and other slice block junctions in the same volume did not show such an artefact. The erroneous strains are not limited to the slice block junction for two reasons. First, the mesh nodes are nominally spaced in this analysis by 2 mm, so the discontinuity was between two registration subregion centres, increasing the strain for the FE element defined by the nodes at the centres of the subregions. Secondly, the subregions used for analysis were 2.67 mm on a side, so measurement locations that were up to 1.3 mm away from the image discontinuity could have been affected.

The results of this analysis indicate that the HR-pQCT image acquisition protocol lacks the

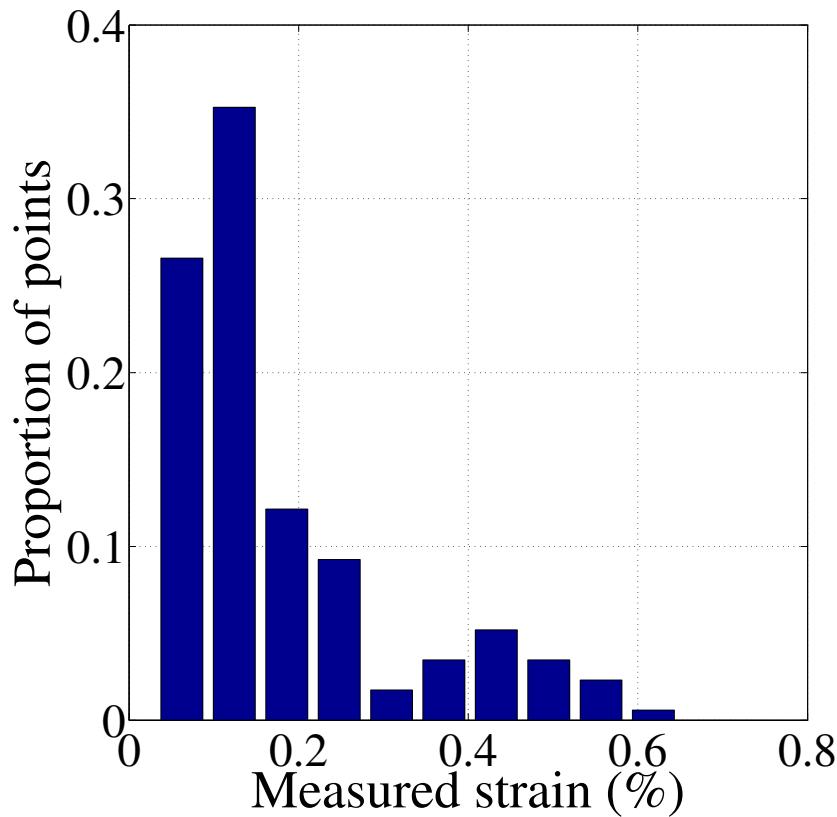


Figure C.18: Results histogram for the zero strain measurement. The results show a bimodal distribution, with the majority of the volume having an error of approximately 0.12%, and a small number of centrally located voxels having an error of 0.45% strain. Graphic ©Seth Gilchrist, 2013.

accuracy to be able to acquire images needed for DVC reliably. However, the errors noted here are not pervasive, and certain analyses could be performed as long as the acquired images are cleared of slice block discontinuities.

C.4 DVC experimental results

Two experiments have been conducted using the DVC algorithm. One experiment used the axial loading apparatus to apply compressive forces to a mouse vertebral body. The second used image data from compression of a bone in a high resolution microCT. The methods and results are detailed here, along with a discussion about the utility and effectiveness of the DVC algorithm.



Figure C.19: The rabbit vertebra in the PMMA potting before testing. The callipers are open 1 cm for scale. Graphic ©Seth Gilchrist, 2013.

C.4.1 Axial compression of a rabbit vertebra

A single, fresh frozen vertebra from a New Zealand white rabbit (female, 5 months) was obtained in conjunction with an ongoing study at the University of British Columbia (UBC). The bone was excised and stored in saline soaked cloth at -20° C until the current study began. It was thawed to room temperature overnight and soft tissues and processes were removed. The bone was potted in PMMA (PMMA, Bosworth Co, Skokie, IL) using the potting device described in §C.2.8 (Figure C.19). Axial loads were applied using the axial loading device detailed in §C.2.8. Force was applied in 180 N increments, and the bone was allowed to relax for 15 min after loading. The bone was then scanned in an HR-pQCT (XtremeCT, Scanco, Brüttisellen, Switzerland) at an isotropic resolution of 0.041 mm.

The bone was segmented from the surroundings using 3DSlicer (3.0, 3DSlicer, <http://www.slicer.org/>) and saved as a stereolithography (stl) file. The stl file was imported into Gmsh

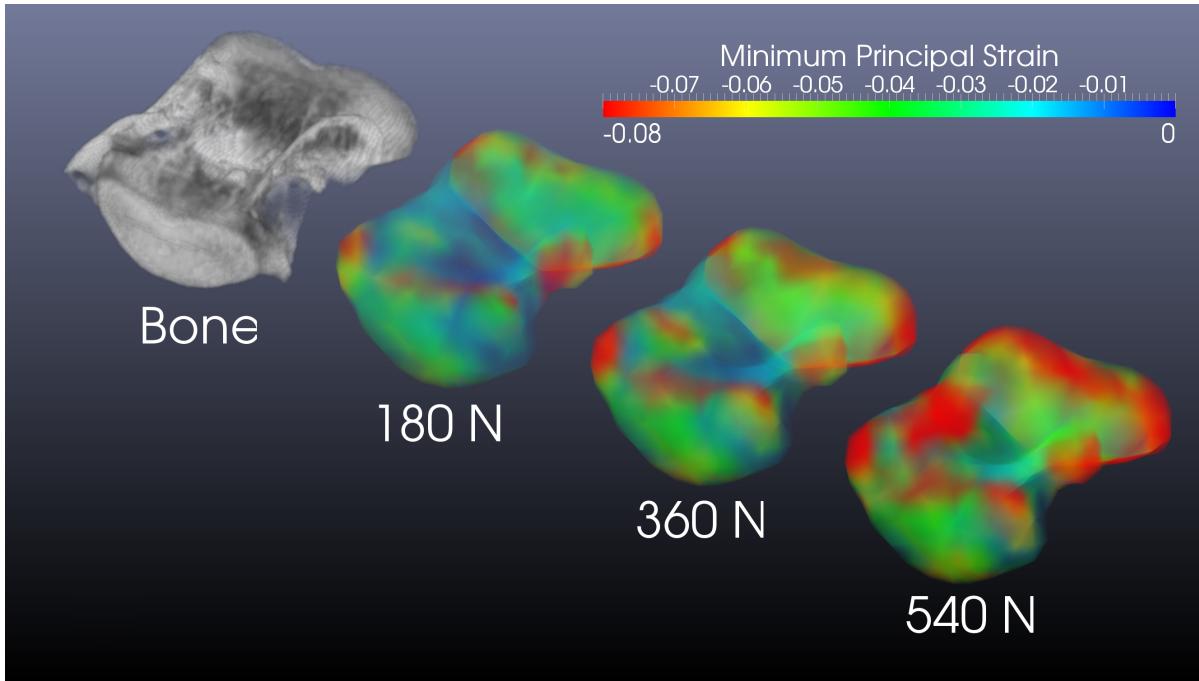


Figure C.20: The compressed rabbit vertebra showed increasing strain as force increased. The highest strain was seen in the caudal end plate. Graphic ©Seth Gilchrist, 2013.

(2.8, Gmsh, <http://geuz.org/gmsh/>) and a 3D, linear tetrahedral mesh with node spacing of 1.6 mm was created. Results were visualized in Paraview (3.98, Paraview, <http://paraview.org/>).

The rabbit vertebra failed between 540 N and 720 N. The strain in the vertebra increased steadily as loads were increased, with the highest strains being measured in the end plates (Figure C.20). The strain in the caudal end plate averaged 10.1% at 540 N. The cranial end plate also showed high strain, but not to the same degree as the caudal one. The specimen fractured in the transverse plane, in what appeared to be a bending mode, in a location of purely cortical bone (Figure C.21).

Extreme strains were measured by the DVC algorithm in the end plate region of the rabbit vertebra. These strain levels were above failure strain for cancellous bone, however, in this specimen the strains appear to be accurate when compared to a 2D approximation obtained from examination of the slice data shown in Figure C.22. New Zealand white rabbits do not reach skeletal maturity until an age of 7–9 months [150], meaning that the end plates in the



Figure C.21: The rabbit vertebra failed in the transverse plane in what appeared to be a bending mode, in predominantly cortical bone. The callipers are open 1 cm. Graphic ©Seth Gilchrist, 2013.

4 month old vertebra used in this experiment would not have fused yet. Direct observation of the CT images showed that the strain was developed by the collapse of a growth plate and the subsequent compression of the soft bone at in the reserve, proliferating and hypertrophic growth plate zones (Figure C.22).

The DVC algorithm also detected high strains in the central portion of the bone, in a region that was predominantly cortical bone. The algorithm isn't sensitive to deformations of cortical bone because the images lack texture for tracking in the cortical bone at the 0.041 mm resolution. Examination of the displacement field showed an erroneous vector in the region of this high strain that was approximately 2.8x the average displacement in the rest of the bone. This single displacement vector was responsible for the region of high strain, and it could be discounted based on its identification.

This tests shows the utility of the DVC algorithm for investigating the internal strain fields of bones loaded under compressive forces. It identified and quantified the strain in the collapse of the growth plate, which was not anticipated. This demonstrates the ability of the DVC algorithm to act as an independent tool, uninformed about anticipated material properties, for

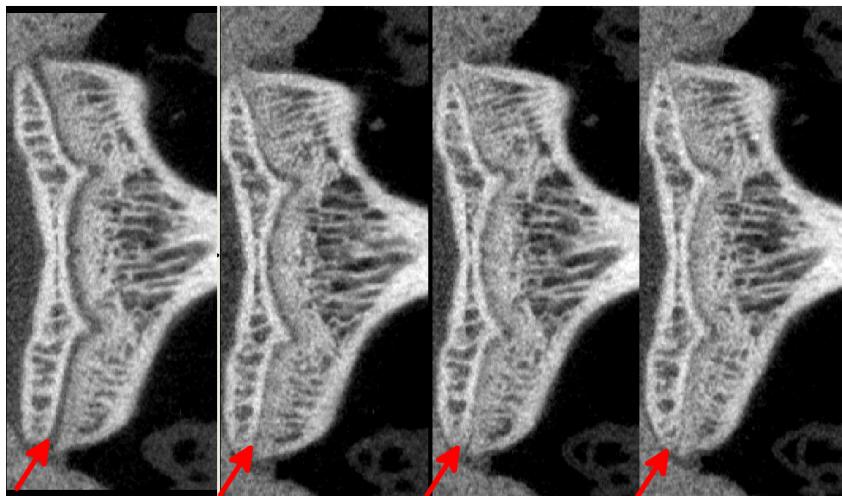


Figure C.22: High strains were developed in the growth plate (red arrows) of the rabbit vertebra. The plate collapsed and then compression of the soft bone allowed for increased strain. Graphic ©Seth Gilchrist, 2013.

measurement of deformation phenomena.

C.4.2 Axial compression of a human bone core

For this experiment, x-ray data were obtained from another lab (Dr. Michael Liebschnre, Baylor Collage of Medicine, Houston, TX) which had performed axial compression of human cancellous bone in a microCT machine at an isotropic resolution of 0.018 mm. The bone was imaged first in an uncompressed state, and then successively imaged at increasing levels of compression. No information was available on the displacement or load levels for each compression.

The specimens were cylindrical, 10 mm tall and 8 mm in diameter. They were loaded axially along their longest dimension. Because the specimens were of simple geometry, no segmentation was required. A cylinder was constructed in Gmsh (2.8, Gmsh, <http://geuz.org/gmsh/>) and meshed using linear, tetrahedral elements, with an node spacing of 2.8 mm. The DVC was conducted using square subregions, 3.6 mm on a side.

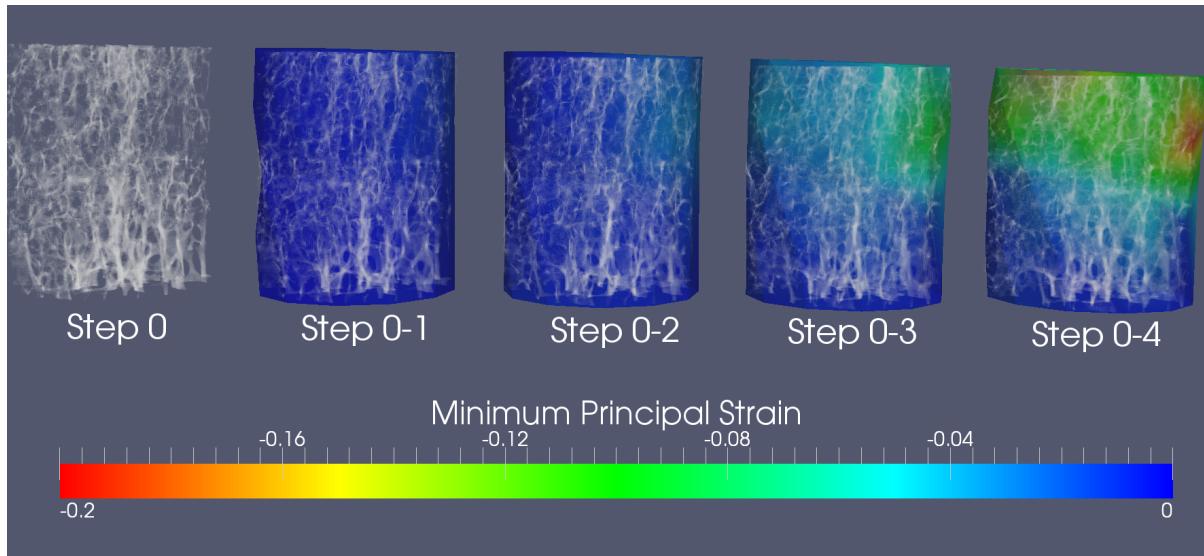


Figure C.23: The human bone did not compress uniformly, but showed high compression near the moving platen, and lower compression at the bottom support. In this graphic, the mesh has been deformed by the measured displacement. Graphic ©Seth Gilchrist, 2013.

The analysis showed a linear deformation with increasing strains in the top of the cylinder with each successive compression (Figure C.23). Additional steps were available, however the DVC began to produce erratic results when the strain magnitudes were above 20%. The calculated nominal strain at step four was 5.7%, with a maximum local strain on the top, side of the cylinder of 22% strain. The average displacement in the first loading step was 0.05 mm, with a calculated range of 0.030–0.099 mm.

The increased strain on the side of the cylinder was observed from the first loading step (not visible in Figure C.23 due to the large colour bar range.), indicating that the specimen was likely either not aligned perfectly to the loading direction or the ends were not milled completely parallel. This was corroborated by the range of measured displacements in the first loading step. The specimen continued to exhibit high strain in the one location, which indicates that the specimen was probably slightly taller in that location, leading to increased strain. Additionally, examination of the bone image in Figure C.23 showed that the location of high strain had fewer trabeculi, indicating that this location may have been weaker, and possibly prone to strain localization before failure of the opposite side. Indeed as the overall

compression increased, we saw a more equitable distribution of the strain in the top of the specimen. If the analysis had continued, it would have been possible to examine the strain relaxation after failure.

High strains in the specimen at steps five and greater lead to erratic strain results. It is possible that this could have been prevented through the use of differential analysis. In the steps up to step four, the deformed bone specimen was compared directly to the undeformed specimen. However, at higher deformations the shape functions provided by the affine transform were not sufficient, and the radius of convergence of even the regular step gradient descent optimizer was not large enough to identify the subregion in the original image. A differential analysis would compare the image at step five with the image at step four using the step 4's results as an initial guess of the solution, and potentially improving the results. This technique is, however, not ideal for DVC because measurement errors in the comparison of, e.g., steps 4 and 5, would be transferred into the measurement of steps 5 and 6, and a general accumulation of error would result. At the time of this experiment, the DVC algorithm did not have this capability. However, it has since been added in the form of a "restart" mesh, i.e., a mesh file that is already populated with deformation data at each node which acts as the starting point for the next registration. While the ability to perform differential DVC has been incorporated, it has not been tested.

This experiment displayed the DVC algorithm's capability to examine the detailed deformation and strain development in bone samples. The identification of a presumed edge loading phenomenon displays the utility of this method as a finite element (FE) validation. The displacement outputs of the DVC algorithm could be used as boundary conditions for an FE analysis, and the resulting strain fields compared. Identification of edge loading, as seen here, would be nearly impossible using typical axial compression techniques. Small changes in the boundary conditions can have a profound influence on the FE calculated strains and independent measures of such as this could have a positive influence on calculation accuracy.

C.5 DVC discussion

The digital volume correlation algorithm presented in this appendix has was developed with the intent that it could be used to assist in the validation of FE analysis results. The algorithm was developed using state of the art, open source image registration techniques, optimized based on the particular requirements of the application. It has been shown to be capable of measuring strains that are well below yield strains of cancellous bone in synthetically deformed images. It is also capable of detecting small strains in human and animal bones, without the need for a priori knowledge of potential deformations or reliance on precise boundary conditions.

Even though these advantages are important, it is also important to note the limitations of the technique. The DVC algorithm is itself a computational technique that cannot be taken as a ground truth solution. A next step would be to use the technique in conjunction with finite element analysis, or more traditional digital image correlation and use a mutual validation method such as proposed by Bland and Altman [22]. Practically, the algorithm has shown issues with measuring extremely high strains, particularly those over 20% in the case of a compressively loaded human bone, and above 4% in a synthetically deformed human bone image. The latter was more likely an issue measuring the quadratic displacement field than with the ability of the algorithm, per se. The application of the algorithm to measurements taken in the XtremeCT, HR-pQCT scanner also showed the potential for large measurement errors due to aspects of how the CT machine performs its reconstructions. While these problems were not pervasive in nature, they have the ability to hamper the use of DVC in larger specimens. An improved scanning technique, in which slice blocks are acquired with overlap and then registered using image, rather than stepper motor, data has been proposed and is in early stages of testing. If successful, this technique could make the use of DVC reliable in the HR-pQCT.

Overall, the DVC technique and its current manifestation presented in this document and detailed in the code given in §E.4 show a high degree of promise. While it is still a research tool that requires an knowledgeable user, the possibility to inform and validate FE analyses using the algorithm makes it a worthwhile tool for continued exploration.

Appendix D

Anatomical correlations in the femur

D.1 Abstract

Femoral version, or twist angle, is an important gross anatomical feature of the femur. It has relevance to mechanical testing, falls analysis and surgery of the proximal femur. In order to determine version using traditional methods, access to the entire bone, either by direct inspection or through imaging, is required. In some cases this is not possible. An investigation of the relationship between femoral version and the linea aspera angle at the shaft 50% length was conducted, searching for a correlation that would allow version determination from the inspection of only the proximal half of the femur. 3D surface scans of 14 femurs were obtained and the version, posterior neck, and linea aspera angles were measured using the digital models. The version and posterior neck angles were found to be moderately correlated ($R^2 = 0.52$), with a regression coefficient near unity. The version and linea aspera angles were found to be moderately correlated ($R^2 = 0.48$), with a regression coefficient of near negative one. These results indicate that an approximation of a femur's version angle can be found by evaluating the angle that separates linea aspera at the 50% length position and the femoral neck. This result has implications in mechanical testing of proximal femora and orthopaedic surgery when the angle of the femoral neck must be determined without access to either the proximal or distal end of the bone.

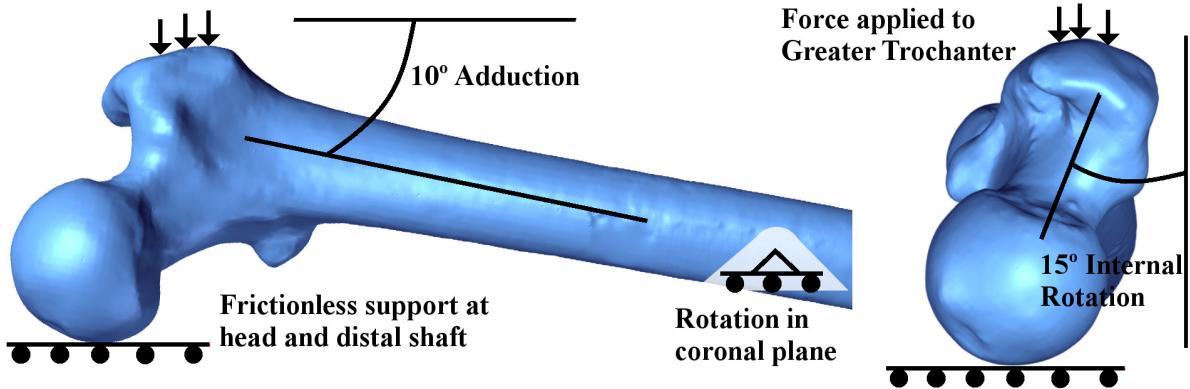


Figure D.1: Constraint and orientation of the proximal femur during mechanical testing. Anterior on the left and superior on the right. Graphic ©Seth Gilchrist, 2013.

D.2 Introduction

Fractures of the proximal femur are common and debilitating injuries. They are associated with a high level of morbidity and mortality and are estimated to cost the US health care system nearly \$2 billion each year [2, 217]. Critical to preventative screening for such injuries is an understanding of the fracture mechanics, which are ideally observed under conditions mimicking those experienced during a real fall to the side.

Previous femoral fracture researchers recognized that one of the most critical aspects the testing was orientation of the femur to the loading vector [7, 71, 189]. It was observed that changing the orientation could lead to changes in fracture load equivalent to multiple decades of bone mineral loss [71, 189]. To control for this, injury biomechanists developed a standard loading orientation which was defined as 10° of adduction of the shaft and 15° of internal rotation of the neck (Fig. D.1) [7, 46, 47, 57, 137, 148].

Standardization of the proximal femur orientation can be thought of as standardizing the fall being modelled. Fall researchers have shown that, in falls to the side, the impact happens at a location that can be characterized with reference to the mid-coronal plane [68]. If the orientation of the leg during a fall is determined by lower limb position, then using the femoral neck to orient the proximal femur during fall simulation would effectively model a different

leg position for each specimen as the angle of the femoral neck varies widely in the general population [227]. This fact presents a potential limitation in study of femoral fractures through laboratory fall simulation. Some specimens may display significantly different fracture characteristics, and researchers currently cannot determine if this was due to the fall being modelled. If the orientation of the femoral neck relative to the coronal plane in life could be ascertained on a specimen-by-specimen basis, a more generalized method for orienting the proximal femur during testing, or identifying bones with extreme anatomical characteristics, might be feasible.

Femoral version is the acute angle made by the axis of the femoral neck with the femoral plane (Figs. 1.3 and 1.4). Since the femoral plane and coronal plane are approximately coincident [152, 153], knowledge of the femoral version could allow researchers greater ability to determine the fall being modelled in each test. Researchers who study proximal femur fracture often work with only the proximal femur and do not have access to the distal end, which is required for determination of femoral version. In these cases, knowledge of the femoral version would need to be determined based solely on proximal femur landmarks. The linea aspera angle was selected as a potential proximal landmark as it is the insertion of both proximal and distal muscles and its location may be dictated by femoral version.

The goal of the current study was to determine if the angle of the femoral neck and the angle of the linea aspera at 50% shaft length could be used to determine femoral version. If the angle of the linea aspera at 50% shaft length is influenced by the femoral version, then knowledge of the angle between the femoral neck and the linea aspera may be able to inform specimen version without access to the femoral condyles. Thus, we hypothesised that the angle between the femoral neck and the linea aspera at 50% shaft length would be correlated with femoral version.

D.3 Materials and methods

15 femurs were obtained from the historical bone collection at UBC, Department of Anatomy. The bones surfaces were scanned using a 3D laser scanner (Vivid i9, Konica-Minolta, Ramsey,

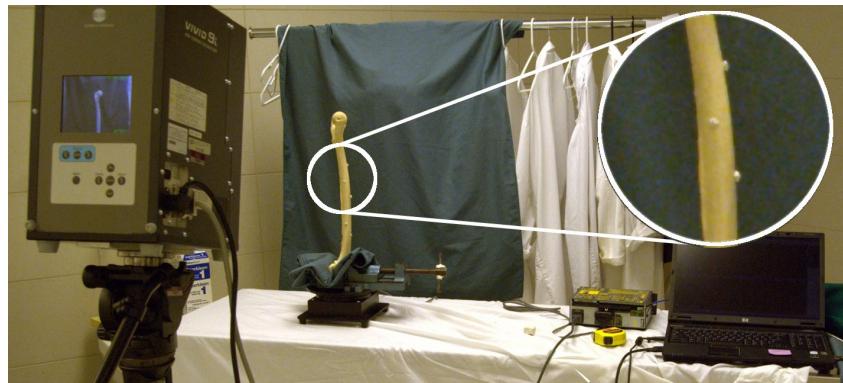


Figure D.2: An image of the scanning set up. The femur is mounted on a software controlled rotary table. Each end of the femur was scanned independently and the scans were aligned with the aid of fiducial markers on the shaft, visible in the insert. Image ©Seth Gilchrist, 2013.

NJ). Because the bones were too long to fit into the field of view of the scanner, they were scanned in two parts, proximal and distal. Fiducial markers made of modelling clay were fixed to the bone surface to aid in registering of the two halves during 3D modelling (Fig. D.2). The bones were mounted on a rotary table and 9 scans in 40° increments were made of each end.

The scans were imported into 3D modelling software (RapidForm XOR3, INUS Technology, Seoul, South Korea) where they were aligned, merged and cleaned. The femoral plane was defined by creating a surface that contacted the posterior femoral condyles distally and the trochanter proximally. The condylar plane as defined as perpendicular to the femoral plane, contacting both inferior condyles, and the third plane was defined as perpendicular to the other two, passing through the femoral head (Fig. 1.3). Measurements of the version angle (A_v) and posterior neck angle (A_{pn}) were made using the femoral plane for reference, while the linea aspera angle (A_{la}) measurements used the third plane for reference.

Version measurement was performed using a technique based on the method developed by Kingsley and Olmsted [118], adapted for the virtual environment (Fig. D.3). To our knowledge, no standard protocols exist for measurement of the posterior neck (Fig. D.5) or linea aspera (Fig. D.4) angles, and as such a new methods were required.

Repeatability was assessed by randomly selecting three specimens for repeated evaluation. Each selected specimen was analysed by the same researcher, three times, separated by

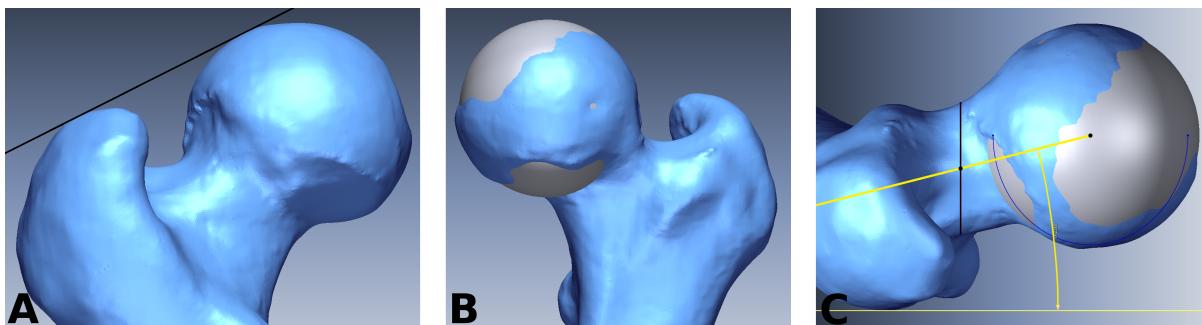


Figure D.3: Version was measured in four steps. A plane perpendicular to the femoral plane, contacting the head and greater trochanter was defined (A). A sphere was fit to the femoral head to locate its centre (B). The view was aligned perpendicular to the plane defined in (A) and two lines were drawn. One was perpendicular to the femoral plane and across the femoral neck. The second was from the centre of the femoral head to the midpoint of the previously drawn line (C). The acute angle from the femoral plane to the second line defined the version of the specimen. Anterior version was taken as positive. Graphic ©Seth Gilchrist, 2013.

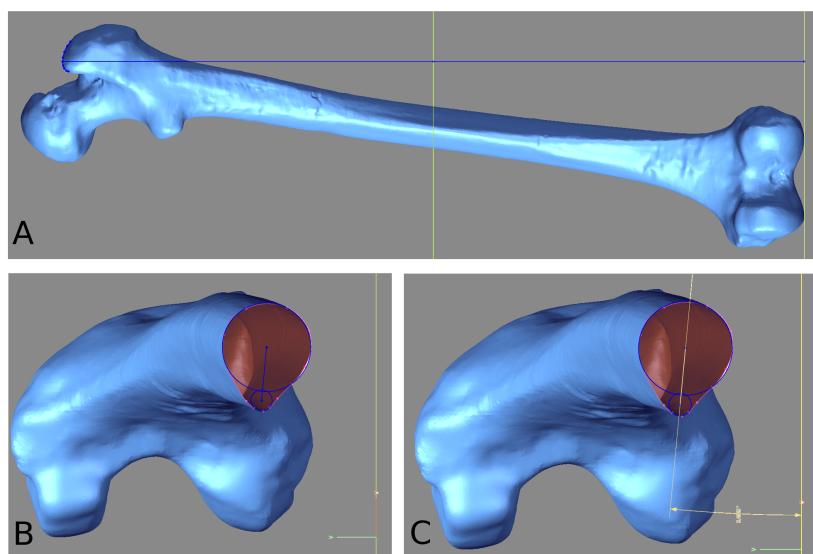


Figure D.4: Linea aspera measurements were conducted in three steps. The proximal end of the femur was masked out at the 50% shaft location (A). Two circles were defined, one was fit to the entire shaft cross section, and the other defined using the posterior prominence of the linea aspera (B). A line was drawn through the centres of both circles, and the acute angle of this line with the third plane was the linea aspera angle (C). Medial rotation was taken as positive. Graphic ©Seth Gilchrist, 2013.

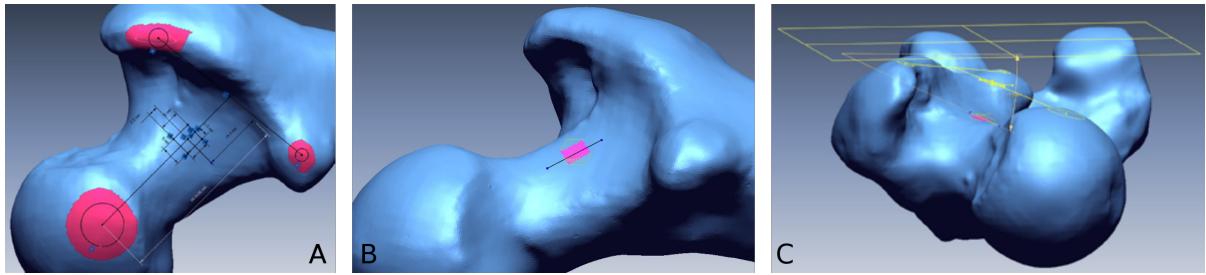


Figure D.5: The posterior neck angle measurement was done in four steps. A plane was defined that touched the posterior head and trochanters. Drawn on this plane was a line that ran from the superior to inferior intertrochanteric crest and as second line that ran from the midpoint of the first line to the centre of the femoral head. A 5x5 mm square was drawn with its centre 1/3rd of the distance from the first line on the second line (A). The square was projected down onto the posterior neck and used to segment a patch of the neck. A surface was fit to this patch, and a line drawn on the surface, parallel to the second line drawn in A (B). The angle between the line in B and the femoral plane defined the posterior neck angle (C).. Graphic ©Seth Gilchrist, 2013.

a minimum of 24 hours. The angles for the specimens were tabulated and a linear regression was performed on A_v vs. A_{pn} , A_v vs. A_{la} and separation angle (A_s) vs. A_v . A_s was defined as the total angular distance between the linea aspera and the femoral neck axis and was calculated using Equation D.1, where the 90° constant was applied because the angles were defined from orthogonal planes.

$$A_s = (A_v + 90^\circ) - A_{la} \quad (\text{D.1})$$

D.4 Results

One specimen was omitted due to degradation of the posterior greater trochanter, making it impossible to define the femoral plane. ***The repeatability study gave an average standard error of the mean of 0.955% (range: 0.59-1.27%), indicating that the method was highly repeatable when conducted by a single observer.***

All three angles showed similar means and standard deviations (SDs) (Table D.1, and Fig. D.6). The posterior neck angle had a regression coefficient of just over unity, and a small

Table D.1: The version, posterior neck and linea aspera angles.

Specimen	Version (degrees)	Posterior Neck (degrees)	Linea Aspera (degrees)
1	15.2	17.8	3.1
2	14.2	11.5	9.5
3	9.4	12.8	7.9
4	12.3	17.4	13.5
5	14.9	12.1	8.5
6	7.7	6.1	24.3
7	9.1	5.7	11.4
8	12.0	20.4	12.7
9	4.3	9.1	21.5
10	14.2	20.6	4.1
11	8.8	7.3	5.0
12	17.1	27.6	4.0
13	20.2	20.4	5.4
14	13.5	22.6	16.4
Average (SD)	12.3 (4.2)	15.1 (6.8)	10.5 (6.6)

offset. In general A_{pn} was a good proxy measurement for femoral version at low angles, but tended to overestimated version at higher angles (Fig. D.7).

The linea aspera angle was inversely correlated with version, meaning that as the femoral neck rotated clockwise around the shaft, the linea aspera rotated counter clockwise (Fig. D.8). This result indicates that the lateral muscles of the leg might have more impact on the location of the linea aspera than those on the medial side. As the version increases, the shaft of the femur would tend to shift posteriorly relative to the acetabulum. If the medial muscles drove the position, then one might expect the linea aspera to move medially, and if the lateral muscles dictate its position, one would expect it to move laterally.

The separation angle between the femoral neck axis and the linea aspera increased as version increased with a slope of about 2:1 (Fig. D.9). This result is a mathematical consequence of the angle between the two being inversely related (Equation D.2). Additionally, because the conversion from linea aspera to separation angle has a higher slope, this compresses the data in the x -direction, which has the effect of increasing the correlation coefficient.

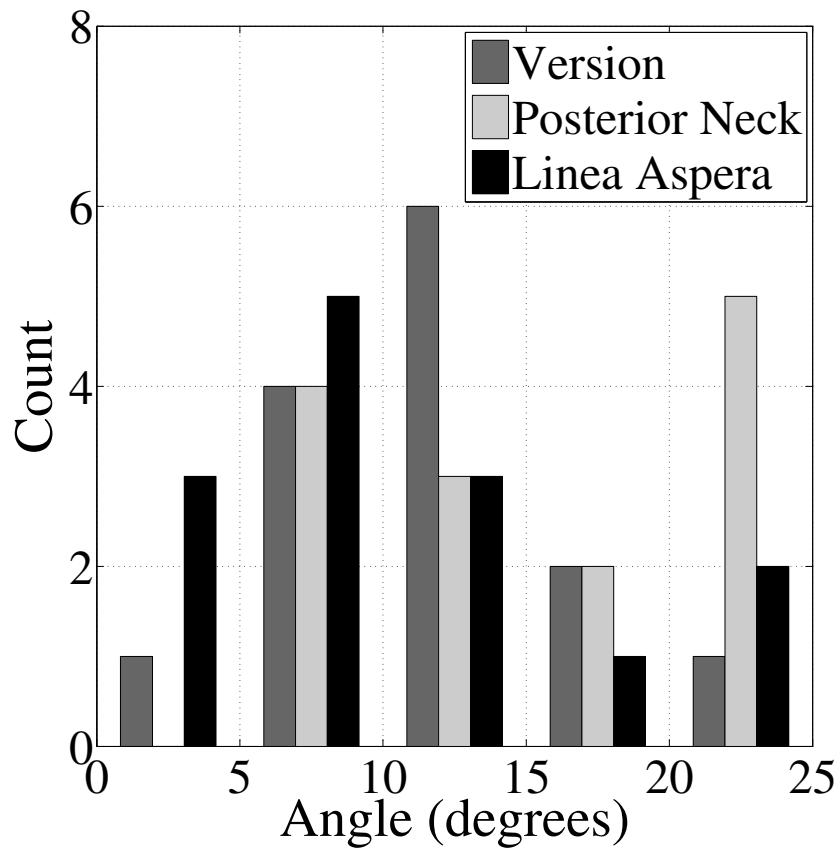


Figure D.6: Distribution of version and linea aspera angles for the 14 specimens.
Graphic ©Seth Gilchrist, 2013.

$$\begin{aligned}
A_{la} &= -1.04 \cdot A_v + 23.3 \\
A_s &= A_v + 90 - A_{la} \\
\Rightarrow A_{la} &= A_v + 90 - A_s \\
\therefore A_v + 90 - A_s &= -1.04 \cdot A_v + 23.3 \\
A_s &= 2.04 \cdot A_v + 66.7
\end{aligned} \tag{D.2}$$

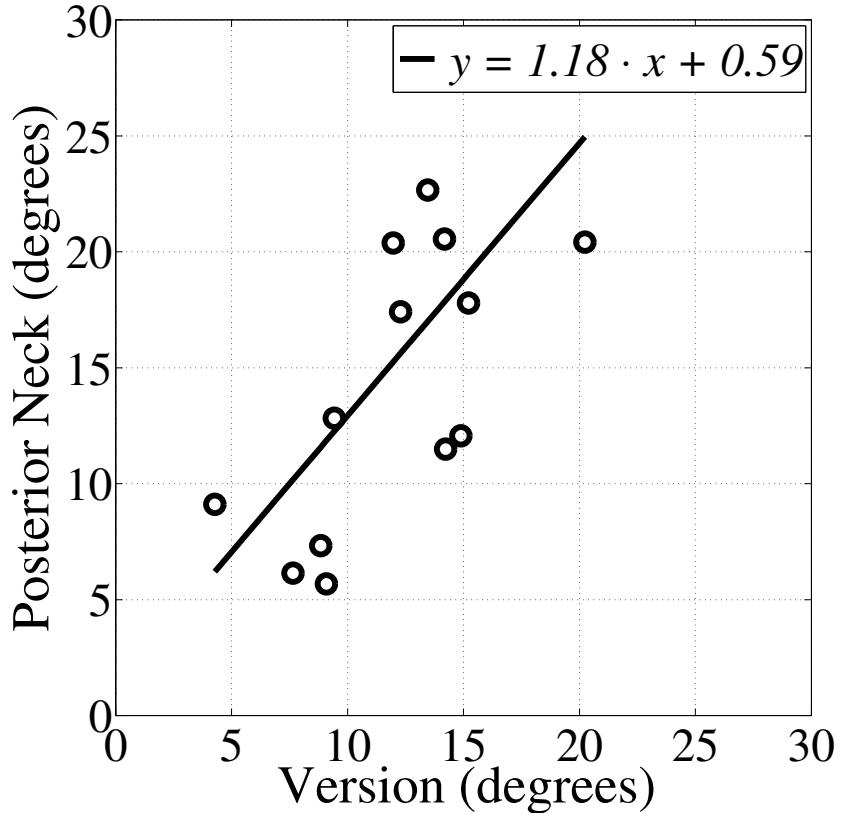


Figure D.7: Posterior neck angle vs. femoral version ($R^2 = 0.52$, $p = 0.004$). Graphic ©Seth Gilchrist, 2013.

D.5 Discussion

This research sought to find a relationship between the linea aspera angle and the femoral version angle. Because the two are inversely related, the separation angle changes as a version changes. These results lead us to reject the null hypothesis and conclude that the version angle can be determined based on the separation angle, allowing researchers to determine a likely value of femoral version by measuring the angle separating the 50% shaft linea aspera and the femoral neck.

Version values measured in our specimen group compare well with the previously published literature. The exact origins of our specimens are unknown due to the University of British Columbia Anatomy department not collecting donor data when the specimens were originally obtained (1950's), but it was reported to the authors that they are believed to be from a South

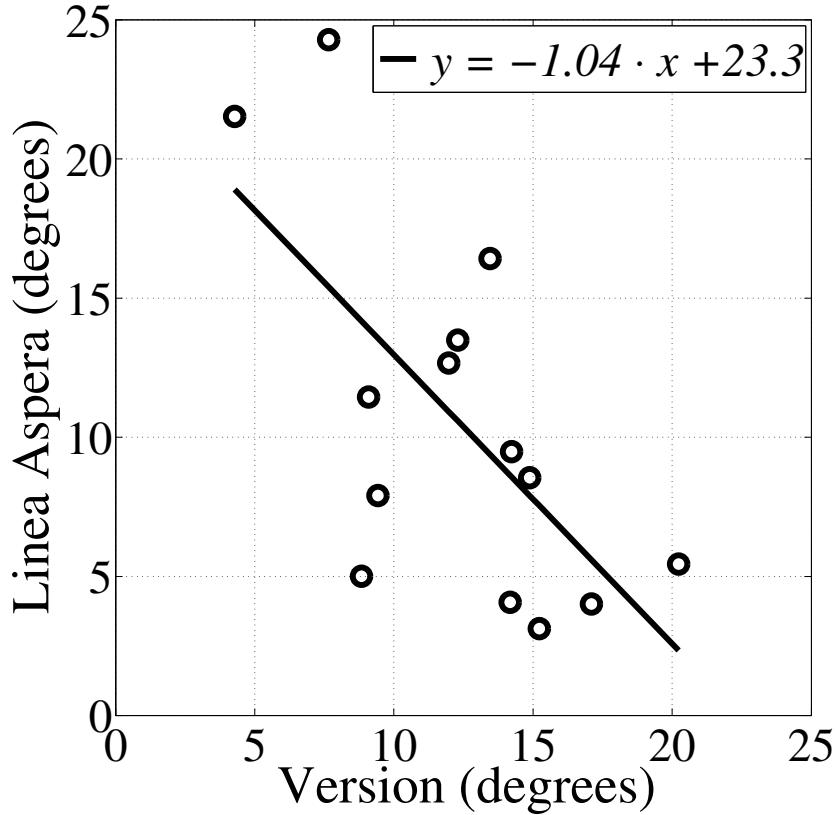


Figure D.8: Femoral version vs. linea aspera angle. ($R^2 = 0.48$, $p = 0.082$). Graphic ©Seth Gilchrist, 2013.

Asian population. Jain et al. [103] measured version angles in 300 South Asian femora and found an average of $8.1^\circ \pm 6.6^\circ$ (range = -17.2° to 36.7°) using the Kingsley-Olmsted method. Khang et al. [116] measured version angles of 200 Korean femora and found an average angle of $17.9^\circ \pm 10.7^\circ$ using CT scan data. Our average version value lies between and within one standard deviation of these studies.

The correlation coefficient of 0.48 does not signify a relationship strong enough to determine one angle based on the value of the other, however, it does allow for stratification of specimens based on normal or extreme version angles. Combining our relationship with the larger bone database analysed by Jain et al. [103], if the separation angle is between 70° and 97° a specimen of South Asian origins can be considered to have a normal version, specimens with separation angles outside of these bounds should be considered to have possible extreme version angles. During mechanical testing of femurs, this approach would allow those speci-

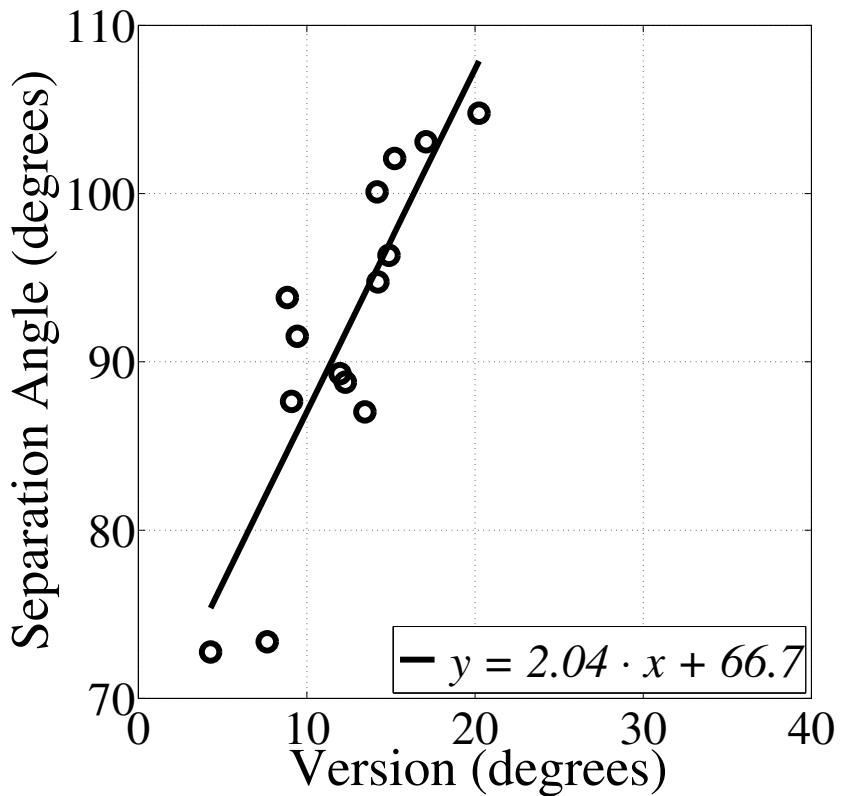


Figure D.9: Femoral version vs. separation angle ($R^2 = 0.74$, $p = 0.0025$). Graphic ©Seth Gilchrist, 2013.

mens thought to have abnormal anatomical characteristics to be identified during analysis.

The use of proximal landmarks to determine femoral version is not limited to mechanical testing of femurs. One could consider using this reference clinically where version needs to be determined in the absence of usual proximal and distal landmarks. This kind of situation could arise in extreme trauma where the proximal or distal ends of both femurs have been damaged and preoperative measurement of contralateral femoral version is not possible. Additionally, in cases of metastatic cancer that require resection of the entire proximal femur, it may be possible to align the femoral arthroplasty component based on the more proximal linea aspera.

While the work provides an insight into the potential to determine femoral version using the linea aspera, there are a few limitations which must be considered before application. First is the limited sample size, with only 14 specimens this study lacks the statistical power needed to provide a robust method for use outside of the research arena. Additionally, the limited

knowledge of the specimen population casts doubt on the applicability of these data to a wider population. However, the consistency of the relationship in this study indicates that it is worth further investigation. A final limitation was that the digital method of version measurement was not compared directly to a photographic method, like that used by Toogood et al. [227] and Kingsley and Olmsted [118]. That said, the method used in this research was modelled directly off the Kingsley-Olmsted method and is functionally the same, representing only a minor modification.

This paper provides data allowing hip fracture researchers to determine if a specimen has a potentially extreme femoral version. This is important because it may change the orientation of the neck to the loading vector during a fall, and may also influence bone remodelling due to different lines of action of forces in hips with extreme version angles. While the authors would not currently recommend altering the orientation used in proximal femur testing, stratification of data based on normal and extreme version could help to identify if the dearth of knowledge of specimen version is limiting our ability to determine what makes a specific femur susceptible to low trauma fracture.

D.6 Acknowledgements

We thank the Centre for Hip Health and Mobility for use of their equipment, specifically the Konica-Minolta Vivid i9 scanner and the University of British Columbia for access to their historical bone collection.

Appendix E

Computer Code

Critical for being able to reproduce the result of any analysis presented in this thesis is knowledge of the computer code used to analyse the data. This section contains the code used to analyse data and create synthetic data sets for the DVC. Each section contains a link to an on-line version of the code, hosted at GitHub (<http://www.github.com>). Any changes made to the code after publication of this thesis will be available on GitHub.

E.1 Drop tower and Instron analysis

This section contains the MatLab code used to analyse the fall simulator and Instron data. All of the code, except for the example input (§E.1.1), can be found at <https://github.com/sethgilchrist/DroptowerInstronAnalysis>.

The tests in the fall simulator and Instron generated huge amounts of data. The organization and analysis of this data was standardized through the use of classes for each dataset, test and specimen. There are eight classes in each experiment (Figure E.1). The Instron test contains a DIC dataset, a DAQ dataset and a link to the specimen associated with the experiment. The drop tower test contains a DIC dataset, a DAQ dataset, a displacement dataset and a link to the specimen associated with the experiment. Note that due to different data begin collected in the Instron DAQ dataset than the drop tower DAQ dataset, these are two different classes.

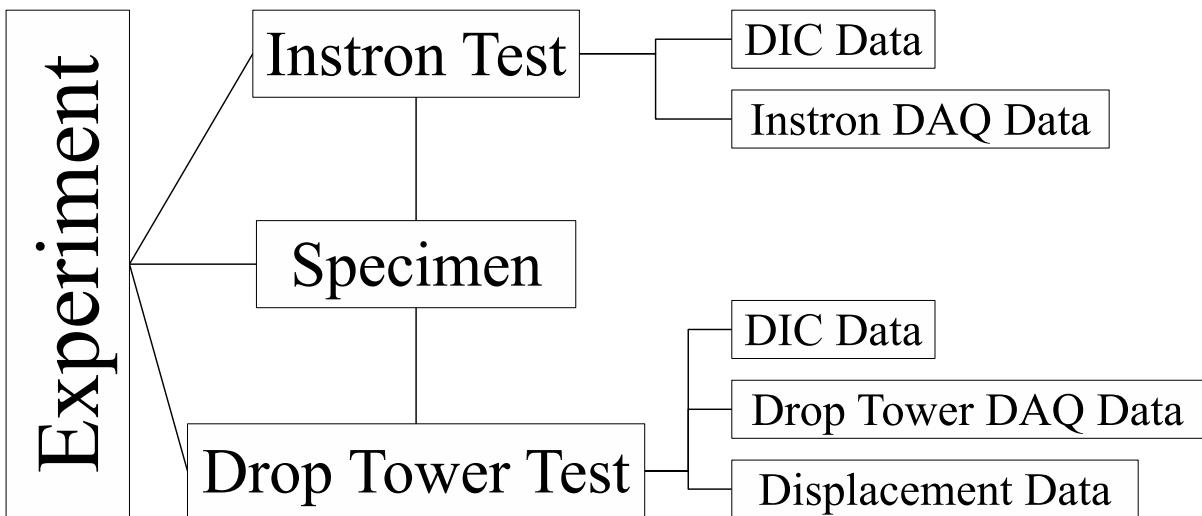


Figure E.1: The hierarchy of the experimental data analysis. Each box is a Matlab class.
Graphic ©Seth Gilchrist, 2013.

The reason for implementing the code in this way was so that all details of a given analysis would be preserved, in one location, for later review. If, for example, one wanted to know the filter order used to prepare the displacement data, this could be found using the command

```
>> experiment.GetDropTower.GetDisplacement.GetFilterOrder().
```

Raw data, as well as analysed data, are also stored in the class. In the end, saving the class object to disk allows for unified saving of all raw data, analysis routines and analysed data in a single *mat*.

The code for each of these data analysis classes are given below. Documentation is built into the MatLab files, so once they are placed in the current MatLab directory or path, help can be found by typing “`help <Class Name>. <Method Name>`” at the command line. Alternatively, all methods are documented in the code.

E.1.1 Example input

Assuming that the data for the desired analysis are stored in “`C:\Data\ TestData\`”, an analysis can be carried out using the following code.

```

1 % Fill in the specimen data
2 name = 'specimen01';
3 gender = 'm';
4 age = 80;

```

```

5 height = 170;
6 weight = 60;
7 dxa = struct('neck',0.67,'troch',.70,'inter',.70,'total',.75,'wards',0);
8 op = 'osteopenia';
9 data = struct('InstronDAQ',1,'InstronDIC',1,'DropTowerDAQ',1,'DropTowerDisplacement',1,
   ↳ 'DropTowerDIC',1);
10
11 % create the specimen object
12 specimen = Specimen(name,gender,age,height,weight,dxa,op,data);
13
14 % create the experiment object. All sub-objects (DAQ analyses, etc) will be created based on the
   ↳ data structure.
15 experiment = Experiment(specimen);
16
17 % Get the Instron analysis
18 instronAnalysis = experiment.GetInstron();
19
20 % Setup the Instron DAQ parameters
21 instronAnalysis.GetDAQData.SetFileName('C:\Data\ TestData\InstronDAQ.csv');
22 instronAnalysis.GetDAQData.ReadFile();
23
24 instronAnalysis.GetDAQData.SetSampleRate(20000);
25 instronAnalysis.GetDAQData.SetFilterCutoff(500);
26 instronAnalysis.GetDAQData.SetGainDisplacement(.35);
27 instronAnalysis.GetDAQData.SetGainLoad(1200);
28
29 % Setup the Instron DIC
30 instronAnalysis.GetDICData.SetFileName('C:\Data\ TestData\InstronDIC.csv');
31 instronAnalysis.GetDICData.ReadFile();
32
33 instronAnalysis.GetDICData.SetSampleRate(100)
34 instronAnalysis.GetDICData.SetStartTime(0.85)
35
36 % Get the drop tower analysis
37 dropTowerAnalysis = experiment.GetDropTower();
38
39 % Setup the drop tower DAQ
40 dropTowerDAQ = dropTowerAnalysis.GetDAQData();
41 dropTowerDAQ.SetFileName('C:\Data\ TestData\DropTowerDAQ.csv');
42 dropTowerDAQ.ReadFile();
43
44 dropTowerDAQ.SetExcitation(12);
45 dropTowerDAQ.SetSampleRate(20000);
46 dropTowerDAQ.SetFilterCutoff(500);
47 dropTowerDAQ.SetFilterOrder(4);
48
49 % Setup the drop tower displacement
50 dropTowerDisp = dropTowerAnalysis.GetDisplacementData();
51 dropTowerDisp.SetFileName('C:\Data\ TestData\DropTowerDAQ.csv');
52 dropTowerDisp.ReadFile();
53
54 dropTowerDisp.SetSampleRate(9600);
55 dropTowerDisp.SetFilterCutoff(500);
56 dropTowerDisp.SetTimeStart(-0.05);
57 dropTowerDisp.SetFilterOrder(2);
58
59 % Setup the drop tower DIC
60 dropTowerDIC = dropTowerAnalysis.GetDICData();
61 dropTowerDIC.SetFileName('C:\Data\ TestData\DropTowerDIC.csv');
62 dropTowerDIC.ReadFile();
63
64 dropTowerDIC.SetStartTime(-.03);
65 dropTowerDIC.SetSampleRate(10000);
66
67 % Run the analysis
68 experiment.update(1);
69
70 % Save the resulting data

```

```
71 save('C:\Data\ TestData\Specimen01.mat','experiment');
```

E.1.2 Specimen class

```
1 classdef Specimen < handle
2     properties (SetAccess = private ,Hidden = true)
3         % immutables properties of each specimen
4         m_specimenName;
5         m_gender;
6         m_dxa;
7         m_opStatus;
8         m_age;
9         m_height;
10        m_weight;
11        m_dataAvailable;
12        % these are bools to say if the data is available
13    end
14    methods
15        function SP = Specimen(name,gender,age,height,weight,dxa,op,data)
16            % The constructor takes the name, DXA values the osteoporosis
17            % status and the available data as inputs.
18            % name is a string
19            % gender is one of: 'm' or 'f' – use 'u' for unknown
20            % age is a number in years – use 0 for unknown
21            % height is a number in cm – use 0 for unknown
22            % weight is a number in kg – use 0 for unknown
23            % dxa is a structure with fields:
24            %     'neck','troch','inter','total','wards'
25            % op is one of:
26            %     'normal','osteopenia','osteoporosis','unknown'
27            % data is a structure of bools indicating what data with
28            % is available. It is comprised of the fields:
29            %     'InstronDAQ','InstronDIC','DropTowerDAQ','DropTowerDisplacement','DropTowerDIC
30            %     ↪
31            % Sp = Specimen([SpecimenName],[gender],[age],[height],[weight],[DXAValues],[OPStatus
32            %     ↪ ],[Data])
33            SP.SetSpecimenName(name);
34            SP.SetDXA(dxa);
35            SP.SetOpStatus(op);
36            SP.SetAge(age);
37            SP.SetHeight(height);
38            SP.SetWeight(weight);
39            SP.SetGender(gender);
40            SP.SetDataAvailable(data);
41        end
42        function SetSpecimenName(SP,name)
43            % A function to set the specimen name.
44            %
45            % SP.SetSpecimenName(name)
46            %
47            if ~ischar(name)
48                error('No specimen name given. Aborting.')
49            end
50            SP.m_specimenName = name;
51        end
52    end
53
54    function SetDXA(SP,dxa)
55        % A function to set the specimen DXA structure in g/cm^2.
56        % The structure must have fields:
```

```

57      % neck
58      % troch
59      % inter
60      % total
61      % wards
62      %
63      % SP.SetDXA(DXA)
64      %
65      if (~isfield(dxa,'neck') || ~isfield(dxa,'troch') || ~isfield(dxa,'inter') || ~
66          ~isfield(dxa,'total') || ~isfield(dxa,'wards') )
67          error('Specimen:dxAFault','Malformed DXA structure. Please check the DXA data for
68              → %s and retry. Aborting.\n',SP.m_specimenName);
69      end
70      SP.m_dxa = dxa;
71  end
72
73  function SetOpStatus(SP,op)
74      % A function to set the osteoporosis state of the specimen.
75      % The value must be one of:
76      % normal
77      % osteopenia
78      % osteoporosis
79      %
80      % SP.SetOpStatus(status)
81      %
82      if (~strcmp(op,'normal') && ~strcmp(op,'osteopenia') && ~strcmp(op,'osteoporosis')
83          → &&~strcmp(op,'unknown'))
84          error('Specimen:opFault','Invalid osteoporosis state specified. Please check the
85              → value for %s and retry. Aborting.\n',SP.m_specimenName);
86      end
87      SP.m_opStatus = op;
88  end
89
90  function SetAge(SP,age)
91      % A function to set the age of the specimen donor in years.
92      % The value must be numeric. Use 0 for unknown.
93      %
94      if (~isnumeric(age))
95          error('Specimen:ageFault','A non-numeric age was specified for %s.\n',SP.
96              → m_specimenName);
97      end
98      SP.m_age = age;
99  end
100
101  function SetHeight(SP,height)
102      % A function to set the height of the specimen donor in cm.
103      % The value must be numeric. Use 0 for unknown.
104      %
105      if (~isnumeric(height))
106          error('Specimen:heightFault','A non-numeric height was specified for %s.\n',SP.
107              → m_specimenName);
108      end
109      SP.m_height = height;
110  end
111
112  function SetWeight(SP,weight)
113      % A function to set the weight of the specimen donor in kg.
114      % The value must be numeric. Use 0 for unknown.
115      %
116      if (~isnumeric(weight))
117          error('Specimen:weightFault','A non-numeric weight was specified for %s.\n',SP.
118              → m_specimenName);

```

```

118         end
119         SP.m_weight = weight;
120     end
121
122     function SetGender(SP,gender)
123         % A function to set the gender of the specimen donor. The
124         % value must be one of:
125         %   m
126         %   f
127         %
128         % SP.SetGender(gender)
129         %
130         if (~strcmp(gender,'m') && ~strcmp(gender,'f') && ~strcmp(gender,'u'))
131             error('Specimen:genderFault','An invalid gender was supplied for %s.\n',SP.
132                 → m_specimenName);
133             end
134             SP.m_gender = gender;
135         end
136
137     function SetDataAvailable(SP,data)
138         % A function to set the data available structure for the
139         % specimen. The structure uses bools to indicated if a certain
140         % type of data is available for analysis. The structure
141         % must have fields:
142         %   InstronDAQ
143         %   InstronDIC
144         %   DropTowerDAQ
145         %   DropTowerDisplacement
146         %   DropTowerDIC
147         %
148         % SP.SetDataAvailable(data)
149         %
150         if (~isfield(data,'InstronDAQ') || ~isfield(data,'InstronDIC') || ~isfield(data,'
151             → DropTowerDAQ') || ~isfield(data,'DropTowerDisplacement') || ~isfield(data,'
152             → DropTowerDIC'))
153             error('Specimen:dataFault','Malformed data available structure. Please check the
154                 → values for %s and retry.\n',SP.m_specimenName);
155         end
156         SP.m_dataAvailable = data;
157     end
158
159     % get functions for each property
160     function o = GetSpecimenName(SP)
161         % A function to get the specimen name.
162         %
163         % Name = SP.GetSpecimenName()
164         %
165         o = SP.m_specimenName;
166     end
167     function o = GetDXA(SP)
168         % A function to get the specimen DXA in g/cm^2. The structure
169         % will have fields:
170         %
171         %   neck
172         %   troch
173         %   inter
174         %   total
175         %   wards
176         %
177         % DXA = SP.GetDXA()
178         %
179         o = SP.m_dxa;
180     end
181     function o = GetOpStatus(SP)
182         % A function to get the specimen osteoporosis state.
183         %
184         % OPStatus = SP.GetOpStatus()
185         %
186         o = SP.m_opStatus;

```

```

182     end
183 function o = GetAge(SP)
184     % A function to get the specimen donor age in years.
185     %
186     % Age = SP.GetAge()
187     %
188     o = SP.m_age;
189 end
190 function o = GetHeight(SP)
191     % A function to get the height of the specimen donor in cm.
192     %
193     % Height = SP.GetHeight()
194     %
195     o = SP.m_height;
196 end
197 function o = GetWeight(SP)
198     % A function to get the weight of the specimen donor in kg.
199     %
200     % Weight = SP.GetWeight()
201     %
202     o = SP.m_weight;
203 end
204 function o = GetGender(SP)
205     % A function to get the gender of the specimen donor. The
206     % value will be one of:
207     %   m
208     %   f
209     %
210     % Gender = SP.GetGender()
211     %
212     o = SP.m_gender;
213 end
214 function o = GetDataAvailable(SP)
215     % A function to get the data available structure. The structure
216     % uses bools to indicate the availability of certain data
217     % sources. The structure will have fields:
218     %   InstronDAQ
219     %   InstronDIC
220     %   DropTowerDAQ
221     %   DropTowerDisplacement
222     %   DropTowerDIC
223     %
224     % Data = SP.GetDataAvailable()
225     %
226     o = SP.m_dataAvailable;
227 end
228
229
230 function PrintSelf(SP)
231     % A function to print the state of the specimen object.
232     %
233     % SP.PrintSelf()
234     %
235     fprintf(1,'%%%%%%%%% Specimen Class Data %%%%%%%%%\n');
236     fprintf(1,'Specimen name: %s\n',SP.GetSpecimenName());
237     fprintf(1,'Specimen donor gender: %s\n',SP.GetGender());
238     fprintf(1,'Specimen donor age: %d years\n',SP.GetAge());
239     fprintf(1,'Specimen donor height: %d cm\n',SP.GetHeight());
240     fprintf(1,'Specimen donor weight: %0.4f kg\n',SP.GetWeight());
241     dxaData = SP.GetDXA();
242     fprintf(1,'Specimen DXA values (g/cm^2):\n\tNeck: %f\n\tTroc: %f\n\tInter: %f\n\t
243     \tTotal: %f\n\tWards: %f\n',dxaData.neck,dxaData.troc,dxaData.inter,dxaData.
244     \ttotal,dxaData.wards);
245     fprintf(1,'Specimen osteoporosis state: %s\n',SP.GetOpStatus());
246     dataAva = SP.GetDataAvailable();
247     fprintf(1,'Specimen data fields available:\n\tInstronDAQ: %d\n\tInstronDIC:
248     \tDropTowerDAQ: %d\n\tDropTowerDisplacement: %d\n\t
249     \tDropTowerDIC: %d\n',dataAva.InstronDAQ,dataAva.InstronDIC,dataAva.

```

```

246     end      → DropTowerDAQ, dataAva.DropTowerDisplacement, dataAva.DropTowerDIC);
247
248 end
249

```

E.1.3 Experiment class

```

1 classdef Experiment < handle
2     properties (SetAccess = private, Hidden = false)
3         % objects for data sets
4         m_specimen;
5         m_dropTower;
6         m_instron;
7     end
8     properties (SetAccess = private, Hidden = true)
9         % analysis properties
10        m_stiffnessDelta;
11        m_energyToForceInstronMaxDelta;
12        m_strainAtForceInstronMaxDelta;
13    end % properties
14
15    methods
16        function EXP = Experiment(specimen)
17            % A constructor for an experiment. Takes a specimen as the
18            % input. See Specimen.m for details on creating a specimen.
19            %
20            % EXP = Experiment(specimen)
21            %
22            EXP.m_specimen = specimen;
23            if (specimen.GetDataAvailable().InstronDAQ || specimen.GetDataAvailable().InstronDIC
24                → )
25                EXP.m_instron = InstronAnalysis(specimen);
26            if (specimen.GetDataAvailable().DropTowerDAQ || specimen.GetDataAvailable().DropTowerDIC
27                → DropTowerDisplacement || specimen.GetDataAvailable().DropTowerDIC)
28                EXP.m_dropTower = DropTowerAnalysis(specimen);
29            end
30        end
31
32        function o = GetSpecimen(EXP)
33            % A function to get the specimen object used to create the
34            % Experiment object
35            %
36            % Specimen = EXP.GetSpecimen()
37            %
38            o = EXP.m_specimen;
39        end
40
41        function o = GetDropTower(EXP)
42            % A function to get the drop tower analysis object.
43            %
44            % DropTowerAnalysis = EXP.GetDropTower()
45            %
46            o = EXP.m_dropTower;
47        end
48
49        function o = GetInstron(EXP)
50            % A function to get the instron analysis object.
51            %
52            % InstronAnalysis = EXP.GetInstron()
53            %
54            o = EXP.m_instron;
55        end
56
57        function o = GetStiffnessDelta(EXP)

```

```

57 % A function to get the difference in stiffness between the
58 % instron and drop tower in N/m, calculated as:
59 %     InstronStiffness - DropTowerStiffness
60 %
61 % Difference = EXP.GetStiffnessDelta()
62 %
63 o = EXP.m_stiffnessDelta;
64 end
65
66 function o = GetEnergyToForceInstronMaxDelta(EXP)
67 % A function to get the difference in energy in J between the
68 % instron and drop tower, calculated as:
69 %     InstronEnergy - DropTowerEnergy
70 %
71 % Difference = EXP.GetEnergyToForceInstronMaxDelta()
72 %
73 o = EXP.m_energyToForceInstronMaxDelta;
74 end
75
76 function o = GetStrainAtForceInstronMaxDelta(EXP)
77 % A function to get the difference in strain at the
78 % max instron force in absolute strain. Uses the strain gauge
79 % on the instron side, and DIC on the drop tower.
80 % The calculation is:
81 %     InstronStrain - DropTowerStrain
82 %
83 % Difference = EXP.GetStrainAtForceInstronMaxDelta()
84 %
85 o = EXP.m_strainAtForceInstronMaxDelta;
86 end
87
88 function Update(EXP, recalcMax)
89 % A function to update the state of the analysis.
90 %
91 % The optional input "recalcMax" is a bool flag to indicate if
92 % the drop tower max force should be recalculated. The default
93 % value is 1 (yes, recalculate)
94 %
95 % EXP.Update(recalcMax)
96 %
97 if nargin<1
98     recalcMax = 1;
99 end
100 ins = EXP.GetInstron();
101 dt = EXP.GetDropTower();
102
103 % update the child objects
104 if ~isempty(ins)
105     ins.Update();
106     insExist = 1;
107     % if the drop tower analysis is present, set the instron max force
108     if ~isempty(dt)
109         dt.SetForceInstronMax(ins.GetForceMax())
110     end
111 else
112     insExist = 0;
113 end
114 if ~isempty(dt)
115     dt.Update(recalcMax);
116     dtExist = 1;
117 else
118     dtExist = 0;
119 end
120
121 % if both stiffnesses are available, calculate the stiffness difference
122 if ( dtExist && insExist )
123     if ( ~isempty(dt.GetStiffness()) && ~isempty(ins.GetStiffness()) )

```

```

125             EXP.CalcStiffnessDelta();
126         end
127     end
128
129     % if both energies to max instron are available , calc difference
130     if ( dtExist && insExist )
131         if ( ~isempty(dt.GetEnergyToForceInstronMax()) && ~isempty(ins.GetEnergy()) )
132             EXP.CalcEnergyDelta();
133         end
134     end
135
136     % if both strains to max instron ara available , calc difference
137     if (dtExist && insExist )
138         if ( ~isempty(dt.GetStrainDICAtForceInstronMax()) && ~isempty(ins.
139             → GetStrainAtMaxGauge()) )
140             EXP.CalcStrainAtForceInstronMaxDelta();
141         end
142     end
143
144     function PrintSelf(EXP)
145         % A function to print the current state of the experimental
146         % analysis.
147         %
148         % EXP.PrintSelf()
149         %
150
151         fprintf(1,'\\n%%%%%%%%% Experiment Class Data %%%%%%%\n');
152         EXP.GetSpecimen().PrintSelf();
153         fprintf(1,'\\n %%% Scalar Members %%%\n');
154         fprintf(1,'Instron - Drop tower stiffness: %f N/m\n',EXP.GetStiffnessDelta());
155         fprintf(1,'Instron - Drop tower energy: %f J\n',EXP.GetEnergyToForceInstronMaxDelta()
156             → );
157         fprintf(1,'Instron - Drop tower strain: %f strain\n',EXP.
158             → GetStrainAtForceInstronMaxDelta());
159
160         if ~isempty(EXP.GetInstron())
161             EXP.GetInstron.PrintSelf();
162         else
163             fprintf(1,'No InstronAnalysis object associated');
164         end
165         if ~ isempty(EXP.GetDropTower())
166             EXP.GetDropTower().PrintSelf();
167         else
168             fprintf(1,'No DropTowerAnalysis object associated');
169         end
170     end % public methods
171
172     methods (Access = private, Hidden = true)
173         function CalcStiffnessDelta(EXP)
174             % A function to calculate the difference in stiffness between
175             % the instron and drop tower tests. The calculation is:
176             %   InstronStiffness - DropTowerStiffness
177             %
178             % EXP.CalcStiffnessDelta()
179             stiffnessInstron = EXP.GetInstron().GetStiffness();
180             stiffnessDropTower = EXP.GetDropTower().GetStiffness();
181
182             EXP.m_stiffnessDelta = stiffnessInstron - stiffnessDropTower;
183         end
184
185         function CalcEnergyDelta(EXP)
186             % A function to calculate the differenc in energy between the
187             % instron and drop tower tests. The calculation is:
188             %   InstronEnergy - DropTowerEnergy
189             %

```

```

190      % EXP.CalcEnergyDelta()
191      %
192      energyInstron = EXP.GetInstron().GetEnergy();
193      energyDropTower = EXP.GetDropTower().GetEnergyToForceInstronMax();
194
195      EXP.m_energyToForceInstronMaxDelta = energyInstron - energyDropTower;
196  end
197
198  function CalcStrainAtForceInstronMaxDelta(EXP)
199      % A function to calculate the difference in strain at the
200      % max instron force in absolute strain. Uses the strain gauge
201      % on the instron side, and DIC on the drop tower.
202      % The calculation is:
203      %   InstronStrain - DropTowerStrain
204      %
205      % EXP.CalcStrainAtForceInstronMaxDelta()
206      %
207      strainInstron = EXP.GetInstron().GetStrainAtMaxGauge();
208      strainDropTower = EXP.GetDropTower().GetStrainDICAtForceInstronMax();
209
210      EXP.m_strainAtForceInstronMaxDelta = strainInstron - strainDropTower;
211  end
212  end % private methods
213 end

```

E.1.4 Instron test class

```

1  classdef InstronAnalysis < handle
2      properties (SetAccess = private , Hidden = false)
3          % members from the specimen
4          m_specimen;
5          % members from the DAQ equipment
6          m_daqData;
7          % members from the dic
8          m_dicData;           % only used if DIC data is available.
9      end
10
11     properties( SetAccess = private , Hidden = true)
12         % machine members
13         m_instronCompliance = 1/30118000; % m/N loading plate compliance
14         m_commonTimeRate = 5000; % Hz
15
16         % result vectors members from interpolation analysis
17         m_time;             % in seconds
18         m_force;            % in newtons, compressive force
19         m_displacementTroc; % in m compression
20         m_displacementPlaten; % in m compression
21         m_compression;       % in m. Specimen compression
22         m_strainGauge; % [Gauge1, Gauge2, Gauge3]
23         m_strainPrincipalGauge; %[P1, P2, Angle]
24         m_strainDIC;
25         m_strainError;
26
27         % results members from analysis
28         m_stiffness;        % in kN/mm
29         m_energyToForceMax; % J
30         m_strainAtMaxDIC;
31         m_strainAtMaxGauge; % in strain, minimum principal stain
32         m_frameAtMax;        % the dic frame at max force
33         m_forceMax;
34         m_timeForceMax;
35         m_indexForceMax;
36         m_strainErrorMean;
37         m_strainErrorStdev;
38     end % properties
39

```

```

40     methods
41         function IA = InstronAnalysis(specimen)
42             % The constructor for the InstronAnalysis class. See
43             % Specimen.m for details.
44             %
45             % IA = InstronAnalysis(specimen)
46             %
47             IA.m_specimen = specimen;
48             if IA.GetSpecimen().GetDataAvailable().InstronDAQ
49                 IA.m_daqData = DAQInstron(specimen);
50             end
51             if IA.GetSpecimen().GetDataAvailable().InstronDIC
52                 IA.m_dicData = DICData(specimen);
53             end
54         end
55
56         function o = GetSpecimen(IA)
57             % A function to get the specimen object
58             %
59             % Specimen = IA.GetSpecimen()
60             %
61             o = IA.m_specimen;
62         end
63
64         function o = GetDICData(IA)
65             % A function to get the DIC data object.
66             %
67             % DICData = IA.GetDICData()
68             %
69             o = IA.m_dicData;
70         end
71
72         function o = GetDAQData(IA)
73             % A function to get the DAQ data object.
74             %
75             % DAQData = IA.GetDAQData()
76             %
77             o = IA.m_daqData;
78         end
79
80         function SetInstronCompliance(IA, compliance)
81             % A function to set the compliance of the testing rig in m/N.
82             % The default value is 1/30118000 m/N.
83             %
84             % Compliance = IA.SetInstronCompliance(compliance)
85             %
86             IA.m_instronCompliance = compliance;
87         end
88
89         function o = GetInstronCompliance(IA)
90             % A function to get the compliance of the instron in m/N.
91             %
92             % Compliance = GetInstronCompliance()
93             %
94             o = IA.m_instronCompliance;
95         end
96
97         function o = GetCommonTimeRate(IA)
98             % A function to get the common time rate in Hz.
99             %
100            % Rate = IA.GetCommonTimeRate()
101            %
102            o = IA.m_commonTimeRate;
103        end
104
105        function SetCommonTimeRate(IA, rate)
106            % A function to set the common time rate in Hz. The default
107            % is 5 kHz.

```

```

108 %
109 % IA.SetCommonTimeRate(rate)
110 %
111 if IA.m_commonTimeRate ~= rate
112     IA.m_commonTimeRate = rate;
113 end
114
115
116 function InterpolateDICToCommonTime(IA)
117     % A function to interpolate the data from the DIC object to the
118     % common time vector.
119 %
120 % IA.InterpolateDICToCommonTime()
121 %
122 IA.m_strainDIC = interp1(IA.GetDICData.GetTime(), IA.GetDICData().GetStrainData(), IA
123 → .GetTime());
124
125
126 function o = GetTime(IA)
127     % A function to get the time vector in seconds.
128 %
129 % Time = IA.GetTime()
130 %
131 if isempty(IA.m_time)
132     IA.CreateCommonTimeVector();
133 end
134 o = IA.m_time;
135
136
137 function o = GetForce(IA)
138     % A function to get the force in the instron analysis time
139     % frame in newtons.
140 %
141 % Force = IA.GetForce()
142 %
143 o = IA.m_force;
144
145
146 function o = GetPrincipalStrainGauge(IA)
147     % A function to get the first principal strain from the gauge
148     % in the instron analysis time frame in absolute strain.
149     % Outputs strain as
150     % [principal1,principal2,angle]
151     % with the strains in absolute and the angle in radians
152 %
153 % PrincipalStrain = IA.GetPrincipalStrainGauge();
154 %
155 o = IA.m_strainPrincipalGauge;
156
157
158 function o = GetStrainGauge(IA)
159     % A function to get the stain gauge data in the instron
160     % anslysis time frame. Output as [gauge1,gauge2,gauue3] in
161     % absolute strain.
162 %
163 % Strain = IA.GetStrainGauge()
164 %
165 o = IA.m_strainGauge;
166
167
168 function o = GetStrainDIC(IA)
169     % A function to get the DIC strain in the instron analysis time
170     % frame in absolute strain.
171 %
172 % Strain = IA.GetStrainDIC()
173 %
174 o = IA.m_strainDIC;
175

```

```

175
176 function o = GetDisplacementTroch(IA)
177 % A function to get the trochanter displacement in mm in the
178 % instron analysis time frame.
179 %
180 % Displacement = IA.GetDisplacementTroch()
181 %
182 o = IA.m_displacementTroch;
183 end
184
185 function o = GetDisplacementPlaten(IA)
186 % A function to get the platen displacement in mm in the
187 % instron analysis time frame.
188 %
189 % Displacement = IA.GetDisplacementPlaten()
190 %
191 o = IA.m_displacementPlaten;
192 end
193
194 function o = GetCompression(IA)
195 % A function to get the specimen compression in mm in the
196 % instron analysis time frame.
197 %
198 % Compression = IA.GetCompression()
199 %
200 o = IA.m_compression;
201 end
202
203 function o = GetForceMax(IA)
204 % A function to get the max force in newtons.
205 %
206 % Force = IA.GetForceMax()
207 %
208 o = IA.m_forceMax;
209 end
210
211 function o = GetTimeForceMax(IA)
212 % A function to get the time of the max force in seconds.
213 %
214 % Time = IA.GetTimeForceMax()
215 %
216 o = IA.m_timeForceMax;
217 end
218
219 function o = GetIndexForceMax(IA)
220 % A function to get the index of the max force.
221 %
222 % Index = IA.GetIndexForceMax()
223 %
224 o = IA.m_indexForceMax;
225 end
226
227 function o = GetIndexAtTime(IA,time)
228 % A function to get the index at a give time in seconds.
229 % Rounds the index down.
230 %
231 % Index = IA.GetIndexAtTime(time)
232 %
233 o = find(IA.GetTime() < time,1,'last');
234 end
235
236 function o = GetStiffness(IA)
237 % A function to get the stiffness of the specimen in N/m.
238 %
239 % Stiffness = IA.GetStiffness()
240 %
241 if isempty(IA.m_stiffness)
242 IA.CalcStiffness();

```

```

243         end
244         o = IA.m_stiffness;
245     end
246
247     function o = GetEnergy(IA)
248         % A function to get the energy during loading in J
249         %
250         % Energy = IA.GetEnergy(IA)
251         %
252         if isempty(IA.m_energyToForceMax())
253             IA.CalcEnergy();
254         end
255         o = IA.m_energyToForceMax;
256     end
257
258     function o = GetStrainAtMaxGauge(IA)
259         % A function to calc the minimum principal strain at the strain
260         % gauge location at the max force. Returns a value that has
261         % been median filtered using a radius of 2.
262         %
263         % Strain = GetStrainAtMaxGauge()
264         %
265         if isempty(IA.m_strainAtMaxGauge)
266             IA.CalcStrainAtMaxGauge();
267         end
268         o = IA.m_strainAtMaxGauge;
269     end
270
271     function o = GetStrainAtMaxDIC(IA)
272         % A function to calc the minimum principal strain at the strain
273         % gauge location at the max force. Returns a value that has
274         % been median filtered using a radius of 2.
275         %
276         % Strain = GetStrainAtMaxGauge()
277         %
278         if isempty(IA.m_strainAtMaxDIC)
279             IA.CalcStrainAtMaxDIC();
280         end
281         o = IA.m_strainAtMaxDIC;
282     end
283
284     function o = GetFrameAtMax(IA)
285         % A function to get the DIC frame at the max force
286         %
287         % Frame = IA.GetFrameAtMax()
288         %
289         if isempty(IA.m_frameAtMax)
290             IA.CalcFrameAtMax();
291         end
292         o = IA.m_frameAtMax;
293     end
294
295     function o = GetStrainError(IA)
296         % A function to get the strain error vector in absolute strain.
297         %
298         % StrainError = IA.GetStrainError()
299         %
300         if isempty(IA.m_strainError)
301             IA.CalcStrainError();
302         end
303         o = IA.m_strainError;
304     end
305
306     function o = GetStrainErrorMean(IA)
307         % A function to get the mean strain error in absolute strain.
308         %
309         % MeanError = IA.GetStrainErrorMean()
310         %

```

```

311         if isempty(IA.m_strainErrorMean)
312             IA.CalcStrainErrorMean();
313         end
314         o = IA.m_strainErrorMean;
315     end
316
317     function o = GetStrainErrorStdev(IA)
318         % A function to get the strain error standard deviation in
319         % absolute strain.
320         %
321         % StrainStdev = IA.GetStrainErrorStdev()
322         %
323         if isempty(IA.m_strainErrorStdev)
324             IA.CalcStrainErrorStdev();
325         end
326         o = IA.m_strainErrorStdev;
327     end
328
329
330
331     function o = GetCompressionAtTime(IA,time)
332         % A function to get the specimen compression in mm at a given time
333         % in seconds. Output is linearly interpolated from the compression
334         % vector.
335         %
336         % Compression = GetCompressionAtTime(time)
337         %
338         o = interp1(IA.GetTime(),IA.GetCompression(),time);
339     end
340
341     function o = GetForceAtTime(IA,time)
342         % A function to get the force in newtons at a give time in
343         % seconds. The output is linearly interpolated from the time
344         % vecvtor.
345         %
346         % Force = IA.GetForceAtTime(time)
347         %
348         o = interp1(IA.GetTime(),IA.GetForce(),time);
349     end
350
351     function Update(IA)
352         % A function to update the state of the Instron analysis. Does
353         % not execute ReadFile() which must be done by the user.
354         %
355         % IA.Update()
356         %
357
358         % Check if DAQ analysis will be done
359         if ~isempty(IA.GetDAQData())
360             % if there is a DAQ data object. Call its update function
361             IA.GetDAQData.Update();
362
363             % next put everything into the common time vector for the
364             % analysis. If there is DIC data it will also be
365             % interpolated into this time space
366             IA.InterpolateDAQToCommonTime();
367
368             % Find the max force and its time and index
369             IA.CalcForceMax()
370             % Find the stiffness
371             IA.CalcStiffness()
372             % Find the energy to max force
373             IA.CalcEnergy()
374             % Find the gauge strain at max force
375             IA.CalcStrainAtMaxGauge()
376         end
377
378         if ~isempty(IA.GetDICData()) % check for DIC data

```

```

379         errorFlag = 0;
380         if ~ischar(IA.GetDICData.GetFileName)
381             warning('InstronAnalysis:FileNameDIC','This error is fatal. No DIC file name
382             ↪ for specimen %s was provided before calling AnalyzeIntronData.\n',IA.
383             ↪ GetSpecimen().GetSpecimenName());
384             errorFlag = errorFlag + 1;
385         end
386         if isempty(IA.GetDICData.GetTimeStart)
387             warning('InstronAnalysis:StartTimeDIC','This error is fatal. No DIC start
388             ↪ time has been set for specimen %s. Without the start time the DIC data
389             ↪ cannot be matched to the DAQ data.\n',IA.GetSpecimen().
390             ↪ GetSpecimenName());
391             errorFlag = errorFlag + 1;
392         end
393         if isempty(IA.GetDICData.GetSampleRate)
394             warning('InstronAnalysis:SampleRateDIC','This error is fatal. No DIC sample
395             ↪ rate has been set for specimen %s. Without this sample rate the DIC
396             ↪ frame corresponding to max force cannot be found.\n',IA.GetSpecimen().
397             ↪ GetSpecimenName());
398             errorFlag = errorFlag + 1;
399         end
400         if errorFlag
401             error('InstronAnalysis:AnalyzeDICData','%d errors were detected when
402             ↪ preparing to analyze the Instron DIC data for specimen %s.\n',
403             ↪ errorFlag,IA.GetSpecimen().GetSpecimenName());
404         end
405
406         % next interpolate the data to the common time vector
407         IA.InterpolateDICToCommonTime()
408     end
409
410     if ~isempty(IA.GetDICData()) && ~ isempty(IA.GetDAQData()) % things that require both
411         ↪ for calculation
412         % calculate the error
413         IA.CalcStrainError()
414         % calculate the mean error
415         IA.CalcStrainErrorMean()
416         % calculate the error standard deviation
417         IA.CalcStrainErrorStdev()
418         % determine the frame at which the max force occurred
419         IA.CalcFrameAtMax()
420         % calculate the strain from the DIC at the max force using
421         % a median filter radius of 2 (the default)
422         IA.CalcStrainAtMaxDIC(2)
423     end
424 end
425
426 function PrintSelf(IA)
427     % A function to print the current state of the Instron analysis
428     % object
429     %
430     % IA.PrintSelf()
431     %
432     fprintf(1,'%%%%%%%%%%%% Instron Analysis Class Data %%%%%%\n');
433     IA.GetSpecimen().PrintSelf();
434
435     fprintf(1,'\n %%%% Instron Analysis Class Parameters %%%%\n');
436     fprintf(1,'Instron compliance: %e m/N\n',IA.GetInstronCompliance());
437     fprintf(1,'Specimen stiffness: %f N/m\n',IA.GetStiffness());
438     fprintf(1,'Maximum force: %f N\n',IA.GetForceMax());
439     fprintf(1,'Time at max force: %f seconds\n',IA.GetTimeForceMax());
440     fprintf(1,'Index at max force: %d\n',IA.GetIndexForceMax());
441     fprintf(1,'Energy to max force: %f J\n',IA.GetEnergy());
442     fprintf(1,'DIC min principal strain at max force: %f strain\n',IA.GetStrainAtMaxDIC()
443             ↪ );
444     fprintf(1,'Gauge min principal strain at max force: %f strain\n',IA.
445             ↪ GetStrainAtMaxGauge());
446     fprintf(1,'DIC frame at max force: %d\n',IA.GetFrameAtMax());

```

```

434     fprintf(1,'DIC min principal strain mean error: %f strain\n',IA.GetStrainErrorMean())
435         ↵ ;
436     fprintf(1,'DIC min principal strain error stdev: %f strain\n',IA.GetStrainErrorStdev
437         ↵ ());
438
439     fprintf(1,'\n %%/% Instron Analysis Data %%/%\n');
440     fprintf(1,'Instron time: [%d,%d] in seconds\n',size(IA.GetTime()));
441     fprintf(1,'Instron force: [%d,%d] in newtons\n',size(IA.GetForce()));
442     fprintf(1,'Instron trochanter displacement: [%d,%d] in m\n',size(IA.
443         ↵ GetDisplacementTroch()));
444     fprintf(1,'Instron platen displacement: [%d,%d] in m\n',size(IA.GetDisplacementPlaten
445         ↵()));
446     fprintf(1,'Instron specimen compression: [%d,%d] in m\n',size(IA.GetCompression()));
447     fprintf(1,'Instron strain gauge: [%d,%d] in strain\n',size(IA.GetStrainGauge()));
448     fprintf(1,'Instron gauge principal strain: [%d,%d] in strain and radians\n',size(IA.
449         ↵ GetPrincipalStrainGauge()));
450     fprintf(1,'Instron DIC principal strain: [%d,%d] in strain\n',size(IA.GetStrainDIC())
451         ↵);
452     fprintf(1,'Instron DIC-Guage strain error: [%d,%d] in strain\n',size(IA.
453         ↵ GetStrainError()));

454 if ~isempty(IA.GetDAQData())
455     IA.GetDAQData().PrintSelf();
456 else
457     fprintf(1,'`%/%/%/%/%/%/% No DAQ Data Available %/%/%/%/%/%`\n');
458 end
459 if ~isempty(IA.GetDICData())
460     IA.GetDICData().PrintSelf();
461 else
462     fprintf(1,'`%/%/%/%/%/%/% No DIC Data Available %/%/%/%/%/%`\n');
463 end
464
465 end % public methods
466 methods (Access = private ,Hidden = true)
467     function CreateCommonTimeVector(IA)
468         % A function to create the common time vector to use in the
469         % instron analysis in seconds.
470         %
471         % IA.CreateCommonTimeVector()
472         %
473         time = IA.GetDAQData.GetTime;
474         IA.m_time = 0.2:1/IA.GetCommonTimeRate():max(time);
475     end
476
477     function InterpolateDAQToCommonTime(IA)
478         % A function to interpolate the data from the DAQ data object
479         % into the common time vector
480         %
481         % IA.InterpolateDAQToCommonTime()
482         %
483         IA.m_force = -interp1(IA.GetDAQData.GetTime(), IA.GetDAQData.GetForce(), IA.GetTime()
484             ↵); % negative to get compressive force
485
486         IA.m_displacementTroch = -interp1(IA.GetDAQData.GetTime(), IA.GetDAQData.
487             ↵ GetDisplacement(), IA.GetTime());
488         IA.m_displacementPlaten = IA.GetForce() .* IA.GetInstronCompliance();
489         IA.m_compression = IA.GetDisplacementTroch() - IA.GetDisplacementPlaten;
490
491         IA.m_strainGauge(:,1) = interp1(IA.GetDAQData.GetTime(), IA.GetDAQData.
492             ↵ GetStrainGauge1(), IA.GetTime());
493         IA.m_strainGauge(:,2) = interp1(IA.GetDAQData.GetTime(), IA.GetDAQData.
494             ↵ GetStrainGauge2(), IA.GetTime());
495         IA.m_strainGauge(:,3) = interp1(IA.GetDAQData.GetTime(), IA.GetDAQData.
496             ↵ GetStrainGauge3(), IA.GetTime());
497
498         IA.m_strainPrincipalGauge(:,1) = interp1(IA.GetDAQData.GetTime(), IA.GetDAQData.
499             ↵ GetPrincipalStrain1(), IA.GetTime());

```

```

488     IA.m_strainPrincipalGauge(:,2) = interp1(IA.GetDAQData.GetTime(), IA.GetDAQData.
489         ↪ GetPrincipalStrain2(), IA.GetTime());
490     IA.m_strainPrincipalGauge(:,3) = interp1(IA.GetDAQData.GetTime(), IA.GetDAQData.
491         ↪ GetPrincipalStrainAngle(), IA.GetTime());
492 end
493
494 function CalcForceMax(IA)
495     % A funtion to find the max force in netons.
496     %
497     % IA.CalcForceMax()
498     %
499     [maxF,maxFI] = max(IA.GetForce());
500     IA.m_forceMax = maxF;
501     time = IA.GetTime();
502     IA.m_timeForceMax = time(maxFI);
503     IA.m_indexForceMax = maxFI;
504 end
505
506 function CalcStiffness(IA)
507     % A function to calculate the stiffness of the specimen. Uses
508     % the 25% and 75% of the max force to calcualte the
509     % stiffness.
510     %
511     % IA.CalcStiffness()
512     %
513     if isempty(IA.GetForceMax())
514         IA.CalcForceMax()
515         warning('InstronAnalysis:ExecutionOrder','Stiffness requested for %s before
516             ↪ calculation of max force.\nMax force calculation being executed now.\n',IA
517             ↪ .GetSpecimen().GetSpecimenName())
518     end
519     % get the second force level for stiffness calculation
520     forceOne = IA.GetForceMax() *.75;
521     forceTwo = IA.GetForceMax() *.25;
522     % get the index for the force at half max force
523     indexForceOne = find(IA.GetForce() > forceOne,1,'first');
524     indexForceTwo = find(IA.GetForce() > forceTwo,1,'first');
525     % get the displacement at max force
526     compression = IA.GetCompression();
527     % get the displacement at force two
528     dispForceOne = compression(indexForceOne);
529     dispForceTwo = compression(indexForceTwo);
530     % calculate the stiffness between force two and force max
531     stiffness = (forceOne - forceTwo)/(dispForceOne - dispForceTwo);
532     % convert to N/m
533     IA.m_stiffness = stiffness;
534 end
535
536 function CalcEnergy(IA)
537     % A function to calculate the energy to the max force
538     %
539     % IA.CalcEnergy()
540     %
541     if isempty(IA.GetForceMax())
542         warning('InstronAnalysis:ExecutionOrder','Energy to max force requested for %s
543             ↪ before calculation of max force.\nMax force calculation being executed now
544             ↪ .\n',IA.GetSpecimen().GetSpecimenName())
545     end
546     % Get the data
547     compression = IA.GetCompression();
548     force = IA.GetForce();
549     % find the valid data
550     % numerically integrate using the valid indexes up to the max
551     % force index using compression in m.
552     IA.m_energyToForceMax = trapz(compression(1:IA.GetIndexForceMax()-1),force(1:IA.
553         ↪ GetIndexForceMax()-1));
554 end

```

```

549
550     function CalcStrainAtMaxGauge(IA, radiusMedianFilter)
551         % A function to calculate the minimum principal strain from the
552         % strain gague at the max force in absolute strain. Uses a
553         % median filter for noise reduction with a default radius of
554         % 2.
555         %
556         % IA.CalcStrainAtMaxGauge(radius(optional))
557         %
558         if nargin < 2
559             radiusMedianFilter = 2;
560         end
561         principal = IA.GetPrincipalStrainGauge();
562
563         IA.m_strainAtMaxGauge = median(principal( IA.GetIndexForceMax() - radiusMedianFilter :
564                                         → IA.GetIndexForceMax() + radiusMedianFilter, 2 ) );
564     end
565
566     function CalcStrainAtMaxDIC(IA, radiusMedianFilter)
567         % A function to calc the DIC strain at the max force. Returns
568         % the value in absolute strain, median filtered using a default
569         % radius of 2. The optional input can change that radius.
570         %
571         % IA.CalcStrainAtMaxDIC(radius(optional))
572         %
573         if nargin < 2
574             radiusMedianFilter = 2;
575         end
576         strain = IA.GetStrainDIC();
577         IA.m_strainAtMaxDIC = median(strain( IA.GetIndexForceMax() - radiusMedianFilter : IA.
578                                         → GetIndexForceMax() + radiusMedianFilter ) );
578     end
579
580     function CalcFrameAtMax(IA)
581         % A function to calculate the DIC frame at the max force.
582         %
583         % IA.CalcFrameAtMax()
584         %
585         if isempty(IA.GetTimeForceMax())
586             error('InstronAnalysis:DataAvailability','DIC frame at max load for %s requested
587                   → before time at max load has been set.\n',IA.GetSpecimen().GetSpecimenName()
588                   → ());
589         end
590         if isempty(IA.GetDICData())
591             error('InstronAnalysis:DataAvailability','DIC frame at max load for %s requested
592                   → when no DIC data is available.\n',IA.GetSpecimen().GetSpecimenName());
593         end
594         IA.m_frameAtMax = floor(( IA.GetTimeForceMax() - IA.GetDICData.GetTimeStart ) * IA.
595                               → GetDICData.GetSampleRate);
596     end
597
598     function CalcStrainError(IA)
599         % A function to calculate the strain error vector, that is
600         % (gauge strain) - (DIC strain) in absolute strain
601         %
602         % IA.CalcStrainError()
603         %
604         if ( isempty(IA.GetPrincipalStrainGauge) || isempty(IA.GetStrainDIC()) )
605             error('InstronAnalysis:DataAvailability','Strain error requested for %s when
606                   → either gauge minimum principal strain or DIC minimum principal strain are
607                   → unavailable.\n',IA.GetSpecimen().GetSpecimenName());
608         end
609         % subtract the strain gauge P2 from StrainDIC for all time
610         principal = IA.GetPrincipalStrainGauge();
611         IA.m_strainError = principal(:,2) - IA.GetStrainDIC();
612     end
613
614     function CalcStrainErrorMean(IA)

```

```

609         % A function to calculate the mean strain error in absolute
610         % strain.
611         %
612         % IA.CalcStrainErrorMean()
613         %
614         if isempty(IA.GetStrainError())
615             error('InstronAnalysis:DataAvailability','Mean strain error requested for %s when
616                 %> strain error vector is unavailable.\n',IA.GetSpecimen().GetSpecimenName()
617                 %> );
618         end
619         % find the last index for which DIC strain is defined and
620         % subtract 1 second to remove spike at end of data
621         strainError = IA.GetStrainError();
622         validData = ~isnan(strainError);
623         lastIndex = IA.GetIndexAtTime(IA.GetTimeForceMax+1.5);
624         IA.m_strainErrorMean = mean(strainError(validData(1:lastIndex)));
625     end
626
627     function CalcStrainErrorStdev(IA)
628         % A function to calculate the strain error standard deviation
629         % in absolute strain
630         %
631         % IA.CalcStrainErrorStdev()
632         %
633         if isempty(IA.GetStrainError())
634             error('InstronAnalysis:DataAvailability','The standard deviation of the strain
635                 %> error requested for %s when strain error vector is unavailable.\n',IA.
636                 %> GetSpecimen().GetSpecimenName());
637         end
638         % find the last index for which DIC strain is defined and
639         % subtract 1 second to remove spike at end of data
640         strainError = IA.GetStrainError();
641         validData = ~isnan(strainError);
642         lastIndex = find(validData == 1,1,'last')-5000;
643         IA.m_strainErrorStdev = std(strainError(validData(1:lastIndex)));
644     end
645
646     end % private methods
647 end % classdef

```

E.1.5 Instron DAQ class

```

1  classdef DAQInstron < handle
2      properties (SetAccess = private , Hidden = false)
3          m_specimen;
4      end
5
6      properties (SetAccess = private , Hidden = true)
7          m_forceDAQVoltage;
8          m_forceDAQ;
9          m_displacementDAQVoltage ;
10         m_displacementDAQ;
11         m_strainGauge1DAQ;
12         m_strainGauge2DAQ;
13         m_strainGauge3DAQ;
14         m_triggerDAQ;
15         m_timeDAQ;
16         m_fileNameDAQ = '';
17
18         % members for filtering and analysis
19         m_sampleRate = 0;           % Hz
20         m_samplePeriod = 0;        % s
21         m_filterCutoff = 0;
22         m_filterOrder = 4;
23         m_gainDisplacement = 0;    % mm/V
24         m_gainLoad = 0;           % N/V

```

```

25
26     % post filtering data
27     m.force;
28     m.displacement;
29     m.strainGauge1;
30     m.strainGauge2;
31     m.strainGauge3;
32     m.strainGaugeP1;
33     m.strainGaugeP2;
34     m.strainGaugePhi;
35     m.trigger;
36     m.time;
37 end % properties
38
39 methods (Access = public)
40     function DI = DAQInstron(specimen)
41         % Constructor for the instron DAQ data class. The single input
42         % is a Specimen data object. See Specimen.m for details on
43         % the specimen data class.
44         %
45         % DI = DAQInstron(specimen)
46         %
47         DI.m.specimen = specimen;
48 end
49
50     function o = GetSpecimen(DI)
51         % A function to get the specimen data object used to construct
52         % the DAQ data object.
53         %
54         % Specimen = DI.GetSpecimen()
55         %
56         o = DI.m.specimen;
57 end
58
59     function SetFileName(DI,file)
60         % A function to set the name of the DAQ data file .
61         %
62         % DI.SetFileName(file)
63         %
64         if ~strcmp(DI.m.fileNameDAQ,file)
65             if ~exist(file , 'file ')
66                 error('DAQInstron:DataAvailability ','The specified instron DAQ file for %s
67                                     does not exist.\n',DI.GetSpecimen().GetSpecimenName());
68             end
69             DI.m.fileNameDAQ = file ;
70         end
71     end
72
73     function o = GetFileName(DI)
74         % A function to get the name of the DAQ data file .
75         %
76         % File = DI.GetFileName()
77         %
78         o = DI.m.fileNameDAQ;
79     end
80
81     function SetSampleRate(DI,rate)
82         % A function to set the DAQ sample rate in Hz. The sampling
83         % period is automatically updated.
84         %
85         % DI.SetSampleRate(rate)
86         %
87         if DI.m.sampleRate ~= rate
88             DI.m.sampleRate = rate;
89             DI.m.samplePeriod = 1/rate;
90         end
91     end
92     function o = GetSampleRate(DI)

```

```

92         % A function to get the DAQ sample rate in Hz
93         %
94         % Rate = DI.GetSampleRate()
95         %
96         o = DI.m_sampleRate;
97     end
98     function SetSamplePeriod(DI,period)
99         % A function to set the DAQ sampling period in seconds. The
100        % sampling rate is automatically updated.
101        %
102        % DI.SetSamplePeriod(period)
103        %
104        if DI.m_samplePeriod ~= period
105            DI.m_samplePeriod = period;
106            DI.m_sampleRate = 1/period;
107        end
108    end
109    function o = GetSamplePeriod(DI)
110        % A function to get the DAQ sampling period in seconds.
111        %
112        % Period = DI.GetSamplePeriod()
113        %
114        o = DI.m_samplePeriod;
115    end
116    function SetFilterCutoff(DI,cutoff)
117        % A function to set the filter cutoff frequency in Hz.
118        %
119        % DI.SetFilterCutoff(cutoff)
120        %
121        if DI.m_filterCutoff ~= cutoff
122            DI.m_filterCutoff = cutoff;
123        end
124    end
125    function o = GetFilterCutoff(DI)
126        % A function to get the filter cutoff frequency in Hz.
127        %
128        % Cutoff = DI.GetFilterCutoff()
129        %
130        o = DI.m_filterCutoff;
131    end
132    function SetFilterOrder(DI,order)
133        % A function to set the filter order. The filter order must
134        % be even due to the use of the filtfilt algorithm. The value
135        % passed to filtfilt will be the specified order/2. Since
136        % filtfilt doubles the order of the filter , the resulting
137        % order will be the same as specified here. If an odd order
138        % is specified , it will be incremented by one.
139        %
140        % DI.SetFilterOrder(order)
141        %
142        if DI.m_filterOrder ~= order
143            if mod(order,2)
144                warning('InstronDAQ:DataValues','The filter order for %s was set to an odd
145                % number. Only even orders are accepted. The order is being increased by
146                % one.\n',DI.GetSpecimen().GetSpecimenName());
147                order = order + 1;
148            end
149            DI.m_filterOrder = order;
150        end
151    function o = GetFilterOrder(DI)
152        % A function to get the filter order.
153        %
154        % Order = DI.GetFilterOrder()
155        %
156        o = DI.m_filterOrder;
157    end
158    function SetGainDisplacement(DI,gain)

```

```

158     % A function to set the displacement gain in mm/V.
159     %
160     % DI.SetGainDisplacement(gain)
161     %
162     if DI.m_gainDisplacement ~= gain
163         DI.m_gainDisplacement = gain;
164     end
165 end
166 function o = GetGainDisplacement(DI)
167     % A function to get the displacement gain in mm/V.
168     %
169     % Gain = DI.GetGainDisplacement()
170     %
171     o = DI.m_gainDisplacement;
172 end
173 function SetGainLoad(DI,gain)
174     % A function to set the load gain in N/V.
175     %
176     % DI.SetGainLoad(gain)
177     %
178     if DI.m_gainLoad ~= gain
179         DI.m_gainLoad = gain;
180     end
181 end
182 function o = GetGainLoad(DI)
183     % A function to get the load gain in N/V.
184     %
185     % Gain = DI.GetGainLoad()
186     %
187     o = DI.m_gainLoad;
188 end
189
190 function ReadFile(DI)
191     % A function to read the file specified by DI.SetFileName(file).
192     %
193     % DI.ReadFile()
194     %
195     if isempty(DI.m_fileNameDAQ)
196         error('InstronDAQ:DataAvailability','File read was called for %s when no file name
197             → was set.\n',DI.GetSpecimen().GetSpecimenName());
198     end
199     % read the file
200     instron = importdata(DI.m_fileNameDAQ,',');
201     % if there was a file header, the result would be a struct. We
202     % want only the data.
203     if isstruct(instron)
204         instron = instron.data;
205     end
206     % put the raw data into the correct vectors
207     DI.m_timeDAQ = instron(:,1);
208     DI.m_forceDAQVoltage = instron(:,6);
209     DI.m_displacementDAQVoltage = instron(:,5);
210     DI.m_triggerDAQ = instron(:,7);
211     DI.m_strainGauge1DAQ = instron(:,2);
212     DI.m_strainGauge2DAQ = instron(:,3);
213     DI.m_strainGauge3DAQ = instron(:,4);
214 end
215
216 function o = GetTime(DI)
217     % A function to get the time vector in seconds, with t = 0
218     % at the time of the trigger.
219     %
220     % Time = DI.GetTime()
221     %
222     if isempty(DI.m_trigger)
223         error('InstronDAQ:DataAvailability','GetTime called for %s before the trigger
224             → data has been set.\nPerhapse call DAQIntron.CalcFilteredData()',DI.
225             → GetSpecimen().GetSpecimenName());

```

```

223     end
224     if isempty(DI.m_time)
225         DI.ZeroTimeAtTrigger();
226     end
227     o = DI.m_time;
228 end
229
230 function o = GetForceVoltage(DI)
231 % A function to get the force voltage read from the input
232 % file in volts.
233 %
234 % Voltage = DI.GetForceVoltage()
235 %
236 o = DI.m.forceDAQVoltage;
237 end
238 function o = GetForceRaw(DI)
239 % A function to get the unfiltered force in newtons.
240 %
241 % Force = DI.GetForceRaw()
242 %
243 o = DI.m.forceDAQ;
244 end
245 function o = GetDisplacementVoltage(DI)
246 % A function to get the displacement voltage read from the
247 % input file in volts.
248 %
249 % Voltage = DI.GetDisplacementVoltage()
250 %
251 o = DI.m.displacementDAQVoltage;
252 end
253 function o = GetDisplacementRaw(DI)
254 % A function to get the unfiltered displacement in mm.
255 %
256 % Displacement = DI.GetDisplacement()
257 %
258 o = DI.m.displacementDAQ;
259 end
260 function o = GetStrainGauge1Raw(DI)
261 % A function to get the unfiltered strain gauge data
262 %
263 % Strain = DI.GetStrainGauge1Raw()
264 %
265 o = DI.m.strainGauge1DAQ;
266 end
267 function o = GetStrainGauge2Raw(DI)
268 % A function to get the unfiltered strain gauge data
269 %
270 % Strain = DI.GetStrainGauge2Raw()
271 %
272 o = DI.m.strainGauge2DAQ;
273 end
274 function o = GetStrainGauge3Raw(DI)
275 % A function to get the unfiltered strain gauge data
276 %
277 % Strain = DI.GetStrainGauge3Raw()
278 %
279 o = DI.m.strainGauge3DAQ;
280 end
281 function o = GetTriggerRaw(DI)
282 % A function to get the unfiltered trigger data. Note that
283 % the trigger is never filtered, but is passed to the output
284 % vector when CalcFilteredData is called internally.
285 %
286 % Trigger = DI.GetTriggerRaw()
287 %
288 o = DI.m.triggerDAQ;
289 end
290 function o = GetTimeRaw(DI)

```

```

291 % A function to get the raw time vector read in from the
292 % input file in seconds. This will not have t = 0 with the
293 % trigger.
294 %
295 % Time = DI.GetTimeRaw()
296 %
297 o = DI.m_timeDAQ;
298 end
299
300 % methods to get the processed data from the class
301 function o = GetForce(DI)
302     % A function to get the processed force data in newtons.
303     %
304     % Force = DI.GetForce()
305     %
306     o = DI.m_force;
307 end
308 function o = GetDisplacement(DI)
309     % A function to get the processed displacement data in mm.
310     %
311     % Displacement = DI.GetDisplacement()
312     %
313     o = DI.m_displacement;
314 end
315 function o = GetStrainGauge1(DI)
316     % A function to get the processed strain data in absolute strain.
317     %
318     % Strain = DI.GetStrainGauge1()
319     %
320     o = DI.m_strainGauge1;
321 end
322 function o = GetStrainGauge2(DI)
323     % A function to get the processed strain data in absolute strain.
324     %
325     % Strain = DI.GetStrainGauge2()
326     %
327     o = DI.m_strainGauge2;
328 end
329 function o = GetStrainGauge3(DI)
330     % A function to get the processed strain data in absolute strain.
331     %
332     % Strain = DI.GetStrainGauge3()
333     %
334     o = DI.m_strainGauge3;
335 end
336 function o = GetPrincipalStrain1(DI)
337     % A function to get the first principal strain in absolute strain.
338     %
339     % Strain = DI.GetPrincipalStrain1()
340     %
341     o = DI.m_strainGaugeP1;
342 end
343 function o = GetPrincipalStrain2(DI)
344     % A function to get the second principal strain in absolute strain.
345     %
346     % Strain = DI.GetPrincipalStrain2()
347     %
348     o = DI.m_strainGaugeP2;
349 end
350 function o = GetPrincipalStrainAngle(DI)
351     % A function to get the principal strain angle in radians
352     % from gauge A as defined in:
353     % Budynas R.G. Advanced Strength and Applied Stress
354     % Analysis, Second ed. McGraw Hill. ISBN 0-07-008985-X
355     %
356     % Strain = DI.GetStrainGaugeAngle()
357     %
358     o = DI.m_strainGaugePhi;

```

```

359     end
360     function o = GetTrigger(DI)
361         % A function to get the trigger data
362         %
363         % Trigger = DI.GetTrigger()
364         %
365         o = DI.m_trigger;
366     end
367
368     function PrintSelf(DI)
369         % A function to print out the data contained in the class
370         %
371         % instance.
372         %
373         % DI.PrintSelf()
374         %
375         fprintf(1,'\\n%%%%% DAQInstron Class Parameters %%%%%%\n');
376         DI.GetSpecimen().PrintSelf();
377         fprintf(1,'\\n %%% Scalar Properties and Results %%%\n');
378         fprintf(1,'DAQ file name: %s\n',DI.GetFileName());
379         fprintf(1,'DAQ sample rate: %f Hz\n',DI.GetSampleRate());
380         fprintf(1,'DAQ sample period: %f seconds\n',DI.GetSamplePeriod());
381         fprintf(1,'DAQ filter cutoff frequency: %f Hz\n',DI.GetFilterCutoff());
382         fprintf(1,'DAQ filter order: %d\n',DI.GetFilterOrder());
383         fprintf(1,'Instron displacement gain %f mm/V\n',DI.GetGainDisplacement());
384         fprintf(1,'Instron load gain %f N/V\n',DI.GetGainLoad());
385
386         fprintf(1,'\\n %%% Raw input data %%% \n');
387         fprintf(1,'DAQ force voltage: [%d,%d] in volts\n',size(DI.GetForceVoltage()));
388         fprintf(1,'DAQ force raw: [%d,%d] in newtons\n',size(DI.GetForceRaw()));
389         fprintf(1,'DAQ displacement voltage: [%d,%d] in volts\n',size(DI.
390             → GetDisplacementVoltage()));
391         fprintf(1,'DAQ displacement raw: [%d,%d] in mm\n',size(DI.GetDisplacementRaw()));
392         fprintf(1,'DAQ strain gauge 1 raw: [%d,%d] in strain\n',size(DI.GetStrainGauge1Raw())
393             → );
394         fprintf(1,'DAQ strain gauge 2 raw: [%d,%d] in strain\n',size(DI.GetStrainGauge2Raw())
395             → );
396         fprintf(1,'DAQ strain gauge 3 raw: [%d,%d] in strain\n',size(DI.GetStrainGauge3Raw())
397             → );
398         fprintf(1,'DAQ trigger raw: [%d,%d] in volts\n',size(DI.GetTriggerRaw()));
399         fprintf(1,'DAQ time raw: [%d,%d] in seconds\n',size(DI.GetTimeRaw()));
400
401         fprintf(1,'\\n %%% Analyzed data %%% \n');
402         fprintf(1,'DAQ force: [%d,%d] in newtons\n',size(DI.GetForce()));
403         fprintf(1,'DAQ displacement: [%d,%d] in mm\n',size(DI.GetDisplacement()));
404         fprintf(1,'DAQ strain gauge 1: [%d,%d] in strain\n',size(DI.GetStrainGauge1()));
405         fprintf(1,'DAQ strain gauge 2: [%d,%d] in strain\n',size(DI.GetStrainGauge2()));
406         fprintf(1,'DAQ strain gauge 3: [%d,%d] in strain\n',size(DI.GetStrainGauge3()));
407         fprintf(1,'DAQ principal strain 1: [%d,%d] in strain\n',size(DI.GetPrincipalStrain1()
408             → ));
409         fprintf(1,'DAQ principal strain 2: [%d,%d] in strain\n',size(DI.GetPrincipalStrain2()
410             → ));
411         fprintf(1,'DAQ principal strain angle: [%d,%d] in radians\n',size(DI.
412             → GetPrincipalStrainAngle()));
413         fprintf(1,'DAQ trigger: [%d,%d] in volts\n',size(DI.GetTrigger()));
414         fprintf(1,'DAQ time: [%d,%d] in seconds\n',size(DI.GetTime()));
415     end
416
417     function Update(DI)
418         % A function to call check if all the required data is available
419         %
420         % and calculate the output data. Does not call ReadFile(),
421         % which must be done by the user.
422         %
423         % DI.Update()
424         %
425
426         % first check if the input data is available
427         errorFlag = 0;
428         %
429         % now check that the data is available

```

```

420     if ~ischar( DI.GetFileName() )
421         warning('DAQInstron:DataAvailability','This error is fatal. No DAQ file name for
422             ↪ specimen %s was provided before calling Update.\n',DI.GetSpecimen());
423             ↪ GetSpecimenName());
424             errorFlag = errorFlag + 1;
425     end
426     if isempty( DI.GetSampleRate() )
427         warning('DAQInstron:DataAvailability','This error is fatal. The sample rate for
428             ↪ the DAQ for specimen %s was not provided before calling Update.\n',DI.
429             ↪ GetSpecimen().GetSpecimenName());
430             errorFlag = errorFlag + 1;
431     end
432     if isempty( DI.GetFilterCutoff() )
433         warning('DAQInstron:DataAvailability','This error is fatal. The filter cutoff for
434             ↪ DAQ filtering for specimen %s was not provided before calling Update.\n',
435             ↪ DI.GetSpecimen().GetSpecimenName());
436             errorFlag = errorFlag + 1;
437     end
438     if isempty( DI.GetGainDisplacement() )
439         warning('DAQInstron:DataAvailability','This error is fatal. The DAQ displacement
440             ↪ gain for specimen %s was not provided before calling Update.\n',DI.
441             ↪ GetSpecimen().GetSpecimenName());
442             errorFlag = errorFlag + 1;
443     end
444     if isempty( DI.GetGainLoad() )
445         warning('DAQInstron:DataAvailability','This error is fatal. The DAQ load gain for
446             ↪ specimen %s was not provided before calling Update.\n',DI.GetSpecimen().
447             ↪ GetSpecimenName());
448             errorFlag = errorFlag + 1;
449     end
450     if errorFlag
451         error('DAQInstron:AnalyzeDAQData','%d errors were detected when preparing to
452             ↪ analyze the Instron DAQ data for specimen %s.\n',errorFlag,IA.GetSpecimen
453             ↪ ().GetSpecimenName());
454     end
455
456     % apply the gains
457     DI.ApplyGainDisplacement();
458     DI.ApplyGainLoad();
459
460     % filter the data
461     DI.CalcFilteredData();
462
463     % calculate the principal strains
464     DI.CalcPrincipalStrains();
465
466     % zero the time at the trigger
467     DI.ZeroTimeAtTrigger();
468
469 end % public methods
470
471 methods (Access = private , Hidden = true)
472     function ApplyGainDisplacement(DI)
473         % A function to apply the gain to the raw displacement vector.
474         % Also sets the initial displacement to zero.
475         %
476         % DI.ApplyGainDisplacement()
477         %

```

```

472     if ~DI.m_gainDisplacement
473         error('InstronDAQ:DataAvailability','Apply displacement gain for %s was attempted
474             ↪ when no gain was set.\n',DI.GetSpecimen().GetSpecimenName());
475     end
476     dispDAQVoltage = DI.m_displacementDAQVoltage;
477     % apply gain and convert to m
478     DI.m_displacementDAQ = ( (DI.m_displacementDAQVoltage - dispDAQVoltage(1)) * DI.
479             ↪ m_gainDisplacement) ./1000;
480 end
481
482 function ApplyGainLoad(DI)
483     % A function to apply the gain to the raw load vector.
484     %
485     % DI.ApplyGainLoad()
486     %
487     if ~DI.m_gainLoad
488         error('InstronDAQ:DataAvailability','Apply load gain for %s was attempted when no
489             ↪ gain was set.\n',DI.GetSpecimen().GetSpecimenName());
490     end
491     DI.m_forceDAQ = DI.m_forceDAQVoltage * DI.m_gainLoad;
492 end
493
494 function CalcFilteredData(DI)
495     % A function to calculate the filtered data. Filtered data
496     % will be passed into the processed data vectors. The trigger
497     % data will be passed without filtering.
498     %
499     % DI.CalcFilteredData()
500     %
501     % design the filter
502     cutoffNormal = DI.m_filterCutoff/DI.m_sampleRate;
503     [b,a] = butter(DI.m_filterOrder/2,cutoffNormal); % divide order by two since filtfilt
504             ↪ doubles the order
505
506     % filter the data
507     if isempty(DI.m_forceDAQ)
508         warning('InstronDAQ:ExecutionOrder','Filtering of DAQ data was requested for %s
509             ↪ before the force gain had been applied. Applying gain now.\n',DI.
510             ↪ GetSpecimen().GetSpecimenName());
511         DI.ApplyGainLoad;
512     end
513     DI.m_force = filtfilt(b,a,DI.m.forceDAQ);
514     if isempty(DI.m_displacementDAQ)
515         warning('InstronDAQ:ExecutionOrder','Filtering of DAQ data was requested for %s
516             ↪ before the displacement gain had been applied. Applying gain now.\n',DI.
517             ↪ GetSpecimen().GetSpecimenName());
518         DI.ApplyGainDisplacement;
519     end
520     DI.m_displacement = filtfilt(b,a,DI.m.displacementDAQ);
521     DI.m_strainGauge1 = filtfilt(b,a,DI.m.strainGauge1DAQ);
522     DI.m_strainGauge2 = filtfilt(b,a,DI.m.strainGauge2DAQ);
523     DI.m_strainGauge3 = filtfilt(b,a,DI.m.strainGauge3DAQ);
524     DI.m_trigger = DI.m.triggerDAQ; % do not filter the
525             ↪ trigger signal
526 end
527
528 function CalcPrincipalStrains(DI)
529     % A function to calculate the principal strains from the
530     % filtered strain data. Must be called after CalcFilteredData()
531     %
532     % DI.CalcPrincipalStrains()
533     %

```

```

528     if ( isempty(DI.m_strainGauge1) || isempty(DI.m_strainGauge2) || isempty(DI.
529         ↪ m_strainGauge3) )
530         error('InstronDAQ:DataAvailability','Principal strains were requested for %s
531             ↪ before all strain data was available.\nPerhaps you should call DAQIntron
532             ↪ .CalcFilteredData() ?',DI.GetSpecimen().GetSpecimenName());
533     end
534     eA = DI.m_strainGauge1;
535     eB = DI.m_strainGauge2;
536     eC = DI.m_strainGauge3;
537     DI.m_strainGaugeP1 = (eA+eC)./2+1/2.*sqrt((eA-eC).^2+(2.*eB-eA-eC).^2);
538     DI.m_strainGaugeP2 = (eA+eC)./2-1/2.*sqrt((eA-eC).^2+(2.*eB-eA-eC).^2);
539     DI.m_strainGaugePhi = 1/2.*atan((eA-2.*eB+eC)./(eA-eC));
540
541     function ZeroTimeAtTrigger(DI)
542         % A function to zero the time at the trigger index
543         %
544         % DI.ZeroTimeAtTrigger()
545         %
546         DI.m_time = DI.m_timeDAQ - DI.m_timeDAQ(find(DI.m_triggerDAQ < 4.9,1,'first'));
547     end
548
549 end % private methods
549 end % classdef

```

E.1.6 Drop tower test class

```

1 classdef DropTowerAnalysis < handle
2     properties (SetAccess = private, Hidden = false)
3         % members for the specimen
4         m_specimen;
5         % members for the DAQ equipment
6         m_daqData;
7         % members for the displacement
8         m_displacementData;
9         % members for the DIC data
10        m_dicData;
11    end
12
13    properties (SetAccess = private, Hidden = true)
14        % machine members
15        m_complianceDropTower = 1/5640000; % (m/N) measured in project 13-009
16        m_massDropTower = 23.18 + 6.419 + 21.36 + 9.94 + (4*.474+2.327+13.656+14.542+.507); % kg,
17            ↪ mass of (Angle platen) + loadcell + t-slot + (DT base/3) + mounting apparatus
18        m_complianceLoadingPlate = 1/30118000; % m/N the loading plate compliance from the
19            ↪ quasistatic testing
20
21        % result vecotrs members
22        m_time;
23        m_forceSix;
24        m_forceOne;
25        m_displacementTroch;
26        m_displacementHammer;
27        m_displacementPlaten;
28        m_compression
29        m_strainGauge; %[gauge1, gauge2, gauge3]
30        m_strainPrincipalGauge; %[P1,P2, angle(rad)]
31        m_strainDIC;
32
33        % results from analysis
34        m_stiffness;
35        m_indexAtImpactStart;
36        m_timeAtImpactStart;
37        % results at max force
38        m_energyToForceMax;

```

```

37     m_forceMax;
38     m_strainDICAtForceMax;
39     m_strainPrincipalGaugeAtForceMax; % given as [P1, P2, angle] in absolute and radians
40     m_frameAtForceMax;
41     m_timeAtForceMax;
42     m_indexAtForceMax;
43     m_compressionAtForceMax;
44     m_rateCompression;
45     m_rateForce;

46 % results for max instron force
47 m_energyToForceInstronMax;
48 m_forcelnstronMax = 0;
49 m_strainDICAtForcelnstronMax;
50 m_strainGaugeAtForcelnstronMax;
51 m_frameAtForcelnstronMax;
52 m_timeAtForcelnstronMax;
53 m_indexAtForcelnstronMax;
54 m_compressionAtForcelnstronMax;

55 % results for finish
56 m_energyToImpactFinish;
57 m_timeAtImpactFinish; % time of the end of the impact. Will be used to calculate total
58 % energy
59 m_indexAtImpactFinish;

60 % others
61 m_timeCommonRate = 100000; %Hz

62 %% To do list:
63 % print self

64 end % properties

65 methods
66 function DA = DropTowerAnalysis(specimen)
67 % Constructor for the drop tower analysis class. Input a
68 % specimen see Specimen.m for details
69 %
70 % DA = DropTowerAnalysis(specimen)
71 %
72 DA.m_specimen = specimen;
73 if DA.GetSpecimen().GetDataAvailable().DropTowerDAQ
74     DA.m_daqData = DAQDropTower(specimen);
75 end
76 if DA.GetSpecimen().GetDataAvailable().DropTowerDisplacement
77     DA.m_displacementData = DTDisplacementData(specimen);
78 end
79 if DA.GetSpecimen().GetDataAvailable().DropTowerDIC
80     DA.m_dicData = DICData(specimen);
81 end
82 end
83 end

84 function o = GetRateCompression(DA)
85 % A function to get the compression rate of the specimen in m/s
86 %
87 % Rate = DA.GetLoadingRate()
88 %
89 o = DA.m_rateCompression;
90 end

91 function o = GetRateForce(DA)
92 % A function to get the loading rate of the specimen in N/s
93 %
94 % Rate = DA.GetRateForce()
95 %
96 if (isempty(DA.m_rateForce))
97     DA.CalcRateForce();
98 end
99
100
101
102
103

```

```

104         end
105
106     o = DA.m_rateForce;
107
108
109     function o = GetComplianceLoadingPlate(DA)
110         % A function to get the compliance in m/N of the loading plate
111         %
112         % Compliance = DA.GetComplianceLoadingPlate()
113         %
114         o = DA.m_complianceLoadingPlate;
115
116     end
117     function SetComplianceLoadingPlate(DA,comp)
118         % A function to set the compliance in m/N of the loading plate
119         %
120         % DA.SetComplianceLoadingPlate(compliance)
121         %
122         if DA.m_complianceLoadingPlate ~= comp
123             DA.m_complianceLoadingPlate = comp;
124         end
125
126     end
127
128     function o = GetCommonTimeRate(DA)
129         % A function to get the common time sample rate in Hz.
130         %
131         % Rate = DA.GetCommonTimeRate()
132         %
133         o = DA.m_timeCommonRate;
134
135     end
136
137     function SetCommonTimeRate(DA,rate)
138         % A function to set the sample rate of the common time vector
139         % in Hz. The default is 100 kHz.
140         %
141         % DA.SetCommonTimeRate(rate)
142         %
143         if DA.GetCommonTimeRate() ~= rate
144             DA.m_timeCommonRate = rate;
145         end
146
147
148     end
149
150     function o = GetCompressionForceMax(DA)
151         % A function to get the compression in m at the max force.
152         %
153         % Compression = DA.GetCompressionForceMax()
154         %
155         o = DA.m_compressionAtForceMax;
156
157
158     end
159
160
161     function o = GetCompressionForceInstronMax(DA)
162         % A function to get the compression in m at the max instron
163         % force.
164         %
165         % Compression = DA.GetCompressionForceInstronMax()
166         %
167         o = DA.m_compressionAtForceInstronMax;
168
169
170     end
171
172     function o = GetSpecimen(DA)
173         % A function that returns the specimen object used to
174         % construct the analysis class.
175         %
176         % Specimen = DA.GetSpecimen()
177         %
178         o = DA.m_specimen;
179
180
181     end
182
183     function o = GetDAQData(DA)

```

```

172 % A function that returns the DAQ data object.
173 %
174 % DAQData = DA.GetDAQData()
175 %
176 if isempty(DA.m_daqData)
177     error('DropTowerAnalysis:DataAvailable','DAQ data for %s was requested when no
178         → valid specimen was set.\n',DA.GetSpecimen().GetSpecimenName());
179 end
180 o = DA.m_daqData;
181 end

182 function o = GetDisplacementData(DA)
183 % A function that returns the displacement data object.
184 %
185 % DisplacementData = DA.GetDisplacementData()
186 %
187 if isempty(DA.m_displacementData)
188     error('DropTowerAnalysis:DataAvailable','Displacement data for %s was requested
189         → when no valid specimen was set.\n',DA.GetSpecimen().GetSpecimenName());
190 end
191 o = DA.m_displacementData;
192 end

193 function o = GetDICData(DA)
194 % A function that returns the DIC data object.
195 %
196 % DICData = DA.GetDICData()
197 %
198 if isempty(DA.m_dicData)
199     error('DropTowerAnalysis:DataAvailable','DIC data for %s was requested when no
200         → valid specimen was set.\n',DA.GetSpecimen().GetSpecimenName());
201 end
202 o = DA.m_dicData;
203 end

204 function SetComplianceDropTower(DA,compliance)
205 % A function to set the drop tower compliance in m/N.
206 %
207 % DA.SetComplianceDropTower(compliance)
208 %
209 if DA.m_complianceDropTower ~= compliance
210     DA.m_complianceDropTower = compliance;
211 end
212 end

213 function o = GetComplianceDropTower(DA)
214 % A function to get the drop tower compliance in m/N.
215 %
216 % Compliance = DA.GetComplianceDropTower()
217 %
218 o = DA.m_complianceDropTower;
219 end

220 function SetMassDropTower(DA,mass)
221 % A function to set the drop tower mass in kg.
222 %
223 % DA.SetMassDropTower(mass)
224 %
225 if DA.m_massDropTower ~= mass
226     DA.m_massDropTower = mass;
227 end
228 end

229 function o = GetMassDropTower(DA)
230 % A function to get the drop tower mass in kg.
231 %
232 % Mass = DA.GetMassDropTower()
233 %
234 o = DA.m_massDropTower;
235 end
236
```

```

237         o = DA.m.massDropTower;
238     end
239
240     function o = GetTime(DA)
241         % A function to get the time vector of the drop tower data
242         % in seconds.
243         %
244         % Time = DA.GetTime()
245         %
246         o = DA.m.time;
247     end
248
249     function o = GetForceSix(DA)
250         % A function to get the six axis load cell force matrix. The
251         % matrix is provided as:
252         % [F_x,F_y,F_z,M_x,M_y,M_z] with forces in N and moments in Nm.
253         %
254         % Force = DA.GetForceSix()
255         %
256         o = DA.m.forceSix;
257     end
258
259     function o = GetForce(DA)
260         % A function to get the axial force trace from the six axis
261         % load cell. This returns the third column of the matrix
262         % returned by GetForceSix, ie the six axis load cell axial
263         % measurement.
264         % The force is in N.
265         %
266         % Force = DA.GetForce()
267         %
268         forceSix = DA.GetForceSix;
269
270         o = forceSix(:,3);
271     end
272
273     function o = GetForceOne(DA)
274         % A function to get the single axial load cell data vector.
275         % The force is in N.
276         %
277         % Force = DA.GetForceOne()
278         %
279         o = DA.m.forceOne;
280     end
281
282     function o = GetDisplacementTroch(DA)
283         % A function to get the displacement of the trochanter in m.
284         %
285         % Displacement = DA.GetDisplacementTroch()
286         %
287         o = DA.m.displacementTroch;
288     end
289
290     function o = GetDisplacementHammer(DA)
291         % A function to get the displacement of the impact hammer in
292         % m.
293         %
294         % Displacement = DA.GetDisplacementHammer()
295         %
296         o = DA.m.displacementHammer;
297     end
298
299     function o = GetDisplacementPlaten(DA)
300         % A function to get the displacement of the lower (head)
301         % platen in m.
302         %
303         % Displacement = DA.GetDisplacementPlaten()
304         %

```

```

305     o = DA.m_displacementPlaten;
306 end
307
308 function o = GetCompression(DA)
309 % A function to get the compression of a specimen in m. This
310 % is the difference between the trochanter displacement and
311 % the platen displacement.
312 %
313 % Compression = DA.GetCompression()
314 %
315 o = DA.m_compression;
316 end
317
318 function o = GetStrainGauge(DA)
319 % A function to get the strain from the strain gauge in
320 % absolute strain. The format is [gauge1, gauge2, gauge3].
321 %
322 % Strain = DA.GetStrainGauge1()
323 %
324 o = DA.m_strainGauge;
325 end
326
327 function o = GetPrincipalStrain(DA)
328 % A function to get the principal strain from the gauge
329 % in absolute strain. The format is:
330 % [Principal-1, Principal-2, Angle in radians]
331 %
332 % Strain = DA.GetPrincipalStrain()
333 %
334 o = DA.m_strainPrincipalGauge;
335 end
336
337 function o = GetStrainDIC(DA)
338 % A function to get the minimum principal strain from the
339 % DIC in absolute strain.
340 %
341 % Strain = GetStrainDIC()
342 %
343 o = DA.m_strainDIC;
344 end
345
346 function o = GetStiffness(DA)
347 % A function to get the stiffness of the specimen in N/m
348 %
349 % Stiffness = DA.GetStiffness()
350 %
351 o = DA.m_stiffness;
352 end
353
354 function o = GetEnergyToForceMax(DA)
355 % A function to get the energy in J to the maximum force in
356 % the drop tower.
357 %
358 % Energy = DA.GetEnergyToForceMax()
359 %
360 o = DA.m_energyToForceMax;
361 end
362
363 function o = GetEnergyToForceInstronMax(DA)
364 % A function to get the energy in J to the maximum force in
365 % instron analysis.
366 %
367 % Energy = DA.GetEnergyToForceInstronMax()
368 %
369 o = DA.m_energyToForceInstronMax;
370 end
371
372 function o = GetEnergyToImpactFinish(DA)

```

```

373 % A function to get the energy in J to the end of the impact.
374 %
375 % Energy = DA.GetEnergyToImpactFinish()
376 %
377 o = DA.m_energyToImpactFinish;
378 end
379
380 function o = GetForceMax(DA)
381 % A function to get the max force in N. This force is taken
382 % from the six axis load cell z-component.
383 %
384 % Force = DA.GetForceMax()
385 %
386 o = DA.m_forceMax;
387 end
388
389 function o = GetForceInstronMax(DA)
390 % A function to get the value of the max instron force in N.
391 %
392 % Force = DA.GetForceInstronMax()
393 %
394 o = DA.m_forceInstronMax;
395 end
396
397 function SetForceInstronMax(DA,force)
398 % A function to set the maximum force from the instron test
399 % in N.
400 %
401 % DA.SetForceInstronMax(force)
402 %
403 if DA.m_forceInstronMax ~= force
404 DA.m_forceInstronMax = force;
405 end
406 end
407
408 function o = GetStrainDICAtForceMax(DA)
409 % A function to get the DIC strain in absolute strain at
410 % the max force, defined using the z-comp of the six axis
411 % load cell. Returns the median strain with a window of 5.
412 %
413 % Strain = DA.GetStrainDICAtForceMax()
414 %
415 o = DA.m_strainDICAtForceMax;
416 end
417
418 function o = GetStrainDICAtForceInstronMax(DA)
419 % A function to get the DIC strain in absolute strain at
420 % the max instron force, defined in the drop tower using the
421 % z-comp of the six axis load cell. Returns the median strain
422 % with a window of 5.
423 %
424 % Strain = DA.GetStrainDICAtForceInstronMax()
425 %
426 o = DA.m_strainDICAtForceInstronMax;
427 end
428
429 function o = GetPrincipalStrainGaugeAtForceMax(DA)
430 % A function to get the gauge principal strain at the max force
431 % defined by the z-comp of the six axis load cell. The strain
432 % is given in a triplet of
433 % [Principal 1, Principal 2, Angle]
434 % with the strains in absolute and the angle in radians.
435 %
436 % Strain = DA.GetPrincipalStrainGaugeAtForceMax()
437 %
438 o = DA.m_strainPrincipalGaugeAtForceMax;
439 end
440

```

```

441 function o = GetPrincipalStrainGaugeAtForceInstronMax(DA)
442 % A function to get the gauge principal strain at the max
443 % instron force force defined by the z-comp of the six axis
444 % load cell. The strain is given in a triplet of
445 % [Principal 1, Principal 2, Angle]
446 % with the strains in absolute and the angle in radians.
447 %
448 % Strain = DA.GetPrincipalStrainGaugeAtForceInstronMax()
449 %
450 o = DA.m_strainGaugeAtForceInstronMax;
451 end
452
453 function o = GetFrameAtForceMax(DA)
454 % A function to get the DIC frame number at max force defined
455 % by the z-comp of the six axis load cell.
456 %
457 % Frame = DA.GetFrameAtForceMax()
458 %
459 o = DA.m_frameAtForceMax;
460 end
461
462 function o = GetFrameAtForceInstronMax(DA)
463 % A function to get the DIC frame number at the max force in
464 % the instron test.
465 %
466 % Frame = DA.GetFrameAtForceInstronMax()
467 %
468 o = DA.m_frameAtForceInstronMax;
469 end
470
471 function o = GetTimeForceInstronMax(DA)
472 % A function to get the time in seconds to the max force
473 % from the instron testing.
474 %
475 % Time = DA.GetTimeForceInstronMax()
476 %
477 o = DA.m_timeAtForceInstronMax;
478 end
479
480 function o = GetTimeForceMax(DA)
481 % A function to get the time in seconds to the max force as
482 % defined by the z-comp of the six axis load cell.
483 %
484 % Time = DA.GetTimeForceMax()
485 %
486 o = DA.m_timeAtForceMax;
487 end
488
489 function o = GetTimeImpactStart(DA)
490 % A function to get the time in seconds at the start of the
491 % impact.
492 %
493 % Time = DA.GetTimeImpactStart()
494 %
495 o = DA.m_timeAtImpactStart;
496 end
497
498 function o = GetTimeImpactFinish(DA)
499 % A function to get the time in seconds at the finish of the
500 % impact.
501 %
502 % Time = DA.GetTimeImpactFinish()
503 %
504 o = DA.m_timeAtImpactFinish;
505 end
506
507 function o = GetIndexForceInstronMax(DA)
508 % A function to get the index at the max instron force.

```

```

509      %
510      % Index = DA.GetIndexForceInstronMax()
511      %
512      o = DA.m_indexAtForceInstronMax;
513  end
514
515  function o = GetIndexForceMax(DA)
516      % A function to get the index at the max force , as determined
517      % using the z-comp of the six axis load cell.
518      %
519      % Index = GetIndexForceMax()
520      %
521      o = DA.m_indexAtForceMax;
522  end
523
524  function o = GetIndexImpactStart(DA)
525      % A function to get the index at the start of the impact.
526      %
527      % Index = GetIndexImpactStart()
528      %
529      o = DA.m_indexAtImpactStart;
530  end
531
532  function o = GetIndexImpactFinish(DA)
533      % A function to get the index at the finish of the impact.
534      %
535      % Index = GetIndexImpactFinish()
536      %
537      o = DA.m_indexAtImpactFinish;
538  end
539
540  function o = GetForceSixCompressionAtTime(DA, t)
541      % A function to get the compressive force from the six axis
542      % load cell at an arbitrary time using interpolation.
543      % The force is given in N.
544      %
545      % Force = DA.GetSixCompressionForceAtTime(time)
546      %
547      forceSix = DA.GetForceSix();
548
549      o = interp1(DA.GetTime(), forceSix(:, 3), t);
550  end
551
552  function Update(DA, recalcMax)
553      % A function to check the data availability and update the
554      % status of all the class properties. Does not call ReadFile()
555      % which must be done by the user. If recalculation of the
556      % maximum force is not required, pass a "0" for recalcMax,
557      % which is an optional variable.
558      %
559      % DA.Update(recalcMax(optional, default = 1))
560      %
561      if nargin < 1
562          recalcMax = 1;
563      end
564      % check for what data we have and what is needed
565
566      % if the DAQ is available , update and import
567      if DA.GetSpecimen().GetDataAvailable().DropTowerDAQ
568          % update the DAQ data
569          DA.GetDAQData().Update();
570          % interpolate the DAQ data
571          DA.InterpolateDAQToCommonTime();
572      end
573
574      % if displacement is available , update and import
575      if DA.GetSpecimen().GetDataAvailable().DropTowerDisplacement
576          % update the displacement data

```

```

577 DA.GetDisplacementData().Update();
578 % interpolate the data
579 DA.InterpolateDisplacementDataToCommonTime()
580 end
581
582 % if DIC is available, import (DIC has no update method)
583 if DA.GetSpecimen().GetDataAvailable().DropTowerDIC
584 DA.InterpolateDICToCommonTime();
585 end
586
587 % calculations based on the DAQ data
588 if DA.GetSpecimen().GetDataAvailable().DropTowerDAQ
589 % calc the displacement of the lower platen
590 DA.CalcDisplacementPlaten();
591 % if the displacement is also available calc compression
592 if DA.GetSpecimen().GetDataAvailable().DropTowerDisplacement
593 DA.CalcCompression();
594 end
595 % if the instron is available get the instron max data
596 if DA.GetSpecimen().GetDataAvailable().InstronDAQ
597 DA.CalcForceInstronMax();
598 % if dic as well, get the frame at max instron
599 if DA.GetSpecimen().GetDataAvailable().DropTowerDIC
600 DA.CalcFrameForceInstronMax();
601 end
602 end
603 % find the max force
604 if recalcMax
605 DA.CalcForceMax();
606 end
607 % find properties at the max force
608 DA.CalcPropertiesForceMax();
609 % find the start and finish of the impact
610 DA.CalcImpactStart();
611 DA.CalcImpactFinish();
612 % if displacement is available calc compression rate to max
613 % force
614 if DA.GetSpecimen().GetDataAvailable().DropTowerDisplacement
615 DA.CalcRateCompression();
616 end
617 % if dic available find frame at the max force
618 if DA.GetSpecimen().GetDataAvailable().DropTowerDIC
619 DA.CalcFrameForceMax();
620 end
621
622 end
623
624 % if the displacement and force are available
625 if (DA.GetSpecimen().GetDataAvailable().DropTowerDisplacement && DA.GetSpecimen().
626     → GetDataAvailable().DropTowerDAQ)
627 if DA.GetSpecimen().GetDataAvailable().InstronDAQ
628 DA.CalcEnergyToForceInstronMax();
629 end
630 % calculate the stiffness
631 DA.CalcStiffness();
632 % calculate the energy to max
633 DA.CalcEnergyToForceMax();
634 % calculate the energy to finish
635 DA.CalcEnergyToImpactFinish();
636 % if the instron force is available, calc energy
637
638 end
639
640 function PrintSelf(DA)
641 % A function to print out the current state of the drop tower
642 % analysis object.
643 %

```

```

644 % DA.PrintSelf()
645 %
646 fprintf(1,'%%%%%%%%% DropTowerAnalysis Class Parameters %%%%%%%%%\n');
647 DA.GetSpecimen().PrintSelf();
648 fprintf(1,'\\n %%%% Scalar Members and Properties %%%%\n');
649 fprintf(1,'Machine compliance: %e m/N\n',DA.GetComplianceDropTower());
650 fprintf(1,'Loading plate compliance: %e m/N\n',DA.GetComplianceLoadingPlate());
651 fprintf(1,'Mass drop tower platen: %f kg\n',DA.GetMassDropTower());
652 fprintf(1,'Frequency of common time: %f Hz\n',DA.GetCommonTimeRate());
653 fprintf(1,'Max Instron force: %f N\n',DA.GetForceInstronMax());
654
655 fprintf(1,'\\n %%%% Scalar Results %%%% \\n');
656 fprintf(1,'Stiffness: %f N/m\n',DA.GetStiffness());
657 fprintf(1,'Max force: %f N\n',DA.GetForceMax());
658 fprintf(1,'Compression at max force: %f m\n',DA.GetCompressionForceMax());
659 fprintf(1,'Compression at max instron force: %f m\n',DA.GetCompressionForceInstronMax
       → ());
660 fprintf(1,'Compression rate of the specimen: %f m/s\n',DA.GetRateCompression());
661 fprintf(1,'Energy to max force: %f J\n',DA.GetEnergyToForceMax());
662 fprintf(1,'Energy to max instron force: %f J\n',DA.GetEnergyToForceInstronMax());
663 fprintf(1,'Energy to impact finish: %f J\n',DA.GetEnergyToImpactFinish());
664 fprintf(1,'Time of impact start: %f seconds\n',DA.GetTimeImpactStart());
665 fprintf(1,'Time of max force: %f seconds\n',DA.GetTimeForceMax());
666 fprintf(1,'Time of max instron force %f seconds\n',DA.GetTimeForceInstronMax());
667 fprintf(1,'Time of impact finish: %f seconds\n',DA.GetTimeImpactFinish());
668 fprintf(1,'Index of impact start: %d\n',DA.GetIndexImpactStart());
669 fprintf(1,'Index of max force: %d\n',DA.GetIndexForceMax());
670 fprintf(1,'Index of max instron force: %d\n',DA.GetIndexForceInstronMax());
671 fprintf(1,'Index of impact finish: %d\n',DA.GetIndexImpactFinish());
672 fprintf(1,'Principal strain at max force:\n\t%15f strain,\n\t%15f strain,\n\t%15f
       → radians]\n',DA.GetPrincipalStrainGaugeAtForceMax());
673 fprintf(1,'Principal strain at max instron force:\n\t%15f strain,\n\t%15f strain,\n\t%
       → %15f radians]\n',DA.GetPrincipalStrainGaugeAtForceInstronMax());
674 fprintf(1,'DIC strain at max force: %f strain\n',DA.GetStrainDICAtForceMax());
675 fprintf(1,'DIC strain at max instron force: %f strain\n',DA.
       → GetStrainDICAtForceInstronMax());
676 fprintf(1,'DIC frame at max force: %d\n',DA.GetFrameAtForceMax());
677 fprintf(1,'DIC frame at max instron force: %d\n',DA.GetFrameAtForceInstronMax());
678
679
680 fprintf(1,'%%%%% Vector Results %%%%\\n');
681 fprintf(1,'Time: [%d,%d] seconds\n',size(DA.GetTime()));
682 fprintf(1,'Six axis force: [%d,%d] N\n',size(DA.GetForceSix()));
683 fprintf(1,'Single axis force: [%d,%d] N\n',size(DA.GetForceOne()));
684 fprintf(1,'Trochanter displacement: [%d,%d] m\n',size(DA.GetDisplacementTroch()));
685 fprintf(1,'Hammer displacement: [%d,%d] m\n',size(DA.GetDisplacementHammer()));
686 fprintf(1,'Lower platen displacement: [%d,%d] m\n',size(DA.GetDisplacementPlaten()));
687 fprintf(1,'Specimen compression: [%d,%d] m\n',size(DA.GetCompression()));
688 fprintf(1,'Gauge strain: [%d,%d] strain\n',size(DA.GetStrainGauge()));
689 fprintf(1,'Gauge principal strain: [%d,%d] strain\n',size(DA.GetPrincipalStrain()));
690 fprintf(1,'DIC strain: [%d,%d] strain\n',size(DA.GetStrainDIC()));
691
692 fprintf('\\n %%%% Associated Objects %%%% \\n');
693 if ~isempty(DA.GetDAQData())
694     DA.GetDAQData().PrintSelf();
695 else
696     fprintf(1,'No DAQ data class associated\\n');
697 end
698 if ~isempty(DA.GetDICData())
699     DA.GetDICData().PrintSelf();
700 else
701     fprintf(1,'No DIC data class associated\\n');
702 end
703 if ~isempty(DA.GetDisplacementData())
704     DA.GetDisplacementData().PrintSelf();
705 else
706     fprintf(1,'No Displacement data class associated\\n');
707 end

```

```

708         end
709     end % public methods
710
711     methods (Access = private , Hidden = true)
712         function CalcCommonTime(DA)
713             % A function to calculate a common time vector in seconds.
714             %
715             % DA.CalcCommonTime()
716             %
717
718             % define time as 200 ms before to 500 ms after trigger.
719             DA.m_time = -0.200:1/DA.GetCommonTimeRate():0.5;
720     end
721
722     function InterpolateDAQToCommonTime(DA)
723         % A function to interpolate the DAQ data to the common time.
724         % If the common time has not been defined, it will be
725         % defined here.
726         %
727         % DA.InterpolateDAQToCommonTime()
728         %
729
730         % check for common time
731         if isempty(DA.GetTime())
732             DA.CalcCommonTime();
733         end
734
735         daqTime = DA.GetDAQData().GetTime();
736
737         % interpolate the single axis load cell
738         DA.m_forceOne = interp1(daqTime,DA.GetDAQData().GetForceOne(),DA.GetTime());
739
740         % interpolate the six axis load cell
741         forceSix = DA.GetDAQData().GetForceSix();
742         DA.m_forceSix(:,1) = interp1(daqTime,forceSix(:,1),DA.GetTime());
743         DA.m_forceSix(:,2) = interp1(daqTime,forceSix(:,2),DA.GetTime());
744         DA.m_forceSix(:,3) = interp1(daqTime,forceSix(:,3),DA.GetTime());
745         DA.m_forceSix(:,4) = interp1(daqTime,forceSix(:,4),DA.GetTime());
746         DA.m_forceSix(:,5) = interp1(daqTime,forceSix(:,5),DA.GetTime());
747         DA.m_forceSix(:,6) = interp1(daqTime,forceSix(:,6),DA.GetTime());
748
749         % interpolate the strain data
750         DA.m_strainGauge(:,1) = interp1(daqTime,DA.GetDAQData().GetStrainGauge1(),DA.GetTime
751             ↵ ());
751         DA.m_strainGauge(:,2) = interp1(daqTime,DA.GetDAQData().GetStrainGauge2(),DA.GetTime
752             ↵ ());
752         DA.m_strainGauge(:,3) = interp1(daqTime,DA.GetDAQData().GetStrainGauge3(),DA.GetTime
753             ↵ ());
754
755         % interpolate the principal strain data
756
756         DA.m_strainPrincipalGauge(:,1) = interp1(daqTime,DA.GetDAQData().GetPrincipalStrain1
757             ↵ (),DA.GetTime());
757         DA.m_strainPrincipalGauge(:,2) = interp1(daqTime,DA.GetDAQData().GetPrincipalStrain2
758             ↵ (),DA.GetTime());
758         DA.m_strainPrincipalGauge(:,3) = interp1(daqTime,DA.GetDAQData().GetPrincipalStrainAngle
759             ↵ (),DA.GetTime());
760     end
761
761     function InterpolateDisplacementDataToCommonTime(DA)
762         % A function to interpolate the displacement data to the
763         % common time. If the common time has not been defined, it
764         % will be defined here.
765         %
766         % DA.InterpolateDisplacementDataToCommonTime()
767         %
768
769         % check common time vector

```

```

770     if isempty(DA.GetTime())
771         DA.CalcCommonTime();
772     end
773
774     dispTime = DA.GetDisplacementData().GetTime();
775     % interpolate the displacements
776     DA.m_displacementTroch = interp1(dispTime,DA.GetDisplacementData().
777         → GetDisplacementTroch(),DA.GetTime());
778     DA.m_displacementHammer = interp1(dispTime,DA.GetDisplacementData().
779         → GetDisplacementHammer(),DA.GetTime());
780 end
781
782 function InterpolateDICToCommonTime(DA)
783     % A function to interpolate the DIC data to the common time
784     % If the common time has not been defined, it will be
785     % defined here.
786     %
787     % DA.InterpolateDICToCommonTime()
788
789     % check for common time vector
790     if isempty(DA.GetTime())
791         DA.CalcCommonTime();
792     end
793
794     dicTime = DA.GetDICData().GetTime();
795     % interpolate the DIC data
796     DA.m_strainDIC = interp1(dicTime,DA.GetDICData().GetStrainData(),DA.GetTime());
797 end
798
799 function CalcDisplacementPlaten(DA)
800     % A function to calculate the lower platen displacement using
801     % partial derivatives. Only works once the daq data has been
802     % interpolated to the common time.
803     %
804     % DA.CalcPlatenDisplacement()
805     %
806
807     % check for time
808     if isempty(DA.GetForceSix())
809         error('DropTowerAnalysis:DataAvailability','Lower platen displacement calculation
810             → was attempted for %s before daq data was available.\n',DA.GetSpecimen().
811             → GetSpecimenName());
812     end
813
814     initialConditions = [0,0];
815
816     % solve the equations numerically
817     [~,x] = ode45(@(t,x) PlatenODE(DA,t,x),DA.GetTime(),initialConditions);
818
819     % once the solution is finished the platen displacement is the first column of the
820     % vector
821     forceSix = DA.GetForceSix();
822     DA.m_displacementPlaten = x(:,1) + DA.GetComplianceLoadingPlate() .* forceSix(:,3);
823
824 end
825
826 function dxdt = PlatenODE(DA,t,x)
827     % A function for the calculation of the lower platen displacement
828     % using the ode45 function.
829
830     m = DA.m_massDropTower;
831     k = 1/DA.m_complianceDropTower;
832
833     % displacement of the platen is the first component
834     xP = x(1);
835     % velocity of the platen is the second component
836     xPdot = x(2);

```

```

833
834     % create the derivatives matrix
835     dxdt = zeros(size(x));
836     % derivative of the first component is the velocity of the platen
837     dxdt(1) = xPdot;
838     % derivative of the second component is the acceleration of the platen, which sums to
839     %      F/m
840     dxdt(2) = 1/m*(DA.GetForceSixCompressionAtTime(t) - k*xP);
841 end
842
843 function CalcCompressionAtForceMax(DA)
844     % A function to calculate the compression of the specimen at
845     % the max force as determined by CalcForceMax().
846     %
847     % DA.CalcCompressionAtForceMax()
848     %
849     % check if index at max force is available
850     if isempty(DA.GetIndexForceMax())
851         error('DropTowerAnalysis:DataAvailability','The compression at max force for %s
852             % was requested before the max force index was available.\n',DA.GetSpecimen
853             %().GetSpecimenName());
854     end
855     if isempty(DA.GetCompression())
856         error('DropTowerAnalysis:DataAvailability','The compression at max force for %s
857             % was requested before compression was available.\n',DA.GetSpecimen()
858             %().GetSpecimenName());
859     end
860
861 function CalcCompressionAtForceInstronMax(DA)
862     % A function to calculate the compression of the specimen at
863     % the max instron force.
864     %
865     % DA.CalcCompressionAtForceMax()
866     %
867     % check if index at max force is available
868     if isempty(DA.GetIndexForceMax())
869         error('DropTowerAnalysis:DataAvailability','The compression at max instron force
870             % was requested before the max force index was available.\n',DA.
871             %().GetSpecimenName());
872     end
873     if isempty(DA.GetCompression())
874         error('DropTowerAnalysis:DataAvailability','The compression at max instron force
875             % was requested before compression was available.\n',DA.GetSpecimen()
876             %().GetSpecimenName());
877     end
878
879     comp = DA.GetCompression();
880     DA.m_compressionAtForceInstronMax = comp(DA.GetIndexForceInstronMax());
881
882 function CalImpactStart(DA)
883     % A function to get the start of the impact. This will find
884     % the start and set the properties m_timeAtImpactStart and
885     % m_indexAtImpactStart. These can be accessed using:
886     % GetTimeImpactStart() and GetIndexImpactStart()
887     %
888     % This works by finding the first point where z-comp of the
889     % six axis load cell is >100 N. It then finds the last time
890     % before that point when force was <20 N. This is taken as
891     % the start of the impact.
892     %
893     % DA.CalImpactStart()

```

```

892 %
893
894 % check that the six axis force is available
895 if isempty(DA.GetForceSix())
896     error('DropTowerAnalysis:DataAvailability','The start of the impact was requested
897         → for %s before the six axis load cell data was available.\n',DA.
898         → GetSpecimen().GetSpecimenName());
899 end
900
901 % find the first index where force is >100 N
902 forceSix = DA.GetForceSix();
903 index100 = find(forceSix(:,3) > 100,1,'first');
904
905 % find the last time that the force was <20 N, using index100 as an upper limit of
906 % search
907 index20 = find(forceSix(1:index100,3) < 20,1,'last');
908
909 % save that index, and the time of that index
910 DA.m_indexAtImpactStart = index20;
911 expTime = DA.GetTime();
912 DA.m_timeAtImpactStart = expTime(index20);
913 end
914
915 function CalcForceMax(DA)
916 % A function to get the max force after the linear part of the
917 % impact. Uses the z-comp of the six axis load cell. This will
918 % set the properties m_ForceMax, m_indexForceMax, m_timeForceMax,
919 % m_strainDICAtForceMax, m_strainPrincipalGaugeAtForceMax
920 % which all have methods that will allow for access.
921 %
922 % Since the absolute max force in the trace may be different
923 % than the max after the linear portion of the impact, the user
924 % must help identify the max force. This function will open a
925 % plot and the user should click near the true max force, the
926 % fucntion will take a window .5 ms wide and find the max in
927 % that window.
928 %
929 % DA.CalcForceMax()
930 %
931 % check that the six axis data is available
932 if isempty(DA.GetForceSix())
933     error('DropTowerAnalysis:DataAvailability','The max force for %s was requested
934         → before the six axis load cell data was available.\n',DA.GetSpecimen().
935         → GetSpecimenName());
936 end
937
938 % find the max for the six axis load cell after the start of the impact
939 forceSix = DA.GetForceSix();
940 time = DA.GetTime();
941 % plot a the force vs time
942 maxForcePlotH = figure;
943 plot(time,forceSix(:,3));
944 msgBoxH = msgbox('Please zoom in and press OK to select the max force ','Maximum Force
945         → Selection ','help');
946 uiwait(msgBoxH);
947 % select the approx max
948 [x,~] = ginput(1);
949 % find the actual max within 2.5 ms of the approx
950 nIndexes = .0005*DA.GetCommonTimeRate();
951 range = find(time > (x-.00025),nIndexes,'first');
952 [rangeVM,rangelM] = max(forceSix(range,3));
953 close(maxForcePlotH)
954
955 valueM = rangeVM;
956 indexM = rangelM-1+range(1);
957
958 % set the max force properties

```

```

954     DA.m_forceMax = valueM;
955     DA.m_indexAtForceMax = indexM;
956 end
957
958 function CalcPropertiesForceMax(DA)
959     % A function to calculate the properties that depend on the
960     % maximum force , such as stiffness and strain at max force.
961     % This should be called after CalcForceMax. If the max force is
962     % not being selected , but reused from a previously set value ,
963     % then this can be called without calling CalcForceMax
964     %
965     % DA.CalcPropertiesAtForceMax()
966     %
967
968     indexM = DA.GetIndexForceMax();
969     % Time at force max
970     expTime = DA.GetTime();
971     DA.m_timeAtForceMax = expTime(indexM);
972
973     % gauge strain at force max
974     strainG = DA.GetPrincipalStrain();
975     DA.m_strainPrincipalGaugeAtForceMax = strainG(indexM,:);
976
977     % If DIC data is available , get the dic strain at max force
978     if DA.GetSpecimen().GetDataAvailable().DropTowerDIC
979         dic = DA.GetStrainDIC();
980         DA.m_strainDICAtForceMax = median(dic(indexM-2:indexM+2));
981     end
982
983     % If disp data is available , get the compression at max force
984     if DA.GetSpecimen().GetDataAvailable().DropTowerDisplacement
985         disp = DA.GetCompression();
986         DA.m_compressionAtForceMax = disp(indexM);
987     end
988
989 end
990
991
992
993 function CalCImpactFinish(DA)
994     % A function to get the finish of the impact. This will find
995     % the first time after max force that the force goes below
996     % 200 N and will set the properties m_timeAtImpactFinish and
997     % m_indexAtImpactFinish. These can be accessed using:
998     % GetTimeImpactFinish() and GetIndexImpactFinish()
999     %
1000     % DA.CalCImpactFinish()
1001     %
1002
1003     % check if the max force has been calculated
1004     if isempty(DA.GetIndexForceMax())
1005         error('DropTowerAnalysis:DataAvailability','The end of the impact was requested
1006             ↪ for %s before the maximum force was found. The calculation of the end
1007             ↪ relies on knowledge of the time of the max force.\n',DA.GetSpecimen().
1008             ↪ GetSpecimenName() );
1009     end
1010
1011     % find the first time the force is below zero after the max force
1012     forceSix = DA.GetForceSix();
1013     index0Trunk = find(forceSix(DA.GetIndexForceMax:end,3) < 200,1,'first');
1014     % account for the limited scope of the find
1015     index0 = index0Trunk-1+DA.GetIndexForceMax();
1016
1017     % set the impact finish properties .
1018     expTime = DA.GetTime();
1019     DA.m_indexAtImpactFinish = index0;
1020     DA.m_timeAtImpactFinish = expTime(index0);
1021
1022 end

```

```

1019
1020 function CalcForceInstronMax(DA)
1021 % A function to get the index and time when the z-comp of the
1022 % six axis load cell first exceeds the max force in the instron.
1023 % Requires that the max instron force and the six axis load
1024 % cell forces are defined. Will set the properties:
1025 % m_indexAtForceInstronMax and m_timeAtForceInstronMax
1026 % both of which have Get methods.
1027 %
1028 % Returns the first time and index when the max force is '
1029 % greater than the instron force.
1030 %
1031 % DA.CalcForceInstronMax()
1032 %
1033
1034 % check for instron force max
1035 if isempty(DA.GetForceInstronMax())
1036     error('DropTowerAnalysis:DataAvailability','The time and index at the max instron
1037         ⇔ force for %s were requested before the max instron force was set.\n',DA.
1038             ⇔ GetSpecimen().GetSpecimenName());
1039 end
1040 if isempty(DA.GetForceSix())
1041     error('DropTowerAnalysis:DataAvailability','The time and index at the max instron
1042         ⇔ force for %s were requested before the six axis load cell data was
1043             ⇔ available.\n',DA.GetSpecimen().GetSpecimenName());
1044 end
1045
1046 forceSix = DA.GetForceSix();
1047 indexIM = find(forceSix(:,3) > DA.GetForceInstronMax(),1,'first');
1048
1049 expTime = DA.GetTime();
1050
1051 DA.m_indexAtForceInstronMax = indexIM;
1052 DA.m_timeAtForceInstronMax = expTime(indexIM);
1053
1054 % Gauge Strains
1055 strainG = DA.GetPrincipalStrain();
1056 DA.m_strainGaugeAtForceInstronMax = strainG(indexIM,:);
1057
1058 % If DIC for the DT is available , get the DIC strain at max instron force
1059 if DA.GetSpecimen().GetDataAvailable().DropTowerDIC
1060     dic = DA.GetStrainDIC();
1061     DA.m_strainDICAtForceInstronMax = median(dic(indexIM-2:indexIM+2));
1062 end
1063
1064 % If disp is available , get the compression at max instron force
1065 if DA.GetSpecimen().GetDataAvailable().DropTowerDisplacement
1066     comp = DA.GetCompression();
1067     DA.m_compressionAtForceInstronMax = comp(indexIM);
1068 end
1069
1070 function CalcStiffness(DA)
1071 % A function to calculate the stiffness from the beginning
1072 % of the impact to the max force as defined in CalcForceMax()
1073 % Gets the stiffness in N/m.
1074 %
1075 % DA.CalcStiffness()
1076 %
1077
1078 % verify that the needed data is available
1079 if isempty(DA.GetForceMax())
1080     error('DropTowerAnalysis:DataAvailability','Stiffness for %s was requested before
1081         ⇔ max force was calculated.\n',DA.GetSpecimen().GetSpecimenName());
1082 end
1083 if isempty(DA.GetCompressionForceMax())
1084     error('DropTowerAnalysis:DataAvailability','Stiffness for %s was requested before
1085         ⇔ the compression at max force was calculated.\n',DA.GetSpecimen());

```

```

1081           ↪ GetSpecimenName() );
1082
1083     end
1084
1085     force = DA.GetForce;
1086     compression = DA.GetCompression;
1087     indexMax = DA.GetIndexForceMax;
1088
1089     % find the forces and compressions at 25% and 90% of force max
1090     index90 = find(force(1:indexMax)<DA.GetForceMax*0.9,1,'last');
1091     force90 = force(index90);
1092     comp90 = compression(index90);
1093
1094     index25 = find(force(1:indexMax)<DA.GetForceMax*0.25,1,'last');
1095     force25 = force(index25);
1096     comp25 = compression(index25);
1097
1098     DA.m_stiffness = (force90-force25)/(comp90-comp25);
1099
1100   end
1101
1102   function CalcEnergyToForceMax(DA)
1103     % A function to calculate the energy in J to the max force as
1104     % defined in CalcForceMax()
1105     %
1106     if isempty(DA.GetIndexImpactStart())
1107       error('DropTowerAnalysis:DataAvailability','Energy to the max force for %s was
1108           ↪ requested before the start of the impact was defined.\n',DA.GetSpecimen());
1109     end
1110     if isempty(DA.GetIndexForceMax())
1111       error('DropTowerAnalysis:DataAvailability','Energy to the max force for %s was
1112           ↪ requested before the max force was found.\n',DA.GetSpecimen());
1113     end
1114     if isempty(DA.GetCompression())
1115       error('DropTowerAnalysis:DataAvailability','Energy to the max force for %s was
1116           ↪ requested the compression was available.\n',DA.GetSpecimen());
1117     end
1118     if isempty(DA.GetForceSix())
1119       error('DropTowerAnalysis:DataAvailability','Energy to the max force for %s was
1120           ↪ requested the six axis load cell data was available.\n',DA.GetSpecimen());
1121     end
1122
1123     forceSix = DA.GetForceSix();
1124     compression = DA.GetCompression();
1125     range = DA.GetIndexImpactStart():DA.GetIndexForceMax();
1126
1127     DA.m_energyToForceMax = trapz(compression(range), forceSix(range,3));
1128
1129   end
1130
1131   function CalcEnergyToForceInstronMax(DA)
1132     % A function to calculate the energy in J to the maximum force
1133     % in the instron.
1134     %
1135     % DA.CalcEnergyToForceInstronMax()
1136     %
1137     if isempty(DA.GetIndexImpactStart())
1138       error('DropTowerAnalysis:DataAvailability','Energy to the max instron force for %
1139           ↪ s was requested before the start of the impact was defined.\n',DA.
1140           GetSpecimen().GetSpecimenName());
1141     end
1142     if isempty(DA.GetIndexForceInstronMax())
1143       error('DropTowerAnalysis:DataAvailability','Energy to the max instron force for %
1144           ↪ s was requested before the max force was set.\n',DA.GetSpecimen());
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237

```

```

1137           → GetSpecimenName() );
end
1138 if isempty(DA.GetCompression())
1139     error('DropTowerAnalysis:DataAvailability ','Energy to the max instron force for %
           → s was requested the compression was available.\n',DA.GetSpecimen().
           → GetSpecimenName());
end
1140 if isempty(DA.GetForceSix())
1141     error('DropTowerAnalysis:DataAvailability ','Energy to the max instron force for %
           → s was requested the six axis load cell data was available.\n',DA.
           → GetSpecimen().GetSpecimenName());
end
1142
1143
1144
1145 forceSix = DA.GetForceSix();
1146 compression = DA.GetCompression();
1147 range = DA.GetIndexImpactStart():DA.GetIndexForceInstronMax();
1148
1149 DA.m_energyToForceInstronMax = trapz(compression(range), forceSix(range,3));
1150 end
1151
1152 function CalcEnergyToImpactFinish(DA)
1153 % A function to calculate the energy in J to the finish of
1154 % the impact as defined in CalcImpactFinish().
1155 %
1156 % DA.CalcEnergyToImpactFinish()
1157 %
1158 if isempty(DA.GetIndexImpactStart())
1159     error('DropTowerAnalysis:DataAvailability ','Energy to the impact finish for %s
           → was requested before the start of the impact was defined.\n',DA.
           → GetSpecimen().GetSpecimenName());
end
1160 if isempty(DA.GetIndexImpactFinish())
1161     error('DropTowerAnalysis:DataAvailability ','Energy to the impact finish for %s
           → was requested before the end of the impact was defined.\n',DA.GetSpecimen
           → ().GetSpecimenName());
end
1162
1163 if isempty(DA.GetCompression())
1164     error('DropTowerAnalysis:DataAvailability ','Energy to the impact finish for %s
           → was requested the compression was available.\n',DA.GetSpecimen().
           → GetSpecimenName());
end
1165
1166 if isempty(DA.GetForceSix())
1167     error('DropTowerAnalysis:DataAvailability ','Energy to the impact finish for %s
           → was requested the six axis load cell data was available.\n',DA.GetSpecimen
           → ().GetSpecimenName());
end
1168
1169
1170
1171 forceSix = DA.GetForceSix();
1172 compression = DA.GetCompression();
1173 range = DA.GetIndexImpactStart():DA.GetIndexImpactFinish();
1174
1175 DA.m_energyToImpactFinish = trapz(compression(range), forceSix(range,3));
1176 end
1177
1178 function CalcFrameForceMax(DA)
1179 % A function to calculate the DIC frame number at the maximum
1180 % force , as determined by CalcForceMax().
1181 %
1182 % DA.CalcFrameForceMax()
1183 %
1184
1185 % check for the six axis load cell
1186 if isempty(DA.GetTimeForceMax())
1187     error('DropTowerAnalysis:DataAvailability ','DIC frame at max force for %s was
           → requested before the time of the max force was available.\n',DA.
           → GetSpecimen().GetSpecimenName());
end
1188
1189 % check for the DIC data

```

```

1190     if isempty(DA.GetDICData())
1191         error('DropTowerAnalysis:DataAvailability','DIC frame at max force for %s was
1192             ↪ requested when no DIC data was available.\n',DA.GetSpecimen());
1193             ↪ GetSpecimenName());
1194     end
1195
1196     timeDICForceMax = DA.GetTimeForceMax() - DA.GetDICData().GetTimeStart();
1197     DA.m_frameAtForceMax = floor(timeDICForceMax*DA.GetDICData().GetSampleRate());
1198 end
1199
1200 function CalcFrameForceInstronMax(DA)
1201     % A function to calculate the DIC frame number at the maximum
1202     % force in the instron.
1203     %
1204     % DA.CalcFrameForceInstronMax()
1205
1206     % check for the max instron force
1207     if isempty(DA.GetTimeForceInstronMax())
1208         error('DropTowerAnalysis:DataAvailability','DIC frame at max instron force for %s
1209             ↪ was requested before the time of the max instron force was available.\n',
1210             ↪ DA.GetSpecimen().GetSpecimenName());
1211     end
1212
1213     % check for the DIC data
1214     if isempty(DA.GetDICData())
1215         error('DropTowerAnalysis:DataAvailability','DIC frame at max instron force for %s
1216             ↪ was requested when no DIC data was available.\n',DA.GetSpecimen());
1217             ↪ GetSpecimenName());
1218 end
1219
1220 timeDICForceInstronMax = DA.GetTimeForceInstronMax() - DA.GetDICData().GetTimeStart()
1221     ↪ ;
1222 DA.m_frameAtForceInstronMax = floor(timeDICForceInstronMax*DA.GetDICData().
1223     ↪ GetSampleRate());
1224 end
1225
1226 function CalcCompression(DA)
1227     % A function to calculate the compression of the specimen
1228     % based on the displacement of the trochanter and the
1229     % motion of the lower platen
1230     %
1231     % DA.CalcCompression()
1232
1233     % check for displacement
1234     if isempty(DA.GetDisplacementTroch())
1235         error('DropTowerAnalysis:DataAvailability','The compression of %s was requested
1236             ↪ before the trochanter displacement was available.\n',DA.GetSpecimen());
1237             ↪ GetSpecimenName());
1238     end
1239     if isempty(DA.GetDisplacementPlaten())
1240         error('DropTowerAnalysis:DataAvailability','The compression of %s was requested
1241             ↪ before the platen displacement was available.\n',DA.GetSpecimen());
1242             ↪ GetSpecimenName());
1243     end
1244
1245     dispTroch = DA.GetDisplacementTroch();
1246
1247     DA.m_compression = dispTroch(:,1) - DA.GetDisplacementPlaten();
1248 end
1249
1250 function CalcRateCompression(DA)
1251     % A function to calculate the rate of displacement of the specimen
1252     % in m/s. Finds the average loading rate between the start
1253     % of the impact and the max force.
1254     %
1255     % DA.CalcRateDisplacement()
1256
1257

```

```

1246     if isempty(DA.GetIndexForceMax())
1247         error('DropTowerAnalysis:DataAvailability','Compression rate was requested for %s
1248             ↪ before max force was available.\n',DA.GetSpecimen().GetSpecimenName());
1249     end
1250     if isempty(DA.GetIndexImpactStart())
1251         error('DropTowerAnalysis:DataAvailability','Compression rate was requested for %s
1252             ↪ before the impact start was available.\n',DA.GetSpecimen().
1253             ↪ GetSpecimenName());
1254     end
1255     comp = DA.GetCompression();
1256     time = DA.GetTime();
1257
1258     DA.m_rateCompression = (comp(DA.GetIndexForceMax()) - comp(DA.GetIndexImpactStart()))/
1259             ↪ /(time(DA.GetIndexForceMax())-time(DA.GetIndexImpactStart())));
1260
1261     function CalcRateForce(DA)
1262         % A function to calculate the rate of loading of the specimen
1263         % in N/s. Finds the average slope of the force time curve from
1264         % the start of the impact to the fracture force.
1265         %
1266         % DA.CalcRateDisplacement()
1267         %
1268         if isempty(DA.GetIndexForceMax())
1269             error('DropTowerAnalysis:DataAvailability','Loading rate was requested for %s
1270                 ↪ before max force was available.\n',DA.GetSpecimen().GetSpecimenName());
1271         end
1272         if isempty(DA.GetIndexImpactStart())
1273             error('DropTowerAnalysis:DataAvailability','Loading rate was requested for %s
1274                 ↪ before the impact start was available.\n',DA.GetSpecimen().GetSpecimenName());
1275         end
1276
1277         force = DA.GetForce();
1278         time = DA.GetTime();
1279
1280         DA.m_rateForce = (force(DA.GetIndexForceMax()) - force(DA.GetIndexImpactStart()))/(
1281             ↪ time(DA.GetIndexForceMax())-time(DA.GetIndexImpactStart())));
1282
1283     end % private methods
1284 end % classdef

```

E.1.7 Drop tower DAQ class

```

1  classdef DAQDropTower < handle
2      properties (SetAccess = private , Hidden = false)
3          m_specimen;
4      end
5      properties (SetAccess = private , Hidden = true)
6          % members for the raw data
7          m_forceSixDAQVoltage;
8          m_forceSixDAQ;
9          m_forceOneDAQVoltage;
10         m_forceOneDAQ;
11         m_strainGauge1DAQ;
12         m_strainGauge2DAQ;
13         m_strainGauge3DAQ;
14         m_triggerDAQ;
15         m_timeDAQ;
16         m_fileNameDAQ = '';
17         m_forceOneZeroed = false; % flag to check if the force trace has be zeroed. Used in the
18             ↪ filtering method
18         m_forceSixZeroed = false;
19         m_indexTrigger;

```

```

20
21 % members for the filtering and analysis
22 m_sampleRate = 20000; %Hz
23 m_samplePeriod = 1/20000; %s
24 m_filterCutoff = 500; % Hz
25 m_filterOrder = 4;
26 m_excitation = 12; %V
27 m_calibrationForceSix = [13344.7/-0.0023141 13344.7/.0023088 13344.7/-0.0009144
28     ↪ 451.9/-0.0018758 451.9/.0019116 226/.0015509];
29 m_calibrationForceOne = 22241.1/.00300293;
30
31 % post filtering data
32 m_forceSix;
33 m_forceOne;
34 m_strainGauge1;
35 m_strainGauge2;
36 m_strainGauge3;
37 m_strainGaugeP1;
38 m_strainGaugeP2;
39 m_strainGaugePhi;
40 m_trigger;
41 m_time;
42 end % properties
43
44 methods (Access = public)
45     function DD = DAQDropTower(specimen)
46         % A class to perform the Drop Tower DAQ analysis. It is
47         % constructed using a specimen. See Specimen.m for details
48         % on the specimen type.
49         %
50         % DD = DAQDropTower(specimen)
51         %
52         DD.m_specimen = specimen;
53     end
54
55     function SetFileName(DD, file)
56         % A function to set the file name for the drop tower DAQ
57         % analysis.
58         %
59         % DD.SetFileName(file)
60         %
61         if ~strcmp(DD.m_fileNameDAQ, file)
62             if ~exist(file, 'file')
63                 error('DAQDropTower:DataAvailability','The DAQ file specified for %s does not
64                     ↪ exist.\n',DD.GetSpecimen().GetSpecimenName());
65             end
66             DD.m_fileNameDAQ = file;
67         end
68     end
69     function o = GetFileName(DD)
70         % A function to get the file name for the drop tower DAQ.
71         %
72         % FileName = DD.GetFileName()
73         %
74         o = DD.m_fileNameDAQ;
75     end
76
77     function o = GetSpecimen(DD)
78         o = DD.m_specimen;
79     end
80
81     function SetSampleRate(DD, rate)
82         % A function to set the sample rate of the drop tower DAQ in
83         % Hz.
84         % The default value is 20000 Hz.
85         % Automatically updates the sample period.

```

```

86      %
87      % DD. SetSampleRate( rate )
88      %
89      if DD.m_sampleRate ~= rate
90          DD.m_sampleRate = rate;
91          DD.m_samplePeriod = 1/rate;
92      end
93  end
94  function o = GetSampleRate(DD)
95      % A function to get the drop tower sample rate.
96      %
97      % SampleRate = DD.GetSampleRate()
98      %
99      o = DD.m_sampleRate;
100 end
101
102 function SetSamplePeriod(DD,period)
103     % A function to set the sample period of the drop tower DAQ
104     % in seconds.
105     % The default value is 50 microseconds (rate = 20 kHz)
106     % Automatically updates the sample rate.
107     %
108     % DD. SetSamplePeriod(period)
109     %
110     if DD.m_samplePeriod ~= period
111         DD.m_samplePeriod = period;
112         DD.m_sampleRate = 1/period;
113     end
114 end
115 function o = GetSamplePeriod(DD)
116     % A function to get the drop tower sample period.
117     %
118     % SamplePeriod = DD.GetSamplePeriod()
119     %
120     o = DD.m_samplePeriod;
121 end
122
123 function SetFilterCutoff(DD,cutoff)
124     % A function to set the drop tower filter cutoff frequency
125     % in Hz.
126     %
127     % DD. SetFilterCutoff(cutoff)
128     %
129     if DD.m_filterCutoff ~= cutoff
130         DD.m_filterCutoff = cutoff;
131     end
132 end
133 function o = GetFilterCutoff(DD)
134     % A function to get the cutoff frequency of the drop tower
135     % filter in Hz.
136     %
137     % Cutoff = DD.GetFilterCutoff()
138     %
139     o = DD.m_filterCutoff;
140 end
141
142 function SetFilterOrder(DD,order)
143     % A function to set the order of the drop tower DAQ filter.
144     % Since the data is forward-reversed filtered, only even
145     % filter orders are accepted. During filtering a value equal
146     % to order/2 will be passed to filtfilt, resulting in a
147     % final filter order equal to the input of this function.
148     %
149     % The order input here must be even, and if an odd order is
150     % supplied, the final order will be supplied order + 1.
151     %
152     % DD. SetFilterOrder(order)
153     %

```

```

154     if DD.m_filterOrder ~= order
155         if mod(cutoff ,2)
156             warning('DAQDropTower:DataValues','The filter order for %s was set to an odd
157             ↪ number. Only even filter orders are accepted. The order of the filter
158             ↪ will be incremented by one.\n',DD.GetSpecimen().GetSpecimenName());
159             order = order + 1;
160         end
161     end
162     DD.m_filterOrder = order;
163 end
164
165 function o = GetFilterOrder(DD)
166     % A function to get the filter order used to filter the drop
167     % tower data.
168     %
169     % Order = DD.GetFilterOrder()
170     %
171     o = DD.m_filterOrder;
172 end
173
174 function SetExcitation(DD,volts)
175     % A function to set the exitation of the load cells in
176     % volts. The default value is 12 V.
177     %
178     % DD.SetExcitation(volts)
179     %
180     if DD.m_excitation ~= volts
181         DD.m_excitation = volts;
182     end
183 end
184
185 function o = GetExcitation(DD)
186     % A function to get the exictation of the load cells in
187     % volts.
188     %
189     % Excitation = DD.GetExcitation()
190     %
191     o = DD.m_excitation;
192 end
193
194 function SetCalibrationForceSix(DD,cal)
195     % A function to set the calibration vector for the six axis
196     % load cell. The default value is:
197     % [13344.7/-0.0023141 13344.7/.0023088 13344.7/-0.0009144 451.9/-0.0018758
198     %   ↪ 451.9/.0019116 226/.0015509] N/mV/V_excitation
199     %
200     % DD.SetCalibrationForceSix(calibration)
201     %
202     if DD.m_calibrationForceSix ~= cal
203         DD.m_calibrationForceSix = cal;
204     end
205 end
206
207 function o = GetCalibrationForceSix(DD)
208     % A function to get the calibration vector for the six axis
209     % load cell in N/mV/V_excitation.
210     %
211     % Calibration = DD.GetCalibrationForceSix()
212     %
213     o = DD.m_calibrationForceSix;
214 end
215
216 function SetCalibrationForceOne(DD,cal)
217     % A function to set the calibration for the single axis load
218     % cell. The default value is: 22241.1/.00300293 N/mV/V_excitation
219     %
220     % DD.SetCalibrationForceOne(calibration)
221     %
222     if DD.m_calibrationForceOne ~= cal
223         DD.m_calibrationForceOne = cal;
224     end

```

```

219     end
220 function o = GetCalibrationForceOne(DD)
221     % A function to get the calibration for the single axis load
222     % cell in N/mV/ V_excitation.
223     %
224     % Calibration = DD.GetCalibrationForceOne()
225     %
226     o = DD.m_calibrationForceOne;
227 end
228
229 function ReadFile(DD)
230     % A function to read the DAQ file specified. To set the file
231     % name use DD.SetFileName(file)
232     %
233     % DD.ReadFile()
234     %
235
236     % check if m_fileName is full
237     if isempty(DD.GetFileName())
238         error('DropTowerDAQ:DataAvailablity ','File Read was called for %s before the file
239             ↪ name was set.\nUse DD.SetFileName(name) to set the file name.\n',DD.
240             ↪ GetSpecimen().GetSpecimenName());
241     end
242
243     % read file
244     droptowerFID = fopen(DD.GetFileName(),'r');
245     droptower = textscan(droptowerFID,'%f %f %f',
246             ↪ ', ',' ');
247     fclose(droptowerFID);
248
249     % put data into members
250     DD.m_timeDAQ = droptower{1};
251
252     DD.m_strainGauge1DAQ = droptower{2};
253     DD.m_strainGauge2DAQ = droptower{3};
254     DD.m_strainGauge3DAQ = droptower{4};
255
256     DD.m_forceSixDAQVoltage(:,1) = droptower{5}; % f_x
257     DD.m_forceSixDAQVoltage(:,2) = droptower{6}; % f_y
258     DD.m_forceSixDAQVoltage(:,3) = droptower{7}; % f_z
259     DD.m_forceSixDAQVoltage(:,4) = droptower{8}; % m_x
260     DD.m_forceSixDAQVoltage(:,5) = droptower{9}; % m_y
261     DD.m_forceSixDAQVoltage(:,6) = droptower{10};% m_z
262
263     DD.m_forceOneDAQVoltage = droptower{11};
264
265     DD.m_triggerDAQ = droptower{12};
266 end
267
268 function o = GetTime(DD)
269     % A function to get the time vector starting with t = 0s at
270     % the trigger.
271     %
272     % Time = DD.GetTime()
273     %
274     o = DD.m_time;
275 end
276
277 function o = GetForceSix(DD)
278     % A function to get the six axis load cell matrix. The matrix
279     % is in the format [F_x,F_y,F_z,M_x,M_y,M_z]. Returns the
280     % fully processed matrix of data with forces in N and
281     % moments in Nm.
282     %
283     % ForceData = DD.GetForceSix()
284     %
285     o = DD.m_forceSix;
286 end

```

```

284     function o = GetForceOne(DD)
285         % A function to get the single axis load cell vector.
286         % Returns the fully processed vector in N.
287         %
288         % ForceData = DD.GetForceOne()
289         %
290         o = DD.m_forceOne;
291     end
292     function o = GetStrainGauge1(DD)
293         % A function to get the strain data from the first gauge.
294         % Returns the strain in absolute strain.
295         %
296         % Strain = DD.GetStrainGauge1()
297         %
298         o = DD.m_strainGauge1;
299     end
300     function o = GetStrainGauge2(DD)
301         % A function to get the strain data from the second gauge.
302         % Returns the strain in absolute strain.
303         %
304         % Strain = DD.GetStrainGauge2()
305         %
306         o = DD.m_strainGauge2;
307     end
308     function o = GetStrainGauge3(DD)
309         % A function to get the strain from the third gauge.
310         % Returns the strain in absolute strain.
311         %
312         % Strain = DD.GetStrainGauge3()
313         %
314         o = DD.m_strainGauge3;
315     end
316     function o = GetPrincipalStrain1(DD)
317         % A function to get the first principal strain. Returns
318         % the strain in absolute strain.
319         %
320         % Strain = DD.GetPrincipalStrain1()
321         %
322         o = DD.m_strainGaugeP1;
323     end
324     function o = GetPrincipalStrain2(DD)
325         % A function to get the second principal strain. Returns
326         % the strain in absolute strain.
327         %
328         % Strain = DD.GetPrincipalStrain2()
329         %
330         o = DD.m_strainGaugeP2;
331     end
332     function o = GetPrincipalStrainAngle(DD)
333         % A function to get the principal strain angle. Returns the
334         % angle in radians from gauge A as defined in Appendix G of:
335         % Budynas R.G. Advanced Strength and Applied Stress
336         % Analysis , Second ed. McGraw Hill. ISBN 0-07-008985-X
337         %
338         % Angle = DD.GetPrincipalStrainAngle()
339         %
340         o = DD.m_strainGaugePhi;
341     end
342     function o = GetTrigger(DD)
343         % A function to get the trigger vector. The trigger vector
344         % is never filtered , but is transferred from the raw data
345         % vector to the output data vector when
346         % DD.CalcFilteredData() is called.
347         %
348         % Trigger = DD.GetTrigger()
349         %
350         o = DD.m_trigger;
351     end

```

```

352     function o = GetForceSixVoltage(DD)
353         % A function to get the six axis load cell voltage data.
354         % Returns the raw data loaded from the input file in the
355         % form [F_x,F_y,F_z,M_x,M_y,M_z].
356         %
357         % Voltages = DD.GetForceSixVoltage()
358         %
359         o = DD.m_forceSixDAQVoltage;
360     end
361     function o = GetForceSixRaw(DD)
362         % A function to get the six axis load cell raw force data.
363         % Returns the calibrated force data with no offset removal or
364         % filtering in [F_x,F_y,F_z,M_x,M_y,M_z] with forces in N
365         % and moments in Nm.
366         %
367         % Forces = DD.GetForceSixRaw()
368         %
369         o = DD.m_forceSixDAQ;
370     end
371     function o = GetForceOneVoltage(DD)
372         % A function to get the single axis load cell voltage data.
373         % Returns the raw data loaded from the input file .
374         %
375         % Voltage = DD.GetForceOneVoltage(DD)
376         %
377         o = DD.m_forceOneDAQVoltage;
378     end
379     function o = GetForceOneRaw(DD)
380         % A function to get the single axis load cell raw force data.
381         % Returns the calibrated force data with no offset removal or
382         % filtering in N.
383         %
384         % Force = DD.GetForceOneRaw()
385         %
386         o = DD.m_forceOneDAQ;
387     end
388     function o = GetStrainGauge1Raw(DD)
389         % A function to get the raw strain gauge data read from the
390         % input file .
391         %
392         % Strain = DD.GetStrainGauge1Raw()
393         %
394         o = DD.m_strainGauge1DAQ;
395     end
396     function o = GetStrainGauge2Raw(DD)
397         % A function to get the raw strain gauge data read from the
398         % input file .
399         %
400         % Strain = DD.GetStrainGauge2Raw()
401         %
402         o = DD.m_strainGauge2DAQ;
403     end
404     function o = GetStrainGauge3Raw(DD)
405         % A function to get the raw strain gauge data read from the
406         % input file .
407         %
408         % Strain = DD.GetStrainGauge3Raw()
409         %
410         o = DD.m_strainGauge3DAQ;
411     end
412     function o = GetTriggerRaw(DD)
413         % A function to get the raw trigger data from the input file .
414         %
415         % Trigger = DD.GetTriggerRaw()
416         %
417         o = DD.m_triggerDAQ;
418     end
419     function o = GetTimeRaw(DD)

```

```

420      % A function to get the raw time data read from the input
421      % file. This time will not be zeroed at the time of the
422      % trigger.
423      %
424      % Time = DD.GetTimeRaw()
425      %
426      o = DD.m_timeDAQ;
427
428  end
429
430  function Update(DD)
431      % A function that checks for all needed data and executes the
432      % filter in the correct order. It does not read the input file
433      % so that must be done first by the user.
434      %
435      % DD.Update()
436      %
437
438      % first check the input data is available
439      if (isempty(DD.m_forceSixDAQVoltage) || isempty(DD.m_forceOneDAQVoltage) || isempty(
440          ↪ DD.m_strainGauge1DAQ) || isempty(DD.m_strainGauge2DAQ) || isempty(DD.
441          ↪ m_strainGauge3DAQ) || isempty(DD.m_triggerDAQ) || isempty(DD.m_timeDAQ))
442          error('DropTowerDAQ:DataAvailability','Update called for %s when no raw data is
443          ↪ available.\nPerhapse call DD.ReadFile().\n',DD.GetSpecimen());
444          ↪ GetSpecimenName());
445
446      end
447
448      % calibrate the load cells
449      DD.CalibrateForceSix();
450      DD.CalibrateForceOne();
451
452      % zero the data
453      DD.ZeroForceSix();
454      DD.ZeroForceOne();
455
456      % filter the data
457      DD.CalcFilteredData();
458
459      % calculate the principal strains
460      DD.CalcPrincipalStrains();
461
462  end
463
464  function PrintSelf(DD)
465      % A function to print out all of the data contained in the class
466      %
467      % DD.PrintSelf()
468      %
469      fprintf(1,'\\\\n%%%%% DAQDropTower Class Parameters %%%%%\\n');
470      DD.GetSpecimen().PrintSelf();
471      fprintf(1,'\\n %%% Scalar Members and Properties %%%\\n');
472      fprintf(1,'DAQ file name: %s\\n',DD.GetFileName());
473      fprintf(1,'DAQ sample rate: %f Hz\\n',DD.GetSampleRate());
474      fprintf(1,'DAQ sample period: %f s\\n',DD.GetSamplePeriod());
475      fprintf(1,'DAQ filter cutoff frequency: %f Hz\\n',DD.GetFilterCutoff());
476      fprintf(1,'DAQ filter order: %d\\n',DD.GetFilterOrder());
477      fprintf(1,'Load cell excitation: %f V\\n',DD.GetExcitation());
478      fprintf(1,'Six axis calibration vector:\\n\\t[%13.2f\\n\\t %13.2f\\n\\t %13.2f\\n\\t %13.2f\\n
479          ↪ \\t %13.2f\\n\\t %13.2f] (N/mV) / V_excite\\n',DD.GetCalibrationForceSix());
480      fprintf(1,'Single axis calibration: %13.2f (N/mV) / V_excite\\n',DD.
481          ↪ GetCalibrationForceOne());
482      fprintf(1,'Six axis load cell data zeroed: %i\\n',DD.m_forceSixZeroed);
483      fprintf(1,'Single axis load cell data zeroed: %i\\n',DD.m_forceOneZeroed);
484
485
486      fprintf(1,'\\n %%% Raw input data %%% \\n');
487      fprintf(1,'DAQ six axis force voltage: [%d,%d] in volts\\n',size( DD.
488          ↪ GetForceSixVoltage() ));
489      fprintf(1,'DAQ six axis force raw: [%d,%d] in newtons\\n',size( DD.GetForceSixRaw() ))
490          ↪ ;

```

```

480     fprintf(1,'DAQ single axis force voltage: [%d,%d] in volts\n',size( DD.
481             ↪ GetForceOneVoltage() ) );
482     fprintf(1,'DAQ single axis force raw: [%d,%d] in newtons\n',size( DD.GetForceOneRaw()
483             ↪ ) );
484     fprintf(1,'DAQ strain gauge 1 raw: [%d,%d] in strain\n',size( DD.GetStrainGauge1Raw()
485             ↪ ) );
486     fprintf(1,'DAQ strain gauge 2 raw: [%d,%d] in strain\n',size( DD.GetStrainGauge2Raw()
487             ↪ ) );
488     fprintf(1,'DAQ strain gauge 3 raw: [%d,%d] in strain\n',size( DD.GetStrainGauge3Raw()
489             ↪ ) );
490     fprintf(1,'DAQ trigger raw: [%d,%d] in volts\n',size( DD.GetTriggerRaw() ) );
491     fprintf(1,'DAQ time raw: [%d,%d] in seconds\n',size( DD.GetTimeRaw() ) );
492
493     fprintf(1,'\n %%% Analyzed data %%% \n');
494     fprintf(1,'DAQ six axis force: [%d,%d] in newtons\n',size( DD.GetForceSix() ) );
495     fprintf(1,'DAQ single axis force: [%d,%d] in newtons\n',size( DD.GetForceOne() ) );
496     fprintf(1,'DAQ strain gauge 1: [%d,%d] in strain\n',size( DD.GetStrainGauge1() ) );
497     fprintf(1,'DAQ strain gauge 2: [%d,%d] in strain\n',size( DD.GetStrainGauge2() ) );
498     fprintf(1,'DAQ strain gauge 3: [%d,%d] in strain\n',size( DD.GetStrainGauge3() ) );
499     fprintf(1,'DAQ principal strain 1: [%d,%d] in strain\n',size( DD.GetPrincipalStrain1
500             ↪ () ) );
501     fprintf(1,'DAQ principal strain 2: [%d,%d] in strain\n',size( DD.GetPrincipalStrain2
502             ↪ () ) );
503     fprintf(1,'DAQ principal strain angle: [%d,%d] in radians\n',size( DD.
504             ↪ GetPrincipalStrainAngle() ) );
505     fprintf(1,'DAQ trigger: [%d,%d] in volts\n',size( DD.GetTrigger() ) );
506     fprintf(1,'DAQ time: [%d,%d] in seconds\n',size( DD.GetTime() ) );
507
508 end
509 end % methods public
510
511 methods (Access = private , Hidden = true)
512     function CalibrateForceSix(DD)
513         % A function that applies the exitation value to the
514         % six axis load cell calibration vector and scales the
515         % voltages in the six axis load cell voltage matrix to get
516         % the force.
517         %
518         % DD.CalibrateForceSix()
519         %
520         if isempty(DD.GetForceSixVoltage())
521             error('DropTowerDAQ:DataAvailability','Unable to calibrate six axis load cell for
522             ↪ %s. Load cell data not loaded.\n',DD.GetSpecimen().GetSpecimenName() )
523         end
524
525         cal = DD.GetCalibrationForceSix() .*1/DD.GetExcitation();
526
527         DD.m_forceSixDAQ = zeros(size(DD.GetForceSixVoltage()));
528         forceVoltages = DD.GetForceSixVoltage();
529
530         for i = 1:length(DD.GetForceSixVoltage())
531             DD.m_forceSixDAQ(i,:) = forceVoltages(i,:).*cal;
532         end
533     end
534
535     function CalibrateForceOne(DD)
536         % A function that applies the exitation value to the single
537         % axis load cell calibration value and scales teh voltages
538         % in the single axis load cell vector to get forces.
539         %
540         % DD.CalibrateForceOne()
541         %
542         if isempty(DD.GetForceOneVoltage())
543             error('DropTowerDAQ:DataAvailability','Unable to calibrate single axis load cell
544             ↪ for %s. Load cell data not loaded.\n',DD.GetSpecimen().GetSpecimenName() )
545         end
546
547         cal = DD.GetCalibrationForceOne() *1/DD.GetExcitation();

```

```

538
539     DD.m_forceOneDAQ = DD.GetForceOneVoltage()*cal;
540
541 end
542
543 function ZeroForceSix(DD)
544 % A function to zero the six axis load cell data. Uses the
545 % average data before the trigger to zero the entire data
546 % trace. Operates only on calibrated data, so will call
547 % DD.CalibrateForceSix() if the six axis data vectors are
548 % empty.
549 %
550 % DD.ZeroForceSix()
551 %
552 if isempty(DD.GetForceSixRaw())
553     warning('DropTowerDAQ:DataAvailability','Zeroing of the six axis load cell for %s
554     ↪ was requested before calibration. Calibration will be carried out now.\n
555     ↪ ',DD.GetSpecimen().GetSpecimenName());
556     DD.CalibrateForceSix();
557 end
558
559 idxTrig = DD.GetIndexTrigger();
560 forceRaw = DD.GetForceSixRaw();
561
562 forcePreTrig = mean(forceRaw(1:idxTrig,:));
563 DD.m_forceSixDAQ(:,1) = DD.m_forceSixDAQ(:,1) - forcePreTrig(1);
564 DD.m_forceSixDAQ(:,2) = DD.m_forceSixDAQ(:,2) - forcePreTrig(2);
565 DD.m_forceSixDAQ(:,3) = DD.m_forceSixDAQ(:,3) - forcePreTrig(3);
566 DD.m_forceSixDAQ(:,4) = DD.m_forceSixDAQ(:,4) - forcePreTrig(4);
567 DD.m_forceSixDAQ(:,5) = DD.m_forceSixDAQ(:,5) - forcePreTrig(5);
568 DD.m_forceSixDAQ(:,6) = DD.m_forceSixDAQ(:,6) - forcePreTrig(6);
569
570 DD.m_forceSixZeroed = true;
571 end
572
573 function ZeroForceOne(DD)
574 % A function to zero the single axis load cell data. Uses
575 % the average data before the trigger to zero the entire
576 % data trace. Operates only on calibrated data, so will call
577 % DD.CalibrateForceOne() if the single axis data vector is
578 % empty.
579 %
580 % DD.ZeroForceOne()
581 %
582 if isempty(DD.GetForceOneRaw())
583     warning('DropTowerDAQ:DataAvailability','Zeroing of the single axis load cell for %s
584     ↪ was requested before calibration. Calibration will be carried out now\n
585     ↪ ',DD.GetSpecimen().GetSpecimenName());
586     DD.CalibrateForceOne();
587 end
588
589 idxTrig = DD.GetIndexTrigger();
590 forceRaw = DD.GetForceOneRaw();
591
592 forcePreTrig = mean(forceRaw(1:idxTrig));
593 DD.m_forceOneDAQ = DD.m_forceOneDAQ - forcePreTrig;
594
595 DD.m_forceOneZeroed = true;
596 end
597
598 function CalcFilteredData(DD)
599 % A function to filter the data supplied in the input file.
600 % To supply input data, use DD.SetFileName(file), followed
601 % by DD.ReadFile(). The filtering is performed using a
602 % Butterworth filter of order found in DD.GetFilterOrder(),
603 % and at a cut off frequency found in DD.GetFilterCutoff().
604 %

```

```

602 % DD.CalcFilteredData()
603 %
604
605 % Check if the data has been zeroed and calibrated
606 if (~DD.m_forceOneZeroed || ~DD.m_forceSixZeroed)
607     error('DropTowerDAQ:ExecutionOrder','Data filtering for %s called before the data
608         → was calibrated and zeroed.\n',DD.GetSpecimen().GetSpecimenName())
609 end
610
611 % design the filter
612 cutoffNomal = DD.GetFilterCutoff() /DD.GetSampleRate();
613 [b,a] = butter(DD.GetFilterOrder/2,cutoffNomal);
614 forceSixRaw = DD.GetForceSixRaw();
615
616 % filter the six axis load cell
617 for i = 1:size(DD.GetForceSixRaw(),2)
618     DD.m_forceSix(:,i) = filtfilt(b,a,forceSixRaw(:,i));
619 end
620 % filter the single axis load cell
621 DD.m_forceOne = filtfilt(b,a,DD.GetForceOneRaw());
622 % filter the strain gauge data
623 DD.m_strainGauge1 = filtfilt(b,a,DD.GetStrainGauge1Raw());
624 DD.m_strainGauge2 = filtfilt(b,a,DD.GetStrainGauge2Raw());
625 DD.m_strainGauge3 = filtfilt(b,a,DD.GetStrainGauge3Raw());
626
627 % transfer the trigger data without filtering
628 DD.m_trigger = DD.m_triggerDAQ;
629
630 % zero the time vector at the trigger
631 DD.ZeroTimeAtTrigger();
632
633 end
634
635 function CalcPrincipalStrains(DD)
636     % A function to calculate the principal strains from the
637     % strain gauge data.
638     %
639     % DD.CalcPrincipalStrains()
640     %
641     % check if strain data is available
642     if (isempty(DD.GetStrainGauge1()) || isempty(DD.GetStrainGauge2()) || isempty(DD.
643         → GetStrainGauge3()))
644         error('DropTowerDAQ:DataAvailability','An attempt to calculate the principal
645             → strains for %s was attempted before the strains were available.\nPossibly
646             → DD.CalcFilteredData() needs to be called?\n',DD.GetSpecimen().
647             → GetSpecimenName());
648 end
649
650 % calculate the principal strains.
651 gA = DD.GetStrainGauge1();
652 gB = DD.GetStrainGauge2();
653 gC = DD.GetStrainGauge3();
654 DD.m_strainGaugeP1 = ( (gA+gC)/2 + 1/2.*sqrt( (gA-gC).^2 + (2.*gB-gA-gC).^2 ) );
655 DD.m_strainGaugeP2 = ( (gA+gC)/2 - 1/2.*sqrt( (gA-gC).^2 + (2.*gB-gA-gC).^2 ) );
656 DD.m_strainGaugePhi = 0.5.*atan( (2.*gB-gA-gC) ./ (gA-gC) );
657
658 end
659
660 function o = GetIndexTrigger(DD)
661     % A function to return the trigger index in the DAQ index
662     % (rather than experiment) index space.
663     %
664     % Index = DD.GetIndexTrigger()
665     %
666     if isempty(DD.m_indexTrigger)
667         DD.m_indexTrigger = find(DD.m_triggerDAQ < 4.9,1,'first');
668     end
669     o = DD.m_indexTrigger;

```

```

665      end
666
667      function ZeroTimeAtTrigger(DD)
668          % A function to zero the time at the moment of the trigger.
669          %
670          % DD.ZeroTimeAtTrigger()
671          %
672          rawTime = DD.GetTimeRaw;
673          DD.m_time = DD.GetTimeRaw() - rawTime(DD.GetIndexTrigger());
674      end
675
676      end % private methods
677
678 end % classdef

```

E.1.8 Drop tower displacement class

```

1 classdef DTDisplacementData < handle
2     properties (SetAccess = private, Hidden = false)
3         m_specimen;
4     end
5
6     properties (SetAccess = private, Hidden = true)
7         m_displacementTroch;    % displacement of the trochanter
8         m_displacementHammer;   % displacement of the impact hammer
9         m_displacementTrochFilt; % displacement of the trochanter, filtered
10        m_displacementHammerFilt; % displacement of the hammer, filtered
11        m_timeDisplacement;    % time of the values in displacement camera time
12        m_time;
13        m_timeStart = 0;        % time of the first data point, experiment time
14        m_sampleRate = 0;       % camera sample rate in Hz
15        m_filterCutoff = 500;   % filter cutoff frequency in Hz
16        m_filterOrder = 2;      % butterworth filter order. Must be even, odd orders will be
17            ↪ increased by 1
18        m_fileName = '';       % the raw TEMA data file
19    end % properties
20
21    methods
22        % Constructor
23        function DTDD = DTDisplacementData(specimen)
24            % The constructor to create a drop tower displacement data
25            % object. The only input is a specimen object. See Specimen.m
26            % for details on the creation of a specimen object.
27            %
28            % DTDD = DTDisplacementData(specimen)
29            %
30            DTDD.m_specimen = specimen;
31        end
32
33        function o = GetSpecimen(DTDD)
34            % A function to get the specimen object associated with the
35            % DT Displacement Data object at creation time.
36            %
37            % Specimen = DTDD.GetSpecimen()
38            %
39            o = DTDD.m_specimen;
40        end
41
42        function SetFileName(DTDD, file)
43            % A function to set the name of the displacement data file.
44            %
45            % DTDD.SetFileName(file)
46            %
47            if ~strcmp(DTDD.m_fileName, file)
48                if ~exist(file, 'file')

```

```

48             error('DropTowerDisplacement:DataAvailability','The specified displacement
49                         ← file for %s does not exist.\n',DTDD.GetSpecimen().GetSpecimenName());
50
51             DTDD.m_fileName = file;
52         end
53     end
54     function o = GetFileName(DTDD)
55         % A funtion to get the name of the file containing the drop
56         % tower displacement data.
57         %
58         % Name = DTDD.GetFileName()
59         %
60         o = DTDD.m_fileName;
61     end
62
63     function SetTimeStart(DTDD,time)
64         % A function to set the displacement data in seconds. In some
65         % cases the first data point of the displacement data will
66         % not line up with the trigger (ie experiment time) and this
67         % value is used to correct for that offset.
68         %
69         % DTDD.SetTimeStart(time)
70         %
71         if DTDD.m_timeStart ~= time
72             DTDD.m_timeStart = time;
73         end
74     end
75     function o = GetTimeStart(DTDD)
76         % A function to get the time of the first data point in the
77         % displacement data in terms of time post trigger. The value is
78         % in seconds
79         %
80         % Time = DTDD.GetTimeStart()
81         %
82         o = DTDD.m_timeStart;
83     end
84
85     function SetSampleRate(DTDD,rate)
86         % A function to set the sampling rate of the displacement
87         % data in Hz.
88         %
89         % DTDD.SetSampleRate(rate)
90         %
91         if DTDD.m_sampleRate ~= rate
92             DTDD.m_sampleRate = rate;
93         end
94     end
95     function o = GetSampleRate(DTDD)
96         % A function to get the sample rate of the displacement data in
97         % Hz.
98         %
99         % Rate = DTDD.GetSampleRate()
100        %
101        o = DTDD.m_sampleRate;
102    end
103
104    function ReadFile(DTDD)
105        % A function to read the specified file containing the
106        % displacement data. Time should be in ms and the displacement
107        % in mm.
108        %
109        % DTDD.ReadFile()
110        %
111        if isempty(DTDD.m_fileName)
112            error('DropTowerDisplacement:DataAvailability','File read was called for %s when
113                         ← no file name has been set.\n',DTDD.GetSpecimen().GetSpecimenName());

```

```

114     data = importdata(DTDD.m_fileName, '\t');
115     % find the shortest data filed
116     validData = ~isnan(data.data);
117     startIndex = max([ find(validData(:,1),1,'first'), find(validData(:,2),1,'first'), find(
118         ↪ validData(:,3),1,'first'), find(validData(:,4),1,'first'), find(validData(:,5)
119         ↪ ,1,'first') ]);
120     endIndex = min( [ find(validData(:,1),1,'last'), find(validData(:,2),1,'last'), find(
121         ↪ validData(:,3),1,'last'), find(validData(:,4),1,'last'), find(validData(:,5)
122         ↪ ,1,'last')] );
123
124     % import the time data
125     DTDD.m_timeDisplacement = data.data(startIndex:endIndex,1)./1000;
126     % import the impact hammer data and convert to m
127     DTDD.m_displacementHammer = [data.data(startIndex:endIndex,2), data.data(startIndex:
128         ↪ endIndex,3)]./1000;
129     DTDD.m_displacementTroch = [data.data(startIndex:endIndex,4), data.data(startIndex:
130         ↪ endIndex,5)]./1000;
131 end
132
133 function o = GetDisplacementTrochUnfiltered(DTDD)
134     % A function to get the unfiltered displacement of the
135     % trochanter in m.
136     %
137     % Displacement = DTDD.GetDisplacementTrochUnfiltered()
138     %
139     o = DTDD.m_displacementTroch;
140 end
141
142 function o = GetDisplacementHammerUnfiltered(DTDD)
143     % A function to get the unfiltered displacement of the impact
144     % hammer in m.
145     %
146     % Displacement = DTDD.GetDisplacementHammerUnfiltered()
147     %
148     o = DTDD.m_displacementHammer;
149 end
150
151 function SetFilterCutoff(DTDD,rate)
152     % A function to set the filter cutoff frequency in Hz. The
153     % default value is 500 Hz.
154     %
155     % DTDD.SetFilterCutoff(cutoff)
156     %
157     if DTDD.m_filterCutoff ~= rate
158         DTDD.m_filterCutoff = rate;
159     end
160 end
161
162 function o = GetFilterCutoff(DTDD)
163     % A function to get the filter cutoff frequency in Hz.
164     %
165     % Cutoff = DTDD.GetFilterCutoff()
166     %
167     o = DTDD.m_filterCutoff;
168 end
169
170
171 % functions to set/get the filter order
172 function SetFilterOrder(DTDD,order)
173     % A funtion to set the order of the butterworth filter. Since
174     % the algorithm uses the filtfilt function only even filter
175     % orders will be accepted. If an odd filter order is proveded,
176     % it will be incremented by one. The filtfilt function is
177     % passed the order supplied/2. The forward reverse nature of
178     % filtfilt effectively doubles the filter order.
179     %
180     % DTDD.SetFilterOrder(order)
181     %
182     if mod(order, 2)
183         warning('DropTowerDisplacement:DataValues','The filter order for %s was set to an
184             ↪ odd number. Only even orders are accepted. The order is being increased

```

```

175           → by one.\n',DTDD.GetSpecimen().GetSpecimenName() );
176           order = order +1;
177       end
178   DTDD.m_filterOrder = order;
179 end
180 function o = GetFilterOrder(DTDD)
181     % A function to get the filter order.
182     %
183     % Order = DTDD.GetFilterOrder()
184     %
185     o = DTDD.m_filterOrder;
186 end
187
188 function o = GetDisplacementTroch(DTDD)
189     % A function to get the filtered trochanter displacement data
190     % in m.
191     %
192     % Displacement = DTDD.GetDisplacementTroch()
193     %
194     if isempty(DTDD.m_displacementTrochFilt)
195         if isempty(DTDD.m_displacementTroch)
196             error('DropTowerDisplacement:DataAvailability ','Trochanter displacement was
197             → requested for %s before displacement data was available.\n',DTDD.
198             → GetSpecimen().GetSpecimenName() );
199         end
200         DTDD.CalcFilteredData();
201     end
202     o = DTDD.m_displacementTrochFilt;
203 end
204
205 function o = GetDisplacementHammer(DTDD)
206     % A function to get the filtered impact hammer displacement data
207     % in m.
208     %
209     % Displacement = DTDD.GetDisplacementHammer()
210     %
211     if isempty(DTDD.m_displacementTrochFilt)
212         if isempty(DTDD.m_displacementTroch)
213             error('DropTowerDisplacement:DataAvailability ','Hammer displacement was
214             → requested for %s before displacement data was available.\n',DTDD.
215             → GetSpecimen().GetSpecimenName() );
216         end
217         DTDD.CalcFilteredData();
218     end
219     o = DTDD.m_displacementHammerFilt;
220 end
221
222 function o = GetTimeDisplacement(DTDD)
223     % A function to get the time in seconds in the displacement data
224     % time frame. If there is an offset between the first data
225     % point in the displacement this data will not line up with the
226     % experimental data. Use DTDD.GetTime() to get time in the
227     % experimental time frame.
228     %
229     % Time = DTDD.GetTimeDisplacement()
230     %
231     o = DTDD.m_timeDisplacement;
232 end
233
234 function o = GetTime(DTDD)
235     % A function to get the time in seconds in the experimental
236     % time frame. If no start time or sample rate has been set, a
237     % warning will be issued.
238     %
239     % The original data given by TEMA was rounded to the nearest
240     % 0.1 ms. This function returns a time vector of the correct
241     % length that has a spacing of 1/GetSampleRate(), resulting
242     % in a smooth time vector.
243     %
244

```

```

238     % Time = DTDD.GetTime()
239     %
240     if isempty(DTDD.GetSampleRate())
241         error('DropTowerDisplacement:DataAvailability','Time was requested for %s before
242             → sample rate was supplied. The time will not be referenced to the
243             → experiment.\n',DTDD.GetSpecimen().GetSpecimenName());
244     end
245     if isempty(DTDD.GetTimeStart())
246         warning('DropTowerDisplacement:DataAvailability','Time was requested for %s
247             → before start time was supplied. The time will not be referenced to the
248             → experiment.\n',DTDD.GetSpecimen().GetSpecimenName());
249         DTDD.SetTimeStart(0);
250     end
251     o = (0:1/DTDD.GetSampleRate():(length(DTDD.GetTimeDisplacement)-1)/DTDD.GetSampleRate
252         → ()) + DTDD.GetTimeStart();
253
254
255     function Update(DTDD)
256         % A function to update the state of the drop tower
257         % displacement data object. This method does not call
258         % ReadFile(), which must be done by the user.
259         %
260         % DTDD.Update()
261         %
262
263         % check if sample rate and start time have been set
264         if isempty(DTDD.GetSampleRate())
265             error('DropTowerDisplacement:DataAvailable','Update was called for %s when no
266                 → sample rate has been set.\n',DTDD.GetSpecimen().GetSpecimenName());
267         end
268         if isempty(DTDD.GetTimeStart())
269             error('DropTowerDisplacement:DataAvailable','Update was called for %s when no
270                 → start time has been set.\n',DTDD.GetSpecimen().GetSpecimenName());
271
272         % check for raw data
273         if (isempty(DTDD.GetDisplacementHammerUnfiltered) || isempty(DTDD.
274             → GetDisplacementTrocUnfiltered))
275             error('DropTowerDisplacement:DataAvailable','Update was called for %s when no raw
276                 → data is available.\nPerhaps call DTDD.ReadFile().\n',DTDD.GetSpecimen().
277                 → GetSpecimenName());
278
279         % filter the data
280         DTDD.CalcFilteredData();
281
282     end
283
284     function PrintSelf(DTDD)
285         % A function to print the current state of the displacement
286         % data object
287         %
288         % DTDD.PrintSelf()
289         %
290         fprintf(1,'\\n%%%%%% DTDisplacementData Class Parameters %%%%%%\n');
291         DTDD.GetSpecimen().PrintSelf();
292         fprintf(1,'\\n %%% Scalar Inputs %%%\n');
293         fprintf(1,'File name: %s\n',DTDD.GetFileName());
294         fprintf(1,'Time of first data point in experiment time: %f seconds\n',DTDD.
295             → GetTimeStart());
296         fprintf(1,'Sample rate: %f Hz\n',DTDD.GetSampleRate());
297         fprintf(1,'Filterind cutoff: %f Hz\n',DTDD.GetFilterCutoff());
298         fprintf(1,'Filter order: %d\n',DTDD.GetFilterOrder());
299
300         fprintf(1,'\\n %%% Vector Outputs %%%\n');
301         fprintf(1,'Time in displacement time frame: [%d,%d] seconds\n',size(DTDD.
302             → GetTimeDisplacement()));
303         fprintf(1,'Time in experimental time frame: [%d,%d] seconds\n',size(DTDD.GetTime()));

```

```

293     fprintf(1,'Filtered displacement of the trochanter: [%d,%d] m\n',size(DTDD.
294         ↪ GetDisplacementTroch()));
295     fprintf(1,'Raw displacement of the trochanter: [%d,%d] m\n',size(DTDD.
296         ↪ GetDisplacementTrochUnfiltered()));
297     fprintf(1,'Filtered displacement of the hammer: [%d,%d] m\n',size(DTDD.
298         ↪ GetDisplacementHammer()));
299     fprintf(1,'Raw displacement of the hammer: [%d,%d] m\n',size(DTDD.
300         ↪ GetDisplacementHammerUnfiltered()));
301 end % public methods
302
303 methods (Access = private , Hidden = true)
304
305     function CalcFilteredData(DTDD)
306         % A function to calculate the filtered data.
307         %
308         % DTDD.CalcFilteredData()
309         %
310         if (isempty(DTDD.m_sampleRate) || isempty(DTDD.m_filterCutoff) || isempty(DTDD.
311             ↪ m_filterOrder) )
312             error('DropTowerDisplacement:DataAvailability','Filtering was requested for %s
313                 ↪ when either sample rate, filter cutoff or filter order had not been set.\n
314                 ↪ ',DTDD.GetSpecimen().GetSpecimenName());
315         end
316         cutoffNormal = DTDD.m_filterCutoff / DTDD.m_sampleRate;
317         [b,a] = butter(DTDD.m_filterOrder/2,cutoffNormal); % divide filter order by two since
318             ↪ filtfilt doubles the order
319
320         troch = filtfilt(b,a,DTDD.GetDisplacementTrochUnfiltered());
321         hammer = filtfilt(b,a,DTDD.GetDisplacementHammerUnfiltered());
322
323         % zero the data at the start
324         troch(:,1) = troch(:,1) - troch(1,1);
325         troch(:,2) = troch(:,2) - troch(1,2);
326         hammer(:,1) = hammer(:,1) - hammer(1,1);
327         hammer(:,2) = hammer(:,2) - hammer(1,2);
328         DTDD.m_displacementTrochFilt = troch;
329         DTDD.m_displacementHammerFilt = hammer;
330     end
331
332 end % private methods
333 end % classdef

```

E.1.9 DIC data class

```

1 classdef DICData < handle
2     properties (SetAccess = private , Hidden = false)
3         m_specimen      % the specimen data class
4     end
5
6     properties (SetAccess = private , Hidden = true)
7         m_dicData        % strain
8         m_dicTime        % seconds
9         m_dicStartTime    % seconds
10        m_dicSampleRate   % Hz
11        m_dicDataFile = ''; % string
12    end
13    methods
14        % Constructor function
15        function DD = DICData(specimen)
16            % Constructor requires a reference specimen. See Specimen.m
17            % for details on creating a specimen.

```

```

18      %
19      % DD = DICData(specimen)
20      %
21      DD.m_specimen = specimen;
22  end
23
24  function SetFileName(DD,fileName)
25      % A function to set the name of the DIC data file .
26      %
27      % DD.SetFileName(file)
28      %
29      while (~exist(fileName,'file'))
30          fprintf(1,'The specified DIC data file does not exist for specimen %s\n',DD.
31                  ← GetSpecimen.GetSpecimenName);
32          fileName = input('Please enter a valid file location: ');
33          end
34          DD.m_dicDataFile = fileName;
35  end
36  function o = GetFileName(DD)
37      % A function to get the name of the DIC data file
38      %
39      % File = DD.GetFileName()
40      %
41      o = DD.m_dicDataFile;
42  end
43
44  function SetStartTime(DD,startTime)
45      % A function to set the time of the first data point in the
46      % DIC data in seconds. Time in the input file starts from
47      % zero, in some cases this may not be coincident with the
48      % trigger. This value is used to align the data with the
49      % experiment time.
50      %
51      % DD.SetStartTime(time)
52      %
53      DD.m_dicStartTime = startTime; % must be supplied by user
54  end
55
56  function SetSampleRate(DD,rate)
57      % A function to set the data frequency of the DIC in Hz.
58      %
59      % DD.SetSampleRate(rate)
60      %
61      DD.m_dicSampleRate = rate;
62  end
63  function o = GetSampleRate(DD)
64      % A function to get the sample rate of the DIC data in Hz.
65      %
66      % Rate = DD.GetSampleRate()
67      %
68      o = DD.m_dicSampleRate;
69  end
70
71  function o = GetSpecimen(DD)
72      % A function to return the specimen object associated with
73      % the DIC data.
74      %
75      % Specimen = DD.GetSpecimen()
76      %
77      o = DD.m_specimen;
78  end
79
80  function o = GetStrainData(DD)
81      % A function to get the DIC strain data vector read in from
82      % the input file. In the same units provided in the input
83      % file . Should be percent minimum principal strain.
84      %
85      % Strain = DD.GetStrainData()

```

```

85      %
86      o = DD.m_dicData;
87  end
88
89  function o = GetDICTime(DD)
90      % A function to get the DIC time in seconds. This time may
91      % not be aligned with the experiment time. The offset is
92      % set/get using DD.SetStartTime(time) and DD.GetTimeStart()
93      %
94      % Time = DD.GetDICTime()
95      %
96      o = DD.m_dicTime;
97  end
98
99  function o = GetTime(DD)
100     % A function to get the DIC time in seconds, aligned with the
101     % experimental time. If the DIC start time has not been set
102     % using SetStartTime(time) a warning will be issued.
103     %
104     % Time = DD.GetTime()
105     %
106     if isempty(DD.m_dicStartTime)
107         warning('DICData:DataAvailable','Warning! There is no start time for the DIC.\n'
108             '→ nTime will not be referenced to the rest of the experiment.\n');
109     end
110     o = DD.m_dicTime + DD.m_dicStartTime;
111 end
112
113 function o = GetTimeStart(DD)
114     % A function to get the time of the first DIC data point in
115     % seconds.
116     %
117     % Time = DD.GetTimeStart()
118     %
119     o = DD.m_dicStartTime;
120 end
121
122 function ReadFile(DD)
123     % A function to read the DaVis strain output file.
124     % Strain must be percent minimum principal strain for this
125     % class to integrate properly with the rest of the analysis
126     %
127     % DD.ReadDataFile()
128     %
129     if strcmp(DD.GetFileName(), '')
130         error('DICData:DataAvailable','The DIC ReadDataFile() method was called for %'
131             '→ before a file name was set.\n',DD.GetSpecimen().GetSpclmenName());
132     end
133     inFid = fopen(DD.m_dicDataFile,'r');
134     if inFid == -1
135         fclose(inFid);
136         error('DICData:ReadError','The DIC data file specified for %s does not exist.'
137             '→ Please check the file name and try again.\n',DD.GetSpecimen().
138             GetSpclmenName());
139     end
140     fgetl(inFid);           % skip headerline
141     cline = fgetl(inFid);   % read in the y axis
142     %#ok<ST2NM> - Suppress the strin to num warning
143     DD.m_dicData = str2num(cline)/100; % convert to number and make % into strain
144     cline = fgetl(inFid);   % read in the x axis
145     DD.m_dicTime = str2num(cline); % convert to number
146     fclose(inFid);
147
148 function o = GetStrainAtTime(DD,time)
149     % A function to get the strain at a experimental time in seconds.
150     %
151     % Strain = DD.GetStrainAtTime(time)

```

```

149      %
150      dicTime = time - DD.m_dicStartTime;
151      index = find(DD.m_dicTime > dicTime, 1, 'first');
152      o = DD.m_dicData(index);
153  end
154
155  function PrintSelf(DD)
156      % A function to print out the state of the DIC data object.
157      %
158      % DD.PrintSelf()
159      %
160      fprintf(1, '\n%%%%%%%%% DICData Class Parameters %%%%%%%%%\n');
161      DD.GetSpecimen().PrintSelf();
162      fprintf(1, 'DIC file name: %sf\n', DD.m_dicDataFile);
163      fprintf(1, 'DIC sample rate: %f Hz\n', DD.m_dicSampleRate);
164      fprintf(1, 'DIC start time: %f seconds\n', DD.m_dicStartTime);
165
166      fprintf(1, '\n %%%% DIC Data %%%%\n');
167      fprintf(1, 'DIC strain: [%d,%d] in percent strain (or as defined in input file)\n',
168             size(DD.m_dicData));
169      fprintf(1, 'DIC time: [%d,%d] in seconds\n\n', size(DD.m_dicTime));
170  end
171 end
172 end

```

E.2 Idealized fall simulator model

The idealized model discussed in §4.4 was solved using MatLab (R2010b, The Mathworks, Natick, MA). Code used to solve the equations and create the plots is given below. The code contained herein is available for download from
<https://github.com/sethgilchrist/IdealFallSimModel>.

E.2.1 Solving the system ODE

```

1 function [displacementRate, loadingRate] = stiffVsRatesData(kFemur, bFemur)
2 % This function takes a vector of femoral stiffnesses (length = s) and a
3 % vector of femoral damping values (length = d) as inputs and returns two
4 % d by s matrices of displacement and loading rates. Each row corresponds
5 % to a damping value and each column to a stiffness value.
6
7 % define stiffnesses
8 kPlevis = 50000; %N/m
9
10 % preallocate
11 displacementRate = zeros(length(bFemur), length(kFemur));
12 loadingRate = zeros(size(displacementRate));
13
14 % solve the ode
15 for i = 1:size(displacementRate, 1)
16     for j = 1:size(displacementRate, 2)
17         % get the time and displacement values from the ODE solver
18         [t, x] = displacementSimple(kFemur(j), kPlevis, bFemur(i));
19         % find the first time that the compression and damping are greater
20         % than 1500 N

```

```

21      cIndex = find(x(:,3)*kFemur(j)+x(:,4)*bFemur(i) > 1500,1,'first')-1;
22      % calculate the displacement rate
23      displacementRate(i,j) = x(cIndex,3)/t(cIndex);
24      % calculate the loading rate based on the average displacement rate
25      % for the velocity of the compression
26      loadingRate(i,j) = (x(cIndex,3)*kFemur(j) + bFemur(i)*displacementRate(i,j))/t(cIndex);
27  end
28 end
29 end
30
31
32 function [t,x] = displacementSimple(kf,ks,bf)
33 % A simple model of the fall simulator consisting of two masses, two
34 % springs and one damper.
35 % Usage
36 % [t,x] = displacementSimple(femurStiffness ,springStiffness ,femurDamping)
37 %
38 % t = vector of time
39 % x(1) = displacement of top of spring
40 % x(2) = velocity of top of spring
41 % x(3) = displacement of trochanter
42 % x(4) = velocity of trochanter
43
44 springMass = 3.5; %kg
45
46 M1 = springMass/3 + 1 + 32; % kg: top 1/3 spring + top plate + body mass
47 M2 = springMass/3 + 10 + 1.98; %kg: bottom 1/3 spring + bottom plate + stabalizer + loadcell +
    ↳ loader + falling mass
48
49 initialConditions = [0,2.9,0,0];
50
51 [t,x] = ode45(@(t,x) system(t,x,M1,M2,kf,ks,bf),linspace(0,0.05,5000),initialConditions);
52 end
53
54 function dxdt = system(t,x,M1,M2,Kf,Ks,Bf)
55 if t < 0.005
56     f2pulse = 250*sin(2*pi/.01*t); % repace this line with "f2pulse = 0" to omit the force pulse
        ↳ on mass 2
57 else
58     f2pulse = 0;
59 end
60
61 xM1 = x(1);
62 xM1dot = x(2);
63 xM2 = x(3);
64 xM2dot = x(4);
65
66 dxdt = zeros(size(x));
67 dxdt(1) = xM1dot;
68 dxdt(2) = 1/M1 * (9.81*M1 + (xM2-xM1)*Ks);
69 dxdt(3) = xM2dot;
70 dxdt(4) = 1/M2 * (9.81*M2 + f2pulse - xM2*Kf - (xM2-xM1)*Ks - xM2dot*Bf);
71 end

```

E.2.2 Plotting the system response

```

1 % house keeping
2 close all
3 clear all
4
5 % femur stiffness values to analyse
6 kFemur = (.4:.25:4.5)*1000000; %N/m
7 % femur damping values to analyse
8 bFemur = [0 300 600 4000]; %Ns/m
9
10 % calculate the loading and displacement rates for the different

```

```

11 % stiffnesses and damping values
12 [disp,load] = stiffVsRatesData(kFemur,bFemur);
13
14 % create the figure
15 fH1 = figure(1);
16 set(fH1,'position',[463 29 1066 945], 'paperpositionMode', 'auto');
17 for i = 1:4
18     % calculate a subplot position
19     aH = subplot(2,2,i);
20     % plot and get plot object handles
21     [aHyy(i,:),L1(i),L2(i)] = plotyy(kFemur/1000000,load(i,:)/1000,kFemur/1000000,disp(i,:)*1000
22         ↪ ; %#ok<*SAGROW*
23     % move plotyy axes to the right place based on the subplot location
24     oP = get(aH,'position'); % original position
25     set(aHyy(i,1),'position',[oP(1)-.025 oP(2:4)],... % location
26         'ycolor','k','fontname','times','fontsize',20,... % axes appearance
27         'xlim',[0 5],'xtick',[0 1 2 3 4 5],... % x limits and ticks
28         'ylim',[0 300],'ytick',[0 150 300]); % y limits and ticks
29
30     set(aHyy(i,2),'position',[oP(1)-.025 oP(2:4)],... % location
31         'ycolor','k','fontname','times','fontsize',20,... % axes appearance
32         'xlim',[0 5],'xtick',[0 1 2 3 4 5],... % x limits and ticks
33         'ylim',[0 400],'ytick',[0 200 400]); % y limits and ticks
34
35     grid(aHyy(i,1)); % turn the grid on for one axes
36     % set the data styles
37     set(L1(i),'marker','o','markersize',10,'linestyle','none','markerEdgeColor','k','lineWidth
38         ↪ ',2)
39     set(L2(i),'marker','x','markersize',10,'linestyle','none','markerEdgeColor','k','lineWidth
40         ↪ ',2)
41     % calculate fit lines
42     loadingFit(i,:) = polyfit(kFemur,load(i,:),1);
43     dispFit(i,:) = polyfit(kFemur,disp(i,:),1);
44     % plot the fit lines
45     hold(aHyy(i,1))
46     plot(aHyy(i,1),kFemur/1000000,polyval(loadingFit(i,:),kFemur)/1000,'--k','linewidth',2)
47     hold(aHyy(i,2))
48     plot(aHyy(i,2),kFemur/1000000,polyval(dispFit(i,:),kFemur)*1000,'--k','linewidth',2)
49     % set the text with the damping values in the upper left corner
50     damping = sprintf('%0.1f',bFemur(i)/1000);
51     hText = text(.3,260,[damping ' $$\frac{N}{mm\cdot s}$$'], 'fontname','times','fontsize',25,
52         ↪ 'interpreter','latex','HorizontalAlignment','left','backgroundcolor','w');
53 end
54 % put axes labels in position
55 LyP = get(get(aHyy(3,1),'ylabel'),'position');
56 set(get(aHyy(3,1),'ylabel'), 'string', 'Loading Rate (kN/s)', 'fontname', 'times', 'fontsize',30,
57     ↪ 'position',[LyP(1) 375 1])
58 BxP = get(get(aHyy(3,1),' xlabel'),'position');
59 set(get(aHyy(3,1),' xlabel'), 'string', 'Stiffness (kN/mm)', 'fontname', 'times', 'fontsize',30,
60     ↪ 'position',[6 BxP(2) 1])
61 RyP = get(get(aHyy(4,2),' ylabel'),'position');
62 set(get(aHyy(4,2),' ylabel'), 'string', 'Displacement Rate (mm/s)', 'fontname', 'times', 'fontsize
63     ↪ ',30,'position',[RyP(1) 500 1])
64
65 % put tips in the first plot to indicate which curves are loading and
66 % displacement.
67 set(fH1,'currentAxes',aHyy(1,1));
68 hLoading = text(3.15,225,'Loading','fontname','times','fontsize',20);
69 set(fH1,'currentAxes',aHyy(1,2));
70 hDisplace = text(2.5,50,'Displacement','fontname','times','fontsize',20,'HorizontalAlignment',
71     ↪ 'right');
72
73 % save the plots as eps (for latex), png (for review) and fig (for easy editing)
74 print(fH1,'./ratesVsDamping.eps','r300','deps');
75 print(fH1,'ratesVsDamping.png','r150','dpng');
76 saveas(fH1,'ratesVsDamping.fig')

```

E.3 DIC to DIC registration and comparison

This section contains C++ code used to register two surfaces created by DaVis and transfer a single set of point-data from one to the other. The code contained herein is available for download at <https://github.com/sethgilchrist/RegisterCompareDICSurfaces>.

E.3.1 Functions library for converting DaVis to VTK format

Header file:

```

1 // ReadInputFiles.h
2 //
3 // Copyright 2011 Seth Gilchrist <seth@sethgilchrist.com>
4 //
5 // This program is free software; you can redistribute it and/or modify
6 // it under the terms of the GNU General Public License as published by
7 // the Free Software Foundation; either version 3 of the License, or
8 // (at your option) any later version.
9 //
10 // This program is distributed in the hope that it will be useful,
11 // but WITHOUT ANY WARRANTY; without even the implied warranty of
12 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 // GNU General Public License for more details.
14 //
15 // You should have received a copy of the GNU General Public License
16 // along with this program; if not, write to the Free Software
17 // Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
18 // MA 02110-1301, USA.
19
20 #ifndef READDAVIS_H
21 #define READDAVIS_H
22
23
24 #include <cstring>
25 #include <sstream>
26 #include <iterator>
27 #include <boost/tokenizer.hpp>
28 #include <vtkSmartPointer.h>
29 #include <vtkPointData.h>
30 #include <vtkPolyData.h>
31 #include <vtkImageData.h>
32 #include <vtkDoubleArray.h>
33 #include <vtkDelaunay2D.h>
34
35
36 class ReadDaVis
37 {
38 public:
39     /** Constructor */
40     ReadDaVis();
41
42     /** Set/Get the height file name.*/
43     void SetHeightFileName( std::string fileName );
44     std::string GetHeightFileName();
45     /** Set the strain file name. */
46     void SetStrainFileName( std::string fileName );
47
48     /** Read the height file */
49

```

```

50     void ReadHeightFile();
51     /** Read the strain file */
52     void ReadStrainFile();
53     /** Read a file given in fileName and put the results in the pointset */
54     void ReadFile(std::string fileName, vtkSmartPointer<vtkImageData> pointData);
55     /** Put the height data into a surface and put the z-comp of the strain
56      * point data as a dataset at the points of the height data. */
57     void CreateDataSurface();
58
59     /** Get the height point data. */
60     vtkSmartPointer<vtkImageData> GetHeightData();
61     /** Get the strain point data. */
62     vtkSmartPointer<vtkImageData> GetStrainData();
63     /** Get the surface. */
64     vtkSmartPointer<vtkPolyData> GetSurface();
65     // vtkSmartPointer<vtkUnstructuredGrid> GetSurface();
66
67
68 private:
69     std::string m_heightFileName;
70     std::string m_strainFileName;
71     vtkSmartPointer<vtkImageData> m_heightData;
72     vtkSmartPointer<vtkImageData> m_strainData;
73     vtkSmartPointer<vtkPolyData> m_surface;
74     // vtkSmartPointer<vtkUnstructuredGrid> m_surface;
75
76 };
77 #endif

```

C++ file:

```

1 //      ReadDaVis.cpp
2 //
3 //      Copyright 2011 Seth Gilchrist <seth@sethgilchrist.com>
4 //
5 //      This program is free software; you can redistribute it and/or modify
6 //      it under the terms of the GNU General Public License as published by
7 //      the Free Software Foundation; either version 3 of the License, or
8 //      (at your option) any later version.
9 //
10 //      This program is distributed in the hope that it will be useful,
11 //      but WITHOUT ANY WARRANTY; without even the implied warranty of
12 //      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 //      GNU General Public License for more details.
14 //
15 //      You should have received a copy of the GNU General Public License
16 //      along with this program; if not, write to the Free Software
17 //      Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
18 //      MA 02110-1301, USA.
19
20 #include "ReadDaVis.h"
21
22 /** Constructor */
23 ReadDaVis::ReadDaVis()
24 {
25     // m_heightFileName; // Must be provided by user
26     // m_strainFileName; // Must be provided by user
27     m_heightData = vtkSmartPointer<vtkImageData>::New();
28     m_strainData = vtkSmartPointer<vtkImageData>::New();
29     m_surface = vtkSmartPointer<vtkPolyData>::New();
30     // m_surface = vtkSmartPointer<vtkUnstructuredGrid>::New();
31 }
32
33 void ReadDaVis::SetHeightFileName( std::string fileName )
34 {
35     if (m_heightFileName.compare(fileName) != 0) {m_heightFileName = fileName;}
36 }
37 std::string ReadDaVis::GetHeightFileName()

```

```

38 {
39     return m_heightFileName;
40 }
41
42 void ReadDaVis::SetStrainFileName( std::string fileName )
43 {
44     if (m_strainFileName.compare(fileName) != 0) {m_strainFileName = fileName;}
45 }
46
47 void ReadDaVis::ReadHeightFile()
48 {
49     ReadFile(m_heightFileName, m_heightData);
50 }
51
52 void ReadDaVis::ReadStrainFile()
53 {
54     ReadFile(m_strainFileName, m_strainData);
55 }
56
57 void ReadDaVis::ReadFile(std::string fileName, vtkSmartPointer<vtkImageData> pointData)
58 {
59     // create a stringstream for passing info to the user
60     std::stringstream msg(" ");
61     // open the file
62     std::ifstream inFile(fileName.c_str());
63     if (!inFile){
64         msg.str(" ");
65         msg << "Cannot open\n" <<fileName << "\nPlease check the name and try again."<<std::endl;
66         std::cout << msg;
67     }
68
69     // Get the header line
70     std::string headerLine;
71     std::getline(inFile,headerLine);
72     std::vector<std::string> headerTokens;
73     boost::char_separator<char> sep(" ");
74     boost::tokenizer< boost::char_separator<char> > tok(headerLine,sep); // the boost library
    ↪ tokenizer
75     for (boost::tokenizer< boost::char_separator<char> >::iterator beg=tok.begin(); beg!=tok.end
    ↪ () ;++beg){
76         headerTokens.push_back(*beg);
77     }
78
79     int xDimension = atoi(headerTokens[3].c_str());
80     float xScale = atof(headerTokens[6].c_str());
81     float xOffset = atof(headerTokens[7].c_str());
82     int yDimension = atoi(headerTokens[4].c_str());
83     float yScale = atof(headerTokens[10].c_str());
84     float yOffset = atof(headerTokens[1].c_str());
85     pointData->SetDimensions(xDimension,yDimension,1);
86     pointData->SetOrigin(xOffset,yOffset,0);
87     pointData->SetSpacing(xScale,yScale,1);
88
89     // now step through the file and create the points
90     std::string cline;
91     unsigned int i = 0; // the x point number, will be used with x-scale and x-offset to produce
    ↪ an point
92
93     while( std::getline (inFile ,cline))
94     {
95         // break the current line into values of height
96         std::vector<double> values;
97         boost::tokenizer< boost::char_separator<char> > tok(cline,sep);
98         for (boost::tokenizer< boost::char_separator<char> >::iterator beg=tok.begin(); beg!=tok.
    ↪ end();++beg)
99         {
100             std::string current = *beg;
101             values.push_back(atof(current.c_str()));

```

```

102     }
103     for (int j = 0; j < yDimension; ++j)
104     {
105         pointData->SetScalarComponentFromDouble(i ,j ,0 ,0 ,values[ j ]);
106     }
107     ++i;
108
109 }
110 inFile .close ();
111
112 }
113
114 void ReadDaVis :: CreateDataSurface ()
115 {
116     // find out which point set has fewer points
117     int* heightDimensions = m_heightData->GetDimensions ();
118     int* strainDimensions = m_strainData->GetDimensions ();
119     int xEnd = (heightDimensions[0] > strainDimensions[0]) ? strainDimensions[0] :
120         ↪ heightDimensions[0];
121     int yEnd = (heightDimensions[1] > strainDimensions[1]) ? strainDimensions[1] :
122         ↪ heightDimensions[1];
123
124     vtkSmartPointer<vtkPoints> surfacePoints = vtkSmartPointer<vtkPoints >::New();
125     vtkSmartPointer<vtkDoubleArray> surfaceArray = vtkSmartPointer<vtkDoubleArray >::New();
126     surfaceArray->SetNumberOfComponents(1);
127     surfaceArray->SetName("MinPStrain");
128
129     // iterate through x and y, if the data for both != 0, then save the point
130     for (int i = 0; i < xEnd; ++i)
131     {
132         for (int j = 0; j < yEnd; ++j)
133         {
134             double heightPoint = m_heightData->GetScalarComponentAsDouble(i ,j ,0 ,0 );
135             double strainPoint = m_strainData->GetScalarComponentAsDouble(i ,j ,0 ,0 );
136             if (heightPoint!=0 && strainPoint!=0)
137             {
138                 int pointIndex[3];
139                 pointIndex[0] = i ;
140                 pointIndex[1] = j ;
141                 pointIndex[2] = 0;
142                 double* pointLocation = m_heightData->GetPoint(m_heightData->ComputePointId(
143                     ↪ pointIndex));
144                 surfacePoints->InsertNextPoint(pointLocation[0],pointLocation[1],heightPoint);
145                 surfaceArray->InsertNextTuple1(strainPoint);
146
147             }
148         }
149     }
150     m_surface->SetPoints(surfacePoints);
151     m_surface->GetPointData ()->AddArray(surfaceArray);
152
153     // use Delaunay2D to create a mesh, ignoring the z-dimension
154     vtkSmartPointer<vtkDelaunay2D> delauney = vtkSmartPointer<vtkDelaunay2D >::New();
155     delauney->SetInput(m_surface);
156     delauney->Update();
157     m_surface = delauney->GetOutput();
158
159 }
160
161 vtkSmartPointer<vtkImageData> ReadDaVis :: GetHeightData ()
162 {
163     return m_heightData;
164 }
165
166 vtkSmartPointer<vtkImageData> ReadDaVis :: GetStrainData ()
167 {

```

```

167     return m_strainData;
168 }
169
170 vtkSmartPointer<vtkPolyData> ReadDaVis::GetSurface()
171 {
172     return m_surface;
173 }
```

E.3.2 Main program for converting output from DaVis to VTK format

```

1  /*
2  * ConvertSurfaces.cpp
3  *
4  * Copyright 2013 Seth Gilchrist <seth@sethgilchrist.com>
5  *
6  * This program is free software; you can redistribute it and/or modify
7  * it under the terms of the GNU General Public License as published by
8  * the Free Software Foundation; either version 3 of the License, or
9  * (at your option) any later version.
10 *
11 * This program is distributed in the hope that it will be useful,
12 * but WITHOUT ANY WARRANTY; without even the implied warranty of
13 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14 * GNU General Public License for more details.
15 *
16 * You should have received a copy of the GNU General Public License
17 * along with this program; if not, write to the Free Software
18 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
19 * MA 02110-1301, USA.
20 *
21 *
22 */
23
24 #include "../lib/ReadDaVis/ReadDaVis.h"
25 #include <vtkIterativeClosestPointTransform.h>
26 #include <vtkLandmarkTransform.h>
27 #include <vtkTransformPolyDataFilter.h>
28 #include <vtkXMLPolyDataWriter.h>
29
30 int main(int argc, char **argv)
31 {
32     if (argc < 6)
33     {
34         std::cerr << "Not enough inputs.\nUsage:" << std::endl;
35         std::cerr << argv[0] << " [DropTower Surface] [DropTower Strain] [Instron Surface] [
36             ↳ Instron Strain] [Output Folder]" << std::endl << "Aborted." << std::endl;
37     }
38     if (argc > 6)
39     {
40         std::cout << "Too many inputs give. Extra inputs ignored.\nUsage:" << std::endl;
41         std::cout << argv[0] << " [DropTower Surface] [DropTower Strain] [Instron Surface] [
42             ↳ Instron Strain] [Output Folder]" << std::endl;
43     }
44
45     ReadDaVis *dtReader = new ReadDaVis;
46     dtReader->SetHeightFileName(argv[1]);
47     dtReader->SetStrainFileName(argv[2]);
48
49     ReadDaVis *inReader = new ReadDaVis;
50     inReader->SetHeightFileName(argv[3]);
51     inReader->SetStrainFileName(argv[4]);
52
53     std::cout << "Reading drop tower file..." << std::endl;
      dtReader->ReadHeightFile();
```

```

54     dtReader->ReadStrainFile();
55     dtReader->CreateDataSurface();
56     std::cout<<"Drop tower file successfully read. Number of points in droptower surface: "<<
57         <> dtReader->GetSurface()->GetNumberOfPoints()<<std::endl;
58
59     std::cout<<"Reading instron file..."<<std::endl;
60     inReader->ReadHeightFile();
61     inReader->ReadStrainFile();
62     inReader->CreateDataSurface();
63     std::cout<<"Instron file successfully read. Number of points in instron surface: "<<
64         <> inReader->GetSurface()->GetNumberOfPoints()<<std::endl;
65
66     std::string outPath = argv[5];
67     int pathLength = outPath.length();
68     if (outPath.compare(pathLength - 1, 1, "/"))
69     {
70         outPath.append("/");
71
72     std::string dtOutFile = outPath;
73     dtOutFile.append("dropTowerSurface.vtp");
74     std::string inOutFile = outPath;
75     inOutFile.append("instronSurface.vtp");
76
77     std::cout<<"Writing the drop tower surface to "<<dtOutFile<<std::endl;
78     vtkSmartPointer<vtkXMLPolyDataWriter> dtWriter = vtkSmartPointer<vtkXMLPolyDataWriter>::
79         New();
80     dtWriter->SetInput(dtReader->GetSurface());
81     dtWriter->SetFileName(dtOutFile.c_str());
82     dtWriter->Write();
83     std::cout<<"Drop tower file successfully written."<<std::endl;
84
85     std::cout<<"Writing the instron surface to "<<inOutFile<<std::endl;
86     vtkSmartPointer<vtkXMLPolyDataWriter> polyWriter = vtkSmartPointer<vtkXMLPolyDataWriter>::
87         New();
88     polyWriter->SetInput(inReader->GetSurface());
89     polyWriter->SetFileName(inOutFile.c_str());
90     polyWriter->Write();
91     std::cout<<"Instron file successfully written."<<std::endl;
92 }

```

CMakeLists file:

```

1 cmake_minimum_required(VERSION 2.6)
2 project( ConvertSurfaces )
3 FIND_PACKAGE(VTK)
4
5 IF(VTK_FOUND)
6     INCLUDE(${VTK_USE_FILE})
7 ELSE(VTK_FOUND)
8     MESSAGE(FATAL_ERROR
9         "VTK not found. Please set VTK_DIR.")
10 ENDIF(VTK_FOUND)
11
12 ADD_LIBRARY( ReadDaVis .. / lib / ReadDaVis / ReadDaVis .cpp )
13 ADD_EXECUTABLE( ConvertSurfaces ConvertSurfaces .cpp )
14 TARGET_LINK_LIBRARIES( ConvertSurfaces ReadDaVis ${ITK_LIBRARIES} vtkHybrid )

```

E.3.3 Functions library for comparing strains stored on two surfaces

Header file:

```

1 /*
2  * CompareSurfaces-InputTransform.h
3  *
4  * Copyright 2013 Seth Gilchrist <seth@fake.com>
5  */

```

```

6  * This program is free software; you can redistribute it and/or modify
7  * it under the terms of the GNU General Public License as published by
8  * the Free Software Foundation; either version 3 of the License, or
9  * (at your option) any later version.
10 *
11 * This program is distributed in the hope that it will be useful,
12 * but WITHOUT ANY WARRANTY; without even the implied warranty of
13 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14 * GNU General Public License for more details.
15 *
16 * You should have received a copy of the GNU General Public License
17 * along with this program; if not, write to the Free Software
18 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
19 * MA 02110-1301, USA.
20 *
21 *
22 */
23
24 #ifndef COMPARESURFACESINPUTTRANSFORMH
25 #define COMPARESURFACESINPUTTRANSFORMH
26
27 #include <vtkSmartPointer.h>
28 #include <vtkPolyData.h>
29 #include <vtkIterativeClosestPointTransform.h>
30 #include <vtkLandmarkTransform.h>
31 #include <vtkTransformPolyDataFilter.h>
32 #include <vtkTransform.h>
33 #include <vtkUnstructuredGrid.h>
34 #include <vtkIdList.h>
35 #include <vtkCellLocator.h>
36 #include <vtkWedge.h>
37 #include <vtkCell.h>
38 #include <vtkDoubleArray.h>
39 #include <vtkAppendFilter.h>
40 #include <vtkXMLPolyDataReader.h>
41 #include <vtkThreshold.h>
42 #include <vtkPointData.h>
43 #include <vtkCellArray.h>
44 #include <vtkXMLPolyDataWriter.h>
45
46 class CompareSurfaces
47 {
48     public:
49         CompareSurfaces();
50         virtual ~CompareSurfaces();
51
52         /** A function to get the input file reader */
53         vtkSmartPointer<vtkXMLPolyDataReader> GetRecieverReader()
54         {
55             return m_recieverReader;
56         }
57
58         vtkSmartPointer<vtkXMLPolyDataReader> GetDonorReader()
59         {
60             return m_donorReader;
61         }
62
63         /** A function to extrude the a surface. It extrudes the surface
64         * in the direction defined by a vector from the centroid of
65         * one of the input reader's surfaces to the other.*/
66         void ExtrudeSurface(vtkSmartPointer<vtkPolyData> surface, double direction[3]);
67
68         /** A function to get the volume created by ExtrudeSurface. */
69         vtkSmartPointer<vtkUnstructuredGrid> GetExtrudedVolume()
70         {
71             return m_extrudedVolume;
72         }
73

```

```

74  /** A function to get and return the centroid of a surface , or any
75   * poly data for that mater. The passed variable centroid is modified
76   * in place. */
77  void GetSurfaceCentroid( vtkSmartPointer<vtkPolyData> surface , double centroid[3]);
78
79  /** A function to probe an extruded volume. Returns the surface
80   * with data information from the volume projected onto it. */
81  vtkSmartPointer<vtkPolyData> ProbeVolume(vtkSmartPointer<vtkUnstructuredGrid> volume,
82                                         vtkSmartPointer<vtkPolyData> surface);
83
84  /** A function to align the surfaces , with the points set using
85   * SetInitialPoints as the initial transform. Returs recieverSurf
86   * transformed to be aligned with donorSurf. */
87  vtkSmartPointer<vtkPolyData> AlignSurfaces(vtkSmartPointer<vtkPolyData> recieverSurf ,
88                                              vtkSmartPointer<vtkPolyData> donorSurf);
89
90  /** A function to set the initial transform. A translation vector
91   * and rotation vector are provided and used to rigidly transform
92   * the reciever surface before the ICP alignment starts. */
93  void SetInitialTransform(double translate[3], double rotate[3])
94  {
95      if (m_translate[0] != translate[0]) m_translate[0] = translate[0];
96      if (m_translate[1] != translate[1]) m_translate[1] = translate[1];
97      if (m_translate[2] != translate[2]) m_translate[2] = translate[2];
98
99      if (m_rotate[0] != rotate[0]) m_rotate[0] = rotate[0];
100     if (m_rotate[1] != rotate[1]) m_rotate[1] = rotate[1];
101     if (m_rotate[2] != rotate[2]) m_rotate[2] = rotate[2];
102 }
103
104 /** reciever and donor are identical surfaces. The donor surface has
105  * the data transferred from the donor data. Data is compiled onto a
106  * single surface and and the names of the datasets are taken from
107  * those set in SetRecieverDataName() and SetDonorDataName(). The
108  * defalut names are "reciever" and "donor". A thrid dataset is
109  * created called "delta" which is the difference between the two and
110  * is calculated as (donor - reciever).
111 *
112 * Finally , the data is thresholded. The ProbeVolume() method
113 * uses a value of -1000000 in locations where there was no overlap
114 * between the volume ans surface. This step removes cells that
115 * have values in the "delta" field of <-999999. */
116 void CompileData( vtkSmartPointer<vtkPolyData> recieverSurf , vtkSmartPointer<vtkPolyData>
117                   → donorSurf);
118
119 /** A function to return the surface with the compiled data in it.
120  * The surface will have the data fields: DTStrain , InstronStrain
121  * and StrainDifference. */
122 vtkSmartPointer<vtkUnstructuredGrid> GetCompiledData()
123 {
124     return m_compiledSurf;
125 }
126
127 /** A function to set the reciever data name used when CompileData()
128  * is called. The default is "reciever".*/
129 void SetRecieverDataName(std::string recieverName)
130 {
131     if (m_recieverName.compare(recieverName))
132     {
133         m_recieverName = recieverName;
134     }
135 }
136
137 /** A function to set the donor data name used when CompileData()
138  * is called. The default is "donor". */
139 void SetDonorDataName(std::string donorName)
140 {
141     if (m_donorName.compare(donorName))

```

```

141         {
142             m_recieverName = donorName;
143         }
144     }
145
146     /** A function to write the strain difference to a file given as
147      * an input. The header will contain the format of the file.
148      * where the strains are in whatever units were specified in
149      * the input files. */
150     void WriteDataToFile(std::string fileName);
151
152
153 protected:
154 private:
155
156     /** Private types */
157     vtkSmartPointer<vtkXMLPolyDataReader> m_recieverReader;
158     vtkSmartPointer<vtkXMLPolyDataReader> m_donorReader;
159     double m_translate[3];
160     double m_rotate[3];
161     vtkSmartPointer<vtkUnstructuredGrid> m_compiledSurf;
162     vtkSmartPointer<vtkUnstructuredGrid> m_extrudedVolume;
163     std::string m_recieverName;
164     std::string m_donorName;
165
166 };
167
168 #endif // COMPARESURFACES-INPUTTRANSFORM.H

```

C++ file:

```

1  /*
2   * CompareSurfaces-InputTransform.cpp
3   *
4   * Copyright 2013 Seth Gilchrist <seth@fake.com>
5   *
6   * This program is free software; you can redistribute it and/or modify
7   * it under the terms of the GNU General Public License as published by
8   * the Free Software Foundation; either version 3 of the License, or
9   * (at your option) any later version.
10  *
11  * This program is distributed in the hope that it will be useful,
12  * but WITHOUT ANY WARRANTY; without even the implied warranty of
13  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14  * GNU General Public License for more details.
15  *
16  * You should have received a copy of the GNU General Public License
17  * along with this program; if not, write to the Free Software
18  * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
19  * MA 02110-1301, USA.
20  *
21  *
22  */
23
24 #include "CompareSurfaces-InputTransform.h"
25
26 CompareSurfaces::CompareSurfaces()
27 {
28
29     // create a new readers
30     m_recieverReader = vtkSmartPointer<vtkXMLPolyDataReader>::New();
31     m_donorReader = vtkSmartPointer<vtkXMLPolyDataReader>::New();
32     /* set the points for the initial transform. The default
33      * values don't transform anything. */
34     m_translate[0] = 0; m_translate[1] = 0; m_translate[2] = 0;
35     m_rotate[0] = 0; m_rotate[1] = 0; m_rotate[2] = 0;
36     // create the output surface
37     // m_compiledSurf = vtkSmartPointer<vtk>::New();

```

```

38     // create the extruded volume made from the donor
39     m_extrudedVolume = vtkSmartPointer<vtkUnstructuredGrid>::New();
40     // set the default data names
41     m_recieverName = "reciever";
42     m_donorName = "donor";
43 }
44
45 CompareSurfaces::~CompareSurfaces()
46 {
47     // destructor. Nothing to do here.
48 }
49
50 void CompareSurfaces::ExtrudeSurface(vtkSmartPointer<vtkPolyData> surf, double vect[3])
51 {
52     // make the vector 5 mm long
53     double length = sqrt(pow(vect[0],2)+pow(vect[1],2)+pow(vect[2],2));
54     double scale = 5/length;
55     vect[0] = vect[0]*scale;
56     vect[1] = vect[1]*scale;
57     vect[2] = vect[2]*scale;
58     // transform the surface to be extruded in the -vect
59     vtkSmartPointer<vtkTransform> transformer = vtkSmartPointer<vtkTransform>::New();
60     transformer->Translate(-vect[0], -vect[1], -vect[2]);
61     vtkSmartPointer<vtkTransformPolyDataFilter> polyMover = vtkSmartPointer<
62         > vtkTransformPolyDataFilter>::New();
63     polyMover->SetTransform(transformer);
64     polyMover->SetInput(surf);
65
66     // use the append filter to convert the PolyData to UnstructuredGrid
67     vtkSmartPointer<vtkAppendFilter> appender = vtkSmartPointer<vtkAppendFilter>::New();
68     appender->SetInputConnection(polyMover->GetOutputPort());
69     appender->Update();
70     vtkSmartPointer<vtkUnstructuredGrid> tempGrid = appender->GetOutput();
71
72     // iterate through the points
73     unsigned int originalNumberOfPoints = tempGrid->GetNumberOfPoints();
74     for (unsigned int i = 0; i < originalNumberOfPoints; ++i)
75     {
76         double cPoint[3];
77         tempGrid->GetPoint(i, cPoint);
78         // for each point, create a child point in the point data at a location 2*vect away
79         tempGrid->GetPoints()->InsertNextPoint(cPoint[0]+2*vect[0], cPoint[1]+2*vect[1], cPoint
80             <-> [2]+2*vect[2]);
81         // get the data for the parent point and copy it into the child point
82         tempGrid->GetPointData()->GetArray(0)->InsertNextTuple(tempGrid->GetPointData()->GetArray
83             <-> (0)->GetTuple(i));
84     }
85
86     // iterate through the cells (trangles) of the original data
87     unsigned int originalNumberOfCells = tempGrid->GetNumberOfCells();
88     for (unsigned int i = 0; i < originalNumberOfCells; ++i)
89     {
90         // for each cell, get a list of the defining points
91         vtkSmartPointer<vtkIdList> cPoints = vtkSmartPointer<vtkIdList>::New();
92         tempGrid->GetCellPoints(i, cPoints);
93         // create a list of points for the new, wedge cell. The points are the original list
94         // plus the child poit of each one of those. They are sepearted by the originial number
95         //      <- of points
96         vtkSmartPointer<vtkIdList> newPoints = vtkSmartPointer<vtkIdList>::New();
97         newPoints->InsertNextId(cPoints->GetId(0));
98         newPoints->InsertNextId(cPoints->GetId(1));
99         newPoints->InsertNextId(cPoints->GetId(2));
100        newPoints->InsertNextId(cPoints->GetId(0)+originalNumberOfPoints);
101        newPoints->InsertNextId(cPoints->GetId(1)+originalNumberOfPoints);
102        newPoints->InsertNextId(cPoints->GetId(2)+originalNumberOfPoints);
103
104        // insert a new wedge cell, defined by the points
105        tempGrid->InsertNextCell(13, newPoints);
106
107

```

```

102     }
103
104     // put the data into the classes extruded volume.
105     m_extrudedVolume = tempGrid;
106 }
107
108 void CompareSurfaces::GetSurfaceCentroid(vtkSmartPointer<vtkPolyData> surface, double centroid[3])
109 {
110     double ptx = 0;
111     double pty = 0;
112     double ptz = 0;
113     for (vtkIdType i = 0; i < surface->GetPoints()->GetNumberOfPoints(); ++i)
114     {
115         double pt[3];
116         surface->GetPoints()->GetPoint(i, pt);
117         ptx += pt[0];
118         pty += pt[1];
119         ptz += pt[2];
120     }
121     centroid[0] = ptx/surface->GetPoints()->GetNumberOfPoints();
122     centroid[1] = pty/surface->GetPoints()->GetNumberOfPoints();
123     centroid[2] = ptz/surface->GetPoints()->GetNumberOfPoints();
124 }
125
126 vtkSmartPointer<vtkPolyData> CompareSurfaces::ProbeVolume(vtkSmartPointer<vtkUnstructuredGrid>
127   → volume, vtkSmartPointer<vtkPolyData> surface)
128 {
129     vtkSmartPointer<vtkPolyData> outputSurface = vtkSmartPointer<vtkPolyData>::New();
130     outputSurface->DeepCopy(surface);
131     unsigned int numberOfArrays = outputSurface->GetPointData()->GetNumberOfArrays();
132     for (unsigned int i = 0; i < numberOfArrays; ++i)
133     {
134         outputSurface->GetPointData()->RemoveArray(i);
135     }
136
137     // create a new array to hold the data
138     vtkSmartPointer<vtkDoubleArray> newArray = vtkSmartPointer<vtkDoubleArray>::New();
139     newArray->SetNumberOfComponents(1);
140     newArray->SetNumberOfTuples(outputSurface->GetNumberOfPoints());
141     newArray->SetName("Extracted Data");
142     outputSurface->GetPointData()->AddArray(newArray);
143
144     // create a cell locator to aid in finding the cells that points belong to
145     vtkSmartPointer<vtkCellLocator> cellLocator =
146     vtkSmartPointer<vtkCellLocator>::New();
147     cellLocator->SetDataSet(volume);
148     cellLocator->BuildLocator();
149
150     // these will be used in the loop to hold data
151     vtkSmartPointer<vtkWedge> cCell;
152     double blankTuple = -1000000.;
153     double cPoint[3];
154     double closestPoint[3];
155     int subId;
156     double pcoords[3];
157     double dist2;
158     double weights[6];
159     double cellData[6];
160     double cValue;
161     vtkIdType cCellNo;
162     vtkSmartPointer<vtkIdList> cellPoints;
163
164     // loop through each point in the input surface
165     for (unsigned int i = 0; i < outputSurface->GetNumberOfPoints(); i++)
166     {
167         // get the point location
168         outputSurface->GetPoint(i, cPoint);
169         // find the cell that contains the point

```

```

169     cCellNo = cellLocator->FindCell(cPoint);
170     if (cCellNo == -1) // if the point is outside of cells , enter a value of zero in the
171         → data array
172     {
173         newArray->SetTuple(i,&blankTuple);
174         continue;
175     }
176     if (volume->GetCellType(cCellNo) != 13) // if the cell isn't a wedge, skip it.
177     {
178         continue;
179     }
180     // get the point IDs that define the containing cell
181     cellPoints = volume->GetCell(cCellNo)->GetPointIds();
182     // get the strain values for each point in the containing cell
183     cellData[0] = *volume->GetPointData()->GetArray(0)->GetTuple(cellPoints->GetId(0));
184     cellData[1] = *volume->GetPointData()->GetArray(0)->GetTuple(cellPoints->GetId(1));
185     cellData[2] = *volume->GetPointData()->GetArray(0)->GetTuple(cellPoints->GetId(2));
186     cellData[3] = *volume->GetPointData()->GetArray(0)->GetTuple(cellPoints->GetId(3));
187     cellData[4] = *volume->GetPointData()->GetArray(0)->GetTuple(cellPoints->GetId(4));
188     cellData[5] = *volume->GetPointData()->GetArray(0)->GetTuple(cellPoints->GetId(5));
189
190     // use the EvaluatePosition method of the cell to get the interpolation function weight
191         → values. The rest of the data is not used
192     volume->GetCell(cCellNo)->EvaluatePosition(cPoint,closestPoint,subId,pcoords,dist2,
193         → weights);
194     // calculate the value at the point by multiplying each weight by the data
195     cValue = cellData[0]*weights[0] + cellData[1]*weights[1] + cellData[2]*weights[2] +
196         → cellData[3]*weights[3] + cellData[4]*weights[4] + cellData[5]*weights[5];
197     // set the data.
198     newArray->SetTuple(i,&cValue);
199
200 }
201 return outputSurface;
202 }
203
204 vtkSmartPointer<vtkPolyData> CompareSurfaces::AlignSurfaces(vtkSmartPointer<vtkPolyData>
205     → recieverSurf, vtkSmartPointer<vtkPolyData> donorSurf)
206 {
207     // create the icp transform
208     vtkSmartPointer<vtkIterativeClosestPointTransform> icp = vtkSmartPointer<
209         → vtkIterativeClosestPointTransform >::New();
210     // use the output of icp as the final transform.
211     vtkSmartPointer<vtkTransformPolyDataFilter> finalTransform = vtkSmartPointer<
212         → vtkTransformPolyDataFilter >::New();
213
214     // if an initial transfor was given, use it
215     if (m_translate[0]!=0 || m_translate[1]!=0 || m_translate[2]!=0
216         || m_rotate[0]!=0 || m_rotate[1]!=0 || m_rotate[2]!=0)
217     {
218         std::cout<<"Setting the initial transform"<<std::endl;
219         // create a general transform for the initial transform
220         vtkSmartPointer<vtkTransform> initialTranRot = vtkSmartPointer<vtkTransform >::New();
221         initialTranRot->Translate(m_translate);
222         initialTranRot->RotateX(m_rotate[0]);
223         initialTranRot->RotateY(m_rotate[1]);
224         initialTranRot->RotateZ(m_rotate[2]);
225
226         // transform recieverSurf using the rough transform
227         vtkSmartPointer<vtkTransformPolyDataFilter> initialTransform = vtkSmartPointer<
228             → vtkTransformPolyDataFilter >::New();
229         initialTransform->SetInput(recieverSurf);
230         initialTransform->SetTransform(initialTranRot);
231         initialTransform->Update();
232
233         // set the output of the initial tranform to be the input of the ICP
234         icp->SetSource(initialTransform->GetOutput());
235
236         // set the output of the initial transform to be the input to the final transform

```

```

229         finalTransform->SetInput(initialTransform->GetOutput());
230     }
231     else
232     {
233         std::cout<<"Matching centroids"<<std::endl;
234         // if no initial transform was set start by matching centroids
235         icp->StartByMatchingCentroidsOn();
236         // make the receiver surface the source
237         icp->SetSource(receiverSurf);
238         // set the final transform input to be the receiver surface
239         finalTransform->SetInput(receiverSurf);
240     }
241
242     // use the output of the of the rough transform as the input to the fine icp calculation
243     icp->SetTarget(donorSurf);
244     icp->GetLandmarkTransform()->SetModeToRigidBody();
245     icp->Modified();
246     icp->Update();
247
248     // perform the final transform
249     finalTransform->SetTransform(icp);
250     finalTransform->Update();
251
252     // return the output from the final transfrom
253     return finalTransform->GetOutput();
254 }
255
256 void CompareSurfaces::CompileData( vtkSmartPointer<vtkPolyData> receiverSurf, vtkSmartPointer<
257   ↗ vtkPolyData> donorSurf)
258 {
259     // copy the structure of the receiver surface as this is the of the compiled surface
260     vtkSmartPointer<vtkPolyData> tempSurface = vtkSmartPointer<vtkPolyData>::New();
261     tempSurface->CopyStructure(receiverSurf);
262
263     // create a new data array for the drop tower strain
264     vtkSmartPointer<vtkDoubleArray> receiverData = vtkSmartPointer<vtkDoubleArray>::New();
265     receiverData->DeepCopy(receiverSurf->GetPointData()->GetArray(0));
266     receiverData->SetName(m.receiverName.c_str());
267
268     // create a new data array for the instron strain
269     vtkSmartPointer<vtkDoubleArray> donorData = vtkSmartPointer<vtkDoubleArray>::New();
270     donorData->DeepCopy(donorSurf->GetPointData()->GetArray(0));
271     donorData->SetName(m.donorName.c_str());
272
273     // create a new data array for the difference between them
274     vtkSmartPointer<vtkDoubleArray> diff = vtkSmartPointer<vtkDoubleArray>::New();
275     diff->SetNumberOfTuples(tempSurface->GetNumberOfPoints());
276     diff->SetNumberOfComponents(1);
277     diff->SetName("delta");
278
279     // iterate through the points in the compiled surface and fill in the data arrays.
280     for (unsigned int i = 0; i<tempSurface->GetNumberOfPoints();++i)
281     {
282         double* cDonor = donorData->GetTuple(i);
283         double* cReceiver = receiverData->GetTuple(i);
284         double cDiff;
285         // in the ProbvVolume method, -1000000 was used to indicate a point with no data. Carry
286         // → that though.
287         if (*cReceiver == -1000000)
288         {
289             cDiff = -1000000;
290         }
291         else
292         {
293             cDiff = *cDonor-*cReceiver;
294         }
295         diff->SetTuple(i ,&cDiff);
296     }
}

```

```

295 // add the arrays to the point data
296 tempSurface->GetPointData()->AddArray(recieverData);
297 tempSurface->GetPointData()->AddArray(donorData);
298 tempSurface->GetPointData()->AddArray(diff);
299
300 // threshold the temporary surface to remove data where there was no overlap
301 vtkSmartPointer<vtkThreshold> threshold = vtkSmartPointer<vtkThreshold>::New();
302 threshold->SetInput(tempSurface);
303 threshold->SetInputArrayToProcess(0,0,0,
304                                     vtkDataObject::FIELD_ASSOCIATION_POINTS,
305                                     "delta");
306 threshold->ThresholdByUpper(-999990);
307 threshold->Update();
308
309 // return the compiled surface
310 m_compiledSurf = threshold->GetOutput();
311
312 }
313
314 void CompareSurfaces::WriteDataToFile(std::string fileName)
315 {
316     // open the file for writing
317     std::ofstream outFile;
318     outFile.open(fileName.c_str(), std::ios::trunc);
319     if (!outFile.is_open()) // if it fails to open, exit
320     {
321         std::cerr << "Error opening output file: " << fileName << "\nPlease check the name and try
322             again." << std::endl;
323         return;
324     }
325     // write the header line
326     outFile << "Reviewer (Moving) File Name: " << m_recieverReader->GetFileName() << std::endl;
327     outFile << "Donor (Fixed) File Name:" << m_donorReader->GetFileName() << std::endl;
328     outFile << "Initial Transform. Translate (" << m_translate[0] << "," << m_translate[1] << ","
329             << m_translate[2] << "). Rotate (" <<
330             m_rotate[0] << "," << m_rotate[1] << "," << m_rotate[2] << ")" << std::endl;
331     outFile << "Point," << m_compiledSurf->GetPointData()->GetArray(0)->GetName() << ","
332             << m_compiledSurf->GetPointData()->GetArray(1)->GetName() << ",Diff ,x,y,z" << std::endl;
333
334     // write the rest of the file
335     double aStrain;
336     double bStrain;
337     double diff;
338     double loc[3];
339     for (int i = 0; i < m_compiledSurf->GetNumberOfPoints(); ++i)
340     {
341         m_compiledSurf->GetPointData()->GetArray(0)->GetTuple(i,&aStrain);
342         m_compiledSurf->GetPointData()->GetArray(1)->GetTuple(i,&bStrain);
343         m_compiledSurf->GetPointData()->GetArray(2)->GetTuple(i,&diff);
344         m_compiledSurf->GetPoint(i,loc);
345
346         outFile << i << "," << aStrain << "," << bStrain << "," << diff << ","
347             << loc[0] << "," << loc[1] << "," << loc
348             << [2] << std::endl;
349     }
350
351     outFile.close();
352 }

```

E.3.4 Main program for comparison of strains stored on two surfaces

```

1 /*
2 * StrainCompare.cpp
3 *
4 * Copyright 2013 Seth Gilchrist <seth@sethgilchrist.com
5 *

```

```

6  * This program is free software; you can redistribute it and/or modify
7  * it under the terms of the GNU General Public License as published by
8  * the Free Software Foundation; either version 3 of the License, or
9  * (at your option) any later version.
10 *
11 * This program is distributed in the hope that it will be useful,
12 * but WITHOUT ANY WARRANTY; without even the implied warranty of
13 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14 * GNU General Public License for more details.
15 *
16 * You should have received a copy of the GNU General Public License
17 * along with this program; if not, write to the Free Software
18 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
19 * MA 02110-1301, USA.
20 *
21 *
22 */
23
24 #include <iostream>
25 #include "../lib/CompareSurfaces-InputTransform/CompareSurfaces-InputTransform.h"
26 #include <vtkSmartPointer.h>
27 #include <vtkXMLPolyDataWriter.h>
28 #include <vtkXMLUnstructuredGridWriter.h>
29
30 int main(int argc, char **argv)
31 {
32     if (argc < 4 || (argc > 4 && argc != 10))
33     {
34         std::cerr<<"### Execution ERROR ###"<<std::endl;
35         std::cerr<<"Not enough inputs. \n Usage:"<<std::endl;
36         std::cerr<<argv[0]<<" [Reciever Surface] [Donor Surface] [Output Path] [Optional Initial
37             → Transform]"<<std::endl;
38         std::cerr<<std::endl;
39         std::cerr<<"[Reciever Surface] and [Donor Surface] files must be VTK Polydata files (.vt
40             → ). The output will be"<<std::endl;
41         std::cerr<<"an VTK unstructured grid file (strainCompare.vtu) and a text file (
42             → strainCompare.txt) containing"<<std::endl;
43         std::cerr<<"the input file names, any initial transform data and the point data."<<std::
44             → endl;
45         std::cerr<<"This program assumes that the receiver will be from the drop tower and the
46             → donor from the instron."<<std::endl;
47         std::cerr<<std::endl;
48         std::cerr<<"The receiver surface will be aligned to the donor surface and the data from
49             → the donor surface."<<std::endl;
50         std::cerr<<"will be transferred to the receiver. The program writes the two output files
51             → files to [Output Path]."<<std::endl;
52         std::cerr<<"strainCompare.vtu will have three point data sets."<<std::endl;
53         std::cerr<<"Drop Tower Strain", "Instron Strain" and "diff". diff is calculated as
54             → (donor (instron) - receiver (drop tower))<<std::endl;
55         std::cerr<<std::endl;
56         std::cerr<<"The optional initial transform is to roughly align the receiver surface with
57             → the donor surface before"<<std::endl;
58         std::cerr<<"the ICP registration, and is not necessary in all cases. If no transform is
59             → given then no rough alignment is made."<<std::endl;
60         std::cerr<<"The initial transform must be given as six values the order:"<<std::endl;
61         std::cerr<<"[Translate x] [Translate y] [Translate z] [Rotate x] [Rotate y] [Rotate z]"<<
62             → std::endl;
63         std::cerr<<"These values can be obtained by manipulating the surfaces in ParaView."<<std::
64             → endl;
65         std::cerr<<std::endl<<"### ABORTED ###"<<std::endl;
66         return EXIT_FAILURE;
67     }
68     CompareSurfaces* compare = new CompareSurfaces;
69     if (argc == 10)
70     {
71         double translate[3] = {atof(argv[4]), atof(argv[5]), atof(argv[6])};
72         double rotate[3] = {atof(argv[7]), atof(argv[8]), atof(argv[9])};
73         compare->SetInitialTransform(translate, rotate);

```



```
122         return 0;  
123 }
```

CMakeLists.txt file:

```
1 cmake_minimum_required(VERSION 2.6)  
2 project( StrainCompare-InputTransform )  
3 FIND_PACKAGE(VTK)  
4  
5 IF (VTK_FOUND)  
6   INCLUDE(${VTK_USE_FILE})  
7 ELSE (VTK_FOUND)  
8   MESSAGE(FATAL_ERROR  
9     "VTK not found. Please set VTK_DIR.")  
10 ENDIF(VTK_FOUND)  
11  
12 ADD_LIBRARY( CompareSurfaces-InputTransform .../lib/CompareSurfaces-InputTransform/CompareSurfaces  
13   ↪ -InputTransform.cpp)  
13 ADD_EXECUTABLE( StrainCompare-InputTransform StrainCompare-InputTransform.cpp )  
14 TARGET_LINK_LIBRARIES( StrainCompare-InputTransform CompareSurfaces-InputTransform ${  
15   ↪ ITK_LIBRARIES} vtkHybrid )
```

E.4 Digital volume correlation

The digital volume correlation (DVC) algorithm is written in C++ and used to conduct DVC calculations on 3D image datasets. The code is available for download at <https://github.com/sethgilchrist/DigitalVolumeCorrelation>.

E.5 Digital volume correlation verification

The digital volume correlation (DVC) code to verify the output of the algorithm is available for download at <https://github.com/sethgilchrist/SyntheticStrainImage>. The code applies a deformation field to a digital image, which creates strain for analysis using the DVC algorithm. There are two algorithms included, the first applies the strain to an image to create an analysis image for the DVC. The second takes the mesh output from the analysis and interpolates the true strain value at each point in the mesh, creating a second mesh containing the output, the ground truth and comparisons.